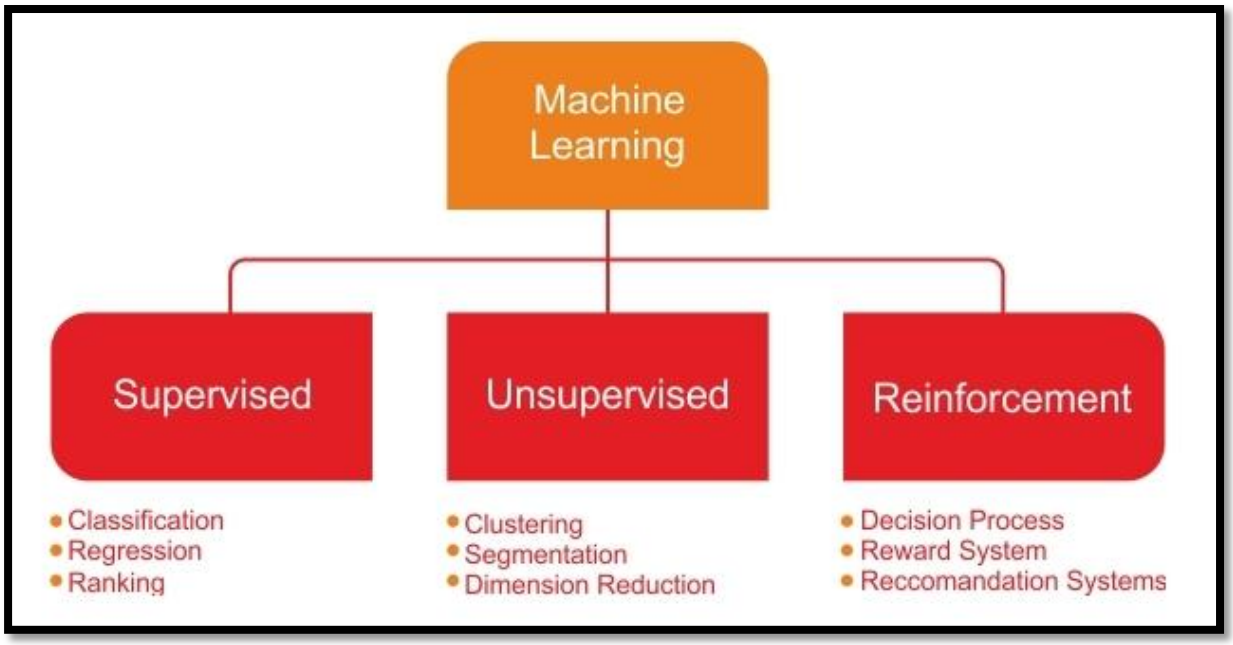
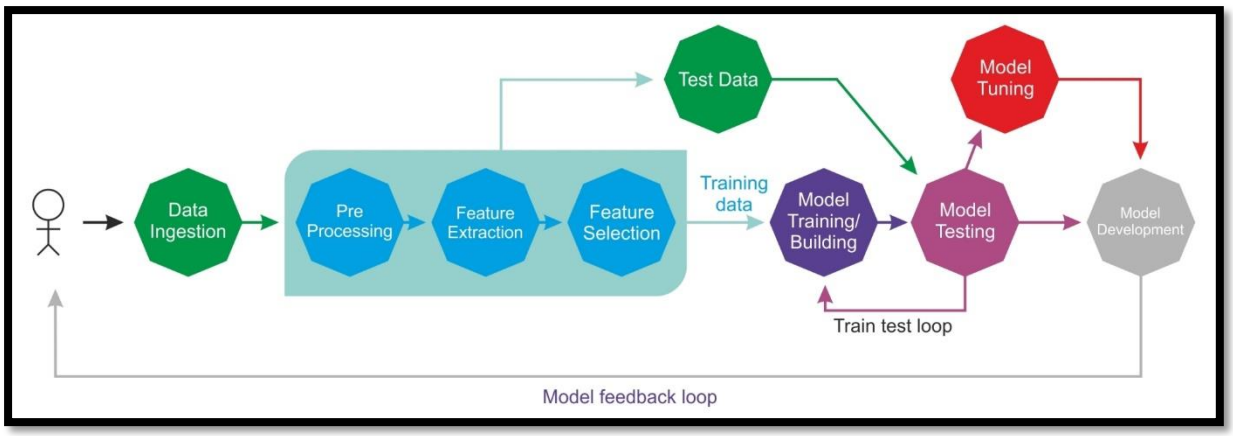
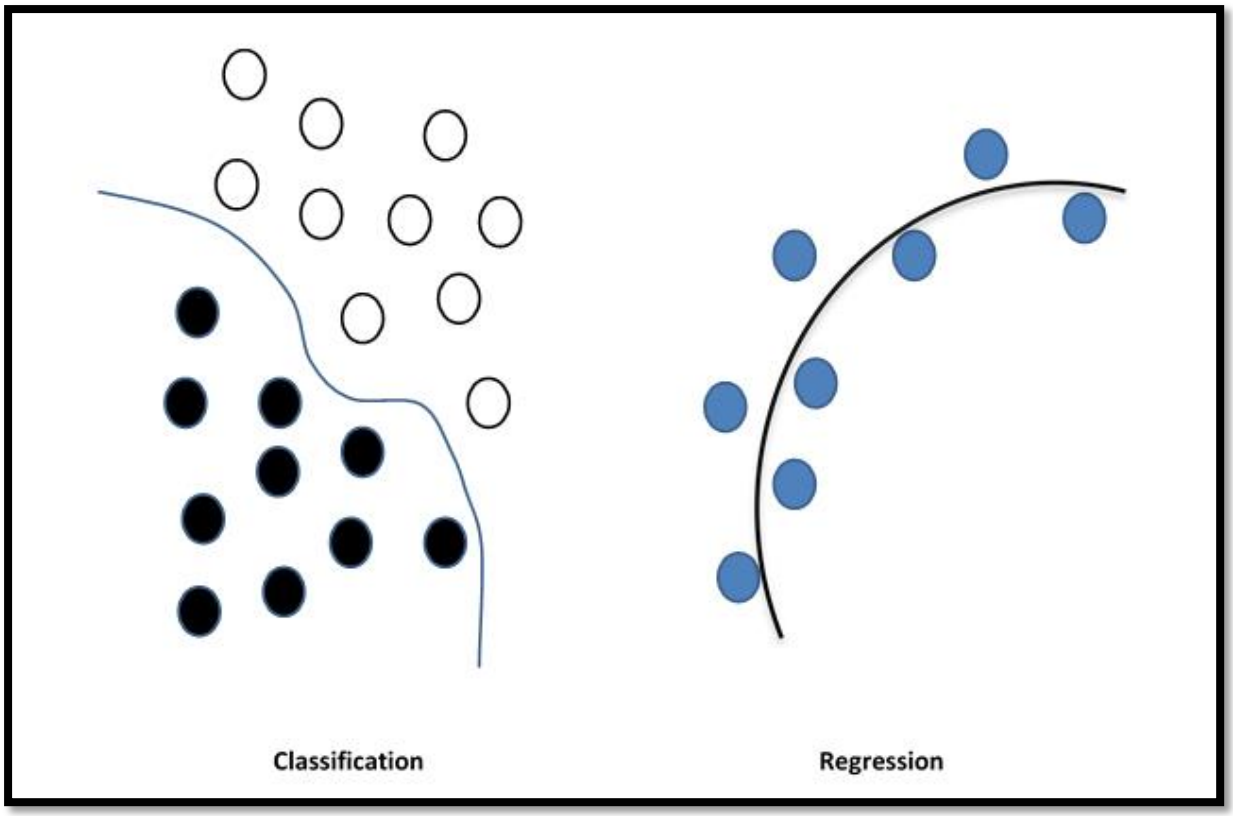
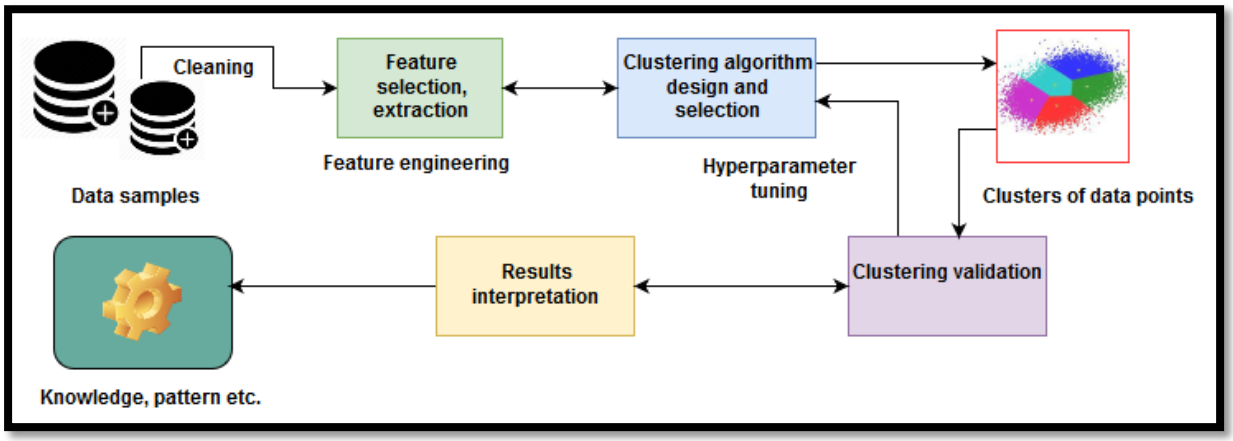
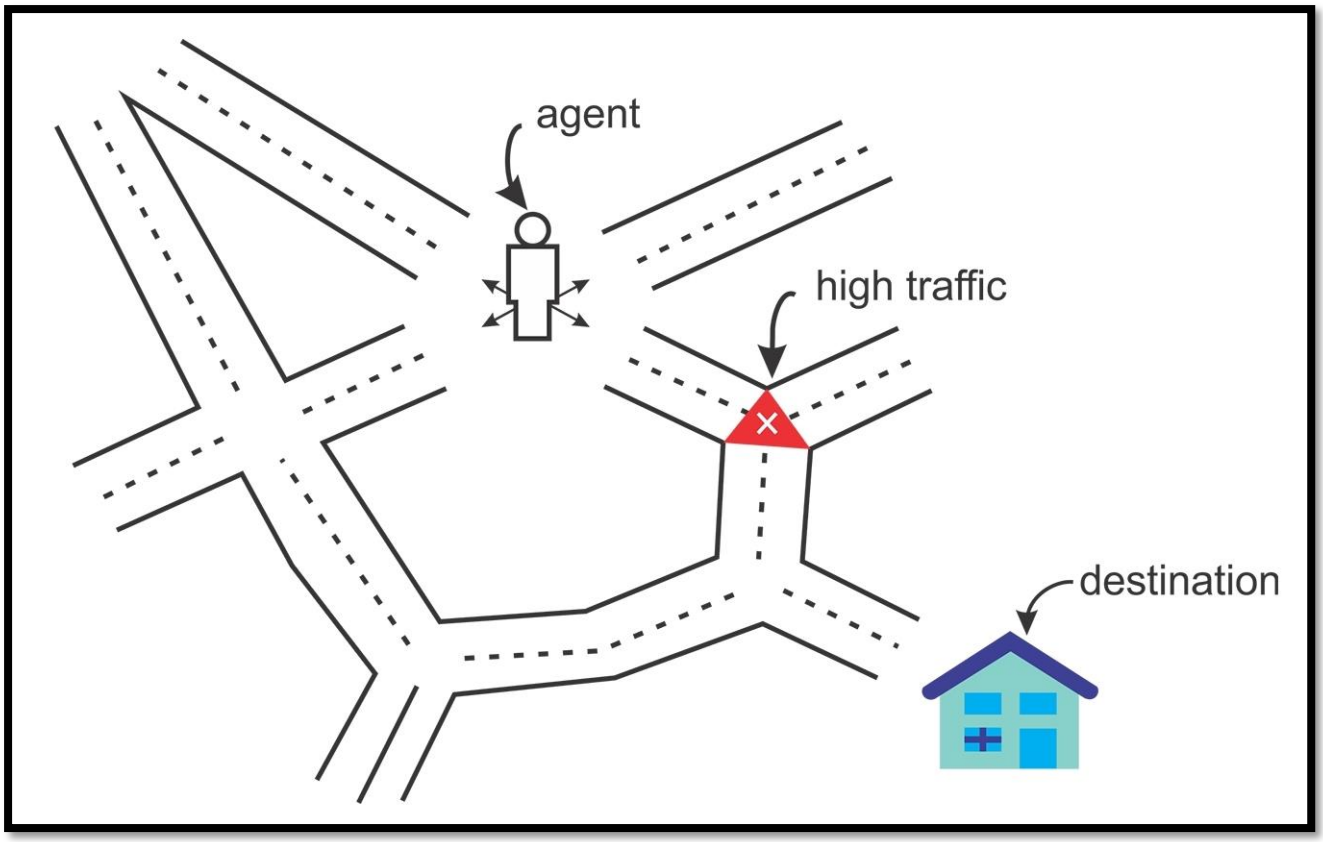
Chapter 1: Getting Started with Deep Learning

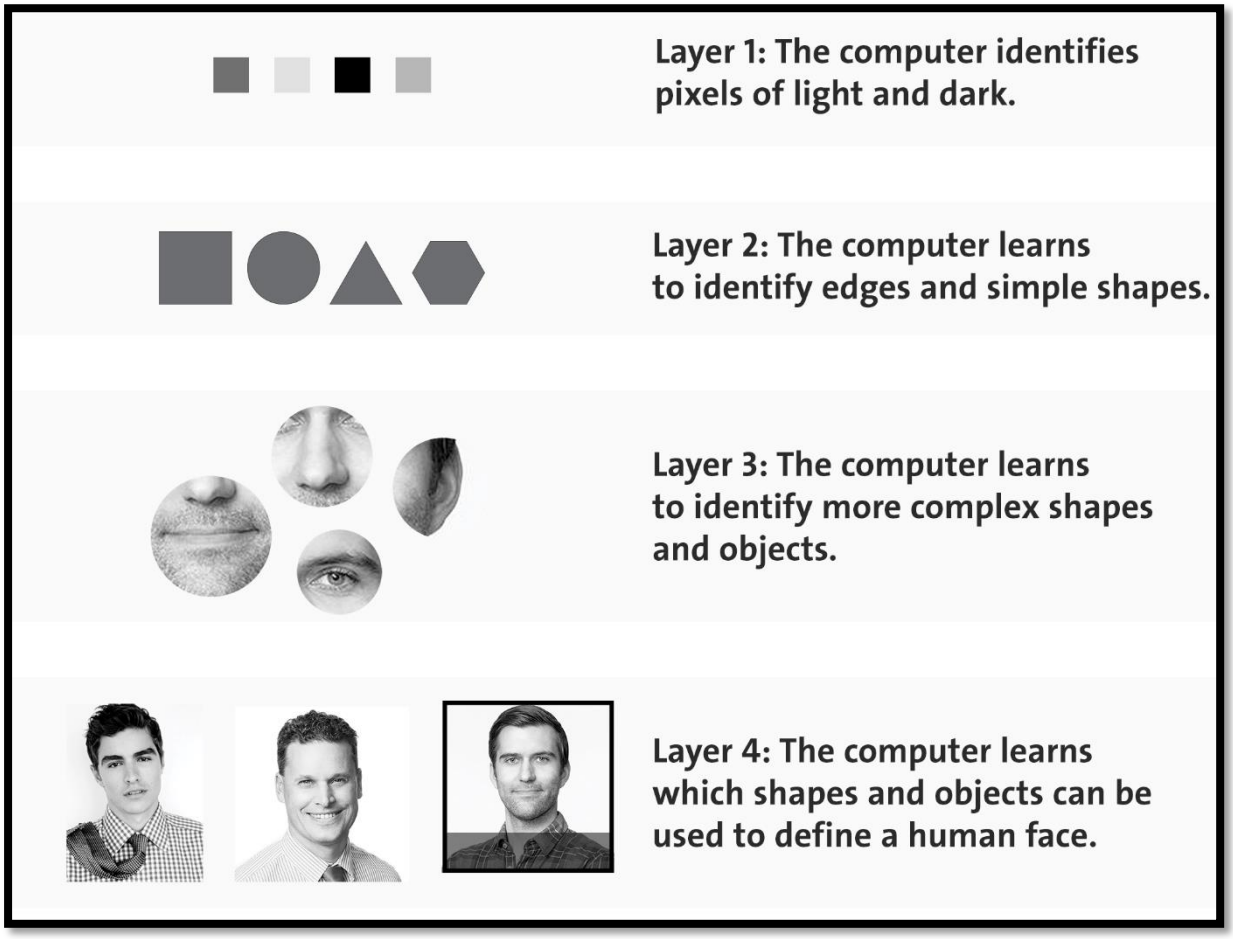


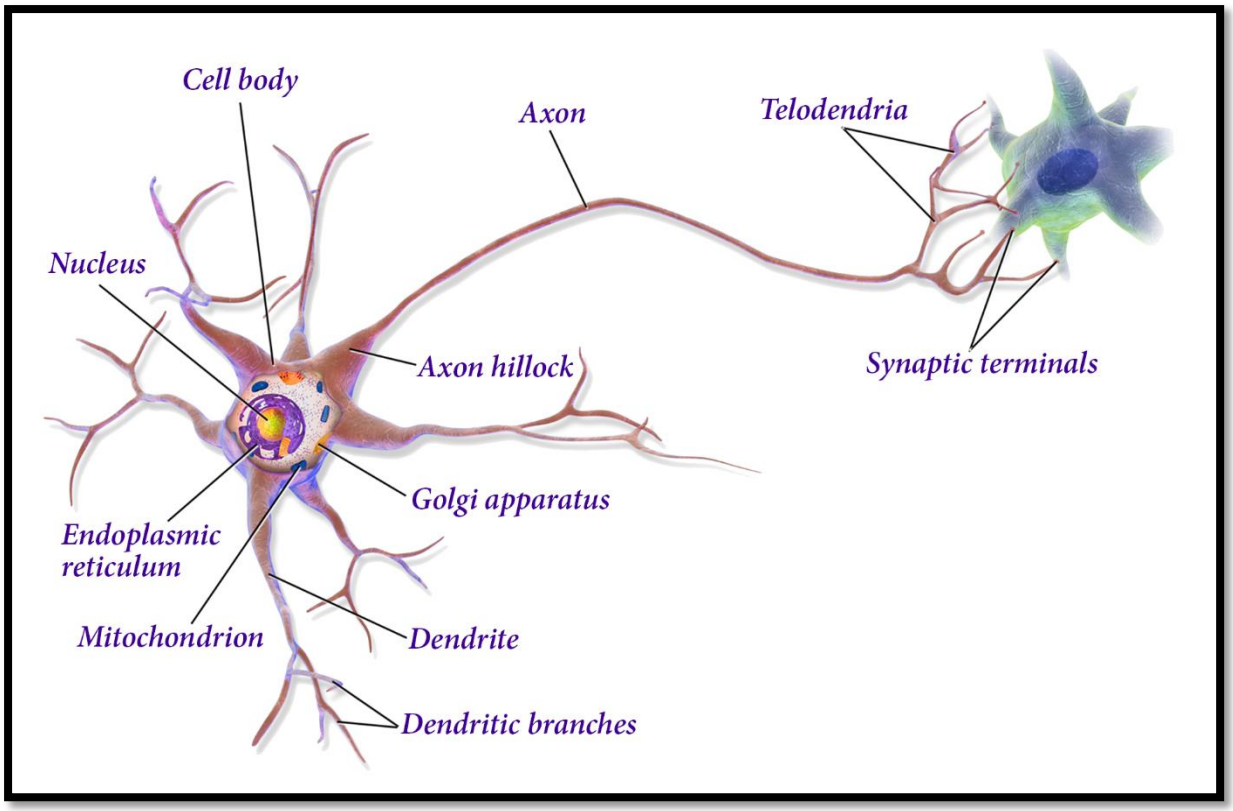


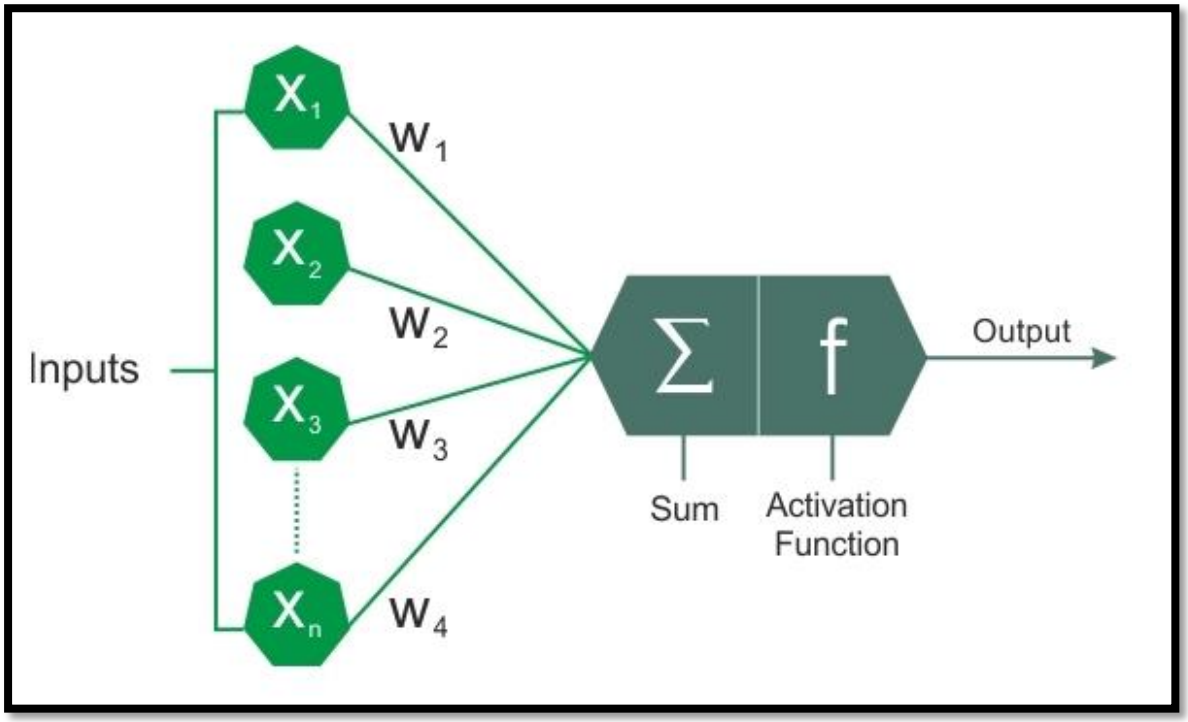

Classification
Regression

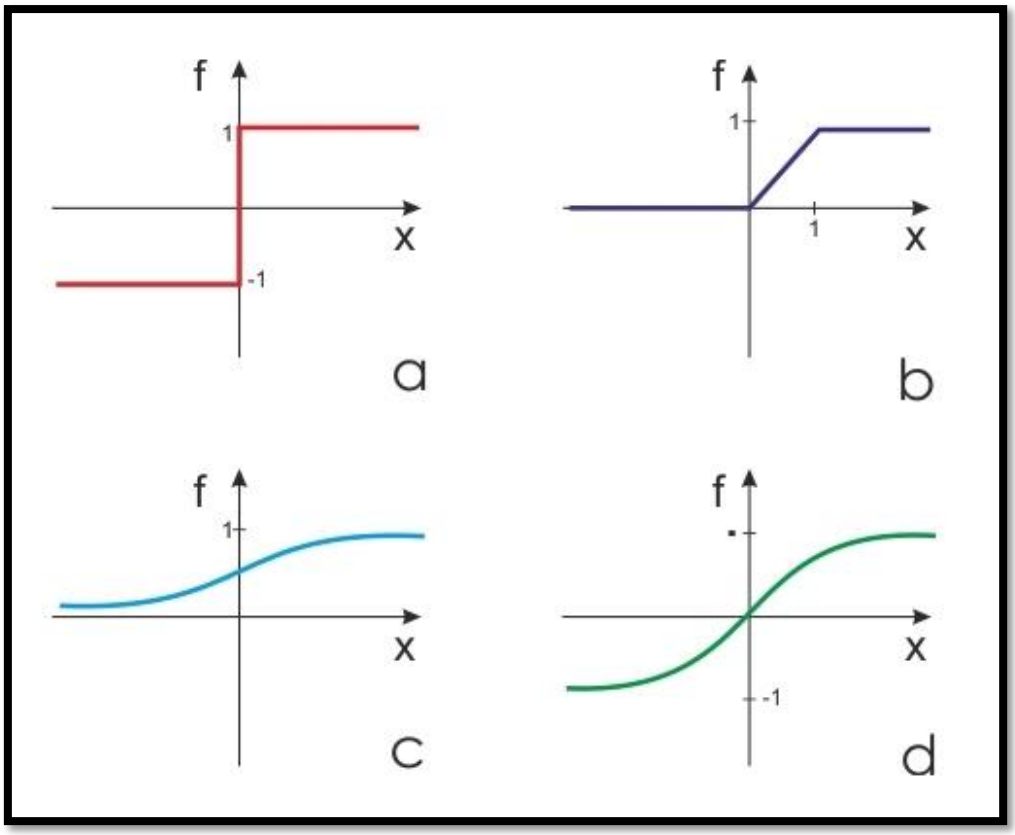

Cleaning
Feature selection, extraction
Clustering algorithm design and selection
Data samples
Feature engineering
Hyperparameter tuning
Clusters of data points
Results interpretation
Clustering validation
Knowledge, pattern etc.

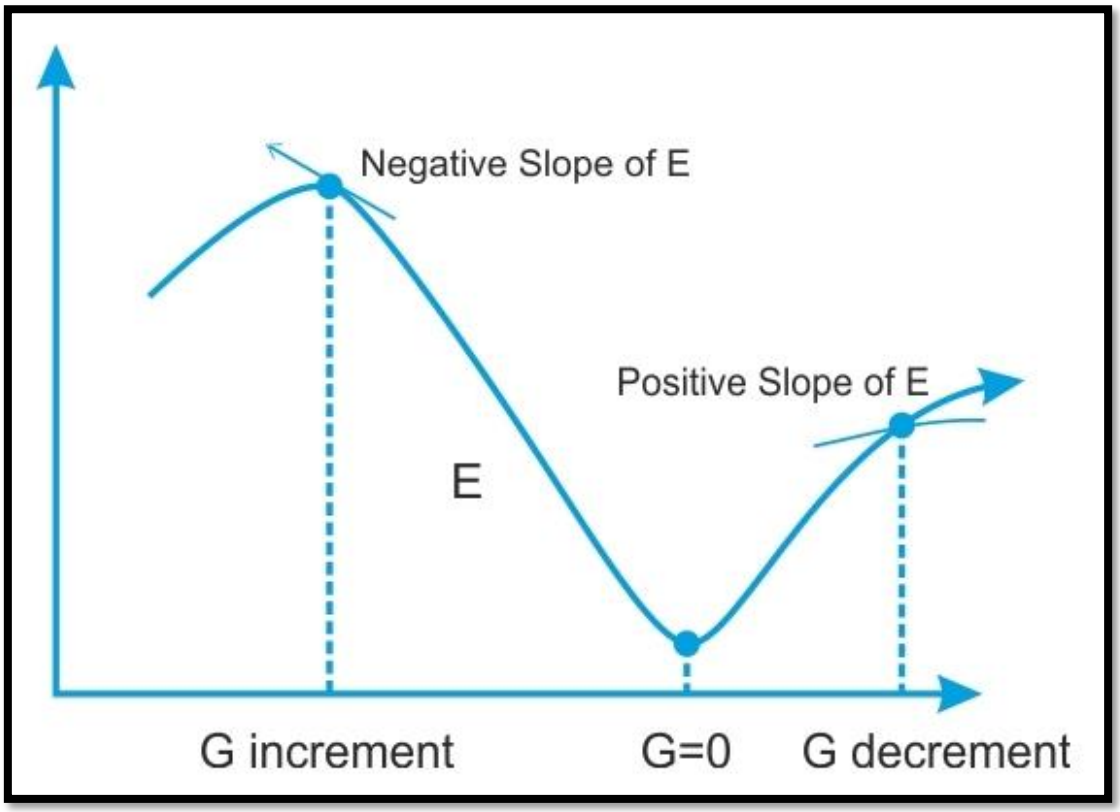

agent
high traffic
destination

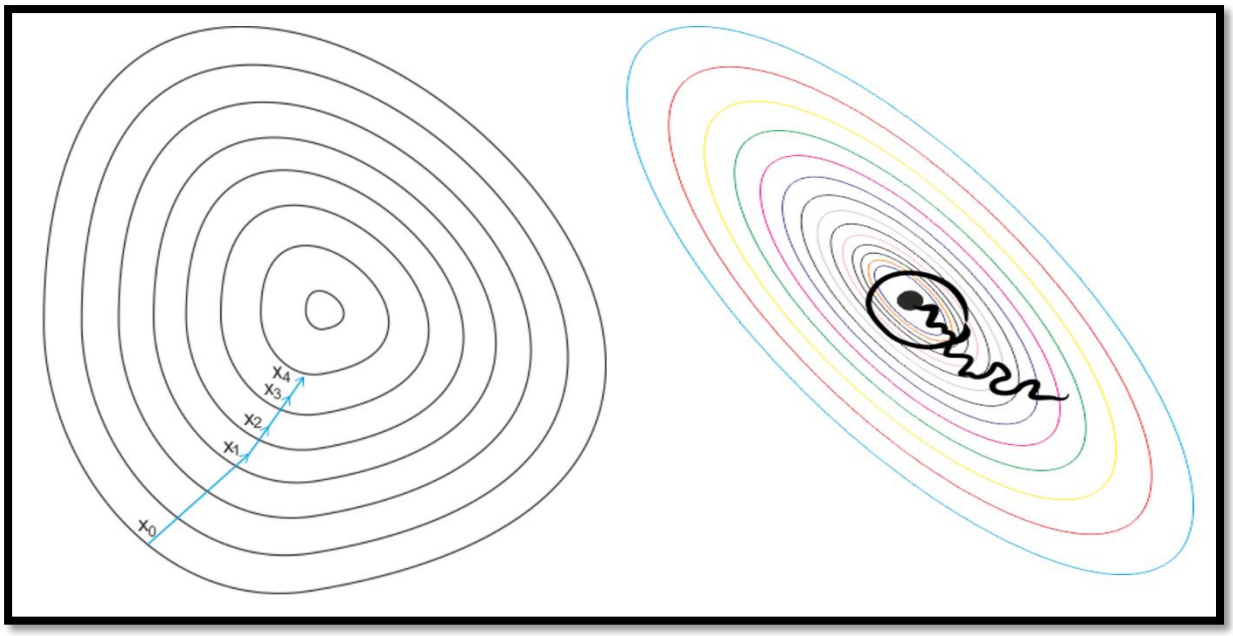

Layer 1: The computer identifies pixels of light and dark.
Layer 2: The computer learns to identify edges and simple shapes.
Layer 3: The computer learns to identify more complex shapes and objects.
Layer 4: The computer learns which shapes and objects can be used to define a human face.

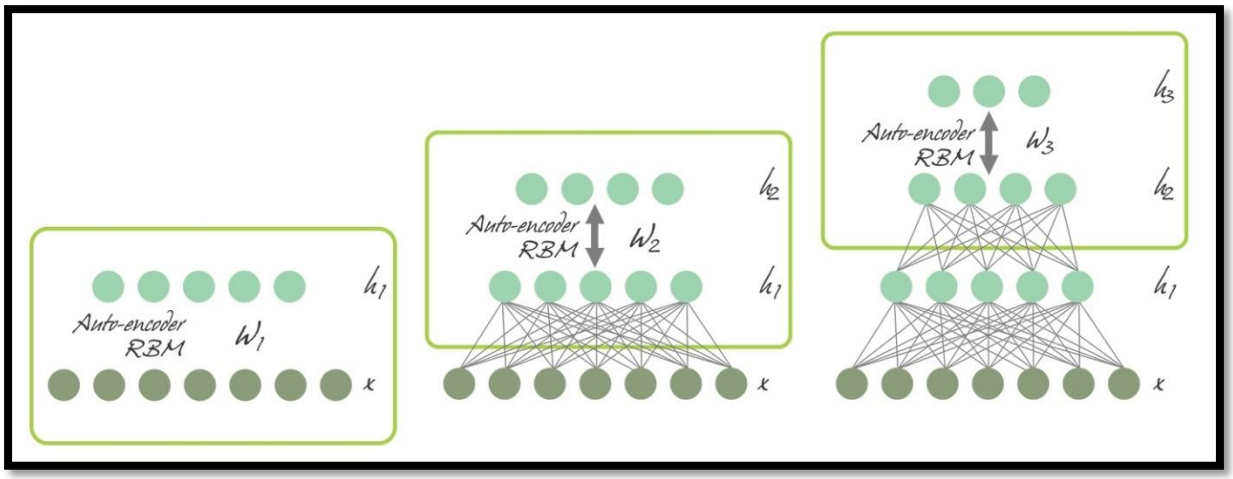

Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic reticulum
Mitochondrion
Dendrite
Dendritic branches

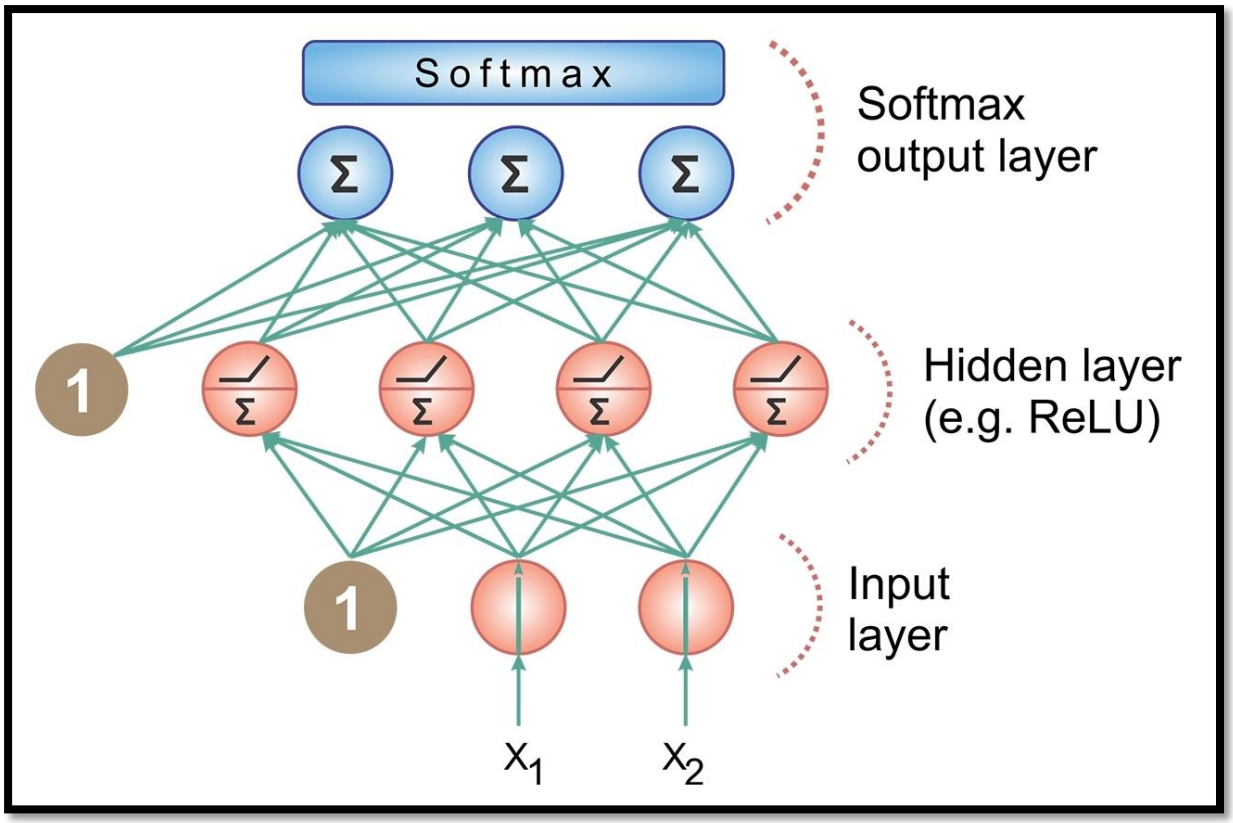

X1
X2
X3
Xn
W1
W2
W3
W4
Inputs
Σ
f
Output
Sum
Activation Function

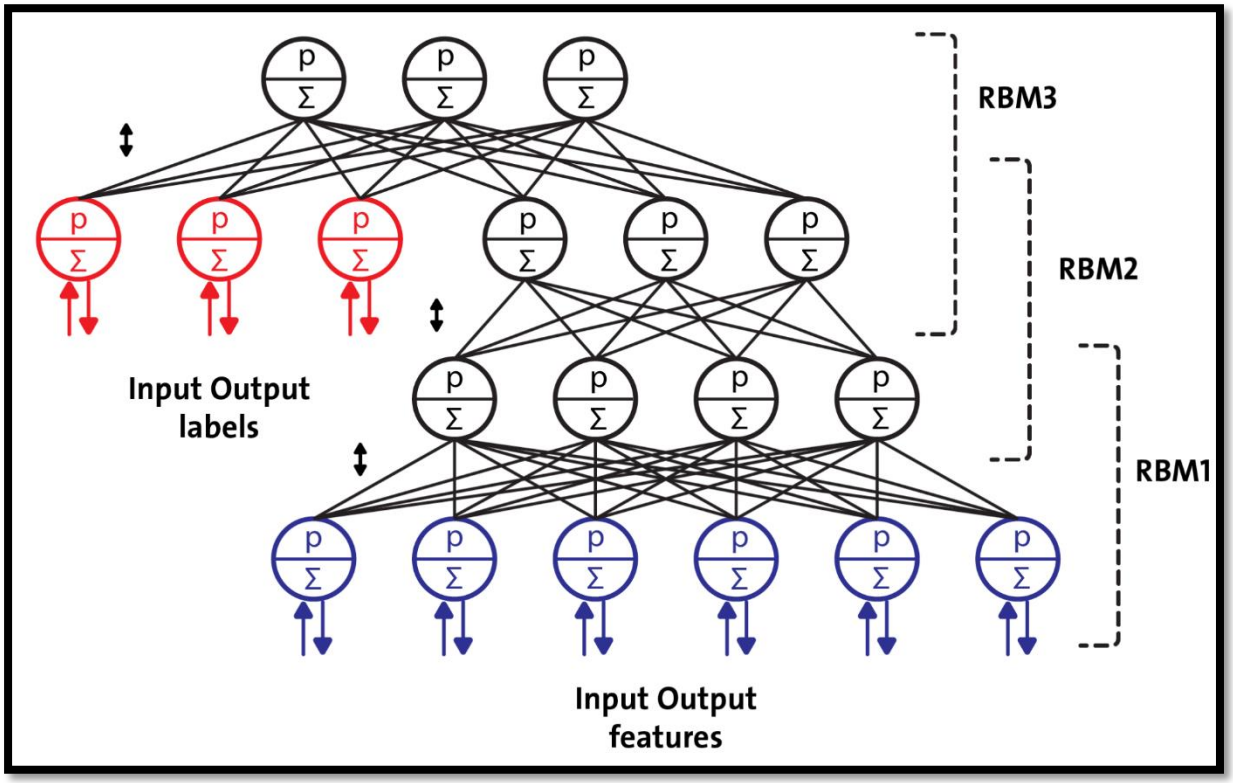

f
1
x
-1
a
f
1
1
x
b
f
1
x
c
f
-1
x
d

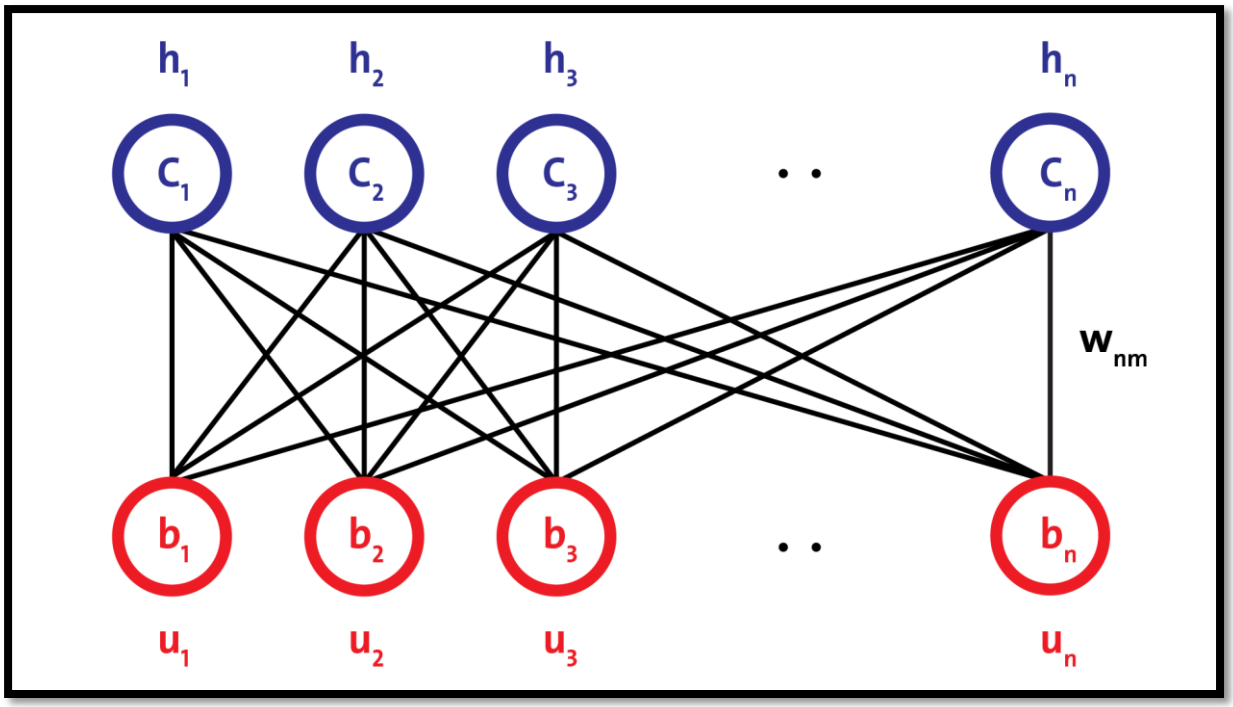

Negative Slope of E
Positive Slope of E
E
G increment
G=0
G decrement

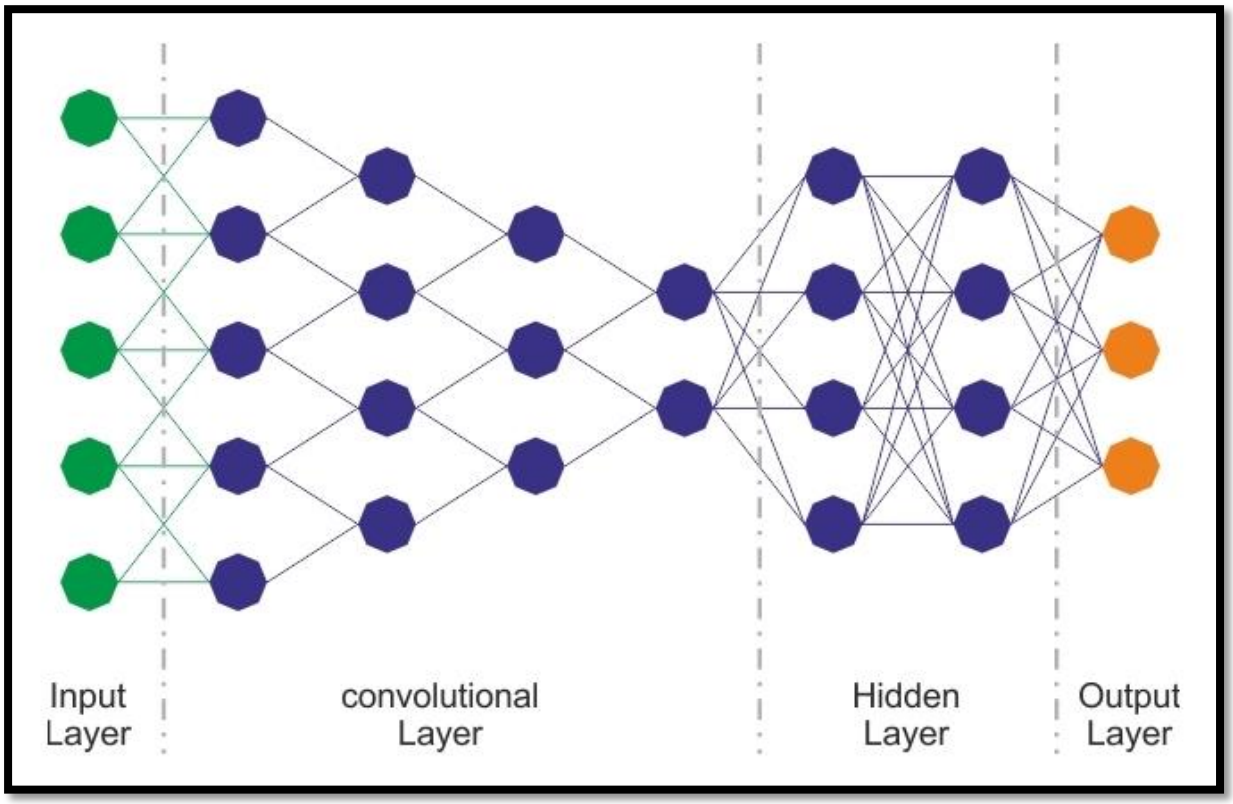

x4
x3
x2
x1
x0

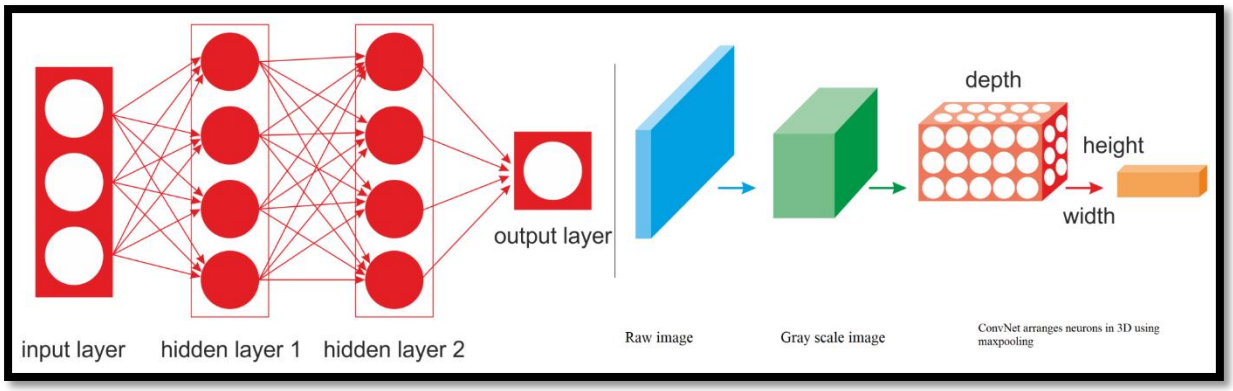

Auto-encoder
RBM
W1
h1
x
Auto-encoder
RBM
W2
h2
h1
x
Auto-encoder
RBM
W3
h3
h2
h1
x

