CS 4803 / 7643: Deep Learning

Topics:
- Visualizing CNNs

Ramprasaath R. Selvaraju
Georgia Tech
Plan for Today

• What do individual neurons look for in images?
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches

• How pixels affect model decisions?
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations

• Do CNNs look at same regions as humans?
  – How to evaluate visualizations?

• Can we synthesize network-specific images?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion
Plan for Today

• What do individual neurons look for in images?
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches

• How pixels affect decisions?
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations

• Do CNNs look at same regions as humans?
  – How to evaluate visualizations?

• Can we synthesize network-specific images?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion
What do individual neurons look for in images?
Class Scores: 1000 numbers

Input Image:
3 x 224 x 224

What are the intermediate features looking for?

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figure reproduced with permission.
Visualizing filters in first layer

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

Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Visualizing filters in intermediate layers

We can visualize filters at higher layers, but not that interesting

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
What do neuron activations look like?
Visualizing activations in intermediate layers
Maximally Activating Patches

Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations
What does the last layer learn?

4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors
Last Layer: Nearest Neighbors

Recall: Nearest neighbors in pixel space

Test image  L2 Nearest neighbors in feature space

Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012. Figures reproduced with permission.
Last Layer: Dimensionality Reduction

Visualize the “space” of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

Simple algorithm: Principal Component Analysis (PCA)

More complex: t-SNE

Van der Maaten and Hinton, "Visualizing Data using t-SNE", JMLR 2008
Figure copyright Laurens van der Maaten and Geoff Hinton, 2008. Reproduced with permission.
Last Layer: Dimensionality Reduction

Van der Maaten and Hinton, "Visualizing Data using t-SNE", JMLR 2008
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012. Figure reproduced with permission.

See high-resolution versions at http://cs.stanford.edu/people/karpathy/cnnembed/

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Plan for Today

• What do individual neurons look for in images?
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches

• **How pixels affect decisions from CNNs?**
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations

• Do CNNs look at same regions as humans?
  – How to evaluate visualizations?

• Can we synthesize network-specific images?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion

(C) Dhruv Batra
How pixels affect decisions?
Visual Explanations

Where does an intelligent system “look” to make its predictions?
Which pixels matter: Occlusion Maps

Idea: Mask part of the image before feeding to CNN, check how much predicted probabilities change

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014
Which pixels matter: Occlusion Maps

Mask part of the image before feeding to CNN, check how much predicted probabilities change

Faithful 😊

Very expensive 😞
What if our model was linear?

\[ \langle \mathbf{w}_c, \mathbf{x} \rangle + b = S_c(x) \]
What if our model was linear?

$$\langle \begin{pmatrix} 100 \\ 0.1 \\ -0.1 \\ 510 \\ -200 \end{pmatrix}, \begin{pmatrix} 1 \\ 0.9 \\ -0.2 \\ 0.5 \\ -0.9 \end{pmatrix} \rangle + b = S_c(x)$$
But it’s not 😞

\[ \langle \mathbf{w}_c , \mathbf{x} \rangle + b = S_c(\mathbf{x}) \]
Can we make it linear?

\[ f(x) = S_c(x) \]
Taylor Series

\[ f(x) \approx f(x_0) + f'(x_0)(x - x_0) \]
Feature Importance in Deep Models

\[ w_c = \left. \frac{\partial S_c}{\partial x} \right|_{x_0} \]

\[ \langle w_c, x \rangle + b \approx S_c(x) \]

Backprop!
Gradient-based visualizations

Backpropagation

\[ \langle w_c, x \rangle + b \approx f(x) \]

\[ w_c = \frac{\partial y_c}{\partial x} \bigg|_{x=x_0} \]

Backprop for `cat'
Backprop for `dog'
Noisy
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.
Saliency Maps

Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Saliency Maps: Segmentation without supervision

Use GrabCut on saliency map

Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.
Rother et al, “Grabcut: Interactive foreground extraction using iterated graph cuts”, ACM TOG 2004

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Remember ReLUs?

\[ h^{l+1} = \text{ReLU}(h^l) = \max\{0, h^l\} \]

\[ \frac{\partial h^{l+1}}{\partial h^l} = \begin{cases} 
0 & \text{if } h^l < 0 \\
1 & \text{if } h^l > 0 
\end{cases} = [h^l > 0] \]
\[ h^{l+1} = \max\{0, h^l\} \]

