Today

• Deep Learning
  • History
  • A Few Recent trends

https://qdata.github.io/deep2Read/
Early History

• In 1950 English mathematician Alan Turing wrote a landmark paper titled “Computing Machinery and Intelligence” that asked the question: “Can machines think?”

• Further work came out of a 1956 workshop at Dartmouth sponsored by John McCarthy. In the proposal for that workshop, he coined the phrase a “study of artificial intelligence”

• 1950s
  • Samuel’s checker player: start of machine learning
  • Selfridge’s Pandemonium

• **1952-1969: Enthusiasm**: Lots of work on neural networks

• 1970s-80s: Expert systems, Knowledge bases to add on rule-based inference, Decision Trees, Bayes Nets

• **1990s**: CNN, RNN, ...

• **2000s**: SVM, Kernel machines, Structured learning, Graphical models, semi-supervised, matrix factorization, ...

Adapted From Prof. Raymond J. Mooney’s slides
Deep Learning (CNN) in the 90’s

- Prof. Yann LeCun invented Convolutional Neural Networks (CNN) in 1998
- First NN successfully trained with many layers

Deep Learning (RNN) in the 90’s

• Prof. Schmidhuber invented "Long short-term memory" – Recurrent NN (LSTM-RNN) model in 1997


Image Credits from Christopher Olah
“Winter of Neural Networks” in ~2000s

• Non-convex

• Need a lot of tricks to play with
  • How many layers ?
  • How many hidden units per layer ?
  • What topology among layers ? .......

• Hard to perform theoretical analysis

• Large labeled datasets are rare
ImageNet Challenge

- 2010-11: hand-crafted computer vision pipelines
- 2012-2016: ConvNets
  - 2012: AlexNet
    - major deep learning success
  - 2013: ZFNet
    - improvements over AlexNet
  - 2014
    - VGGNet: deeper, simpler
    - InceptionNet: deeper, faster
  - 2015
    - ResNet: even deeper
  - 2016
    - ensembled networks
  - 2017
    - Squeeze and Excitation Network

Adapt from From NIPS 2017 DL Trend Tutorial
Why breakthrough?

10 Breakthrough Technologies
2013
Think of the most frustrating, intractable, or simply annoying problems you can imagine. Now think about what technology is doing to fix them. That’s what we did in coming up with our annual list of 10 Breakthrough Technologies. We’re looking for technologies that we believe will expand the scope of human possibilities.

Deep Learning

10 Breakthrough Technologies
2017
These technologies all have staying power. They will affect the economy and our politics, improve medicine, or influence our culture. Some are unfolding now; others will take a decade or more to develop. But you should know about all of them right now.

Deep Reinforcement Learning

10 BREAKTHROUGH TECHNOLOGIES
2018
Generative Adversarial Network (GAN)
DNNs help us build more intelligent computers

- **Perceive the world,**
  - e.g., objective recognition, speech recognition, ...
- **Understand the world,**
  - e.g., machine translation, text semantic understanding
- **Interact with the world,**
  - e.g., AlphaGo, AlphaZero, self-driving cars, ...
- **Being able to think / reason,**
  - e.g., learn to code programs, learn to search deepNN, ...
- **Being able to imagine / to make analogy,**
  - e.g., learn to draw with styles, ......
Some Recent Trends

- **1. CNN / Residual / Dynamic parameter**
- **2. RNN / Attention / Seq2Seq / BERT ...**
- **3. Neural Architecture with explicit Memory**
- **4. Learning to optimize / Learning DNN architectures**
- **5. Autoencoder / layer-wise training**
- **6. Learning to learn / meta-learning / few-shots**
- **7. DNN on graphs / trees / sets**
- **8. NTM for program induction / sequential decisions**
- **9. Generative Adversarial Networks (GAN)**
- **10. Deep Generative models, e.g., autoregressive**
- **11. Deep reinforcement learning**
- **12. Validate / Evade / Test / Understand / Verify DNNs**

New Network topology
New Losses
New Inputs
New Tasks
New Model Properties

https://qdata.github.io/deep2Read/
Machine (Deep) Learning in a Nutshell

Task:

\[ \text{Representation:} \]

\[ \text{Score Function:} \]

\[ \text{Search/Optimization} \]

Check / Validate (Models, Parameters)
A nutshell of Variations in Deep NN: Five Aspects

