

Visualizing CNN

Samuel Cheng

(Slide credits: Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung, Geoffrey Hinton)

School of ECE
University of Oklahoma

Spring, 2018

Table of Contents

- 1 Visualizing conv-nets
- 2 CNN for arts
- 3 Fooling conv-net
- 4 Conclusions

- We talked about the basics of CNNs and several CNN architectures earlier
- How to visualize a CNN
- CNNs and arts
- Fooling a CNN

- We talked about the basics of CNNs and several CNN architectures earlier
- How to visualize a CNN
 - CNNs and arts
 - Fooling a CNN

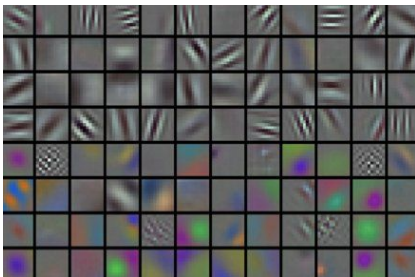
- We talked about the basics of CNNs and several CNN architectures earlier
- How to visualize a CNN
- CNNs and arts
- Fooling a CNN

- We talked about the basics of CNNs and several CNN architectures earlier
- How to visualize a CNN
- CNNs and arts
- Fooling a CNN

Visualizing and understanding conv-nets

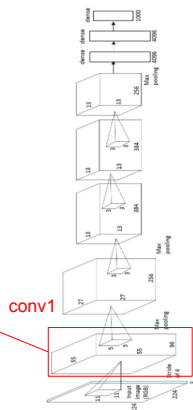
- Study weights directly
- Occlusion experiment
- Visualizing representation
 - t-SNE
 - through deconvolution
 - through optimization

Visualize the filters/kernels (raw weights)



only interpretable on the first layer :(

one-stream AlexNet



Visualize the filters/kernels (raw weights)

you can still do it for higher layers, it's just not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)

Weights:


layer 1 weights

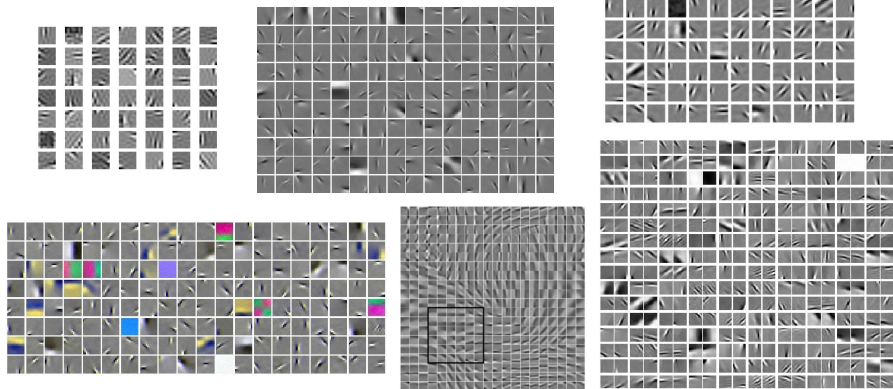
Weights:


layer 2 weights

Weights:


layer 3 weights

The gabor-like filters fatigue



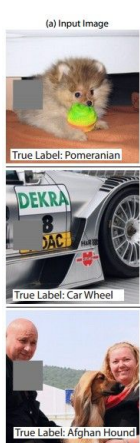
Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 10

3 Feb 2016

Occlusion experiments

[Zeiler & Fergus 2013]



(d) Classifier, probability of correct class

(as a function of the position of the square of zeros in the original image)

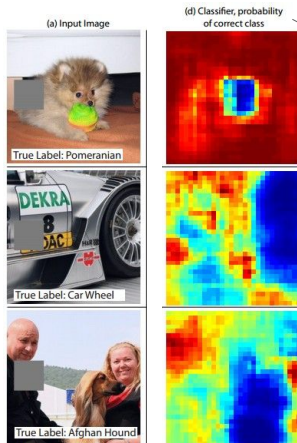
Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 14

3 Feb 2016

Occlusion experiments

[Zeiler & Fergus 2013]



(as a function of the position of the square of zeros in the original image)

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 15

3 Feb 2016

Visualizing the representation

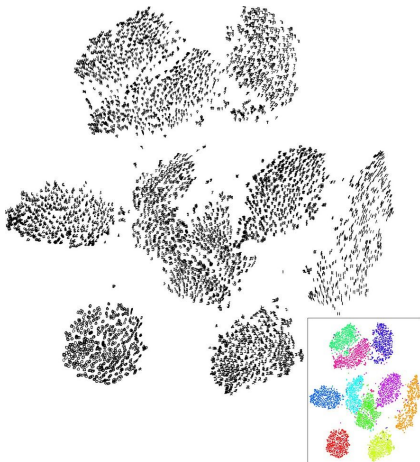
t-SNE visualization

[van der Maaten & Hinton]

Embed high-dimensional points so that locally, pairwise distances are conserved

i.e. similar things end up in similar places.
dissimilar things end up wherever

Right: Example embedding of MNIST digits (0-9) in 2D



t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{j|i} + p_{i|j} - q_{j|i} - q_{i|j})(y_i - y_j)$

t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_j^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_j^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (P_{j|i} + P_{i|j} - Q_{j|i} - Q_{i|j})(y_i - y_j)$

t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{j|i} + p_{i|j} - q_{j|i} - q_{i|j})(y_i - y_j)$

t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (P_{j|i} + P_{i|j} - Q_{j|i} - Q_{i|j})(y_i - y_j)$

t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (P_{j|i} + P_{i|j} - Q_{j|i} - Q_{i|j})(y_i - y_j)$

t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (P_{j|i} + P_{i|j} - Q_{j|i} - Q_{i|j})(y_i - y_j)$

t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (P_{j|i} + P_{i|j} - Q_{j|i} - Q_{i|j})(y_i - y_j)$

t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (P_{j|i} + P_{i|j} - Q_{j|i} - Q_{i|j})(y_i - y_j)$

t-SNE and SNE

- t-SNE is an improvement of SNE (Stochastic Neighborhood Embedding)
- SNE:
 - Match the distribution of distances between points in the original high dimensional space and the distribution of distances between points in the reduced low-dimensional space
 - x_i : location of point i in original space; y_i : location of point i in the reduced space
 - $P_{i|j} \triangleq \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$; $Q_{i|j} \triangleq \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$
 - Minimize $C \triangleq \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j P_{i|j} \log \frac{P_{i|j}}{Q_{i|j}}$. Note that KL-divergence is not symmetric
 - x_i, x_j far apart \Rightarrow small cost despite values of y_i, y_j
 - x_i, x_j close \Rightarrow small cost only if y_i, y_j close
 - $\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{j|i} + p_{i|j} - q_{j|i} - q_{i|j})(y_i - y_j)$

More on t-SNE

- SNE tends to have a “crowding problem”
- t-SNE resolved this by assuming a t-distribution rather than a Gaussian distribution for the distance between points in the reduced space

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}}$$

- Student t-distribution is much more heavy tail. Allow y_i 's to be farther away without incurring significant cost
- t-SNE also symmetrized the conditional distribution: $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$
- $\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} \underbrace{(p_{ij} - q_{ij})}_{\text{Force}} \underbrace{(1 + \|y_i - y_j\|^2)^{-1} (y_i - y_j)}_{\text{spring}}$

