ConvNet for image-to-image tasks

INPUT

OUTPUT

CS194: Computer Vision and Comp. Photo
Alexei Efros, UC Berkeley, Fall 2022

Many slides from Herr Prof. David Fouhey
Regression Objective

This is what you implement in part 1. But this is not what the SoTA models do in practice.
Downsides of regression objective

Locally a lot of things look similar!!

With regression objective you have to commit to ONE location and only get one training signal on how correct that location was.
Regression Loss (e.g. Part 1)

\[
L_2(\hat{Y}, Y) = \frac{1}{2} \sum_{h,w} \| Y_{h,w} - \hat{Y}_{h,w} \|^2
\]

Keypoint #1 X
Keypoint #1 Y
Keypoint #2 X
Classification Loss (e.g. ImageNet)

Slide by David Fouhey
Loss function for classification

Network output

- dolphin
- cat
- grizzly bear
- angel fish
- chameleon
- **clown fish**
- iguana
- elephant

Ground truth label

- "clown fish"

Loss → error

Slide by Philip Isola
Loss function for classification

Network output

dolphin

cat

grizzly bear

angel fish

chameleon

clown fish

iguana

elephant

Ground truth label

“clown fish”

Loss → small

Slide by Philip Isola
Loss function for classification

Network output
- dolphin
- cat
- grizzly bear
- angel fish
- chameleon
- clown fish
- iguana
- elephant

Ground truth label
- “grizzly bear”

Loss \rightarrow \text{large}

Slide by Philip Isola
Loss function for classification

Network output

<table>
<thead>
<tr>
<th>( \hat{z} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>dolphin</td>
</tr>
<tr>
<td>cat</td>
</tr>
<tr>
<td>grizzly bear</td>
</tr>
<tr>
<td>angel fish</td>
</tr>
<tr>
<td>chameleon</td>
</tr>
<tr>
<td>clown fish</td>
</tr>
<tr>
<td>iguana</td>
</tr>
<tr>
<td>elephant</td>
</tr>
</tbody>
</table>

Ground truth label

<table>
<thead>
<tr>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Cross-entropy Loss:
Probability of the observed data under the model

\[
H(\hat{z}, z) = - \sum_c \hat{z}_c \log z_c
\]

Results in learning a probability model \( p(c|x) \)

Slide by Philip Isola
Belief/Confidence map formulation

For K keypoints, train model to predict K many sheets (h x w) of scores of how likely the pixel is k-th keypoint

Problem

So far, we’ve only seen examples that output a vector representation out of an image.

How do we do dense (per-pixel) predictions?
So Far…

Convert HxW image into a F-dimensional vector

Is this image a cat?
At what distance was this photo taken?
Is this image fake?

Slide by David Fouhey
Pixel Labeling

Convert $H \times W$ 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?
e.g. 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

Slide by David Fouhey
Surface Normals

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

Color Image

Room

Legend

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

Slide by David Fouhey
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


Slide by David Fouhey
“Semantic Segmentation”

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

Usually visualized by colors.

Note: don’t distinguish between object *instances*


Slide by David Fouhey
“Semantic Segmentation”

“Semantic”: a usually meaningless word. Meant to indicate here that we’re **naming** things.
Great opensource tool, builds on convolutional pose machine architecture, adapted to multiple people.
Generic Image-to-Image Translation
First – Two “Wrong” Ways

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

$n$ 3x3 convs have a receptive field of $2n+1$ 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!

Slide by David Fouhey
Idea #2

Crop out every sub-window and predict the label in the middle.
Idea #2

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

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

Image credit: PASCAL VOC, Everingham et al.

Slide by David Fouhey
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

Slide by David Fouhey
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

Slide by David Fouhey
How to upsample with convnets?
Simple solution

- Upsample, followed by a regular Convolution

Input: $B \times C_{in} \times H \times W$

After upsample, factor 2: $B \times C_{in} \times 2H \times 2W$

After conv: $B \times C_{out} \times 2H \times 2W$
# How to Upsample

<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bed of Nails</strong></td>
<td>$C \times 2 \times 2$</td>
<td>$C \times 4 \times 4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Nearest Neighbor</strong></td>
<td>$C \times 2 \times 2$</td>
<td>$C \times 4 \times 4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bilinear Interpolation</strong></td>
<td>$C \times 2 \times 2$</td>
<td>$C \times 4 \times 4$</td>
</tr>
</tbody>
</table>
Recall from Morphing Lecture: Inverse warping

