CS 4803 / 7643: Deep Learning

Topics:
  – Visualizing CNNs

Zsolt Kira
Georgia Tech
Plan for Today

- Visualizing CNNs
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
  - How to evaluate visualizations?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion
  - Style Transfer

(C) Dhruv Batra and Zsolt Kira
What’s going on inside ConvNets?

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.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
First Layer: Visualize Filters

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
Visualize the filters/kernels (raw weights)

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

(these are taken from ConvNetJS CIFAR-10 demo)

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

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

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

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.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Visualizing Activations

Figure copyright Jason Yosinski, 2014. Reproduced with permission.
Visualizing Activations

conv5 feature map is 128x13x13; visualize as 128 13x13 grayscale images

https://youtu.be/AgkfIQ4IGaM?t=92

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

Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

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

• Visualizing CNNs
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches
  – **Occlusion maps**
    – Salient or “important” pixels
      • Gradient-based visualizations
    – How to evaluate visualizations?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion
Visual Explanations

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

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

P(elephant) = 0.95

P(elephant) = 0.75

Zeller and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Karts image is CC0 public domain
Which pixels matter: Occlusion Maps

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

Zeller and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Kart image is CC0 public domain

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
What if our model was linear?

\[ \langle w_c, x \rangle + b = S_c(x) \]
What if our model was linear?

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

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

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

\[ \langle w_c, x \rangle + b \approx S_c(x) \]
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels

Why unnormalized class scores?

Saliency Maps
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
Aside: DeconvNet

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Gradient-based Visualizations

Identical for all layers except ReLU
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**

$${\partial L \over \partial h^l} = \begin{cases} [h^l > 0] & {\partial L \over \partial h^{l+1}} \\ \end{cases}$$

**Backward pass: backpropagation**

$${\partial L \over \partial h^l} = [[h^{l+1} > 0]] \cdot {\partial L \over \partial h^{l+1}}$$

**Backward pass: “deconvnet”**

$${\partial L \over \partial h^l} = [[h^l > 0] \& \& (h^{l+1} > 0)] \cdot {\partial L \over \partial h^{l+1}}$$

**Backward pass: guided backpropagation**

(C) Dhruv Batra and Zsolt Kira
Backprop vs Deconv vs Guided BP

- Guided Backprop tends to be “cleanest”
Intermediate features via (guided) backprop

Maximally activating patches
(Each row is a different neuron)

Guided Backprop

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Intermediate features via (guided) backprop

Maximally activating patches
(Each row is a different neuron)

Guided Backprop

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Visualizing Activations

Figure copyright Jason Yosinski, 2014. Reproduced with permission.
Problem with Guided Backup

- Not very “class-discriminative”

GB for “airliner”

GB for “bus”
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
Grad-CAM

Slide Credit: Ram Selvaraju
Guided Grad-CAM

Guided Backpropagation

Grad-CAM

Rectified Conv Feature Maps

Backprop till conv

Grad-CAM

Guided Grad-CAM

Slide Credit: Ram Selvaraju
Analyzing Failure Modes with Grad-CAM

Reasonable predictions are made in many failure cases.
Grad-CAM Visual Explanations for Captioning

Guided Backprop  Grad-CAM  Guided Grad-CAM

A bathroom with a toilet and a sink

A horse is standing in a field with a fence in the background
Result of Grad-CAM for Visual Question Answering

Enter the question

Answer(Optional)

Submit

Credits

Code for VQA Model
Built by @deshraj
Plan for Today

• Visualizing CNNs
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations
  – How to evaluate visualization?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion
How we evaluate explanations?

• Class-discriminative?
  – Show what they say they found?

• Building Trust with a User?
  – Help users?

• Human-like?
  – Do machines look where humans look?
Is Grad-CAM more class discriminative?

- Can people tell which class is being visualized?
  - Images from Pascal VOC’07 with exactly 2 categories.

What do you see?

- Intuition: A good explanation produces discriminative visualizations for the class of interest.
Is Grad-CAM more class discriminative?

- Human accuracy for 2-class classification

<table>
<thead>
<tr>
<th>Method</th>
<th>Human Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guided Backpropagation</td>
<td>44.44</td>
</tr>
<tr>
<td>Guided Grad-CAM</td>
<td>61.23 (increase of +17%)</td>
</tr>
</tbody>
</table>

Grad-CAM makes existing visualizations class discriminative.
Help establish trust with a user?

• Given explanations from 2 models,
  – VGG16 and AlexNet
  which one is more trustworthy?

• Pick images where both models = correct prediction
• Show these to AMT workers and evaluate
Help establish trust in a user?

Both robots predicted: horse

Robot A based it's decision on

Robot B based it's decision on

Which robot is more reasonable?

- 1. Robot A seems clearly more reasonable than robot B
- 2. Robot A seems slightly more reasonable than robot B
- 3. Both robots seem equally reasonable
- 4. Robot B seems slightly more reasonable than robot A
- 5. Robot B seems clearly more reasonable than robot A

<table>
<thead>
<tr>
<th>Method</th>
<th>Relative Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guided Backpropagation</td>
<td>+1.00</td>
</tr>
<tr>
<td>Guided Grad-CAM</td>
<td>+1.27</td>
</tr>
</tbody>
</table>

Users place higher trust in a model that generalizes better.
Where do humans choose to look to answer visual questions?
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
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.
Plan for Today

• Visualizing CNNs
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations
  – How to evaluate visualizations?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion
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

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

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

score for class c (before Softmax)
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
Visualizing CNN features: Gradient Ascent on Pixels

Simple regularizer: Penalize L2 norm of generated image

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

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


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

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

• Visualizing CNNs
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations
  – How to evaluate visualizations?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion
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](https://research.googleblog.com/). Images are licensed under [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/)
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:
\[ I^* = \arg \max_I \sum_i f_i(I)^2 \]

Mordvintsev, Olah, and Tyka, “Inceptionism: Going Deeper into Neural Networks”, Google Research Blog. Images are licensed under CC-BY 4.0.
"Admiral Dog!"  "The Pig-Snail"  "The Camel-Bird"  "The Dog-Fish"
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)

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

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

\[
\mathcal{R}_V(\mathbf{x}) = \sum_{i,j} \left( (x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\beta/2}
\]


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Side-effect - style transfer

- **Content representation**: feature map at each layer
- **Style representation**: Covariance matrix at each layer
  - Spatially invariant
  - Average second-order statistics

- Idea: Optimize $x$ to match content of one image and style of another

Neural Style

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

e.g.
at CONV5_1 layer we would have a [14x14x512] array of target activations
Neural Style

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

Style gram matrices

\[ G = V^T V \]

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

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)

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

Adapted from Andrej Karpathy
Style transfer
Plan for Today

• Visualizing CNNs
  – Visualizing filters
  – Last layer embeddings
  – Visualizing activations
  – Maximally activating patches
  – Occlusion maps
  – Salient or “important” pixels
    • Gradient-based visualizations
  – How to evaluate visualizations?
  – Creating “prototypical” images for a class
  – Creating adversarial images
  – Deep dream
  – Feature inversion
  – Style Transfer