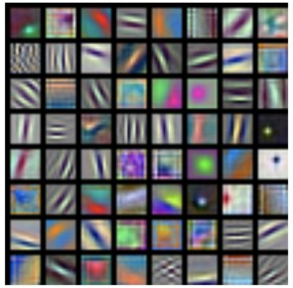
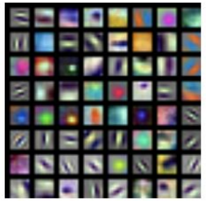
Visualizing and Understanding Neural Networks

TJ Machine Learning

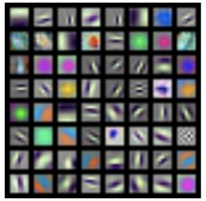
Visualizing First Layer Filters

- Filters of first layer will have 3 channels, so we can just visualize them using RGB-color scheme
- Since activations of the layer are inner products of filter and image, highest activations will be the same same shape as the filters
- Not easy to do with filters in later layers because of differing dimensions


AlexNet:
64 x 3 x 11 x 11


ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

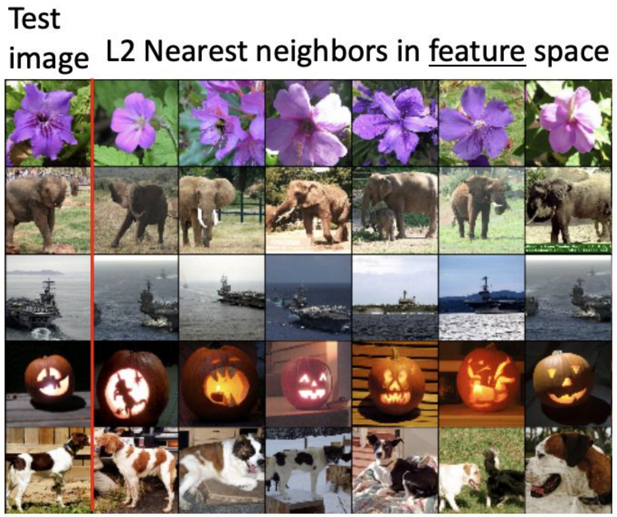

DenseNet-121:
64 x 3 x 7 x 7

L2 Nearest Neighbors of Fully Connected Layers

- Find L2 Nearest Neighbors of activations of fully connected layers near the end of a network

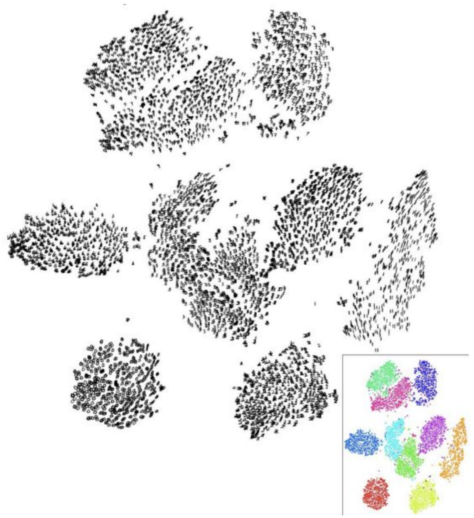
$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

From https://www.saedsayad.com/k_nearest_neighbors.htm



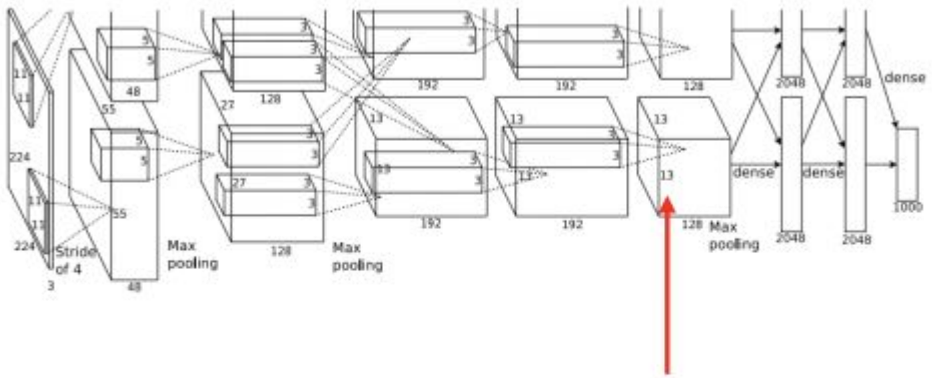
Dimensionality Reduction Algorithms for Last Layer