Don’t splat! Do inverse warping.
You know this from project 3
Recall: Bilinear Interpolation

http://en.wikipedia.org/wiki/Bilinear_interpolation
Help interp2
Putting it Together

Convolutions + pooling downsample/compress/encode
Transpose convs./unpoolings upsample/uncompress/decode

Slide by David Fouhey
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.
Application to pose detection:
Predict heat maps

Target: $K+1 \times H \times W$ Gaussian around $(x,y)$ for $k$-th keypoint in the $k$-th channel

$K+1$ for $K$ parts + background
You will implement this in project 5!

**L2 Training Loss**

- L2 loss on the target heatmap (peaky gaussian around the gt keypoint)

\[ L = \sum_{k=1}^{K+1} \sum_{(x,y)} \|b^k(x, y) - b^k_*(x, y)\| \]

Target “belief map” : 
K+1 x H x W
Gaussian around (x,y) for k-th keypoint in the k-th channel
You will implement this in project 5!

Log Loss Training Loss

- Log loss (or cross entropy loss) on the target heatmap probabilities
- The target must also sum to 1
- Mask RCNN just uses 1 at the target, 0 everywhere else.
- Experiment

Target “belief map” :  
K+1 \times H \times W  
1 at Ground truth location (x,y) for k-th keypoint in the k-th channel
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

Slide by David Fouhey
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

Slide by David Fouhey
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.

Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI 2015"
U-Net improves performance

Input Image → CNN → Predicted Classes
Grayscale image: $L$ channel

$X \in \mathbb{R}^{H \times W \times 1}$

Color information: $ab$ channels

$\hat{Y} \in \mathbb{R}^{H \times W \times 2}$

Grayscale image: $L$ channel

$X \in \mathbb{R}^{H \times W \times 1}$

Concateenate $(L, ab)$ channels

$\left( X, \hat{Y} \right)$

Regressing to pixel values doesn’t work 😞

Input

Output

Ground truth

\[ L_2(\hat{Y}, Y) = \frac{1}{2} \sum_{h,w} \| Y_{h,w} - \hat{Y}_{h,w} \|^2 \]

Slide by Richard Zhang
\[ L_2(\hat{Y}, Y) = \frac{1}{2} \sum_{h,w} \| Y_{h,w} - \hat{Y}_{h,w} \|_2^2 \]
Better Loss Function

\[ \theta^* = \arg \min_{\theta} \ell(\mathcal{F}_{\theta}(X), Y) \]

- Regression with L2 loss inadequate
  \[ L_2(\hat{Y}, Y) = \frac{1}{2} \sum_{h,w} \| Y_{h,w} - \hat{Y}_{h,w} \|_2^2 \]
- Use per-pixel multinomial classification
  \[ L(\hat{Z}, Z) = -\frac{1}{HW} \sum_{h,w} \sum_{q} z_{h,w,q} \log(\hat{Z}_{h,w,q}) \]
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Slide by Richard Zhang
Designing pixel loss functions

Color distribution cross-entropy loss with colorfullness enhancing term.

[Zhang, Isola, Efros, ECCV 2016]
Designing pixel loss functions

Image colorization

Super-resolution

Cross entropy loss, with colorfulness term

“semantic feature loss” (VGG feature covariance matching objective)

[Johnson et al. 2016]

[Zhang et al. 2016]

[Johnson et al. 2016]
Universal loss?
Generative Adversarial Network (GANs)

Generated vs Real (classifier)

[Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio 2014]
Generator

[Goodfellow et al., 2014]
\( \mathbf{G} \) tries to synthesize fake images that fool \( \mathbf{D} \).

\( \mathbf{D} \) tries to identify the fakes.

[Goodfellow et al., 2014]
\[
\text{arg max}_D \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]
\]

[Goodfellow et al., 2014]
\textbf{G} tries to synthesize fake images that \textit{fool} \textbf{D}:

\[
\arg\min_G \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]
\]

[Goodfellow et al., 2014]
G tries to synthesize fake images that fool the best D:

$$\arg\min_G \max_D \mathbb{E}_{x,y}[ \log D(G(x)) + \log(1 - D(y)) ]$$

[Goodfellow et al., 2014]
**G**'s perspective: **D** is a loss function.

