Table of Contents

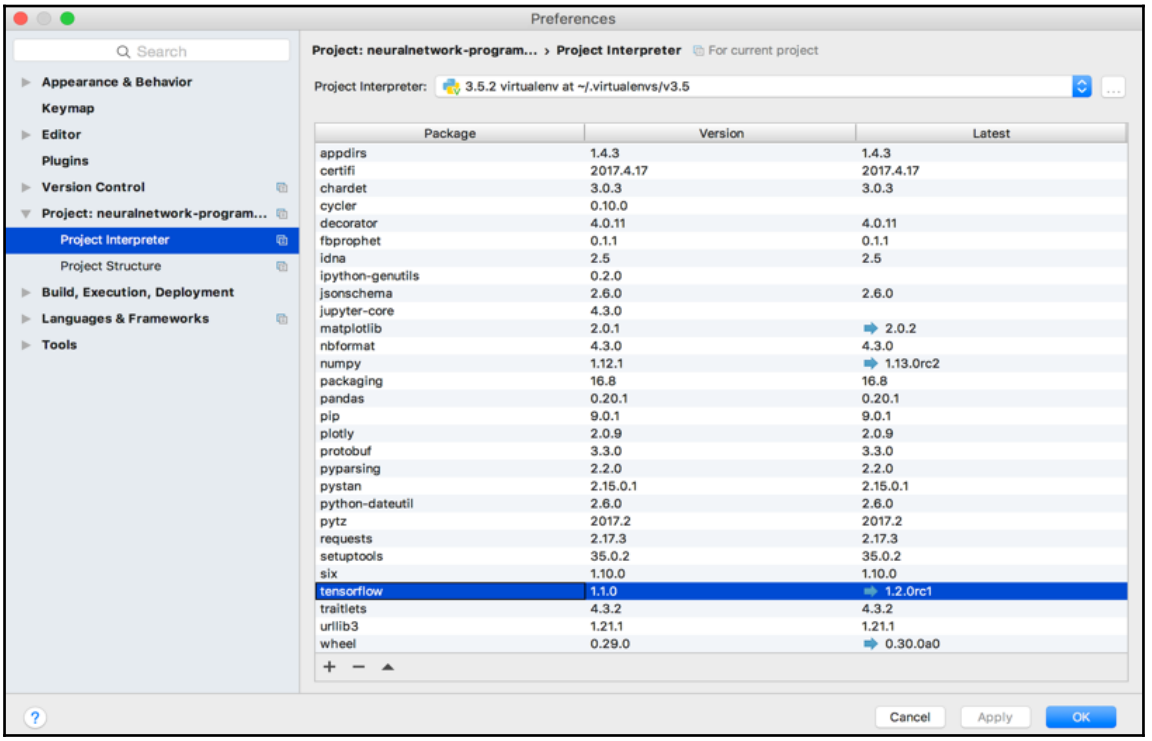
Chapter 1: Graphic bundle 1

- **Chapter 1: Maths for Neural Networks** 1
- **Chapter 2: Deep Feedforward Networks** 6
- **Chapter 3: Optimization for Neural Networks** 13
- **Chapter 4: Convolutional Neural Networks** 17
- **Chapter 5: Recurrent Neural Networks** 36
- **Chapter 6: Generative Models** 44
- **Chapter 7: Deep Belief Networking** 47
- **Chapter 8: Autoencoders** 49
- **Chapter 9: Research in Neural Networks** 58
- **Appendix : Getting started with TensorFlow** 62

