PSET 3 Part 1 + Neural Nets

Andrey Kurenkov

CS231A

02/14/2023
Midterm

Will grade midterm ASAP - by end of tomorrow or thursday

Project proposal grading is also ASAP

Will skip lecture recap this week
PSET 3

Space carving
Representation Learning
Supervised Monocular Depth Estimation
Unsupervised Monocular Depth Estimation
Space Carving

Objective:

● Implement the process of space carving.

Lectures:

● Active Stereo & Volumetric Stereo
Review: Space Carving

Visual hull:
an upper bound estimate
Review: Space Carving

Silhouette 1
Review: Space Carving

Silhouette 1

object

Silhouette 2
Goal of Space Carving

Silhouette 1

Silhouette 2
Review: Space Carving

Silhouette 1

voxels

Silhouette 2
Review: Space Carving

Silhouette 1

voxels

Silhouette 2
Review: Space Carving

Silhouette 1

voxels

Silhouette 2
Review: Space Carving

Silhouette 1

voxels
Review: Space Carving

Image 1

voxels
Review: Space Carving

Silhouette 1

voxels

Silhouette 2
Review: Space Carving

Silhouette 1

voxels

object

Silhouette 2
Space carving - overview

Steps:
- Estimate silhouettes of images (could be based on some heuristics, e.g. color)
- Form the initial voxels as a cuboid
- Iterate over cameras and remove the voxels which project to the dark part of each silhouette
Space carving - (a) (b) (c)

Steps:
- Estimate silhouettes of images (could be based on some heuristics, e.g. color)
- Form the initial voxels as a cuboid
  - You may find these functions useful: np.meshgrid, np.repeat, np.tile
  - Also boolean indexing, ie keep = (x>=0) & (x<=w) & (y>=0) & (y<=h)
  - keep = [idx for idx, val in enumerate(keep) if val]
  - x = x[keep]
  - y = y[keep]
- Iterate over cameras and remove the voxels which project to the dark part of each silhouette
  - Question: What will the voxels look like after the first, second, … iteration?
Space carving - (a) (b) (c)

Steps:
- Iterate over cameras and remove the voxels which project to the dark part of each silhouette
  - Question: What will the voxels look like after the first, second, … iteration?
Space carving - (d)

What if we first find the rough size of the object instead of just looking at camera positions?
Space carving - (e)

Steps:
- Estimate silhouettes of images (could be based on some heuristics, e.g. color)
  - Problem: The quality of silhouettes is not perfect.
  - The silhouette from each camera is not perfect, but the result is ok. Why?
  - Experiment: Use only a few of the silhouettes.
PSET 3 - Colab

Need colab for parts 2, 3, and 4.
Intro to Neural Networks

- Background and Applications
- Fully-connected Neural Networks (MLP)
- Convolutional Neural Networks (CNN)
- Backpropagation Algorithm
- PyTorch Example
Background

History

• 1957: Frank Rosenblatt designs the Mark I Perceptron, an early learning-based computer
Background

History

- 1957: Frank Rosenblatt designs the Mark I Perceptron, an early learning-based computer

- 1969: Multi-layer perceptron (early fully-connected neural networks) by Minksy and Papert

Tuning hyperparameters used to take a lot longer in Rosenblatt’s day
Background

History

• 1957: Frank Rosenblatt designs the Mark I Perceptron, an early learning-based computer

• 1969: Multi-layer perceptron (early fully-connected neural networks) by Minsky and Papert

• 1986: Rumelhart, Hinton, and Williams (and others) develop the backpropagation algorithm (BP)
Background

History

• 1957: Frank Rosenblatt designs the Mark I Perceptron, an early learning-based computer

• 1969: Multi-layer perceptron (early fully-connected neural networks) by Minsky and Papert

• 1986: Rumelhart, Hinton, and Williams (and others) develop the backpropagation algorithm (BP)

• 1989: LeCun et al. develop BP for Convolutional Neural Networks (CNNs), and introduce MNIST dataset
Background

History

• 1957: Frank Rosenblatt designs the Mark I Perceptron, an early learning-based computer

• 1969: Multi-layer perceptron (early fully-connected neural networks) by Minsky and Papert

• 1986: Rumelhart, Hinton, and Williams (and others) develop the backpropagation algorithm (BP)

• 1989: LeCun et al. develop BP for Convolutional Neural Networks (CNNs), and introduce MNIST dataset

• 2012: AlexNet uses GPUs to train CNNs fast enough to be practical
A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]
Applications: Convolutional Networks

Detection

Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]
Applications: Convolutional Networks

Detection

Segmentation

Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]


[Farabet et al., 2012]
DeepFace (Face Verification)

Two-Stream Convolutional Networks for Action Recognition in Videos

Activations of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.

[Taigman et al. 2014]
DeepFace (Face Verification)

[Taigman et al. 2014]

Two-Stream Convolutional Networks for Action Recognition in Videos

[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

Activation of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.

Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.
Dense Captioning
[Johnson et al. 2016]
Dense Captioning
[Johnson et al. 2016]

Visualizing Circuits
[Voss et al. 2021]
Background

Signal Relay

Starting from V1 primary visual cortex, visual signal is transmitted upwards, forming a more complex and abstract representation at every level. 

Fully-Connected Neural Networks
Fully-Connected Neural Networks

Components
Fully-Connected Neural Networks

Components

- A single input layer, $h_0 \in \mathbb{R}^n$
Fully-Connected Neural Networks

Components

• A single input layer, $h_0 \in \mathbb{R}^n$

• $k$- hidden layers, $a_i \in \mathbb{R}^{d_i}$

• Weight matrices, $W_i \in \mathbb{R}^{d_{i-1} \times d_i}$

• Bias vectors, $b_i \in \mathbb{R}^{d_i}$
Fully-Connected Neural Networks

Components

- A single input layer, $h_0 \in \mathbb{R}^n$
- $k$- hidden layers, $a_i \in \mathbb{R}^{d_i}$
  - Weight matrices, $W_i \in \mathbb{R}^{d_{i-1} \times d_i}$
  - Bias vectors, $b_i \in \mathbb{R}^{d_i}$
- Output layer, $\hat{y} \in \mathbb{R}^m$
Fully-Connected Neural Networks

Components

- A single input layer, $h_0 \in \mathbb{R}^n$
- $k$- hidden layers, $a_i \in \mathbb{R}^{d_i}$
  - Weight matrices, $W_i \in \mathbb{R}^{d_{i-1} \times d_i}$
  - Bias vectors, $b_i \in \mathbb{R}^{d_i}$
- Output layer, $\hat{y} \in \mathbb{R}^m$

- For each layer, $a_i = f(z_i) = f(W_i h_i + b_i)$, where $f$ is an activation function
Fully-Connected Neural Networks

Components

• A single input layer, \( h_0 \in \mathbb{R}^n \)

• \( k \)- hidden layers, \( a_i \in \mathbb{R}^{d_i} \)
  
  • Weight matrices, \( W_i \in \mathbb{R}^{d_{i-1} \times d_i} \)
  
  • Bias vectors, \( b_i \in \mathbb{R}^{d_i} \)

• Output layer, \( y \hat{=} \in \mathbb{R}^m \)

• For each layer, \( a_i = f(z_i) = f(W_i h_i + b_i) \), where \( f \) is an activation function
  
  • Series of stacked layers compose multiple function together (e.g. \( (f \circ g)(x) \))
Fully-Connected Neural Networks
Cost Function
Fully-Connected Neural Networks

Cost Function

• To train parameters, compute a cost associated with every predicted/labeled output pair, $y, \hat{y}$. 
Fully-Connected Neural Networks

Cost Function

• To train parameters, compute a cost associated with every predicted/labeled output pair, $y$, $y$.

• Requirements: can be averaged over a batch, can be computed with outputs from network
Fully-Connected Neural Networks

Cost Function

• To train parameters, compute a cost associated with every predicted/labeled output pair, $y, \hat{y}$.

• Requirements: can be averaged over a batch, can be computed with outputs from network

• Common loss functions:
  • Least squares (quadratic):
    \[
    \frac{1}{2m} \sum_{i=1}^{m} \| y_i - \hat{y}_i \|^2
    \]
  • Binary Cross-Entropy:
    \[
    y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})
    \]
  • Cross entropy (classification, $y_j$ is one-hot encoding at $j$):
    \[
    \sum_{i=1}^{m} y_i \log(\hat{y}_i)
    \]
Fully-Connected Neural Network

Example

\[ a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1) \]
\[ a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2) \]
\[ a_3 = f(W_{31}x_1 + W_{32}x_2 + W_{33}x_3 + b_3) \]

Sigmoid (logit) transform. \( \sigma(z) = \frac{1}{1+e^{-z}} \)

Hyperbolic tangent (tanh). \( \tanh(z) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \)

Rectified Linear Unit (ReLU). \( \text{ReLU}(z) = \max(0, z) \)
Convolutional Neural Networks

Introduction

• For computer vision applications, convolutional networks are used to learn feature detectors from images

• Advantages:
  • Images are high-dimensional data, fully connected layers would require too many parameters to tune
  • Convolution operations preserve spatial structure of data
  • Convolution operation can be computed efficiently on GPUs (using CUDA)

• Analogues:
  • Inputs/activations are “what” the network “sees”
  • Weights are “how” the network computes one layer from the previous one (feature-detection)
  • As architectures become more complex, interpretability of these learned features becomes more difficult
Convolutional Neural Networks

Components
Convolutional Neural Networks

Components

• Each convolutional “layer” is represented by a 3D tensor of shape
  
  \[ [h \times w \times n_{channels}] \]
Convolutional Neural Networks

Components

• Each convolutional “layer” is represented by a 3D tensor of shape
  \[ h \times w \times n_{\text{channels}} \]

• Between two convolutional layers, the weights are of the shape
  \[ \text{relative x-position, relative y-position, input channels, output channels} \]
Convolutional Neural Networks

Components

• Each convolutional “layer” is represented by a 3D tensor of shape
  \[ h \times w \times n_{\text{channels}} \]

• Between two convolutional layers, the weights are of the shape
  \[ \text{relative x-position, relative y-position, input channels, output channels} \]

• “Convolve” operation consists of 4 hyperparameters:
Convolutional Neural Networks

Components

• Each convolutional “layer” is represented by a 3D tensor of shape $[h \times w \times n_{\text{channels}}]$

• Between two convolutional layers, the weights are of the shape $[\text{relative x-position, relative y-position, input channels, output channels}]$

• “Convolve” operation consists of 4 hyperparameters:
  
  • Number of filters, or depth (each channel also called an “activation map”)
Convolutional Neural Networks

Components

• Each convolutional “layer” is represented by a 3D tensor of shape
  \[ h \times w \times n_{\text{channels}} \]

• Between two convolutional layers, the weights are of the shape
  \[ \text{relative x-position, relative y-position, input channels, output channels} \]

• “Convolve” operation consists of 4 hyperparameters:
  
  • Number of filters, or *depth* (each channel also called
    an “activation map”)
Convolutional Neural Networks

Components

• Each convolutional “layer” is represented by a 3D tensor of shape $[h \times w \times n_{channels}]$. 

• Between two convolutional layers, the weights are of the shape $[\text{relative x-position}, \text{relative y-position}, \text{input channels}, \text{output channels}]$. 

• “Convolve” operation consists of 4 hyperparameters:
  
  • Number of filters, or depth (each channel also called an “activation map”)
  
  • Spatial extent, or receptive field
Convolutional Neural Networks

Components

• Each convolutional “layer” is represented by a 3D tensor of shape
  \([h \times w \times n_{\text{channels}}]\)

• Between two convolutional layers, the weights are of the shape
  \([\text{relative x-position}, \text{relative y-position}, \text{input channels}, \text{output channel}]\)

• “Convolve” operation consists of 4 hyperparameters:

  • Number of filters, or depth (each channel also called an “activation map”)
  
  • Spatial extent, or receptive field

  • The stride
Convolutional Neural Networks

Components

- Each convolutional “layer” is represented by a 3D tensor of shape
  \[ h \times w \times n_{\text{channels}} \]

- Between two convolutional layers, the weights are of the shape
  \[ \text{relative x-position, relative y-position, input channels, output channels} \]

- “Convolve” operation consists of 4 hyperparameters:
  - Number of filters, or depth (each channel also called an “activation map”)
  - Spatial extent, or receptive field
  - The stride
  - Amount of zero-padding
Convolutional Neural Networks

Components

- Each convolutional “layer” is represented by a 3D tensor of shape \([h \times w \times n_{\text{channels}}]\).

- Between two convolutional layers, the weights are of the shape \([\text{relative } x\text{-position, relative } y\text{-position, input channels, output channel}]\).

- “Convolve” operation consists of 4 hyperparameters:
  - Number of filters, or \textit{depth} (each channel also called an “activation map”)
  - \textit{Spatial extent}, or \textit{receptive field}
  - The \textit{stride}
  - Amount of zero-padding
Convolutional Neural Networks

Components

- Each convolutional “layer” is represented by a 3D tensor of shape $[h \times w \times n_{\text{channels}}]$

- Between two convolutional layers, the weights are of the shape $[\text{relative x-position, relative y-position, input channels, output channels}]$

- “Convolve” operation consists of 4 hyperparameters:
  - Number of filters, or depth (each channel also called an “activation map”)
  - Spatial extent, or receptive field
  - The stride
  - Amount of zero-padding

- With this, the shape of layer convolved from layer $-1$ is:
  - $[(W - F + 2P)/S + 1, (H - F + 2P)/S + 1, K]$
Convolutional Neural Networks

Components

• Each convolutional “layer” is represented by a 3D tensor of shape $[h \times w \times n_{\text{channels}}]$

• Between two convolutional layers, the weights are of the shape [relative x-position, relative y-position, input channels, output channels]

• “Convolve” operation consists of 4 hyperparameters:
  • Number of filters, or depth (each channel also called an “activation map”)
  • Spatial extent, or receptive field
  • The stride
  • Amount of zero-padding

• With this, the shape of layer convolved from layer $-1$ is:
  • $[(W - F + 2P)/S + 1, (H - F + 2P)/S + 1, K]$
Convolutional Neural Networks

Components

• Each convolutional "layer" is represented by a 3D tensor of shape 
\([h \times w \times n_{\text{channels}}]\)

• Between two convolutional layers, the weights are of the shape \([\text{relative x-position, relative y-position, input channels, output channels}]\)

• "Convolve" operation consists of 4 hyperparameters:
  • Number of filters, or depth (each channel also called an "activation map")
  • Spatial extent, or receptive field
  • The stride
  • Amount of zero-padding

• With this, the shape of layer convolved from layer \(-1\) is:
  • \([ (W - F + 2P)/S + 1, (H - F + 2P)/S + 1, K ]\)
Convolutional Neural Networks

We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)
### Convolutional Neural Networks

#### Pooling and FC layers

- **Max and Average (L2-norm) pooling:**
  - Downsampling operation to reduce width x height (but not depth) of a layer

- **Fully-connected (FC) layers:**
  - Flattens entire input volume to a vector, and treats like a normal FC network layer
Fin

Questions?