### Forward pass

\[
\begin{align*}
\frac{\partial L}{\partial h^l} &= \left[h^l > 0\right] \frac{\partial L}{\partial h^{l+1}} \\
\frac{\partial L}{\partial h^l} &= \left[\frac{\partial L}{\partial h^{l+1}} > 0\right] \frac{\partial L}{\partial h^{l+1}} \\
\frac{\partial L}{\partial h^l} &= \left[h^l > 0 \& \& \frac{\partial L}{\partial h^{l+1}} > 0\right] \frac{\partial L}{\partial h^{l+1}}
\end{align*}
\]

#### Backward pass: \textit{deconvnet}"

<table>
<thead>
<tr>
<th>Backward pass: \textit{deconvnet}&quot;</th>
<th>Backward pass: \textit{guided backpropagation}</th>
<th>Backward pass: \textit{backpropagation}</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\begin{array}{ccc} 0 &amp; 3 &amp; 0 \ 6 &amp; 0 &amp; 1 \ 2 &amp; 0 &amp; 3 \end{array}]</td>
<td>[\begin{array}{ccc} 0 &amp; 0 &amp; 0 \ 6 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 3 \end{array}]</td>
<td>[\begin{array}{ccc} -2 &amp; 3 &amp; -1 \ 6 &amp; -3 &amp; 1 \ 2 &amp; -1 &amp; 3 \end{array}]</td>
</tr>
</tbody>
</table>
Gradient-based visualizations

Backprop for `cat'
Backprop for `dog'

Guided Backprop for `cat'
Guided Backprop for `dog'

Noisy
Not Class-Discriminative
Grad-CAM
Visual Explanations from Deep Networks via Gradient-based Localization
[ICCV ‘17]

Ramprasaath Selvaraju  Michael Cogswell  Abhishek Das  Ramakrishna Vedantam

Devi Parikh  Dhruv Batra

Virginia Tech  Facebook Research  Georgia Institute of Technology
Grad-CAM Motivation

- Perturb semantic neurons in the image and see how it affects the decision

- Last convolutional layer forms a best compromise between high-level semantics and detailed spatial resolution
Guided Grad-CAM
Guided Grad-CAM

Guided Backpropagation

Rectified Conv Feature Maps

Any Task-specific Network

Image Classification

(or)

Image Captioning

(or)

Visual Question Answering

(or)

...
Interesting findings with Grad-CAM

- Even simple non-attention based CNN + LSTM models learn to look at appropriate regions
Grad-CAM for captioning

A group of people flying kites on a beach

A man is sitting at a table with a pizza
Grad-CAM for VQA

Even simple non-attention based CNN+LSTM models attend to appropriate regions
Grad-CAM Visual Explanations for VQA

What animal is in this picture? Dog
Grad-CAM Visual Explanations for VQA

What animal is in this picture? Cat
Interesting findings with Grad-CAM

- Even simple non-attention based CNN + LSTM models learn to look at appropriate regions
- Unreasonable predictions often have reasonable explanations
Analyzing Failure modes with Grad-CAM

Even unreasonable predictions have reasonable explanations
Grad-CAM: Gradient-weighted Class Activation Mapping

Grad-CAM highlights regions of the image the captioning model looks at while making predictions.

Try Grad-CAM: Sample Images

Click on one of these images to send it to our servers (Or upload your own images below)
Plan for Today

• What do individual neurons look for in images?
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches
• **How pixels affect decisions?**
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations
• **Do CNNs look at same regions as humans?**
  – How to evaluate visualizations?
• Can we synthesize network-specific images?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion
Do CNNs look at same regions as humans?
VQA-HAT (Human ATtention)

Question: How many players are visible in the image?

Answer:

3
VQA-HAT (Human ATTention)

What food is on the table? Cake
What animal is she riding? Horse
VQA-HAT (Human ATtention)

What number of cats are laying on the bed? 2
Are Grad-CAM explanations human-like?

- Correlation with human attention maps
  [Das & Agarwal et al. EMNLP’16]

<table>
<thead>
<tr>
<th>Method</th>
<th>Rank Correlation w/ HAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guided Backpropagation</td>
<td>0.122</td>
</tr>
<tr>
<td>Guided Grad-CAM</td>
<td>0.136</td>
</tr>
</tbody>
</table>

Current models look at regions more similar to humans than baselines.