Rather than being hand-designed, it is *learned*.
arg min_G max_D \mathbb{E}_{x, y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]

[Goodfellow et al., 2014]
arg min
\begin{equation*}
\max_{G} \quad \mathbb{E}_{x,y}
\left[
\log D(G(x)) + \log(1 - D(y))
\right]
\end{equation*}

[Goodfellow et al., 2014]
\[
\text{arg min}_{G} \max_{D} \mathbb{E}_{x,y} \left[ \log D(G(x)) + \log(1 - D(y)) \right]
\]

[Goodfellow et al., 2014]
[Isola et al., 2017]
real or fake pair?

arg min_G max_D \mathbb{E}_{x,y}[ \log D(x, G(x)) + \log(1 - D(x, y)) ]

[Goodfellow et al., 2014]
[Isola et al., 2017]
\[
\arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(x, G(x)) + \log(1 - D(x, y)) \right]
\]

[Goodfellow et al., 2014]
[Isola et al., 2017]
\[
\text{arg min}_G \text{ max}_D \quad \mathbb{E}_{x,y} \left[ \log D(x, G(x)) + \log(1 - D(x, y)) \right]
\]

[Goodfellow et al., 2014]
[Isola et al., 2017]
arg \min_G \max_D \mathbb{E}_{x,y} \left[ \log D(x, G(x)) + \log(1 - D(x, y)) \right]

[Goodfellow et al., 2014]
[Isola et al., 2017]
BW → Color

Data from [Russakovsky et al. 2015]
BW $\rightarrow$ Color

Data from [Russakovsky et al. 2015]
Data from [maps.google.com]
Labels → Facades

Data from [Tylecek, 2013]
Labels → Facades

Input

Output

Input

Output

Data from [Tylecek, 2013]
Day → Night

Data from [Laffont et al., 2014]
Thermal $\rightarrow$ RGB

Input  

Ground-truth  

Output
Edges → Images

Edges from [Xie & Tu, 2015]
Sketches $\rightarrow$ Images

Trained on Edges $\rightarrow$ Images

Data from [Eitz, Hays, Alexa, 2012]
Image-to-image translation in PyTorch (e.g., horse2zebra, edges2cats, and more)

- pytorch
- gan
- cyclegan
- pix2pix
- deep-learning
- computer-vision
- computer-graphics
- image-manipulation
- image-generation
- generative-adversarial-network
- gans

- 223 commits
- 3 branches
- 0 releases
- 26 contributors
Scott Eaton (http://www.scott-eaton.com/)
"Do as I Do"
Everybody Dance Now

Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros
UC Berkeley
Results

https://www.youtube.com/watch?v=PCBTZh41Ris&feature=youtu.be
“You need a lot of data if you want to train/use CNNs”
Transfer Learning

“You need a lot of data if you want to train the CNN.”
Deep Features & their Embeddings
The Unreasonable Effectiveness of Deep Features

Classes separate in the deep representations and transfer to many tasks. [DeCAF] [Zeiler-Fergus]
Can be used as a generic feature

(“CNN code” = 4096-D vector before classifier)
ImageNet + Deep Learning

- Image Retrieval
- Detection (RCNN)
- Segmentation (FCN)
- Depth Estimation
- …
ImageNet + Deep Learning

Materials?
Parts?
Geometry?
Boundaries?
Pose?

Beagle
Transfer Learning with CNNs

1. Train on Imagenet
Transfer Learning with CNNs

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier
   i.e. swap the Softmax layer at the end
Transfer Learning with CNNs

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier
   i.e. swap the Softmax layer at the end

3. If you have medium sized dataset, “finetune” instead: use the old weights as initialization, train the full network or only some of the higher layers
   retrain bigger portion of the network, or even all of it.
Transfer Learning with CNNs

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

   i.e. swap the Softmax layer at the end

3. If you have medium sized dataset, “finetune” instead: use the old weights as initialization, train the full network or only some of the higher layers

   retrain bigger portion of the network, or even all of it.

   tip: use only ~1/10th of the original learning rate in finetuning to player, and ~1/100th on intermediate layers