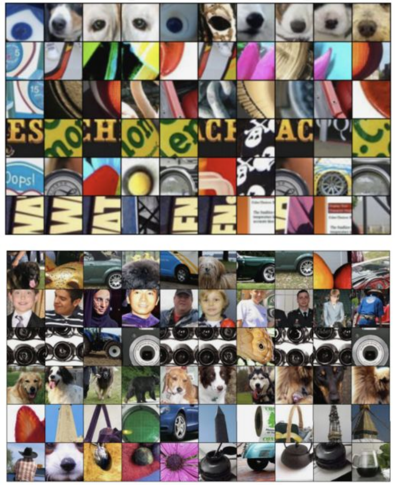
- Take activations of last fully connected layer and run through dimensionality reduction algorithm like PCA or t-SNE
- Different classes will aggregate into different clusters



Maximally Activating Patches

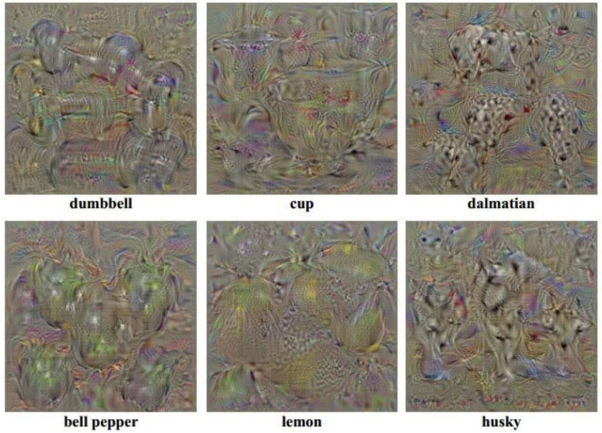
- Find what patches of image the activations of a certain activation depends on (deeper parts of network will correspond to larges patches)
- Run all images in training set and find which images lead to highest activation





Optimizing Image for Some Class

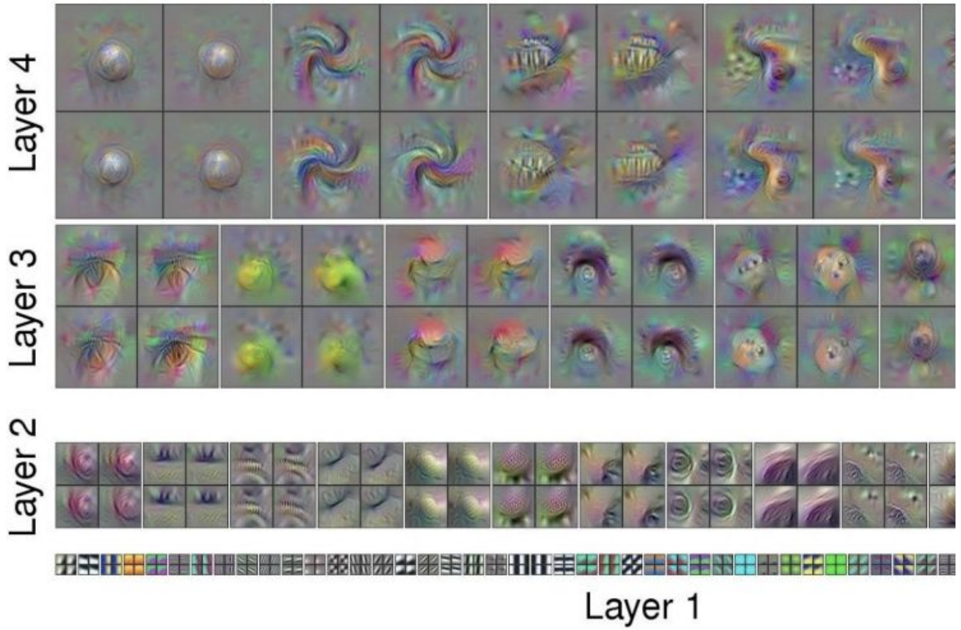
- Find image that maximizes class score (penalized with regularization term that makes image more realistic)
- Uses **gradient ascent**, not gradient descent
- Update image, not weights (network should already be trained)