More on t-SNE

- SNE tends to have a “crowding problem”
- t-SNE resolved this by assuming a t-distribution rather than a Gaussian distribution for the distance between points in the reduced space

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}}$$

- Student t-distribution is much more heavy tail. Allow y_i 's to be farther away without incurring significant cost
- t-SNE also symmetrized the conditional distribution: $p_{ij} = \frac{P_{i|j} + P_{j|i}}{2}$
- $\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} \underbrace{(p_{ij} - q_{ij})}_{\text{Force}} \underbrace{(1 + \|y_i - y_j\|^2)^{-1} (y_i - y_j)}_{\text{spring}}$

More on t-SNE

- SNE tends to have a “crowding problem”
- t-SNE resolved this by assuming a t-distribution rather than a Gaussian distribution for the distance between points in the reduced space

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}}$$

- Student t-distribution is much more heavy tail. Allow y_i 's to be farther away without incurring significant cost
- t-SNE also symmetrized the conditional distribution: $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$
- $\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} \underbrace{(p_{ij} - q_{ij})}_{\text{Force}} \underbrace{(1 + \|y_i - y_j\|^2)^{-1} (y_i - y_j)}_{\text{spring}}$

More on t-SNE

- SNE tends to have a “crowding problem”
- t-SNE resolved this by assuming a t-distribution rather than a Gaussian distribution for the distance between points in the reduced space

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}}$$

- Student t-distribution is much more heavy tail. Allow y_i 's to be farther away without incurring significant cost
- t-SNE also symmetrized the conditional distribution: $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$
- $\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} \underbrace{(p_{ij} - q_{ij})}_{\text{Force}} \underbrace{(1 + \|y_i - y_j\|^2)^{-1} (y_i - y_j)}_{\text{spring}}$

More on t-SNE

- SNE tends to have a “crowding problem”
- t-SNE resolved this by assuming a t-distribution rather than a Gaussian distribution for the distance between points in the reduced space

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}}$$

- Student t-distribution is much more heavy tail. Allow y_i 's to be farther away without incurring significant cost
- t-SNE also symmetrized the conditional distribution: $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$
- $\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} \underbrace{(p_{ij} - q_{ij})}_{\text{Force}} \underbrace{(1 + \|y_i - y_j\|^2)^{-1} (y_i - y_j)}_{\text{spring}}$

More on t-SNE

- For each update, essentially summing up force exerting on a point from all other points
 - For large dataset (with say more than 10K data points), the naive implementation can be too slow
- For far away points from a similar direction, the force can be approximated as a net force from the center of mass from the point cloud
 - This is known as Barnes-Hut approximation
 - Originally introduced from astro-physics
- Can further speed things up by first putting y_i 's in a quad-tree structure
 - Can quickly determine if a point cloud is sufficiently far away from y_i for Barnes-Hut approximation
 - Allow one to pull out the center of mass of a point cloud quickly
- Also check out “How to use t-SNE effectively” for more details

More on t-SNE

- For each update, essentially summing up force exerting on a point from all other points
 - For large dataset (with say more than 10K data points), the naive implementation can be too slow
- For far away points from a similar direction, the force can be approximated as a net force from the center of mass from the point cloud
 - This is known as Barnes-Hut approximation
 - Originally introduced from astro-physics
- Can further speed things up by first putting y_i 's in a quad-tree structure
 - Can quickly determine if a point cloud is sufficiently far away from y_i for Barnes-Hut approximation
 - Allow one to pull out the center of mass of a point cloud quickly
- Also check out “How to use t-SNE effectively” for more details

More on t-SNE

- For each update, essentially summing up force exerting on a point from all other points
 - For large dataset (with say more than 10K data points), the naive implementation can be too slow
- For far away points from a similar direction, the force can be approximated as a net force from the center of mass from the point cloud
 - This is known as Barnes-Hut approximation
 - Originally introduced from astro-physics
- Can further speed things up by first putting y_i 's in a quad-tree structure
 - Can quickly determine if a point cloud is sufficiently far away from y_i for Barnes-Hut approximation
 - Allow one to pull out the center of mass of a point cloud quickly
- Also check out “How to use t-SNE effectively” for more details

More on t-SNE

- For each update, essentially summing up force exerting on a point from all other points
 - For large dataset (with say more than 10K data points), the naive implementation can be too slow
- For far away points from a similar direction, the force can be approximated as a net force from the center of mass from the point cloud
 - This is known as Barnes-Hut approximation
 - Originally introduced from astro-physics
- Can further speed things up by first putting y_i 's in a quad-tree structure
 - Can quickly determine if a point cloud is sufficiently far away from y_i for Barnes-Hut approximation
 - Allow one to pull out the center of mass of a point cloud quickly
- Also check out “How to use t-SNE effectively” for more details

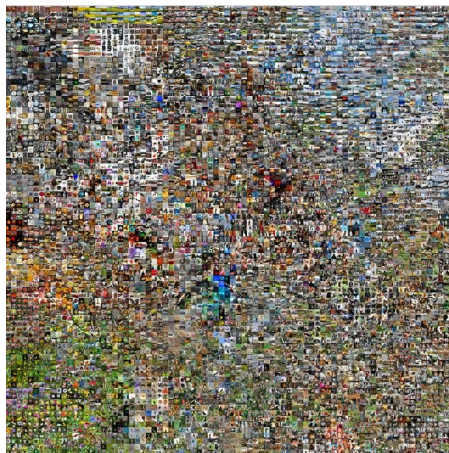
More on t-SNE

- For each update, essentially summing up force exerting on a point from all other points
 - For large dataset (with say more than 10K data points), the naive implementation can be too slow
- For far away points from a similar direction, the force can be approximated as a net force from the center of mass from the point cloud
 - This is known as Barnes-Hut approximation
 - Originally introduced from astro-physics
- Can further speed things up by first putting y_i 's in a quad-tree structure
 - Can quickly determine if a point cloud is sufficiently far away from y_i for Barnes-Hut approximation
 - Allow one to pull out the center of mass of a point cloud quickly
- Also check out “How to use t-SNE effectively” for more details

t-SNE visualization:

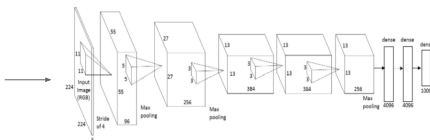
two images are placed nearby if their CNN codes are close. See more:

<http://cs.stanford.edu/people/karpathy/cnnembed/>

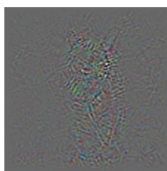


Deconv approaches

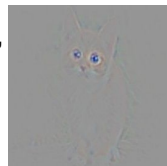
1. Feed image into net



2. Pick a layer, set the gradient there to be all zero except for one 1 for some neuron of interest
3. Backprop to image:



**“Guided
backpropagation:”
instead**



Deconv net

- Appeared in Zeiler and Fergus '13, which also discussed the occlusion experiment mentioned earlier
- Similar to backprop, but information is passed back through a “deconv net”
 - Relu maps back to Relu
 - Unpooling only modifies locations that originally “activates” the pooling operation
 - Filter maps to the transpose of the filter

Deconv net

- Appeared in Zeiler and Fergus '13, which also discussed the occlusion experiment mentioned earlier
- Similar to backprop, but information is passed back through a “deconv net”
 - Relu maps back to Relu
 - Unpooling only modifies locations that originally “activates” the pooling operation
 - Filter maps to the transpose of the filter