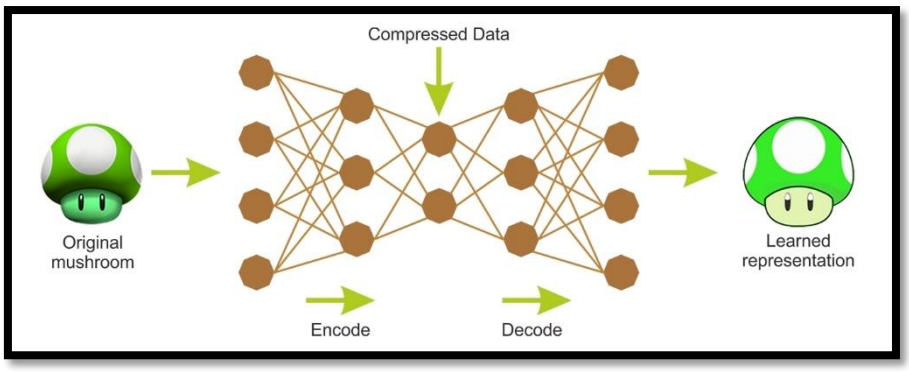

Softmax
Softmax output layer
Hidden layer (e.g. ReLU)
Input layer
1
1
x_1
x_2

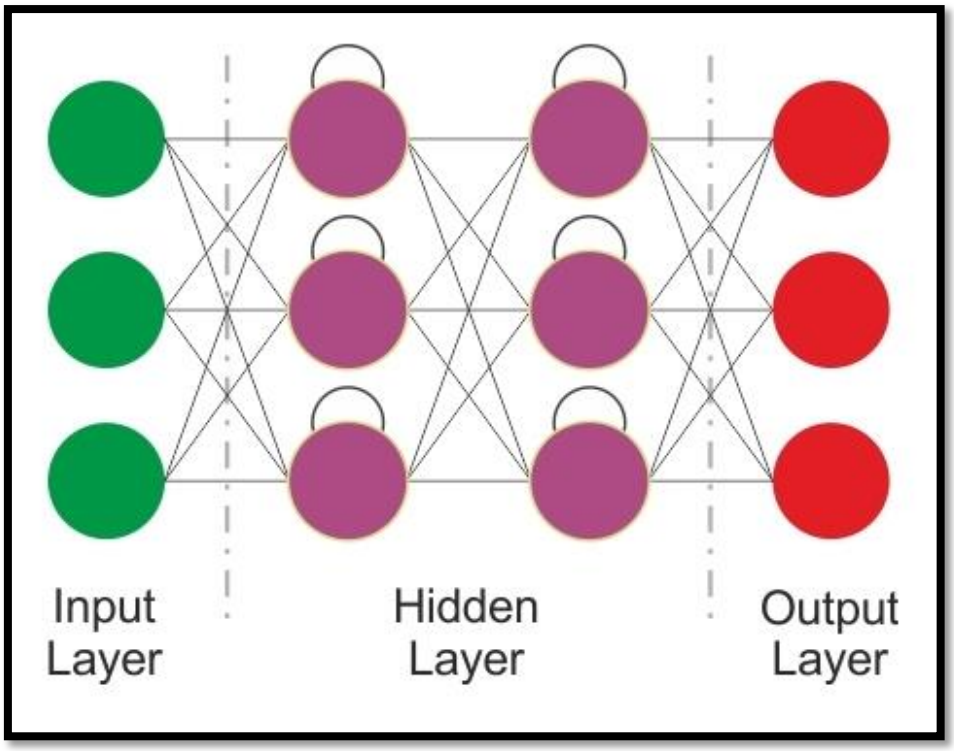

RBM3
RBM2
RBM1
Input Output labels
Input Output features


h1
h2
h3
hn
C1
C2
C3
Cn
wnm
b1
b2
b3
bn
u1
u2
u3
un


Input Layer
convolutional Layer
Hidden Layer
Output Layer

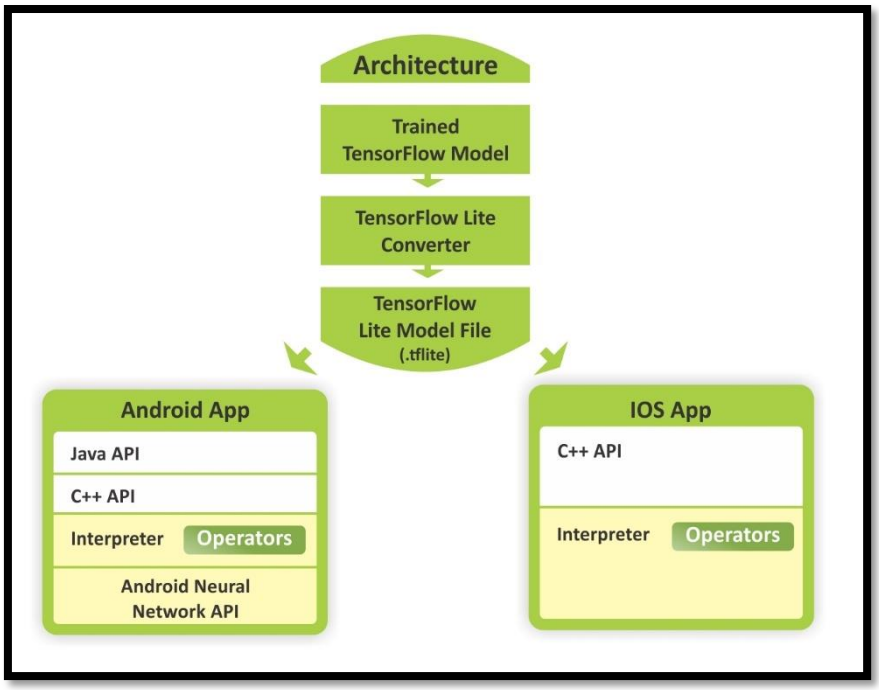

input layer
hidden layer 1
hidden layer 2
output layer
Raw image
Gray scale image
depth
height
width
ConvNet arranges neurons in 3D using maxpooling

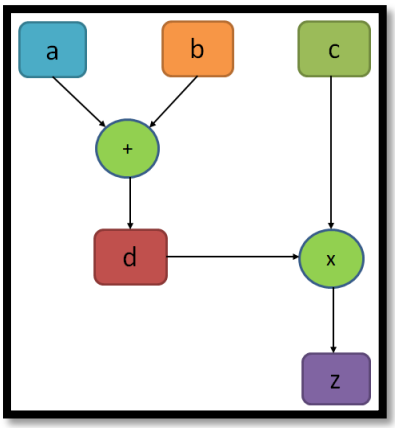

Compressed Data
Original mushroom
Encode
Decode
Learned representation

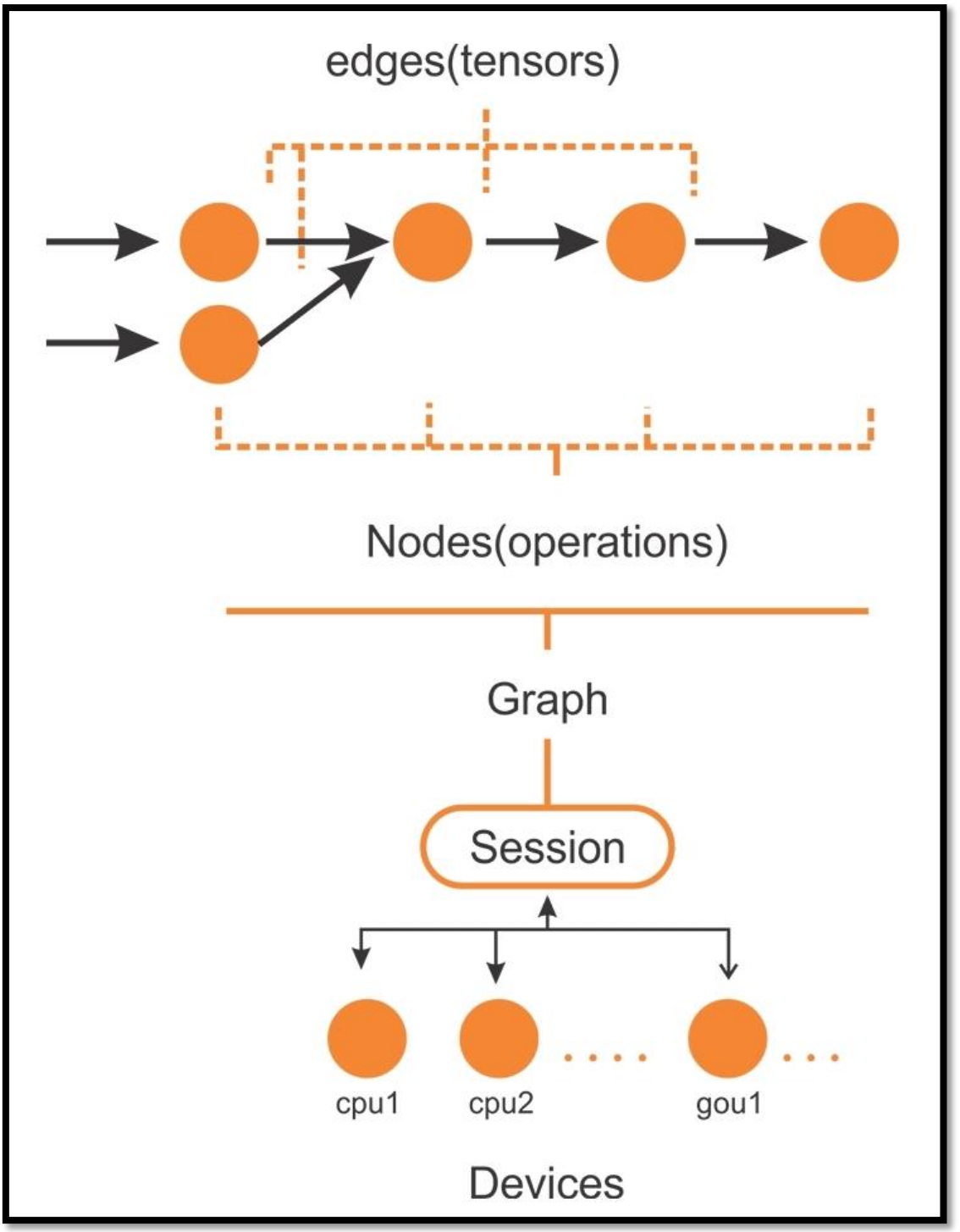


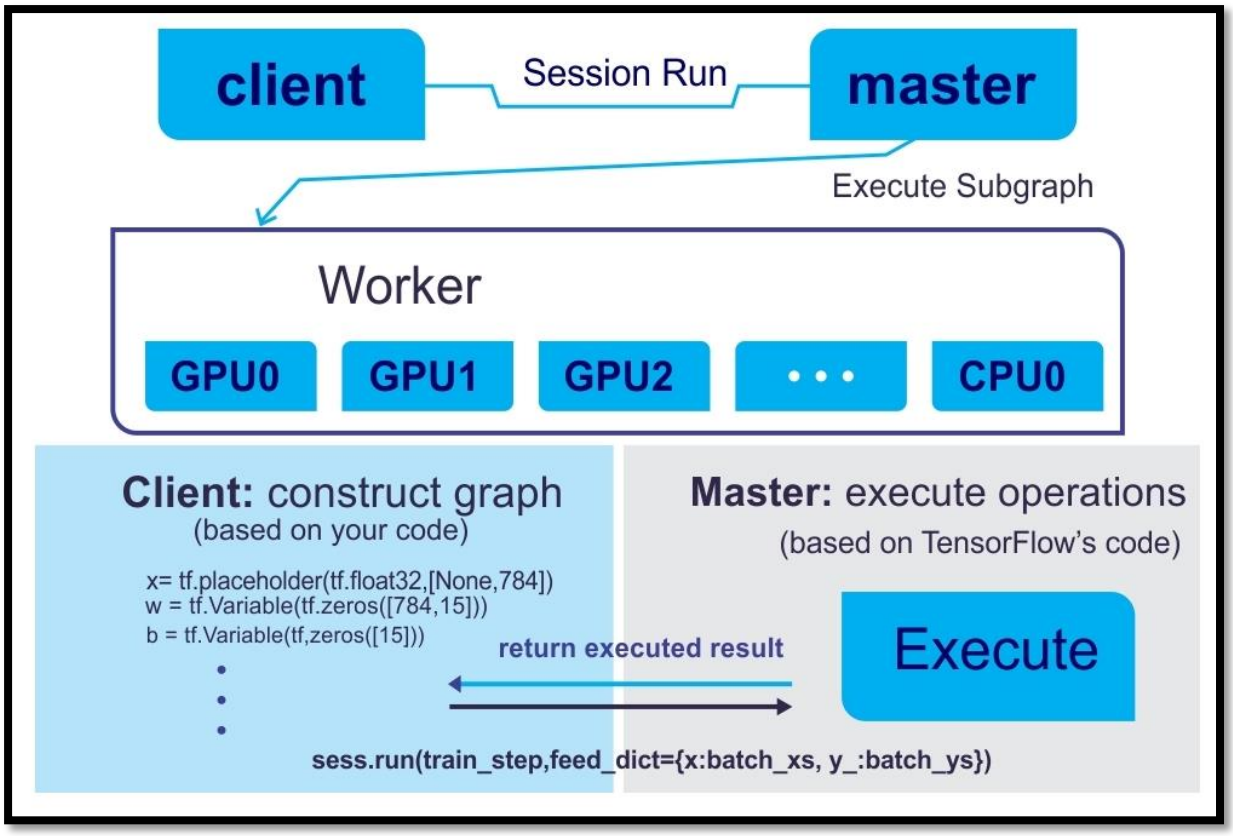
Framework	Supported programming languages	Training materials community	CNN modelling capability	RNN modelling capability	Usability	Multi-GPU support
Theano	Python, C++	++	Ample CNN tutorials and prebuilt models	Ample RNN tutorials and prebuilt models	Modular architecture	No
Neon	Python	+	Fastest tools for CNN	Mininmal resources	Modular architecture	No
Torch	Lua, Python	+	Mininmal resources	Ample RNN tutorials and prebuilt models	Modular architecture	Yes
Caffe	C++	++	Ample CNN tutorials and prebuilt models	Mininmal resources	Creating layers takes time	Yes
MXNet	R.Python,Julia, Scala	++	Ample CNN tutorials and prebuilt models	Mininmal resources	Modular architecture	Yes
CNTK	C++	+	Ample CNN tutorials and prebuilt models	Ample RNN tutorials and prebuilt models	Modular architecture	Yes
TensorFlow	Python, C++	+++	Ample RNN tutorials and prebuilt models	Ample RNN tutorials and prebuilt models	Modular architecture	Yes
Deep Learning4j	Java, Scala	+++	Ample RNN tutorials and prebuilt models	Ample RNN tutorials and prebuilt models	Modular architecture	Yes
Keras	Python	+++	Ample RNN tutorials and prebuilt models	Ample RNN tutorials and prebuilt models	Modular architecture	Yes

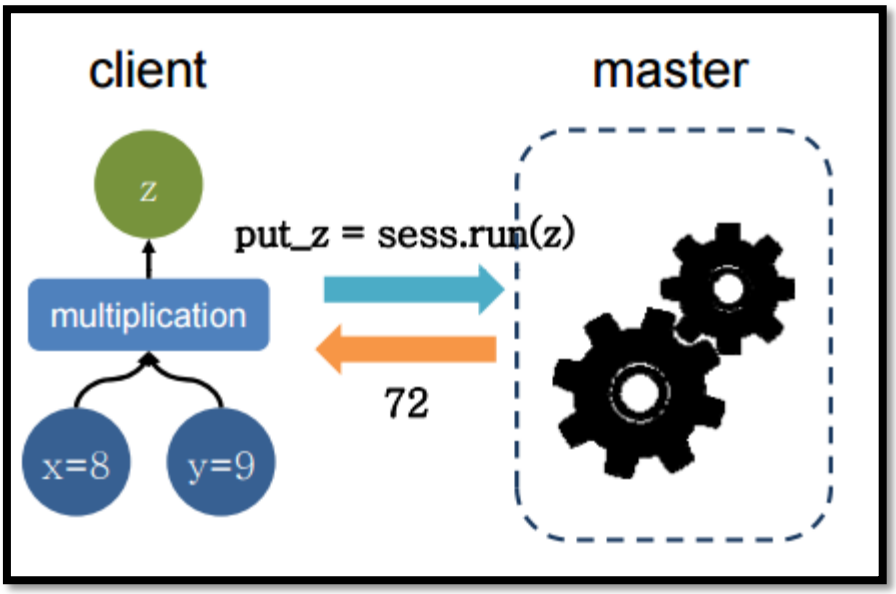
Chapter 2: A First Look at TensorFlow

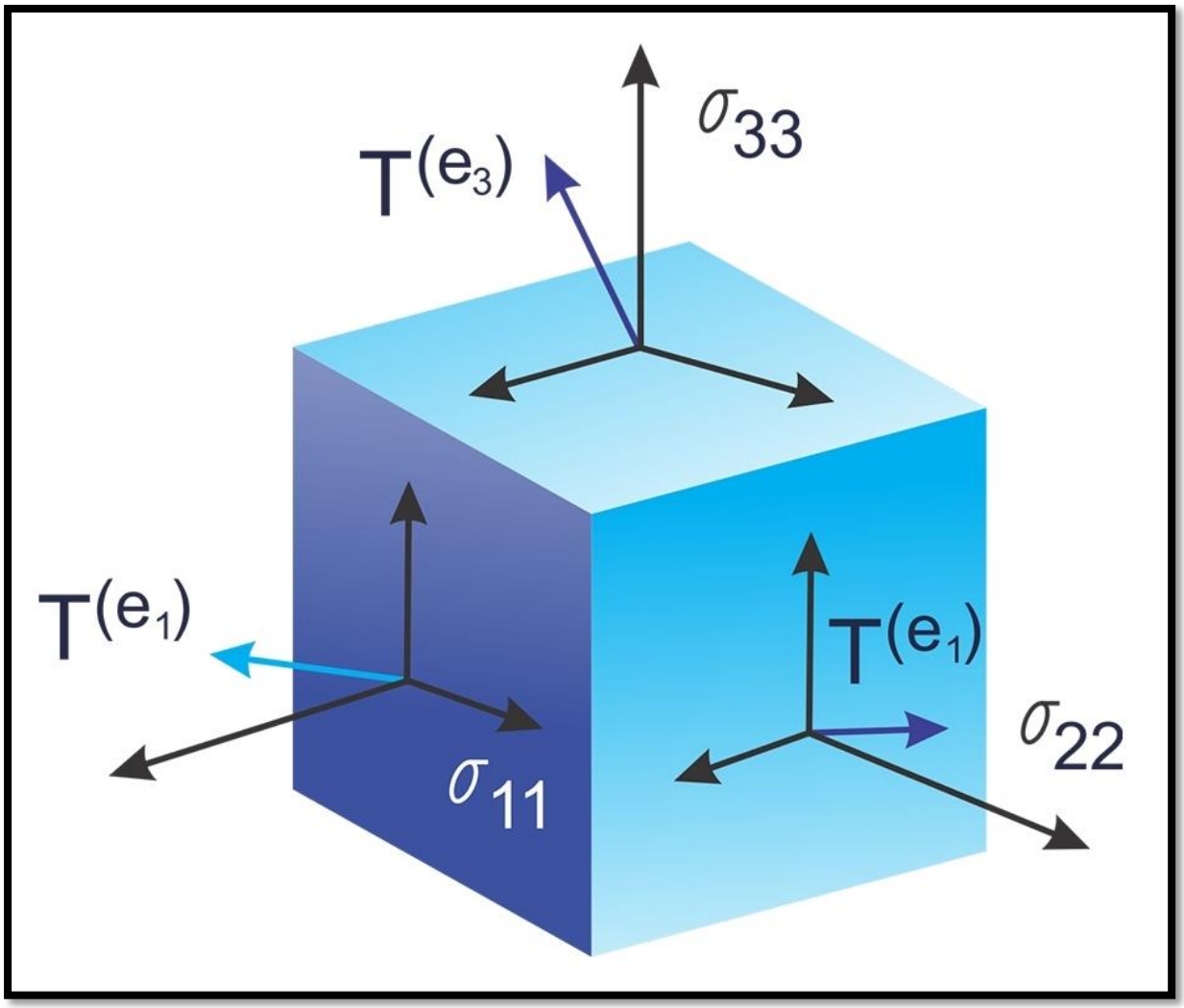




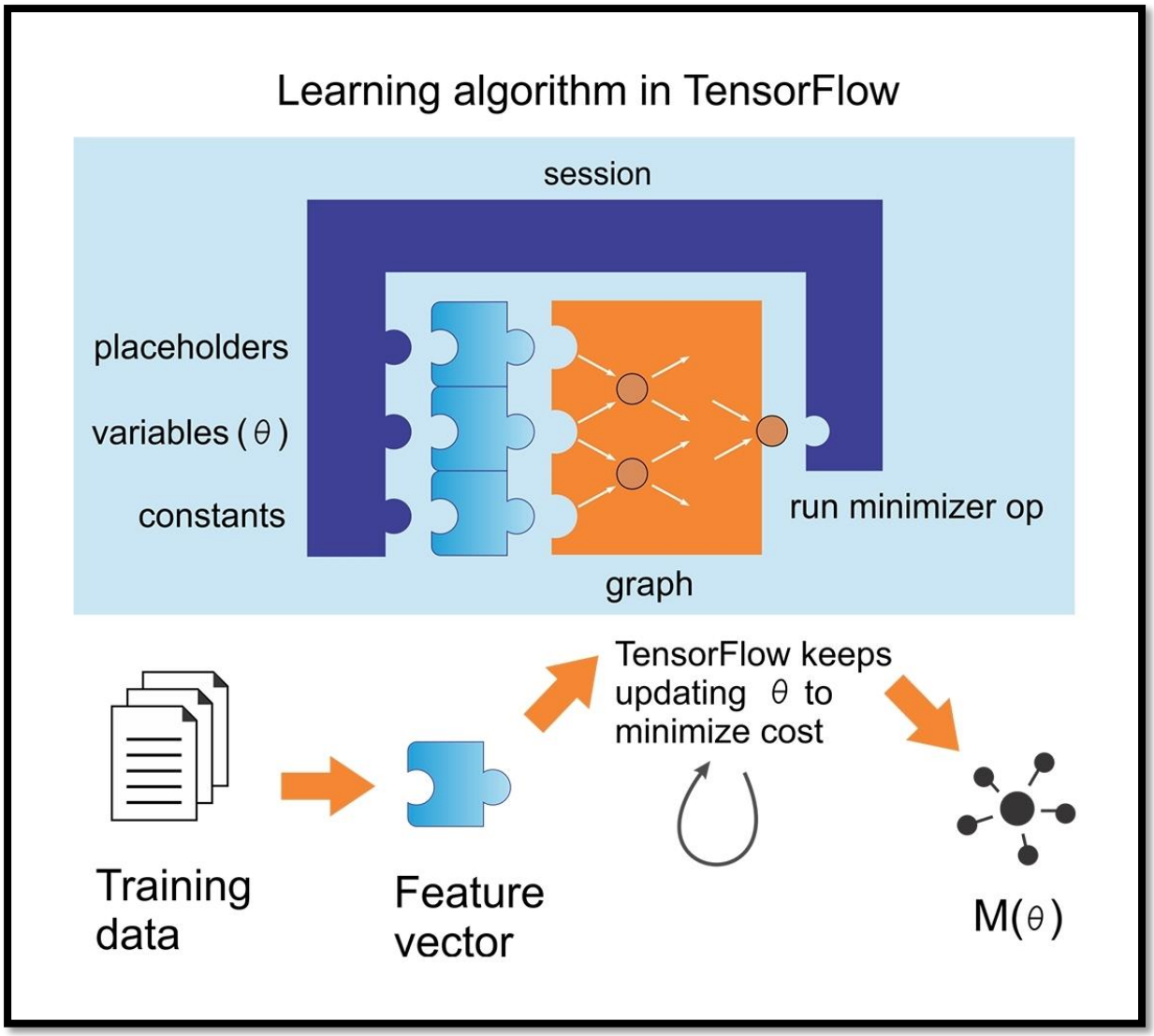

edges(tensors)
Nodes(operations)
Graph
Session
cpu1
cpu2
gou1
Devices

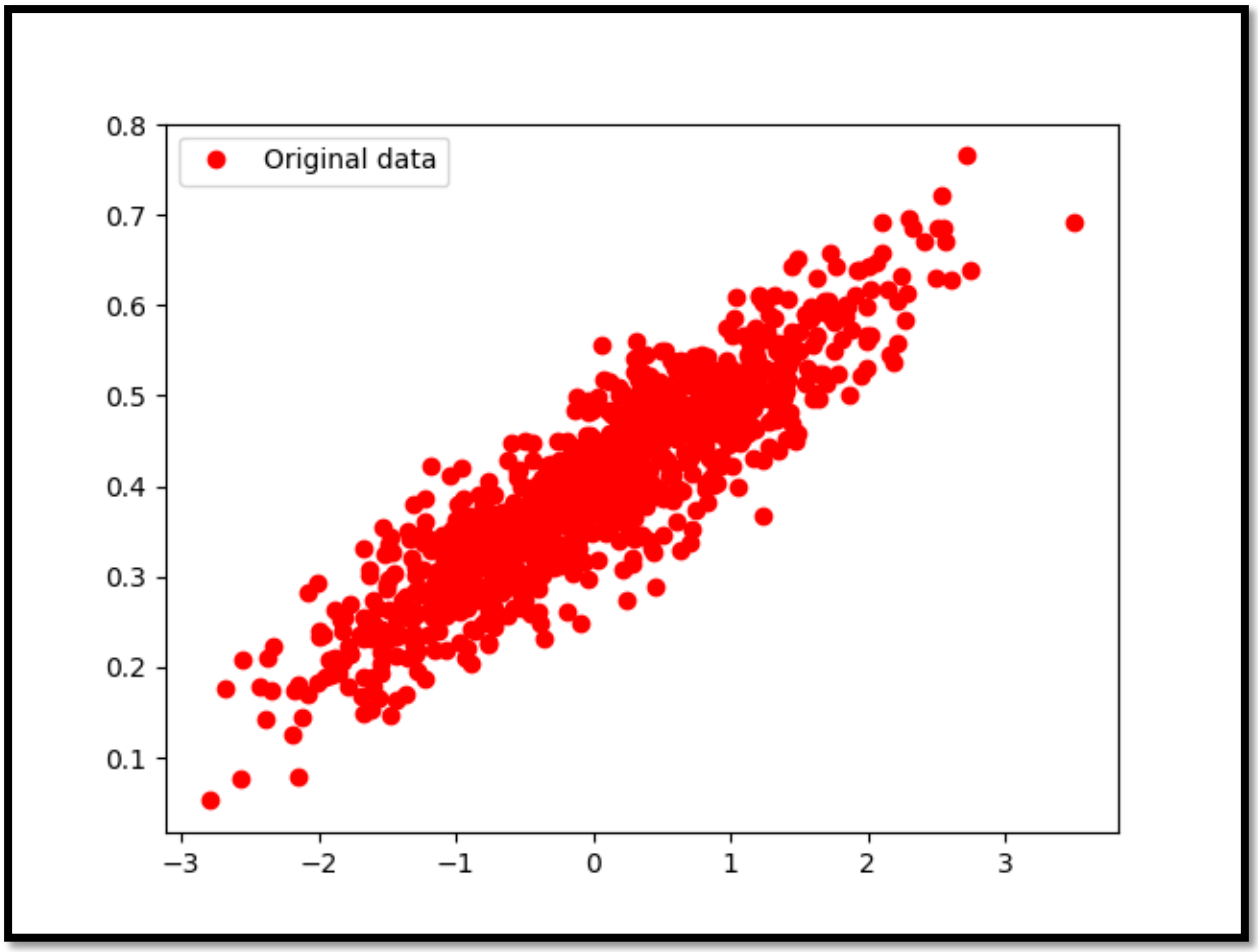

client
Session Run
master
Execute Subgraph
Worker
GPU0
GPU1
GPU2
• • •
CPU0
Client: construct graph
(based on your code)
x= tf.placeholder(tf.float32,[None,784])
w = tf.Variable(tf.zeros([784,15]))
b = tf.Variable(tf,zeros([15]))
return executed result
Master: execute operations
(based on TensorFlow's code)
Execute
sess.run(train_step,feed_dict={x:batch_xs, y_:batch_ys})

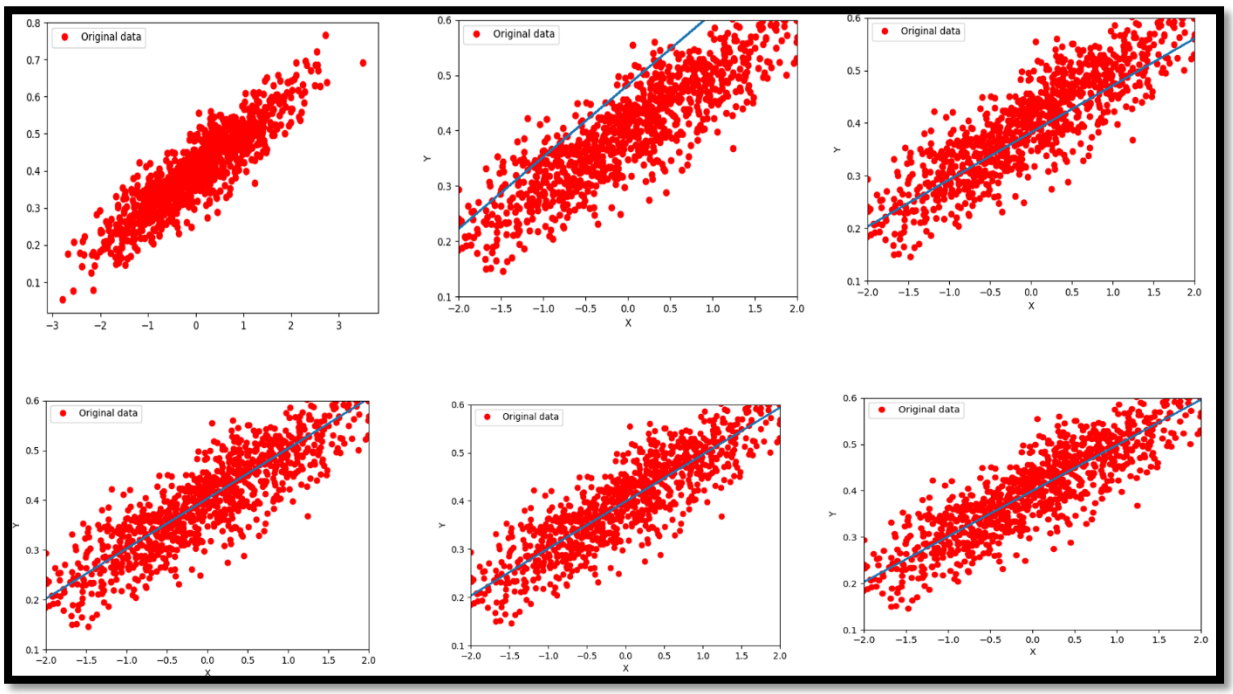

client
master
z
put_z = sess.run(z)
multiplication
72
x=8
y=9

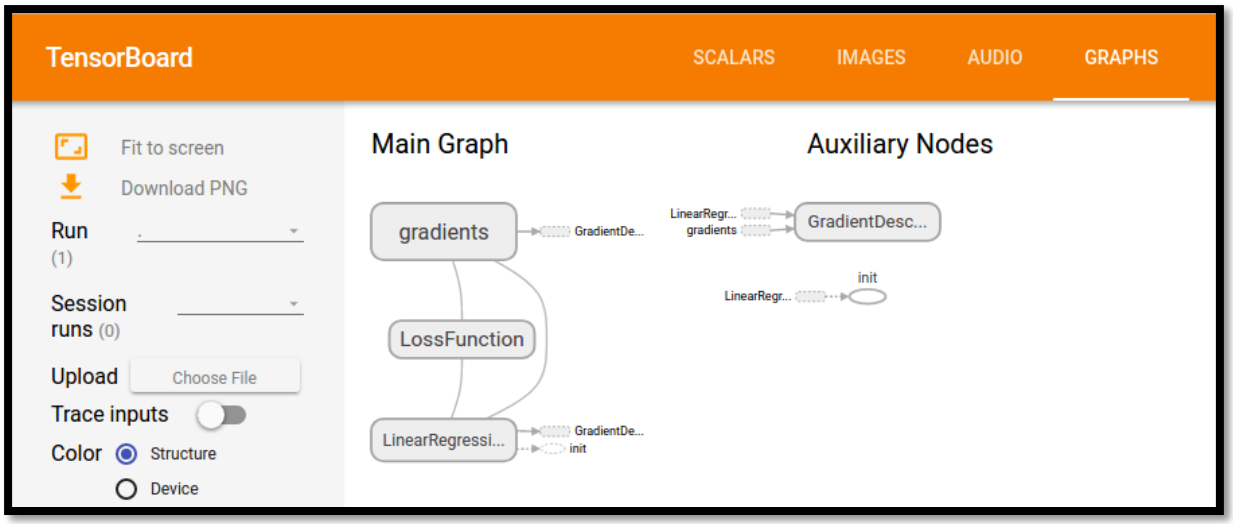

$T(e_3)$
σ_{33}
$T(e_1)$
σ_{11}
$T(e_1)$
σ_{22}

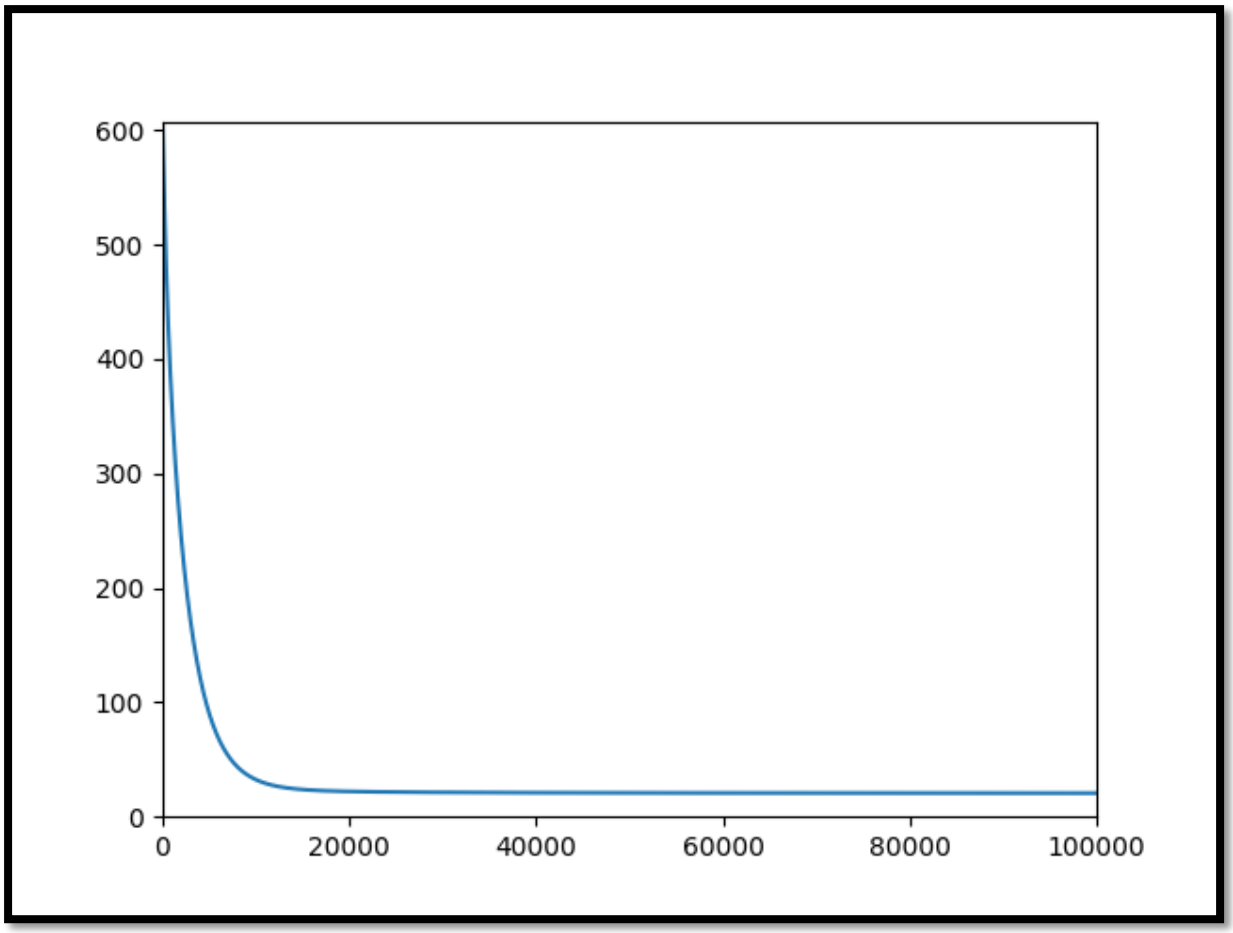

TensorBoard
EVENTS
IMAGES
GRAPH
HISTOGRAMS
input new regex
Split On Underscores:
X Type:
STEP
RELATIVE
WALL
Selected Runs:
data
xentropy
(1)
xentropy_mean
2.20
1.80
1.40
1.00
0.600
0.200
0.000
400.0
800.0
1.200k
1.600k

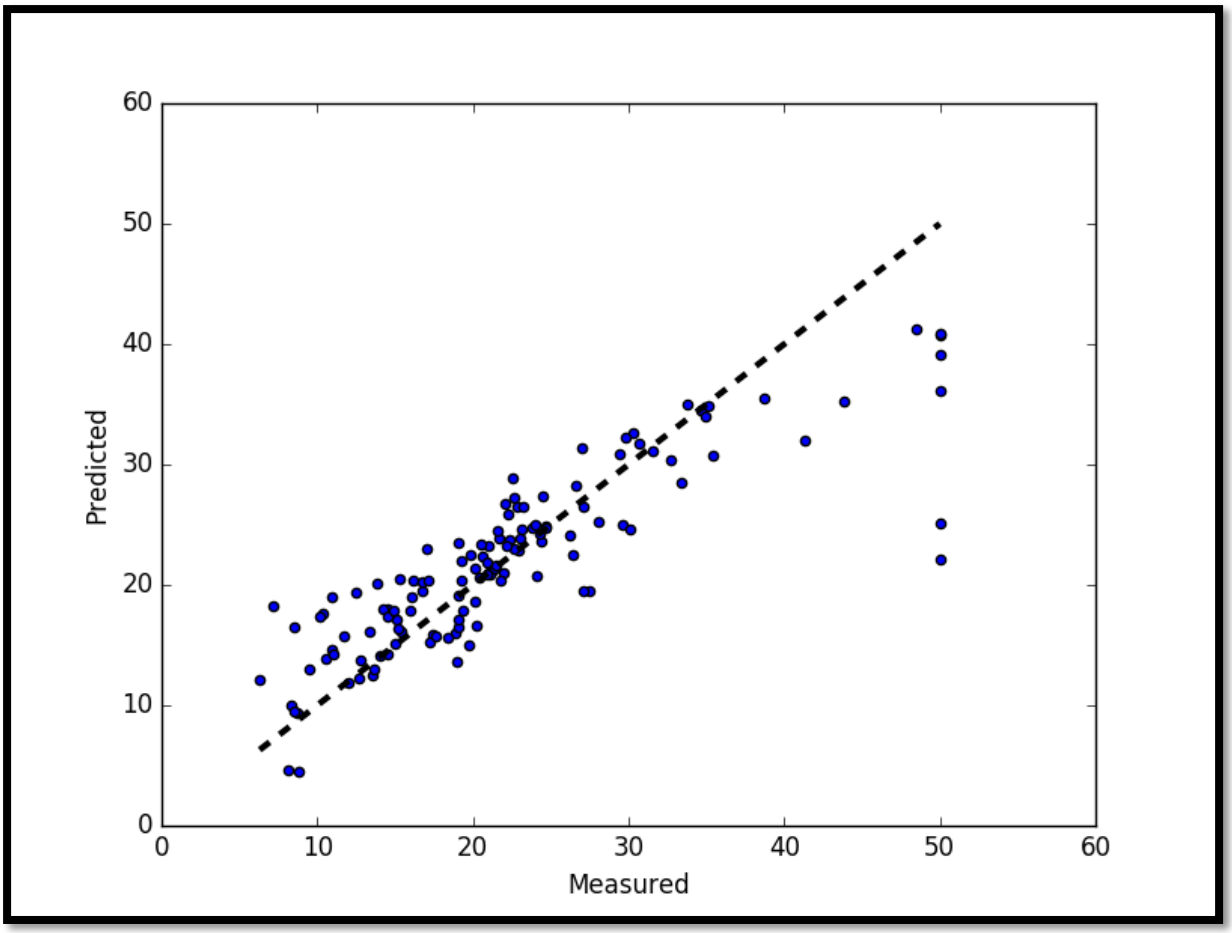

Learning algorithm in TensorFlow
session
placeholders
variables (θ)
constants
run minimizer op
graph
TensorFlow keeps updating θ to minimize cost
Training data
Feature vector
M(θ)


Original data
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
−3
−2
−1
0
1
2
3


Original data
X
Y

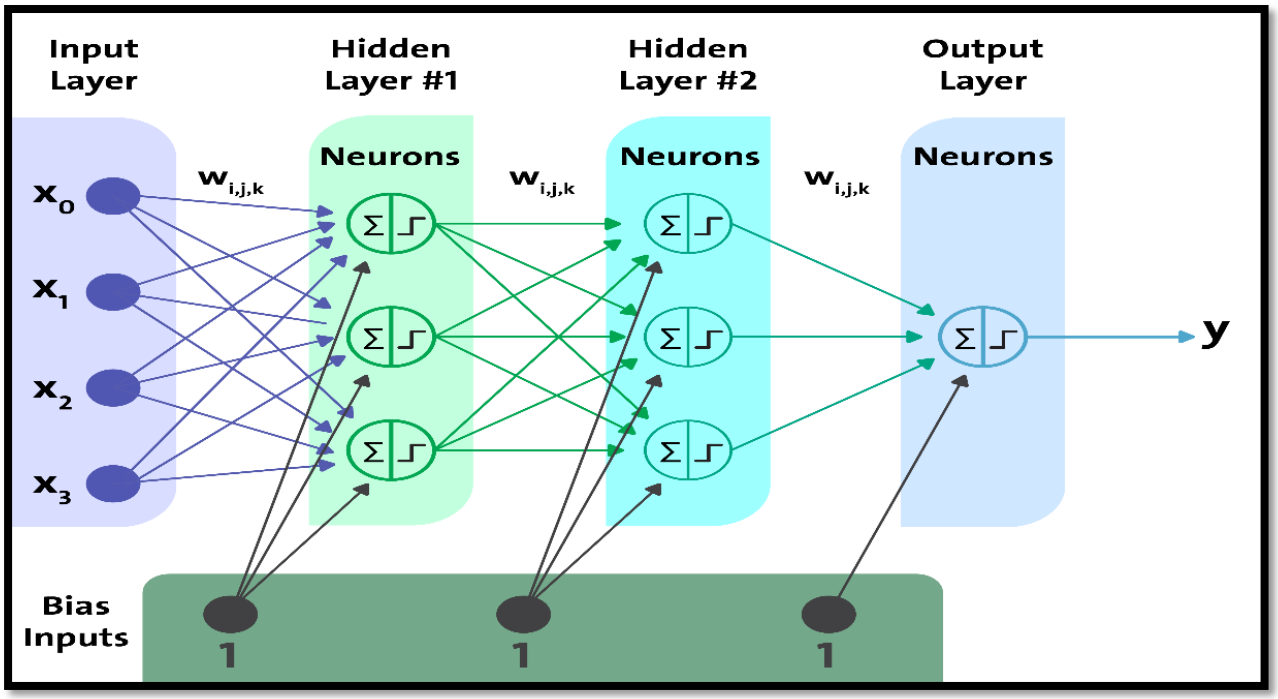

TensorBoard
SCALARS
IMAGES
AUDIO
GRAPHS
Fit to screen
Download PNG
Run
(1)
Session
runs (0)
Upload
Choose File
Trace inputs
Color
Structure
Device
Main Graph
Auxiliary Nodes
gradients
GradientDe...
LossFunction
LinearRegressi...
GradientDe...
init
LinearRegr...
gradients
GradientDesc...
init
LinearRegr...

