Topics:
  - Visualizing CNNs
Recap from last time
Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, stride 1 pad 1

Input: 4 x 4

Output: 4 x 4

Dot product between filter and input

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Learnable Upsampling: Transpose Convolution

**Recall:** Normal 3 x 3 convolution, stride 1 pad 1

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

Input gives weight for filter

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Learnable Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

Input: 2 x 2

Output: 4 x 4

Input gives weight for filter

Sum where output overlaps

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

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

Filter

Output

Output contains copies of the filter weighted by the input, summing at where at overlaps in the output.

Need to crop one pixel from output to make output exactly 2x input.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
In-Network upsampling: “Unpooling”

Nearest Neighbor

```
1 2
3 4
```

Input: 2 x 2

```
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4
```

Output: 4 x 4

“Bed of Nails”

```
1 2
0 0 0 0
3 0 4 0
0 0 0 0
```

Input: 2 x 2

```
1 0 2 0
0 0 0 0
3 0 4 0
0 0 0 0
```

Output: 4 x 4

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

- **Downsampling**: Pooling, strided convolution
- **Upsampling**: Unpooling or strided transpose convolution

Input: $3 \times H \times W$

High-res:
- $D_1 \times H/2 \times W/2$
- $D_3 \times H/4 \times W/4$

Med-res:
- $D_2 \times H/4 \times W/4$

Low-res:
- $D_3 \times H/4 \times W/4$

Predictions: $H \times W$


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Why this operation?
What is deconvolution?

- (Non-blind) Deconvolution

\[
y = \mathbf{W} \mathbf{x} = \mathbf{W} (\mathbf{x} \ast \mathbf{n})
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]

\[
y = \mathbf{W} \mathbf{x}
\]
What is deconvolution?

- (Non-blind) Deconvolution

\[ y = w \ast x \]

\[ y = \begin{bmatrix} w_k & 0 & \cdots & 0 & 0 \\ w_{k-1} & w_k & \cdots & 0 & 0 \\ w_{k-2} & w_{k-1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_1 & w_{k-2} & \cdots & w_k & 0 \\ 0 & w_1 & \cdots & w_{k-1} & w_k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & w_1 & w_2 \\ 0 & 0 & \cdots & 0 & w_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \]

Inversion:

\[ w^{-1} = n \]
What does “deconvolution” have to do with “transposed convolution”? 

\[ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \overset{W}{\mapsto} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \]

\[ \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \overset{W^T}{\mapsto} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \]
We can express convolution in terms of a matrix multiplication.

\[
\mathbf{x} * \mathbf{a} = \mathbf{Xa}
\]

Example: 1D conv, kernel size=3, stride=1, padding=1

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
"transposed convolution" is a convolution!

We can express convolution in terms of a matrix multiplication:

\[
\begin{bmatrix}
 x \\
 y \\
 z \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 \end{bmatrix} \times \begin{bmatrix}
 a \\
 b \\
 c \\
 d \\
 0 \\
 0 \\
 0 \\
 \end{bmatrix} = \begin{bmatrix}
 ay + bz \\
 ax + by + cz \\
 bx + cy + dz \\
 cx + dy \\
 \end{bmatrix}
\]

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

\[
\begin{bmatrix}
 x \\
 y \\
 z \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 \end{bmatrix} \times^T \begin{bmatrix}
 a \\
 b \\
 c \\
 d \\
 0 \\
 0 \\
 0 \\
 \end{bmatrix} = \begin{bmatrix}
 ax \\
 ay + bx \\
 az + by + cx \\
 bz + cy + dx \\
 cx + dy \\
 dz \\
 \end{bmatrix}
\]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
“transposed convolution” is a convolution!