Index 64

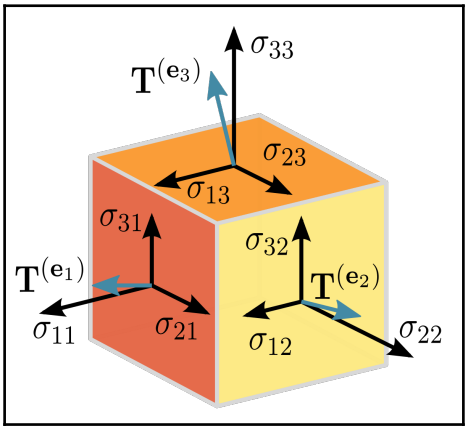
1
Graphic bundle

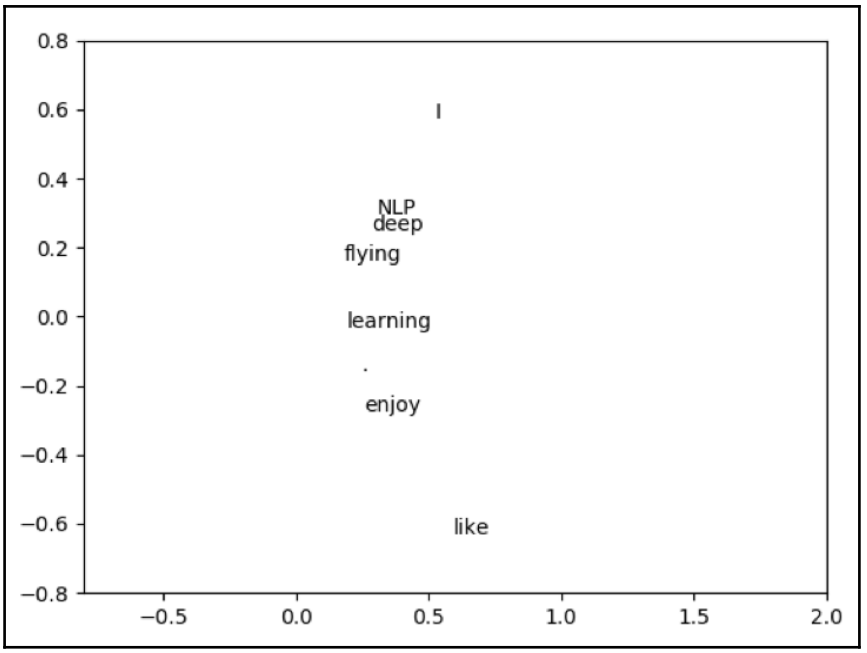
Chapter 1: Maths for Neural Networks

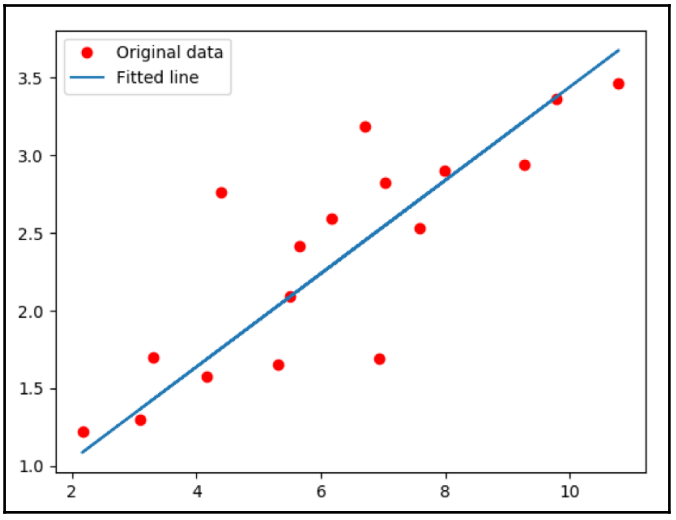


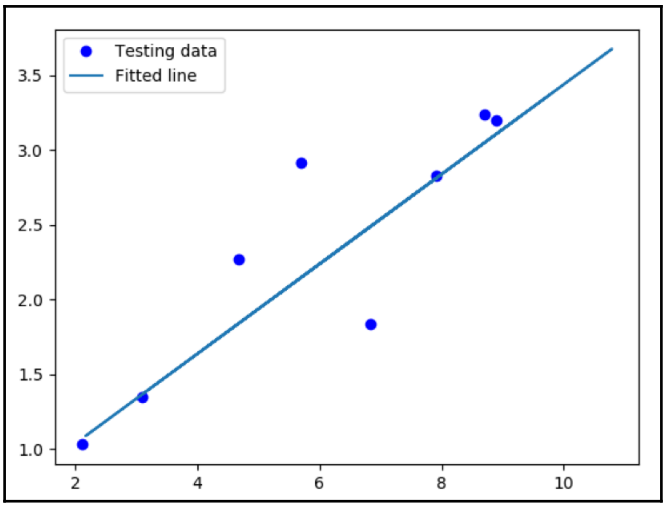
```
convert matrices to tensor objects
mport ...

Define a 2x2 matrix in 3 different ways
1 = [[1.0, 2.0], [3.0, 40]]
2 = np.array([[1.0, 2.0], [3.0, 40]], dtype=np.float32)
3 = tf.constant([[1.0, 2.0], [3.0, 40]])

rint(type(m1))
rint(type(m2))
rint(type(m3))

1 = tf.convert_to_tensor(m1, dtype=tf.float32)
2 = tf.convert_to_tensor(m2, dtype=tf.float32)
3 = tf.convert_to_tensor(m3, dtype=tf.float32)

rint(type(t1))
rint(type(t2))
rint(type(t3))
```

