CS 4803 / 7643: Deep Learning

Topics:
- Convolutional Neural Networks
  - What is a convolution?
  - FC vs Conv Layers

Dhruv Batra
Georgia Tech
Administrativia

- HW2 Reminder
  - Due: 09/23, 11:59pm

- Project Teams
  - [https://gtvault-my.sharepoint.com/:x:/g/personal/dbatra8_gatech_edu/EY4_65XOzWtOkXSSz2WgpoUBY8ux2gY9PsRzR6KngI5EQ?e=4tnKWI](https://gtvault-my.sharepoint.com/:x:/g/personal/dbatra8_gatech_edu/EY4_65XOzWtOkXSSz2WgpoUBY8ux2gY9PsRzR6KngI5EQ?e=4tnKWI)
  - Project Title
  - 1-3 sentence project summary TL;DR
  - Team member names

(C) Dhruv Batra
Thoughts on Gaier & Ha NeurIPS19
Error Decomposition

Reality

Modeling Error

Multi-class Logistic Regression

Softmax

FC HxWx3

Input

Estimation Error

Optimization Error = 0

horse

person

(C) Dhruv Batra
Thoughts on Gaier & Ha NeurIPS19

- Inductive biases
  - Nice timing wrt CNNs
  - See HW2 Q4

- Architecture elements as lego blocks

- Guest lecture on Neural Architecture Search

- Learned vs innate?
Plan for Today

• (Finish) Automatic Differentiation
  – Jacobians in FC+ReLU NNs

• Convolutional Neural Networks
  – What is a convolution?
  – FC vs Conv Layers
Jacobian of ReLU

g(x) = \max(0, x) 
(elementwise)
Q1: what is the size of the Jacobian matrix?

4096-d input vector

\[ g(x) = \max(0, x) \] (elementwise)

4096-d output vector

\[ \frac{\partial \hat{e}}{\partial \hat{e}} \]

4096 x 4096
Jacobian of ReLU

\[ g(x) = \max(0, x) \quad (\text{elementwise}) \]

Q1: what is the size of the Jacobian matrix? 
[4096 x 4096!]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Q1: what is the size of the Jacobian matrix? [4096 x 4096!]

in practice we process an entire minibatch (e.g. 100) of examples at one time:

i.e. Jacobian would technically be a [409,600 x 409,600] matrix :\"
Q1: what is the size of the Jacobian matrix? [4096 x 4096!]

Q2: what does it look like?

\[
\frac{\partial h_i}{\partial h_i} = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{otherwise}
\end{cases}
\]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Jacobians of FC-Layer
Jacobians of FC-Layer

\[ \frac{\partial L}{\partial \bar{h}^e} = \frac{\partial L}{\partial \hat{h}^e} \cdot \frac{\partial \hat{h}^e}{\partial \bar{h}^e} \]

\[ \hat{h}^e = W \bar{h}^e \]

\[ \frac{\partial \hat{h}^e}{\partial \bar{h}^e} = W \]
Jacobians of FC-Layer
Plan for Today

• (Finish) Automatic Differentiation
  – Jacobians in FC+ReLU NNs

• Convolutional Neural Networks
  – What is a convolution?
  – FC vs Conv Layers
Plan for Today

• Convolutional Neural Networks
  – What is a convolution?
  – FC vs Conv Layers
Recall: Linear Classifier

\[ f(x, W) = Wx + b \]

Array of 32x32x3 numbers (3072 numbers total)

\[ W \]

parameters or weights

10 numbers giving class scores

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Input image

Stretch pixels into column

\[
\begin{array}{cccc}
0.2 & -0.5 & 0.1 & 2.0 \\
1.5 & 1.3 & 2.1 & 0.0 \\
0.0 & 0.25 & 0.2 & -0.3 \\
\end{array}
\]

\[
\begin{array}{c}
56 \\
231 \\
24 \\
2 \\
\end{array}
\]

\[
\begin{array}{c}
1.1 \\
3.2 \\
-1.2 \\
\end{array}
\]

\[
\begin{array}{c}
-96.8 \\
437.9 \\
61.95 \\
\end{array}
\]

\[
\begin{array}{c}
\text{Cat score} \\
\text{Dog score} \\
\text{Ship score} \\
\end{array}
\]
Recall: (Fully-Connected) Neural networks

(Before) Linear score function:

\[ f = Wx \]

(Now) 2-layer Neural Network

\[ f = W_2 \max(0, W_1x) \]
Convolutional Neural Networks
(without the brain stuff)
Fully Connected Layer

Example: 200x200 image
40K hidden units

Q: what is the number of parameters in this FC layer?