Deconv net

- Appeared in Zeiler and Fergus '13, which also discussed the occlusion experiment mentioned earlier
- Similar to backprop, but information is passed back through a “deconv net”
 - Relu maps back to Relu
 - Unpooling only modifies locations that originally “activates” the pooling operation
 - Filter maps to the transpose of the filter

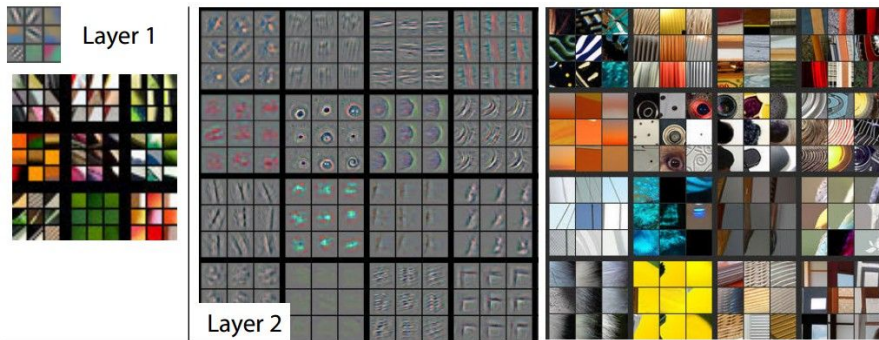
Deconv net

- Appeared in Zeiler and Fergus '13, which also discussed the occlusion experiment mentioned earlier
- Similar to backprop, but information is passed back through a “deconv net”
 - Relu maps back to Relu
 - Unpooling only modifies locations that originally “activates” the pooling operation
 - Filter maps to the transpose of the filter

Deconv net

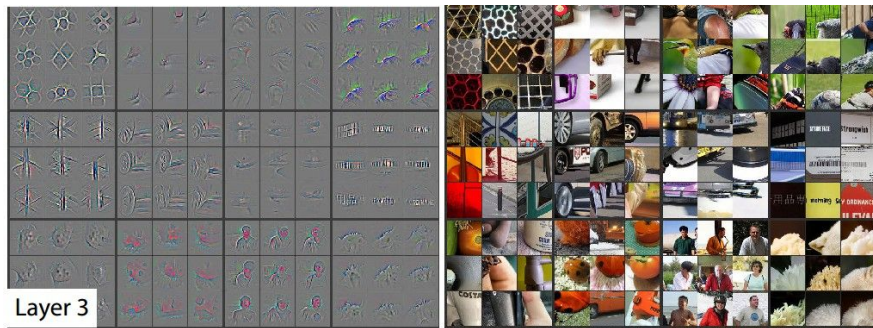
Visualizing and Understanding Convolutional Networks
 Zeiler & Fergus, 2013

Visualizing arbitrary neurons along the way to the top...



Deconv net

Visualizing arbitrary neurons along the way to the top...



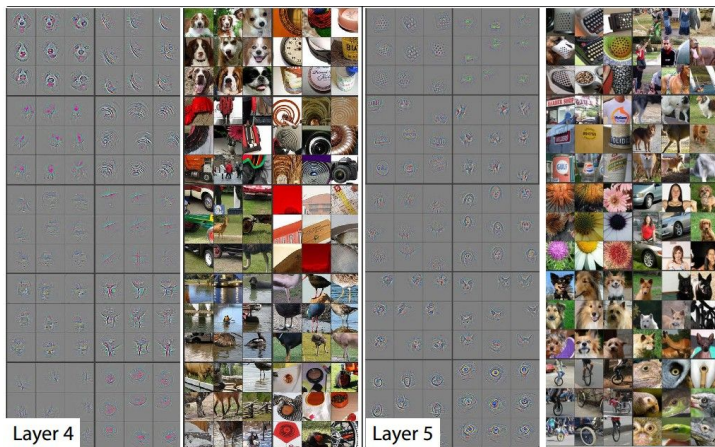
Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 27

3 Feb 2016

Deconv net

Visualizing
arbitrary
neurons along
the way to the
top...



Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 28

3 Feb 2016

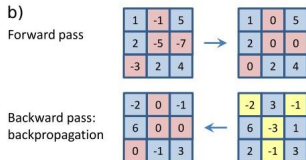
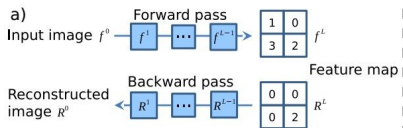
Guided backprop

Deconv approaches

[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]

[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]

[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]



c) activation: $f_i^{l+1} = \text{relu}(f_i^l) = \max(f_i^l, 0)$

backpropagation: $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$

backward 'deconvnet': $R_i^l = (R_i^{l+1} > 0) \cdot R_i^{l+1}$

guided backpropagation: $R_i^l = (f_i^l > 0) \cdot (R_i^{l+1} > 0) \cdot R_i^{l+1}$



bit weird

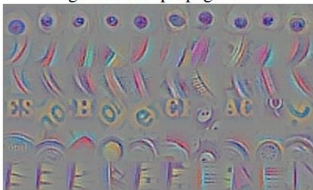
Guided backprop

Visualization of patterns learned by the layer **conv6** (top) and layer **conv9** (bottom) of the network trained on ImageNet.

Each row corresponds to one filter.

The visualization using "guided backpropagation" is based on the top 10 image patches activating this filter taken from the ImageNet dataset.

guided backpropagation



corresponding image crops



guided backpropagation



corresponding image crops



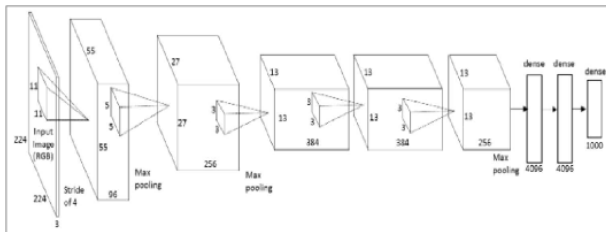
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 24

3 Feb 2016

Finding salient map of an object



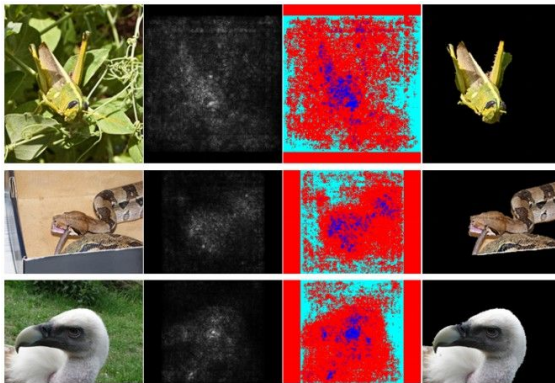
Repeat:

1. Forward an image
2. Set activations in layer of interest to all zero, except for a 1.0 for a neuron of interest
3. Backprop to image
4. Do an "image update"

Finding salient map of an object

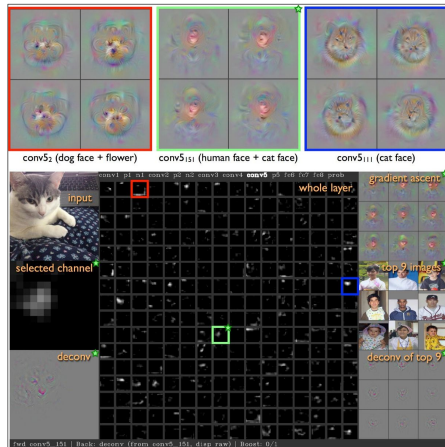
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