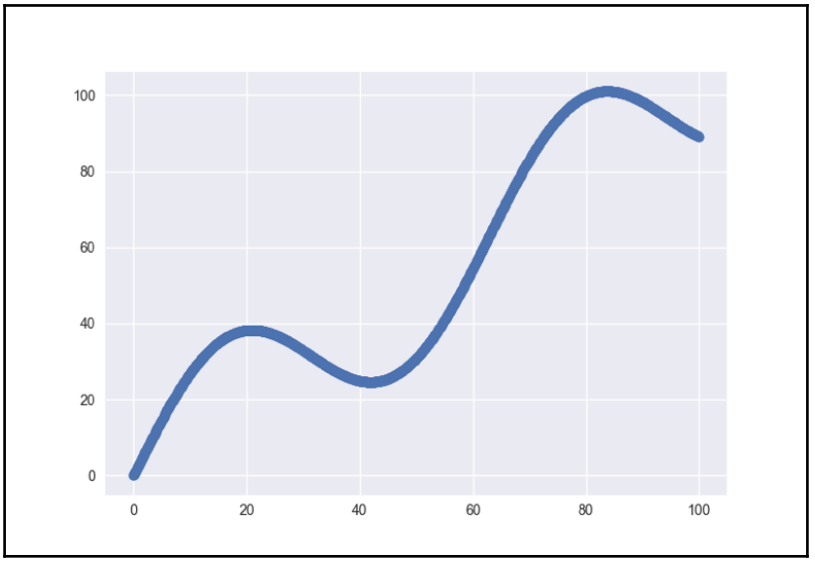
Copy Reference ⌥⇧⌘C
Paste ⌘V
Paste from History... ⇧⌘V
Paste Simple ⌥⇧⌘V
Column Selection Mode ⇧⌘8
Refactor
Folding
Go To
Generate... ⌘N
Run 'matrices' ^⇧R
Debug 'matrices' ^⇧D
Create 'matrices'...
Local History
Git
Execute Line in Console ⌥⇧E
Compare with Clipboard
File Encoding
Open on GitHub
Create Gist...

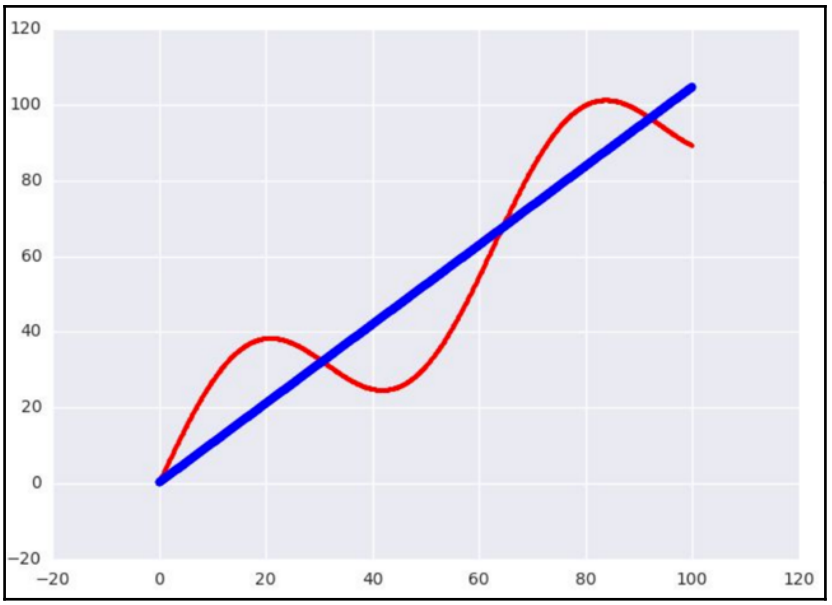



0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.5
0.0
0.5
1.0
1.5
2.0
I
NLP
deep
flying
learning
.
enjoy
like