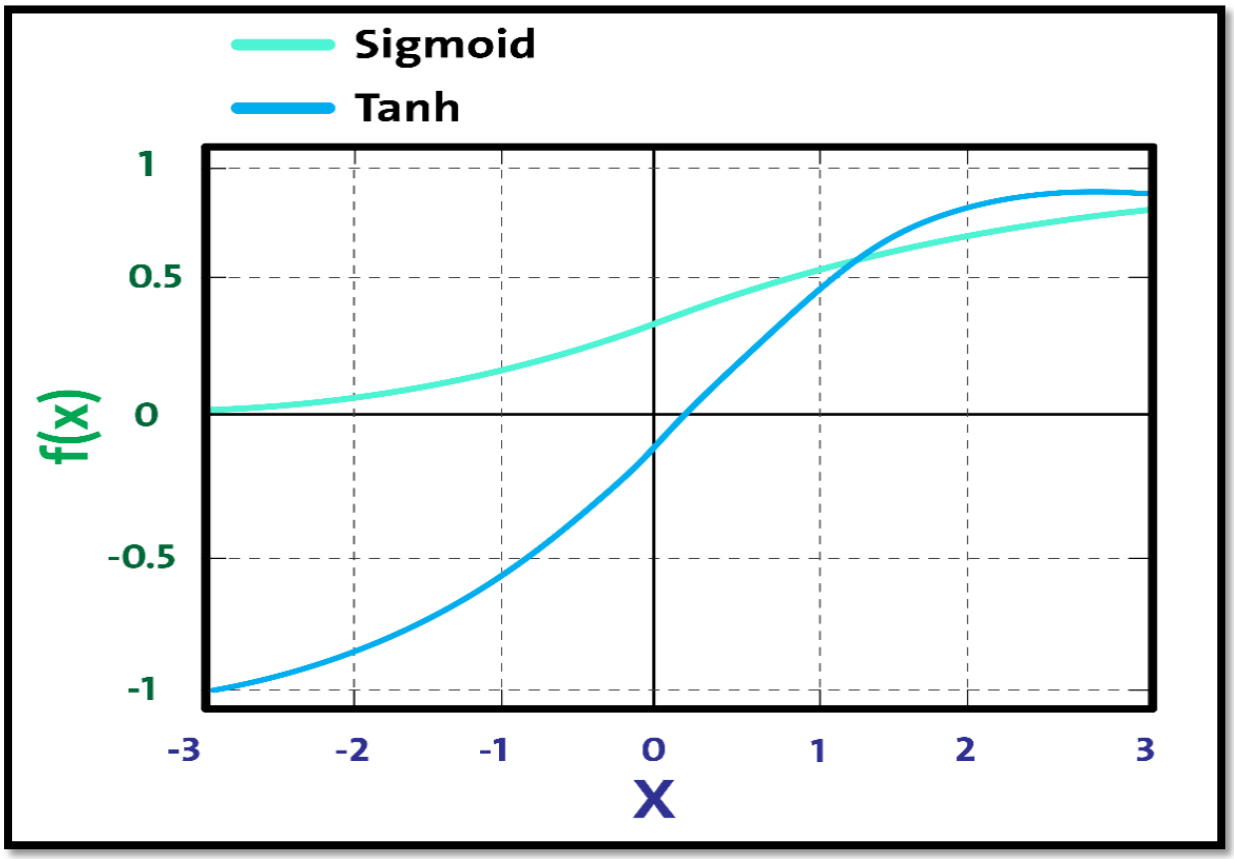

600
500
400
300
200
100
0
0
20000
40000
60000
80000
100000

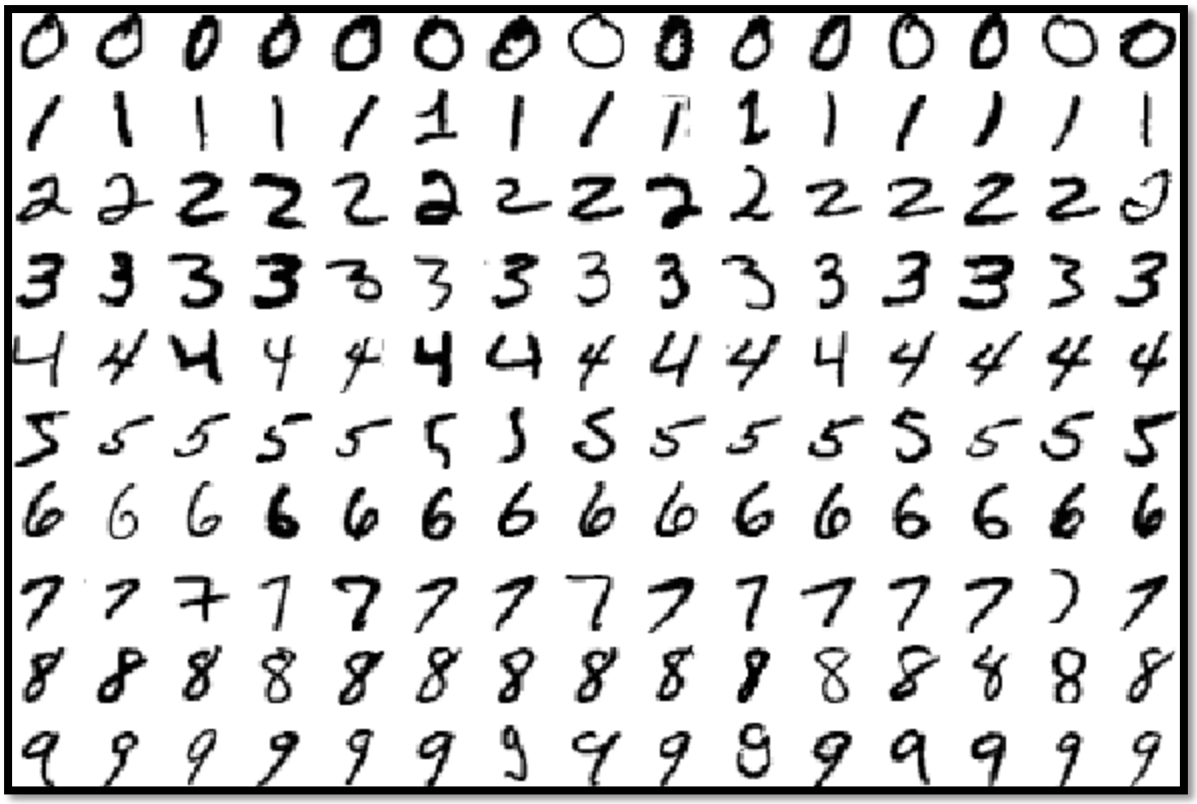


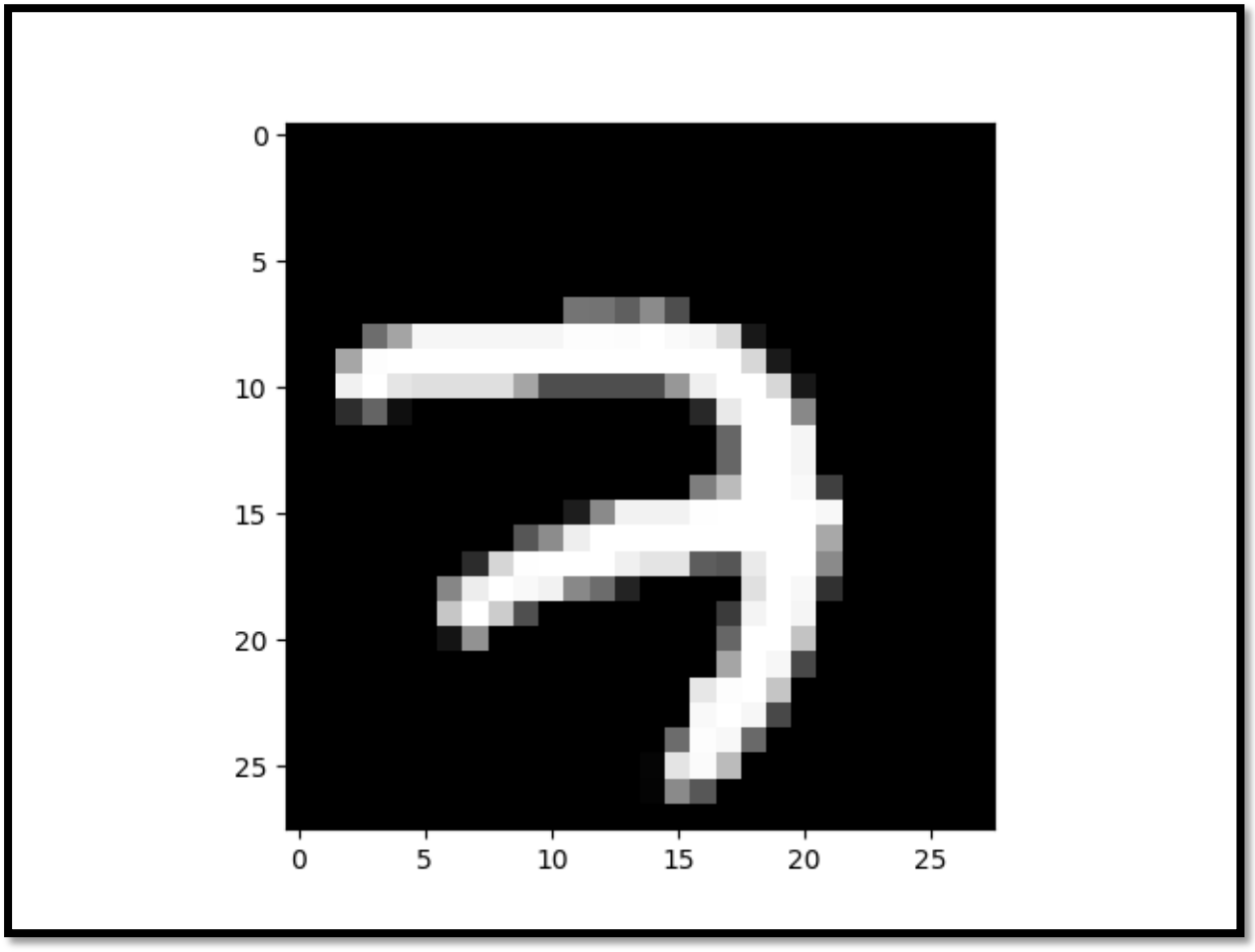
Numpy	TensorFlow
a=np.zeros((2,2));b=np.ones((2,2))	a=tf.zeros((2,2)),b=tf.ones((2,2))
np.sum(b,axis=1)	tf.reduce_sum(a,reduction_indices=[1])
a.shape	a.get_shape()
np.reshape(a,(1,4))	tf.reshape(a,(1,4))
b*5+1	b*5+1
np.dot(a,b)	tf.matmul(a,b)
a[0,0], a[:,0], a[0,:]	a[0,0],a[:,0],a[0,:]

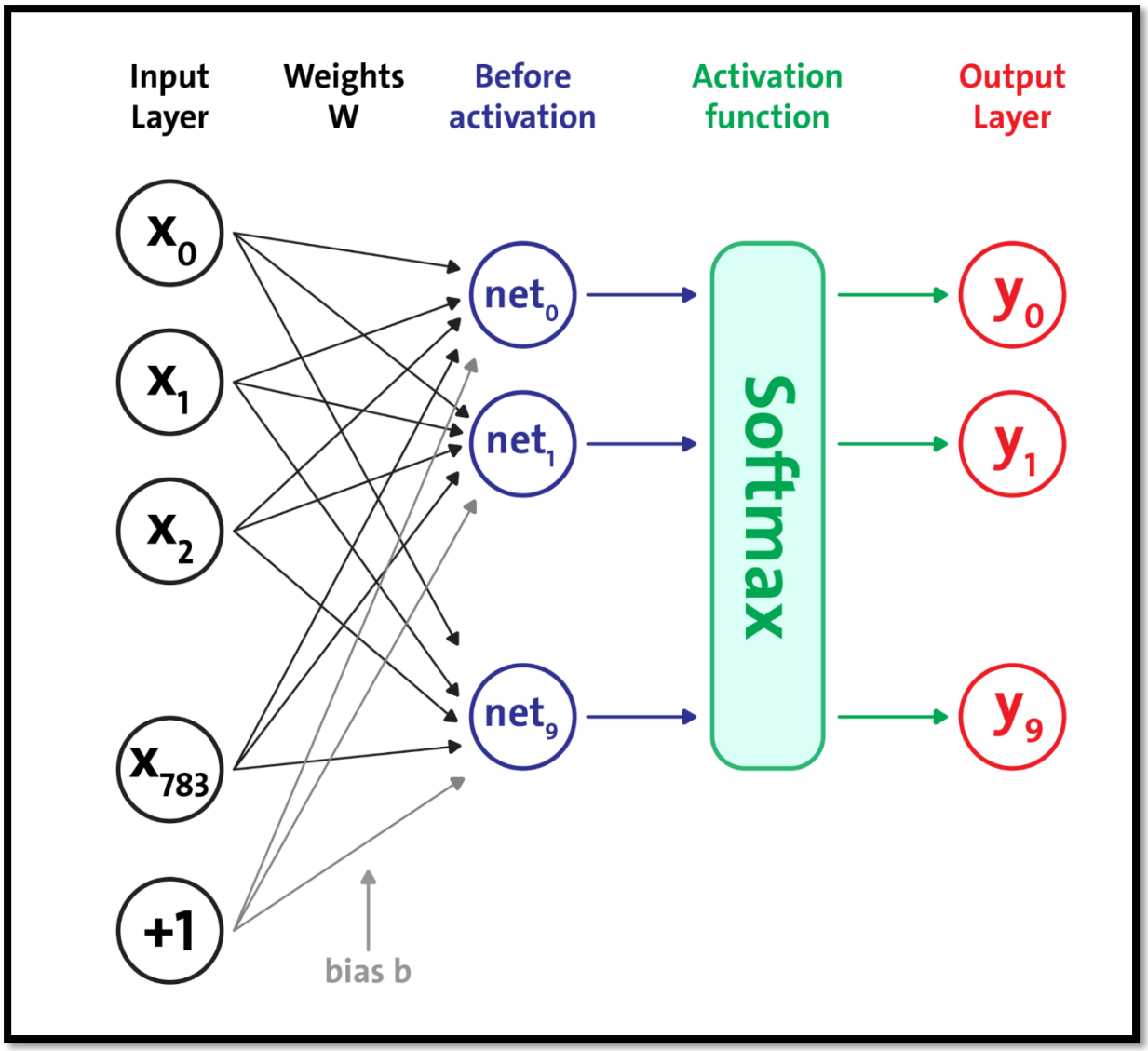
Chapter 3: Feed-Forward Neural Networks with TensorFlow

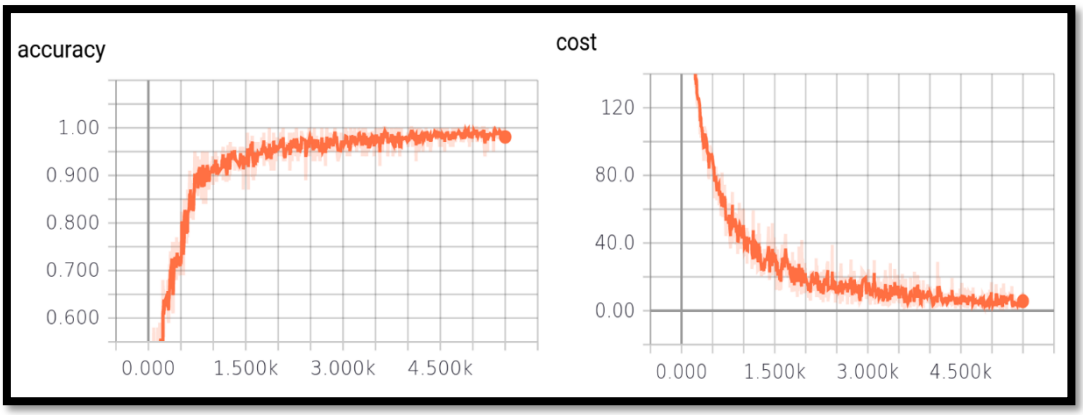


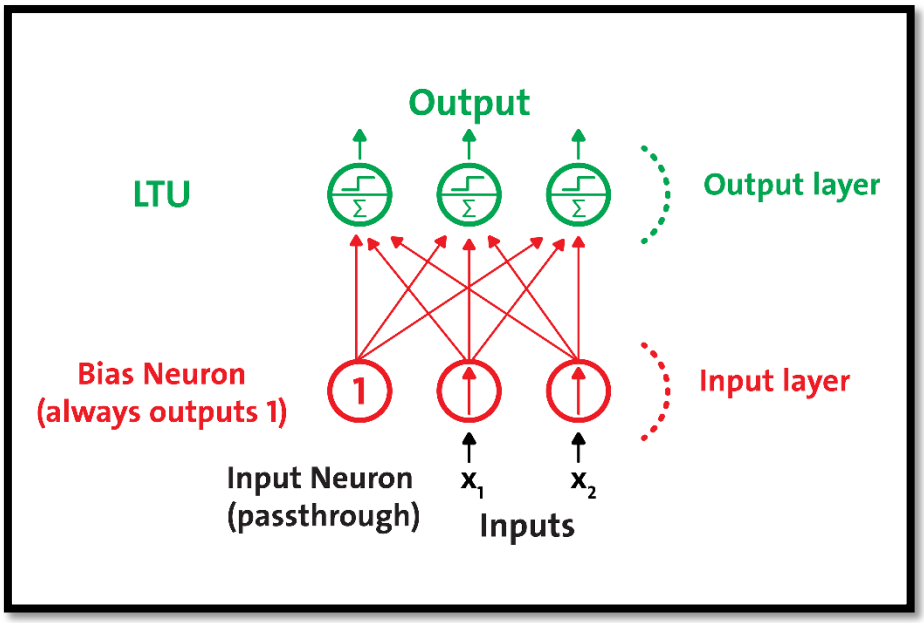


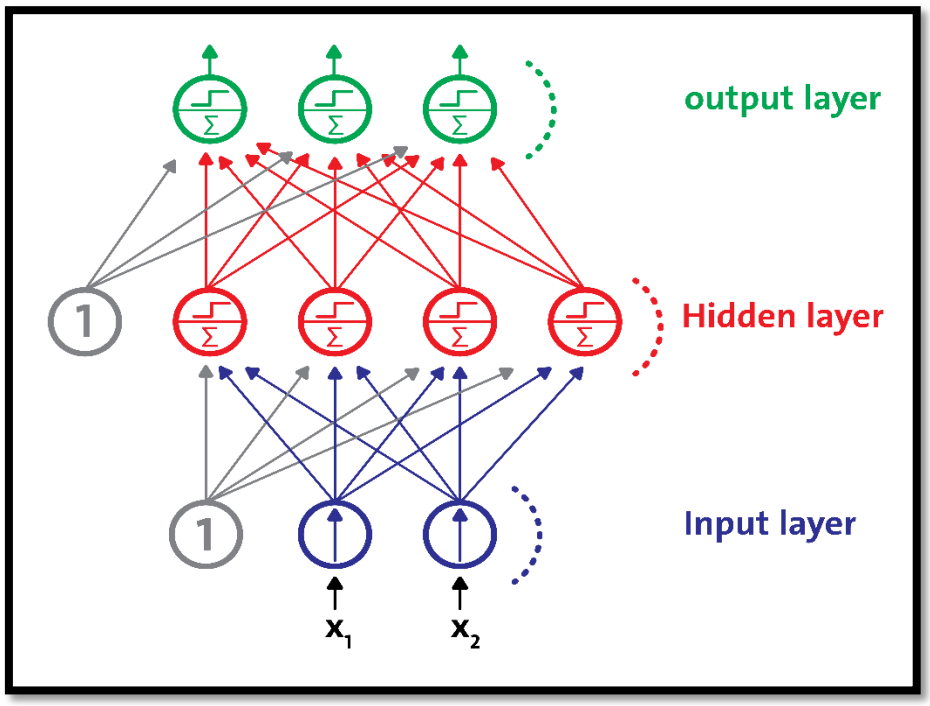


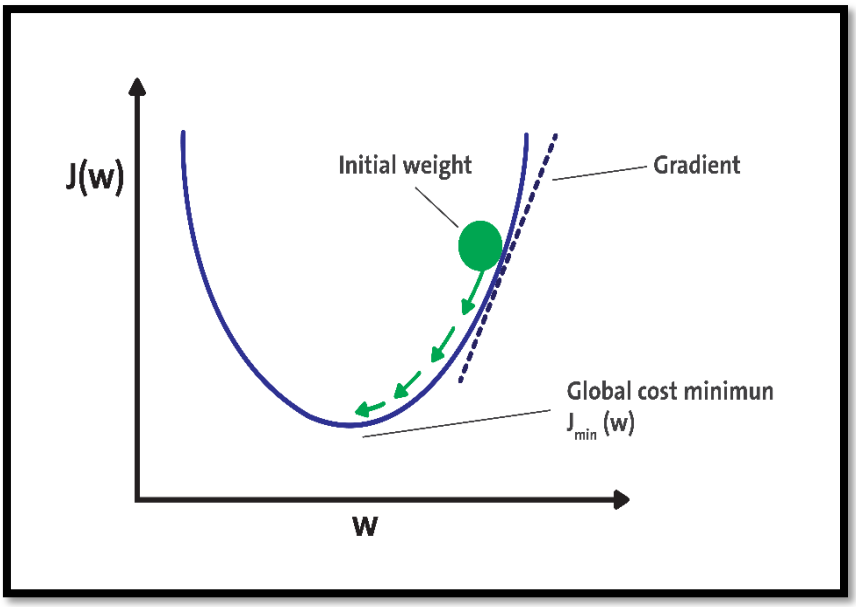

0
5
10
15
20
25
0
5
10
15
20
25

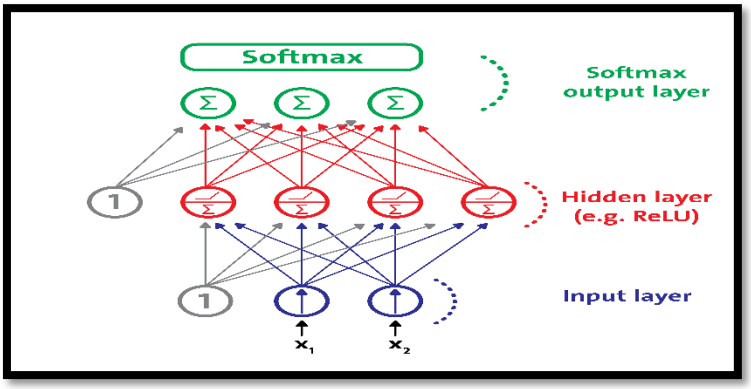

Input Layer
Weights W
Before activation
Activation function
Output Layer
x0
x1
x2
x783
+1
net0
net1
net9
Softmax
y0
y1
y9
bias b

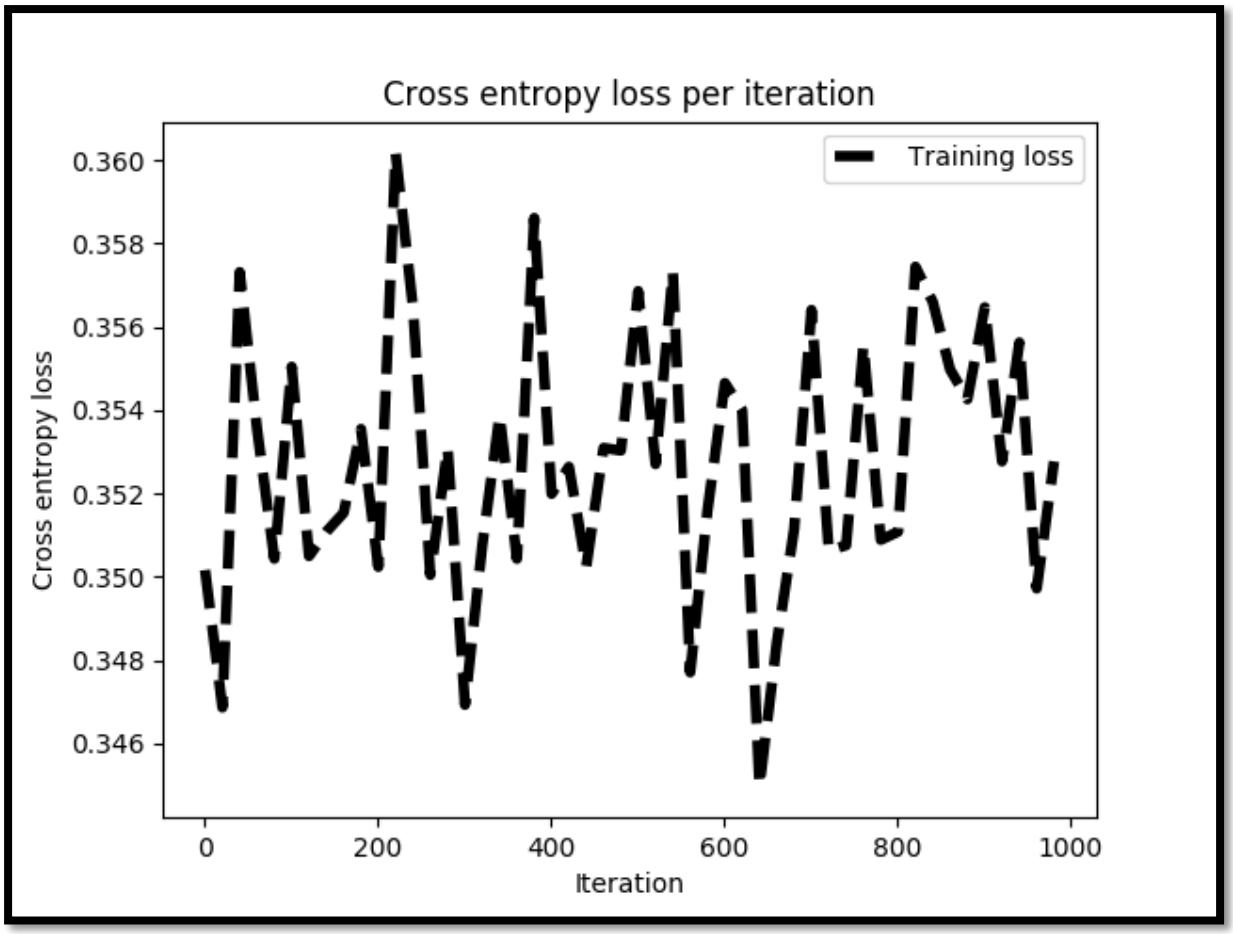

accuracy
1.00
0.900
0.800
0.700
0.600
0.000
1.500k
3.000k
4.500k
cost
120
80.0
40.0
0.00
0.000
1.500k
3.000k
4.500k

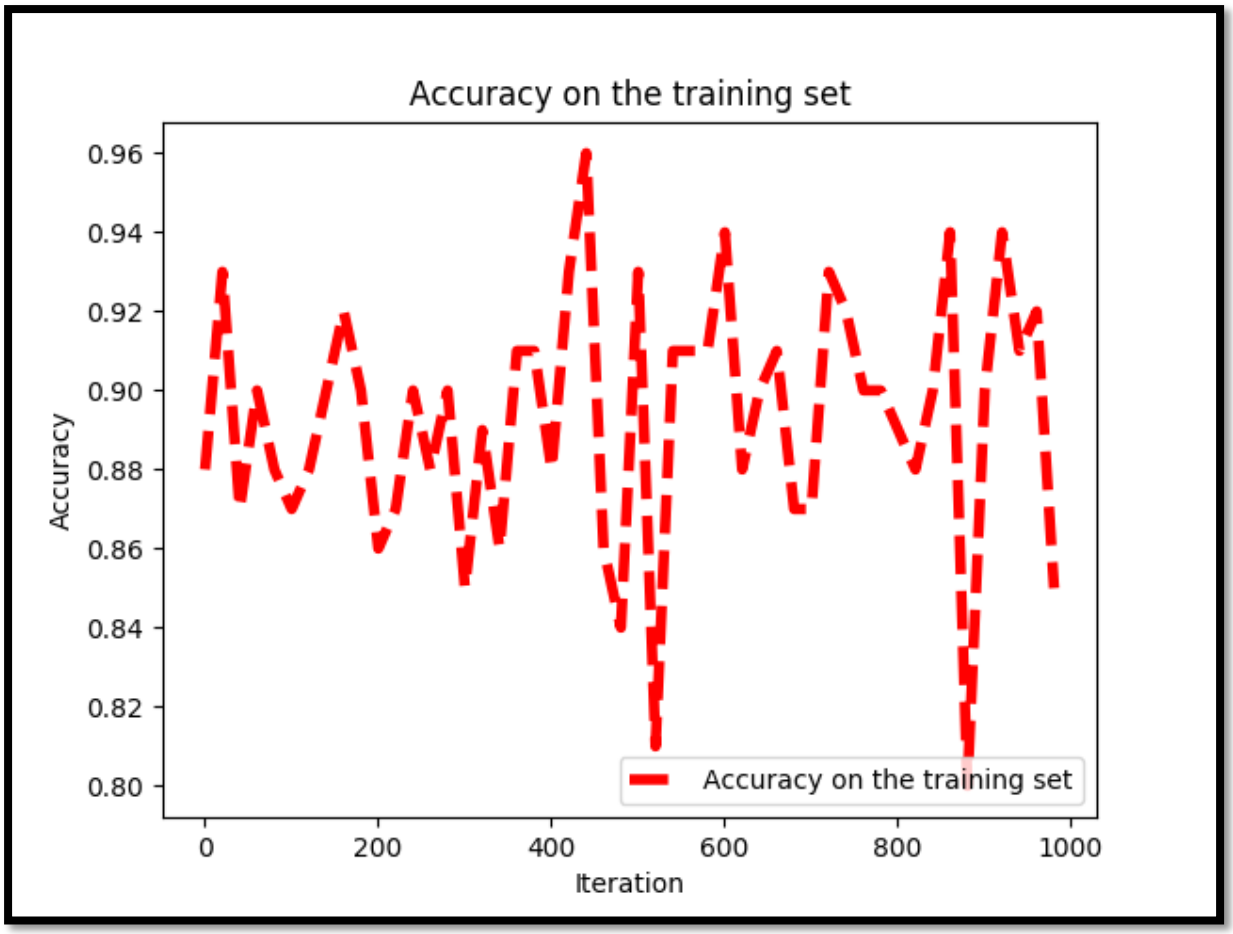

Output
LTU
Output layer
Bias Neuron
(always outputs 1)
Input layer
Input Neuron
(passthrough)
x_1
x_2
Inputs

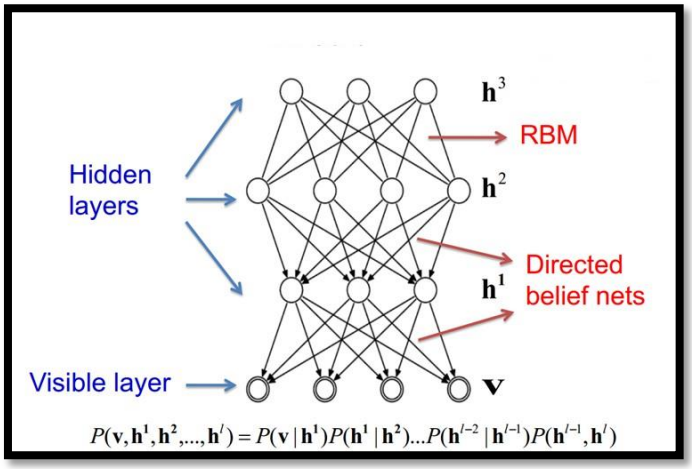

output layer
Hidden layer
Input layer
x_1
x_2

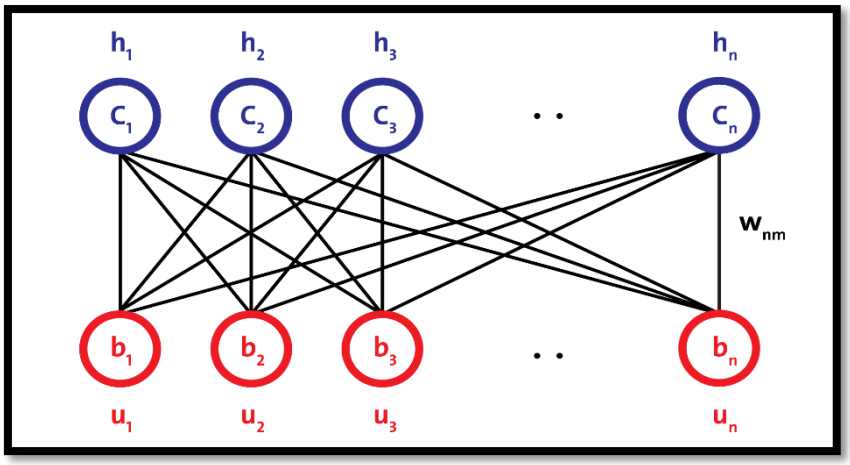

J(w)
Initial weight
Gradient
Global cost minimun
J_{min} (w)
W

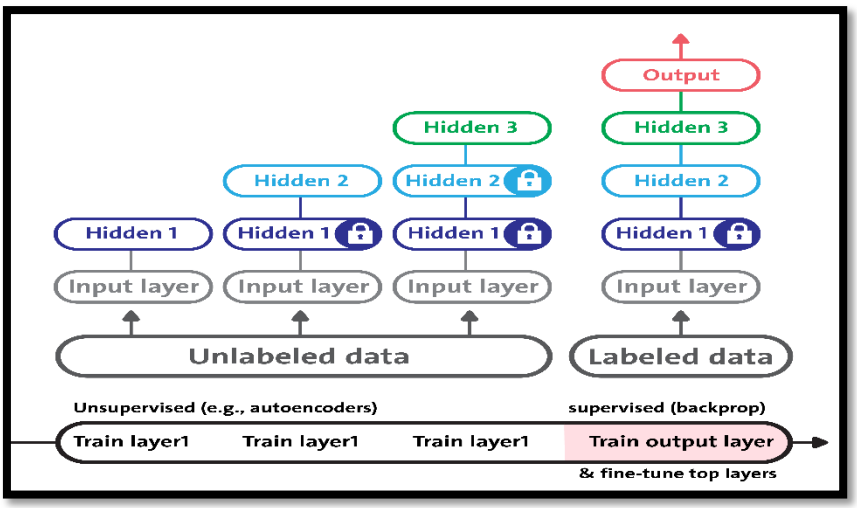

Softmax
Softmax output layer
Hidden layer (e.g. ReLU)
Input layer
1
1
x_1
x_2

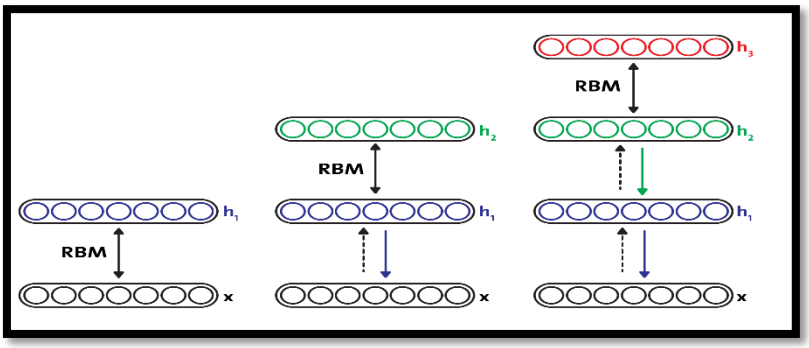

Cross entropy loss per iteration
Training loss
Cross entropy loss
0.360
0.358
0.356
0.354
0.352
0.350
0.348
0.346
0
200
400
600
800
1000
Iteration

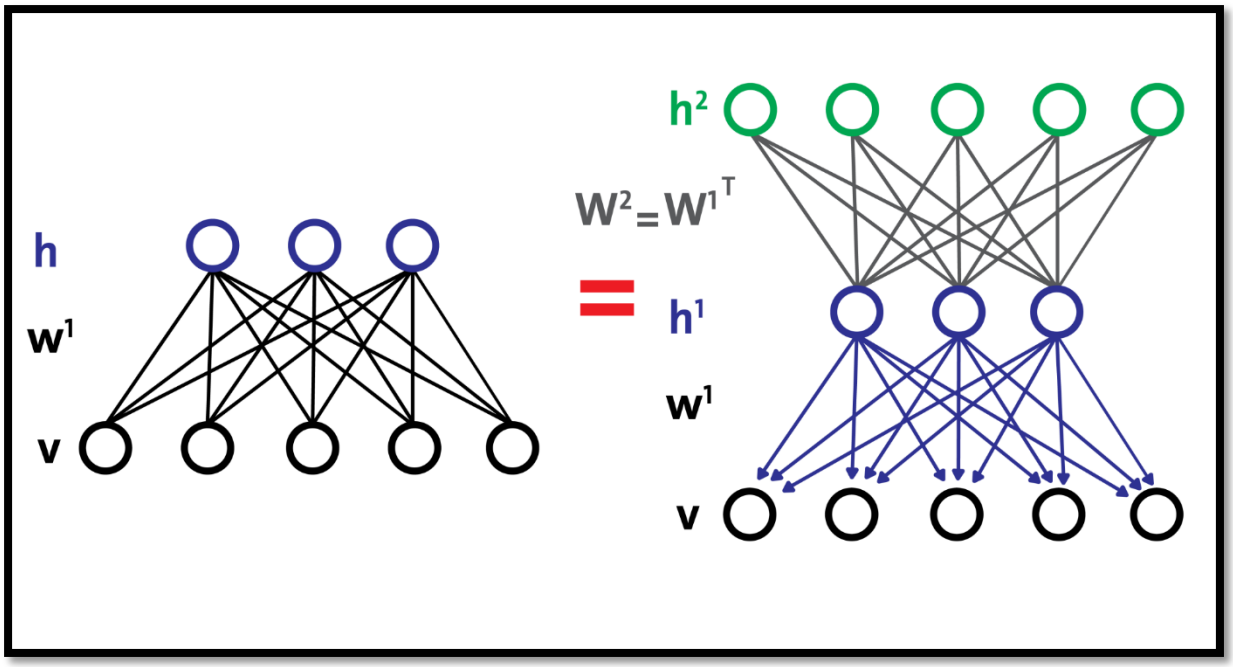

Accuracy on the training set
Accuracy
Iteration
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0
200
400
600
800
1000
Accuracy on the training set

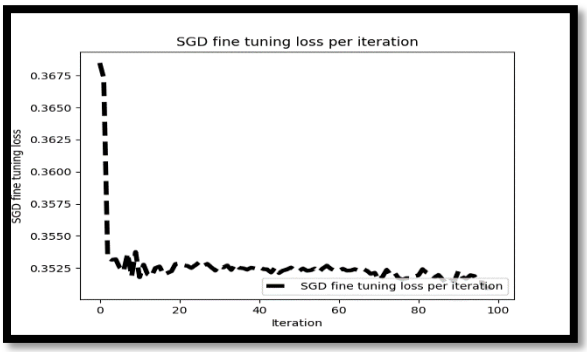

Hidden layers
Visible layer
RBM
Directed belief nets
$\mathbf{h}^3$
$\mathbf{h}^2$
$\mathbf{h}^1$
$\mathbf{v}$
$P(\mathbf{v},\mathbf{h}^1,\mathbf{h}^2,...,\mathbf{h}^l) = P(\mathbf{v}\,|\,\mathbf{h}^1)P(\mathbf{h}^1\,|\,\mathbf{h}^2)...P(\mathbf{h}^{l-2}\,|\,\mathbf{h}^{l-1})P(\mathbf{h}^{l-1},\mathbf{h}^l)$