Slide Credit: Ram Selvaraju
Plan for Today

• What do individual neurons look for in images?
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches

• How pixels affect decisions?
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations

• Do CNNs look at same regions as humans?
  – How to evaluate visualizations?

• Can we synthesize network-specific images?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream: amplifying detected features
  – Feature inversion
Can we synthesize network-specific images?
Generating prototypical images for a class
Visualizing CNN features: Gradient Ascent on Pixels

(Guided) backprop: Find the part of an image that a neuron responds to?

Gradient ascent on pixels: Generate a synthetic image that maximally activates a neuron

\[ I^* = \arg \max_I [f(I) + R(I)] \]

Neuron value Natural image regularizer

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Visualizing CNN features: Gradient Ascent on Pixels

1. Initialize image to zeros

\[
\arg\max_I S_c(I) - \lambda \|I\|^2
\]

score for class c (before Softmax)

Repeat:
2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Simple regularizer: Penalize L2 norm of generated image

\[
\arg \max_I S_c(I) - \lambda \| I \|^2_2
\]
Visualizing CNN features: Gradient Ascent on Pixels

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

Simple regularizer: Penalize L2 norm of generated image

Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.
<table>
<thead>
<tr>
<th>Authors</th>
<th>Year</th>
<th>Intro</th>
<th>Unregularized</th>
<th>Frequency Penalization</th>
<th>Transformation Robustness</th>
<th>Learned Prior</th>
<th>Dataset Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erhan, et al., 2009</td>
<td>[3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mahendran &amp; Vedaldi, 2015</td>
<td>[7]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nguyen, et al., 2015</td>
<td>[14]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Øygard, et al., 2015</td>
<td>[15]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tyka, et al., 2016</td>
<td>[16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mordvintsev, et al., 2016</td>
<td>[17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nguyen, et al., 2016</td>
<td>[18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nguyen, et al., 2016</td>
<td>[10]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Weak Regularization** avoids misleading correlations, but is less connected to real use.

**Strong Regularization** gives more realistic examples at risk of misleading correlations.
Can neural networks be fooled?
Fooling Images / Adversarial Examples

(1) Start from an arbitrary image
(2) Pick an arbitrary class
(3) Modify the image to maximize the class
(4) Repeat until network is fooled

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Fooling Images / Adversarial Examples

- African elephant
- koala
- Difference
- 10x Difference

- schooner
- iPod
- Difference
- 10x Difference

Boat image is CC0 public domain
Elephant image is CC0 public domain

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
DeepDream: Amplify existing features

Rather than synthesizing an image to maximize a specific neuron, instead try to amplify the neuron activations at some layer in the network.

Choose an image and a layer in a CNN; repeat:

1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer equal to its activation
3. Backward: Compute gradient on image
4. Update image

Mordvintsev, Olah, and Tyka, “Inceptionism: Going Deeper into Neural Networks”, Google Research Blog. Images are licensed under CC-BY 4.0

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
DeepDream: Amplify existing features

Rather than synthesizing an image to maximize a specific neuron, instead try to **amplify** the neuron activations at some layer in the network.

Choose an image and a layer in a CNN; repeat:
1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer **equal to its activation**
3. Backward: Compute gradient on image
4. Update image

Equivalent to:
\[ I^* = \arg \max_l \sum_i f_i(l)^2 \]

Mordvintsev, Olah, and Tyka, “Inceptionism: Going Deeper into Neural Networks”, Google Research Blog, Images are licensed under [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0)
"Admiral Dog!"  "The Pig-Snail"  "The Camel-Bird"  "The Dog-Fish"
Given the feature vector can you reconstruct the image?
Feature Inversion

Given a CNN feature vector for an image, find a new image that:
- Matches the given feature vector
- "looks natural" (image prior regularization)

\[
x^* = \arg\min_{x \in \mathbb{R}^{H \times W \times C}} \ell(\Phi(x), \Phi_0) + \lambda \mathcal{R}(x)
\]

\[
\ell(\Phi(x), \Phi_0) = \|\Phi(x) - \Phi_0\|^2
\]

\[
\mathcal{R}_{\nu^\beta}(x) = \sum_{i,j} \left((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2\right)^{\frac{\beta}{2}}
\]

Feature Inversion

Reconstructing from different layers of VGG-16

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