- Use **grabcut** for segmentation



Visualizing Activations

<http://yosinski.com/deepvis>



YouTube video

<https://www.youtube.com/watch?v=AgkFIQ4lGaM>
(4min)

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 16

3 Feb 2016

8

Patches maximally activate a neuron

Visualize patches that maximally activate neurons

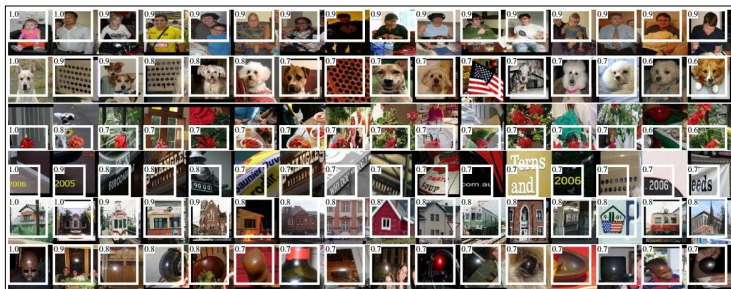
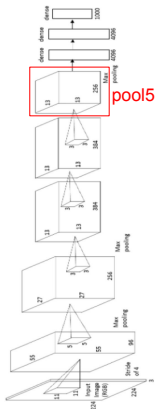


Figure 4: Top regions for six pool₅ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

Rich feature hierarchies for accurate object detection and semantic segmentation
 [Girshick, Donahue, Darrell, Malik]

one-stream AlexNet



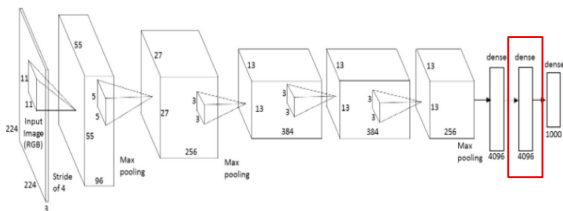
Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 7

3 Feb 2016

Recovering original image

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



Recovering original image

Find an image such that:

- Its code is similar to a given code
- It “looks natural” (image prior regularization)

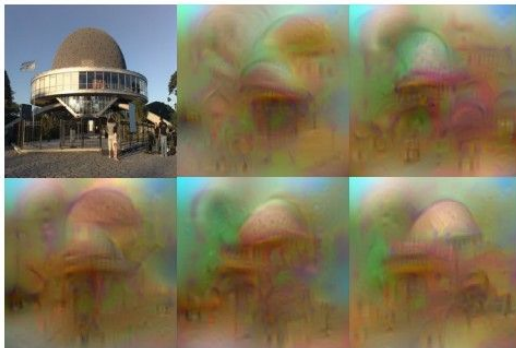
$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^{H \times W \times C}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

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

Recovering original image

Understanding Deep Image Representations by Inverting Them
[Mahendran and Vedaldi, 2014]

original image



reconstructions
from the 1000
log probabilities
for ImageNet
(ILSVRC)
classes

Recovering original image

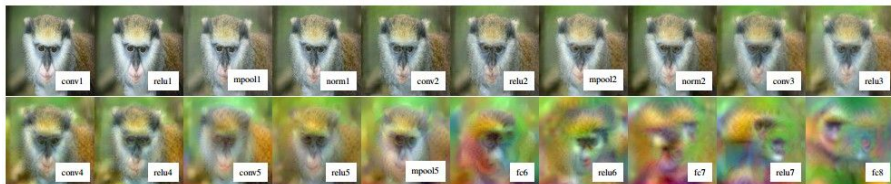
Reconstructions from the representation after last last pooling layer
(immediately before the first Fully Connected layer)



Recovering original image

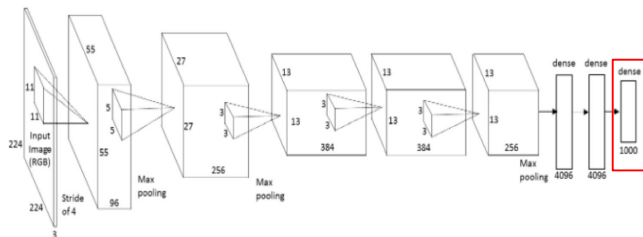


Reconstructions from intermediate layers



Class model visualization

Optimization to Image



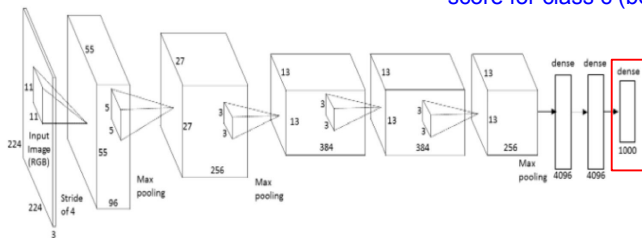
Q: can we find an image that maximizes some class score?

Class model visualization

Optimization to Image

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

score for class c (before Softmax)

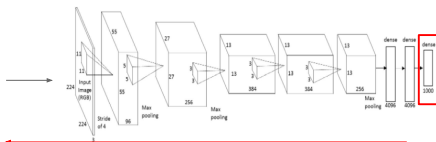
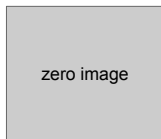


Q: can we find an image that maximizes some class score?

Class model visualization

Optimization to Image

1. feed in zeros.

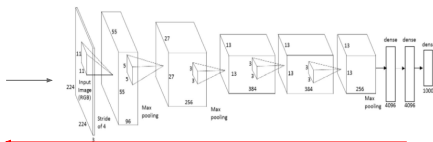


2. set the gradient of the scores vector to be $[0,0,\dots,1,\dots,0]$, then backprop to image

Class model visualization

Optimization to Image

1. feed in zeros.



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

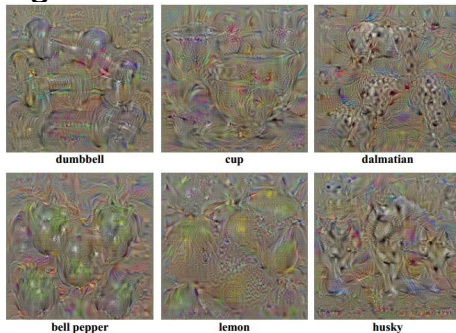
$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

score for class c (before Softmax)

Class model visualization

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

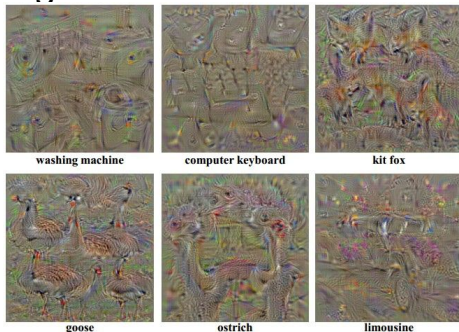
1. Find images that maximize some class score:



Class model visualization

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014


1. Find images that maximize some class score:



Class model visualization

[Understanding Neural Networks Through Deep Visualization, Yosinski et al. , 2015]

Proposed a different form of regularizing the image

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


More explicit scheme:

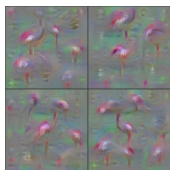
Repeat:

- Update the image \mathbf{x} with gradient from some unit of interest
- Blur \mathbf{x} a bit
- Take any pixel with small norm to zero (to encourage sparsity)

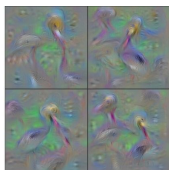
Class model visualization

[Understanding Neural Networks Through Deep Visualization, Yosinski et al. , 2015]

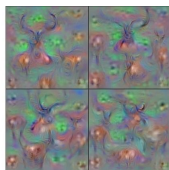
<http://yosinski.com/deepvis>



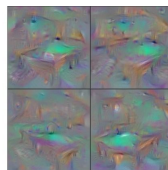
Flamingo



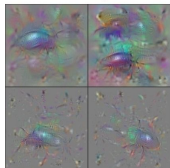
Pelican



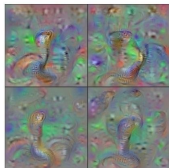
Hartebeest



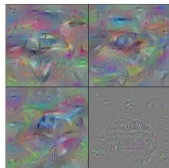
Billiard Table



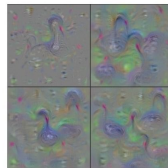
Ground Beetle



Indian Cobra

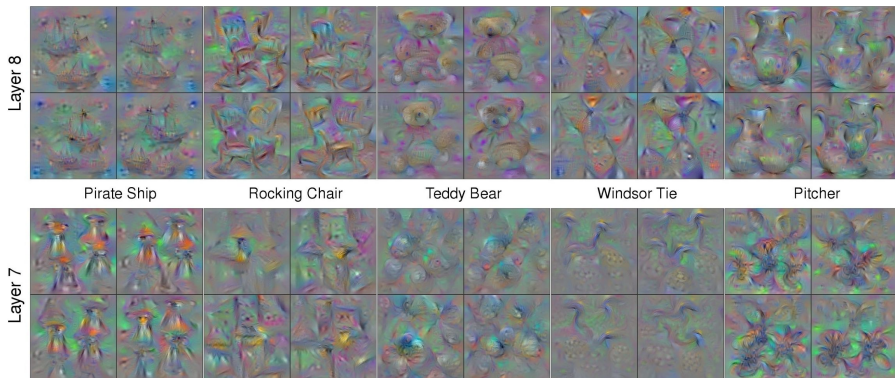


Station Wagon



Black Swan

Class model visualization

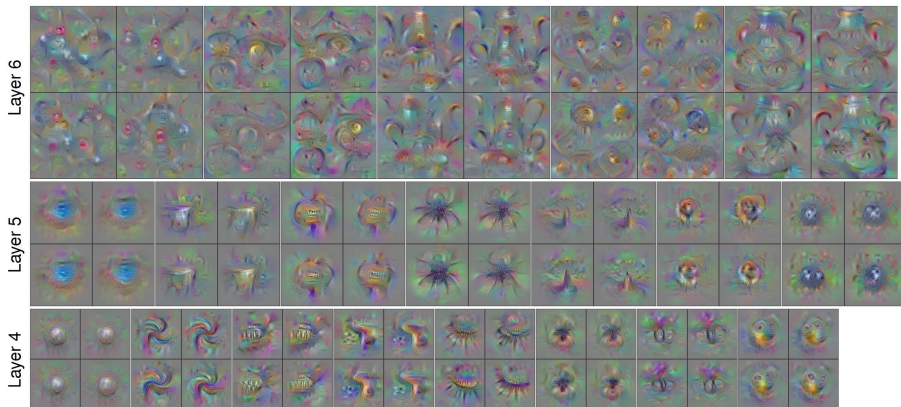


Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 41

3 Feb 2016

Class model visualization

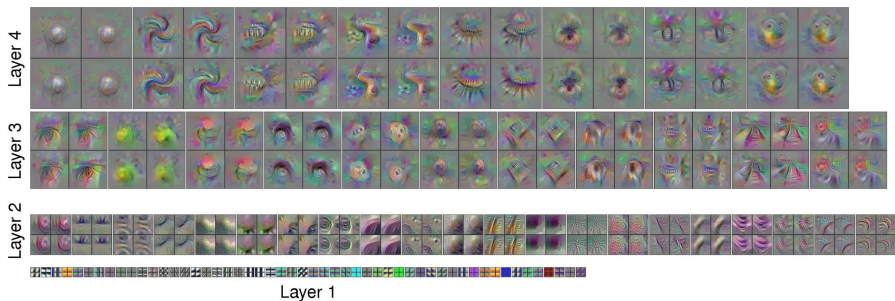


Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 42

3 Feb 2016

Class model visualization



Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 43

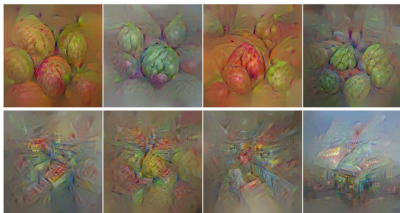
3 Feb 2016

Class model visualization

Visualizing CNN features: Gradient Ascent

Adding “multi-faceted” visualization gives even nicer results:
(Plus more careful regularization, center-bias)

Reconstructions of multiple feature types (facets) recognized
by the same “grocery store” neuron



Corresponding example training set images recognized
by the same neuron as in the “grocery store” class



Nguyen et al. “Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks”, ICML Visualization for Deep Learning Workshop 2016.
Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 31 May 10, 2017

Class model visualization

Visualizing CNN features: Gradient Ascent



Nguyen et al. "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016.
 Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.

Class model visualization

Algorithm 1 Multifaceted Feature Visualization

Input: a set of images U and a number of facets k

1. for each image in U , **compute** high-level (here fc7) hidden code Φ_i

2. **Reduce** the dimensionality of each code Φ_i from 4096 to 50 via PCA.

3. **Run** t-SNE visualization on the entire set of codes Φ_i to produce a 2-D embedding (examples in Fig. 2).

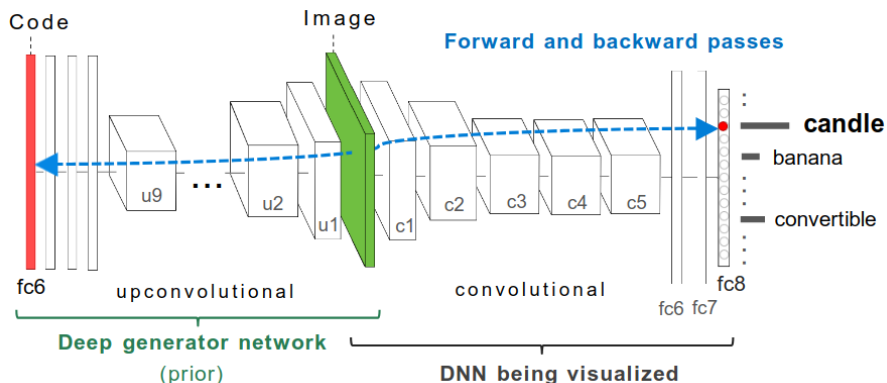
4. **Locate** k clusters in the embedding via k -means.
for each cluster

5. **Compute** a mean image \mathbf{x}_0 by averaging the 15 images nearest to the cluster centroid.

6. **Run** activation maximization (see Section 2.2), but **initialize** it with \mathbf{x}_0 instead of a random image.

Output: a set of facet visualizations $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\}$.

