Tools & Models for Data Science

Neural Networks: Bells and Whistles

Jonas Actor and Risa Myers

Rice University

Motivation

NN are large nonconvex optimization problems

. . . large nonconvex optimization problems are hard to solve

Objective

Learn common ways to ______________ of NNs:

1. combat model complexity
2. improve generalization
3. overcome difficulties in training

Bells and Whistles

1. Convolutional Neural Networks
2. Multi-resolution Networks
3. Dropout
4. Batch Normalization

Table of Contents

What is a discrete convolution?

Linear operator that applies a small kernel everywhere

- local point-wise multiplication
- reduce via sum

What is a discrete convolution?

k: kernel of size n_k a small number
x: vector

$$(x * k)_i = \sum_{j=-n_k}^{n_k} x_{i+j} k_j$$

Example of 1D Convolution

$$x = \begin{bmatrix} 1 & 2 & -3 & 0 & 4 & -1 & 2 \end{bmatrix}$$
$$k = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Example of 1D Convolution

$$\begin{bmatrix} 1 & 2 & -3 & 0 & -4 & -1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} \quad \end{bmatrix}$$

Example of 1D Convolution

$$\begin{bmatrix} 1 & 2 & -3 & 0 & -4 & -1 & 2 \end{bmatrix}$$

$$\underbrace{-1 \quad 0 \quad 1}$$

$$\downarrow$$

$$\begin{bmatrix} & -4 & & & & & \end{bmatrix}$$

Example of 1D Convolution

$$\begin{bmatrix} \quad 1 & 2 & -3 & 0 & -4 & -1 & 2 \quad \end{bmatrix}$$

$$\begin{matrix} & -1 & 0 & 1 & & & & \end{matrix}$$

$$\begin{bmatrix} \quad & -4 & -2 & & & & \quad \end{bmatrix}$$

Example of 1D Convolution

$$
\begin{array}{ccccccccc}
[& 1 & 2 & -3 & 0 & -4 & -1 & 2 &] \\
 & & & -1 & 0 & 1 & & & \\
 & & & & \downarrow & & & & \\
[& & -4 & -2 & -1 & & & &]
\end{array}
$$

Example of 1D Convolution

$$\begin{bmatrix} 1 & 2 & -3 & 0 & -4 & -1 & 2 \end{bmatrix}$$

$$-1 \quad 0 \quad 1$$

$$\downarrow$$

$$\begin{bmatrix} & -4 & -2 & -1 & -1 & & \end{bmatrix}$$

Example of 1D Convolution

$$
\begin{array}{ccccccccc}
[& 1 & 2 & -3 & 0 & -4 & -1 & 2 &] \\
  &   &   &   &   & -1 & 0 & 1 &   \\
  &   &   &   &   &   & \downarrow & & \\
[&   & -4 & -2 & -1 & -1 & 6 & &]
\end{array}
$$

What to do at the boundary

1. ignore them! (‘valid’)
2. pad with zeros (‘same’)
3. wrap around the end (‘periodic’)

Example with Valid Padding

$$\begin{bmatrix} 1 & 2 & -3 & 0 & -4 & -1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} & -4 & -2 & -1 & -1 & 6 & \end{bmatrix}$$

Example with Valid Padding

$$\begin{bmatrix} 1 & 2 & -3 & 0 & -4 & -1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} -4 & -2 & -1 & -1 & 6 \end{bmatrix}$$

Example with Zero Padding

$$\begin{bmatrix} 1 & 2 & -3 & 0 & -4 & -1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} & -4 & -2 & -1 & -1 & 6 & \end{bmatrix}$$

Example with Zero Padding

$$\begin{bmatrix} \color{red}{0} & 1 & 2 & -3 & 0 & -4 & -1 & 2 & \color{red}{0} \end{bmatrix}$$

$$\begin{matrix} -1 & 0 & 1 & & & & -1 & 0 & 1 \end{matrix}$$

$$\begin{bmatrix} 2 & -4 & -2 & -1 & -1 & 6 & 1 \end{bmatrix}$$

Example with Periodic Padding

$$\begin{bmatrix} & 1 & 2 & -3 & 0 & -4 & -1 & 2 & \end{bmatrix}$$

$$\begin{bmatrix} & & & -4 & -2 & -1 & -1 & 6 & & & \end{bmatrix}$$

Example with Periodic Padding

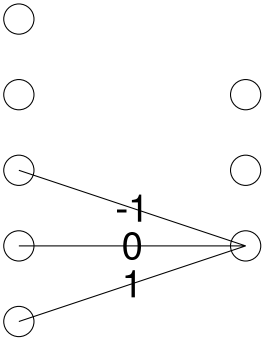
$$\begin{bmatrix} \color{red}{2} & 1 & 2 & -3 & 0 & -4 & -1 & 2 & \color{red}{1} \end{bmatrix}$$

$$\begin{array}{ccccccccc} -1 & 0 & 1 & & & & -1 & 0 & 1 \end{array}$$

$$\begin{bmatrix} 0 & -4 & -2 & -1 & -1 & 6 & 2 \end{bmatrix}$$

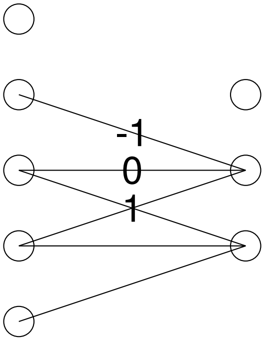
Convolutions in Neural Networks

Learn small kernel instead of big weight matrix



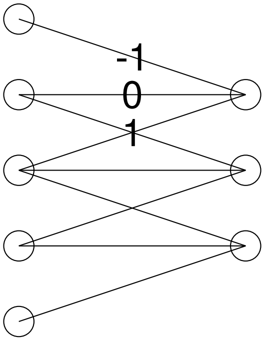
Convolutions in Neural Networks

Learn small kernel instead of big weight matrix



Convolutions in Neural Networks

Learn small kernel instead of big weight matrix



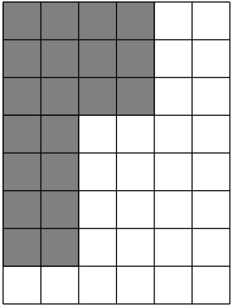
Why CNNs help

- Position matters $\rightarrow$ model complexity
- Only consider local information $\rightarrow$ model complexity
- Only learn a small kernel, not big weight matrix $\rightarrow$ model complexity, size

Problems where CNNs are useful

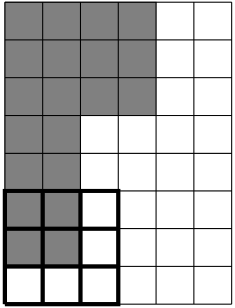
- image/video tasks
  - classification
  - segmentaion
  - . . .
- sequence analysis
- graphical models

Images: 2D convolution



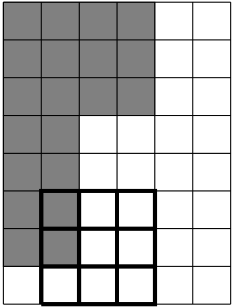
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Images: 2D convolution



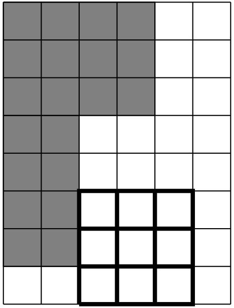
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ -3 & & & \end{bmatrix}$$

Images: 2D convolution



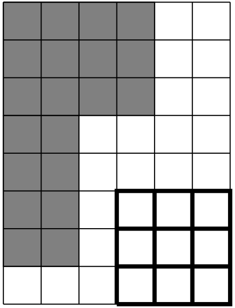
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \\ -3 & -3 & & \\ & & & \end{bmatrix}$$

Images: 2D convolution



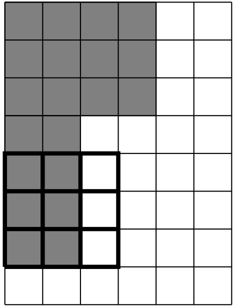
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \\ & & \\ -3 & -3 & 0 \\ & & \end{bmatrix}$$

Images: 2D convolution



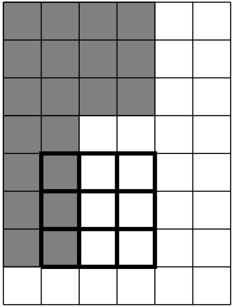
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \\ -3 & -3 & 0 & 0 \\ & & & \end{bmatrix}$$

Images: 2D convolution



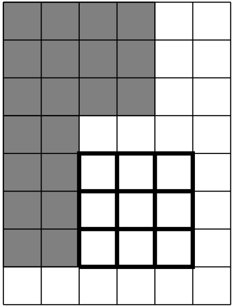
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \\ -4 & & & \\ -3 & -3 & 0 & 0 \end{bmatrix}$$

Images: 2D convolution



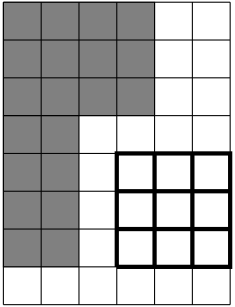
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \\ -4 & -4 & & \\ -3 & -3 & 0 & 0 \end{bmatrix}$$

Images: 2D convolution



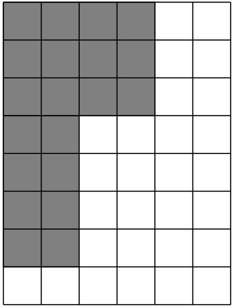
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \\ -4 & -4 & 0 & \\ -3 & -3 & 0 & 0 \\ & & & \end{bmatrix}$$

Images: 2D convolution



$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -4 & -4 & 0 & 0 \\ -3 & -3 & 0 & 0 \end{bmatrix}$$

Images: 2D convolution

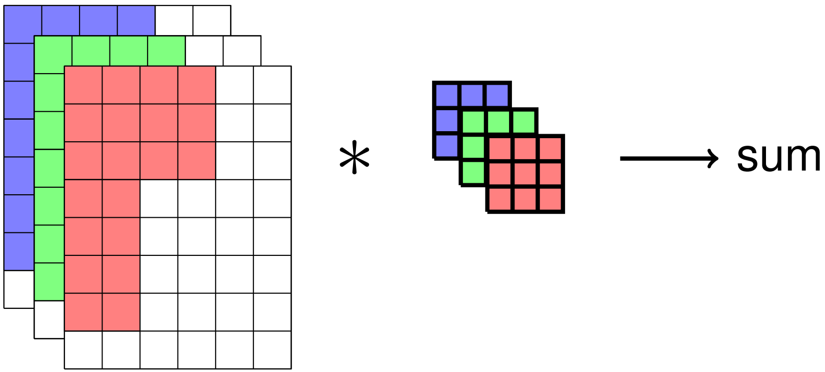


$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -4 & -4 \\ -1 & -1 & -3 & -3 \\ -3 & -3 & -1 & -1 \\ -4 & -4 & 0 & 0 \\ -4 & -4 & 0 & 0 \\ -3 & -3 & 0 & 0 \end{bmatrix}$$

Channels in, channels out

- keep track of *image feature* instead of individual pixels
- node in NN is a channel (= feature)
- add bias to every channel
  $\rightarrow$ add constant to each image feature
- apply σ to each pixel
- channels in $\rightarrow$ # convolutions summed together in output
- channels out $\rightarrow$ # distinct convolution kernels learned

Convolution with multiple channels in



Convolution with multiple channels in

$$(320 \times 240 \times 3) * 3 \times 3 \times 3 \text{ kernel} = (320 \times 240 \times 1)$$

Convolution with multiple channels in

$$(n_x \times n_y \times n_{in}) * n_k \times n_k \times n_{in} \text{ kernel } = (n_x \times n_y \times 1)$$

- n_x : pixels in x direction
- n_y : pixels in y direction
- n_k : window size of convolution kernel
- n_{in} : channels in
- n_{out} : channels out

Convolution with multiple channels out

$$(n_x \times n_y \times n_{in}) * [n_k \times n_k \times n_{in}] \times n_{out} \text{ kernel } = (n_x \times n_y \times n_{out})$$

- n_x : pixels in x direction
- n_y : pixels in y direction
- n_k : window size of convolution kernel
- n_{in} : channels in
- n_{out} : channels out

A complete convolution layer

$$\text{layer}_\ell \longmapsto \vec{\sigma}\,(\,\text{layer}_\ell \,*\, \text{kernel}_\ell \,+ \text{bias}_\ell\,) = \text{layer}_{\ell+1}$$

Downsides

- Longer to train
  $\rightarrow$ backpropagation needs global data to update kernel weights
  $\rightarrow$ generalization: locally connected layers

- Requires a grid
  $\rightarrow$ generalization: graph convolutions

Table of Contents

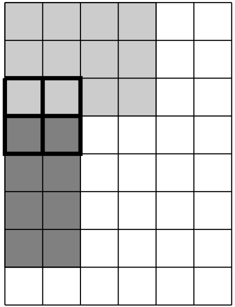
Data at multiple scales

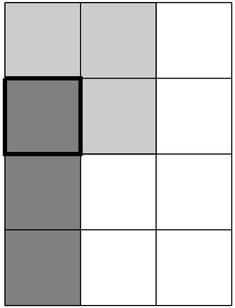
- images
- time series data

Downsampling and Upsampling

- Reduces/increases number of pixels at each node
- Captures data at different resolutions
- Removes noise

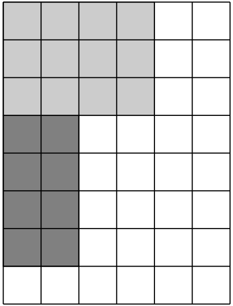
Example: Downsampling via Max Pooling



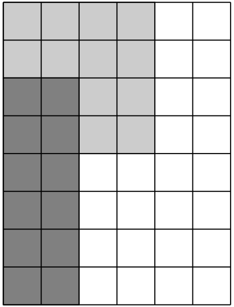


Example: Upsampling

Pooling causes loss of information

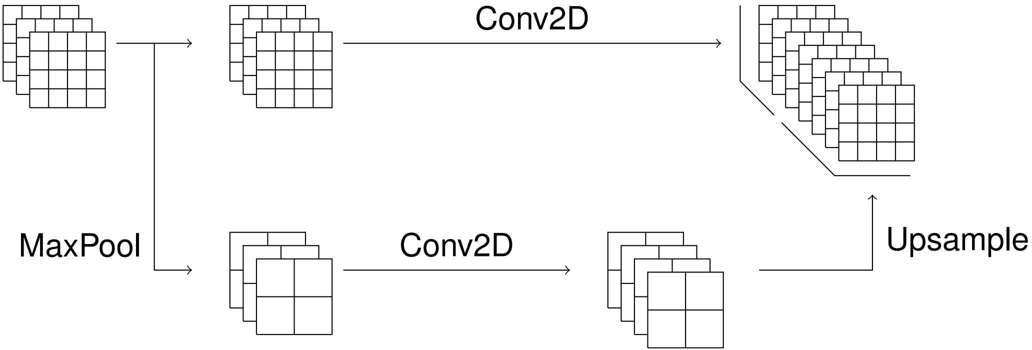


Original



Downsampled and upsampled

Usage



Concerns

- aliasing : lose high frequencies
- loss of information : downsample + upsample $\neq$ original

Table of Contents

Why use Dropout

- Form of regularization
  $\rightarrow$ force net to learn redundancies

- Protects against overfitting
  $\rightarrow$ better generalization

What is Dropout?

1. Before training, fix probability p
2. Each epoch, take $z_i \sim \text{Ber}(p)$ for each output channel i
3. If $z_i = 1$, update channel i during this epoch
4. Else, ignore channel i during this epoch

What is Dropout?

$z_i \sim \text{Ber}(p)$ for each channel i in layer output

$Z = \text{diag}(z_i)$

$$x \mapsto Z\vec{\sigma}(Wx + b)$$

Notes

- Each epoch, a random subset of channels is updated
- At evaluation, all channels are used
- Drop channels, not individual weights
  - Feedforward NN : drop neurons
  - Convolutional NN : drop channels

Changes to training

- Requires more steps of gradient descent
- Fewer parameters to update at every step

  $\rightarrow$ empirically trains faster

- No extra parameters to learn

Table of Contents

Batch Normalization

- combat vanishing and exploding gradients
- assume each layer maintains same relative scale + distribution
- fight *covariance shift*

Details

Normalize each channel individually, then rescale

1. μ_B = batch mean
2. σ_B = batch variance
3. Normalize each channel $x \longmapsto \frac{x-\mu_B}{\sigma_B} := \hat{x}$
4. Affine transformation $\hat{x} \longmapsto \gamma\hat{x} + \beta$

Changes to training

- Allows (empirically) use of bigger learning rates
  $\rightarrow$ easier to train

- Backpropagation to update γ, β

Summary

- convolutions
- upsampling and downsampling
- dropout
- batch normalization

- combat model complexity
- improve generalization
- overcome difficulties in training