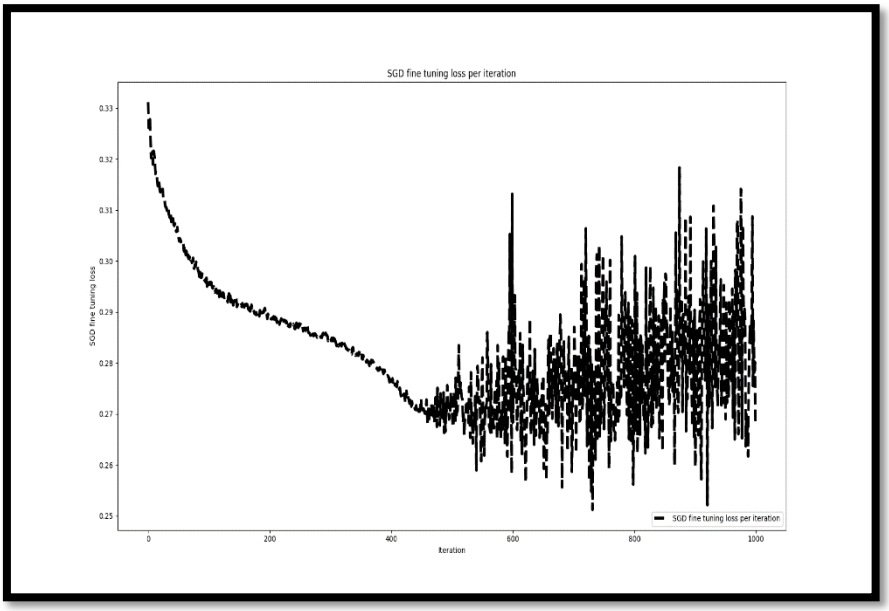

h1
h2
h3
hn
C1
C2
C3
Cn
Wnm
b1
b2
b3
bn
u1
u2
u3
un

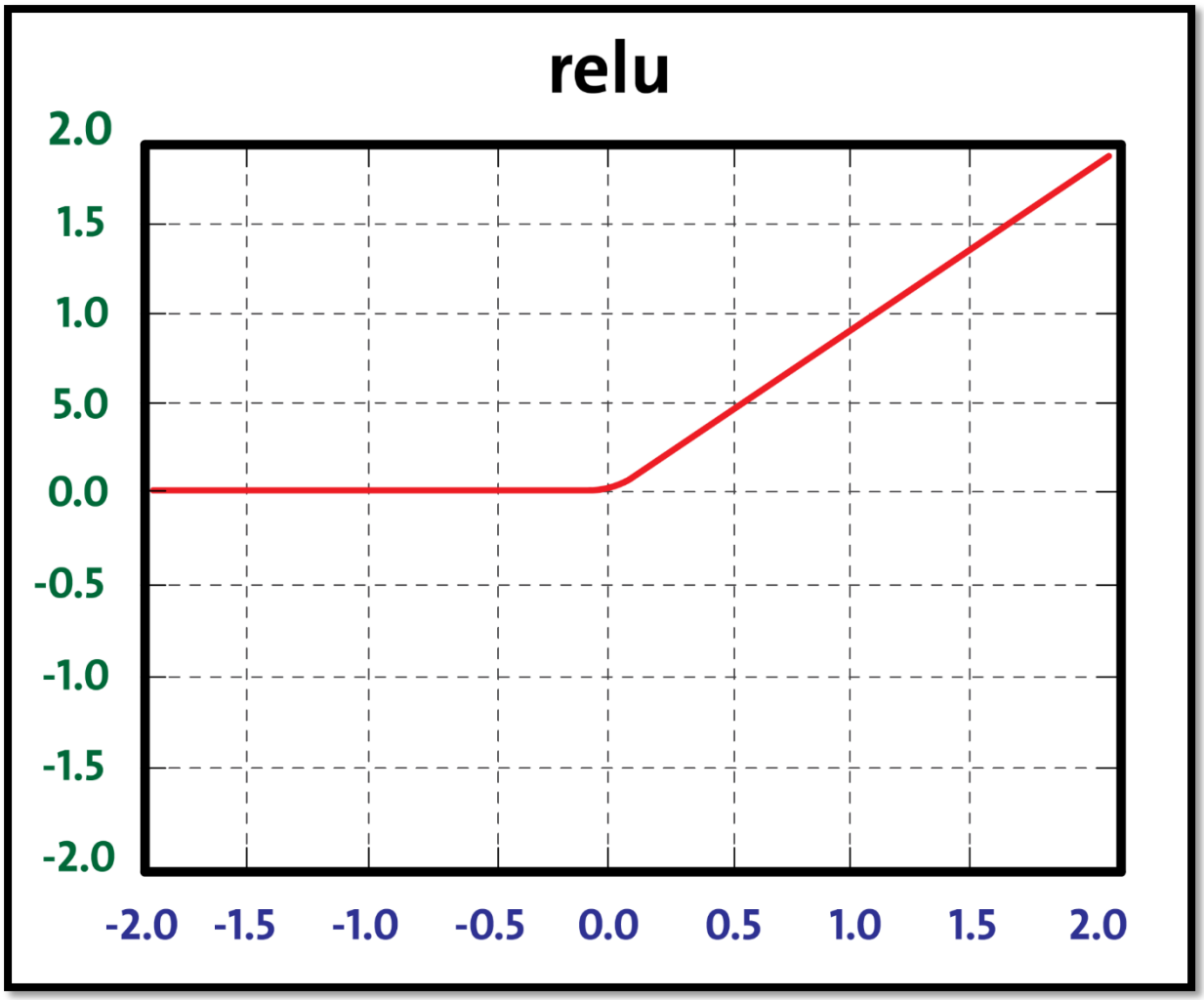

Output
Hidden 3
Hidden 3
Hidden 2
Hidden 2
Hidden 2
Hidden 1
Hidden 1
Hidden 1
Hidden 1
Input layer
Input layer
Input layer
Input layer
Unlabeled data
Labeled data
Unsupervised (e.g., autoencoders)
supervised (backprop)
Train layer1
Train layer1
Train layer1
Train output layer
& fine-tune top layers

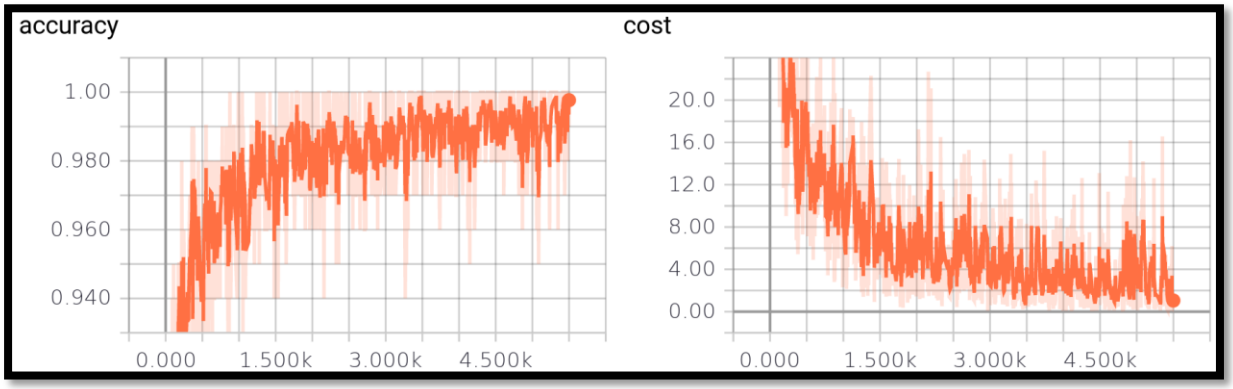

h3
RBM
h2
h2
RBM
h1
h1
h1
RBM
x
x
x

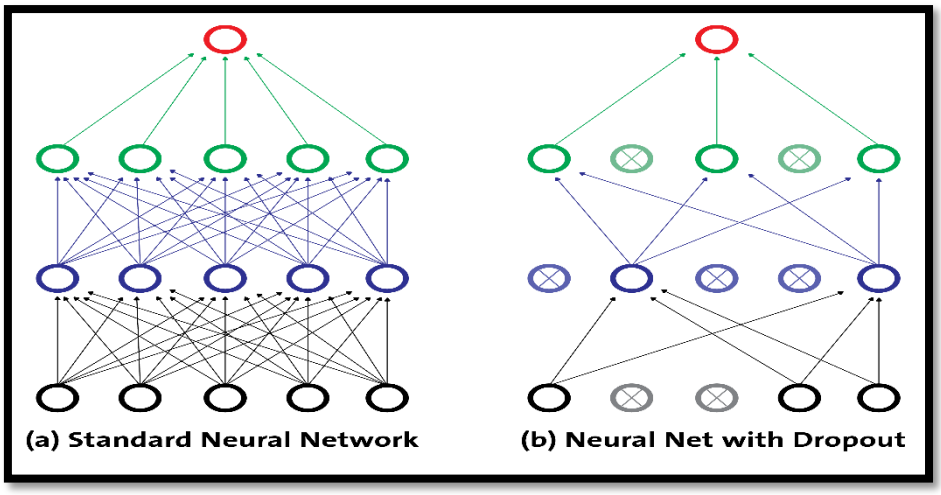

h
w1
v
h2
W2=W1T
h1
w1
v

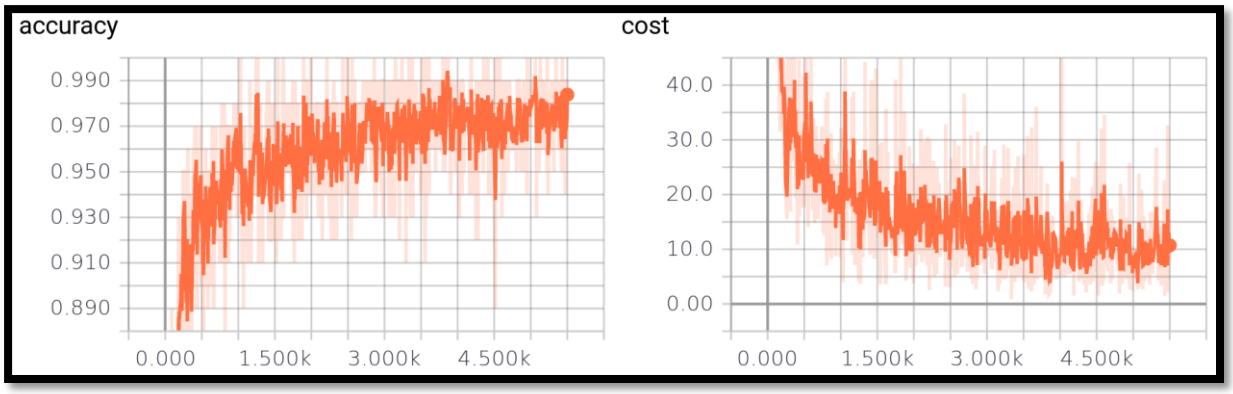

SGD fine tuning loss per iteration
SGD fine tuning loss
0.3675
0.3650
0.3625
0.3600
0.3575
0.3550
0.3525
SGD fine tuning loss per iteration
0
20
40
60
80
100
Iteration


SGD fine tuning loss per iteration
SGD fine tuning loss
0.33
0.32
0.31
0.30
0.29
0.28
0.27
0.26
0.25
SGD fine tuning loss per iteration
0
200
400
600
800
1000
Iteration

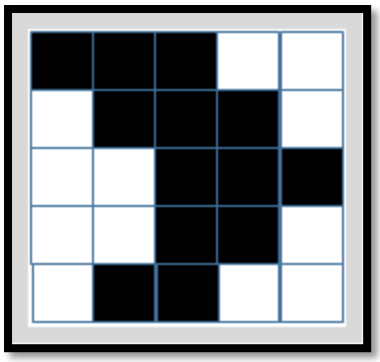

relu
2.0
1.5
1.0
5.0
0.0
-0.5
-1.0
-1.5
-2.0
-2.0 -1.5 -1.0 -0.5 0.0 0.5 1.0 1.5 2.0

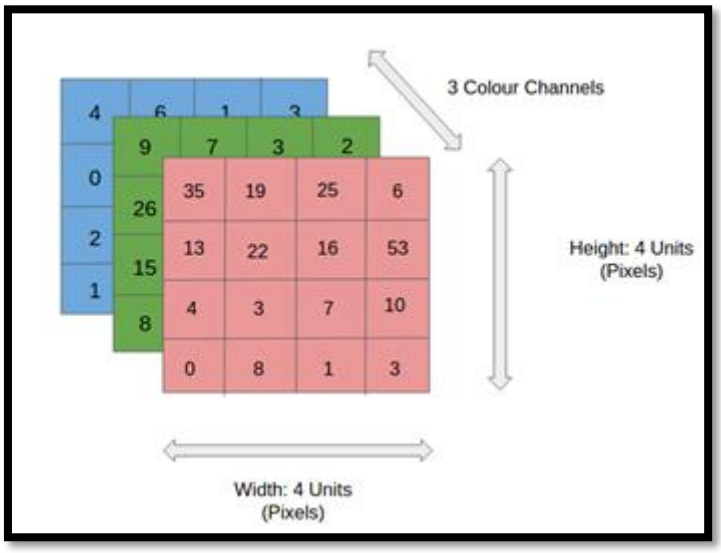

accuracy
1.00
0.980
0.960
0.940
0.000 1.500k 3.000k 4.500k
cost
20.0
16.0
12.0
8.00
4.00
0.00
0.000 1.500k 3.000k 4.500k


(a) Standard Neural Network
(b) Neural Net with Dropout

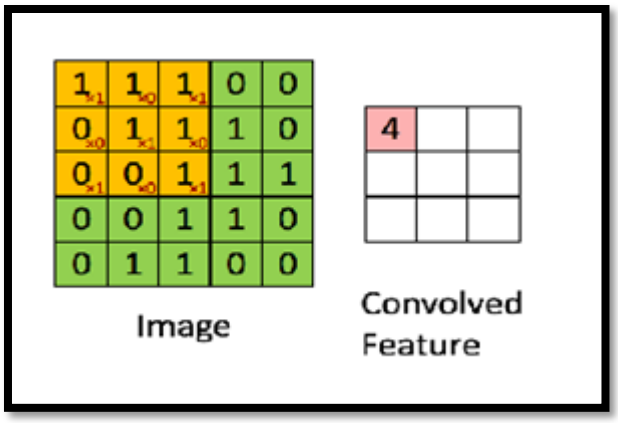

accuracy
0.990
0.970
0.950
0.930
0.910
0.890
0.000
1.500k
3.000k
4.500k
cost
40.0
30.0
20.0
10.0
0.00
0.000
1.500k
3.000k
4.500k

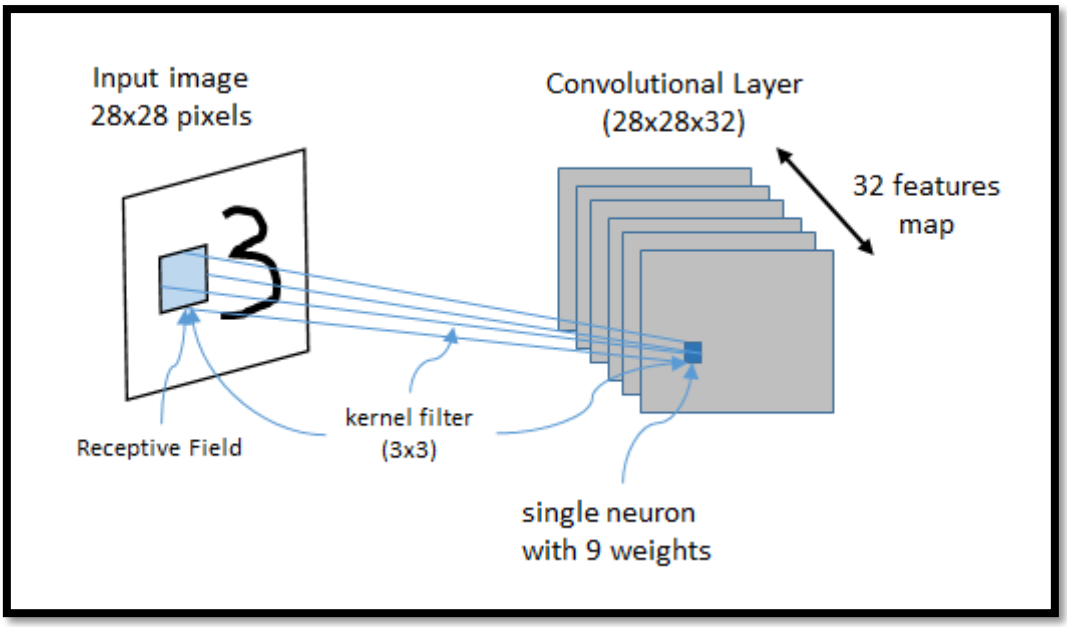
Chapter 4: Convolutional Neural Networks

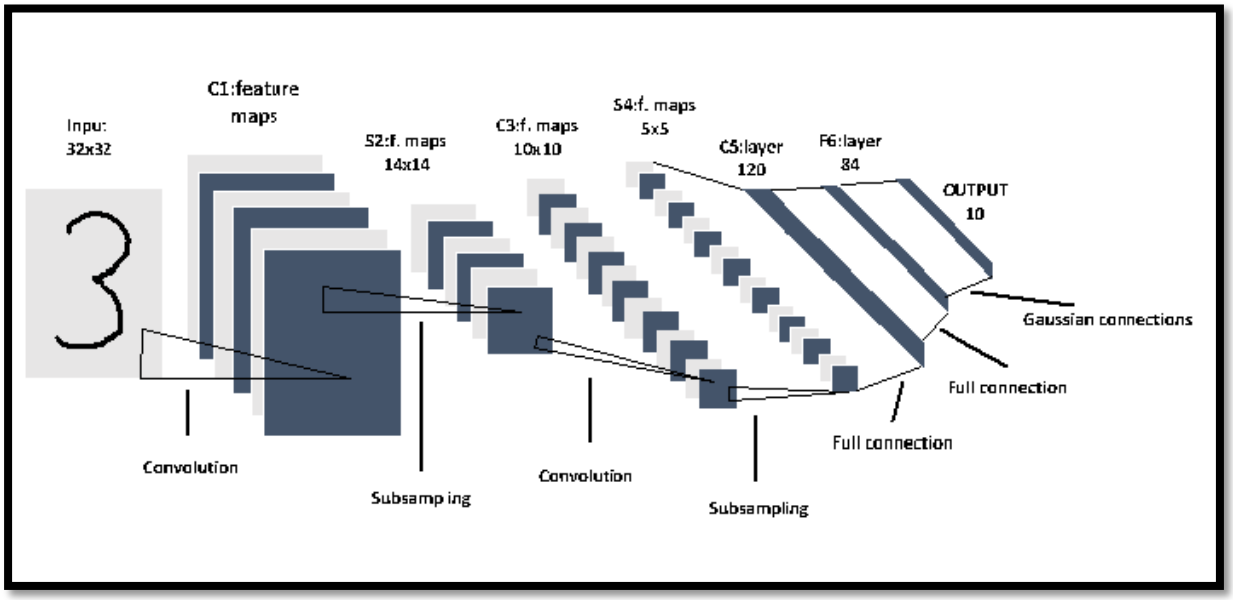


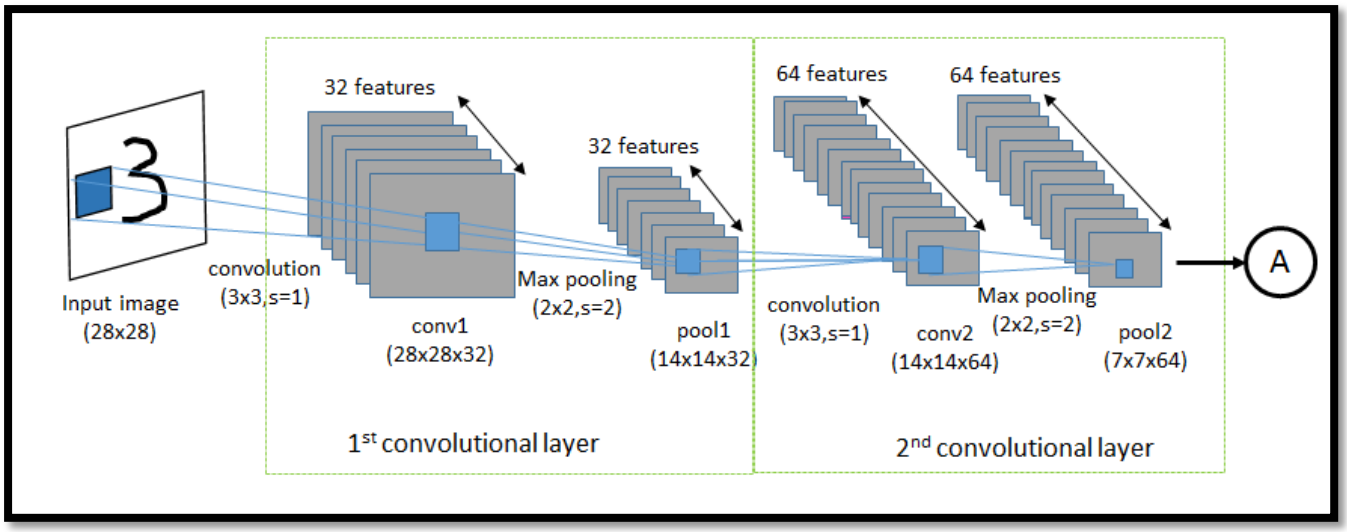


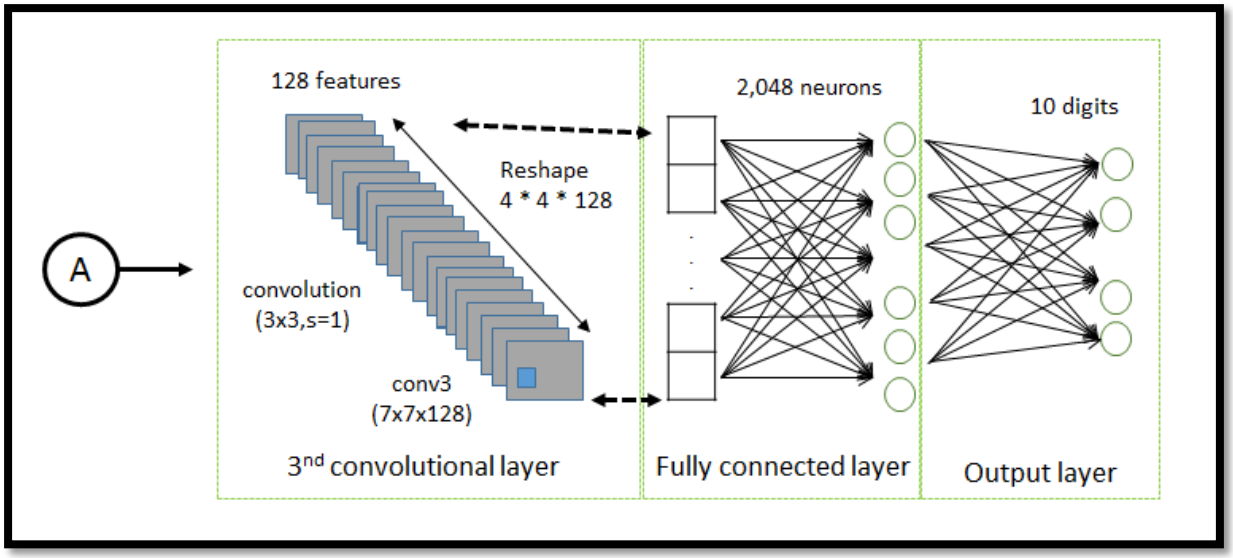
$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

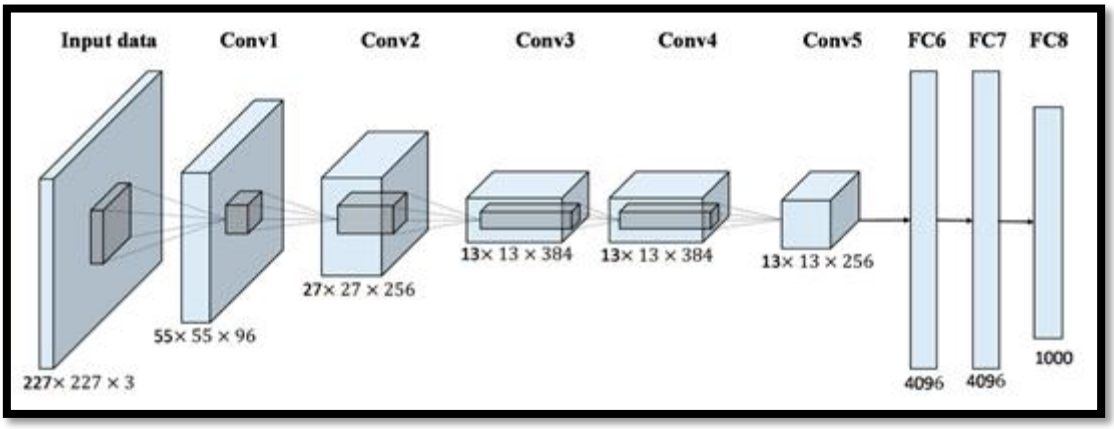



Input image
28x28 pixels
Convolutional Layer
(28x28x32)
32 features
map
Receptive Field
kernel filter
(3x3)
single neuron
with 9 weights


Inpu:
32x32
C1:feature
maps
S2:f. maps
14x14
C3:f. maps
10x10
S4:f. maps
5x5
C5:layer
120
F6:layer
84
OUTPUT
10
Gaussian connections
Full connection
Full connection
Convolution
Subsamp ing
Convolution
Subsampling


32 features
32 features
64 features
64 features
Input image
(28x28)
convolution
(3x3,s=1)
conv1
(28x28x32)
Max pooling
(2x2,s=2)
pool1
(14x14x32)
convolution
(3x3,s=1)
conv2
(14x14x64)
Max pooling
(2x2,s=2)
pool2
(7x7x64)
A
1st convolutional layer
2nd convolutional layer


A
128 features
Reshape
4 * 4 * 128
convolution
(3x3,s=1)
conv3
(7x7x128)
2,048 neurons
10 digits
3nd convolutional layer
Fully connected layer
Output layer


Input data
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
227× 227 × 3
55× 55 × 96
27× 27 × 256
13× 13 × 384
13× 13 × 384
13× 13 × 256
4096
4096
1000

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

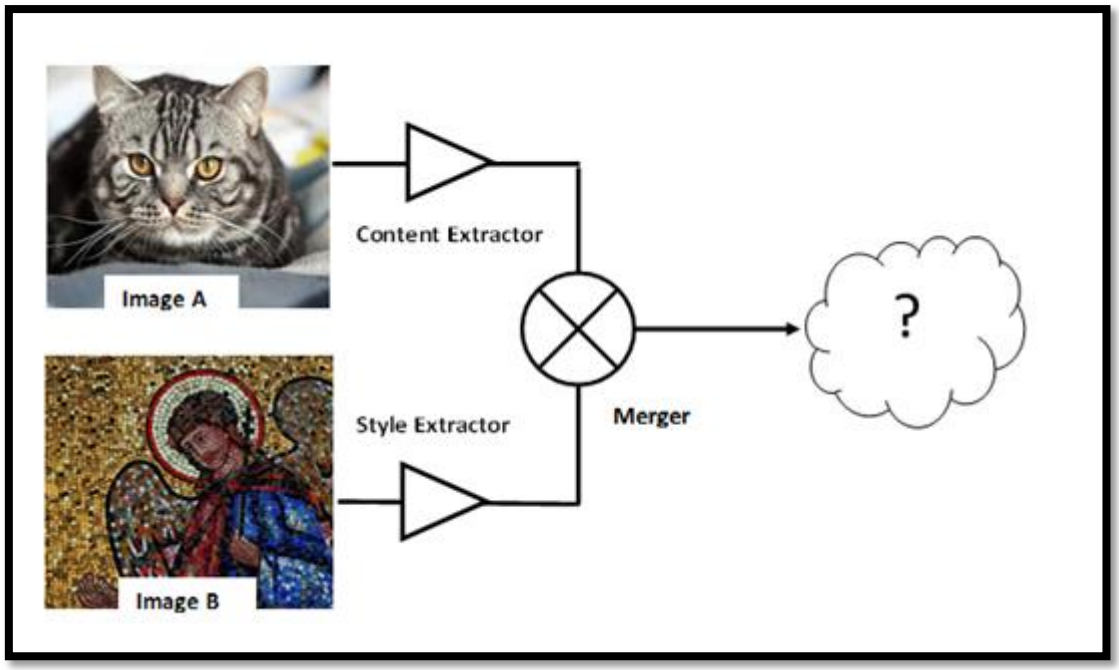
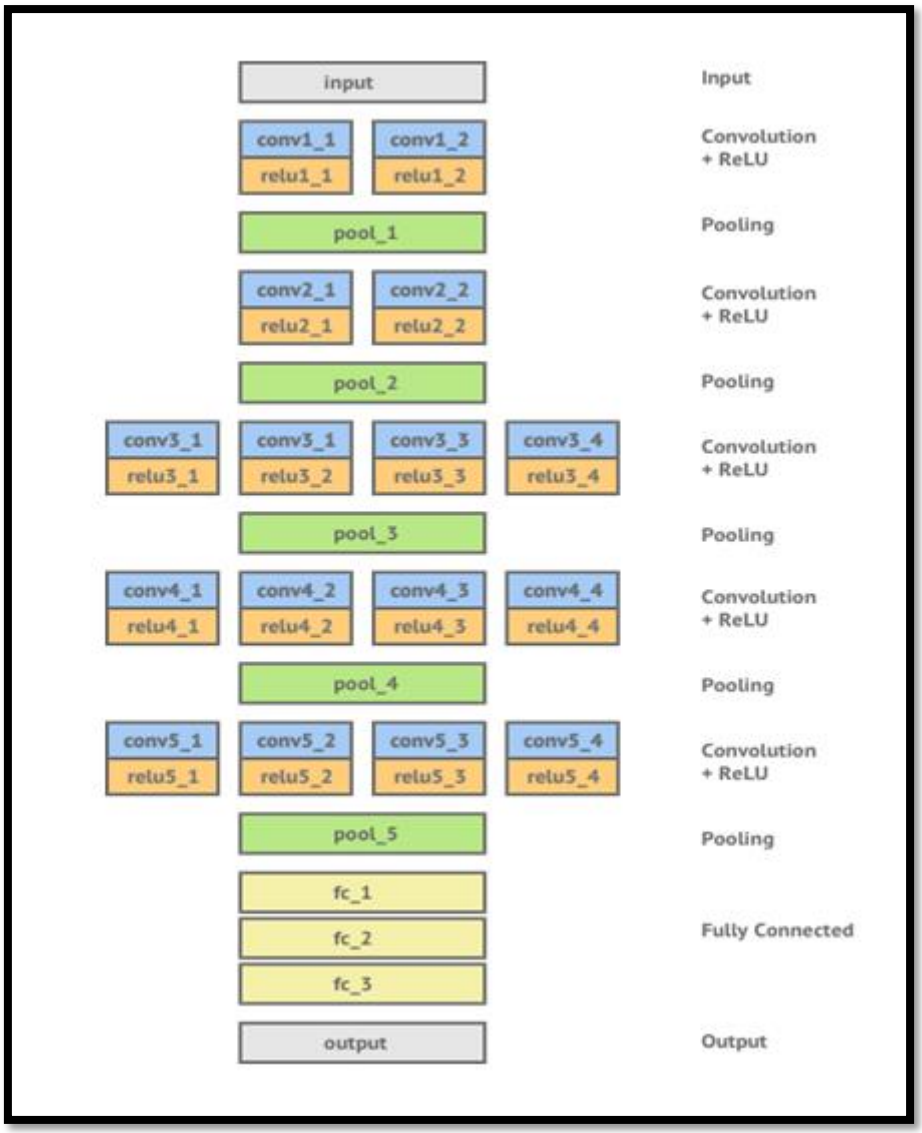
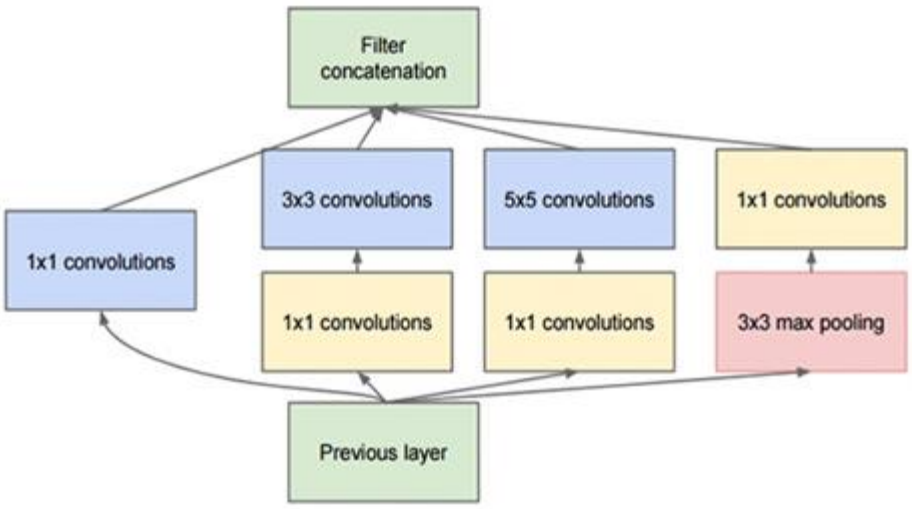
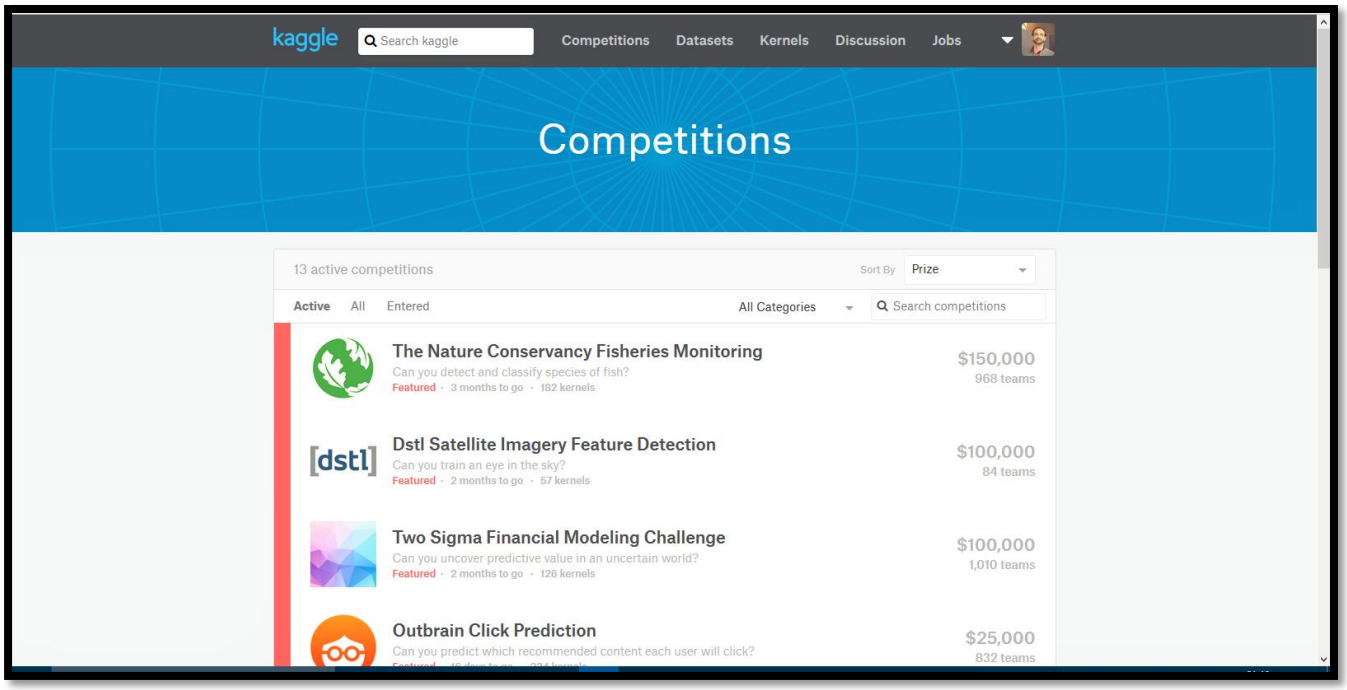
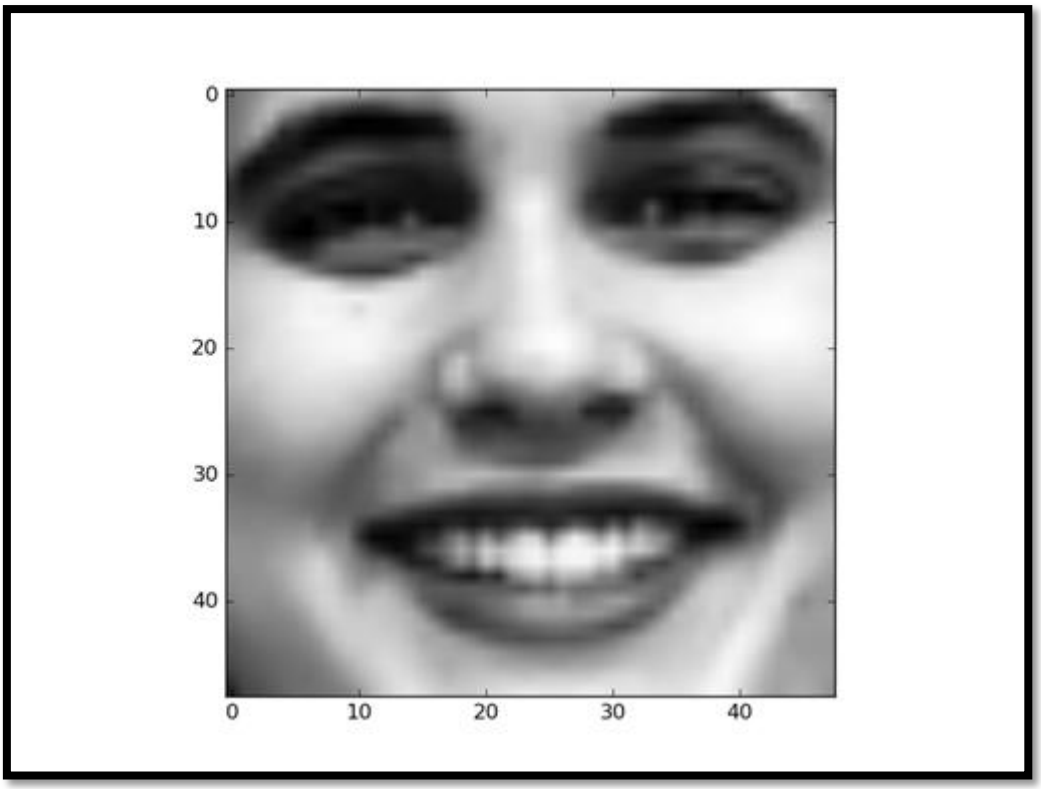


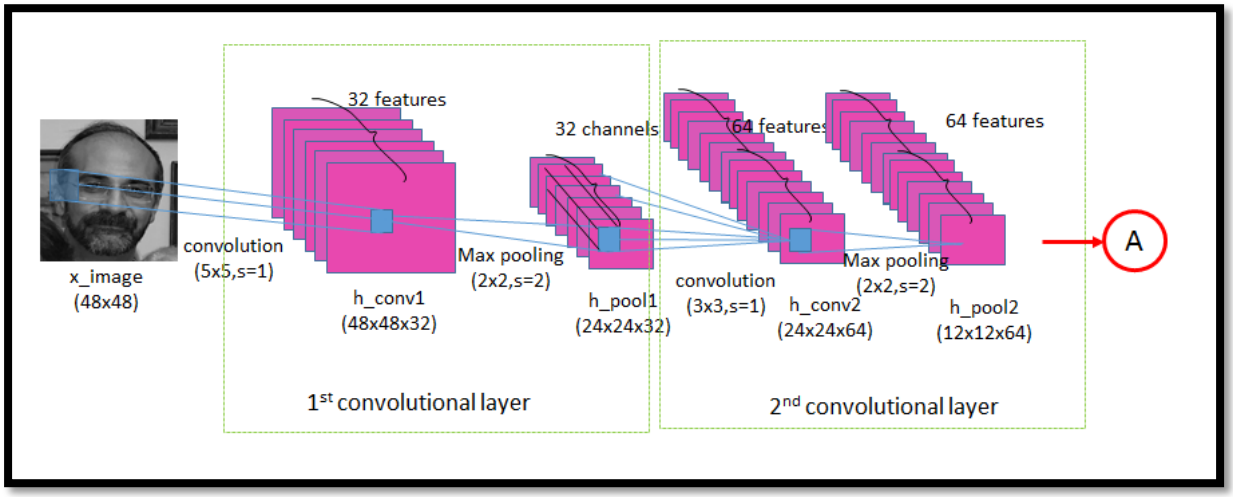
Image A
Content Extractor
Image B
Style Extractor
Merger
?

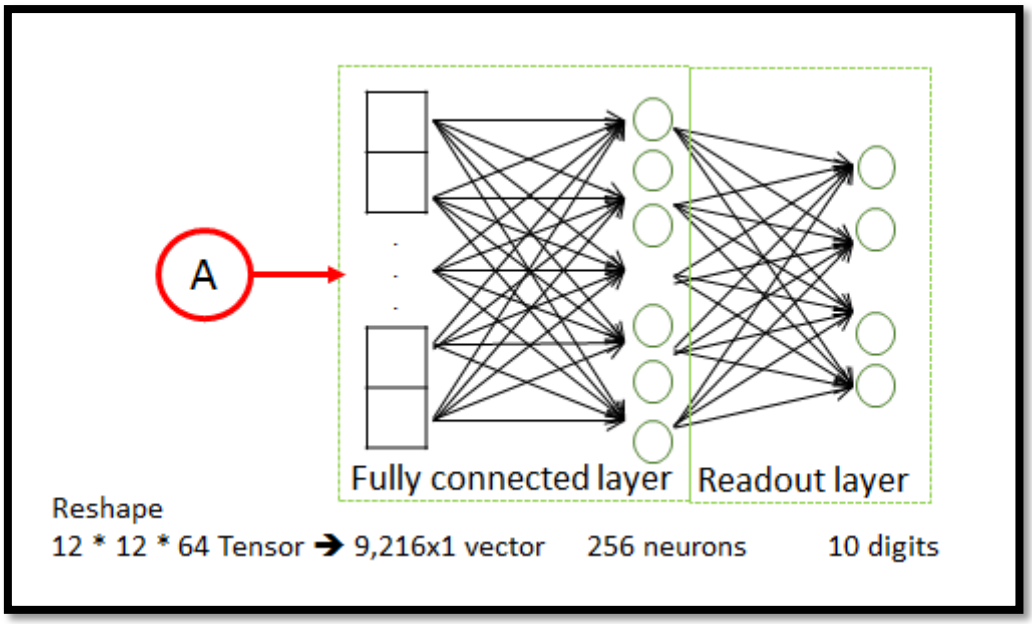

input
Input
conv1_1
relu1_1
conv1_2
relu1_2
Convolution
+ ReLU
pool_1
Pooling
conv2_1
relu2_1
conv2_2
relu2_2
Convolution
+ ReLU
pool_2
Pooling
conv3_1
relu3_1
conv3_1
relu3_2
conv3_3
relu3_3
conv3_4
relu3_4
Convolution
+ ReLU
pool_3
Pooling
conv4_1
relu4_1
conv4_2
relu4_2
conv4_3
relu4_3
conv4_4
relu4_4
Convolution
+ ReLU
pool_4
Pooling
conv5_1
relu5_1
conv5_2
relu5_2
conv5_3
relu5_3
conv5_4
relu5_4
Convolution
+ ReLU
pool_5
Pooling
fc_1
fc_2
fc_3
Fully Connected
output
Output

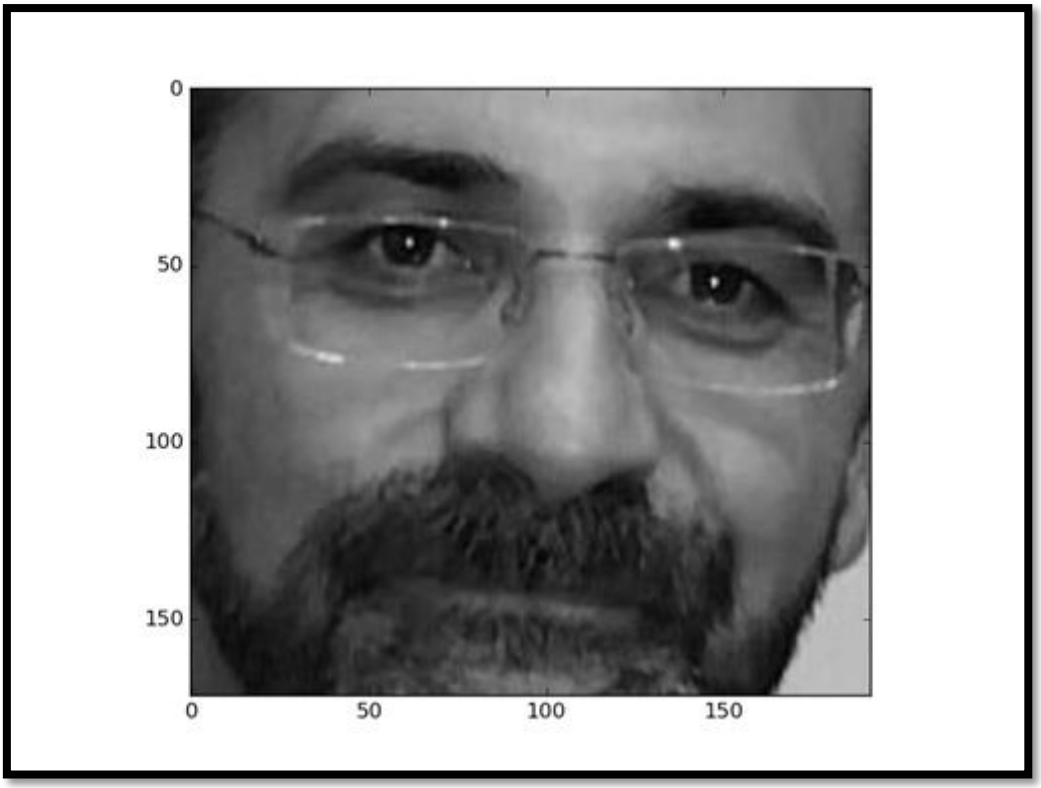

Filter concatenation
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer


kaggle
Search kaggle
Competitions
Datasets
Kernels
Discussion
Jobs
Competitions
13 active competitions
Sort By
Prize
Active
All
Entered
All Categories
Search competitions
The Nature Conservancy Fisheries Monitoring
Can you detect and classify species of fish?
Featured · 3 months to go · 182 kernels
$150,000
968 teams
Dstl Satellite Imagery Feature Detection
Can you train an eye in the sky?
Featured · 2 months to go · 57 kernels
$100,000
84 teams
Two Sigma Financial Modeling Challenge
Can you uncover predictive value in an uncertain world?
Featured · 2 months to go · 126 kernels
$100,000
1,010 teams
Outbrain Click Prediction
Can you predict which recommended content each user will click?
$25,000
832 teams

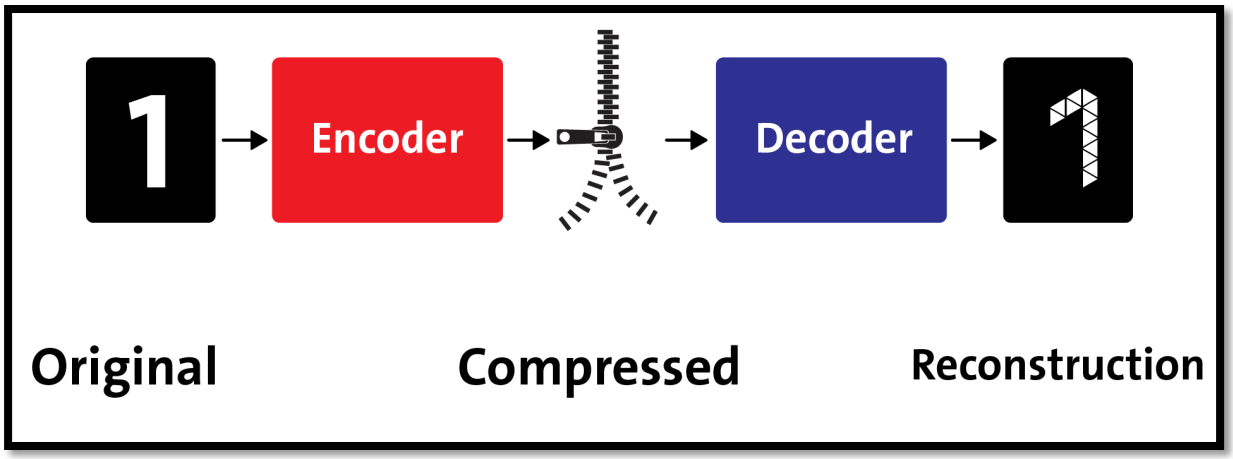

0
10
20
30
40
0
10
20
30
40

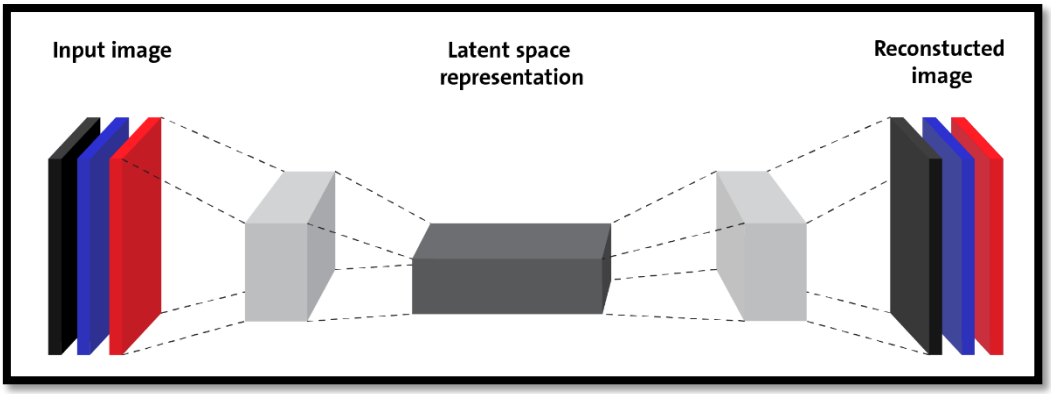

x_image
(48x48)
convolution
(5x5,s=1)
32 features
h_conv1
(48x48x32)
Max pooling
(2x2,s=2)
32 channels
h_pool1
(24x24x32)
1st convolutional layer
convolution
(3x3,s=1)
64 features
h_conv2
(24x24x64)
Max pooling
(2x2,s=2)
64 features
h_pool2
(12x12x64)
2nd convolutional layer
A

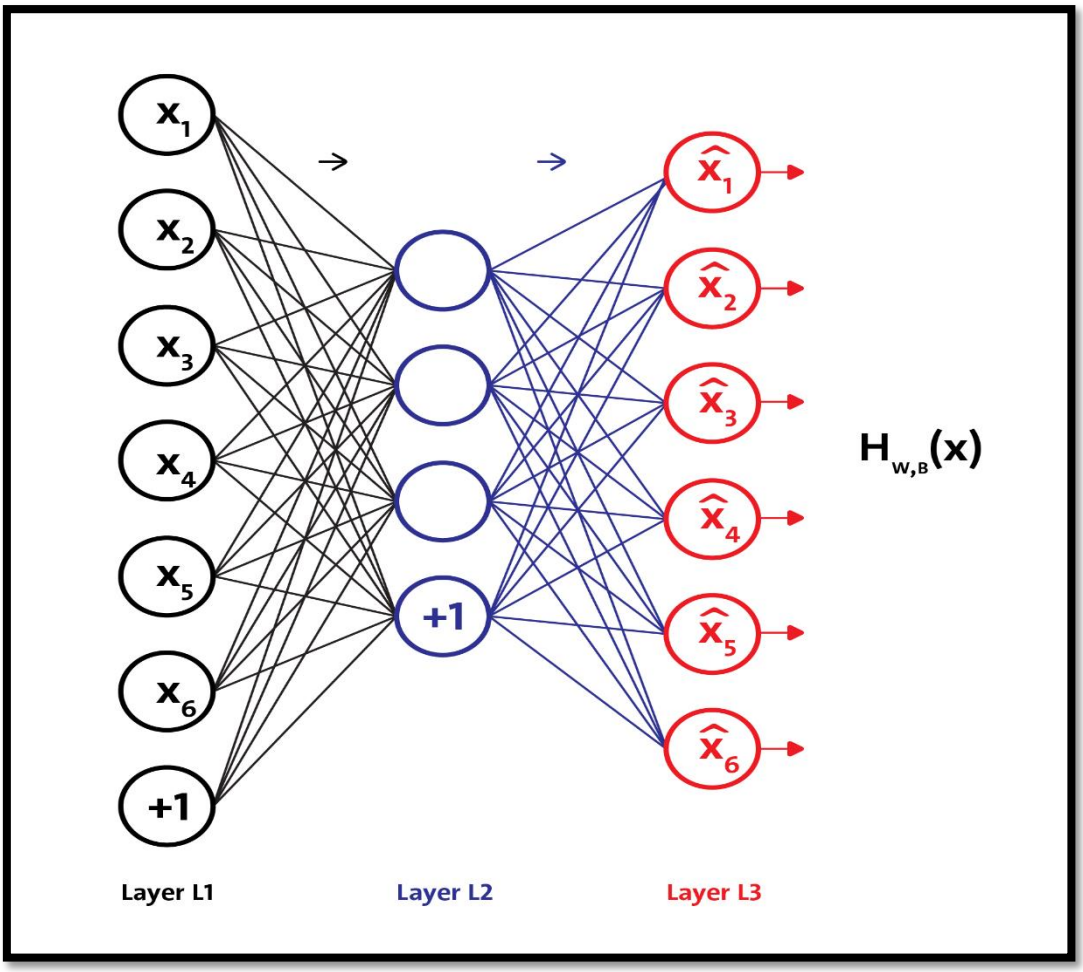

A
Fully connected layer
Readout layer
Reshape
12 * 12 * 64 Tensor ➔ 9,216x1 vector
256 neurons
10 digits

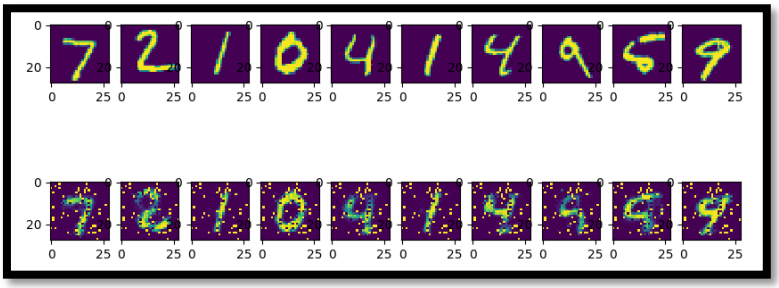

0
50
100
150
0
50
100
150

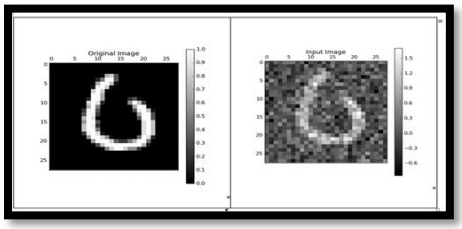
Chapter 5: Optimizing TensorFlow Autoencoders

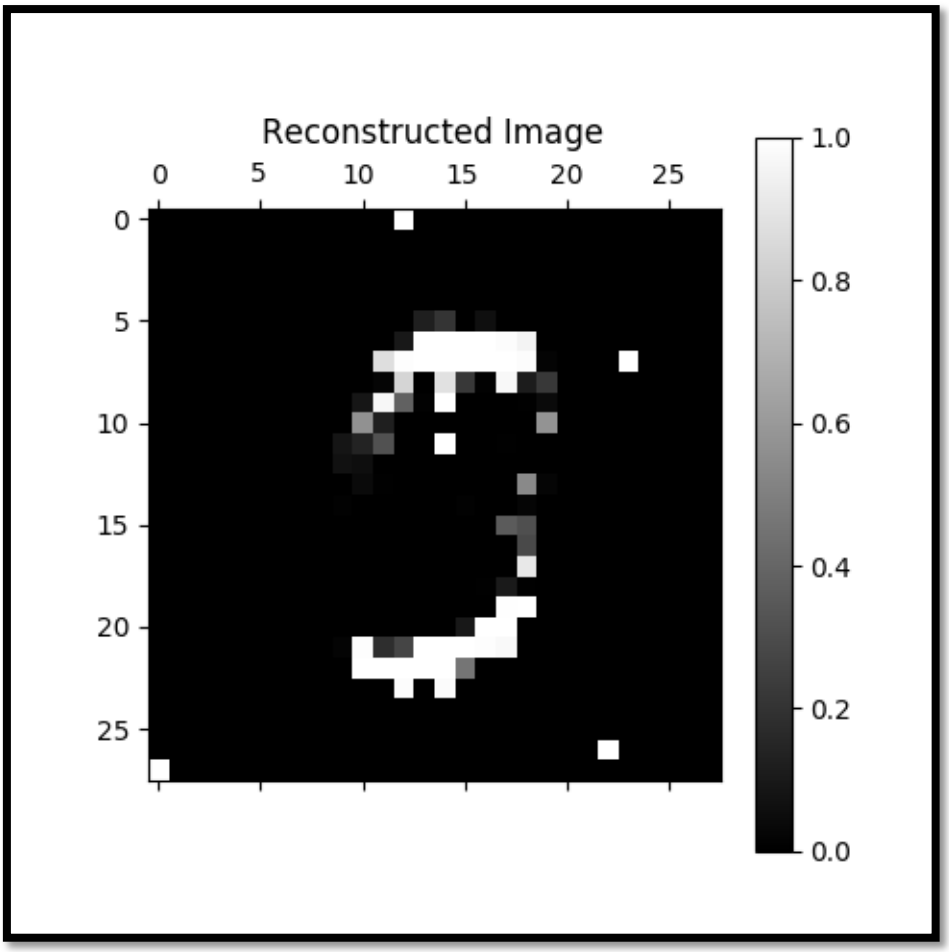


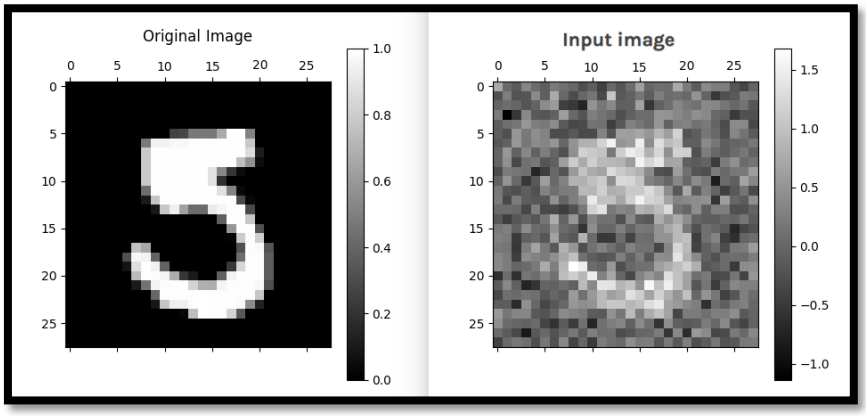


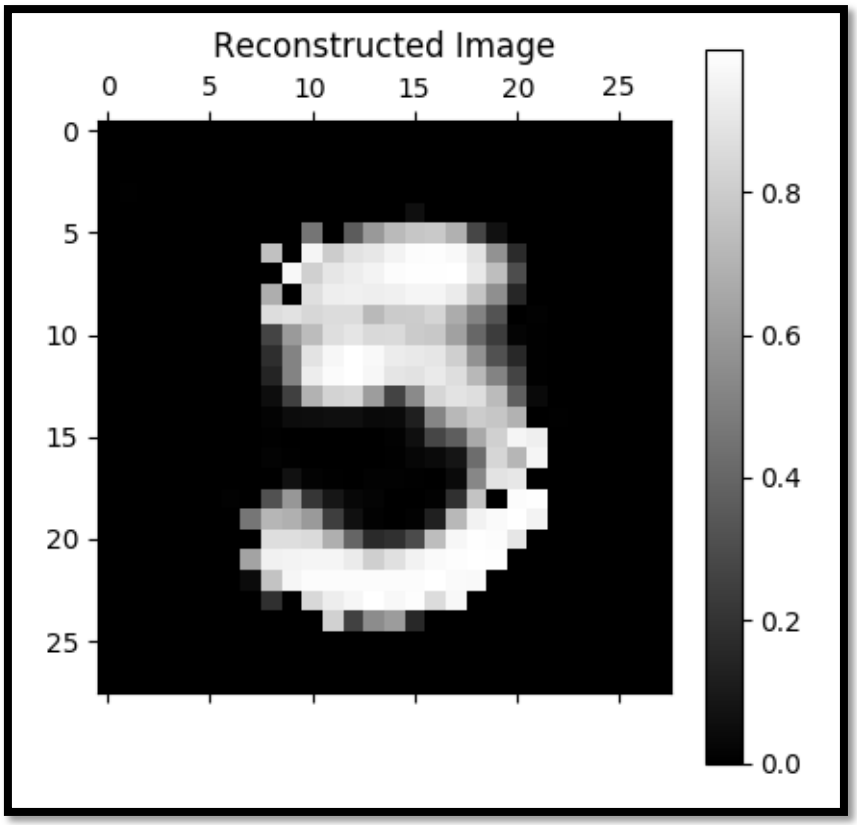

x1
x2
x3
x4
x5
x6
+1
+1
x̂1
x̂2
x̂3
x̂4
x̂5
x̂6
Hw,B(x)
Layer L1
Layer L2
Layer L3

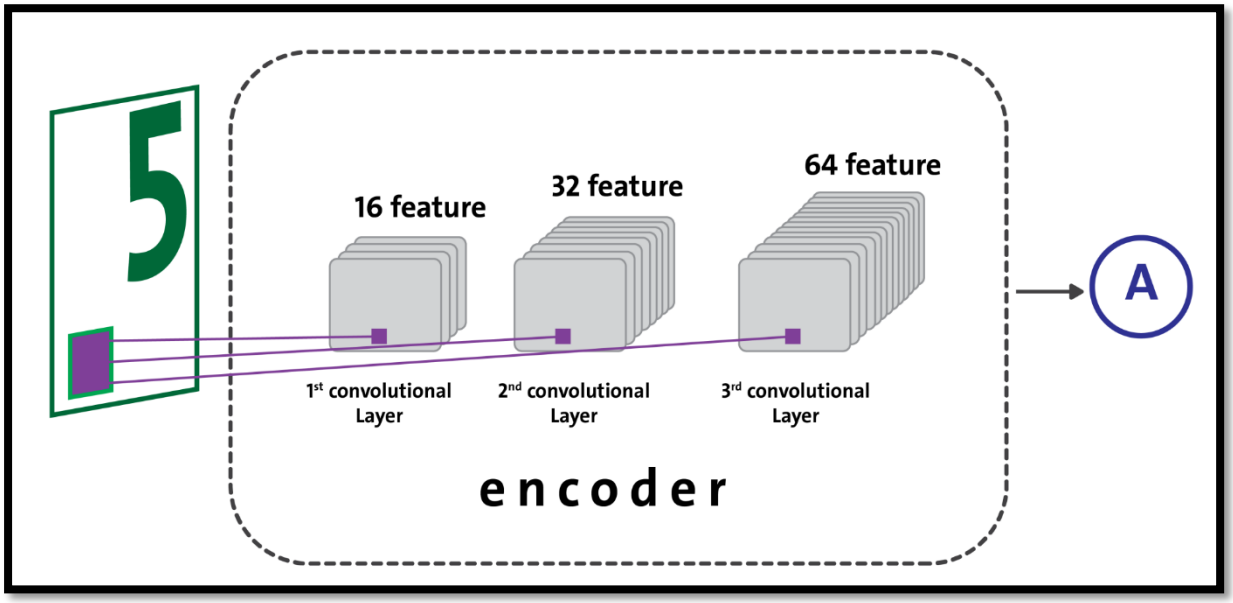


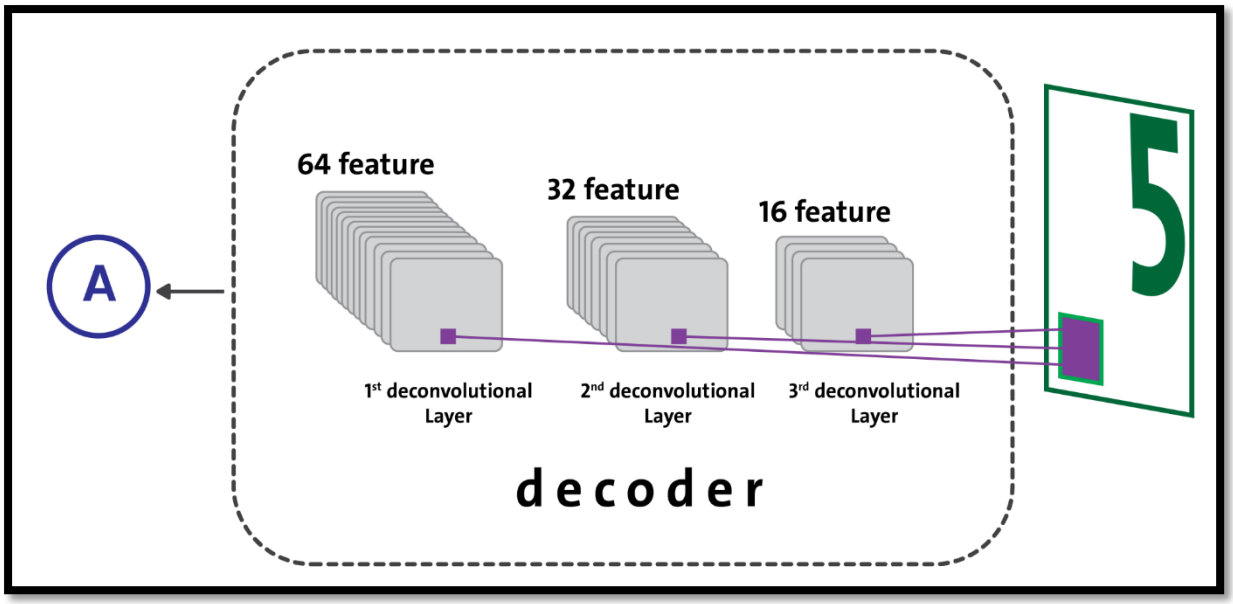

Original Image
Input Image

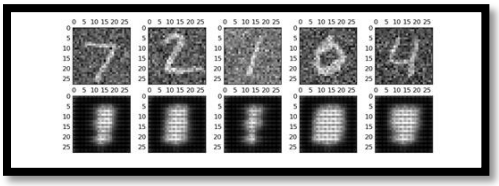

Reconstructed Image
0
5
10
15
20
25
1.0
0.8
0.6
0.4
0.2
0.0

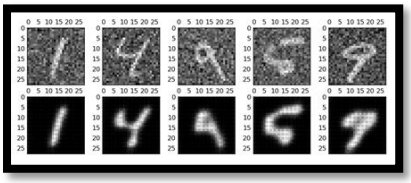

Original Image
Input image

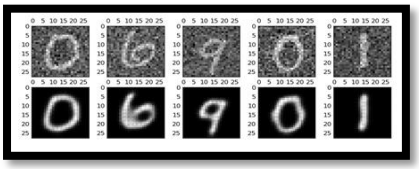

Reconstructed Image
0
5
10
15
20
25
0.8
0.6
0.4
0.2
0.0

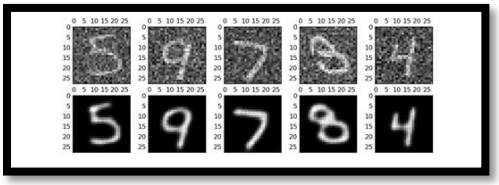

5
16 feature
32 feature
64 feature
1st convolutional Layer
2nd convolutional Layer
3rd convolutional Layer
encoder
A

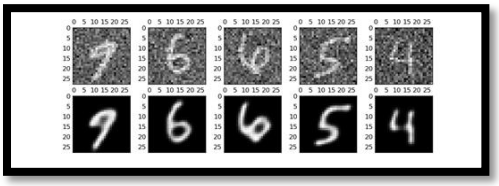

64 feature
32 feature
16 feature
A
1st deconvolutional Layer
2nd deconvolutional Layer
3rd deconvolutional Layer
decoder
5





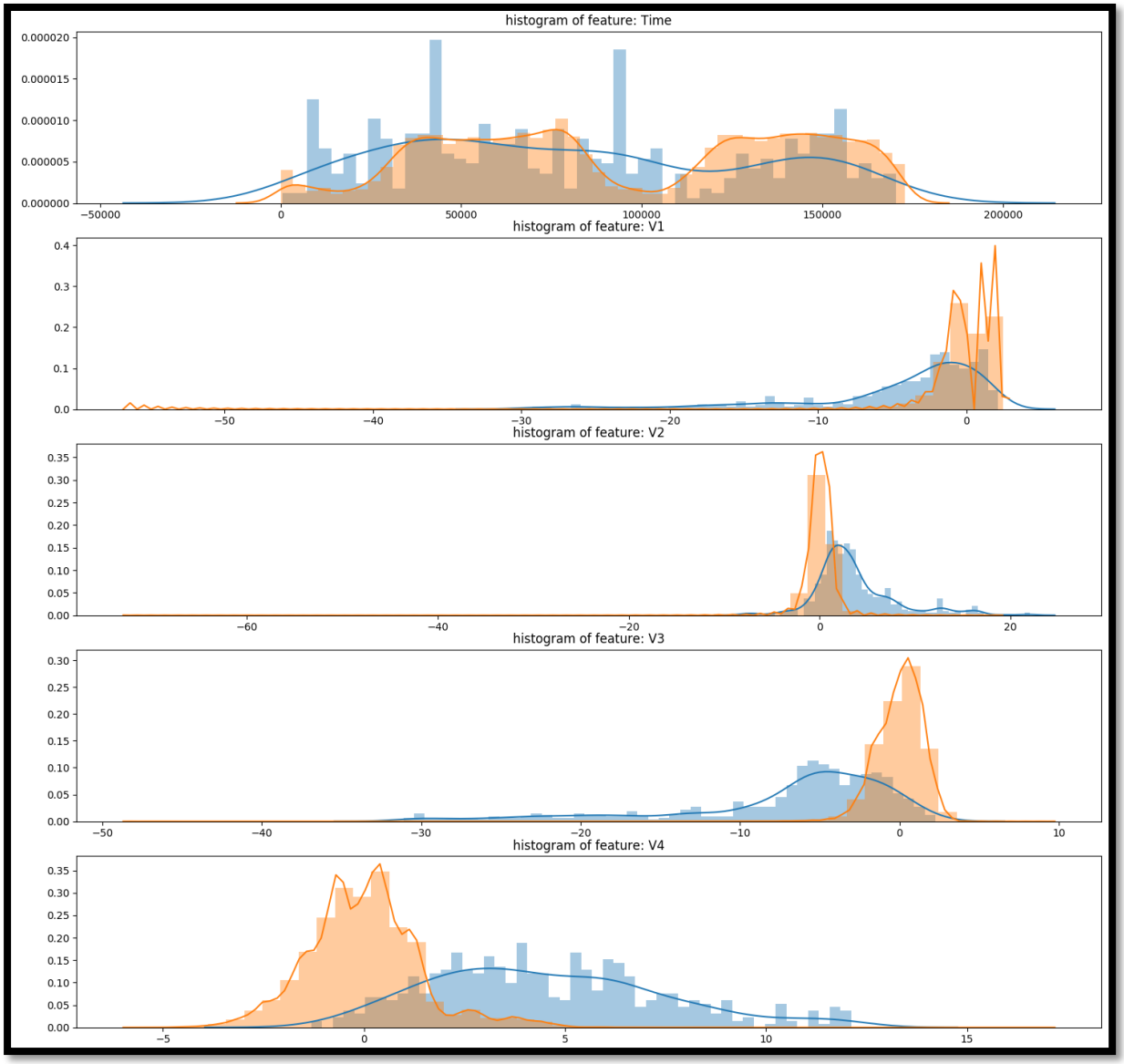


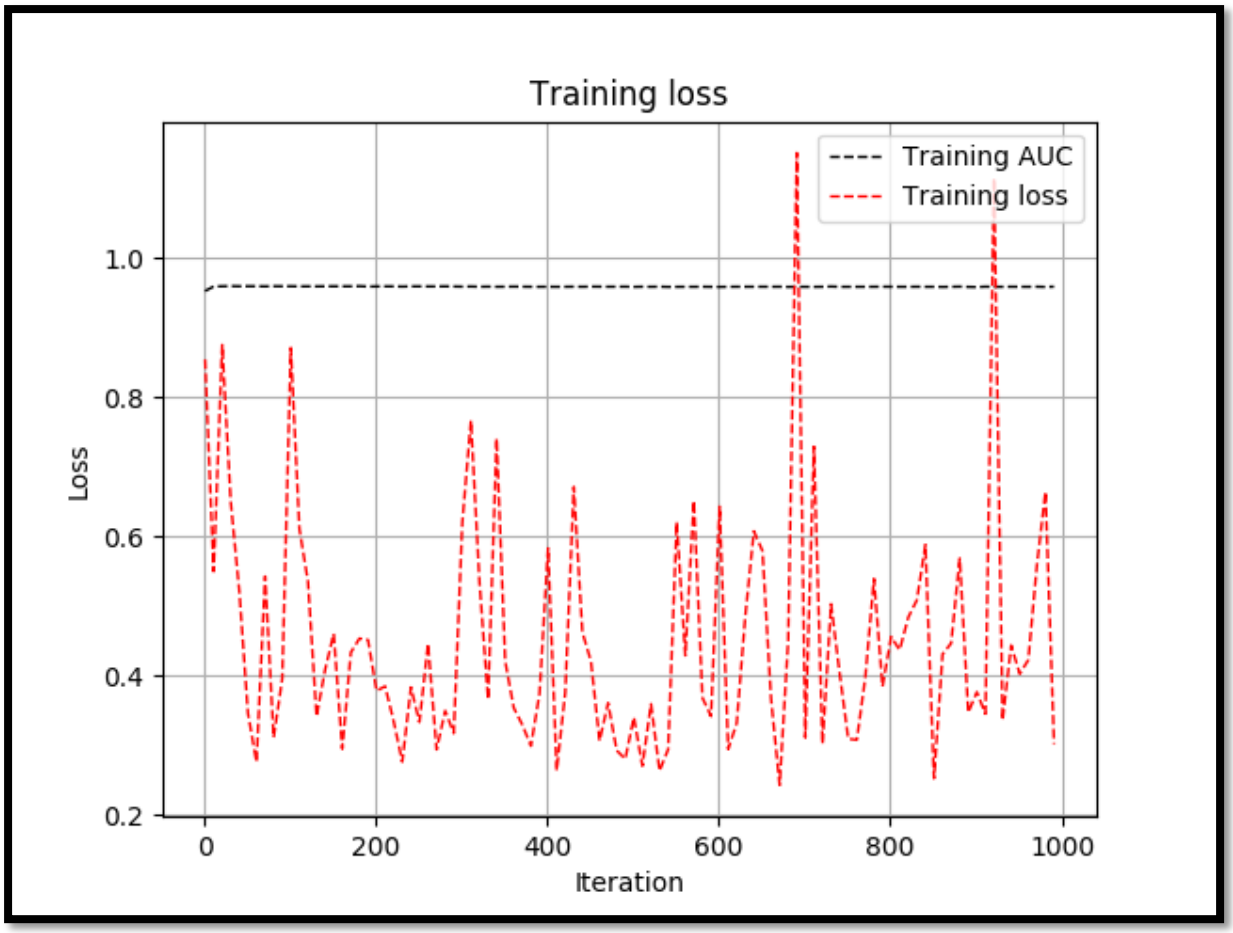


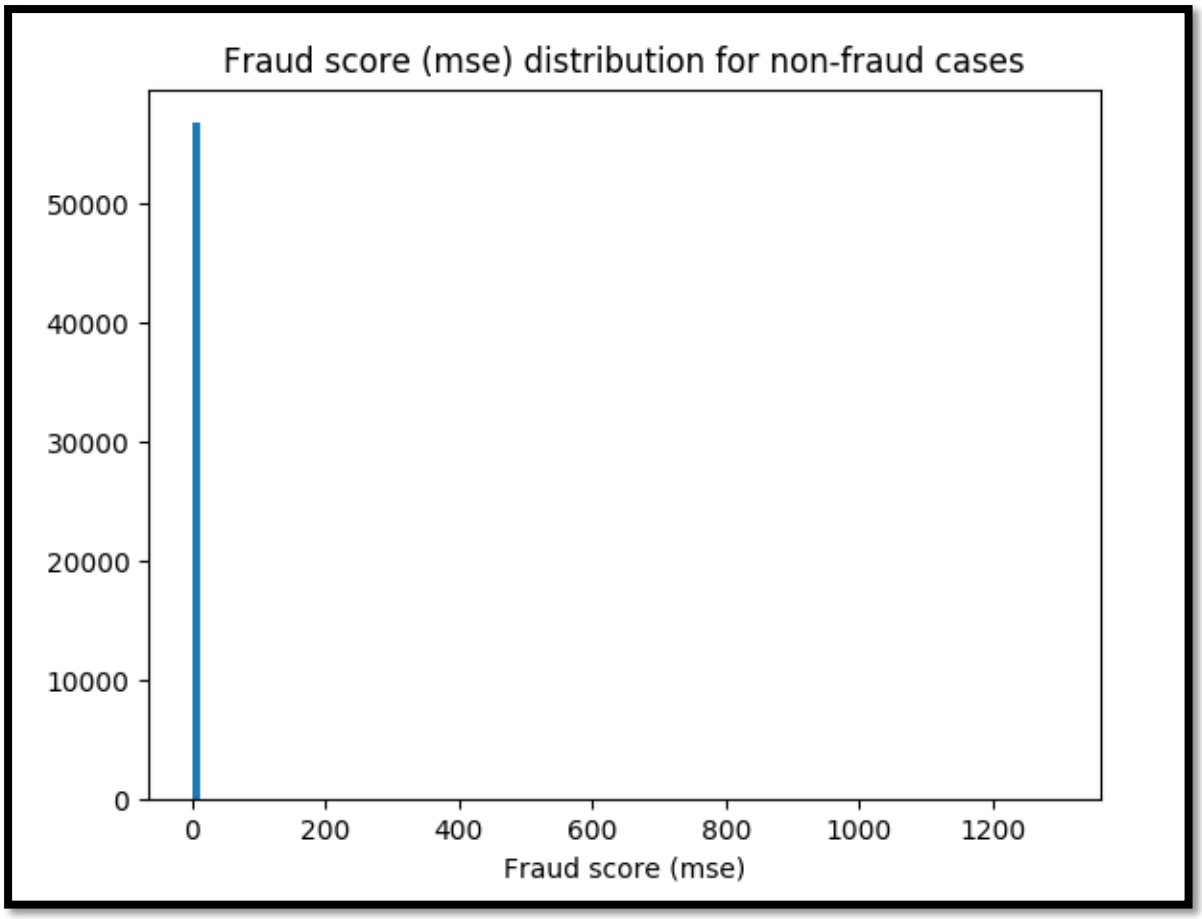


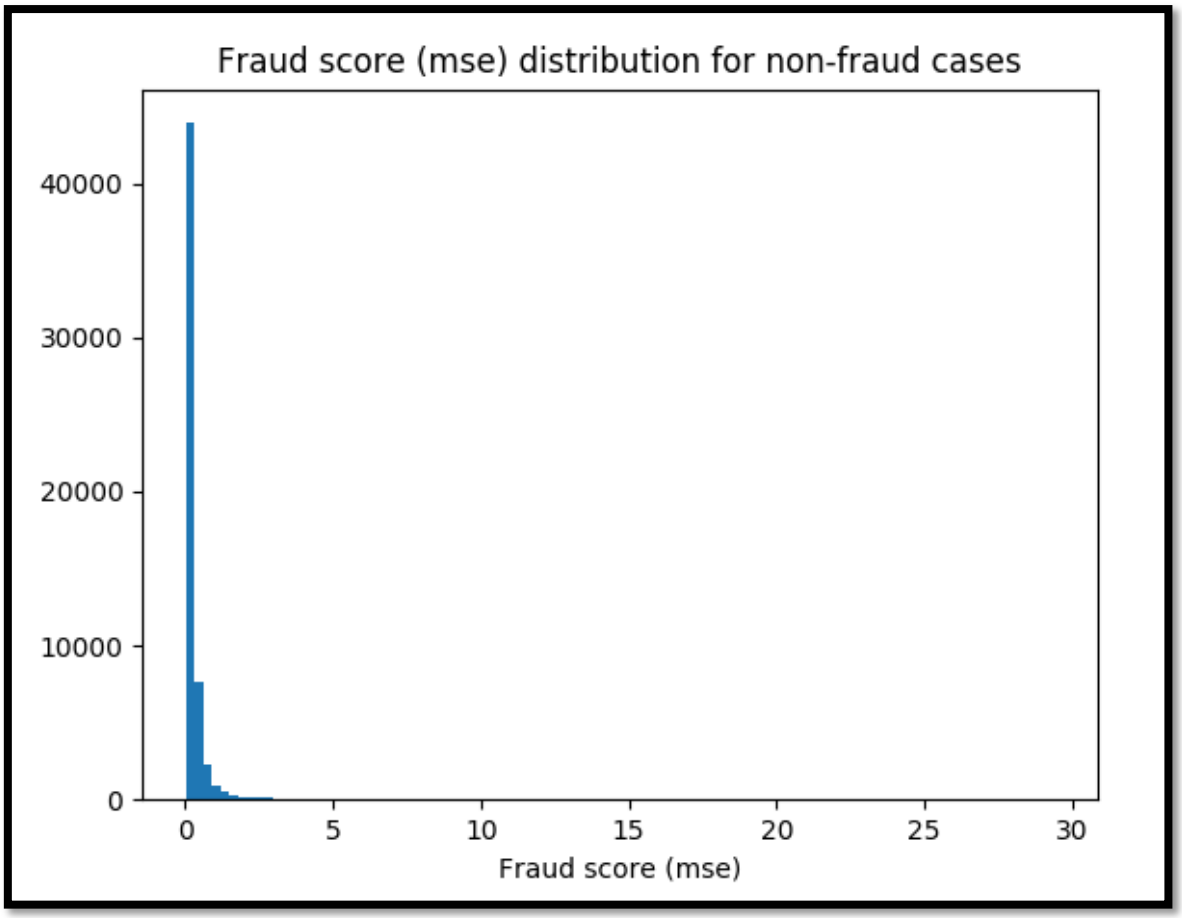
```
+----+--------------------+--------------------+--------------------+--------------------+------+-----+
|Time|                  V1|                  V2|                 V26|                 V27|Amount|Class|
+----+--------------------+--------------------+--------------------+--------------------+------+-----+
|   0|  -1.3598071336738|-0.0727811733098497|-0.1891148483888824|  0.1335583776740387|149.62|    0|
|   0|  1.19185711131486|  0.26615071205963|  0.1258945532368176|-0.0089830991432281|  2.69|    0|
|   1| -1.35835406159823| -1.34016307473609| -0.1390965715141407| -0.0553527940384261|378.66|    0|
|   1|-0.96627171715720807|-0.18522601808828981|-0.2219288444584807|  0.0627228487293033| 123.5|    0|
|   2| -1.15823309349523|  0.87773675484845...|  0.5029222241815697|  0.2194222295133481| 69.99|    0|
|   2|-0.42596588441245...|  0.96052304488229...|  0.1059147790979577|  0.2538442247393377|  3.67|    0|
|   4|  1.22965763450793|  0.14100350704932...|-0.2572368459171397|  0.0345074297438413|  4.99|    0|
|   7|-0.64426944234814...|  1.41796354547385...|-0.0516342969262494|  -1.206921808942587|  40.8|    0|
|   7| -0.8942860822028...|  0.28615719627654...|-0.3841573077022947|  0.0117473564581996|  93.2|    0|
|   9| -0.3382617524257...|  1.11959337641566...|  0.0941988339514961|  0.2462193046199267|  3.68|    0|
+----+--------------------+--------------------+--------------------+--------------------+------+-----+
only showing top 10 rows
```

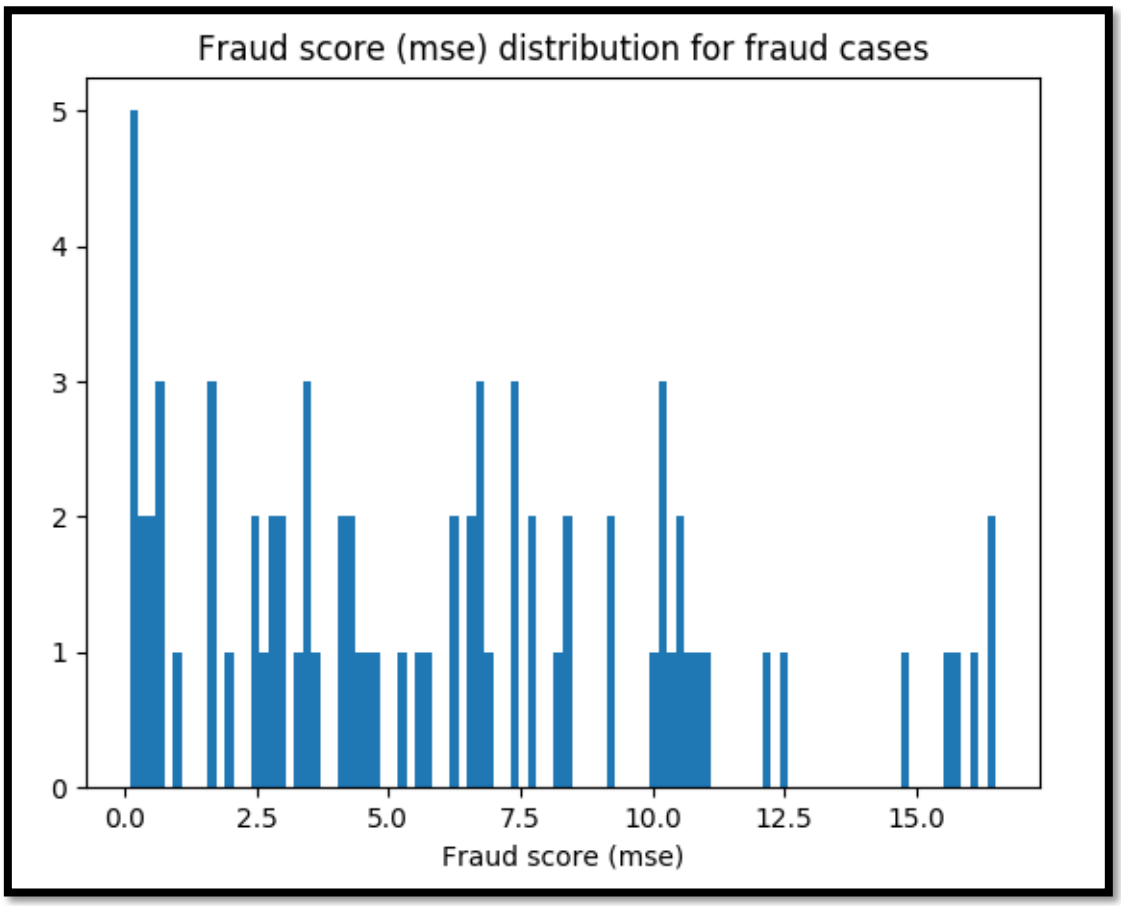
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431



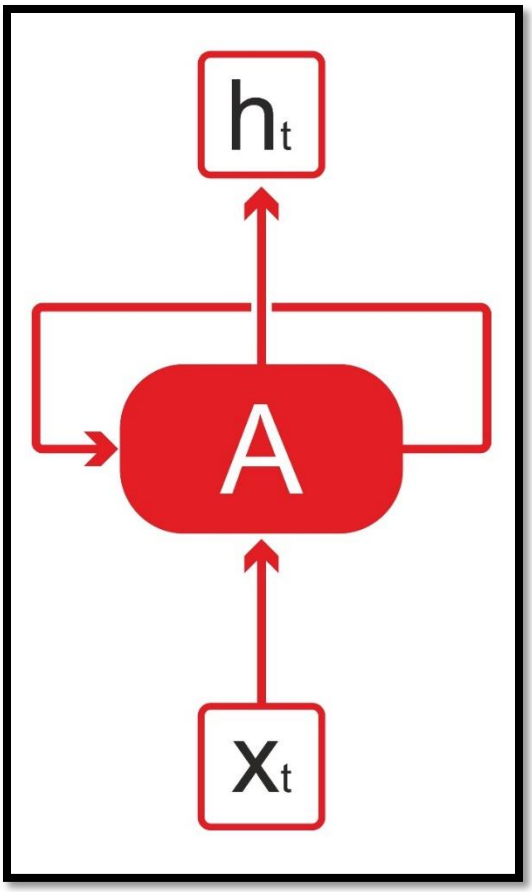

Training loss
Training AUC
Training loss
Loss
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000
Iteration

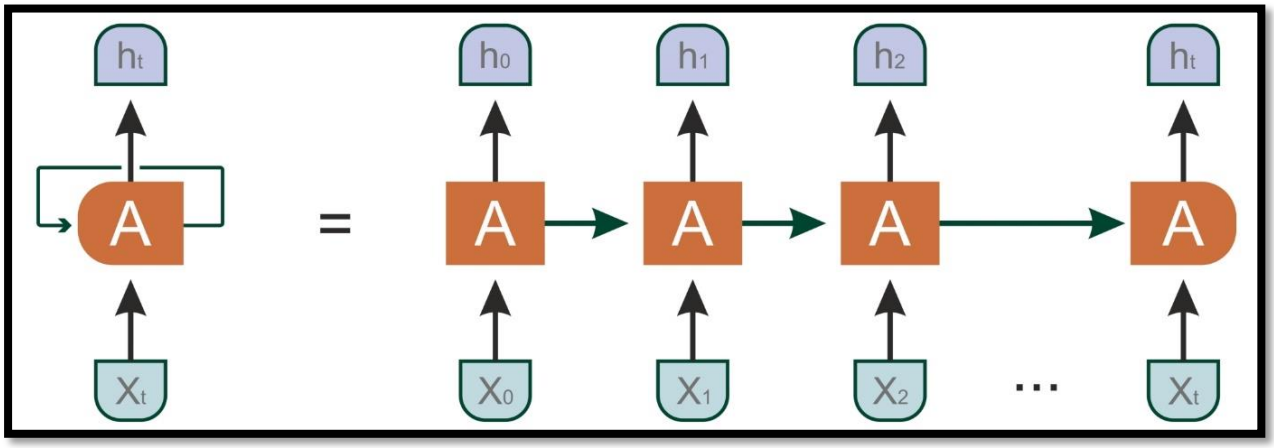

Fraud score (mse) distribution for non-fraud cases
50000
40000
30000
20000
10000
0
0
200
400
600
800
1000
1200
Fraud score (mse)

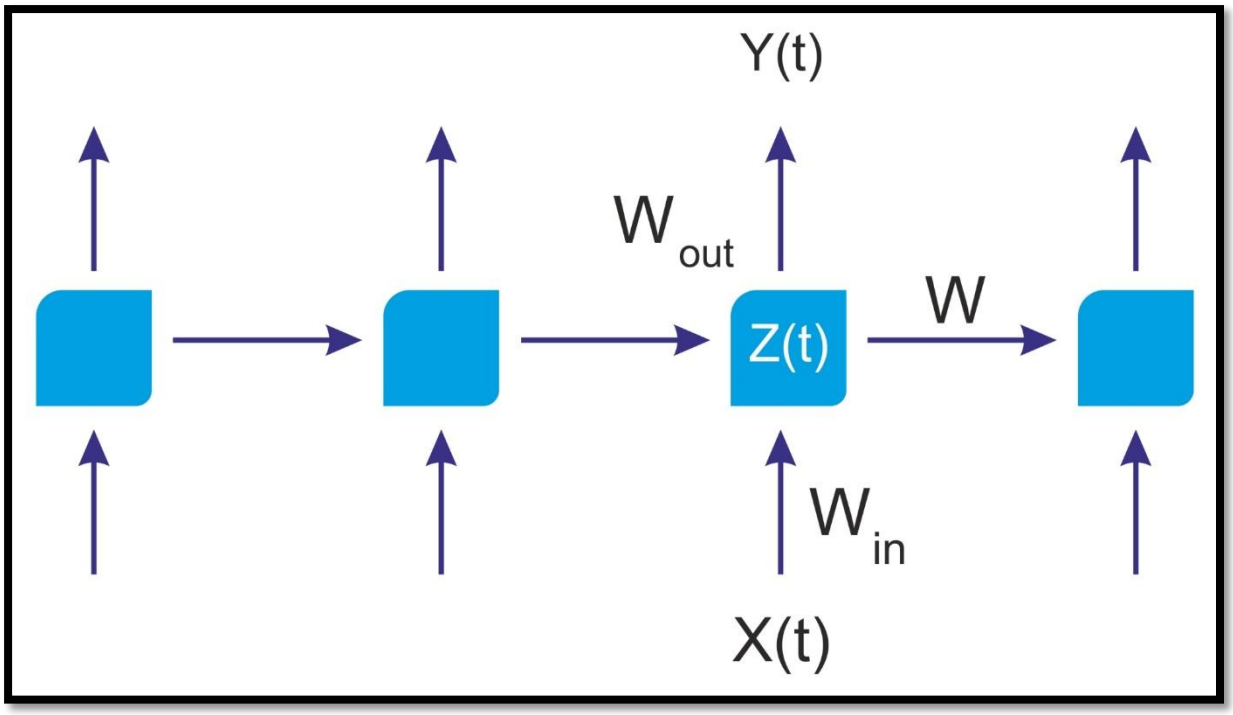

Fraud score (mse) distribution for non-fraud cases
40000
30000
20000
10000
0
0
5
10
15
20
25
30
Fraud score (mse)

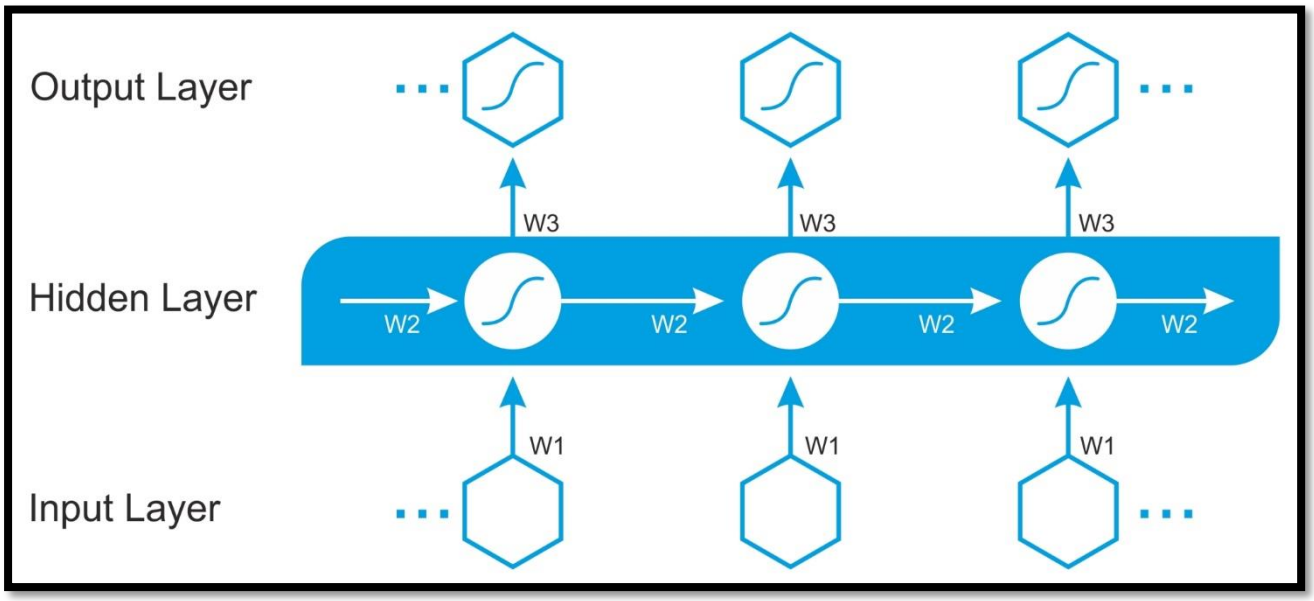

Fraud score (mse) distribution for fraud cases
5
4
3
2
1
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
Fraud score (mse)

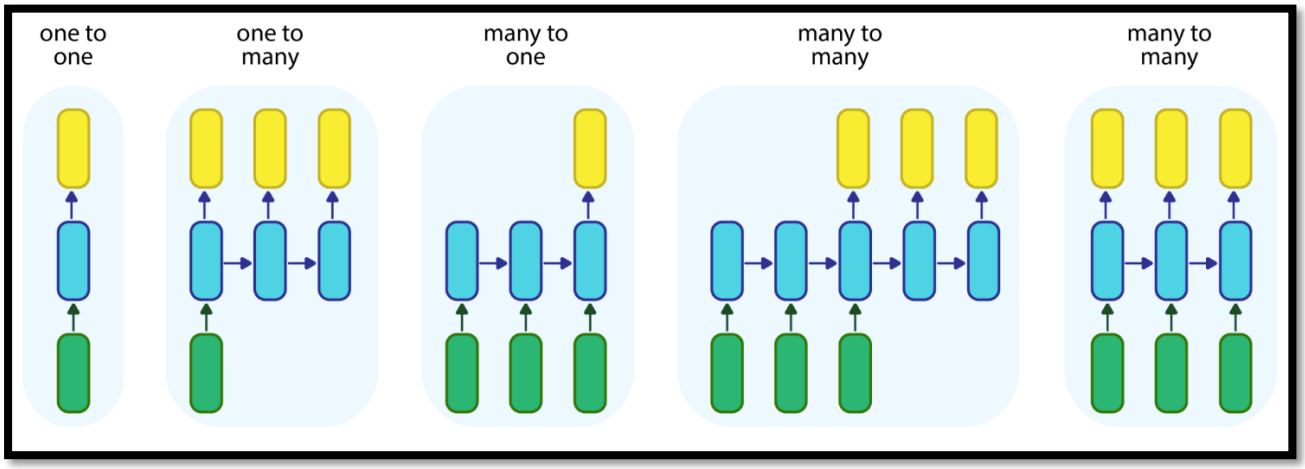
Chapter 6: Recurrent Neural Networks

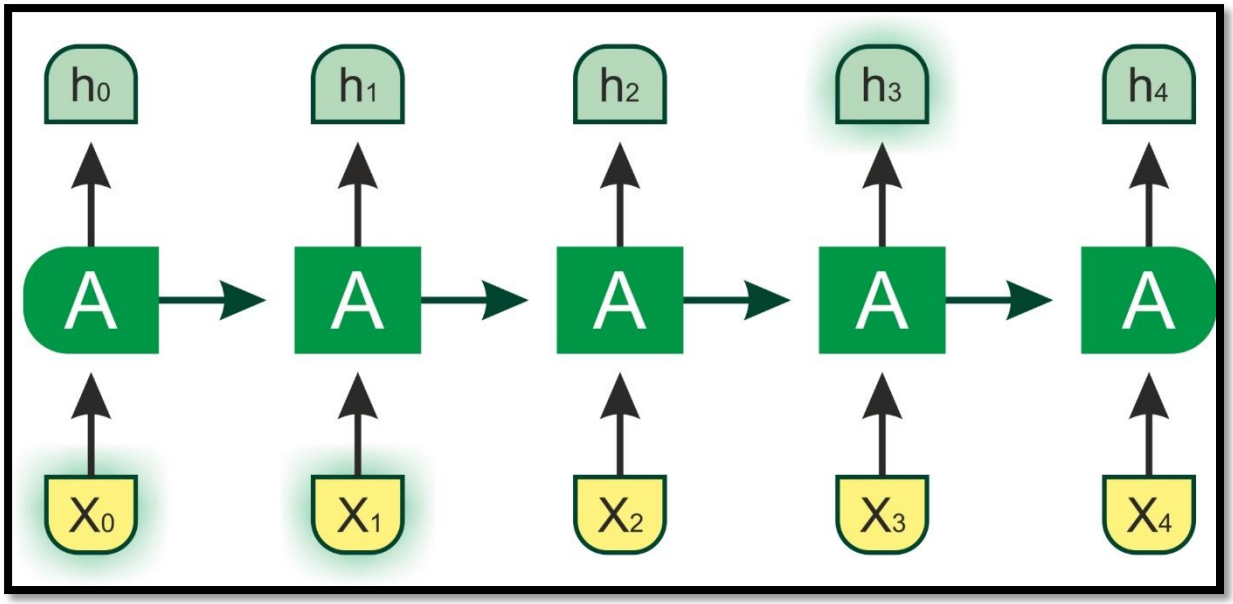


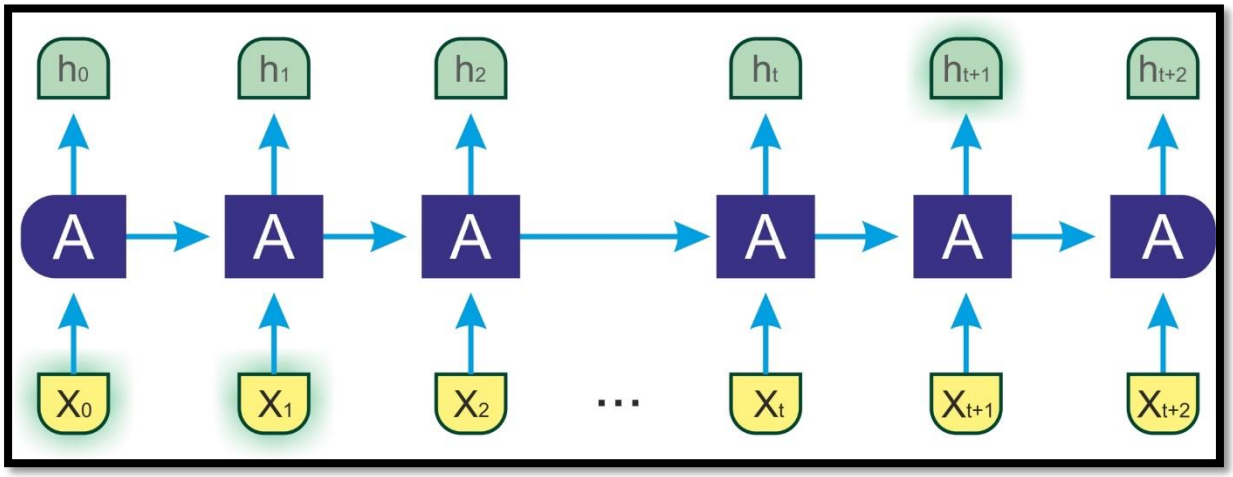


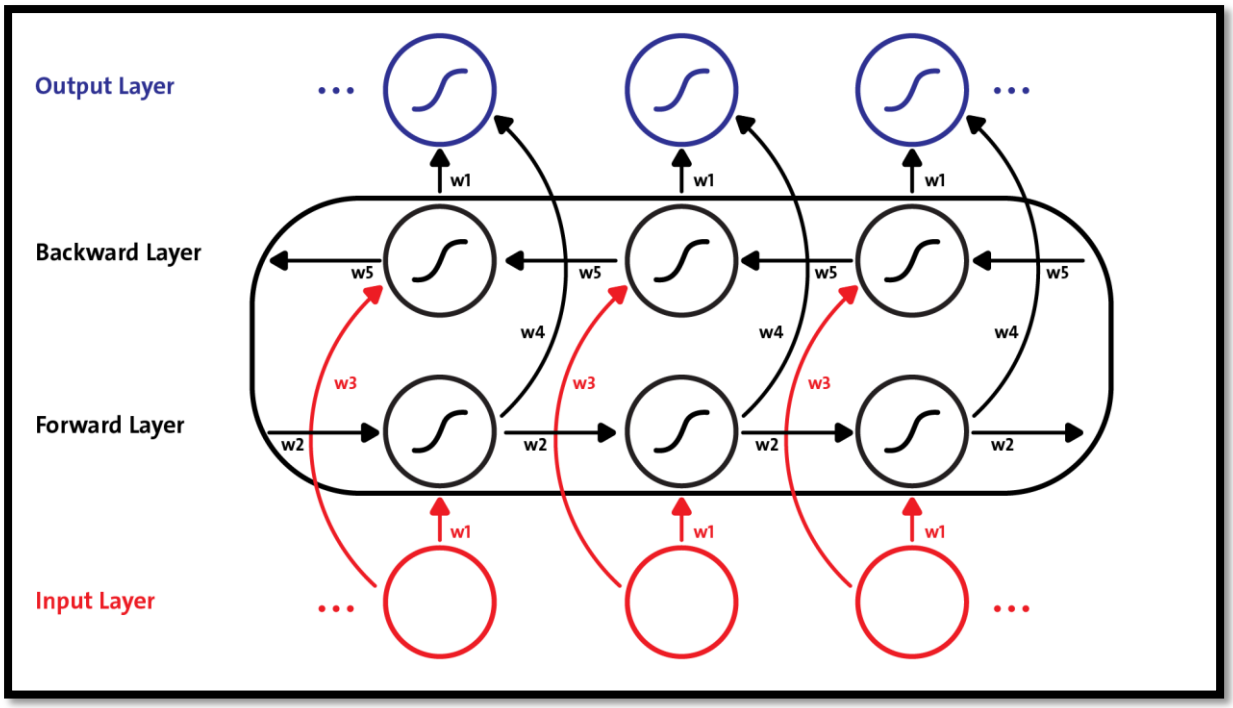

Y(t)
W out
W
Z(t)
W in
X(t)

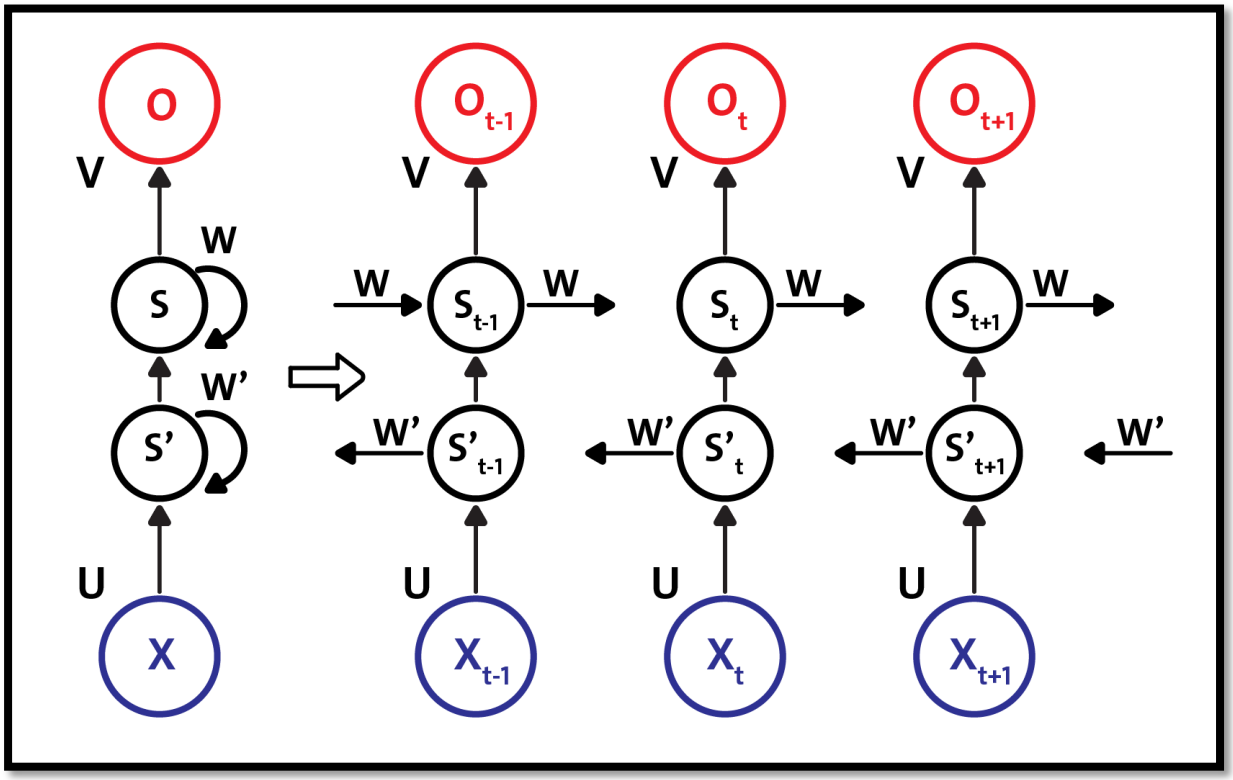

Output Layer
Hidden Layer
Input Layer
W3
W3
W3
W2
W2
W2
W2
W1
W1
W1

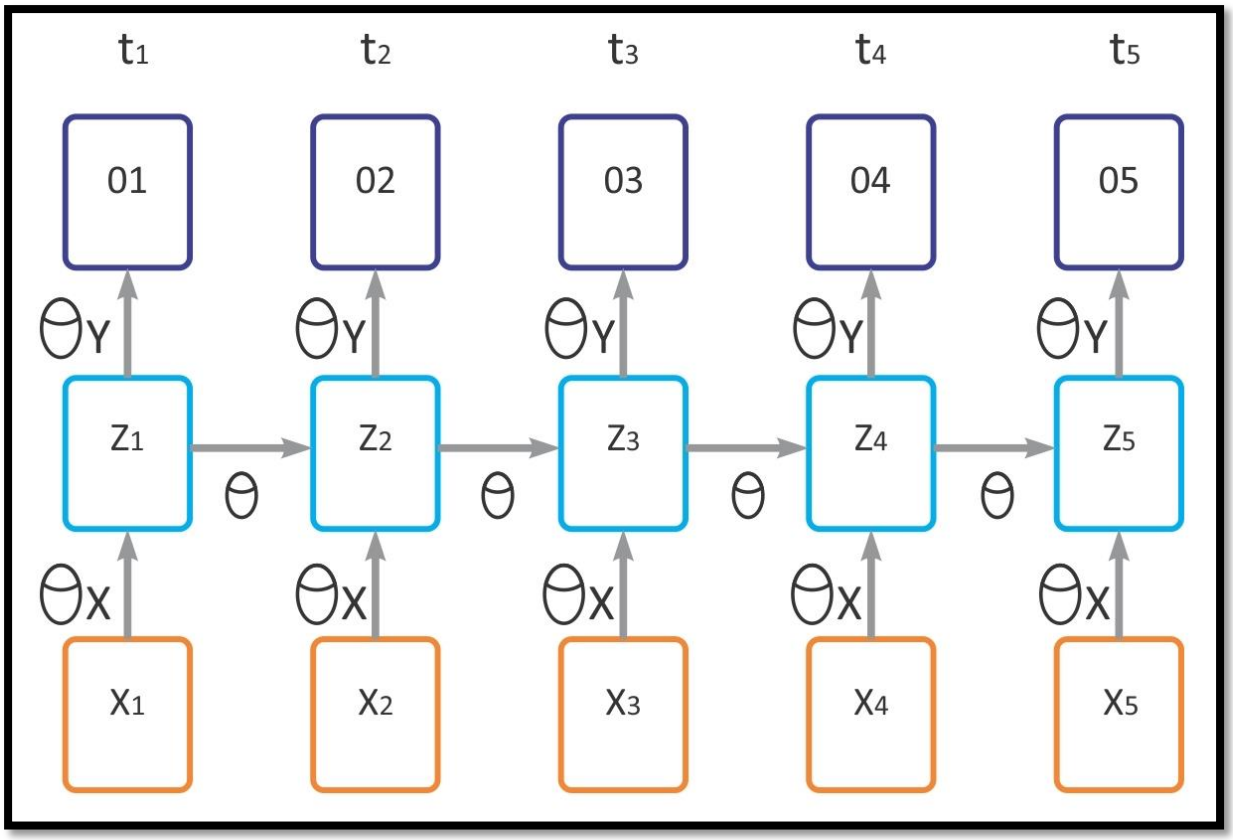

one to one
one to many
many to one
many to many
many to many

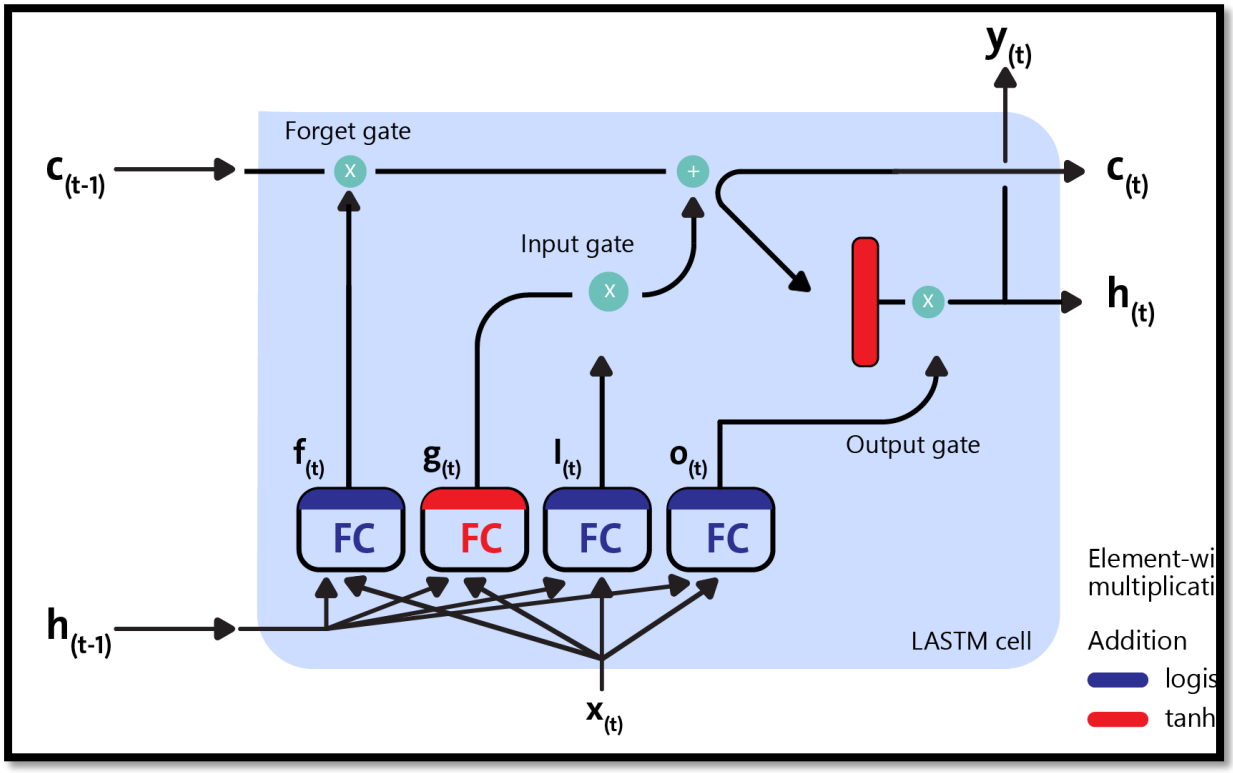

h0
h1
h2
h3
h4
A
A
A
A
A
X0
X1
X2
X3
X4

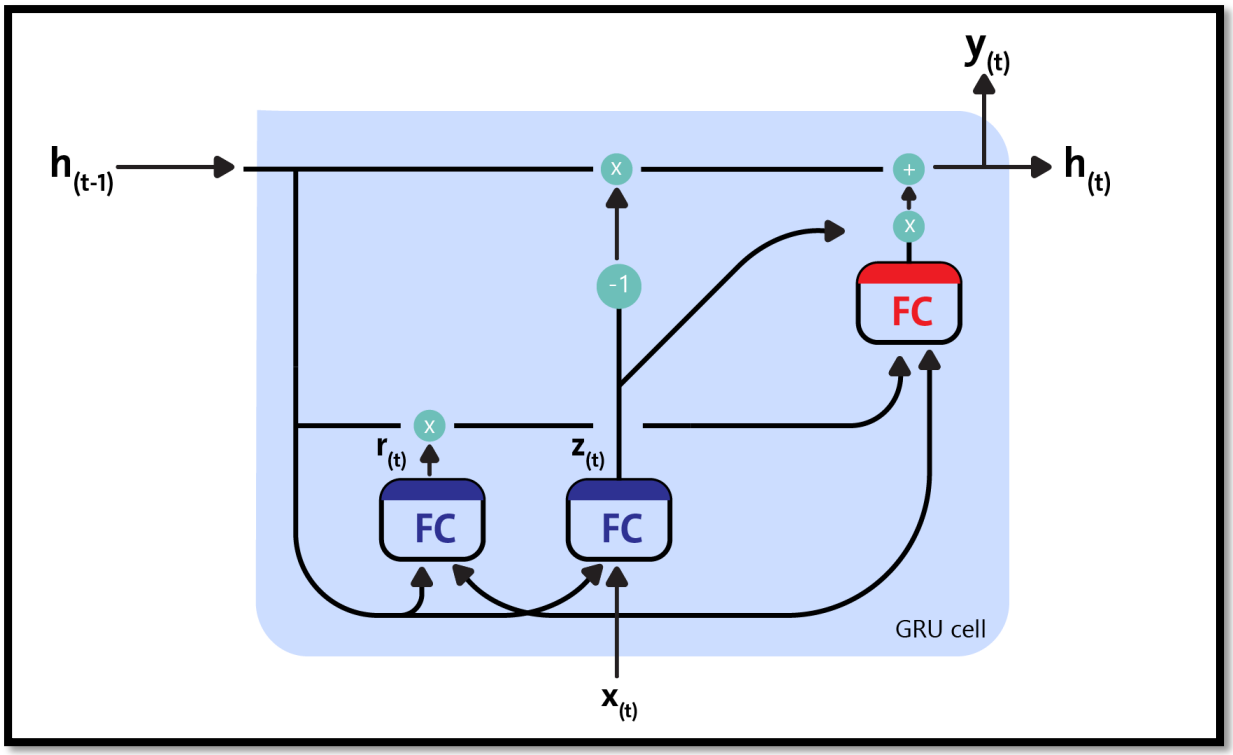

h0
h1
h2
ht
ht+1
ht+2
A
A
A
A
A
A
X0
X1
X2
...
Xt
Xt+1
Xt+2

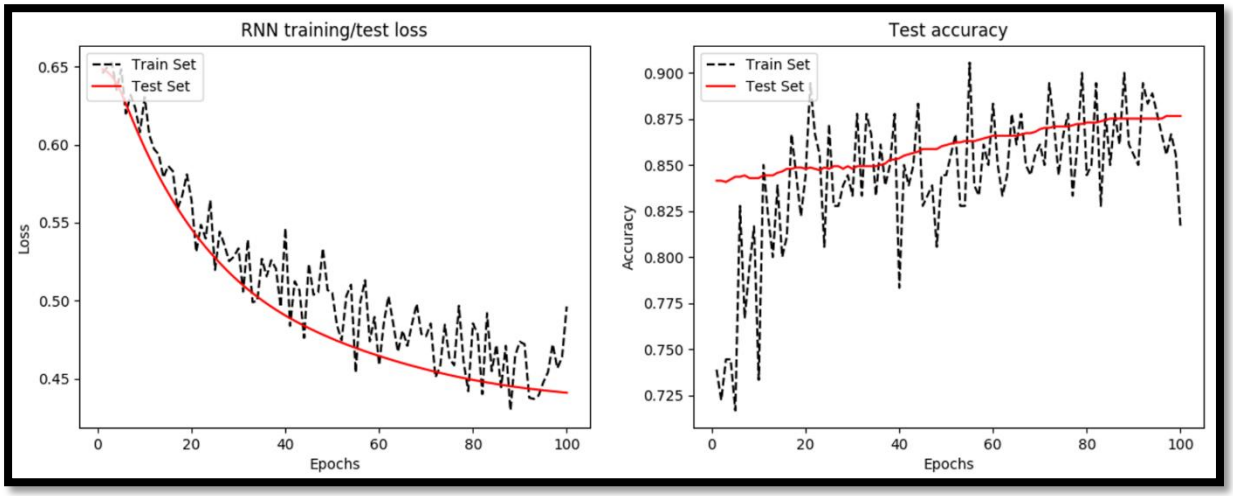

Output Layer
Backward Layer
Forward Layer
Input Layer
w1
w2
w3
w4
w5


O
V
W
S
W'
S'
U
X
O t-1
S t-1
S' t-1
X t-1
O t
S t
S' t
X t
O t+1
S t+1
S' t+1
X t+1

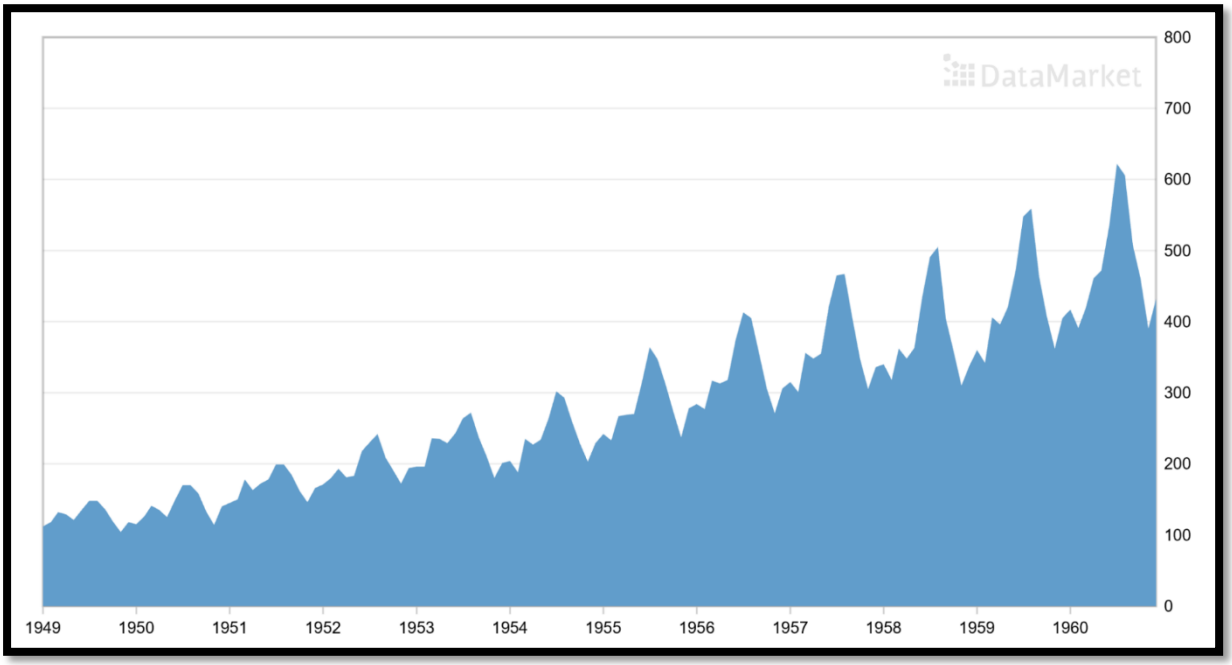

t1
t2
t3
t4
t5
01
02
03
04
05
ΘY
Z1
Z2
Z3
Z4
Z5
Θ
ΘX
X1
X2
X3
X4
X5

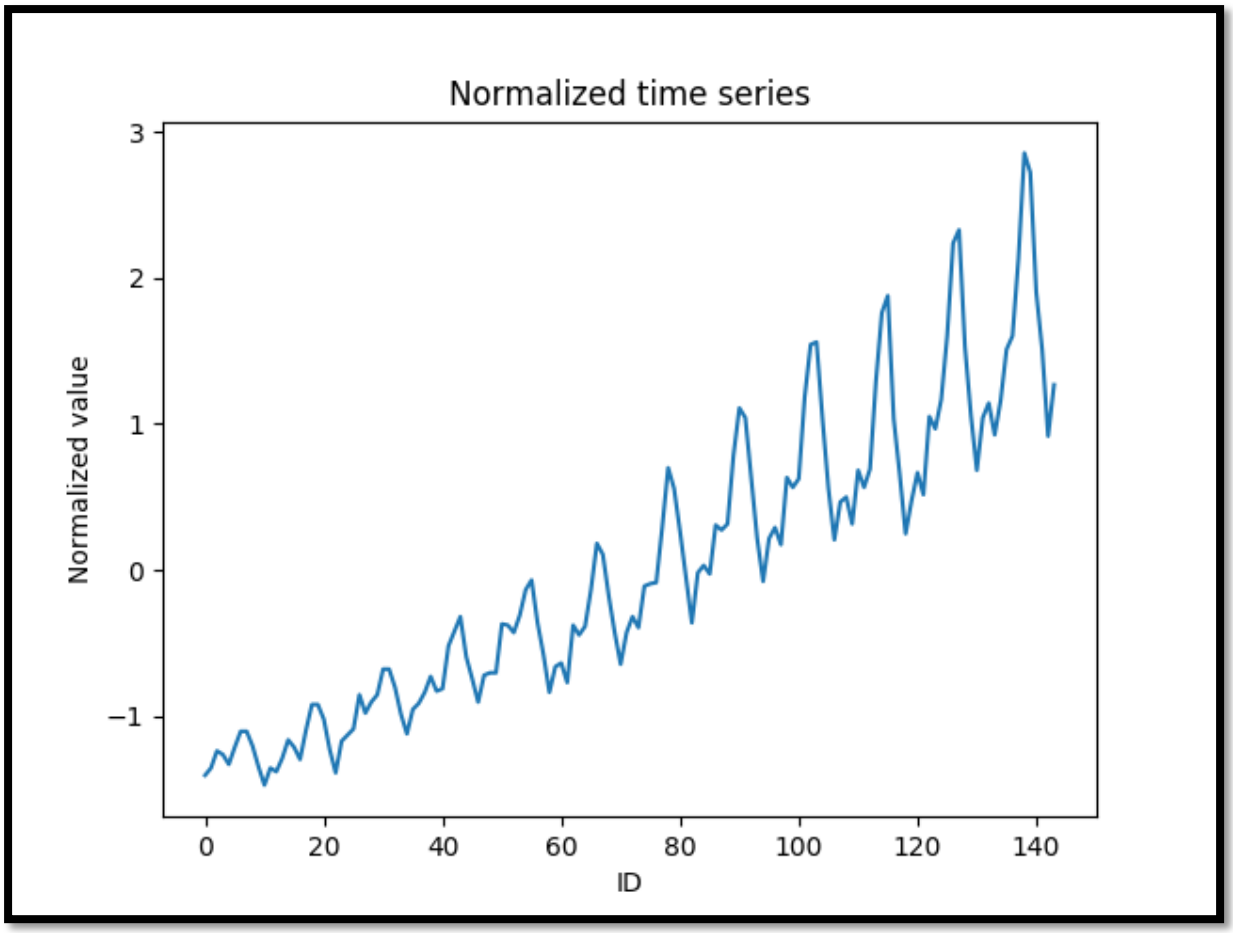

y(t)
Forget gate
c(t-1)
c(t)
Input gate
h(t)
Output gate
f(t)
g(t)
i(t)
o(t)
FC
FC
FC
FC
h(t-1)
x(t)
LASTM cell
Element-wi
multiplicati
Addition
logis
tanh

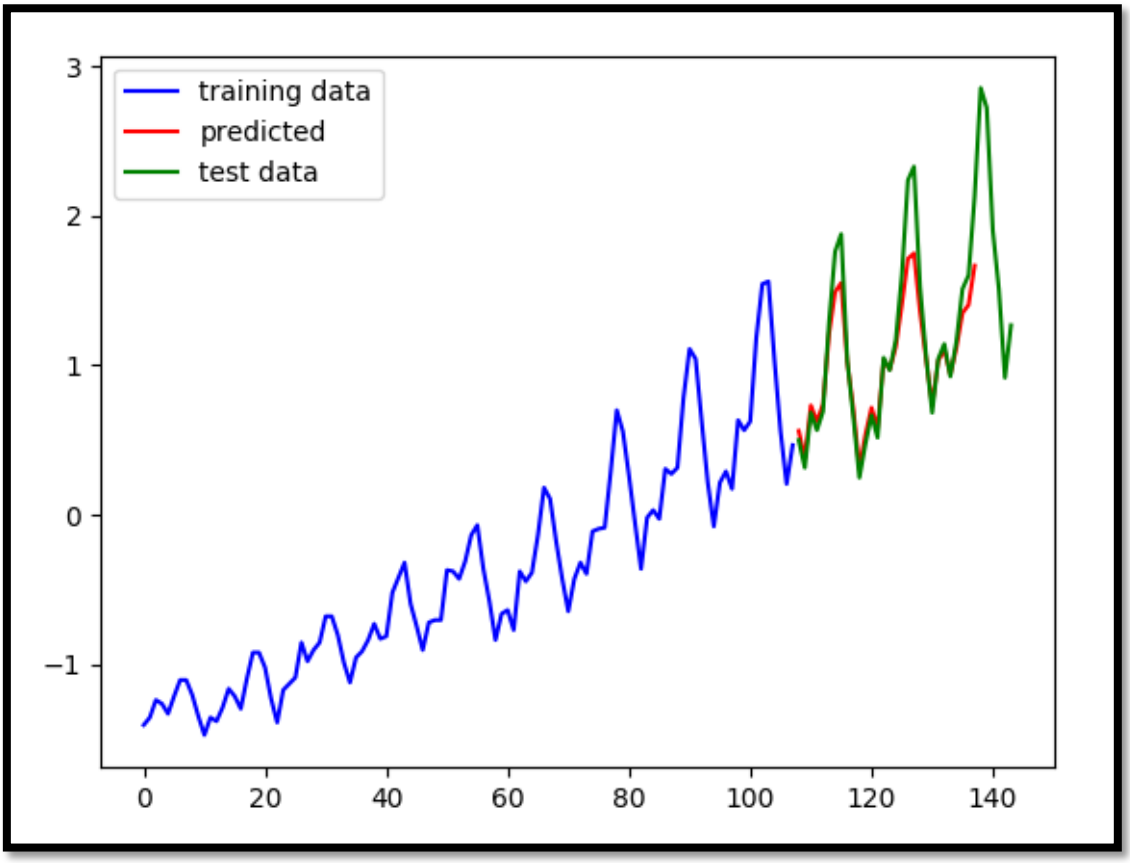

$y_{(t)}$
$h_{(t-1)}$
$h_{(t)}$
x
+
x
-1
FC
$r_{(t)}$
$z_{(t)}$
x
FC
FC
GRU cell
$x_{(t)}$

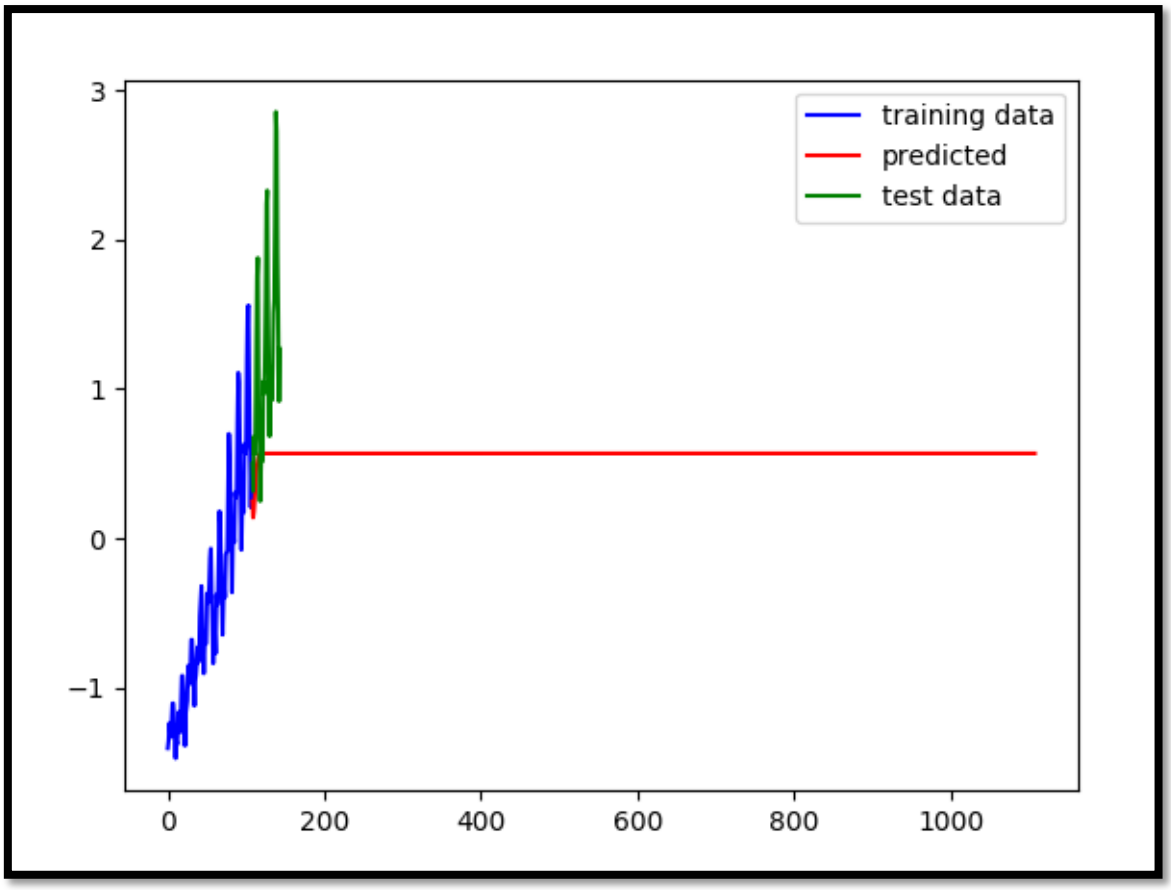

RNN training/test loss
Train Set
Test Set
Loss
0.65
0.60
0.55
0.50
0.45
0
20
40
60
80
100
Epochs
Test accuracy
Train Set
Test Set
Accuracy
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0
20
40
60
80
100
Epochs