Class model visualization

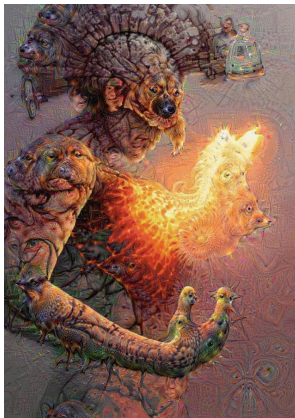
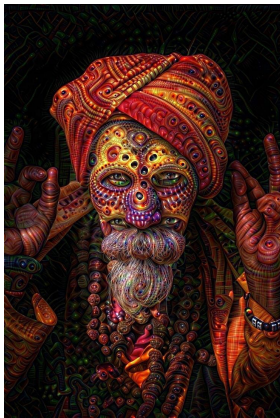


Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, Jeff Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks"

Class model visualization



Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, Jeff Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks"



DeepDream <https://github.com/google/deepdream>

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 50

3 Feb 2016

```

def objective_L2(dst):
    dst.diff[:] = dst.data

def make_step(net, step_size=1.5, end='inception_4c/output',
             jitter=32, clip=True, objective=objective_L2):
    '''Basic gradient ascent step.'''

    src = net.blobs['data'] # input image is stored in Net's 'data' blob
    dst = net.blobs[end]

    ox, oy = np.random.randint(-jitter, jitter+1, 2)
    src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift

    net.forward(end=end)
    objective(dst) # specify the optimization objective
    net.backward(start=end)
    g = src.diff[0]
    # apply normalized ascent step to the input image
    src.data[:] += step_size/np.abs(g).mean() * g

    src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image

    if clip:
        bias = net.transformer.mean['data']
        src.data[:] = np.clip(src.data, -bias, 255-bias)

```

```

def objective_L2(dst):
    dst.diff[:] = dst.data
def make_step(net, step_size=1.5, end='inception_4c/output',
             jitter=32, clip=True, objective=objective_L2):
    '''Basic gradient ascent step.'''

    src = net.blobs['data'] # input image is stored in Net's 'data' blob
    dst = net.blobs[end]

    ox, oy = np.random.randint(-jitter, jitter+1, 2)
    src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift

    net.forward(end=end)
    objective(dst) # specify the optimization objective
    net.backward(start=end)
    g = src.diff[0]
    # apply normalized ascent step to the input image
    src.data[:] += step_size/np.abs(g).mean() * g

    src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image

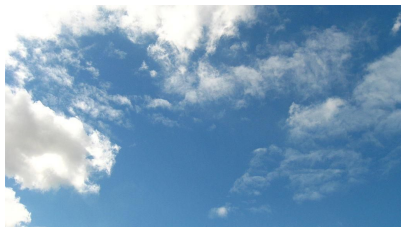
    if clip:
        bias = net.transformer.mean['data']
        src.data[:] = np.clip(src.data, -bias, 255-bias)

```

DeepDream: set dx = x :)

jitter regularizer

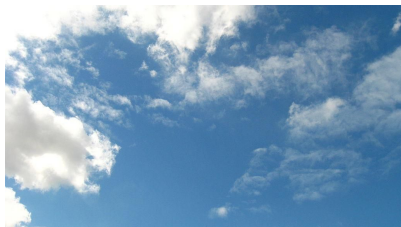
"image update"



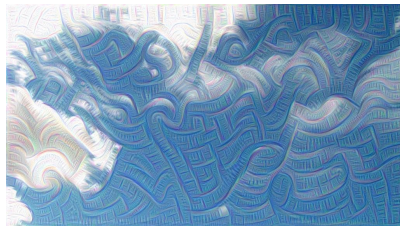
inception_4c/output

DeepDream modifies the image in a way that “boosts” all activations, at any layer
this creates a feedback loop: e.g. any slightly detected dog face will be made more
and more dog like over time





inception_3b/5x5_reduce



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



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 47 May 10, 2017

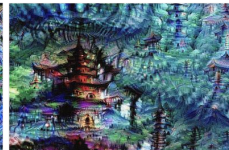
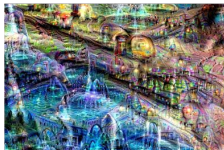
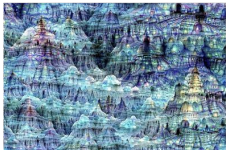
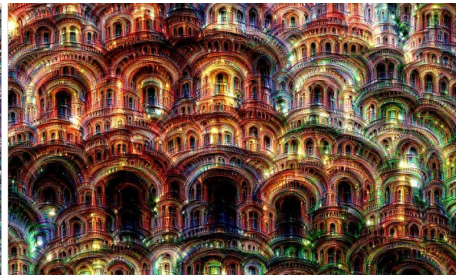
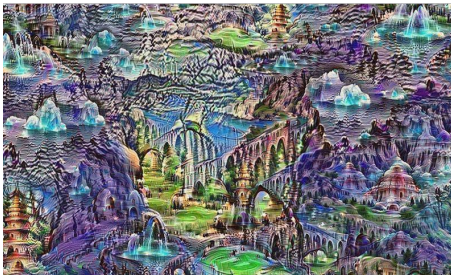


Image is licensed under [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/)

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 48 May 10, 2017

NeuralStyle

[A Neural Algorithm of Artistic Style by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, 2015]

good implementation by Justin in Torch:

<https://github.com/jcjohnson/neural-style>



Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 57

3 Feb 2016



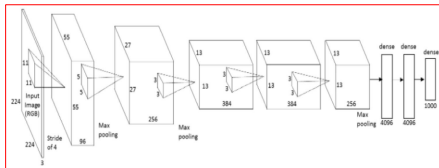
make your own easily on deepart.io

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 58

3 Feb 2016

Step 1: Extract **content targets** (ConvNet activations of all layers for the given content image)

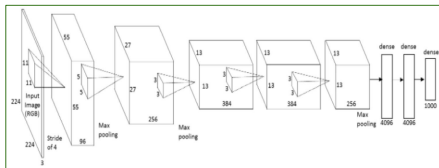


content activations

e.g.

at CONV5_1 layer we would have a [14x14x512] array of target activations

Step 2: Extract **style targets** (Gram matrices of ConvNet activations of all layers for the given style image)



style gram matrices

e.g.

at CONV1 layer (with [224x224x64] activations) would give a [64x64] Gram matrix of all pairwise activation covariances (summed across spatial locations)

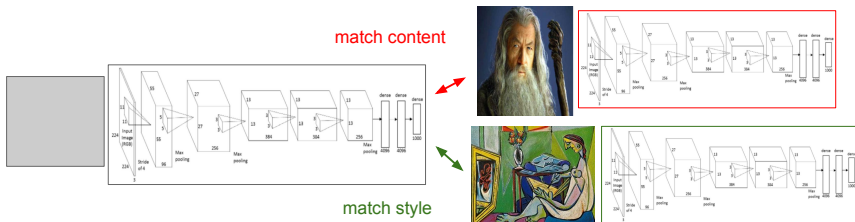
$$G = V^T V$$

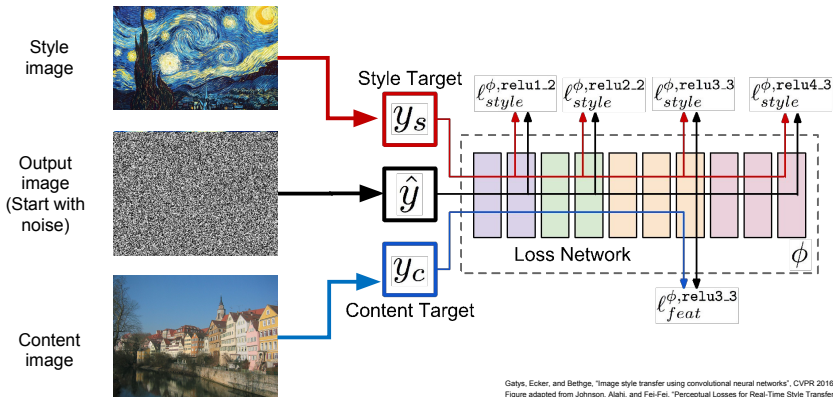
Step 3: Optimize over image to have:

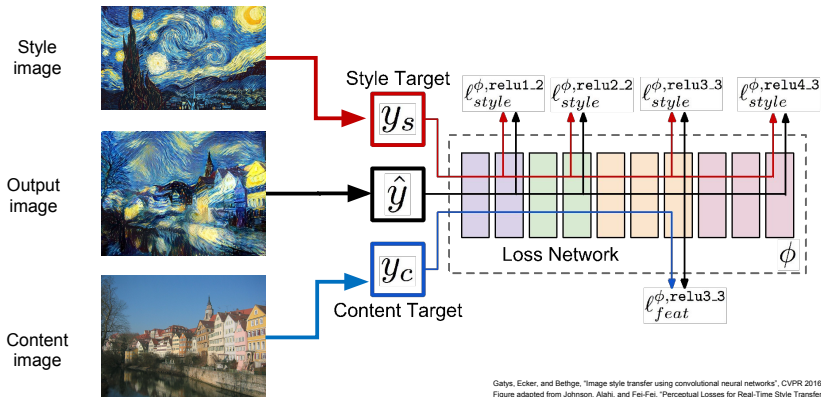
- The **content** of the content image (activations match content)
- The **style** of the style image (Gram matrices of activations match style)

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

(+Total Variation regularization (maybe))







Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 68 May 10, 2017

Neural Style Transfer

Example outputs from
[my implementation](#)
(in Torch)



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure copyright Justin Johnson, 2015.

Neural Style Transfer



More weight to
content loss



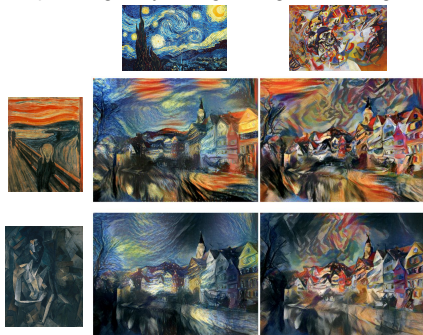
More weight to
style loss

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 70 May 10, 2017

Neural Style Transfer: Multiple Style Images

Mix style from multiple images by taking a weighted average of Gram matrices

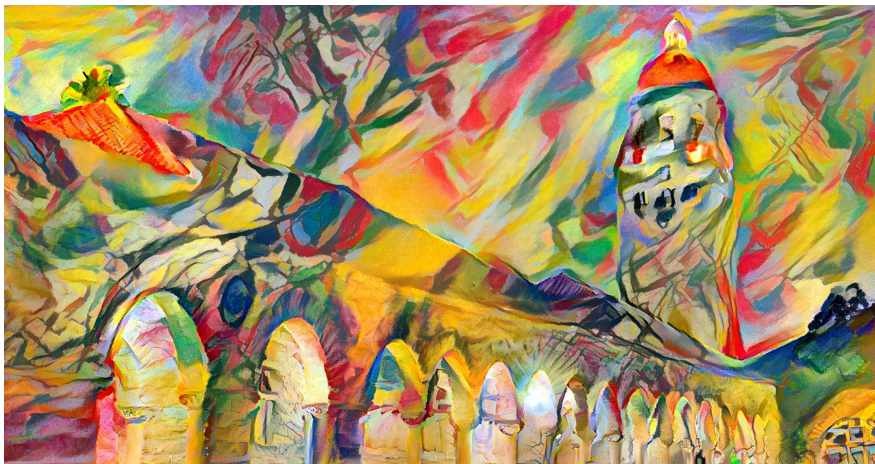


Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure copyright Justin Johnson, 2015.



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 73 May 10, 2017



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 74 May 10, 2017



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 75 May 10, 2017

Neural Style Transfer

Problem: Style transfer requires many forward / backward passes through VGG; very slow!

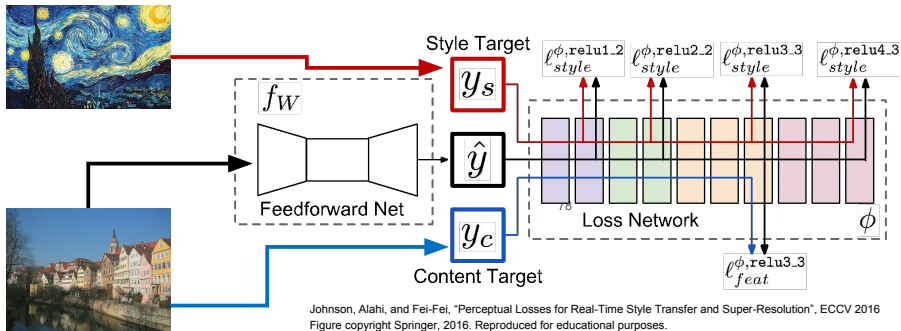
Neural Style Transfer

Problem: Style transfer requires many forward / backward passes through VGG; very slow!

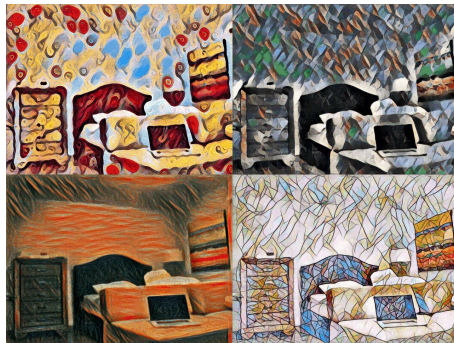
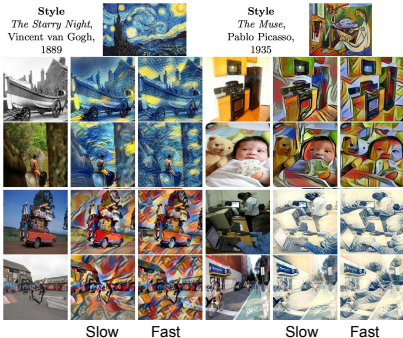
Solution: Train another neural network to perform style transfer for us!

Fast Style Transfer

- (1) Train a feedforward network for each style
- (2) Use pretrained CNN to compute same losses as before
- (3) After training, stylize images using a single forward pass



Fast Style Transfer



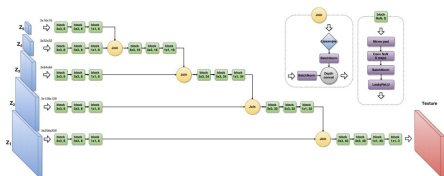
Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016
Figure copyright Springer, 2016. Reproduced for educational purposes.