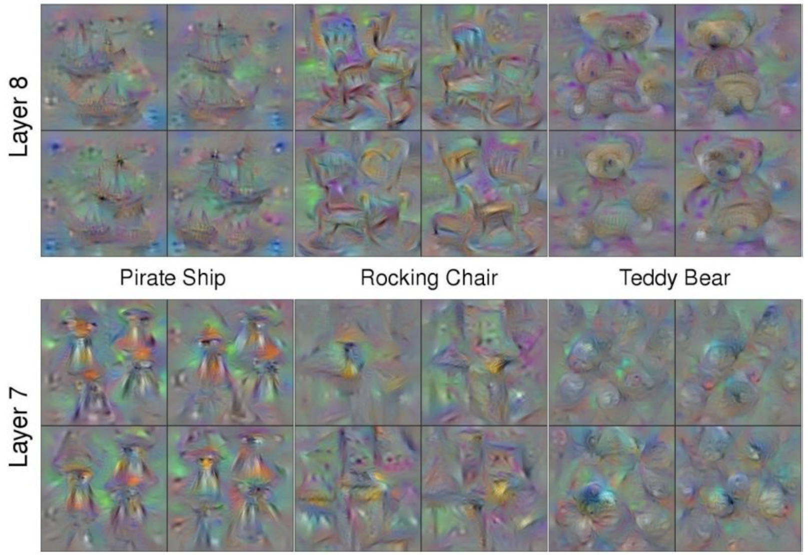


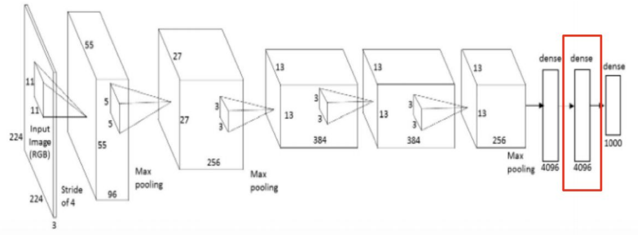
$$\boxed{\arg\max_I \boxed{S_c(I)} - \lambda \|I\|_2^2}$$

score for class c (before Softmax)