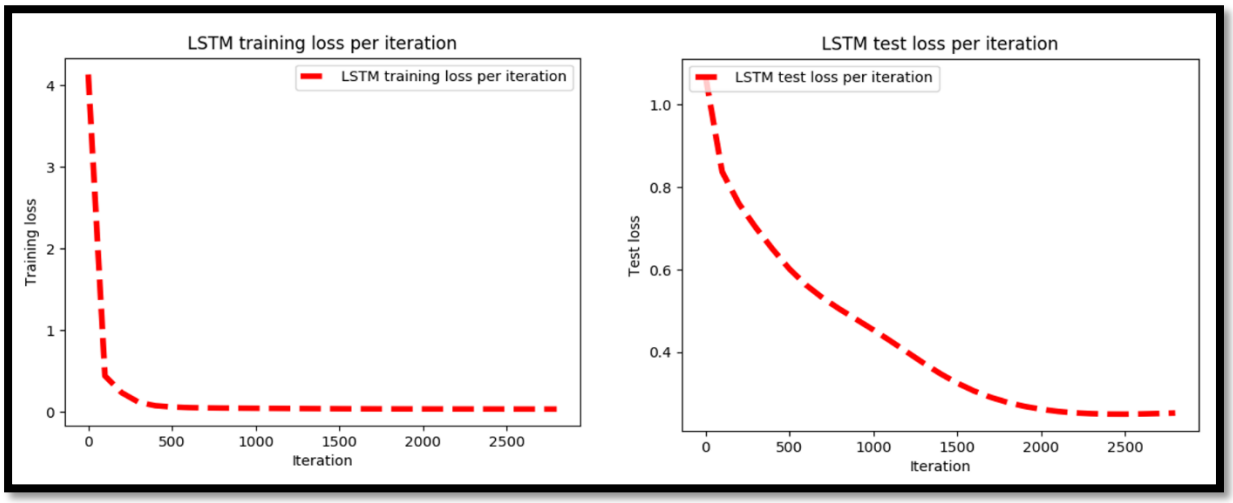
Dataset title	**International airline passengers: monthly totals in thousands. Jan 49 – Dec 60**
Last updated	1 Feb 2014, 19:52
Last updated by source	20 Jun 2012
Provider	Time Series Data Library
Provider source	Box & Jenkins (1976)
Source URL	http://datamarket.com/data/list/?q=provider:tsdl
Units	Thousands of passengers
Dataset metrics	144 fact values in 1 timeseries.
Time granularity	Month
Time range	Jan 1949 – Dec 1960
Language	English
License	Default open license
License summary	This data release is licensed as follows: You may copy and redistribute the data. You may make derivative works from the data. You may use the data for commercial purposes. You may not sublicense the data when redistributing it. You may not redistribute the data under a different license. Source attribution on any use of this data: Must refer source.
Description	Transport and tourism, Source: Box & Jenkins (1976), in file: data/airpass, Description: International airline passengers: monthly totals in thousands. Jan 49 – Dec 60

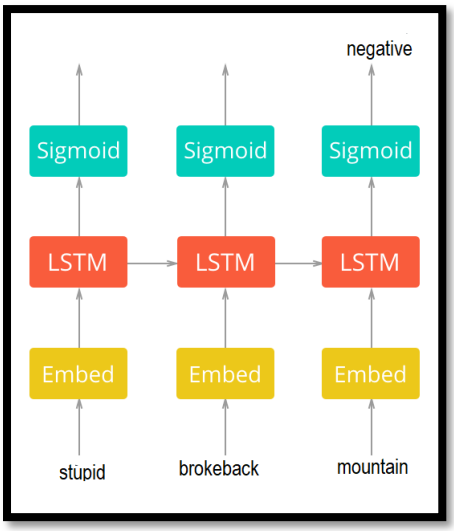


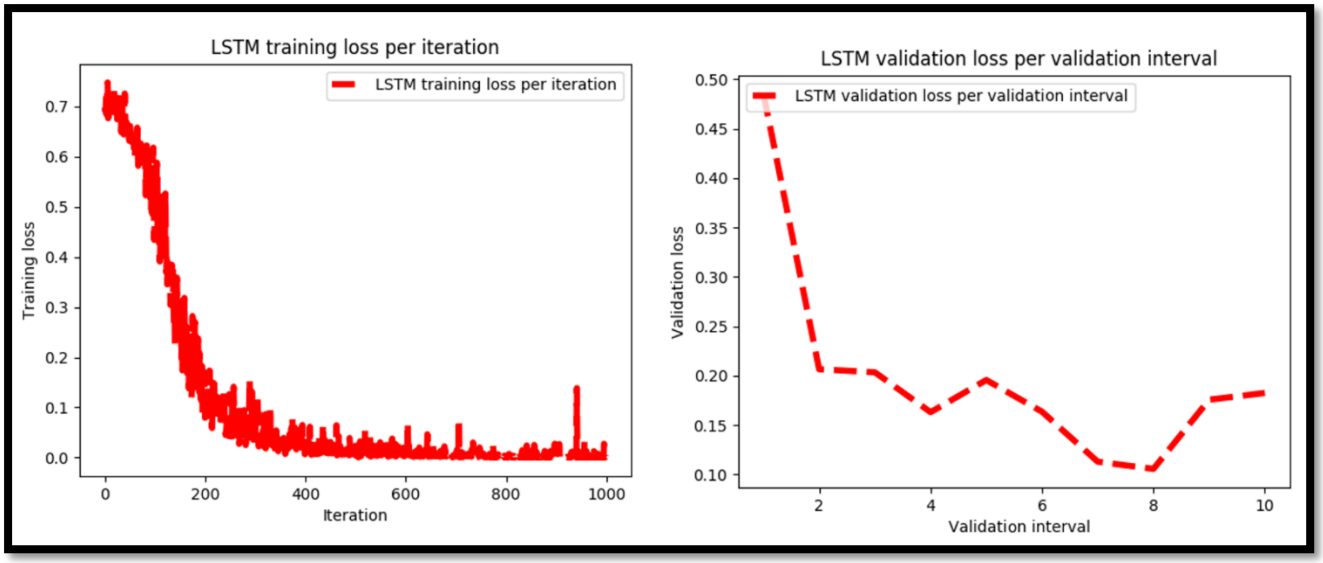

Normalized time series
Normalized value
ID
3
2
1
0
−1
0
20
40
60
80
100
120
140

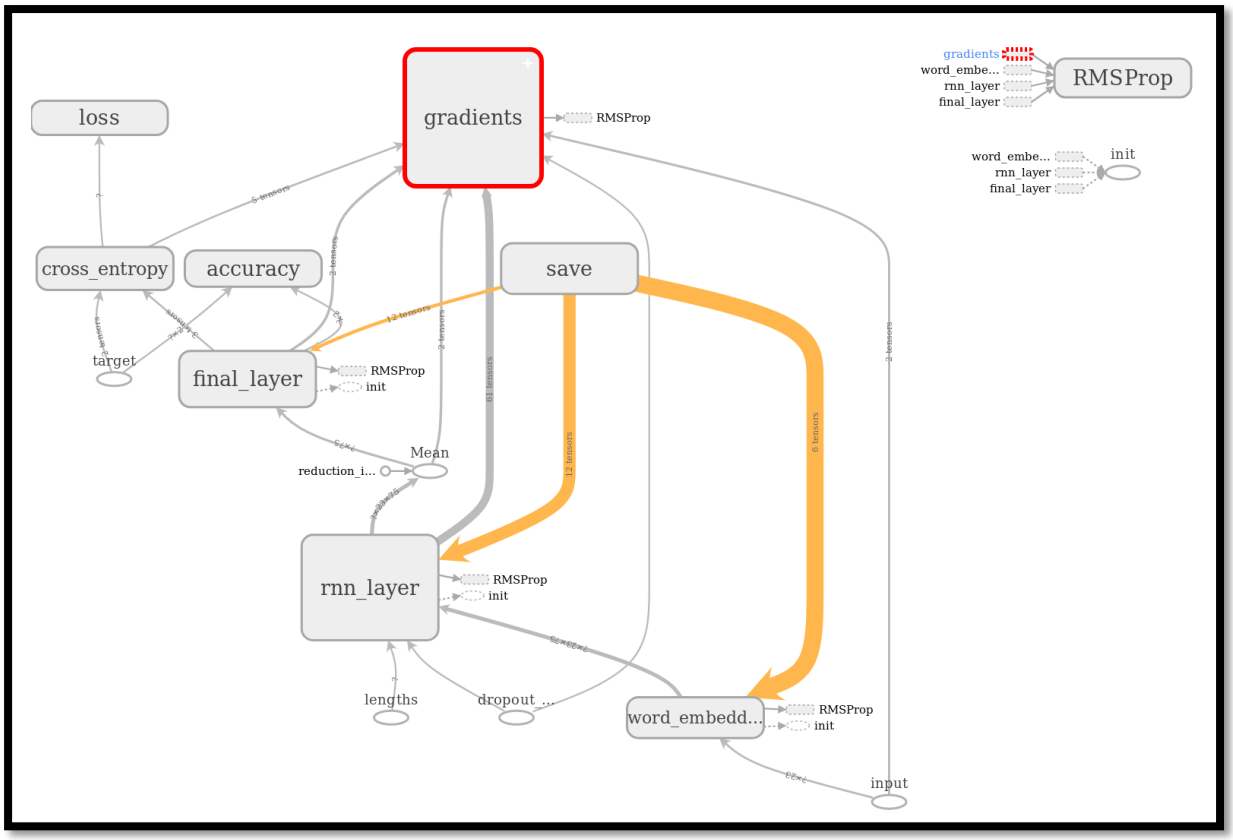

training data
predicted
test data
3
2
1
0
−1
0
20
40
60
80
100
120
140

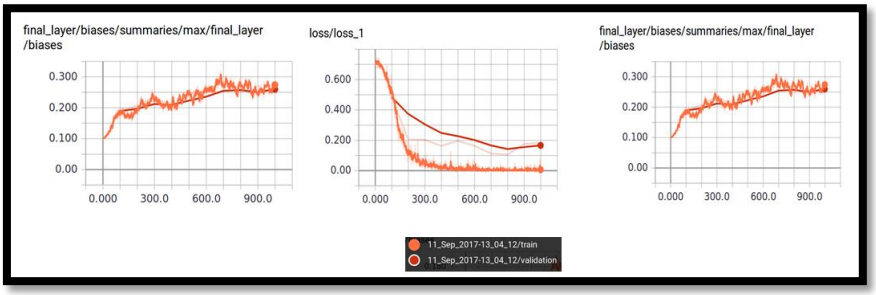

training data
predicted
test data
3
2
1
0
−1
0
200
400
600
800
1000

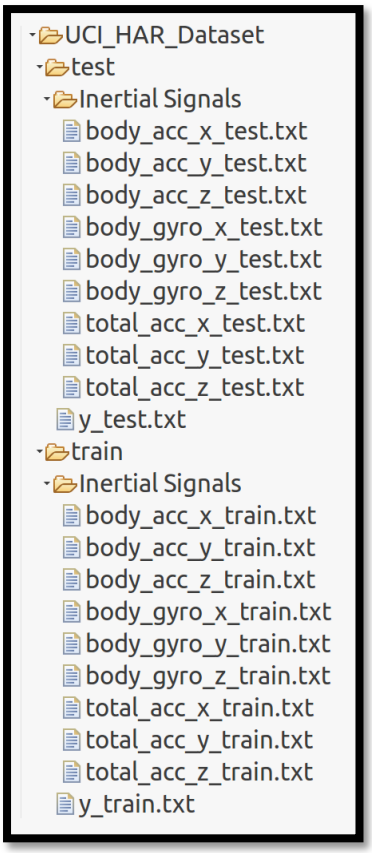

LSTM training loss per iteration
LSTM training loss per iteration
Training loss
Iteration
LSTM test loss per iteration
LSTM test loss per iteration
Test loss
Iteration

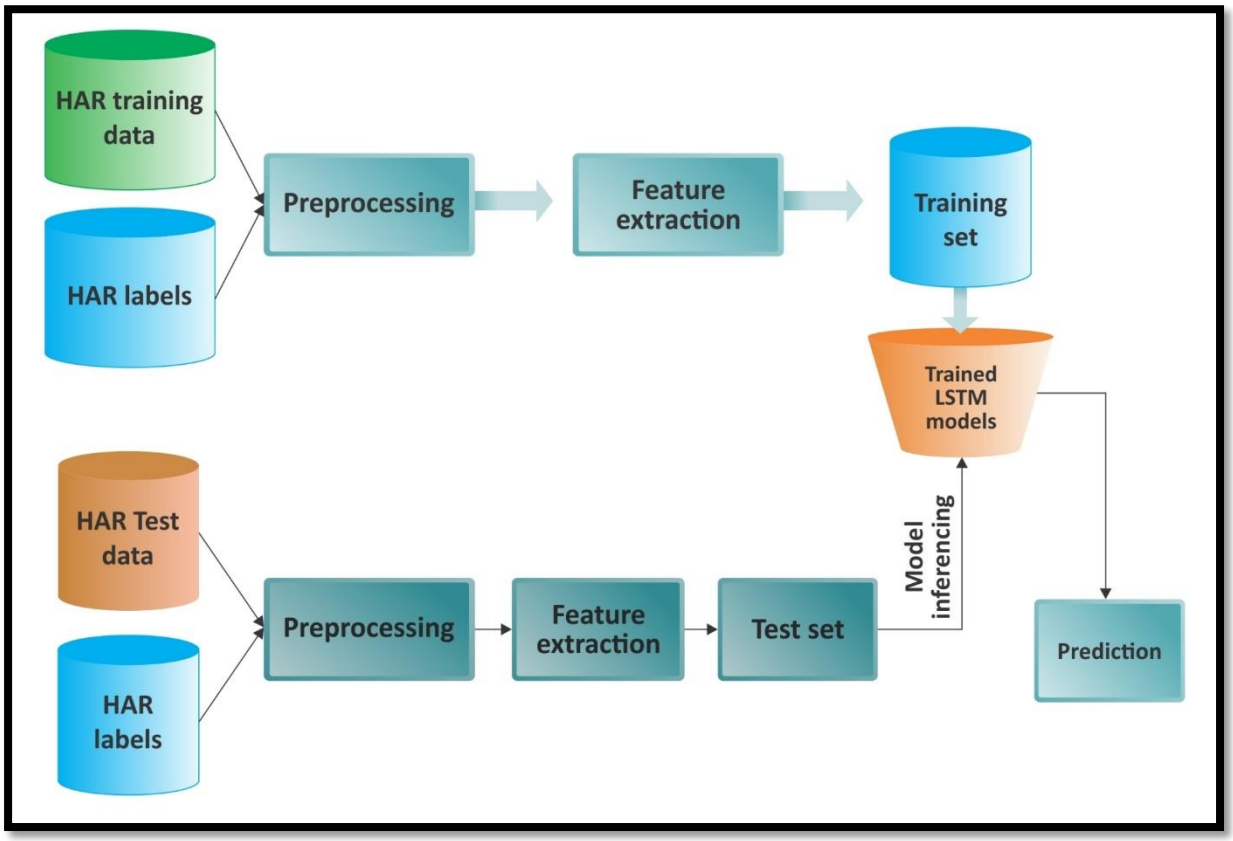

negative
Sigmoid
Sigmoid
Sigmoid
LSTM
LSTM
LSTM
Embed
Embed
Embed
stupid
brokeback
mountain

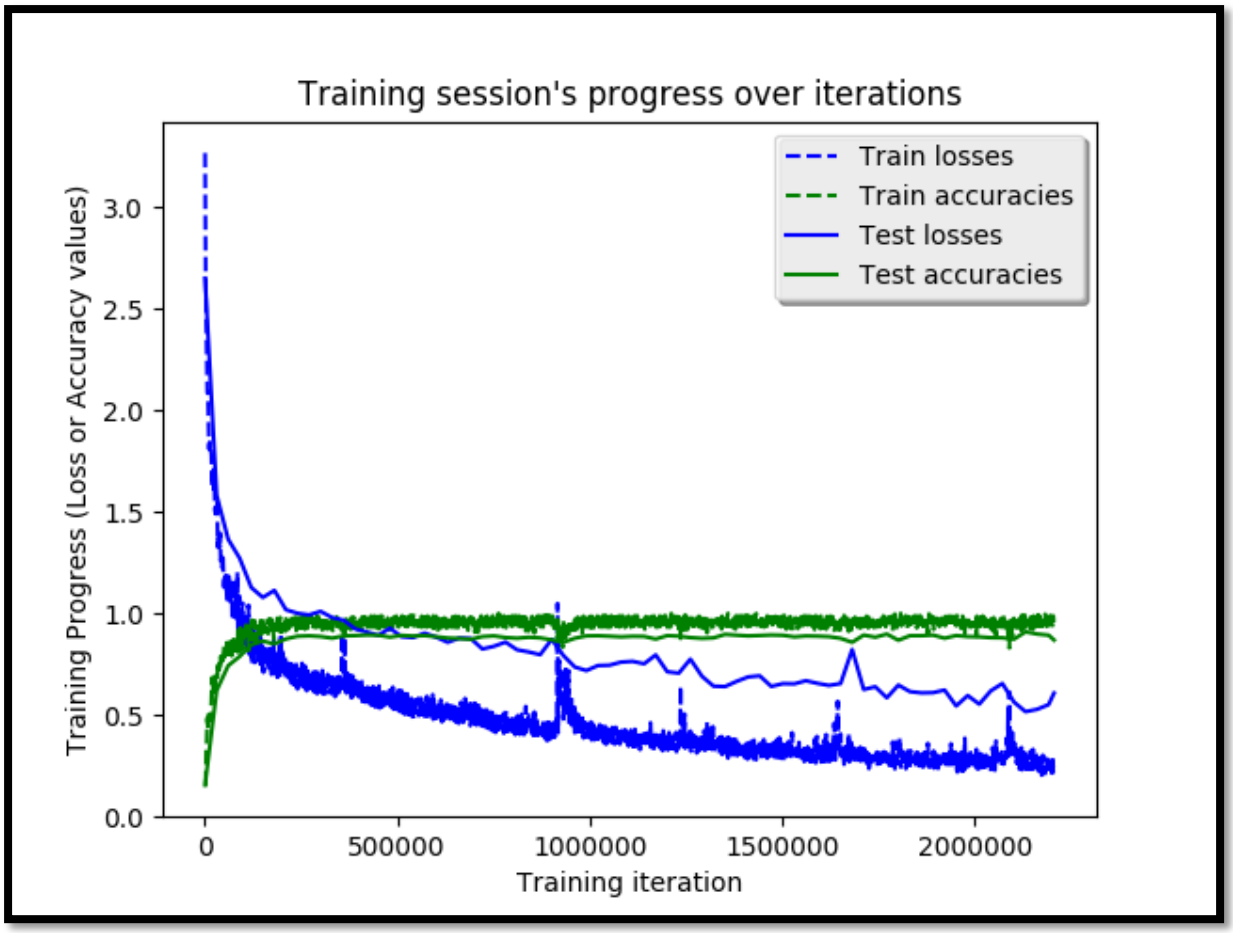

LSTM training loss per iteration
LSTM training loss per iteration
Training loss
Iteration
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000
LSTM validation loss per validation interval
LSTM validation loss per validation interval
Validation loss
Validation interval
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
2
4
6
8
10

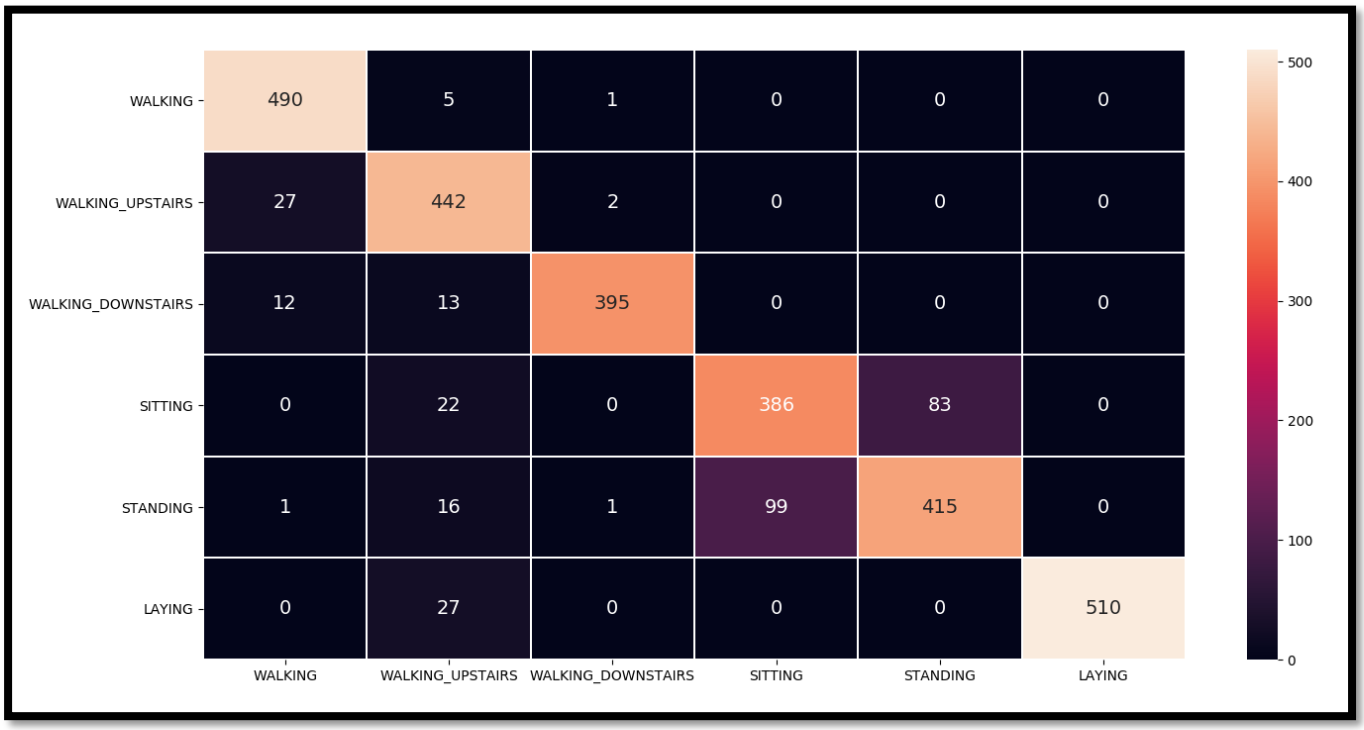

gradients
RMSProp
loss
cross_entropy
accuracy
save
target
final_layer
RMSProp
init
Mean
reduction_i...
rnn_layer
RMSProp
init
lengths
dropout_...
word_embedd...
RMSProp
init
input
gradients
word_embe...
rnn_layer
final_layer
RMSProp
word_embe...
rnn_layer
final_layer
init


final_layer/biases/summaries/max/final_layer/biases
0.300
0.200
0.100
0.00
0.000
300.0
600.0
900.0
loss/loss_1
0.600
0.400
0.200
0.00
0.000
300.0
600.0
900.0
final_layer/biases/summaries/max/final_layer/biases
0.300
0.200
0.100
0.00
0.000
300.0
600.0
900.0
11_Sep_2017-13_04_12/train
11_Sep_2017-13_04_12/validation

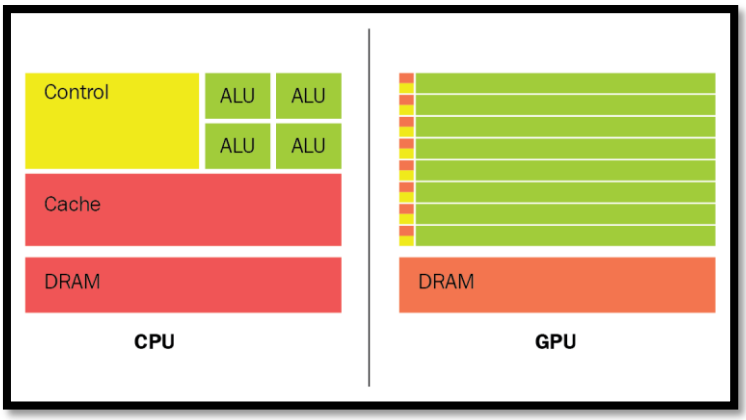

UCI_HAR_Dataset
test
Inertial Signals
body_acc_x_test.txt
body_acc_y_test.txt
body_acc_z_test.txt
body_gyro_x_test.txt
body_gyro_y_test.txt
body_gyro_z_test.txt
total_acc_x_test.txt
total_acc_y_test.txt
total_acc_z_test.txt
y_test.txt
train
Inertial Signals
body_acc_x_train.txt
body_acc_y_train.txt
body_acc_z_train.txt
body_gyro_x_train.txt
body_gyro_y_train.txt
body_gyro_z_train.txt
total_acc_x_train.txt
total_acc_y_train.txt
total_acc_z_train.txt
y_train.txt

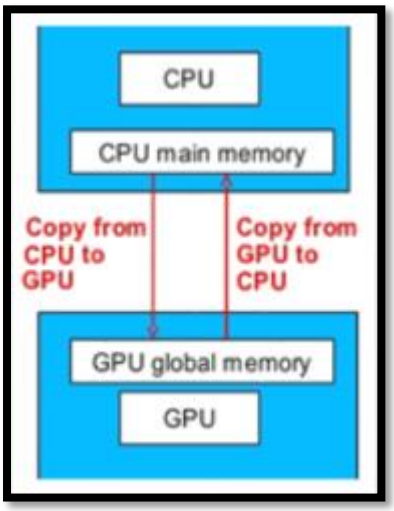

HAR training data
HAR labels
Preprocessing
Feature extraction
Training set
Trained LSTM models
HAR Test data
HAR labels
Preprocessing
Feature extraction
Test set
Model inferencing
Prediction

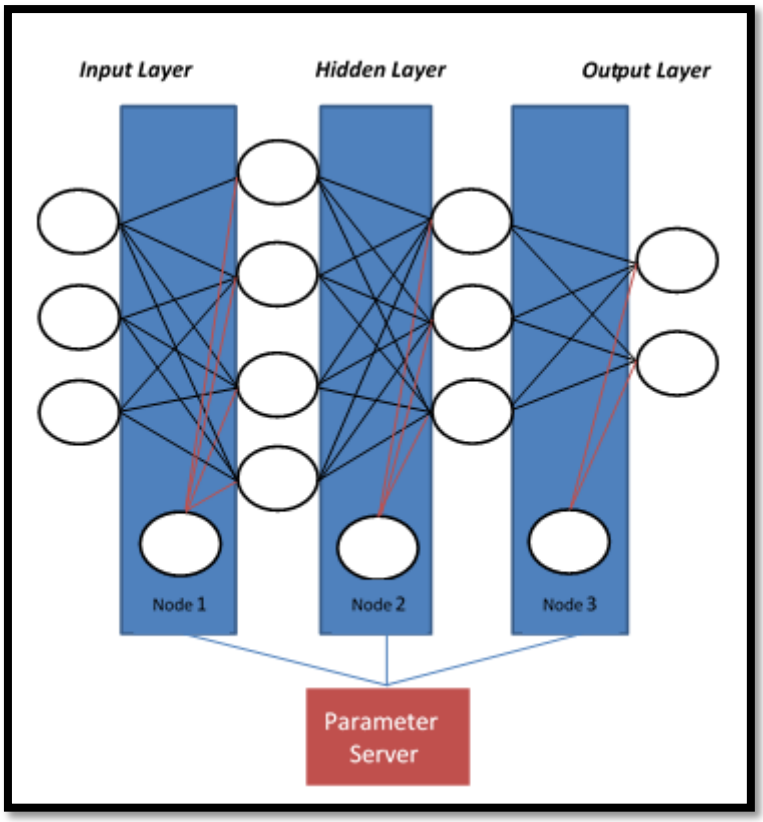

Training session's progress over iterations
Train losses
Train accuracies
Test losses
Test accuracies
Training Progress (Loss or Accuracy values)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
500000
1000000
1500000
2000000
Training iteration

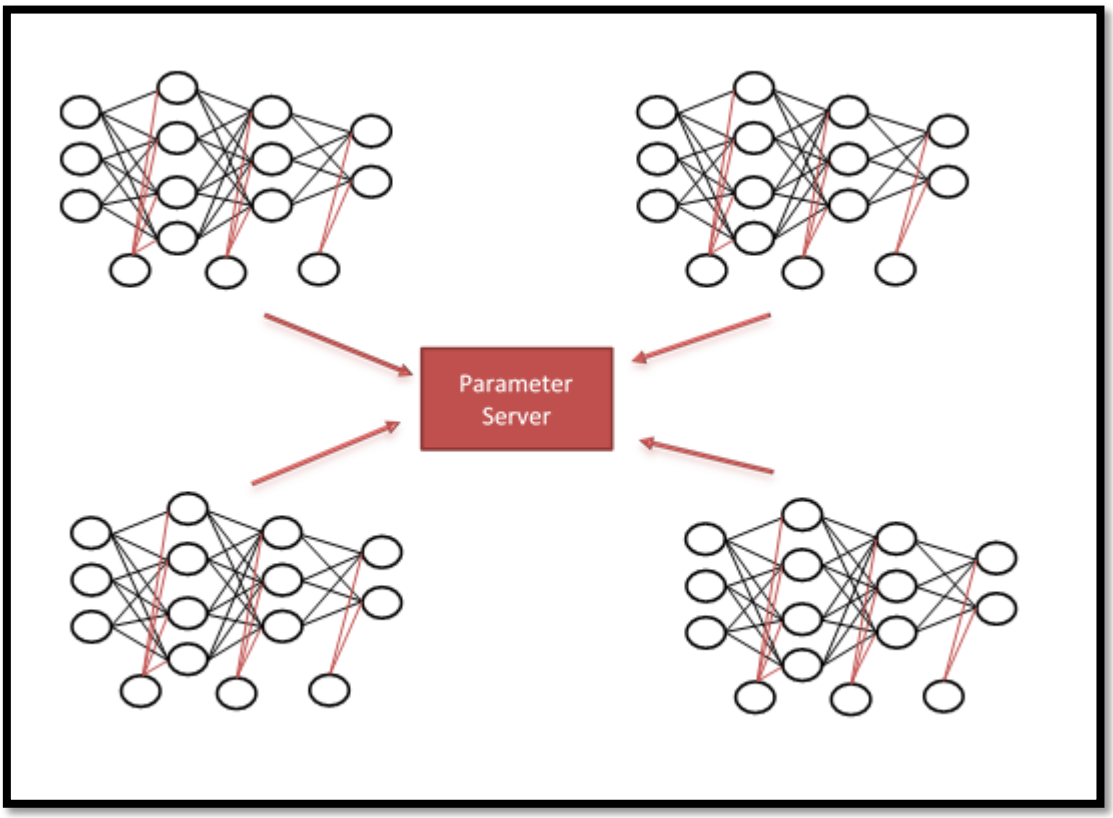

WALKING
WALKING_UPSTAIRS
WALKING_DOWNSTAIRS
SITTING
STANDING
LAYING
490 5 1 0 0 0
27 442 2 0 0 0
12 13 395 0 0 0
0 22 0 386 83 0
1 16 1 99 415 0
0 27 0 0 0 510
WALKING
WALKING_UPSTAIRS
WALKING_DOWNSTAIRS
SITTING
STANDING
LAYING
500
400
300
200
100
0

Chapter 7: Heterogeneous and Distributed Computing





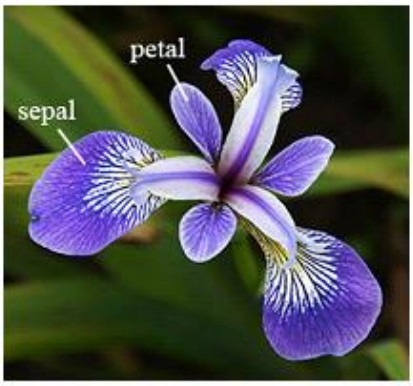

Input Layer
Hidden Layer
Output Layer
Node 1
Node 2
Node 3
Parameter Server


Parameter Server

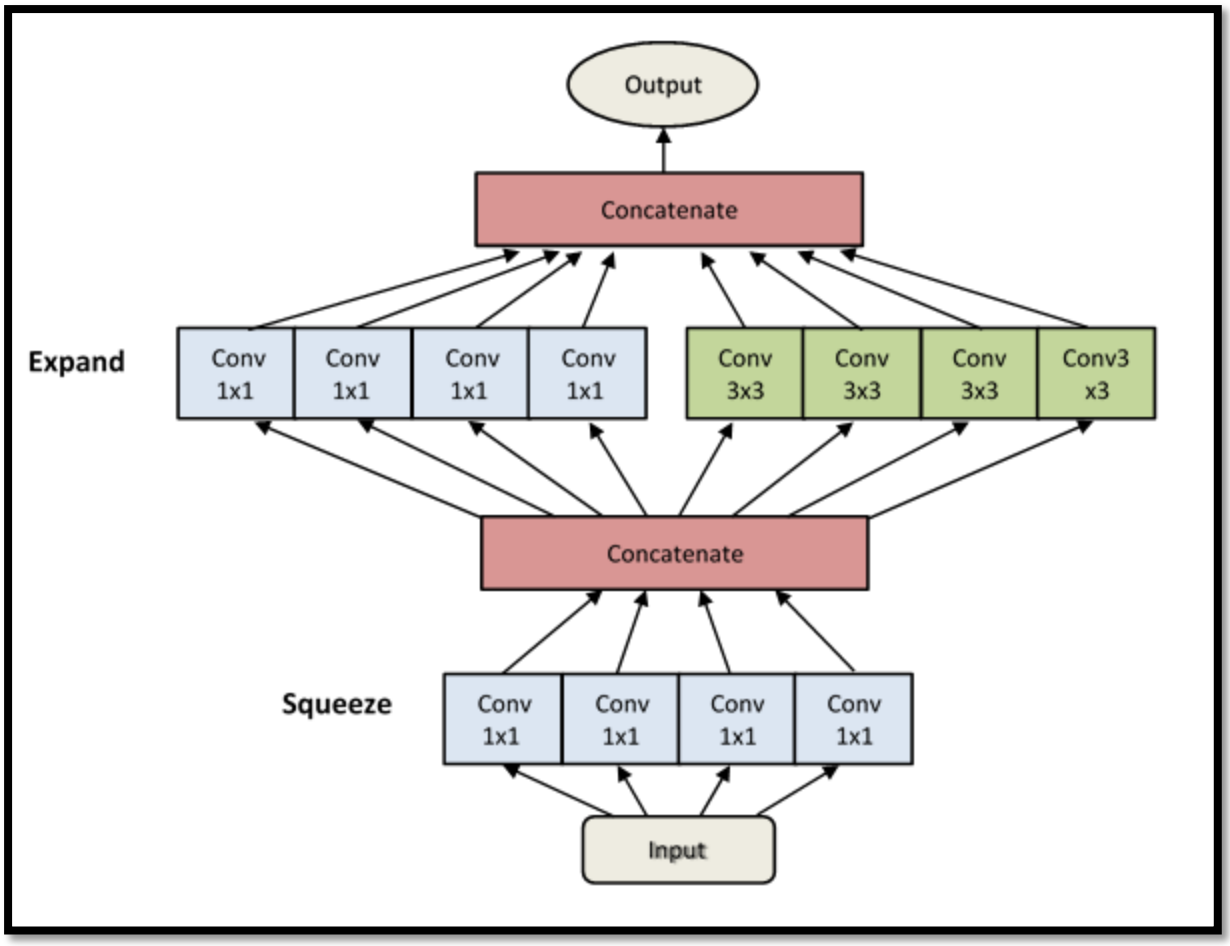
Chapter 8: Advanced TensorFlow Programming

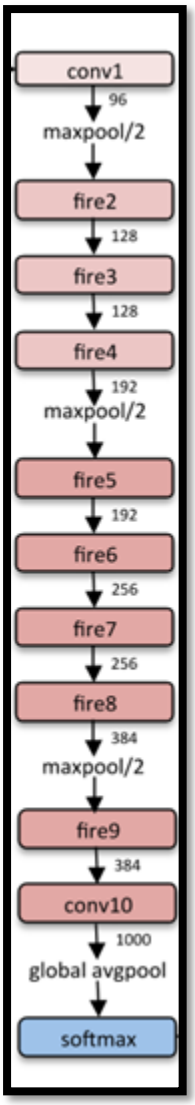
Iris Setosa

Iris Virginica

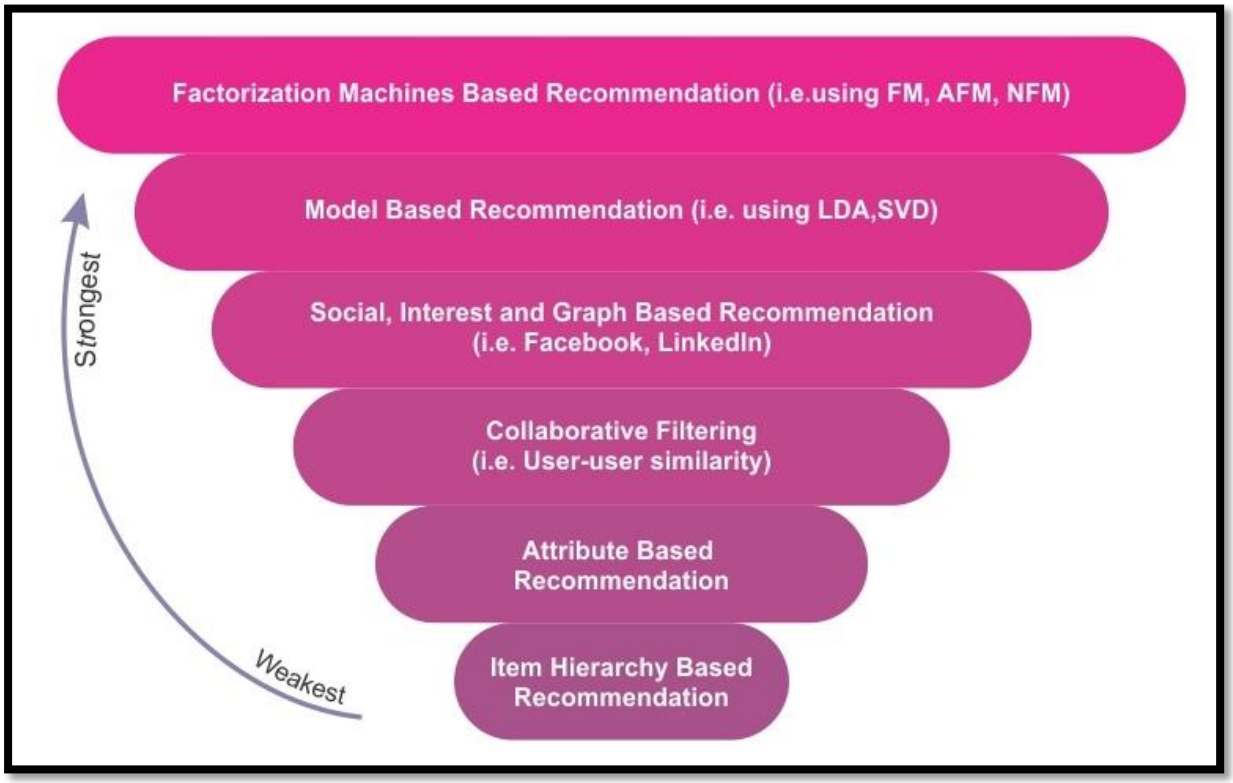


Iris Versicolor

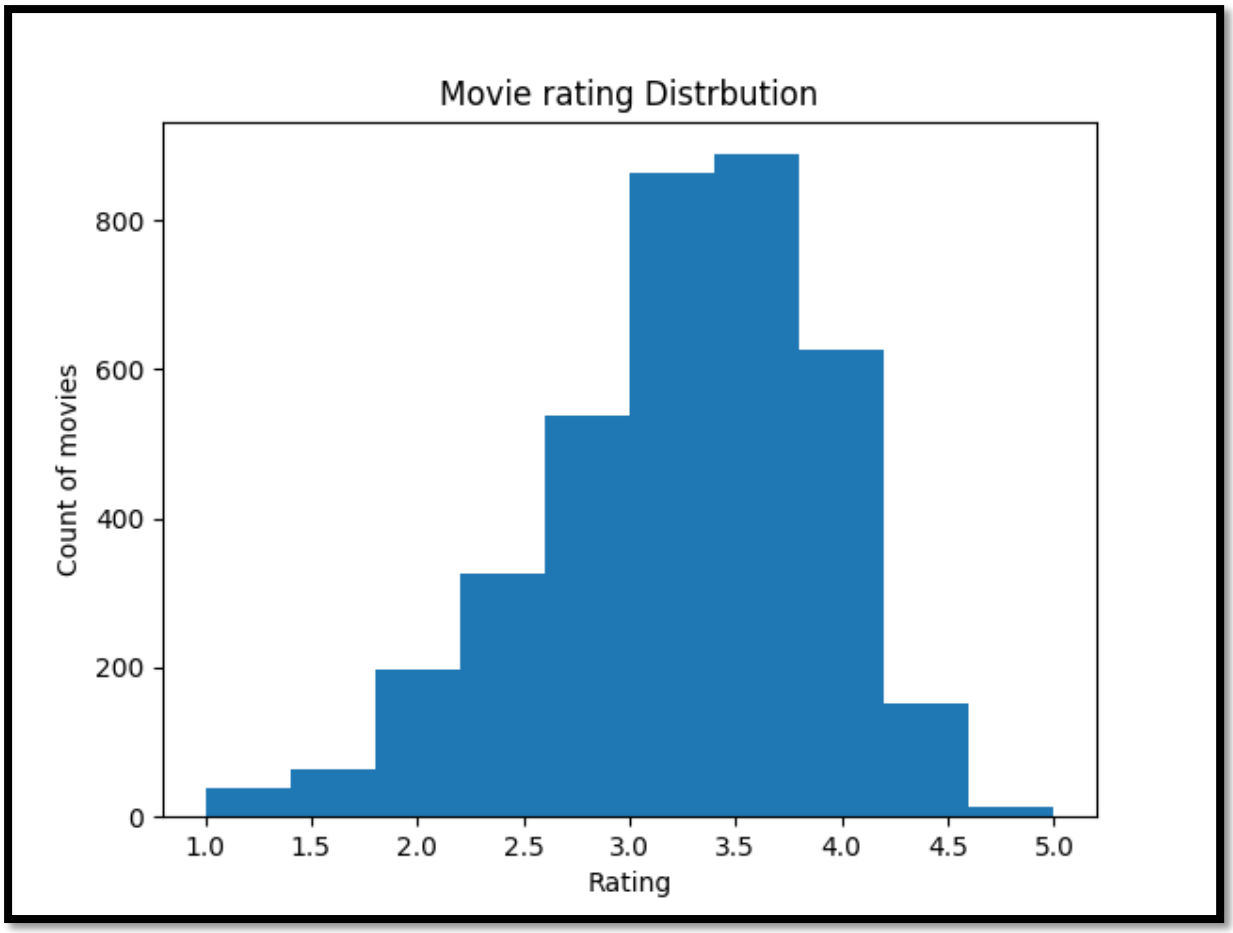

Output
Concatenate
Expand
Conv 1x1
Conv 1x1
Conv 1x1
Conv 1x1
Conv 3x3
Conv 3x3
Conv 3x3
Conv3 x3
Concatenate
Squeeze
Conv 1x1
Conv 1x1
Conv 1x1
Conv 1x1
Input