● Tasks:
  ● Discriminative prediction / Generative / Reinforce / Reasoning

● Formulate Input / Output:
  ● Data representation

● Architecture Design:
  ● Network Topology, Network Parameters

● Training / Searching / Learning
  ● With new losses
  ● With new optimization tricks
  ● New formulation of learning
  ● Scaling up with GPU, Scaling up with distributed optimization, e.g. Asynchronous SGD

● Validation / Trust / Test / Understand ...
  ● Software 2.0
Building Deep Neural Nets

http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf
Today’s Survey: Trends since ~2011

Inputs and Outputs

Architectures:

Losses

Validation

Software 2.0

Adapt from From NIPS 2017 DL Trend Tutorial
Recent Trend (1):
Convolutional Neural Networks
(aka CNNs and ConvNets)

Architectures:
- Convolutions
Machine (Deep) Learning in a Nutshell

- Task:
- Representation:
- Score Function:
- Search/Optimization
- Check / Validate (Models, Parameters)

- New Network Topology, Network Parameters
History of ConvNets

1998

*Gradient-based learning applied to document recognition* [LeCun, Bottou, Bengio, Haffner]

LeNet-5

2012

*ImageNet Classification with Deep Convolutional Neural Networks* [Krizhevsky, Sutskever, Hinton, 2012]
Important **Block**: Convolutional Neural Networks (CNN)

- Prof. Yann LeCun invented CNN in 1998
- First NN successfully trained with many layers

---

The bird occupies a local area and looks the same in different parts of an image. 
**We should construct neural nets which exploit these properties!**


---

Adapt from From NIPS 2017 DL Trend Tutorial
Locality and Translation Invariance

- **Locality**: objects tend to have a local spatial support
- **Translation invariance**: object appearance is independent of location

Can define these properties since an image lies on a grid/lattice
- ConvNet machinery applicable to other data with such properties, e.g. audio/text
CNN models **Locality and Translation Invariance**

Make fully-connected layer **locally-connected** and **sharing** weight

Adapt from From NIPS 2017 DL Trend Tutorial
Why CNN for Image?

Can the MLP network be simplified by considering the properties of images?

[Zeiler, M. D., ECCV 2014]

Represented as pixels

The most basic classifiers

Use 1st layer as module to build classifiers

Use 2nd layer as module …..

Can the MLP network be simplified by considering the properties of images?
Why CNN for Image

• **(1) Locality:** Some patterns are much smaller than the whole image.

A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

“beak” detector

Dr. Hung-yi Lee’s CNN slides
Why CNN for Image

• (2) Translation invariance: The same patterns appear in different regions.

Dr. Hung-yi Lee’s CNN slides
Why CNN for Image

• **(3) Subsampling** the pixels will not change the object

We can subsample the pixels to make image smaller

Less parameters for the network to process the image
The whole CNN

- Convolution
- Max Pooling
- Convolution
- Max Pooling
- Flatten

Can repeat many times

Dr. Hung-yi Lee’s CNN slides
The whole CNN

Property 1
- Some patterns are much smaller than the whole image

Property 2
- The same patterns appear in different regions.

Property 3
- Subsampling the pixels will not change the object

Can repeat many times
The whole CNN

Dr. Hung-yi Lee’s CNN slides
CNN – Convolution

Those are the network parameters to be learned.

<table>
<thead>
<tr>
<th>1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

Filter 1

Matrix $\mathbf{W}_{s1}$

Filter 2

Matrix $\mathbf{W}_{s2}$

Property 1

Each filter detects a small pattern (3 x 3).
CNN – Convolution

6 x 6 image

Filter 1

Dr. Hung-yi Lee’s CNN slides
CNN – Convolution

If \( \text{stride}=2 \)

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

6 x 6 image

Filter 1

\[
\begin{array}{ccc}
1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1 \\
\end{array}
\]

We set \( \text{stride}=1 \) below

3

-3

Dr. Hung-yi Lee’s CNN slides
CNN – Convolution

stride=1

6 x 6 image

Filter 1

Property 2

Dr. Hung-yi Lee’s CNN slides
CNN – Convolution

Do the same process for every filter

6 x 6 image

4 x 4 image

Dr. Hung-yi Lee’s CNN slides
CNN – Convolution

Do the same process for every filter

“detector 2”

Filter 2

6 x 6 image

4 x 4 image

Dr. Hung-yi Lee’s CNN slides
CNN – Convolution

You can do the same process for every filter.

6 x 6 image

Feature Map

4 x 4 image

Dr. Hung-yi Lee’s CNN slides
CNN – Colorful image (from matrix to tensor)

Colorful image (R, G, B)

Filter 1

Filter 2

Dr. Hung-yi Lee’s CNN slides
Convolution v.s. Fully Connected

- **Convolution**: Using filters \( \vec{w}_{s1} \) and \( \vec{w}_{s2} \) to convolve with the image.
- **Fully-connected**: Represents the architecture of a fully connected layer in a neural network.
Convolution v.s. Fully Connected

When 2 filters, 3*3*2=18 parameters!

When 2 filters, 36*2=72 parameters!
(1) Locality:

6 x 6 image

Filter 1

Each filter has 3x3=9 parameters!

Less parameters!

Only connect to 9 input, not fully connected

Dr. Hung-yi Lee’s CNN slides
(2) Translation invariance:

Filter 1

6 x 6 image

Less parameters!

Even less parameters! (weight sharing)

Shared weights (same 3*3 parameters)

Dr. Hung-yi Lee’s CNN slides
The whole CNN

- Convolution
- Max Pooling
- Convolution
- Max Pooling
- Flatten
- Fully Connected Feedforward network
- softmax

Can repeat many times

Dr. Hung-yi Lee’s CNN slides
(3) Subsampling:

CNN – Max Pooling

Filter 1

Filter 2

Dr. Hung-yi Lee’s CNN slides
(3) Subsampling:

CNN – Max Pooling

Filter 1

Filter 2

Dr. Hung-yi Lee’s CNN slides
(3) Subsampling:

CNN – Max Pooling

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

6 x 6 image

Conv

Max Pooling

New image but smaller

Each filter is a channel

Dr. Hung-yi Lee’s CNN slides
(3) Subsampling:

CNN – Max Pooling

6 x 6 image

New image but smaller

2 x 2 image

Each filter is a channel
The whole CNN

A new image

Smaller than the original image

The number of the channel is the number of filters

Can repeat many times

Dr. Hung-yi Lee’s CNN slides
The whole CNN

Fully Connected Feedforward network

Convolution
Max Pooling
Convolution
Max Pooling

Flatten

A new image

A new image

Dr. Hung-yi Lee’s CNN slides
Flatten

Fully Connected Feedforward network

softmax

Dr. Hung-yi Lee’s CNN slides
**CNN in Keras**

*network structure and input format (vector -> 3-D tensor)*

```
model2.add(Convolution2D(25, 3, 3, input_shape=(1, 28, 28)))
```

- **Input:** 1 x 28 x 28
- **Convolution:** 25 x 26 x 26
- **Max Pooling:** 25 x 13 x 13
- **Convolution:** 50 x 11 x 11
- **Max Pooling:** 50 x 5 x 5

How many parameters for each filter?
- 9
- 225

Dr. Hung-yi Lee’s CNN slides
Only modified the **network structure** and **input format (vector -> 3-D tensor)**

**CNN in Keras**

```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

```
model2.add(Flatten())
```

Dr. Hung-yi Lee’s CNN slides
More Application: Playing Go

Next move (19 x 19 positions)

19 x 19 vector

Fully-connected feedforward network can be used

But CNN performs much better.

19 x 19 matrix (image)

Black: 1
white: -1
none: 0

Dr. Hung-yi Lee’s CNN slides
More Application: Speech

The filters move in the frequency direction.

Dr. Hung-yi Lee’s CNN slides
Convolutional Neural Networks

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]
Fast-forward to today: ConvNets are everywhere

[Krizhevsky 2012]
Fast-forward to today: ConvNets are everywhere

Detection

Segmentation

[Faster R-CNN: Ren, He, Girshick, Sun 2015]  [Farabet et al., 2012]
Residual Trick:

Residual/Skip Connections

- A shallower model (18 layers)
- A deeper counterpart (34 layers)

- Richer solution space
- A deeper model should not have higher training error
- A solution by construction:
  - Original layers: copied from a learned shallower model
  - Extra layers: set as identity
  - At least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper...

Revolution of Depth

Adapt from From NIPS 2017 DL Trend Tutorial

ImageNet Classification top-5 error (%)

The diagram illustrates the trade-off between Top-1 accuracy and operations (in G-Ops) for various convolutional neural network architectures. The accuracy is represented on the vertical axis, while the horizontal axis shows the number of operations. Each circle corresponds to a different architecture, with the size of the circle indicating the number of operations. The architectures are color-coded for easy distinction:

- Inception-v3
- Inception-v4
- ResNet-50
- ResNet-101
- ResNet-18
- ResNet-34
- GoogLeNet
- ENet
- BN-NIN
- BN-AlexNet
- AlexNet
- VGG-16
- VGG-19
Adaptive / Dynamic Trick:

- Diet Networks: Thin Parameters for Fat Genomics, ICLR 2017
- Dynamic Filter Networks, NIPS 2016
- Hyper Networks, ICLR 2017
- Optimal Architectures in a Solvable Model of Deep Networks, NIPS16
- AdaNet: Adaptive Structural Learning of Artificial Neural Networks, ICML17
- SplitNet: Learning to Semantically Split Deep Networks for Parameter Reduction and Model Parallelization, ICML17
- Image Question Answering using Convolutional Neural Network with Dynamic Parameter, CVPR 2016
- Many others..
Recent Trend (2): Recurrent Neural Networks

**Architectures:**
- Recurrent, over space and/or time.
  - + attention
- Attention-only!
Machine (Deep) Learning in a Nutshell

- Task:
- Representation:
  - Score Function:
  - Search/Optimization
  - Check / Validate (Models, Parameters)

- New Network Topology, Network Parameters
Important **Block**: Recurrent Neural Networks (RNN)

- Prof. Schmidhuber invented "Long short-term memory" – Recurrent NN (LSTM-RNN) model in 1997

Important **Block**: Recurrent Neural Networks (RNN)

- Prof. Schmidhuber invented "Long short-term memory" – Recurrent NN (LSTM-RNN) model in 1997

\[
h_t = \sigma(W x_t + U h_{t-1} + b) = LSTM(x_t)
\]

RNN models *dynamic temporal dependency*

- Make **fully**-connected layer model **each unit recurrently**
- Units form a **directed chain graph** along a sequence
- Each unit uses **recent history** and current input in modeling

**LSTM for Machine Translation**
(German to English)

Image credit: wildML
Recurrent Neural Networks (RNNs)

Traditional “Feed Forward” Neural Network

Recurrent Neural Network

predict a vector at each timestep
Standard “Feed-Forward” Neural Network
Recurrent Neural Networks (RNNs)

RNNs can handle

- one to one
- one to many
- many to one
- many to many

e.g. **Sentiment Classification**
sequence of words -> sentiment

http://cs231n.stanford.edu/slides/
Recurrent Neural Networks (RNNs)

RNNs can handle

- One to one
- One to many
- Many to one
- Many to many

E.g., Machine Translation
seq of words -> seq of words

http://cs231n.stanford.edu/slides/
In machine translation, the input is a sequence of words in source language, and the output is a sequence of words in target language.

**Encoder**
An RNN to encode the input sentence into a hidden state (feature)

**Decoder**
An RNN with the hidden state of the sentence in source language as the input and output the translated sentence

Encoder-decoder architecture for machine translation

Adapt from Professor Qiang Yang of HK UST
Attention Trick:

Seq2Seq with Attention

- Embedding used to predict output, and compute next hidden state
The attention module gives us a weight for each input.

C1 is a weighted sum of the hidden encodings.

Based: Dr. Yangqiu Song’s slides
Transformer: Exploiting Self Attentions

- A Google Brain model.
  - Variable-length input
  - Fixed-length output (but typically extended to a variable-length output)
  - **No recurrence**
  - Surprisingly not patented.
- Uses 3 kinds of attention
  - Encoder self-attention.
  - Decoder self-attention.
  - Encoder-decoder multi-head attention.

![Transformer - model architecture.](image)

Based: Dr. Yangqiu Song’s slides
Notable pre-trained NLP models

ELMo: Embeddings from Language Models
Pre-trained biLSTM for contextual embedding

BERT: Bidirectional Encoder Representations from Transformers
Pre-trained transformer encoder for sentence embedding

THE TRANSFORMER

ULM-FiT

OpenAI Transformer

ELMo

Based: Dr. Yangqiu Song’s slides
Different tasks use the OpenAI transformer in different ways.

OpenAI's GPT-2 is a really large transformer.

Based: Dr. Yangqiu Song’s slides
Recent Trend (3): Neural Architectures with Memory

Architectures:
memory and multi-hop reasoning to perform AI tasks better
Machine (Deep) Learning in a Nutshell

Task:

Representation:

Score Function:

Search/Optimization

Check / Validate (Models, Parameters)

- New Network Topology, Network Parameters
e.g. for Story Comprehension

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk. Joe travelled to his office. Joe left the milk. Joe went to the bathroom.

Questions from Joe’s angry mother:

Q1: Where is Joe?

Q2: Where is the milk now?

Q3: Where was Joe before the office?
Need external explicit memory for long-range reasoning

Deeper AI tasks require explicit memory and multi-hop reasoning over it

- RNNs have short memory
- Cannot increase memory without increasing number of parameters
- Need for compartmentalized memory
- Read/Write should be asynchronous
End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015

Generate memories

Transform Query

Score memories

Make averaged output

Response

Generate outputs

Weighted Sum

Softmax

Inner Product

Predicted Answer

Weight

Embedding C

Scores

Embedding A

Sentences \{x_i\}

Output

Weights

Input

Question q

Embedding B

Generate memories
Multi-hop MemN2N

Different Memories and Outputs for each Hop

\[ u^{k+1} = u^k + o^k \]

End-To-End Memory Networks, Sukhbaatar et. al., NIPS 2015
Neural Architectures with Memory

• Antoine Bordes, Y-Lan Boureau, Jason Weston, Learning End-to-End Goal-Oriented Dialog, ICLR 2017
• Karol Kurach, Marcin Andrychowicz & Ilya Sutskever Neural Random-Access Machines, ICLR, 2016
• Emilio Parisotto & Ruslan Salakhutdinov Neural Map: Structured Memory for Deep Reinforcement Learning, ArXiv, 2017
• Oriol Vinyals, Meire Fortunato, Navdeep Jaitly Pointer Networks, ArXiv, 2017
• Jack W Rae et al., Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes, ArXiv 2016
• Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, Honglak Lee, Control of Memory, Active Perception, and Action in Minecraft, ICML 2016
• Wojciech Zaremba, Ilya Sutskever, Reinforcement Learning Neural Turing Machines, ArXiv 2016
Attention and Memory Toolbox

Sequence Prediction

Seq2Seq

Multimodality

Attention/Pointers

Read/Write memories

Temporal Hierarchies

Key,Value memories

Recurrent Architectures

Adapt from From NIPS 2017 DL Trend Tutorial
Recent Trend (4): Learning to Optimize / Learning to Search DNN architecture
Machine (Deep) Learning in a Nutshell

Task:

Representation:

Score Function:

Search/Optimization

Check / Validate (Models, Parameters)
Figure 1: An overview of Neural Architecture Search.
**Neural Optimizer Search with Reinforcement Learning, ICML17**

* e.g. hyperparam search

---

**Figure 2.** Computation graph of some commonly used optimizers: SGD, RMSProp, Adam. Here, we show the computation of Adam in 1 step and 2 steps. Blue boxes correspond to input primitives or temporary outputs, yellow boxes are unary functions and grey boxes represent binary functions. $g$ is the gradient, $\hat{m}$ is the bias-corrected running estimate of the gradient, and $\hat{v}$ is the bias-corrected running estimate of the squared gradient.

**Figure 3.** Overview of the controller RNN. The controller iteratively selects subsequences of length 5. It first selects the 1st and 2nd operands $op_1$ and $op_2$, then 2 unary functions $u_1$ and $u_2$ to apply to the operands and finally a binary function $b$ that combines the outputs of the unary functions. The resulting $b(u_1(op_1), u_2(op_2))$ then becomes an operand that can be selected in the subsequent group of predictions or becomes the update rule. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.
Recent Trend (5): Layer-wise pretraining / Auto-Encoder
Machine (Deep) Learning in a Nutshell

- Task:
- Representation:
- Score Function:
- Search/Optimization
- Check / Validate (Models, Parameters)

- Training / Searching / Learning
  - With new losses
  - With new optimization tips
  - New formulation of learning
  - Scaling up with GPU, Scaling up with distributed optimization, e.g. Asynchronous SGD
Recap: “Block View”

- **1st hidden layer**
- **2nd hidden layer**
- **Output layer**
- **Loss Module**

Symbols:
- $x$
- $z_1$
- $h_1$
- $z_2$
- $h_2$
- $z_3$
- $\hat{y}$
- $E(\hat{y}, y)$
an auto-encoder-decoder is trained to reproduce the input

Reconstruction Loss: force the ‘hidden layer’ units to become good / reliable feature detectors
The new way to train multi-layer NNs…
The new way to train multi-layer NNs…

Train this layer first

https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt
The new way to train multi-layer NNs...

Train **this** layer first
then **this** layer
then **this** layer
then **this** layer
finally **this** layer
The new way to train multi-layer NNs...

Each layer can be trained to be an auto-encoder (e.g., via reconstruction loss)

Basically, it is forced to learn good features that describe what comes from the previous layer

https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt
Recent Trend (6): Learning to Learn
Machine (Deep) Learning in a Nutshell

- Task:
- Representation:
- Score Function:
- Search/Optimization:
- Check / Validate (Models, Parameters)
Learning to Learn

- What is Meta Learning / Learning to Learn?
  - Go beyond train/test from same distribution.
  - Task between train/test changes, so model has to “learn to learn”

- Datasets

**Image recognition**

Given 1 example of 5 classes:

Classify new examples

**Reinforcement learning**

Given a small amount of experience

Solve a new task

Lake et al, 2013, 2015

Chelsea Finn, UC Berkeley

How? learn to learn many other tasks

fig. from Duan et al. ‘17

Adapt from From NIPS 2017 DL Trend Tutorial
Learning to Learn

Model Based

- Santoro et al. ‘16
- Duan et al. ’17
- Wang et al. ‘17
- Munkhdalai & Yu ‘17
- Mishra et al. ‘17

Metric Based

- Koch ’15
- Vinyals et al. ‘16
- Snell et al. ‘17
- Shyam et al. ‘17

Optimization Based

- Schmidhuber ’87, ’92
- Bengio et al. ’90, ’92
- Hochreiter et al. ’01
- Li & Malik ‘16
- Andrychowicz et al. ’16
- Ravi & Larochelle ‘17
- Finn et al ‘17

Adapt from From NIPS 2017 DL Trend Tutorial
Recent Trend (7): Variants of Input, e.g., Graphs, Trees, Sets
Machine (Deep) Learning in a Nutshell

Task:

Representation:

Score Function:

Search/Optimization

Check / Validate (Models, Parameters)
Geometric Deep Learning on Graphs and Manifolds, NIPS 2017 Tutorial

Graph Nets (GNs) are a class of models that:
- Use graphs as inputs and/or outputs and/or latent representation
- Manipulate graph-structured representations
- Reflect relational structure
- Share model components across entities and relations

Examples include:
- Graph Neural Networks (Scarselli et al 07; 08)
- Recursive Neural Networks (Goller et al 96)
- Pointer Networks (Vinyals et al 2015)
- Graph Convolutional Networks (Bruna et al 2013; Duvenaud et al 15; Henaff et al 15; Kipf & Welling 16; Defferrard et al 17)
- Gated Graph Neural Networks (Li et al 15)
- Interaction Networks (Battaglia et al 2016; Raposo et al 2017; )
- Message Passing Networks (Gilmer et al. 2017)
Inductive Bias for Graphs

- If we have a graph on N nodes, there are N! possible orderings of the nodes.
- Ideally want a model invariant to the order of nodes.

\[ X = (v_1, v_2, v_3, e_{12}, e_{13}, e_{23}) \]

\[ \text{perm}(X) = (v_3, v_1, v_2, e_{13}, e_{23}, e_{12}) \]

Order Invariant Model

Slides credits: Justin Gilmer

Adapt from From NIPS 2017 DL Trend Tutorial
Recent Trend (8):
Tasks in the form of Symbolic input/outputs/Program Induction

Inputs and Outputs:
- Discrete symbols, (e.g. the program itself)
- Program execution traces
- Program I/O pairs
These can also be mixed with perceptual data.

Architectures:
- (Mostly) recurrent
- Sometimes including ConvNets as a visual front-end.

Losses:
- Differentiable, predicting discrete program outputs or code itself: softmax cross entropy.
- Not differentiable: RL

Adapt from NIPS 2017 DL Trend Tutorial
Machine (Deep) Learning in a Nutshell

Task:

Representation:

Score Function:

Search/Optimization

Check / Validate (Models, Parameters)
Neural Program Induction - Research Landscape

- Neural network is the program:
  - Learning to Execute, Neural Turing Machine, Neural GPU, Neural RAM, Neural Programmer-Interpreter, Neural Task Programmer, Differentiable Forth Interpreter

- Neural network generates source code:
  - DeepCoder, RobustFill, Neural Inductive Logic Programming

- Probabilistic programming with neural networks:
  - TerpreT, Edward, Picture

Adapt from From NIPS 2017 DL Trend Tutorial
Neural Turing Machines

Neural Turing Machines, Graves et. al., arXiv:1410.5401
Task with Sequential Symbolic Form

• Words, letters, strings, ..
• Computer Programs , ...
• Sequence decision making, e.g., games, RL
Recent Trend (9):
Generative Adversarial Networks (GAN)
Machine (Deep) Learning in a Nutshell

1. Task:
2. Representation:
3. Score Function:
4. Search/Optimization
5. Check / Validate (Models, Parameters)
Dueling Neural Networks
Adversarial Nets Framework

\[ D(x) \text{ tries to be near 1} \]

Differentiable function \( D \)

\( x \text{ sampled from data} \)

\[ D \text{ tries to make } D(G(z)) \text{ near 0, } G \text{ tries to make } D(G(z)) \text{ near 1} \]

\( D \)

\( x \text{ sampled from model} \)

Differentiable function \( G \)

Input noise \( z \)

(Goodfellow 2016)
Unsupervised cross-domain image generation

This paper captures special characteristics of one image collection and figures out how these characteristics could be translated into the other image collection, all in the absence of any paired training examples. CycleGANs method can also be applied in variety of applications such as Collection Style Transfer, Object Transfiguration, season transfer and photo enhancement.
Image Super-Resolution

bicubic (21.59dB/0.6423)  SRResNet (23.53dB/0.7832)  SRGAN (21.15dB/0.6868)  original

[Ledig et al. CVPR 2017]
Label2Image

Input

Ground truth

L1

cGAN

L1 + cGAN

[Isola et al. CVPR 2017]
Edges2Image

Input | Ground truth | Output
---|---|---

Input | Ground truth | Output
---|---|---

[Isola et al. CVPR 2017]
Text2Image

this small bird has a pink breast and crown, and black primaries and secondaries.

the flower has petals that are bright pinkish purple with white stigma

this magnificent fellow is almost all black with a red crest, and white cheek patch.

this white and yellow flower have thin white petals and a round yellow stamen
Progressive GAN

PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION, ICLR 2018
Recent Trend (10):
Deep Generative Models: Autoregressive Kind
Machine (Deep) Learning in a Nutshell

Task:
Representation:
Score Function:
Search/Optimization
Check / Validate (Models, Parameters)
Generative models - Research Landscape

- Latent variable models (VAE, DRAW)
- Implicit (GAN, GMMN, Progressive GAN)
- Transform (NICE, IAF, Real NVP)
- Autoregressive (NADE, MADE, RIDE, PixelCNN, WaveNet)

NIPS 2016 Tutorial on Generative Adversarial Networks
Autoregressive Models

\[ P(x; \theta) = \prod_{n=1}^{N} P(x_n | x_{<n}; \theta) \]

- Each factor can be parametrized by \( \theta \), which can be shared.
- The variables can be arbitrarily ordered and grouped, as long as the ordering and grouping is consistent.
Recurrent versus Causal Convolutional Nets

- The architecture is parallelizable along the time dimension (during training or scoring)
- Easy access to many states from the past
Why Generative Models?

• Excellent test of ability to use high-dimensional, complicated probability distributions
• Simulate possible futures for planning or simulated RL
• Missing data
  • Semi-supervised learning
• Multi-modal outputs
• Realistic generation tasks

(Goodfellow 2016)
Recent Trend (11): Deep Reinforcement Learning

10 Breakthrough Technologies 2017

These technologies all have staying power. They will affect the economy and our politics, improve medicine, or influence our culture. Some are unfolding now; others will take a decade or more to develop. But you should know about all of them right now.
Machine (Deep) Learning in a Nutshell

Task:
Representation:
Score Function:
Search/Optimization
Check / Validate (Models, Parameters)
Reinforcement Learning (RL)

• What’s Reinforcement Learning?

• Agent interacts with an environment and learns by maximizing a scalar reward signal
• No labels or any other supervision signal.
• Previously suffering from hand-craft states or representation.

Adapt from Professor Qiang Yang of HK UST
Deep Reinforcement Learning

- Human

So what’s **DEEP RL**?

{Raw Observation, Reward} \[\rightarrow\] Environment \[\rightarrow\] {Actions}

Adapt from Professor Qiang Yang of HK UST
AlphaGO: Learning Pipeline

• Combine Supervised Learning (SL) and RL to learn the search direction in Monte Carlo Tree Search

SL policy Network
• Prior search probability or potential

Rollout:
• combine with MCTS for quick simulation on leaf node

Value Network:
• Build the Global feeling on the leaf node situation

AlphaGo \{Fan, Lee, Master\} \times \textbf{AlphaGo Zero}:

- supervised learning from human expert positions \times \text{from scratch by self-play reinforcement learning ("tabula rasa")}
- additional (auxiliary) input features \times \text{only the black and white stones from the board as input features}
- separate policy and value networks \times \text{single neural network}
- tree search using also Monte Carlo rollouts \times \text{simpler tree search using only the single neural network to both evaluate positions and sample moves}
- (AlphaGo Lee) distributed machines + \text{48 tensor processing units (TPUs)} \times \text{single machines + \text{4 TPUs}}
- (AlphaGo Lee) several months of training time \times \text{72 h of training time (outperforming AlphaGo Lee after 36 h)}

Recent Trend (12):
Robustness / Trustworthiness / Understand / Verify / Test / Evade / Detect Bias / Protect DNN
Machine (Deep) Learning in a Nutshell

- Task:
  - Representation:
    - Score Function:
      - Search/Optimization
        - Check / Validate (Models, Parameters)
Evade DNN, e.g. Adversarial Examples (AE)

“panda”  +  0.007 × [noise]  =  “gibbon”

Francois Chollet - https://blog.keras.io/the-limitations-of-deep-learning.html
Breaking CNNs

Intriguing properties of neural networks [Szegedy ICLR 2014]

Take a correctly classified image (left image in both columns), and add a tiny distortion (middle) to fool the ConvNet with the resulting image (right).
Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen et al. CVPR 2015]
Cat-and-mouse game

[Szegedy+ 2014]: first discover adversarial examples

[Goodfellow+ 2015]: Adversarial training (AT) against FGSM

[Papernot+ 2015]: defensive distillation

[Calini & Wagner 2016]: distillation is not secure

[Papernot+ 2017]: better distillation

[Carlini & Wagner 2017]: All detection strategies fail

[Madry+ 2017]: AT against PGD, informal argument about optimality

[Lu+ July 12 2017]: "NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles"

[Athalye & Sutskever July 17 2017]: break defense with AT on PGD with transformed examples
Figure 1: Five example images from the imSitu visual semantic role labeling (vSRL) dataset. Each image is paired with a table describing a situation: the verb, cooking, its semantic roles, i.e. agent, and noun values filling that role, i.e. woman. In the imSitu training set, 33% of cooking images have man

Table 3: Local adversarial robustness tests. All times are in seconds.

<table>
<thead>
<tr>
<th></th>
<th>$\delta = 0.1$</th>
<th></th>
<th>$\delta = 0.075$</th>
<th></th>
<th>$\delta = 0.05$</th>
<th></th>
<th>$\delta = 0.025$</th>
<th></th>
<th>$\delta = 0.01$</th>
<th></th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point 1</td>
<td>SAT 135</td>
<td>SAT 239</td>
<td>SAT 24</td>
<td>UNSAT 609</td>
<td>UNSAT 57</td>
<td>1064</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point 2</td>
<td>UNSAT 5880</td>
<td>UNSAT 1167</td>
<td>UNSAT 285</td>
<td>UNSAT 57</td>
<td>UNSAT 5</td>
<td>7394</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point 3</td>
<td>UNSAT 863</td>
<td>UNSAT 436</td>
<td>UNSAT 99</td>
<td>UNSAT 53</td>
<td>UNSAT 1</td>
<td>1452</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point 4</td>
<td>SAT 2</td>
<td>SAT 977</td>
<td>SAT 1168</td>
<td>UNSAT 656</td>
<td>UNSAT 7</td>
<td>2810</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point 5</td>
<td>UNSAT 14560</td>
<td>UNSAT 4344</td>
<td>UNSAT 1331</td>
<td>UNSAT 221</td>
<td>UNSAT 6</td>
<td>20462</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Today Recap: Some Recent Trends

1. **CNN** / Residual / Dynamic parameter
2. RNN / Attention / Seq2Seq / BERT …
3. Neural Architecture with explicit Memory
4. Learning to optimize / Learning DNN architectures
5. Autoencoder / layer-wise training
6. Learning to learn / meta-learning/ few-shots
7. DNN on graphs / trees / sets
8. NTM 4program induction / sequential decisions
9. Generative Adversarial Networks (GAN)
10. Deep Generative models, e.g., autoregressive
11. Deep reinforcement learning
12. Validate / Evade / Test / Understand / Verify DNNs

• (Many more exciting trends not covered here!)
References

- Dr. Yann Lecun’s deep learning tutorials
- Dr. Li Deng’s ICML 2014 Deep Learning Tutorial
- Dr. Kai Yu’s deep learning tutorial
- Dr. Rob Fergus’ deep learning tutorial
- Prof. Nando de Freitas’ slides
- Olivier Grisel’s talk at Paris Data Geeks / Open World Forum
- Dr. Hung-yi Lee’s CNN slides
- NIPS 2017 DL Trend Tutorial