Optimizing Image for Certain Features

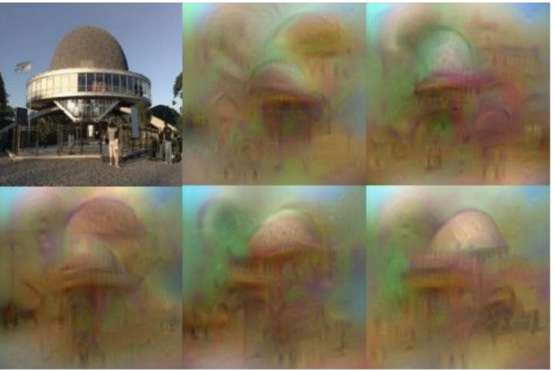






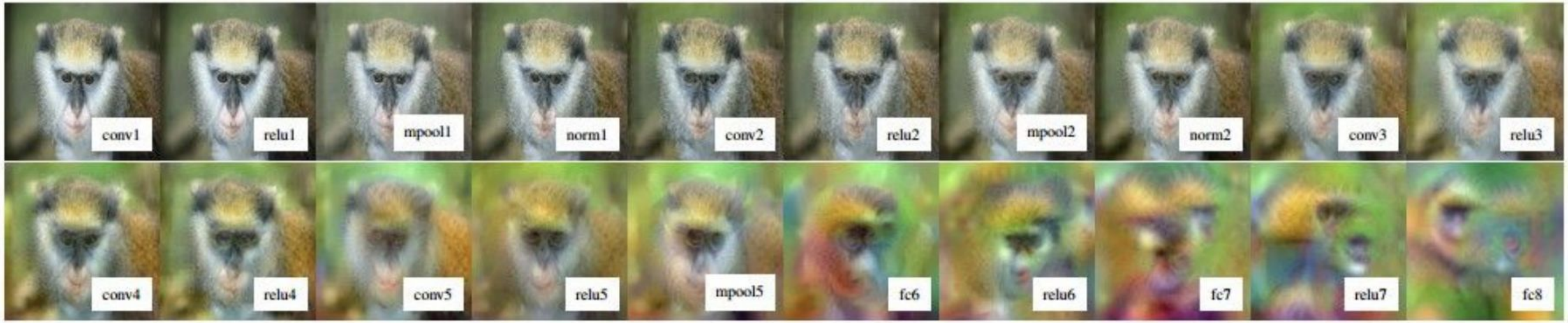
Inverting Deep Image Representations

- Start off at some layer in the ConvNet, backpropagate on random image to get the activations at that layer closer to the activations generated by the forward pass of a real image



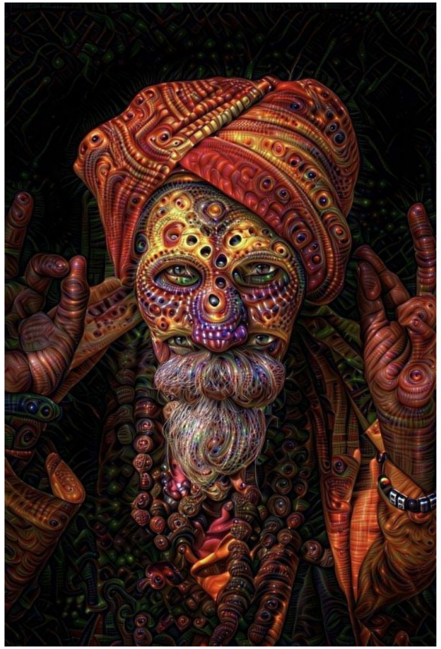
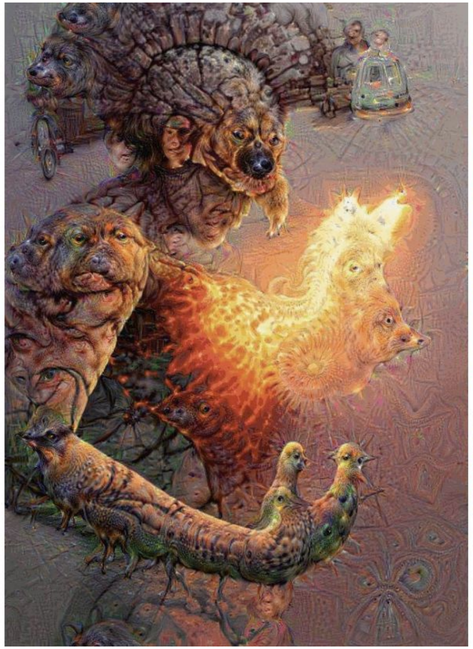
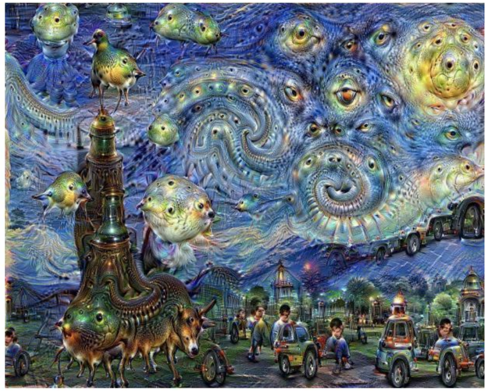
$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