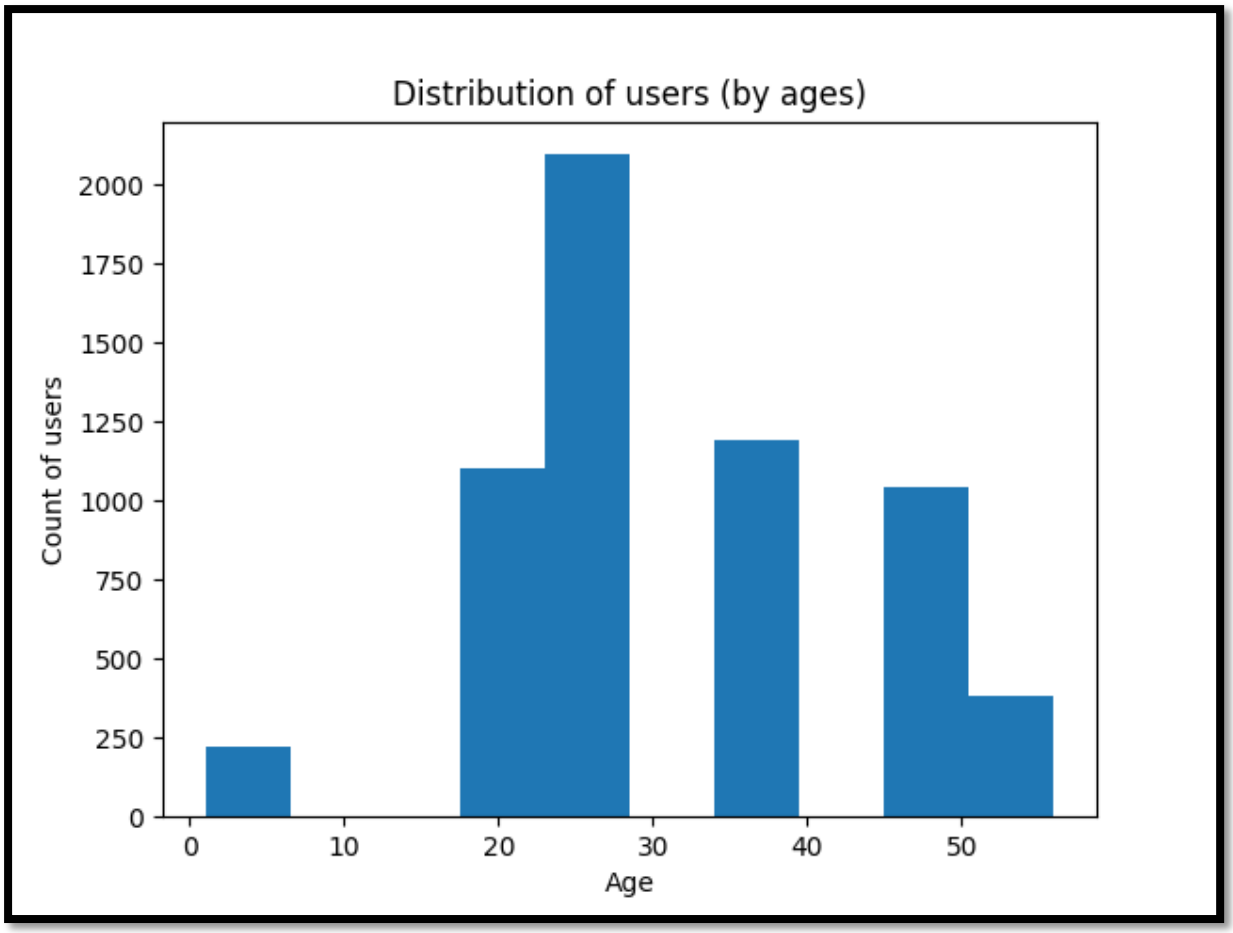

conv1
96
maxpool/2
fire2
128
fire3
128
fire4
192
maxpool/2
fire5
192
fire6
256
fire7
256
fire8
384
maxpool/2
fire9
384
conv10
1000
global avgpool
softmax

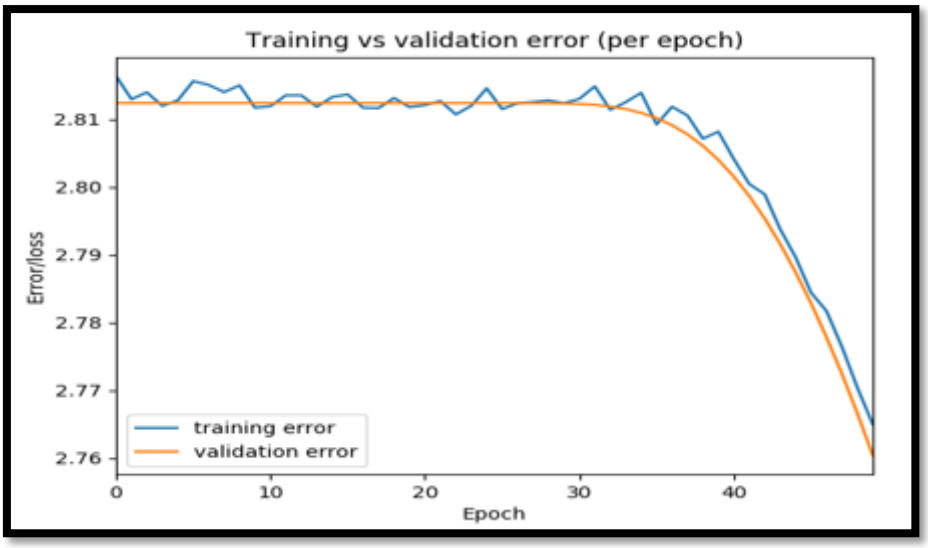
Chapter 9: Recommendation Systems using Factorization Machines

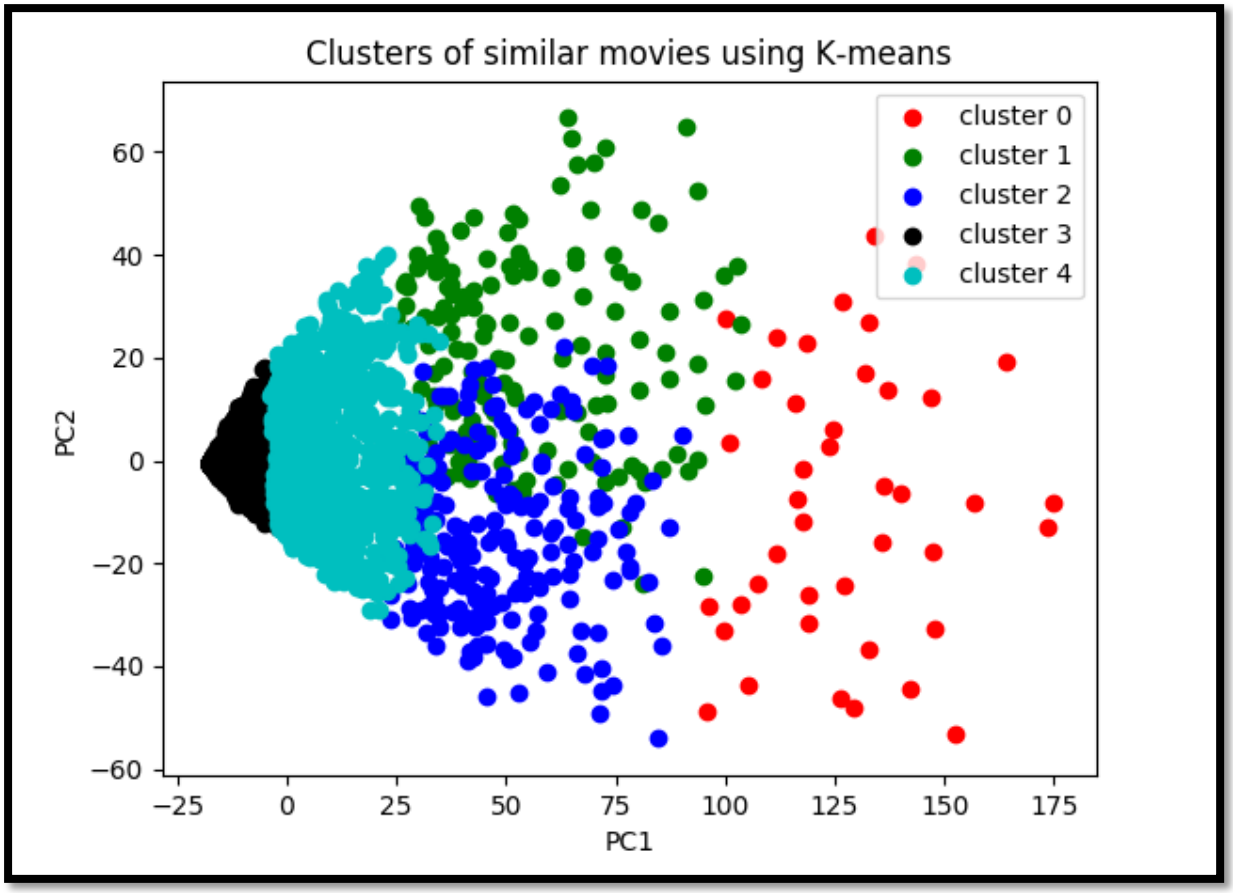


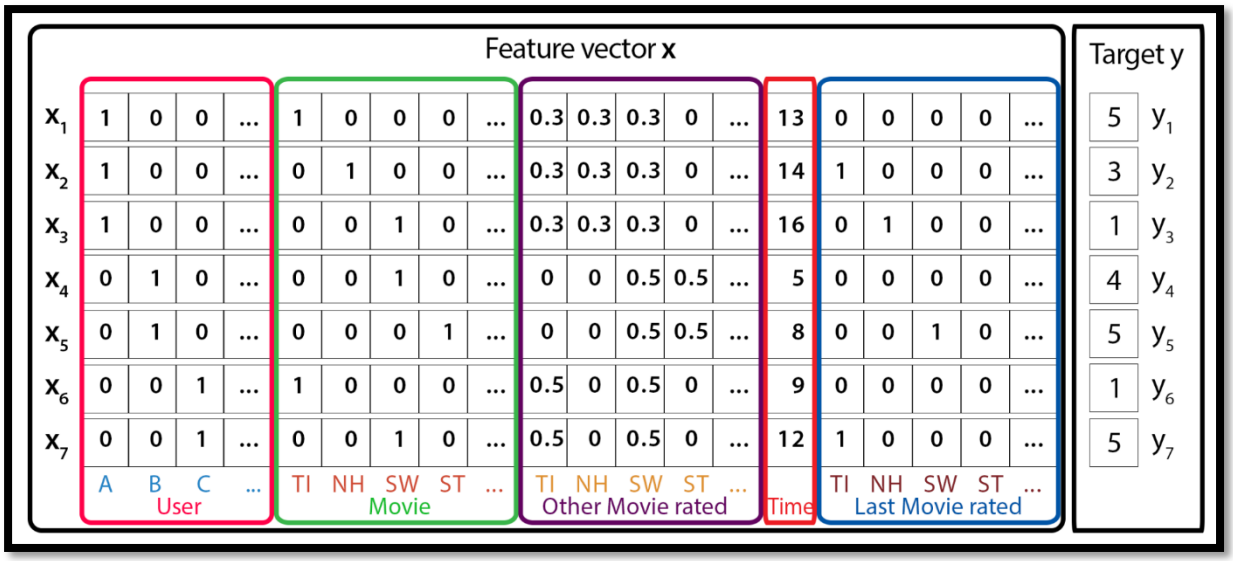
	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

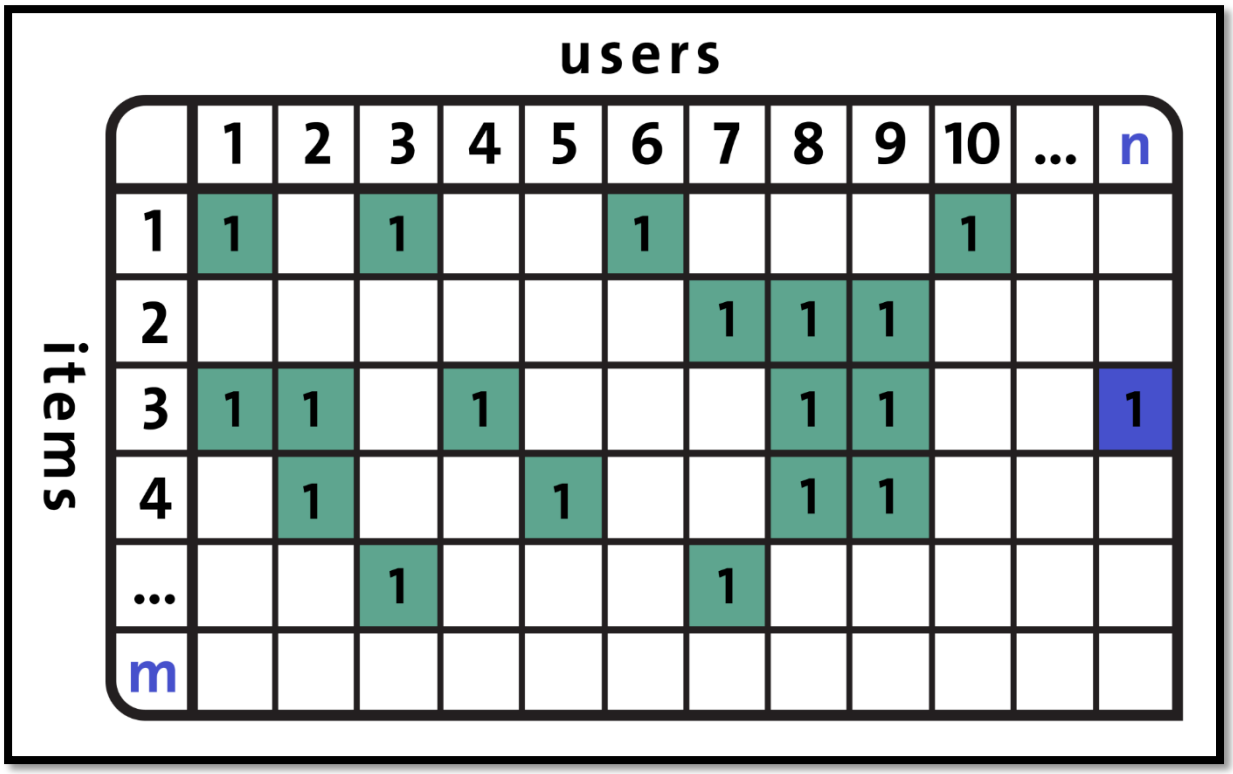

Movie rating Distrbution
Count of movies
800
600
400
200
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Rating


Distribution of users (by ages)
Count of users
2000
1750
1500
1250
1000
750
500
250
0
0
10
20
30
40
50
Age

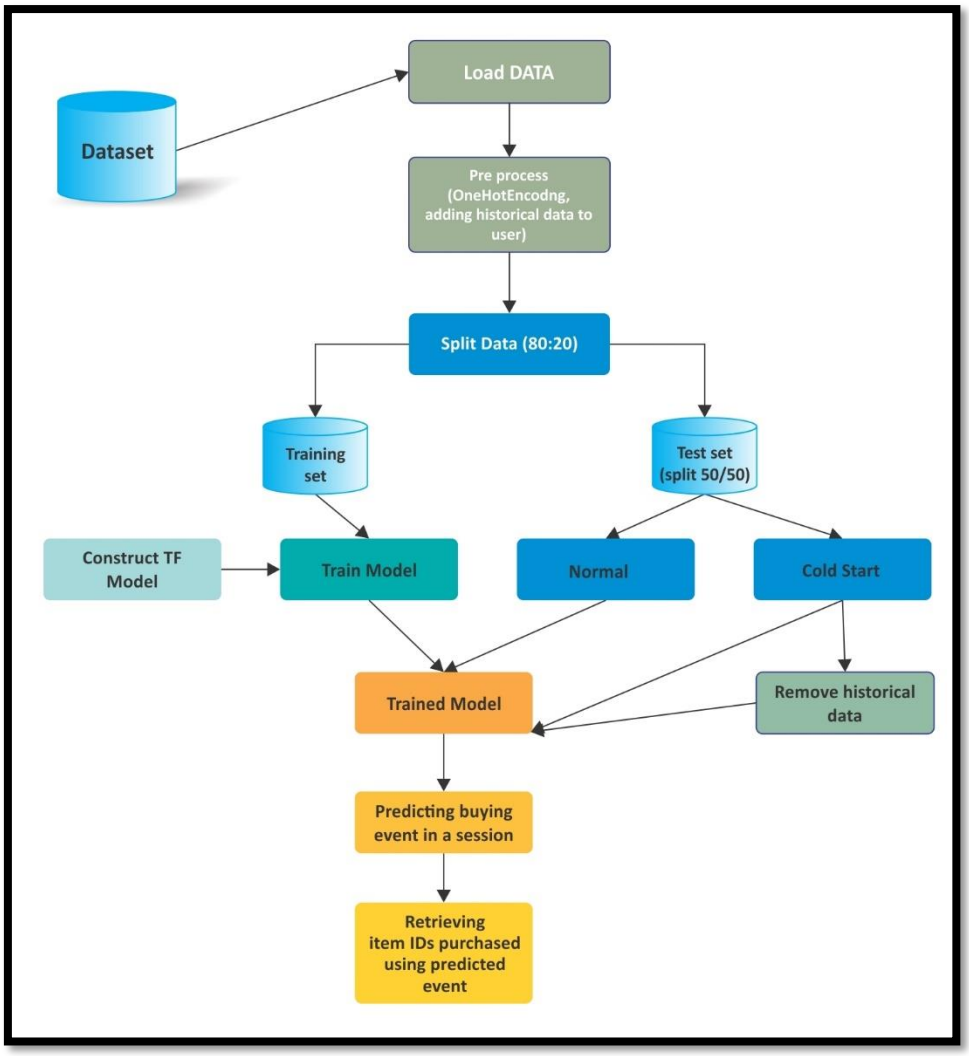

Training vs validation error (per epoch)
Error/loss
Epoch
training error
validation error
2.81
2.80
2.79
2.78
2.77
2.76
0
10
20
30
40


Clusters of similar movies using K-means
PC1
PC2
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
60
40
20
0
−20
−40
−60
−25
0
25
50
75
100
125
150
175





	User				Item				Categories				History				Quantity	
x_1	1	0	1	...	0	1	0	...	1	2	3	...	1	1	0	...	3	y_1
x_2	0	0	1	...	1	0	1	...	8	9	6	...	0	1	0	...	7	y_2
x_3	0	1	1	...	1	0	0	...	5	2	7		1	1	1	...	9	y_3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
x_n	1	0	1	...	1	1	1	...	2	4	6	...	0	1	1	...	8	y_n



	Session ID	Timestamp	Item ID	Category	Quantity
0	420374.0	2014-04-06T18:44:58.314Z	214537888.0	12462	1
1	420374.0	2014-04-06T18:44:58.325Z	214537856.0	10471	1
2	281626.0	2014-04-06T09:40:13.032Z	214535648.0	1883	1
3	420368.0	2014-04-04T06:13:28.848Z	214530576.0	6073	1
4	420368.0	2014-04-04T06:13:28.858Z	214835024.0	2617	1

(1150753, 5)

	Session ID	Timestamp	Item ID	Category
0	1	2014-04-07T10:51:09.277Z	214536502	0
1	1	2014-04-07T10:54:09.868Z	214536500	0
2	1	2014-04-07T10:54:46.998Z	214536506	0
3	1	2014-04-07T10:57:00.306Z	214577561	0
4	2	2014-04-07T13:56:37.614Z	214662742	0

(33003944, 4)

		Item ID	Category	Quantity
Session ID				
420374.0	214537888.0	12462		1
420374.0	214537856.0	10471		1
281626.0	214535648.0	1883		1
420368.0	214530576.0	6073		1
420368.0	214835024.0	2617		1

(1150753, 3)

	Item ID	Category
Session ID		
1	214536502	0
1	214536500	0
1	214536506	0
1	214577561	0
2	214662742	0

(33003944, 2)

Session ID	Item ID	Category	Quantity
420471.0	214717888.0	2092	1
420471.0	214821024.0	1570	1
420471.0	214829280.0	837	1
420471.0	214819552.0	418	1
420471.0	214746384.0	784	1

(106956, 3)

Session ID	Item ID	Category
932	214826906	0
932	214826906	0
932	214826906	0
932	214826955	0
932	214826955	0

(209024, 2)

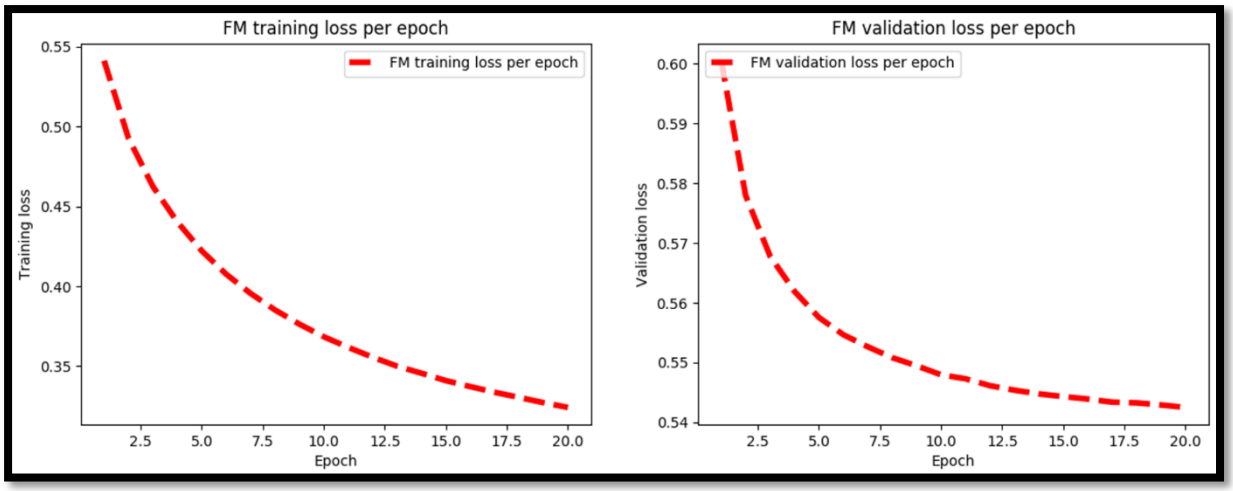
Session ID	Item ID	Category	Quantity	_Session ID
420471.0	214717888.0	2092	1	420471.0
420471.0	214821024.0	1570	1	420471.0
420471.0	214829280.0	837	1	420471.0
420471.0	214819552.0	418	1	420471.0
420471.0	214746384.0	784	1	420471.0

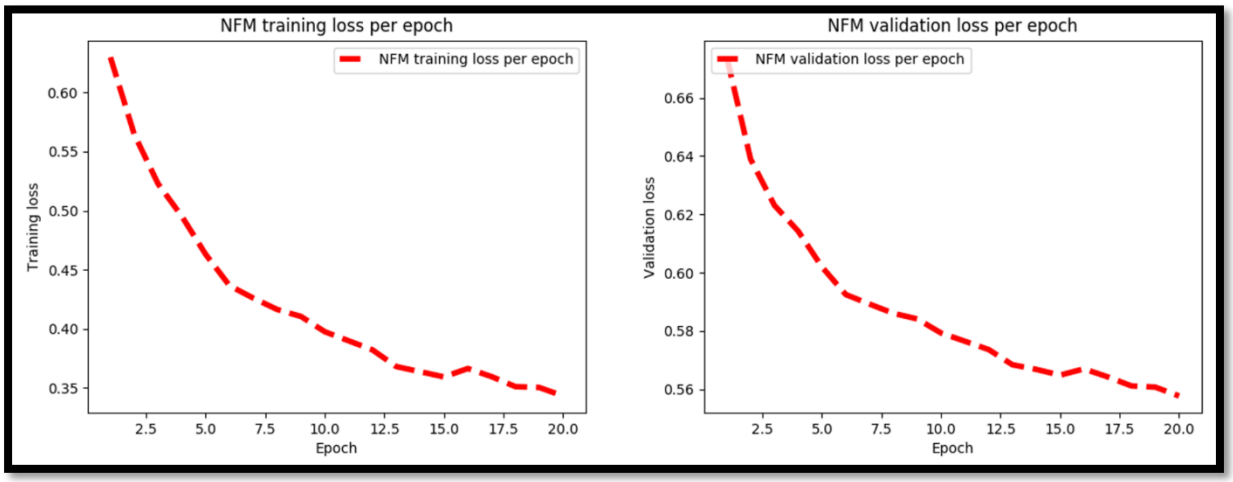
(106956, 4)

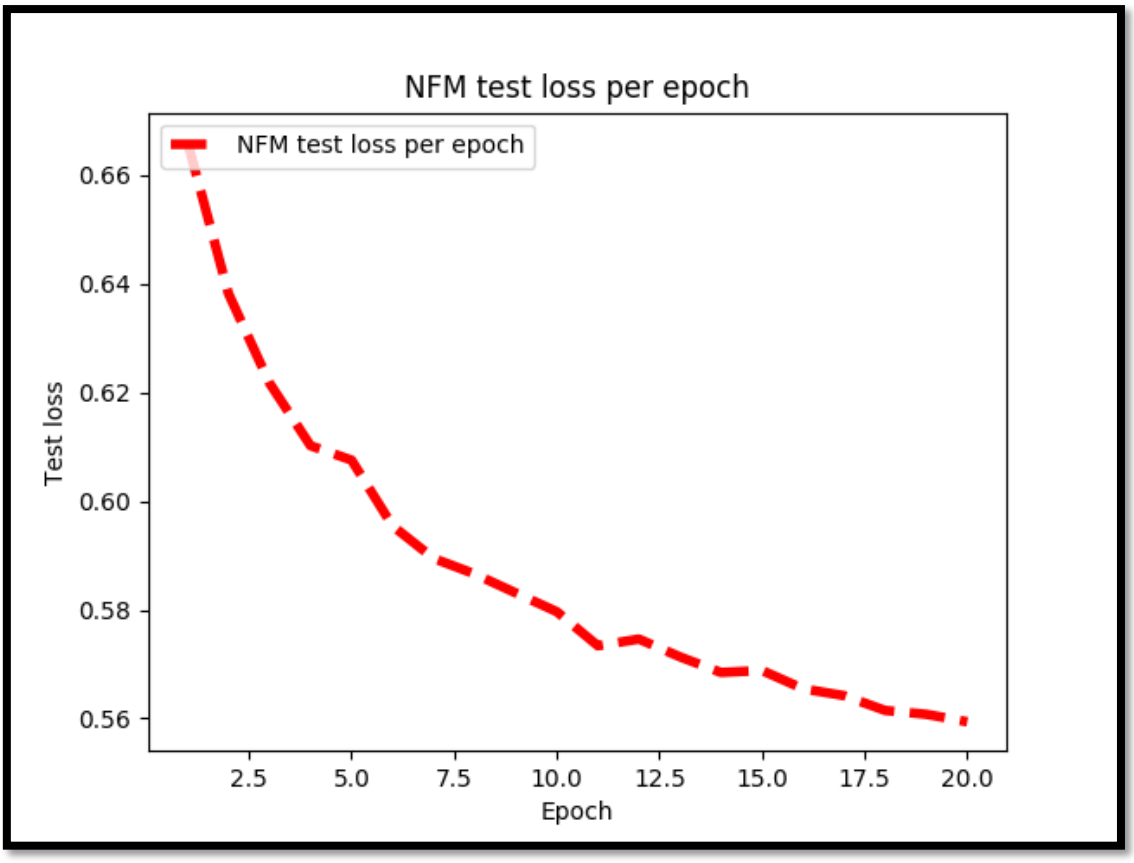
```
[ 2  2  1  2  1  1  2  2  2  1  1  1  1  1  1  1  1  1  1  2  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  2  2  1  2  2  2  1  2  1  1  1  1  1  1
  1  1  2  5  4  2  4  2  2  5  1  2  2  1  2  1  2  1  1  1  1  1  1  1  1
  2  1  1  2  1  1  1  1  1  2  1 10  6  1 10  1  6  1 10  1  6  1  0  0  0]
[2 2 1 ..., 2 1 1] 106956
2
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0]
[1 1 1 ..., 1 1 1] (106956,)
```

Item ID	
Session ID	
2614096	214829888.0
6388687	214845456.0
517818	214837488.0
6498748	214691520.0
2541201	214845104.0
(10696, 1)	

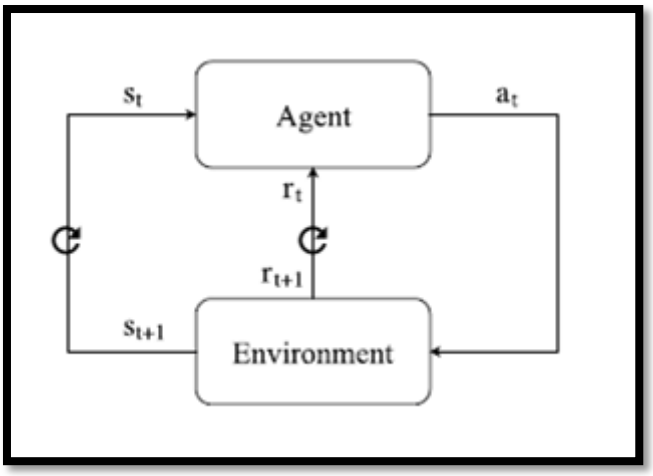
Item ID	
Session ID	
17929	214827008.0
161673	214826928.0
10914216	214854848.0
9075227	214678368.0
8356289	214716672.0

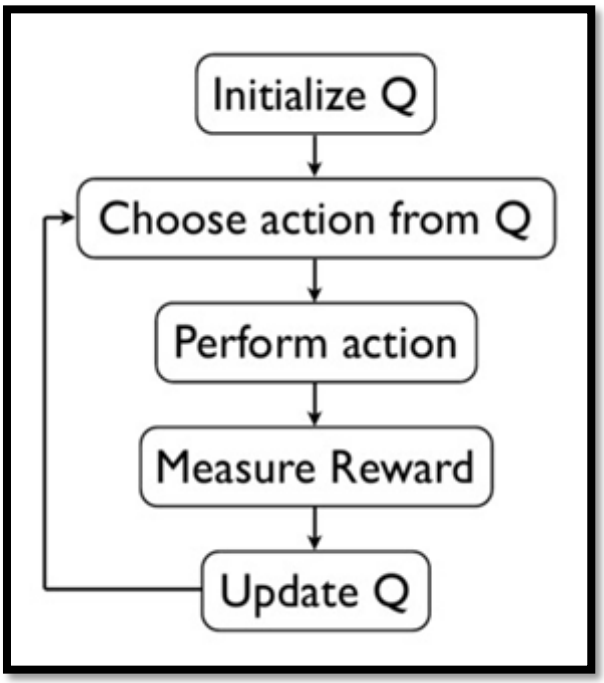

FM training loss per epoch
FM training loss per epoch
Training loss
Epoch
FM validation loss per epoch
FM validation loss per epoch
Validation loss
Epoch


NFM training loss per epoch
NFM training loss per epoch
Training loss
Epoch
NFM validation loss per epoch
NFM validation loss per epoch
Validation loss
Epoch

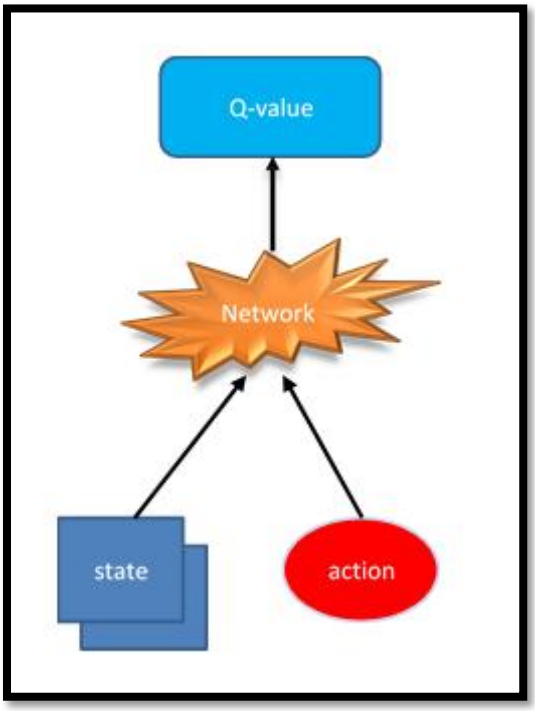

NFM test loss per epoch
NFM test loss per epoch
0.66
0.64
0.62
0.60
0.58
0.56
Test loss
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epoch

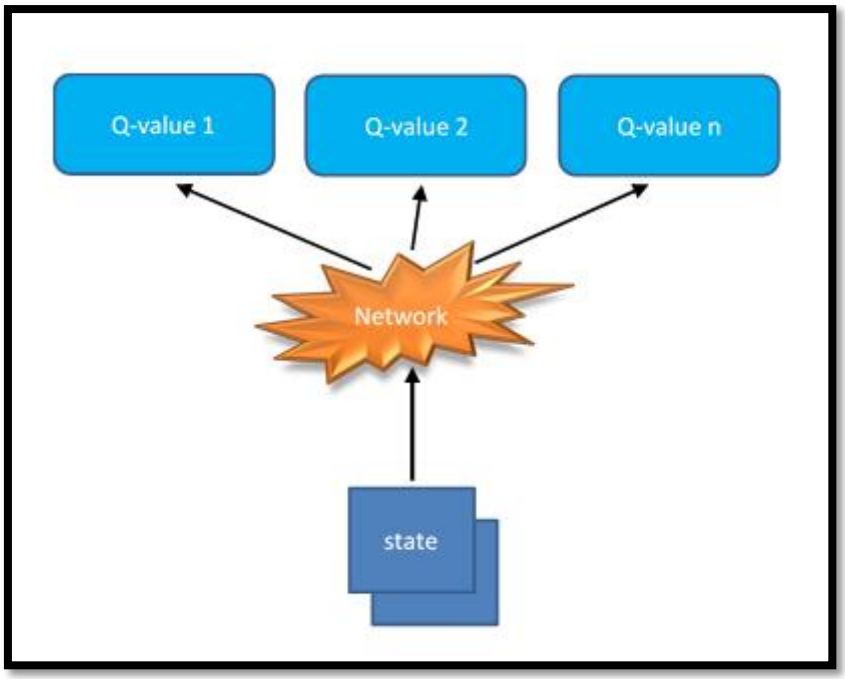
Chapter 10: Reinforcement Learning

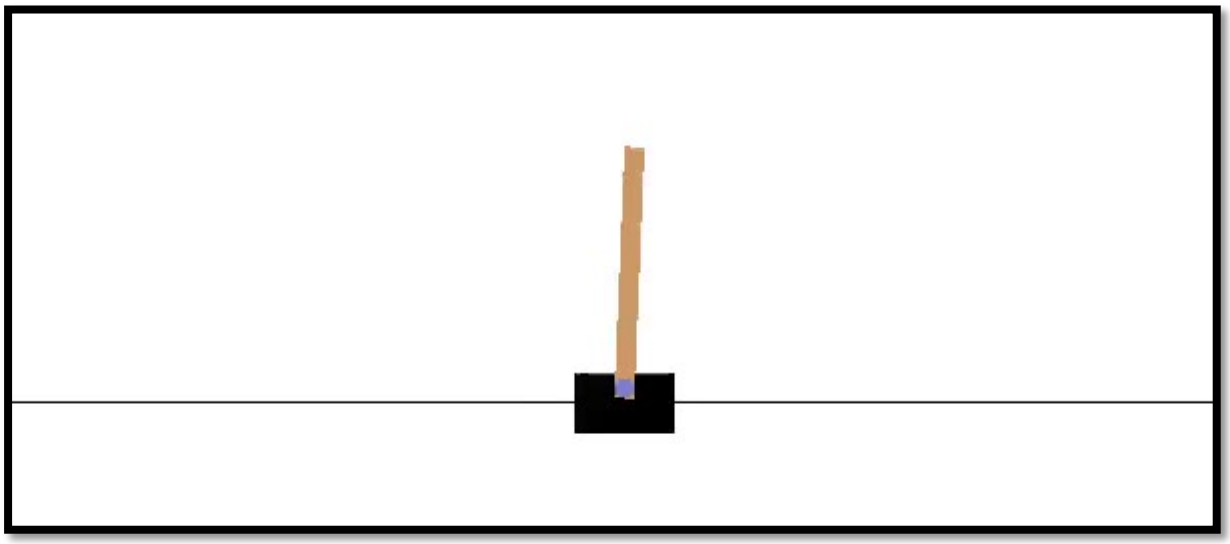


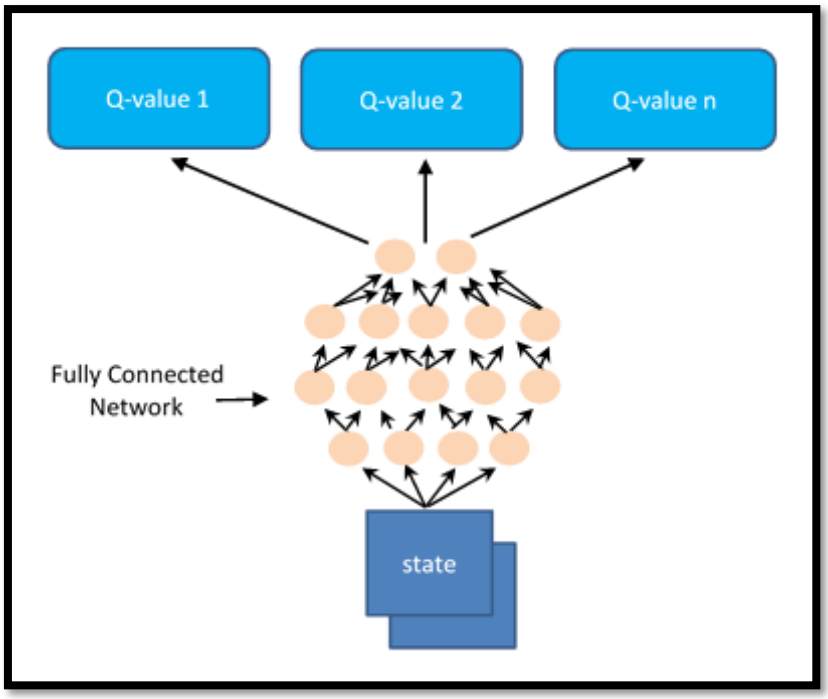


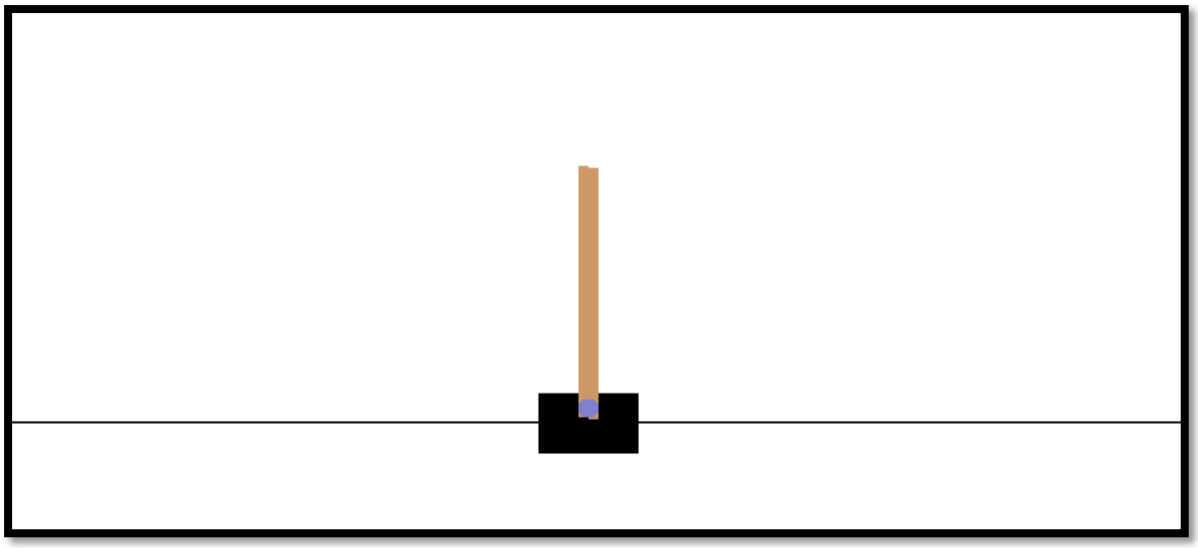

SFFF
FHFH
FFFH
HFFG
(Down)

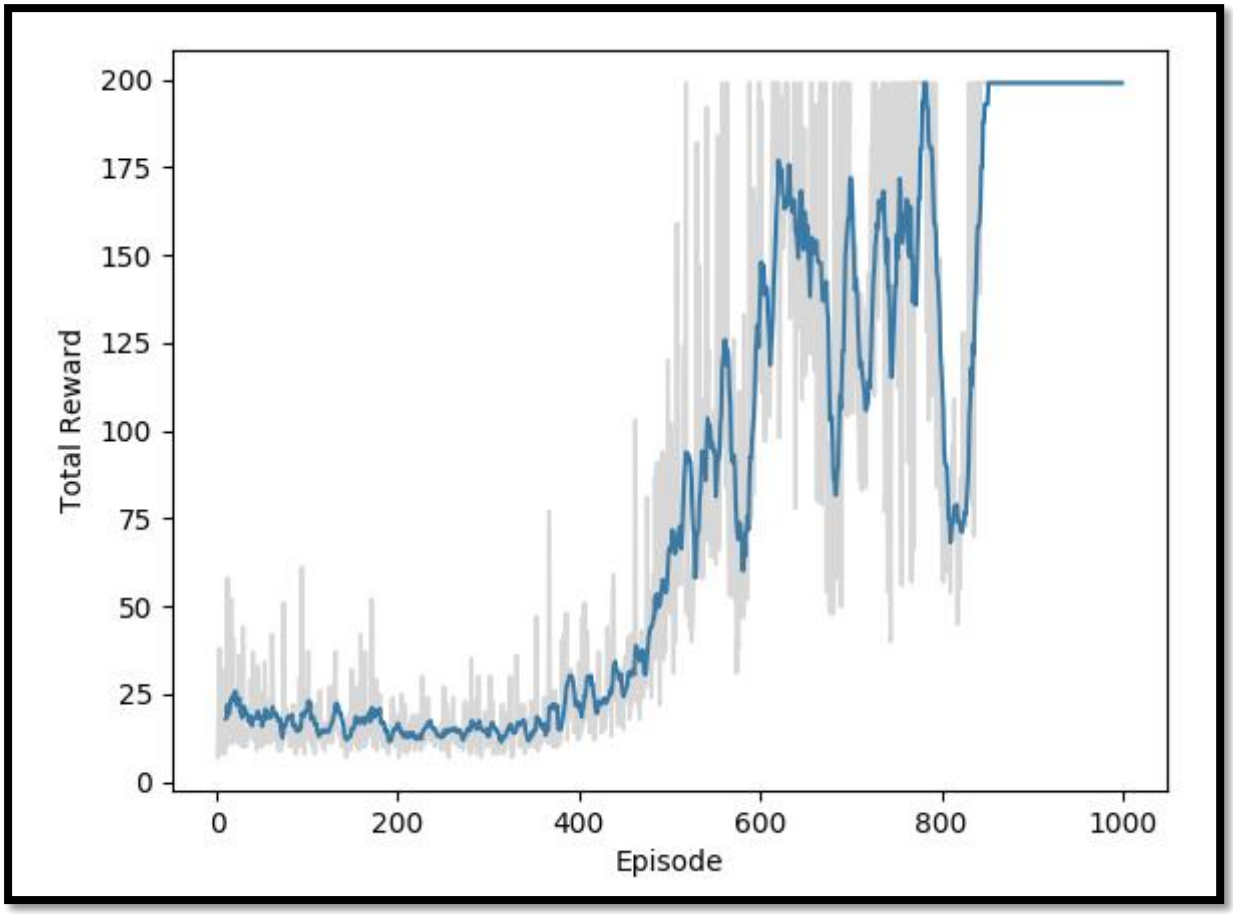

Q-value
Network
state
action


Q-value 1
Q-value 2
Q-value n
Network
state




Q-value 1
Q-value 2
Q-value n
Fully Connected
Network
state




Total Reward
200
175
150
125
100
75
50
25
0
0
200
400
600
800
1000
Episode