94-775 Lecture 10: Introduction to Neural Nets and Deep Learning

George Chen

A few slides are by Phillip Isola
Comments on the Final Project

• Final project presentation times will be randomized
  • Unless your team really, really wants to present next Tuesday

• Minis are short, and we understand that there isn’t that much time to do the project
  • Analysis: prioritize easier things first

• Negative results are fine provided that you’ve correctly put together a well-thought out experiment
Over 10 million images, 1000 object classes

2011: Traditional computer vision achieves accuracy ~74%
2012: Initial deep neural network approach accuracy ~84%
2015 onwards: Deep learning achieves accuracy 96%+

Deep Learning Takeover

Academia:

- Top computer vision conferences (CVPR, ICCV, ECCV) are now nearly all about deep learning
- Top machine learning conferences (ICML, NIPS) have heavily been taken over by deep learning

Extremely useful in practice:

- Near human level image classification (including handwritten digit recognition)
- Near human level speech recognition
- Improvements in machine translation, text-to-speech
- Self-driving cars
- *Better* than humans at playing Go

Heavily dominated by industry now!
Google DeepMind’s AlphaGo vs Lee Sedol, 2016
Is it all hype?
panda  
~58% confidence

+ 0.007 ×

adversarial noise

= 

gibbon  
~99% confidence

Fooling Neural Networks in the Physical World with 3D Adversarial Objects

31 Oct 2017 · 3 min read — shared on Hacker News, Lobsters, Reddit, Twitter

We’ve developed an approach to generate 3D adversarial objects that reliably fool neural networks in the real world, no matter how the objects are looked at.

Neural network based classifiers reach near-human performance in many tasks, and they’re used in high risk, real world systems. Yet, these same neural networks are particularly vulnerable to adversarial examples, carefully perturbed inputs that cause
a cat is sitting on a toilet in a bathroom

Source: Gizmodo article “This Neural Network’s Hilariously Bad Image Descriptions Are Still Advanced AI”. September 16, 2015. (They’re using the NeuralTalk image-to-caption software.)
Another AI Winter?

~1970’s: First Al winter over symbolic AI

~1980’s: Second Al winter over “expert systems”

Every time: Lots of hype, explosion in funding, then bubble bursts
Artificial Intelligence—The Revolution Hasn’t Happened Yet

Artificial Intelligence (AI) is the mantra of the current era. The phrase is intoned by technologists, academicians, journalists and venture capitalists.
What is deep learning?
Basic Idea

Brain/Machine

“clown fish”
Object Recognition

Feature extractors

Edges
Texture
Colors

Segments
Parts

“clown fish”

Classifier

Slide by Phillip Isola
Object Recognition

Feature extractors:
- Edges
- Texture
- Colors

Classifier:
- Segments
- Parts

Learned: "clown fish"

Slide by Phillip Isola
Neural Network

Learned

“clown fish”
Neural Network

Learned

“clown fish”
Deep Neural Network

“clown fish”

Slide by Phillip Isola
Crumpled Paper Analogy

binary classification: 2 crumpled sheets of paper corresponding to the different classes

deepl learning: series (“layers”) of simple unfolding operations to try to disentangle the 2 sheets

Analogy: Francois Chollet, photo: George Chen
Representation Learning

Each layer’s output is another way we could represent the input data.

Learned

“clown fish”
Representation Learning

Each layer’s output is another way we could represent the input data.

Learned

Visualize (e.g., t-SNE)

“clown fish”
Why Does Deep Learning Work?

Actually the ideas behind deep learning are old (~1980’s)

- Big data
- Better hardware
- Better algorithms

Logos of companies like Amazon, Facebook, Netflix, and Google are shown in the diagram.
Structure Present in Data Matters

Neural nets aren’t doing black magic

- **Image analysis:** convolutional neural networks (convnets) neatly incorporates basic image processing structure

- **Time series analysis:** recurrent neural networks (RNNs) incorporates ability to remember and forget things over time

  - Note: text is a time series

  - Note: video is a time series
Handwritten Digit Recognition Example

Walkthrough of building a 1-layer and then a 2-layer neural net
Handwritten Digit Recognition

28x28 image

flatten & treat as 1D vector

length 784 vector (784 input neurons)

weighted sums
(parameterized by a weight matrix $W$ and a bias $b$)

"dense" layer with 10 numbers

activation
(can be thought of as post-processing)

single "dense" layer with 10 neurons

"dense" layer final output
Handwritten Digit Recognition

- Input: length 784 vector (784 input neurons)
- Dense layer: weighted sums parameterized by a weight matrix $W$ and a bias $b$ (1D numpy array with 784 entries)
- “Dense” layer: with 10 numbers (2D numpy array 784-by-10)
- Output: (1D numpy array with 10 entries)
Handwritten Digit Recognition

