Deep Learning for Engineers

John Aynsley
Deep Learning for Engineers

AI, ML, and Deep Learning

Training a Neural Network

Deeper Insights

CNNs and RNNs

Tool Flow
AI versus ML versus Deep Learning

- Knowledge databases
- Logical inference
- Expert systems

- Giving computers the ability to learn without being explicitly programmed

- Neural networks with many hidden layers
"Classical" Machine Learning

Tasks
Classification
Regression
Clustering
Anomaly detection
Dimensionality reduction

Algorithms
Support vector machines
Bayesian statistics
Markov models
Decision trees
Random forests
K-means
... and many more

Could be all you need!
Appropriate for smaller datasets
Why Deep Learning Now?

2012 – a CNN wins ImageNet Challenge

Bigger datasets
Faster computers

Since 2012

Improved neural network architectures
Neural networks often outperforming previous state-of-the-art
## The ImageNet Challenge (ILSVRC)

ImageNet Large Scale Visual Recognition Challenge: 1.2M images in 1000 categories

<table>
<thead>
<tr>
<th>Year</th>
<th>Network</th>
<th>#Layers</th>
<th>Top-5 Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011 winner</td>
<td>(Not a NN)</td>
<td>-</td>
<td>25.8%</td>
</tr>
<tr>
<td>2012 winner</td>
<td>AlexNet (CNN)</td>
<td>8</td>
<td>16.4%</td>
</tr>
<tr>
<td>2013 winner</td>
<td>ZFNet (CNN)</td>
<td>8</td>
<td>11.7%</td>
</tr>
<tr>
<td>2014</td>
<td>VGGNet (CNN)</td>
<td>19</td>
<td>7.3%</td>
</tr>
<tr>
<td>2014 winner</td>
<td>GoogLeNet (Inception)</td>
<td>22</td>
<td>6.7%</td>
</tr>
<tr>
<td>2015 winner</td>
<td>ResNet (residual)</td>
<td>152</td>
<td>3.6%</td>
</tr>
<tr>
<td>2016 winner</td>
<td>CUImage (ensemble)</td>
<td>-</td>
<td>3.0%</td>
</tr>
</tbody>
</table>

Dramatic improvement

Human error rate ~ 5%

3% bad labels

Training typically takes a few weeks on a few GPUs
# Cloud Computing versus Edge Computing

<table>
<thead>
<tr>
<th>Cloud Computing in Data Centers</th>
<th>Edge Computing in Embedded Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Massive, scalable compute power</td>
<td>Limited compute power</td>
</tr>
<tr>
<td>Unlimited storage</td>
<td>Limited storage</td>
</tr>
<tr>
<td>High latency</td>
<td><strong>Low latency (real-time response)</strong></td>
</tr>
<tr>
<td>Restricted bandwidth</td>
<td>Unrestricted bandwidth</td>
</tr>
<tr>
<td>Low energy efficiency</td>
<td>High energy efficiency</td>
</tr>
<tr>
<td>Reliant on internet connection</td>
<td><strong>Can run without internet connection</strong></td>
</tr>
<tr>
<td>Data sent over internet (privacy?)</td>
<td><strong>Data kept local</strong></td>
</tr>
<tr>
<td>Relatively high cost</td>
<td>Low cost</td>
</tr>
</tbody>
</table>
Cloud versus Edge ML/DL Applications

- Recommendation engines for websites
- Fraud detection on financial transactions
- Chat bots

Cloud:

- Images, video, voice, temperature, vibration, ...

Edge / IoT Sensors:
Edge Applications of Deep Learning

Vision
  Image recognition
  Object detection
  Image segmentation
Speech recognition
Text analysis
Anomaly detection
Automotive Applications

ADAS and autonomous vehicles
  Traffic sign recognition
  Lane detection
  Pedestrian detection
  Human pose estimation
  Monitoring for a distracted driver

Detecting vehicle occupancy for car sharing
Detecting driver identity to store seat settings
Industrial, Medical, Retail, IoT

- Touchscreen character recognition
- Voice control - keyword spotting
- Medical diagnosis from images
- Customer counts and demographics from cameras in retail stores
- Real-time failure prediction in industrial equipment
- Face recognition in smart doorbells
- Food classification – allergy advice
Deep Learning for Engineers

AI, ML, and Deep Learning

Training a Neural Network

Deeper Insights

CNNs and RNNs

Tool Flow
Supervised Learning

Input → Learn to predict the output from the input → Output

Novel input → Model → Predicted output

Training

Deployment

Platform

GPU

MCU, CPU, GPU, FPGA, SoC
Training a Neural Network

Labels

 Persian
 Abyssian
 Siamese
 British Shorthair
 Scottish Fold
 Burmese

Ground Truth

Compare

Cost Function

Prediction

Neural Network

Training data
Training a Neural Network

Labels
- Persian
- Abyssian
- Siamese
- British Shorthair
- Scottish Fold
- Burmese

Training data

Ground Truth

Compare

Cost Function

Layers

Units/neurons
An Artificial Neuron

**Linear function**

\[ u = \sum_{i=1}^{3} w_i x_i + b \]

\[ y = activation(u) \]

**Non-linear function**

- **Input units**
- **Hidden unit**
- **Output units**
Common Activation Functions

Sigmoid aka logistic

ReLU
A Deep Neural Network

\[ y_j = \text{RELU} \left( \sum_{i=1}^{3} w_{ji}x_i + b_j \right) \]

\[ y_j = \text{RELU} \left( \sum_{i=1}^{n} w_{ji}x_i + b_j \right) \]
Regression Task

Output $y$

Input $x$

![Graph showing a scatter plot with labeled axes and points]
Define a Hypothesis or Model or Network

$$h_{\text{slope, offset}}(x) = \text{slope} \cdot x + \text{offset}$$
Cost or Loss or Error Function

\[ \text{Loss}(\text{slope}, \text{offset}) = \frac{1}{2m} \sum_i (h(x_i) - y_i)^2 \]

Prediction

Error

Ground truth

slope = 0, offset = 5
Cost as a Function of Slope and Offset
Contour Plot of Cost Function

Which way?

Optimal solution
Gradient Descent

\[
\frac{\partial \text{Loss}}{\partial \text{slope}} - \frac{\partial \text{Loss}}{\partial \text{offset}}
\]

weight $\leftarrow$ weight $- \alpha \cdot \frac{\partial \text{Loss}}{\partial \text{weight}}$
Gradient Descent Algorithm

Initialize weights (trainable parameters)

for each training step:

    Calculate the gradient with respect to every weight

for each weight:

    new_weight = weight - learning_rate * gradient

for each weight:

    weight = new_weight
Converging on the Minimum

Final slope = -2.31499114425  offset = 4.3890585415
Stochastic Gradient Descent

Use a subset of the training data at each step
Non-Linear Regression and Classification
Piecewise Linear Approximation

\[ y = \sum_{i=1}^{4} w_i x_i + b \]

\[ y_j = \text{RELU} (w_j x + b_j) \]
The Predicted Output

Such a network is hard to train
The Landscape of the Cost/Loss Function

Local and global minima and saddle points
A Deep Neural Network

Input unit  16 hidden units  16 hidden units  Output unit

\[ y_j = \text{RELU} \left( \sum_{i=1}^{n} w_{ji} x_i + b_j \right) \]

Cost function (mean squared error)

\[ y = \sum_{i=1}^{4} w_{i} x_i + b \]
The Predicted Output
Forward and Back-Propagation

Forward propagation calculates weighted sums and activation function.

Back propagation calculates gradients and adjusts weights.
Deep Learning for Engineers

AI, ML, and Deep Learning

Training a Neural Network

Deeper Insights

CNNs and RNNs

Tool Flow
Classification

Inputs

Hidden

Outputs

Logits

Probability vector

Cost

Cross-entropy

Classification

\[
\begin{pmatrix}
5.2 \\
0.01 \\
-0.3 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
0.9904 \\
0.0055 \\
0.0040 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\]
Underfitting and Overfitting

1 hidden layer of 6 neurons, 20 runs

4 hidden layers of 16 neurons, 20 runs

Underfitting  ➔  Go Deeper!  ➔  Overfitting
Regularization

\[
\text{Loss}(\vec{w}) = \frac{1}{2m} \sum_{i} (h(x_i) - y_i)^2
\]

Naked cost function

Regularization term

Mean square error for regression

Softmax + cross-entropy for classification
With L2 Regularization

4 hidden layers of 16 neurons, 40 runs
Dropout

Training: Drop half the hidden units for each training step

Testing: Keep all the hidden units, divide activations by 2
Hyperparameters

Number of hidden layers
Number of neurons in each layer
Sigmoid or ReLU activation
Choice of cost function
Choice of gradient descent algorithm
Learning rate
L2 regularization factor
Amount of dropout
(Many more ...
Training, Validation, and Test Datasets

Training:validation:test ~ 3:1:1

Hyperparameter tuning

Estimate generalization error

Biased toward validation dataset?
Deep Learning for Engineers

AI, ML, and Deep Learning

Training a Neural Network

Deeper Insights

CNNs and RNNs

Tool Flow
Kinds of Neural Network

ANN – Artificial Neural Network

CNN – Convolutional Neural Network (e.g. object recognition)

R-CNN – Regional CNN (image segmentation)

RNN – Recurrent Neural Network (e.g. speech & text processing)
Convolutional Neural Network

\[ y_j = RELU \left( \sum_{i=1}^{5} w_{ji} x_i + b_j \right) \]
Multiple Filters and Feature Maps

\[ y_n = \sum_{i=1}^{5} \sum_{j=1}^{5} \sum_{k=1}^{3} w_{ijk} x_{ijk} + b_n \]

5 x 5 x 3 x 30 weights, 30 biases
The Classical CNN Architecture

Input → Conv+ RELU → Max Pool → Conv+ RELU → Max Pool → Flatten → Fully-connected → Fully-connected

<table>
<thead>
<tr>
<th>Shape</th>
<th>32x32x3</th>
<th>32x32x16</th>
<th>16x16x16</th>
<th>16x16x32</th>
<th>8x8x32</th>
<th>2048</th>
<th>128</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td># Parameters</td>
<td>416</td>
<td>832</td>
<td></td>
<td></td>
<td></td>
<td>2,048</td>
<td>128</td>
<td>1,290</td>
</tr>
<tr>
<td># Values</td>
<td>3,072</td>
<td>16,384</td>
<td>4,096</td>
<td>8,192</td>
<td>2,048</td>
<td>2,048</td>
<td>128</td>
<td>10</td>
</tr>
</tbody>
</table>

Features Detection

Classifier
Evolution of CNN Architectures

Traditional CNN:

Convolutions → pool → convolution → pool → full → full → full → output

Hierarchical feature detectors  Classifier

Sledgehammer!

Replace with convolutions – Fully Convolutional Network
Naïve Inception Module

Don't know what the optimal sparse structure is, so hedge our bets:

28x28x100 → 3x3 convolution → Concatenate

28x28x100 → 5x5 convolution → Concatenate

28x28x100 → 7x7 convolution → Concatenate

28x28x400 → Max pooling

Feature depth would explode
Example GoogLeNet Inception Module

14x14x480

1x1
1x1
1x1
max pool
ReLU

1x1
14x14x96
3x3
ReLU

1x1
14x14x16
5x5
ReLU

1x1
14x14x480
1x1
ReLU

14x14x192
Concatenate
14x14x208
14x14x48
14x14x480
14x14x64
14x14x512
Transfer Learning

Low-level features
Conv
Pooling
Conv
Fully-connected
High-level features
Pooling
Fully-connected
Classifier

Reuse trained weights
Train from scratch
Recurrent Neural Network (RNN)

$y_0, y_1, y_2, y_3 \ldots$

Output sequence

Layer

$x_t$

$x_0, x_1, x_2, x_3 \ldots$

Input sequence

Equivalent unrolled network

Each copy has identical weights
RNN Applications

Natural Language Processing

Category / next word
*Output is final state*

Sequence of words
*Output is whole sequence*

Both input and output sequences can be variable length
Very powerful and effective, but training can be tricky
Long Short Term Memory – LSTM

![LSTM Diagram](image-url)

- Output gate
- Forget gate
- Input gate

Output

Internal state

Input

LSTM
LSTM Gates

\[ \text{Gate} = \sigma \left( \sum_i w_i x_i + \sum_i u_i r_i \right) \]
LSTM Trained on Linux Source Code

Generated by a 3-layer LSTM trained on the entire Linux source code ...

Deep Learning for Engineers

- AI, ML, and Deep Learning
- Training a Neural Network
- Deeper Insights
- CNNs and RNNs
- Tool Flow
## Training versus Inference

<table>
<thead>
<tr>
<th>Training</th>
<th>Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop or cloud computing</td>
<td>Cloud or edge computing</td>
</tr>
<tr>
<td>Large dataset</td>
<td>One sample at a time</td>
</tr>
<tr>
<td>Forward and backward passes through neural network</td>
<td>Forward pass only (simpler network)</td>
</tr>
<tr>
<td>Minutes-weeks on GPU</td>
<td>Milliseconds-seconds/sample on edge device</td>
</tr>
<tr>
<td>Weights are computed</td>
<td>Weights are known and can be compressed</td>
</tr>
</tbody>
</table>
Open Source Training Frameworks

Caffe
Berkeley Artificial Intelligence Research (BAIR)
Berkeley Vision

Caffe2
facebook

TensorFlow™
Google

Microsoft | Cognitive Toolkit

torch
mxnet
theano
Keras
Cloud Platforms for Training and Inference

- Amazon SageMaker
- Google Cloud
- Microsoft Azure
- IBM Cloud
- IBM AI OpenScale
- and many others
The Cloud can Provide

Just the hardware (VM)

VM with pre-installed, pre-configured ML software

MLaaS – Machine Learning as a Service

Specific ML services (language translation, chat bots, ...
ML / DL Tool Flow for Edge Computing

- **Training**
  - Caffe
  - TensorFlow

- **Format conversion**

  Open Neural Network Exchange Format (ONNX)
ML / DL Tool Flow for Edge Computing

- Training
  - Caffe
  - TensorFlow™

  - Format conversion

  - Pruning, quantization, compression

  Pruning – setting near-zero weights to zero
Trade-Off Curve

- L2 regularization w/o retrain
- L1 regularization w/o retrain
- L1 regularization w/ retrain
- L2 regularization w/ iterative prune and retrain

ML / DL Tool Flow for Edge Computing

Training

- Caffe
- TensorFlow™

Format conversion

Pruning, quantization, compression

Pruning – setting near-zero weights to zero
Re-training – with sparse weight matrix (iterative)
Quantization – replacing 32-bit floats with 8-bit integers (typical)
Compression – storing sparse weight matrix in compressed format

10 X reduction in weight memory
ML / DL Tool Flow for Edge Computing

Training
- Caffe
- TensorFlow™

Format conversion

Pruning, quantization, compression

Engine-specific optimizations

Using DSP or VLIW or GPU processor instructions

Operation fusing – using specialist neural network instructions

E.g. Arm CMSIS-NN uses DSP instructions of Cortex-M4
E.g. Arm Compute Library uses NEON acceleration for Cortex-A
## Benchmark CNNs

<table>
<thead>
<tr>
<th>ImageNet Challenge Winners</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>SqueezeNet</td>
</tr>
<tr>
<td>VGG</td>
<td>MobileNet</td>
</tr>
<tr>
<td>Inception</td>
<td>DenseNet</td>
</tr>
<tr>
<td>Resnet</td>
<td>SSD</td>
</tr>
<tr>
<td></td>
<td>YOLO</td>
</tr>
</tbody>
</table>

Pre-trained networks and transfer learning
MobileNet V1 and V2

CNN for image recognition and object detection on mobile

Hyperparameters:
- Scale down the size of the feature maps
- Scale down the number of features

MobileNet v2 available as 22 pre-built, pre-trained models:
- From 6M to 1.6M parameters, from 75% to 45% Top-1 accuracy
Deploying Mobilenet

Feature engineering
Collect and curate training, validation, test datasets
Select hyperparameters
Select a classifier (recognition) or detector (object detection)
Train classifier/detector, measure overfitting and generalisation error
For More Information

5-Day Training Course: Practical Deep Learning

www.doulos.com

john.aynsley@doulos.com