Pixel Labeling

EECS 442 – David Fouhey
Winter 2023, University of Michigan

http://web.eecs.umich.edu/~fouhey/teaching/EECS442_W23/
Administrivia

- HW1/HW2 Logistic Snafus – we’ll fix
- Project matching
- Project proposal – purely meant to help
- Info on midterm up; will have review sessions announced later this week.
Recap
Convolutional Neural Network (CNN)
Function of the image that is parameterized by the convolutional filter weights and biases. We design the form of the function and fit the parameters to data.
Training a CNN

• Download a big dataset
• Initialize network weights randomly
• for epoch in range(epochs):
  • Shuffle dataset
  • for each minibatch in dataset:
    • Put data on GPU
    • Compute gradient with respect to loss
    • Update gradient with SGD
Training a CNN from Scratch

Need to start \( \mathbf{w} \) somewhere

- AlexNet: weights \( \sim \) Normal(0,0.01), bias = 1
- “Xavier” initialization: Uniform\( \left( -\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right) \) where \( n \) is the number of neurons
- “Kaiming” initialization: Normal\( (0, \sqrt{2/n}) \)

Take-home: important, but use defaults
Training a ConvNet

- Convnets typically have millions of parameters:
  - AlexNet: 62 million
  - VGG16: 138 million
  - ConvNeXt-L: 198M
- Convnets typically fit on ~1.2 million images
- Remember least squares: if we have fewer data points than parameters, we’re in trouble
- Solution: need regularization / more data
Training a CNN – Weight Decay

SGD Update

\[ w_{t+1} = w_t - \epsilon \frac{\partial L}{\partial w_t} \]

+ Weight Decay

\[ w_{t+1} = w_t - \eta \epsilon w_t \times \epsilon \frac{\partial L}{\partial w_t} \]

What does this remind you of?

Weight decay is similar to regularization but is not be the same for more complex optimization techniques.

See “Decoupled Weight Decay Regularization”, Loshchilov and Hutter.
Quick Quiz

Raise your hand if it’s a hippo

Horizontal Flip   Color Jitter   Image Cropping
Training a CNN – Augmentation

• Apply transformations that don’t affect the output
• Produces more data but you have to be careful that it doesn’t change the meaning of the output
Training a CNN – Fine-tuning

• What if you don’t have data?
Fine-Tuning: Pre-trained Features

1. Extract some layer from an existing network
2. Use as your new feature.
3. Learn a linear model.
   Surprisingly effective

Convolutions that extract a 1x1x4096 feature (*Fixed/Frozen/Locked*)

\[ Wx + b \]
Fine-Tuning: Transfer Learning

• Rather than initialize from random weights, initialize from some “pre-trained” model that does something else.
• Most common model is trained on ImageNet.
• Other pretraining tasks exist but are less popular.
Fine-Tuning: Transfer Learning

Why should this work?
Transferring from objects (dog) to scenes (waterfall)

Recommendations

- <10K images: features
- **Always** try fine-tuning
- >100K images: consider trying from scratch
Convert HxW image into a F-dimensional vector

Is this image a cat?
At what distance was this photo taken?
Is this image fake?
Convert HxW image into a F-dimensional vector

Which pixels in this image are a cat?
How far is each pixel away from the camera?
Which pixels of this image are fake?
Semantic Segmentation

Today’s Running Example

• Predict F-dimensional vector representing probability of each of F classes at every pixel
• Loss computed/backprop’d at every pixel.
Semantic Segmentation

Each pixel has label, inc. **background**, and **unknown**

Usually visualized by colors.

Note: don’t distinguish between object **instances**

Semantic Segmentation

“Semantic”: a usually meaningless word and an indication that someone is trying to trick you. Meant to indicate here that we’re naming things.
Semantic Segmentation

F-way classification loss function \( - \log \frac{\exp((Wx)_{yi}}{\sum_k \exp((Wx)_k))} \) at every pixel:

Other Tasks – Depth Prediction

Instead: give label of depthmap, train network to do regression (e.g., $\|z_i - \hat{z}_i\|$ where $z_i$ is the ground-truth and $\hat{z}_i$ the prediction of the network at pixel $i$).

- **Input HxWx3 RGB Image**
- **Output HxWx1 Depth Image**
- **True HxWx1 Depth Image**

Result credit: Eigen and Fergus, ICCV 2015
Other Tasks – Surface Normals

\[ n = [n_x, n_y, n_z], \|n\| = 1 \]

Color Image  Normals

Image credit: NYU Dataset, Silberman et al. ECCV 2012
Surface Normals

Instead: train normal network to minimize $\|n_i - \hat{n}_i\|$ where $n_i$ is ground-truth and $\hat{n}_i$ prediction at pixel $i$.

Input: HxWx3 RGB Image

Output: HxWx3 Normals

Result credit: X. Wang, D. Fouhey, A. Gupta, Designing Deep Networks for Surface Normal Estimation. CVPR 2014
Other Tasks – Human Pose Estimation


Source: https://www.youtube.com/watch?v=2DiQUX11YaY
Sun

Other Task – Edges to Cats

Train network to minimize $\|I_j - \hat{I}_j\|$ where $I_j$ is GT and $\hat{I}_j$ prediction at pixel j (plus other magic).

Input: HxWx1 Sketch Image

Output: HxWx3 Image

https://affinelayer.com/pixsrv/

Other Task – Edges to Cats

Train network to minimize $\|I_j - \hat{I}_j\|$ where $I_j$ is GT and $\hat{I}_j$ prediction at pixel $j$ (plus other magic).

Input: HxWx1 Sketch Image  
Output: HxWx3 Image


https://affinelayer.com/pixsrv/
Next Time

I’ll have some extra time next time. Should I cover:

1. How to learn features from the data itself?
2. How to use computer vision to build tools for basic science?
Why Is This Task Hard?
Why Is This Task Hard?