<https://github.com/icjohnson/fast-neural-style>

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 79 May 10, 2017

Fast Style Transfer



Concurrent work from Ulyanov et al, comparable results

Ulyanov et al, "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images", ICML 2016
 Ulyanov et al, "Instance Normalization: The Missing Ingredient for Fast Stylization", arXiv 2016
 Figures copyright Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky, 2016. Reproduced with permission.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 80 May 10, 2017

Fast Style Transfer



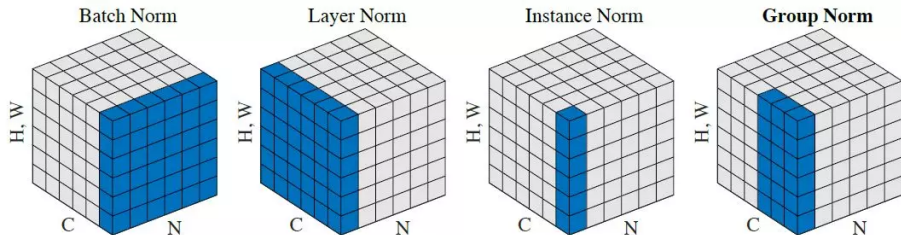
Replacing batch normalization with Instance Normalization improves results

Ulyanov et al, "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images", ICML 2016
 Ulyanov et al, "Instance Normalization: The Missing Ingredient for Fast Stylization", arXiv 2016
 Figures copyright Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky, 2016. Reproduced with permission.

Fei-Fei Li & Justin Johnson & Serena Yeung

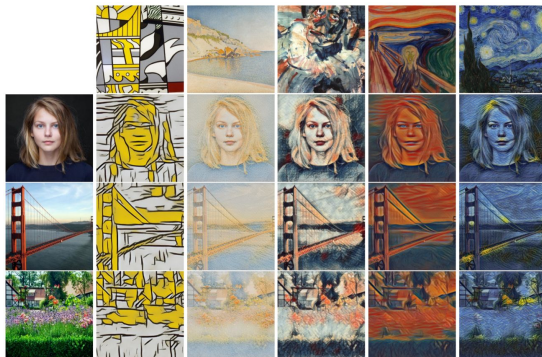
Lecture 11 - 81 May 10, 2017

Different normalizations



C: channel; N: batch size; H,W: height and width

One Network, Many Styles



Dumoulin, Shlens, and Kudlur, "A Learned Representation for Artistic Style", ICLR 2017.

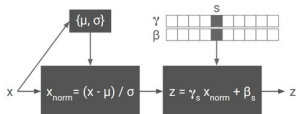
Figure copyright Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur, 2016; reproduced with permission.

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 82 May 10, 2017

One Network, Many Styles

Use the same network for multiple styles using conditional instance normalization: learn separate scale and shift parameters per style



Dumoulin, Shlens, and Kudlur, "A Learned Representation for Artistic Style", ICLR 2017.
Figure copyright Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur, 2016; reproduced with permission.

Single network can blend styles after training

Fei-Fei Li & Justin Johnson & Serena Yeung

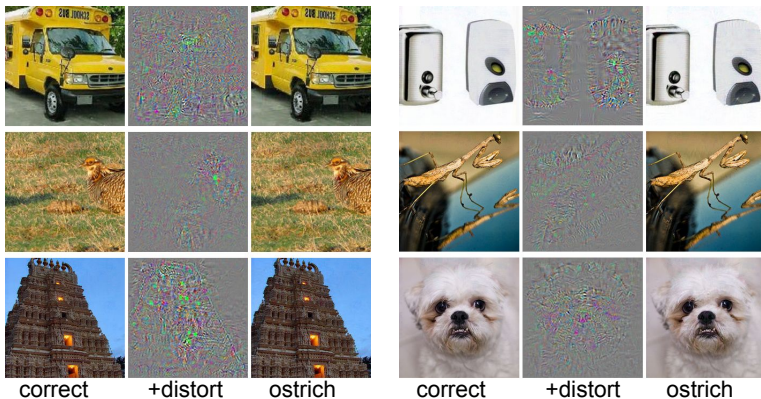
Lecture 11 - 83 May 10, 2017

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?

spoiler alert: yeah

[Intriguing properties of neural networks, Szegedy et al., 2013]



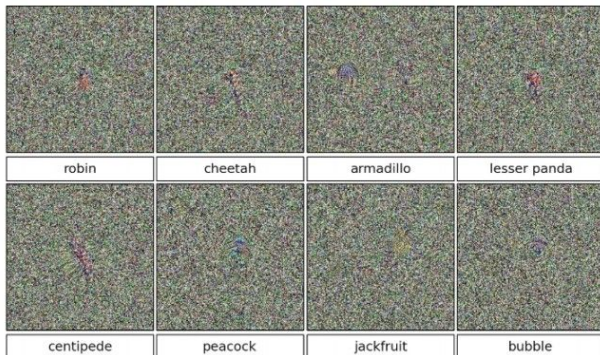
Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 9 - 63

3 Feb 2016

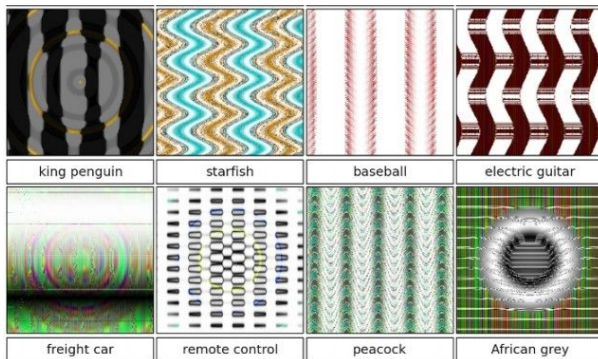
*[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
Nguyen, Yosinski, Clune, 2014]*

>99.6%
confidences



[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images
 Nguyen, Yosinski, Clune, 2014]

>99.6%
 confidences



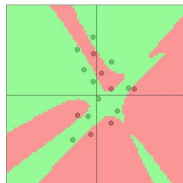
These kinds of results were around even before ConvNets...
[Exploring the Representation Capabilities of the HOG Descriptor, Tatu et al., 2011]



Identical HOG representation

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
 [Goodfellow, Shlens & Szegedy, 2014]

“primary cause of neural networks’ vulnerability to adversarial perturbation is their **linear nature**“
 (and very high-dimensional, sparsely-populated input spaces)



In particular, this is not a problem with Deep Learning, and has little to do with ConvNets specifically. Same issue would come up with Neural Nets in any other modalities.

Conclusions

- Can use optimization and backprop/deconv to visualize weight
 - Can be used to find salient map as well
 - Probably many other uses for this trick as well. Be imaginative!
- CNN for arts (how about not visual data, how about music?)
- Unfortunately, like any other “linear” based classifier, conv-net with softmax layer at the end can be easily fooled

Conclusions

- Can use optimization and backprop/deconv to visualize weight
 - Can be used to find salient map as well
 - Probably many other uses for this trick as well. Be imaginative!
- CNN for arts (how about not visual data, how about music?)
- Unfortunately, like any other “linear” based classifier, conv-net with softmax layer at the end can be easily fooled

Conclusions

- Can use optimization and backprop/deconv to visualize weight
 - Can be used to find salient map as well
 - Probably many other uses for this trick as well. Be imaginative!
- CNN for arts (how about not visual data, how about music?)
- Unfortunately, like any other “linear” based classifier, conv-net with softmax layer at the end can be easily fooled