CS440/ECE448 Lecture 15 Computer Vision: Convolutional Networks

Mark Hasegawa-Johnson, 2/2024

Slides are in Public Domain: Re-Mix, Re-Use, Re-Distribute at will



Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Remember: a neural network learns a PWL approximation of a nonlinear function

- We can approximate any nonlinear classifier using a PWL classifier
- In the limit, as the number of hidden nodes goes to infinity, the approximation becomes provably perfect



Public domain image, Krishnavedala, 2011

The first layer is a matrix followed by a nonlinearity. The second layer is a linear classifier.

$$f = \operatorname{softmax}(w_1@h + b_1)$$

 $h = ReLU(w_0@x + b_0)$



Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

The problem of shift invariance

- These two images each contain a cat.
- How can a neural network tell?





Fully-connected network

A fully-connected network converts the image into a vector, then multiplies it by a matrix.



Fully-connected network

A fully-connected network converts the image into a vector, then multiplies it by a matrix.



The problem of shift invariance

... is that, even though both images contain a cat, the vectorized versions of these two images are dissimilar.



The first layer is a matrix followed by a nonlinearity. The second layer is a linear classifier.

- Since the vectors aren't similar, they won't produce similar hidden features.
- Since the hidden features aren't similar, they won't be recognized as the same animal.



Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition



- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition



- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition



- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition



- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition



- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition



- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition



- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition



- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition





- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition





- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition





How do we measure similarity? The math.



How do we measure similarity? Answer: local dot product = correlation



How do we measure similarity? Answer: local dot product = correlation

$$\sum_{i} \sum_{j} h[i - k, j - l]x[i, j]$$

$$= y[k, l]$$

1

Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Correlation versus Convolution

• This formula is called the *correlation* of h[i, j] with x[i, j]:

$$y[k,l] = \sum_{i} \sum_{j} h[i-k,j-l]x[i,j]$$

• This formula is called the *convolution* of h[i, j] with x[i, j]:

$$y[k,l] = \sum_{i} \sum_{j} h[k-i,l-j]x[i,j]$$

Why use convolution if correlation is easier to visualize?

- Historical reason: Convolution is related to Fourier transform
- Computational reason? Not really, they have exactly the same computational complexity
- The only reason that is kind of useful: convolution, unlike correlation, is symmetric:

$$y[k,l] = \sum_{i} \sum_{j} h[k-i,l-j]x[i,j] = \sum_{i} \sum_{j} h[i,j]x[k-i,l-j]$$

Convolutional neural net

A convolutional neural net is the same thing as a correlational neural net, except that the filter is defined upside down and backward, like this.



Convolutional neural net h[-i, -j] = 1

The convolution operation then

- Flips the filter rightside up,
- Shifts it to h[k i, l j]
- Computes the similarity:

$$y[k,l] = \sum_{i} \sum_{j} h[k-i,l-j]x[i,j]$$



|k|

Convolutional neural net

$$y[k,l] = \sum_{i} \sum_{j} h[k-i,l-j]x[i,j]$$

This is often written as y[k, l] = h[k, l] * x[k, l]



Quiz

• Try the quiz!

https://us.prairielearn.com/pl/course_instance/147925/assessment/ 2400008

Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Back-prop through convolution

- Suppose we've computed $y[k, l] = \sum_{i} \sum_{j} h[i, j] x[k i, l j]$
- Now suppose we know $\frac{d\mathcal{L}}{dy[k,l]}$
- We want to train $h[i, j] = h[i, j] \eta \frac{d\mathcal{L}}{dh[i, j]}$
- How do we find $\frac{d\mathcal{L}}{dh[i,j]}$?

Back-prop through convolution

Answer: use the chain rule!

 $y[k, l] = \sum_{i} \sum_{j} h[i, j] x[k - i, l - j]$, so...

$$\frac{d\mathcal{L}}{dh[i,j]} = \sum_{k} \sum_{l} \frac{d\mathcal{L}}{dy[k,l]} \frac{dy[k,l]}{dh[i,j]}$$
$$= \sum_{k} \sum_{l} \frac{d\mathcal{L}}{dy[k,l]} x[k-i,l-j]$$

Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Too much information







1



Ī





Object detection: convolution, max pooling, vectorize, then use a linear classifier

- Convolution detects an object, no matter where it is,
- Max pooling keeps the best detection of the object in each region,
- Linear classifier tests if the best detection is good enough.



Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through convolution
- Real-World CNNs

Back-prop through max-pooling

- Suppose we've computed $z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$
- Now suppose we know $\frac{d\mathcal{L}}{dz[m,n]}$
- In order to train the neural net, we need $\frac{d\mathcal{L}}{dv[k,l]}$
- How do we find $\frac{d\mathcal{L}}{dy[k,l]}$?

Back-prop through max-pooling

Answer: use the chain rule! $z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l] \text{, so...}$ $\frac{d\mathcal{L}}{dy[k,l]} = \frac{d\mathcal{L}}{dz[m,n]} \frac{dz[m,n]}{dy[k,l]}$ $= \begin{cases} \frac{d\mathcal{L}}{dz[m,n]} & \text{if } y[k,l] = \max_{\substack{(m-1)p+1 \le i \le mp, \\ (n-1)p+1 \le j \le np} \\ 0 & \text{otherwise}} \end{cases} y[i,j]$

Back-prop through max-pooling



Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through convolution
- Real-World CNNs

LeNet, 1998

- (Convolution, Nonlinearity, Max-pooling, Normalization) X 2 layers
- After the second subsampling, the resulting features are concatenated to form a vector, which is then classified.



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, <u>Gradient-based learning applied to document recognition</u>, Proc. IEEE 86(11): 2278–2324, 1998.

AlexNet

- Similar framework to LeNet but:
 - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
 - More data (10⁶ vs. 10³ images)
 - GPU implementation (50x speedup over CPU)
 - Trained on two GPUs for a week



A. Krizhevsky, I. Sutskever, and G. Hinton, <u>ImageNet Classification with Deep Convolutional</u> <u>Neural Networks</u>, NIPS 2012

ImageNet Challenge

IM GENET



[Deng et al. CVPR 2009]

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

A. Krizhevsky, I. Sutskever, and G. Hinton, <u>ImageNet Classification with Deep Convolutional</u> <u>Neural Networks</u>, NIPS 2012

Layer 1 Filters



M. Zeiler and R. Fergus, <u>Visualizing and Understanding Convolutional Networks</u>, arXiv preprint, 2013

Layer 1: Top-9 Patches











Conclusion: Convolution and Max Pooling

$$y[k,l] = h[k,l] * x[k,l] = \sum_{i} \sum_{j} h[k-i,l-j]x[i,j]$$
$$\frac{d\mathcal{L}}{dh[i,j]} = \sum_{k} \sum_{l} \frac{d\mathcal{L}}{dy[k,l]} \frac{dy[k,l]}{dh[i,j]}$$

$$\begin{aligned} z[m,n] &= \max_{\substack{(m-1)p+1 \leq k \leq mp, \\ (n-1)p+1 \leq l \leq np}} y[k,l] \\ \frac{d\mathcal{L}}{dy[k,l]} &= \begin{cases} \frac{d\mathcal{L}}{dz[m,n]} & \text{if } y[k,l] = \max_{\substack{(m-1)p+1 \leq i \leq mp, \\ (n-1)p+1 \leq j \leq np}} y[i,j] \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$