What’s this? (No Cheating!)

(1) Keyboard?  (3) Old cell phone?
(2) Hammer?    (4) Xbox controller?

Image credit: COCO dataset
Why Is This Task Hard?

Image credit: COCO dataset
First – Two “Wrong” Ways

• It’s helpful to see two “wrong” ways to do this.
Why Not Stack Convolutions?

\[ n \times 3 \times 3 \text{ convs have a receptive field of } 2n+1 \text{ pixels} \]

How many convolutions until \( \geq 200 \) pixels? 100
Why Not Stack Convolutions?

Suppose 200 3x3 filters/layer, H=W=400
Storage/layer/image: 200 * 400 * 400 * 4 bytes = 122MB

Uh oh!*  
*100 layers, batch size of 20 = 238GB of memory!
If Memory’s the Issue…

Crop out every sub-window and predict the label in the middle.

Image credit: PASCAL VOC, Everingham et al.
If Memory’s the Issue…

Meet “Gabor”. We extract N×N patches and do independent CNNs. **How many times does Gabor filter the red pixel?**

Answer: 
\[(2n-1)\times(2n-1)\]

Gabor’s looking for a better job with a smarter boss.

Image credit: PASCAL VOC, Everingham et al.
The Big Issue

We need to:

1. Have large receptive fields to figure out what we’re looking at

2. Not waste a ton of time or memory while doing so

These two objectives are in total conflict
Encoder-Decoder

Key idea: First **downsample** towards middle of network. Then **upsample** from middle.

How do we downsample?
Convolutions, pooling
Where Do We Get Parameters?

Convnet that maps images to vectors

Recall that we can rewrite any vector-vector operations via 1x1 convolutions

Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014
Where Do We Get Parameters?

Convnet that maps images to vectors

Convnet that maps images to images

What if we make the input bigger?
Where Do We Get Parameters?

Convnet that maps images to vectors

Convnet that maps images to images

Since it’s convolution, can reuse an image network
How Do We Upsample?

Do the opposite of how we downsample:
1. Pooling → “Unpooling”
2. Convolution → “Transpose Convolution”
Recall: Pooling

Max-pool 2x2 Filter Stride 2
Now: Unpooling

Nearest Neighbor Unpool x2
Recall: Convolution

3x3 Convolution, Stride 2, Pad 1

Dot product between filter $f$ and input

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung
Recall: Convolution

3x3 Convolution, Stride 2, Pad 1

Dot product between filter $f$ and input

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung
Recall: Convolution

3x3 Convolution, Stride 2, Pad 1

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung
Transpose Convolution

**Convolution**

Filter: little lens that looks at a pixel.

**Transpose Conv.**

Filter: tiles used to make an image.

Image credit: ifixit.com, thespruce.com
Transpose Convolution

3x3 Transpose Convolution, Stride 2, Pad 1

Output is filter $F$ weighted by input $I$

$I_{11}F \quad I_{12}F$

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung
 transpose convolution

3x3 transpose convolution, stride 2, pad 1

Output is filter \( F \) weighted by input \( I \)

Sum outputs at overlap (e.g., from \( I_{11}F \) and \( I_{21}F \))

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung
Putting it Together

Convolutions + pooling downsample/compress/encode
Transpose convs./unpoolings upsample/uncompress/decode
Putting It Together – Block Sizes

• Networks come in lots of forms
• **Don’t take any block sizes literally.**
• Often (not always) keep some spatial resolution

Encode to spatially smaller tensor, then decode.

Encode to 1D vector then decode.
Putting It Together – Block Sizes

- Often multiple layers at each spatial resolution.
- Often halve spatial resolution and double feature depth every few layers.

```
H     H/2    H/4    H/8    H/4    H/2    H
W     W/2    W/4    W/8    W/4    W/2    W
D     2D     4D     8D     4D     2D     D
```
An Aside: Autoencoders

Network compresses input to “bottleneck”, decodes it back to input.

Objective:

$\|D(E(X)) - X\|$
Walking the Latent Space*

Interpolation in Latent Space

*In the interest of honesty in advertising: not an autoencoder, but a similar method with the same goal of learning a latent space

Missing Details

While the output is HxW, just upsampling often produces results without details/not aligned with the image.

Why?

Information about details lost when downsampling!

Result from Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014
Missing Details

Where is the useful information about the high-frequency details of the image?

Result from Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014
Missing Details

How do you send details forward in the network?

You copy the activations forward.

Subsequent layers at the same resolution figure out how to fuse things.

Result from Long et al. *Fully Convolutional Networks For Semantic Segmentation*. CVPR 2014

Extremely popular architecture, was originally used for biomedical image segmentation.
Evaluating Pixel Labels

Input Image

Predicted Classes

How do we convert final HxWxF into labels?

argmax over labels
Evaluating Semantic Segmentation

Given predictions, how well did we do?

Input  Prediction ($\hat{y}$)  Ground-Truth ($y$)
Evaluating Semantic Segmentation

Prediction and ground-truth are images where each pixel is one of F classes.

Accuracy: \( \text{mean}(\hat{y} = y) \)

Intersection over union, averaged over classes

Prediction \((\hat{y})\)  
Ground-Truth \((y)\)
Next Time

- Detecting Objects (drawing boxes around them)
- Plus whatever we voted on
More Info
Why “Transpose Convolution”?

Can write convolution as matrix-multiply

Input: 4, Filter: 3, Stride: 1, Pad: 1

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung
Why “Transpose Convolution”?  

Transpose convolution is convolution transposed

Example Credit: L. Fei-Fei, J. Johnson, S. Yeung