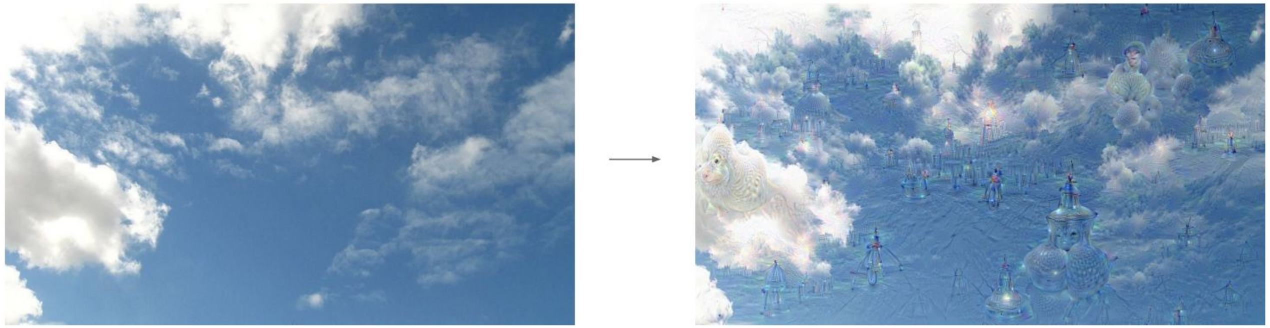


- Using activations from earlier layers leads to more accurate copies of original image
- Has implications for privacy

Google DeepDream

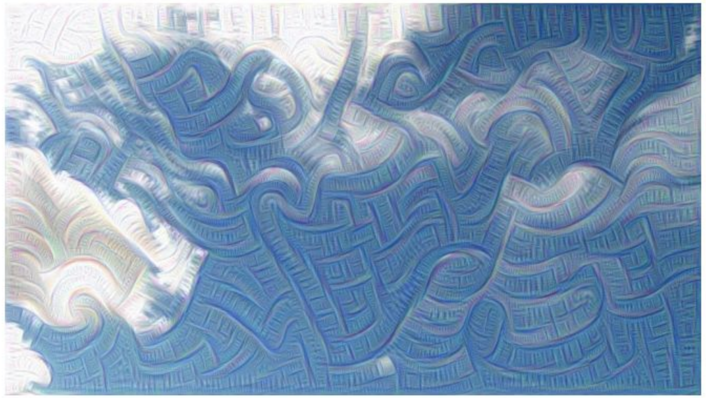




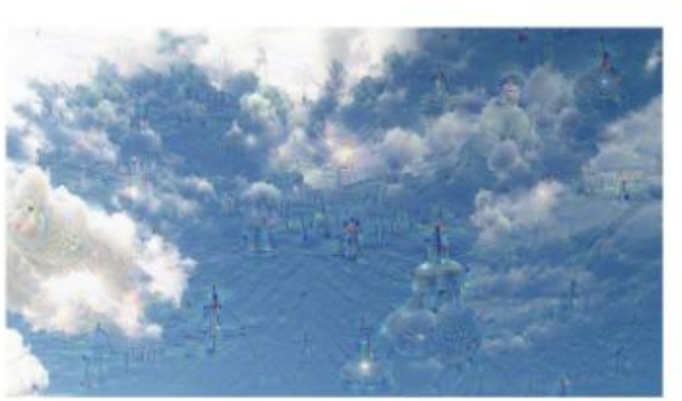
- DeepDream Algorithm:
- Choose some layer of your ConvNet to “dream” at (needs to be after ReLU layer so gradients are clipped at zero)
- Forward propagate until that layer, then set the gradients of that layer **EQUAL** to the activations and perform **gradient ascent**
    - Gradients are normally set to 1 in computation graphs. By setting layers equal to activations, we will be creating a positive feedback loop that boosts the activations by strengthening features



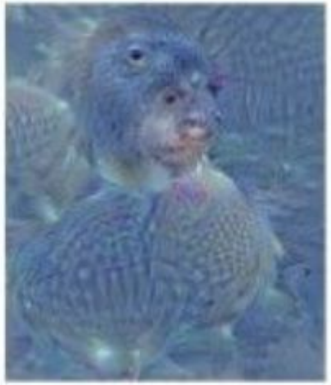
DeepDream modifies the image in a way that “boosts” all activations, at any layer