Slide Credit: Marc'Aurelio Ranzato
Fully Connected Layer

Example: 200x200 image
40K hidden units

Q: what is the number of parameters in this FC layer?
A: ~2 Billion

- Spatial correlation is local
- Waste of resources + too many parameters to learn

Slide Credit: Marc'Aurelio Ranzato
Assumption 1: Locally Connected Layer

Example: 200x200 image
40K hidden units
Connection size: 10x10
4M parameters

Note:
This parameterization is good when input image is registered (e.g., face recognition).

Slide Credit: Marc'Aurelio Ranzato
Assumption 2: Stationarity / Parameter Sharing

Stationarity?
Statistics similar at all locations

\[ \overline{w_i} = \overline{w_j} \quad \forall i, j \]

10x10 = 100 parameters

2B → 4M

Slide Credit: Marc'Aurelio Ranzato
Convolutional Layer

Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels
Convolutions!

\[
\text{math} \rightarrow \text{CS} \rightarrow \text{programming}
\]
Convolutions for mathematicians

\[ x(t) \quad w(t) \quad y(t) \]

\[ x(t) = e^{-\frac{t^2}{\sigma^2}} \]

\[ y(t) = (x \ast w) (t) = \int_{-\infty}^{\infty} x(t-a)w(a) \, da \]

\[ = (w \ast x) (t) = \int_{-\infty}^{\infty} x(a)w(t-a) \, da \]
Convolutions for mathematicians

\[ y(t) = \int_{-\infty}^{\infty} x(a) w(t-a) \, da \]

- Function \( w(a) \) is flipped about the \( y \)-axis.
- Then, it is time shifted by \( t \).

\[ y(t) \]

(C) Dhruv Batra
Convolutions for mathematicians

• One dimension

\[ y(t) = \int_{-\infty}^{\infty} x(t-a) \omega(a) \, da \]

• Two dimensions

\[ y(t_1, t_2) = \iint_{-\infty}^{\infty} x(t_1-a, t_2-b) \omega(a, b) \, da \, db \]
Convolutions for computer scientists

1) No init proc

\[ y[t_1, t_2] = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} x[t_1-a, t_2-b] w[a, b] \]

2) No init memory

\[ \begin{vmatrix} \frac{k-1}{2} \end{vmatrix} \quad \begin{vmatrix} \frac{k-1}{2} \end{vmatrix} \]

\[ a = -\frac{k-1}{2} \quad b = -\frac{k-1}{2} \]

\[ y[t_1, t_2] \]
Convolutions for computer scientists
Convolutions for programmers

\[ y[x, c] = \sum_{a=0}^{k_1-1} \sum_{b=0}^{k_2-1} x[x+a, c+b] w[a, b] \]
Convolutions for programmers
Convolutional Layer
Convolutional Layer

(C) Dhruv Batra

Slide Credit: Marc'Aurelio Ranzato
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer

Slide Credit: Marc'Aurelio Ranzato

(C) Dhruv Batra
Convolutional Layer
Convolution Explained

- [https://github.com/bruckner/deepViz](https://github.com/bruckner/deepViz)
Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters
FC vs Conv Layer
FC vs Conv Layer
Convolution Layer

32x32x3 image

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

A 32x32x3 image is convolved with a 5x5x3 filter. This means sliding the filter over the image spatially, computing dot products.

Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”
Convolution Layer

Filters always extend the full depth of the input volume

Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”
Convolution Layer

32x32x3 image
5x5x3 filter $w$

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. $5\times 5\times 3 = 75$-dimensional dot product + bias)

$$w^T x + b$$

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

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map

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

consider a second, green filter

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation maps

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:

We stack these up to get a “new image” of size 28x28x6!
Im2Col

Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/
GEMM

(C) Dhruv Batra  Figure Credit: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/
Time Distribution of AlexNet

GPU Forward Time Distribution
- fc7: 0.8%
- fc6: 1.8%
- conv5: 17.7%
- conv4: 17.8%
- relu3: 0.2%
- conv3: 17.8%
- 17.8%
- 21.9%
- 16.9%
- 16.9%
- 9.4%
- 19.6%
- 14.7%
- 18.7%
- 23.7%

CPU Forward Time Distribution
- fc6: 2.6%
- conv5: 9.4%
- relu1: 0.7%
- pool1: 1%
- pool2: 0.5%
- conv1: 19.6%
- conv2: 4%
- norm1: 1.2%
- conv2: 23.7%
- relu2: 0.4%
- norm2: 18.7%
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions.
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions.