Deep Learning in Image Analysis

Sajith Kecheril Sadanandanan
sajith.ks@it.uu.se

Uppsala University

March 6-7, 2017
Outline

1. Part I - Introduction
2. Part II - Deep neural network
3. Part III - Deep learning fundamentals
4. Part IV - Deep neural network working
5. Part V - Demo
Part I - Introduction
Introduction

- Deep neural networks, the state-of-the-art in classification now.
- Deep learning algorithms are consistently winning the major competitions.
- Can learn hierarchical features from input.
A few applications

- Classification.
- Segmentation.
- Content based image retrieval.
- Scene recognition.
- Other image analysis tasks.
Object detection

Hui Li, et al., Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs. Jan 2016
Cell segmentation

Fig. 4. Result on the ISBI cell tracking challenge. (a) part of an input image of the “PhC-U373” data set. (b) Segmentation result (cyan mask) with manual ground truth (yellow border) (c) input image of the “DIC-HeLa” data set. (d) Segmentation result (random colored masks) with manual ground truth (yellow border).

Medical image segmentation

Konstantinos Kamnitsas et al., Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. February 2017
Super resolution

Playing Atari 2600 games

/Users/sajithks/Documents/seminars/workshop_cdac/beamer/images/nature14236-sv1.mov
Imagenet Large Scale Visual Recognition Challenge (ILSVRC)

- 1000 classes
- 1.2 million images
- From 2012 onwards all won by deep CNNs
Part II - Deep neural network
Neuron

dendrites

neuron

axon

to other neuron...

Artificial neuron

\[ \text{sum} = \sum_{i=1}^{n} x_i w_i \]

\[ \text{output} = f(\text{sum}) \]
Activation functions

\[
\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]

\[
sigmoid(x) = \frac{1}{1 + e^{-x}}
\]

\[
ReLU(x) = \max(0, x)
\]
Deep neural network

Types of deep neural networks

- Convolutional neural network.
- Recurrent neural network.
- Auto encoders.
Convolutional neural network

- Contains convolutional layers
- Only local connections
- Spatial relationship is preserved
- Parameter sharing
- Widely used in image analysis
Convolutional neural network

- Contains convolutional layers
- Only local connections
- Spatial relationship is preserved
- Parameter sharing
- Widely used in image analysis
Convolutional neural network

- Contains convolutional layers
- Only local connections
- Spatial relationship is preserved
- Parameter sharing
- Widely used in image analysis
Convolutional neural network

- Contains convolutional layers
- Only local connections
- Spatial relationship is preserved
- Parameter sharing
- Widely used in image analysis
2d convolutions

Layer 1

Layer 2
2d convolutions

Layer 1

Layer 2
2d convolutions

Layer 1

Layer 2
2d convolutions

Layer 1

Layer 2
2d convolutions

Layer 1

Layer 2
2d convolutions

Layer 1

Layer 2
2d convolutions

Layer 1

Layer 2
2d convolutions

Layer 1

Layer 2
2d convolutions

Layer 1

Layer 2
3d convolutions

- Filter coefficients are learned from data
- Can be implemented as matrix multiplication (faster)
- Efficient GPU implementations are possible
- Implemented as tensor multiplications/additions
- Hierarchical feature extraction
3d convolutions

- Filter coefficients are learned from data
- Can be implemented as matrix multiplication (faster)
- Efficient GPU implementations are possible
- Implemented as tensor multiplications/additions
- Hierarchical feature extraction
3d convolutions

- Filter coefficients are learned from data
- Can be implemented as matrix multiplication (faster)
- Efficient GPU implementations are possible
- Implemented as tensor multiplications/additions
- Hierarchical feature extraction
Filter visualization

First and second layer features of Alexnet

Filter visualization

Filter visualization

Lenet

Alexnet

Src. Alex Krishevsky et al, ImageNet Classification with Deep Convolutional Neural Networks, 2012
Googlenet

Src. Going deeper with convolutions [6]
Src. Going deeper with convolutions [6]
U-Net

Recurrent neural networks

- Connections between nodes form directed cycles.
- Create internal memory.
- Used to process sequence of data such as speech, text, video etc.
- Long short term memory (LSTM) commonly used now.
Auto encoders

- Unsupervised feature learning
- Uses dimensionality reduction
- Tries to recreate input after mapping to lower dimension
- Can stack multiple auto encoders to create deeper network

Src. by Chervinskii - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45555552
Part III - Deep learning fundamentals
How does the neural network learn?

- Learns from its mistakes.
- Contains hundreds of parameters/variables.
- Find the effect of each parameter in making mistakes.
- Increase/decrease the parameter values so as to make less mistakes.
- Do all the above several times.
How does the neural network learn?

- Learns from its mistakes. **Loss function**
- Contains hundreds of parameters/variables.
- Find the effect of each parameter in making mistakes.
- Increase/decrease the parameter values so as to make less mistakes.
- Do all the above several times.
How does the neural network learn?

- Learns from its mistakes. **Loss function**
- Contains hundreds of parameters/variables.
- Find the effect of each parameter in making mistakes. **Back propagation**
- Increase/decrease the parameter values so as to make less mistakes.
- Do all the above several times.
How does the neural network learn?

- Learns from its mistakes. **Loss function**
- Contains hundreds of parameters/variables.
- Find the effect of each parameter in making mistakes. **Back propagation**
- Increase/decrease the parameter values so as to make less mistakes. **Stochastic Gradient Descent**
- Do all the above several times.
How does the neural network learn?

- Learns from its mistakes. **Loss function**
- Contains hundreds of parameters/variables.
- Find the effect of each parameter in making mistakes. **Back propagation**
- Increase/decrease the parameter values so as to make less mistakes. **Stochastic Gradient Descent**
- Do all the above several times. **Iterations**
Components of deep learning

- Linear algebra.
- Optimization.
- Calculus.
- Information theory/Probability theory.
- Parallel programming.
Linear algebra

- Matrix operations.
- Tensor operations.
- Efficient routines like BLAS.
Optimization

- Loss functions.
- Optimization space.
- Find the parameters (millions).
- Stochastic Gradient Descent and its variants.
- Hyper parameters.
- Problems of over fitting, local minima, saddle points.
- Regularization.
Calculus

- Chain rule of differentiation.
- Matrix calculus.
- Base of back propagation algorithm.
Information theory/Probability theory

- Loss functions.
- Kullback-Leibler (KL) divergence.
Parallel programming

- GPU programming.
- CUDA, OpenCL, OpenMP.
- Nvidia, AMD, Google, Intel.
- Tensor processing units.
Part IV - Deep neural network working
Forward pass

Input → L1 → L2 → L3 → Loss function → Label
Forward pass

Input → L1 → L2 → L3 → Loss function → Label
Forward pass

Input → L1 → L2 → L3 → Loss function → Label
Forward pass

Input — L1 — L2 — L3 — Loss function
Forward pass

Input → L1 → L2 → L3 → Loss function → Loss
Backward pass
Backward pass

\[ \frac{\partial L_3}{\partial x} \quad \frac{\partial \text{Loss}}{\partial x} \]
Backward pass

Input \quad L1 \quad L2 \quad L3 \quad \text{Loss function}

\frac{\partial L2}{\partial x} \quad \frac{\partial L3}{\partial x} \quad \frac{\partial \text{Loss}}{\partial x}

Label
Backward pass

Parameter update

Input  L1  L2  L3  Loss function

\[ \frac{\partial L_1}{\partial x} \quad \frac{\partial L_2}{\partial x} \quad \frac{\partial L_3}{\partial x} \quad \frac{\partial \text{Loss}}{\partial x} \]
Backward pass

Parameter update

Label

Input \[\frac{\partial L_1}{\partial x}\] L1 \[\frac{\partial L_2}{\partial x}\] L2 \[\frac{\partial L_3}{\partial x}\] L3 \[\frac{\partial \text{Loss}}{\partial x}\] Loss function
Deep neural network - working

- Training.
- Loss function.
- Forward pass
- Backward pass/Back propagation.
- Gradient descent optimization.
Training

- Data partitioning - training, validation, testing.
- Preprocessing (0 mean unit std, image size).
- Network architecture.
- Hardware.
- Weight initialization.
  - Non zero
  - Non symmetrical
  - Gaussian distributions like Xavier, MSRA etc
Loss functions

- SVM loss
- Hinge loss
- Softmax cross-entropy loss
- etc
Follow chain rule.
Find the analytical gradient of each components
Find derivative of activation functions
Total gradient is the product of the gradient at that node multiplied with the gradient coming to the node.
If connected to multiple nodes, add the gradients
For ReLU activations gradient pass to positive outputs only
Back propagation

\[ f = q \ast z, \quad \frac{\partial f}{\partial q} = z, \quad \frac{\partial f}{\partial z} = q \]

\[ q = x + y, \quad \frac{\partial q}{\partial x} = 1, \quad \frac{\partial q}{\partial y} = 1 \]
Back propagation - contd (sigmoid)

\[ s = \frac{1}{1 + e^{-x}}, \quad \frac{\partial s}{\partial x} = s(1 - s) \]

\[ e^{-1} \times 0.53 \quad 1 \times -0.53 \quad \frac{1}{x^2} \times 1 \]
Gradient descent [5]

- Gradient descent is a way to minimize an objective function, $J(\theta)$
- Parameters $\theta \in \mathbb{R}^d$
- Gradient of the objective function $\nabla_\theta J(\theta)$
- Update in the opposite direction of function gradient
- Learning rate $\eta$
Gradient descent - variants [5]

- Batch gradient descent
- Stochastic gradient descent
- Mini-batch gradient descent
Batch gradient descent [5]

\[ \theta = \theta - \eta \cdot \nabla_\theta J(\theta) \]

- Calculate the gradients for the whole dataset to perform just one update.
- Can be very slow and is intractable for datasets that don’t fit in memory.
- Doesn’t allow to update the model online, i.e. with new examples on-the-fly.
Stochastic gradient descent [5]

\[ \theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \]

- Performs a parameter update for each training example \( x^{(i)} \) and label \( y^{(i)} \).
- Much faster and can also be used to learn online.
- Performs frequent updates with a high variance that cause the objective function to fluctuate heavily.
- Shows the same convergence behaviour as batch gradient descent if learning rate is decreased slowly.
Stochastic Gradient descent

Mini-batch gradient descent [5]

\[ \theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \]

- Performs an update for every mini-batch of \( n \) training examples.
- Reduces the variance of the parameter updates, which can lead to more stable convergence.
- Min-batch gradients can be computed efficiently using modern deep learning libraries.
Mini-batch gradient descent - challenges [5]

- Default mini-batch gradient descent does not guarantee good convergence.
- Difficult to choose proper $\eta$. If $\eta$ is small convergence is slow and if $\eta$ is large loss oscillates.
- Learning rate schedules, ie, adjust $\eta$ as learning progresses.
- Danger of trapping in local minima or saddle points.
- Many recent publications focus on saddle point problems.
Learning rate

- **Very high learning rate**
- **Low learning rate**
- **High learning rate**
- **Good learning rate**

loss vs. epoch
Mini-batch gradient descent - solutions [5]

- Momentum.
- Nesterov accelerated gradient.
- Adagrad.
- Adadelta.
- RMSprop. (unpublished, Hinton’s slides)
- Adam.
Momentum [4]

\[ v_t = \gamma v_{t-1} + \eta \nabla \theta J(\theta) \]
\[ \theta = \theta - v_t \]

- Adds a fraction \( \gamma \) of the update vector of the past time step to the current update vector.
- Accelerate SGD in the relevant direction and dampens oscillations.
Nesterov accelerated gradient \[3\]

- \( \mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \eta \nabla_{\theta} J(\theta - \gamma \mathbf{v}_{t-1}) \)
- \( \theta = \theta - \mathbf{v}_t \)
- Calculate the gradient not w.r.t. to our current parameters \( \theta \) but w.r.t. the approximate future position of our parameters.
Adagrad [1]

\[ \theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,i}} + \epsilon} \cdot g_{t,i} , \ \eta \text{ is modified} \]

- \( G_t \in \mathbb{R}^{d \times d} \) is a diagonal matrix where each diagonal element, \( i \), is the sum of the squares of the gradients of \( \theta_i \) up to time step \( t \), \( \epsilon \) is used to avoid division by zero.
- Adagrad eliminates tuning of \( \eta \)
- Different learning rate for each parameter
- Accumulation of squared gradient diminishes the learning rate to a very small value.
Adagrad [1]

- Adagrad uses a different learning rate for every parameter $\theta_i$ at every time step $t$
- $g_{t,i} = \nabla_{\theta} J(\theta_i)$, $g_{t,i}$ is the gradient of the objective function w.r.t. to the parameter $\theta_i$ at time step $t$
- $\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$, is the update for each parameter $\theta_i$ at time step $t$
- $\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$, $\eta$ is modified
Adagrad contd.[1]

- $G_t \in \mathbb{R}^{d \times d}$ is a diagonal matrix where each diagonal element, $i$, is the sum of the squares of the gradients of $\theta_i$ up to time step $t$, $\epsilon$ is used to avoid division by zero.
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$, in vectorized form
- Adagrad eliminates tuning of $\eta$
- Accumulation of squared gradient diminishes the learning rate to a very small value.
Adadelta [7]

$$\theta_{t+1} = \theta_t - \frac{RMS[\Delta \theta]_{t-1}}{RMS[g]_t} g_t$$

$$RMS[\Delta \theta]_t = \sqrt{E[\Delta \theta^2]_t + \epsilon}$$

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon}$$

- Running average of update over running average of gradient
- Eliminates the tuning of $\eta$
Adadelta [7]

- The sum of gradients is recursively defined as a decaying average of all past squared gradients.
  \[ E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma)g^2_t \]
- The running average \( E[g^2]_t \) at time step \( t \) depends only on the previous average and the current gradient.
  \[ \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t} + \epsilon} \odot g_t \]
- \( RMS[g]_t = \sqrt{E[g^2]_t} + \epsilon \)

- Since the units in this update do not match an exponentially decaying average of squared parameter update is introduced in place of $\eta$.

\[ E[\Delta \theta^2]_t = \gamma E[\Delta \theta^2]_{t-1} + (1 - \gamma) \Delta \theta^2_t \]

\[ RMS[\Delta \theta]_t = \sqrt{E[\Delta \theta^2]_t + \epsilon} \]

\[ \theta_{t+1} = \theta_t - \frac{RMS[\Delta \theta]_{t-1}}{RMS[g]_t} g_t \]
RMSprop [7]

\[ \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \]

\[ E[g^2]_t = 0.9 E[g^2]_{t-1} + 0.1 g_t^2 \]

RMSprop is an unpublished, adaptive learning rate method proposed by Geoff Hinton in Lecture 6e of his Coursera Class.

A fraction of previous update is added to current update.
Adam [2]

- Uses bias corrected mean and variance of gradients.
  \[
  \hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}
  \]
  \[
  m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t
  \]
  \[
  v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2
  \]
- \( m_t \) and \( v_t \) are the mean and variance of gradients and \( \hat{m}_t \) and \( \hat{v}_t \) are the bias corrected mean and variance.
Adam [2]

\[ \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \]

\[ \hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \]

\[ m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \]

\[ v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \]

\[ m_t \] and \[ v_t \] are the mean and variance of gradients and \[ \hat{m}_t \] and \[ \hat{v}_t \] are the bias corrected mean and variance.
SGD optimization on loss surface contours
SGD optimization on saddle point
Regularization

- To reduce over-fitting to training data.
- Data augmentation.
- Drop out.
- Batch normalization.
- Batch wise learning (mini batch).
- Local response normalization.
- $L_1$, $L_2$ regularization of weights.
Hyper parameters

- Learning rate
- Learning rate decay
- Momentum
- Network depth
- Network width
- Mini batch size
- etc.
Deep learning tools

- Caffe
- Torch
- Theano
- Tensorflow
- Keras
- Neon
- Chainer
- etc...
Part V - Demo
Demo

http://cs.stanford.edu/people/karpathy/convnetjs/
Further reads/links

- Machine learning by Andrew Ng
  https://www.youtube.com/playlist?list=PLZ9qNFMHZ-A4rycgrg0Yma6zzF4BZGGPW

- CS231n deep learning course by Fei Fei’s group
  https://www.youtube.com/watch?v=g-PvXUjD6qg&list=PLlJy-eBtNFt6EuMxFYRiNRS07MCWN5UIA&index=1

- Recent deep learning summer school at Montreal

- Yoshua Bengio’s book on deep learning
  http://www.deeplearningbook.org/

- Stat212b: Topics Course on Deep Learning
  http://joanbruna.github.io/stat212b/

- Gradient-based learning applied to document recognition

- https://arxiv.org
References I

Adaptive subgradient methods for online learning and stochastic optimization.  

Adam: A method for stochastic optimization.  

A method of solving a convex programming problem with convergence rate o (1/k^2).  

On the momentum term in gradient descent learning algorithms.  

An overview of gradient descent optimization algorithms.  

Going deeper with convolutions.  

ADADELTA: an adaptive learning rate method.  