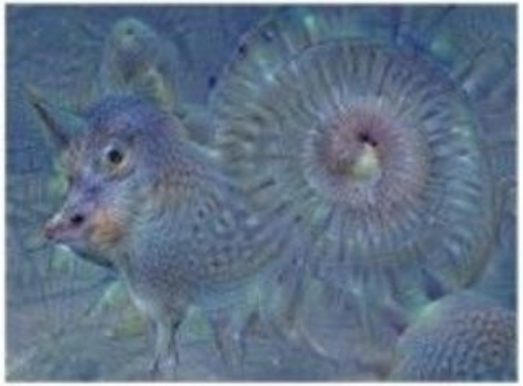
DeepDream at early vs. later layers

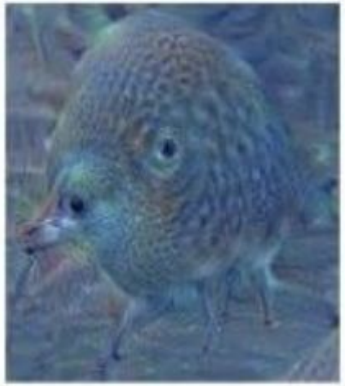


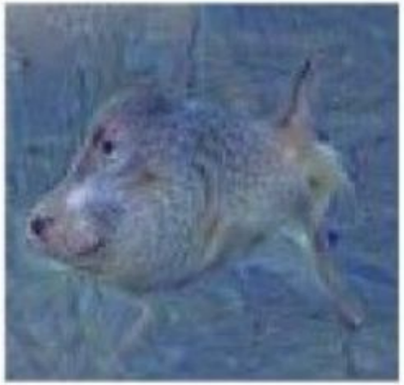


Boosts in activations can lead to...


"Admiral Dog!"


"The Pig-Snail"

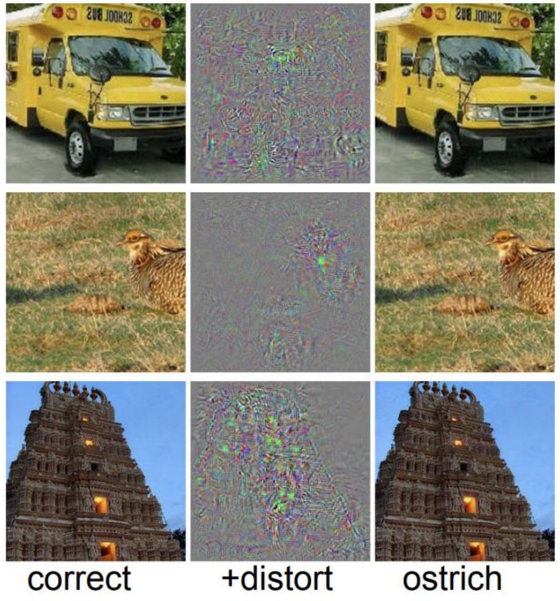

"The Camel-Bird"

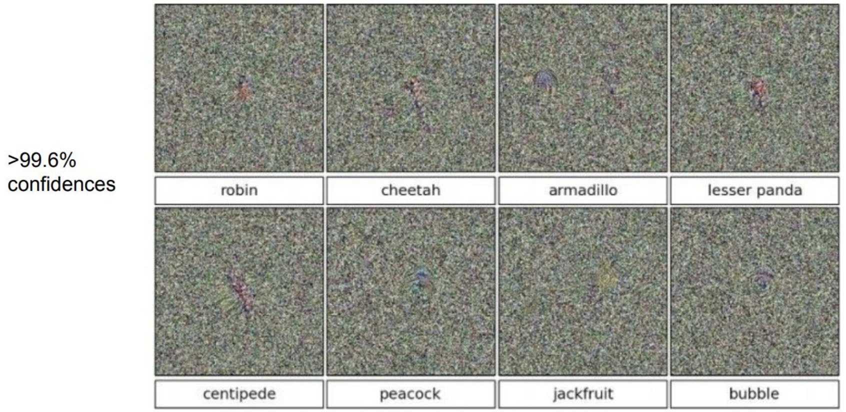

"The Dog-Fish"

From https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Adversarial Examples

- Doing backprop to optimize for some class can lead to distortions that don't appear to change image to us, but fool neural networks quite easily
- Ongoing research to try to fix this problem





All uncited images are screenshots from

- University of Michigan Deep Learning for Computer Vision lecture slides
- Stanford CS231 lecture slides