Today:
Phillip Isola: Big data, Deep learning, Predicting pixels

Next time:
Andrew Owens: More applications of deep learning to graphics
140 billion images
6 billion added monthly

72 hours uploaded every minute

3.5 trillion photographs

90% of net traffic will be visual!
Too Big for Humans

Digital Dark Matter

[Perona 2010]
Big Issue

• How can we efficiently make use of all the knowledge in big visual datasets?

• —> deep learning!
Graphics using big data

A motivating example…
[Hays and Efros. Scene Completion Using Millions of Photographs. SIGGRAPH 2007 and CACM October 2008.]
Diffusion Result
Efros and Leung result
Scene Matching for Image Completion
The Algorithm
Scene Matching
Scene Descriptor
Scene Descriptor

Scene Gist Descriptor (Oliva and Torralba 2001)
Scene Descriptor

+ Scene Gist Descriptor

(Oliva and Torralba 2001)
2 Million Flickr Images
... 200 total
Context Matching
Graph cut + Poisson blending
Result Ranking

We assign each of the 200 results a score which is the sum of:

- The scene matching distance
- The context matching distance (color + texture)
- The graph cut cost
... 200 scene matches
Why does it work?
Nearest neighbors from a collection of 20 thousand images
Nearest neighbors from a collection of 2 million images
Parts of our world can be explained by elegant mathematics
– physics, chemistry, astronomy, etc.

But much cannot
– psychology, economics, genetics, etc.

Enter The Data!
– Great advances in several fields:
  • e.g., speech recognition, machine translation
  • Case study: Google
Limitations

• Slow

• Hand-engineered matching function

• Each method only works in special settings (e.g., outdoor scene photos, textures)
Basic idea

Brain/Machine

“clown fish”
Object recognition

Feature extractors

Edges
Texture
Colors

Segments
Parts

“clown fish”

Classifier
Object recognition

Feature extractors
- Edges
- Texture
- Colors

Classifier
- Segments
- Parts

"clown fish"

Learned
Neural network

“clown fish”

Learned
Neural network

Learned

“clown fish”
Deep neural network

Learned

“clown fish”
Computation in a neural net
Computation in a neural net

\[ y_j = \sum_i w_{ij} x_i \]
Computation in a neural net

Rectified linear unit (ReLU)

\[ g(y) = \max(0, y) \]
Convolutional Neural Nets

- Input features
- A bank of 2 filters
- 2-dimensional output features
Convolutional Neural Nets

Convolution

filter

[Diagram of a clownfish and a convolutional filter]
Slide from Andrea Vedaldi
Computation in a neural net

\[ f(x) = f_L(\ldots f_2(f_1(x))) \]
Computation in a neural net

Last layer

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant

... → argmax → “clown fish”
Learning with deep nets

Learned

“clown fish”
Learning with deep nets

Train network to associate the right label with each image

- “clown fish”
- “grizzly bear”
- “chameleon”
"clown fish" → $w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow w_4 \rightarrow w_5 \rightarrow w_6 \rightarrow \text{Loss} \rightarrow L()$

$\arg\min_w L(w_1, \ldots, w_6)$
Loss function

Network output

Ground truth label

“clown fish”

Loss ➔ error
Loss function

Network output

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

Ground truth label

- “clown fish”

Loss → **small**
Loss function

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

Ground truth label
- “grizzly bear”

Loss → large
Loss function for classification

Network output

\[ \hat{z} \]

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

Ground truth label

\[ z \]

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

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 \( p(c|x) \)!
Learning with deep nets

 Learned

 $z_1$

 "clown fish"

 $x_1$

 $w_1$  $w_2$  $w_3$  $w_4$  $w_5$  $w_6$

 $\ell(z_1, f(x_1; w))$
Learning with deep nets

Learned

\( z_2 \)

“grizzly bear”

\( x_2 \)

\[
\ell(z_2, f(x_2; w))
\]
Learning with deep nets

\[ \argmin_w \sum_i \ell(z_i, f(x_i; w)) \]

\[ \ell(z_i, f(x_i; w)) \]

\[ x_i \boxed{\text{"chameleon"}} \]

\[ z_i \boxed{\text{Learned}} \]
Gradient descent

\[
\text{argmin}_w \sum_i \ell(z_i, f(x_i; w)) = L(w)
\]

One iteration of gradient descent:

\[
\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \frac{\partial L(\mathbf{w}^t)}{\partial \mathbf{w}}
\]

learning rate
Gradient descent

$L(w)$
$p(c | x)$
Texture synthesis by non-parametric sampling

Synthesizing a pixel

Input image

non-parametric sampling

Models \( P(p|N(p)) \)

[Efros & Leung 1999]
Texture synthesis with a deep net

[van der Oord et al. 2016]
Texture synthesis with a deep net

[van der Oord et al. 2016]
Sampling

Network output

\[ P(p_i | p_1, \cdots, p_{i-1}) \]
Sampling

Network output

\[ P(p_i | p_1, \cdots, p_{i-1}) \]
Sampling

Network output

\[ P(p_i | p_1, \cdots, p_{i-1}) \]
Sampling

Network output

\[ P(p_i | p_1, \cdots, p_{i-1}) \]
Sampling

Network output

turquoise
blue
green
red
orange
white
gray
black

\[ P(p_i | p_1, \ldots, p_{i-1}) \]
Sampling

Network output

turquoise
blue
green
red
orange
white
gray
black

\[ P(p_i | p_1, \cdots, p_{i-1}) \]
Sampling full images
Predicting pixels gives a probability model of a whole image!

\[ P(p_i|p_1, \ldots, p_{i-1}) \]

\[ P(p) = \prod_{i=1}^{N} P(p_i|p_1, \ldots, p_{i-1}) \]

This factorization is valid for any probability distribution (chain rule).
Ansel Adams, Yosemite Valley Bridge
Grayscale image: $L$ channel

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

$F$

Color information: $ab$ channels

$\hat{Y} \in \mathbb{R}^{H \times W \times 2}$
Grayscale image: $L$ channel

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

Concatenate $(L, ab)$

$(X, \hat{Y})$
Colors in $ab$ space

(continuous)
Colors in $ab$ space
(discrete)
Some of the better results
Failure Cases
Biases
from Reddit /u/SherySantucci
Recolorized by Reddit ColorizeBot
Next time:
Andrew Owens — more deep learning!