We can express convolution in terms of a matrix multiplication

\[
\begin{align*}
\bar{x} * \bar{a} &= X \bar{a} \\
\begin{bmatrix}
x & y & z & 0 & 0 & 0 \\
0 & x & y & z & 0 & 0 \\
0 & 0 & x & y & z & 0 \\
0 & 0 & 0 & x & y & z \\
\end{bmatrix}
\begin{bmatrix}
0 \\
a \\
b \\
c \\
d \\
0 \\
\end{bmatrix}
\begin{bmatrix}
ay + bz \\
ax + by + cz \\
bx + cy + dz \\
cx + dy \\
\end{bmatrix}
\end{align*}
\]

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

\[
\begin{align*}
\bar{x} * T \bar{a} &= X^T \bar{a} \\
\begin{bmatrix}
x & 0 & 0 & 0 \\
y & x & 0 & 0 \\
z & y & x & 0 \\
0 & z & y & x \\
0 & 0 & z & y \\
0 & 0 & 0 & z \\
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c \\
d \\
\end{bmatrix}
\begin{bmatrix}
ax \\
ay + bx \\
ax + by + cx \\
by + cz + dx \\
\end{bmatrix}
\end{align*}
\]

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

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
What’s going on inside ConvNets?

Input Image: 3 x 224 x 224

Class Scores: 1000 numbers

What are the intermediate features looking for?

Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012. Figure reproduced with permission.
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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Last Layer: Nearest Neighbors

Recall: Nearest neighbors in pixel space

Test image

L2 Nearest neighbors in feature space

4096-dim vector

Figures reproduced with permission.

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

Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008
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
Visualizing Activations

Figure copyright Jason Yosinski, 2014. Reproduced with permission.
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

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.

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

Boat image: cc0 public domain
Elephant image: cc0 public domain
Go-Karts image: cc0 public domain
What if our model was linear?

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

\[
\begin{pmatrix}
100 \\
0.1 \\
-0.1 \\
510 \\
-200
\end{pmatrix}
+ b = S_c(x)
\]
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!
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

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

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


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

Use GrabCut on saliency map
Gradient-based Visualizations

- **Raw Gradients**
  - [Simonyan et al. ICLRw ‘14]

- **‘Deconvolution’**
  - [Zeiler & Fergus, ECCV ‘14]

- **Guided Backprop**
  - [Springenberg et al. ICLR ‘15]

Identical for all layers except ReLU
Remember ReLUs?

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

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

\[ \frac{\partial L}{\partial h^l} = \left[ [h^l > 0] \right] \frac{\partial L}{\partial h^{l+1}} \]

\[ \frac{\partial L}{\partial h^l} = \left[ [h^{l+1} > 0] \right] \frac{\partial L}{\partial h^{l+1}} \]

\[ \frac{\partial L}{\partial h^l} = \left[ (h^l > 0) \& \& (h^{l+1} > 0) \right] \frac{\partial L}{\partial h^{l+1}} \]
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

Zeller 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

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

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

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

- Not very “class-discriminative”

GB for “airliner”  GB for “bus”

Slide Credit: Ram Selvaraju
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
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

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.

Intuition: A good explanation produces discriminative visualizations for the class of interest.

What do you see?

Your options:
- horse
- person
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</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 its decision on Robot B based its 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
What food is on the table? Cake
VQA-HAT (Human ATtention)

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]

What are they doing?

<table>
<thead>
<tr>
<th>Method</th>
<th>Rank Correlation with 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>

Grad-CAM for ‘eating’

Human ATtention map (HAT) for ‘eating’

Current models look at regions more similar to humans than baselines

Slide Credit: Ram Selvaraju
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

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

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

score for class c (before Softmax)

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

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

Simple regularizer: Penalize L2 norm of generated image


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

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

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

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

\[ f(x) = w^T x + b = \begin{bmatrix} w_1 & \ldots & w_n \end{bmatrix} x_i \]

- African elephant
- koala
- schooner
- iPod

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

Boat image = CC0 public domain
Elephant image = CC0 public domain
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.google.com/blog). 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.
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)

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

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

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

Feature Inversion

Reconstructing from different layers of VGG-16

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

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.