In 784 vector input neurons

Weighted sums

(parameterized by a weight matrix \( W \) and a bias \( b \))

\[ \text{dense}[j] = \sum_{i=0}^{783} \text{input}[i] \times W[i, j] + b[j] \]

(2D numpy array of dimensions 784-by-10)

(1D numpy array with 10 entries)

input

(84 entries)

dense

(1D numpy array with 10 entries)
Handwritten Digit Recognition

weighted sums
(parameterized by a weight matrix $W$ and a bias $b$)

length 784 vector (784 input neurons)

“dense” layer with 10 numbers
Handwritten Digit Recognition

28x28 image

flatten & treat as 1D vector

length 784 vector (784 input neurons)

weighted sums (parameterized by a weight matrix $W$ and a bias $b$)

“dense” layer with 10 numbers

activation (can be thought of as post-processing)

“dense” layer final output

single “dense” layer with 10 neurons
Handwritten Digit Recognition

Many different activation functions possible

Example: **Rectified linear unit (ReLU)**
zeros out entries that are negative

```python
dense_final = np.maximum(0, dense)
```

“dense” layer with 10 numbers

dense

“dense” layer final output

dense_final

ReLU
(can be thought of as post-processing)
Many different activation functions possible

Example: **softmax** turns the entries in the dense layer (prior to activation) into a probability distribution (using the “softmax” transformation)

```python
dense_exp = np.exp(dense)
dense_exp /= np.sum(dense_exp)
dense_final = dense_exp
```

(softmax) (can be thought of as post-processing)
Handwritten Digit Recognition

28x28 image

- Flatten and treat as 1D vector
- Length 784 vector (784 input neurons)

weighted sums

(parameterized by a weight matrix $W$ and a bias $b$)

softmax

(can be thought of as post-processing)

Pr(digit 0)
Pr(digit 1)
Pr(digit 2)
Pr(digit 3)
Pr(digit 4)
Pr(digit 5)
Pr(digit 6)
Pr(digit 7)
Pr(digit 8)
Pr(digit 9)

"dense" layer

with 10 numbers

single "dense" layer with 10 neurons

"dense" layer final output
Handwritten Digit Recognition

28x28 image

- flatten & treat as 1D vector
- length 784 vector (784 input neurons)

We want the output of the dense layer to encode probabilities for whether the input image is a 0, 1, 2, ..., 9 but as of now we aren’t providing any sort of information to enforce this dense layer with 10 neurons, softmax activation, parameters $W, b$
Handwritten Digit Recognition

Demo part 1
Handwritten Digit Recognition

28x28 image

flattened &
treat as
1D vector

length 784 vector
(784 input neurons)

dense layer with
10 neurons,
softmax activation,
parameters $W, b$
Handwritten Digit Recognition

Training label: 6

28x28 image

Learning this neural net means learning $W$ and $b$

flatten & treat as 1D vector

length 784 vector (784 input neurons)

Loss/"error"

Error is averaged across training examples

Popular loss function for classification (> 2 classes): categorical cross entropy

$\log \frac{1}{\Pr(\text{digit 6})}$

dense layer with 10 neurons, softmax activation, parameters $W, b$
Handwritten Digit Recognition

Demo part 2
Handwritten Digit Recognition

Training label: 6

28x28 image

Learning this neural net means learning $W$ and $b$

flatten & treat as 1D vector

length 784 vector (784 input neurons)

Loss/“error”

Error is averaged across training examples

Popular loss function for classification (> 2 classes): categorical cross entropy

$\log \frac{1}{\Pr(\text{digit 6})}$

dense layer with 10 neurons, softmax activation, parameters $W$, $b$
Handwritten Digit Recognition

Training label: 6

- 28x28 image (784 input neurons)
- length 784 vector

Learning this neural net means learning parameters of both dense layers!

Popular loss function for classification (> 2 classes): categorical cross entropy

\[ \log \frac{1}{\text{Pr}(\text{digit 6})} \]

Error is averaged across training examples.
Handwritten Digit Recognition

Demo part 3
Architecting Neural Nets

• Increasing number of layers (depth) makes neural net more complex
  • Can approximate more functions
  • More parameters needed
    • More training data may be needed

• Designing neural net architectures is a bit of an art
  • How to select the number of neurons for intermediate layers?
  • Very common in practice: modify existing architectures that are known to work well (e.g., VGG-16 for computer vision/image processing)
Deep Learning

• Inspired by biological neural nets but otherwise not the same at all (biological neural nets do not work like deep nets)

• Learns a layered representation
  • Tries to get rid of manual feature engineering
  • Need to design constraints for what features are learned to account for structure in data (e.g., images, text, …)