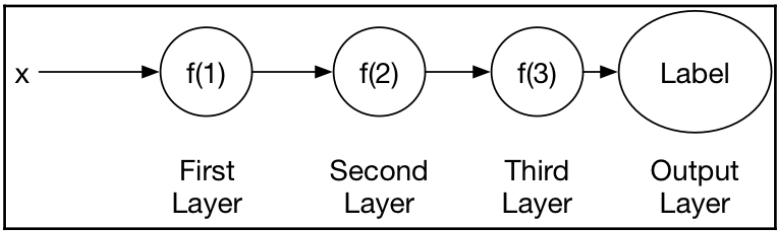

Original data
Fitted line
3.5
3.0
2.5
2.0
1.5
1.0
2
4
6
8
10

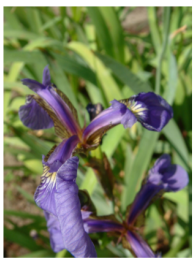

Testing data
Fitted line
3.5
3.0
2.5
2.0
1.5
1.0
2
4
6
8
10


100
80
60
40
20
0
0
20
40
60
80
100


120
100
80
60
40
20
0
−20
−20
0
20
40
60
80
100
120

Chapter 2: Deep Feedforward Networks

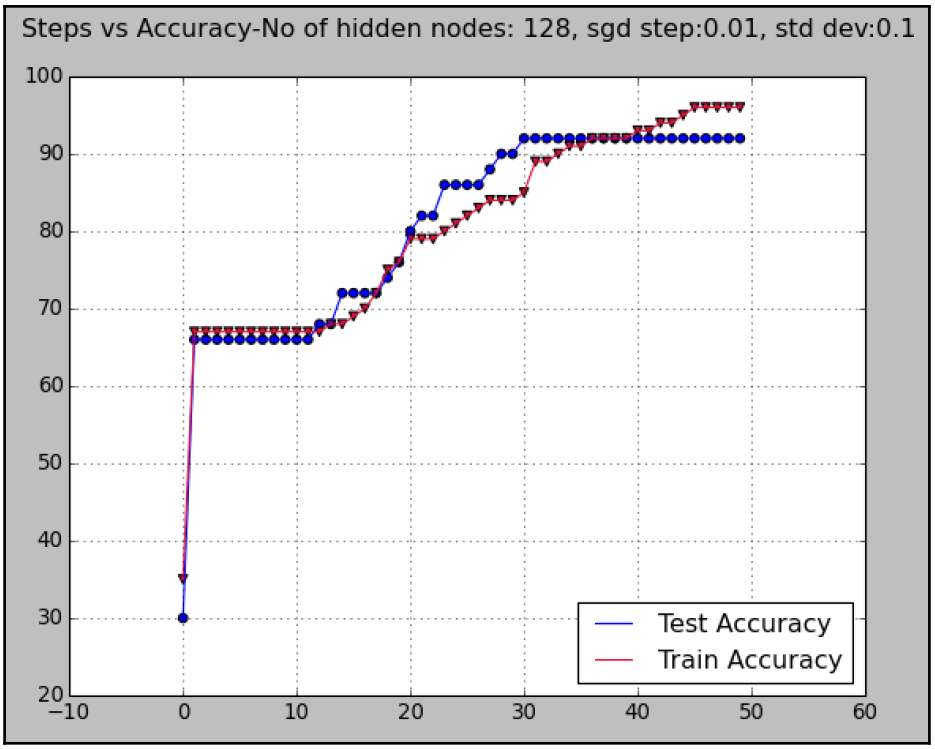


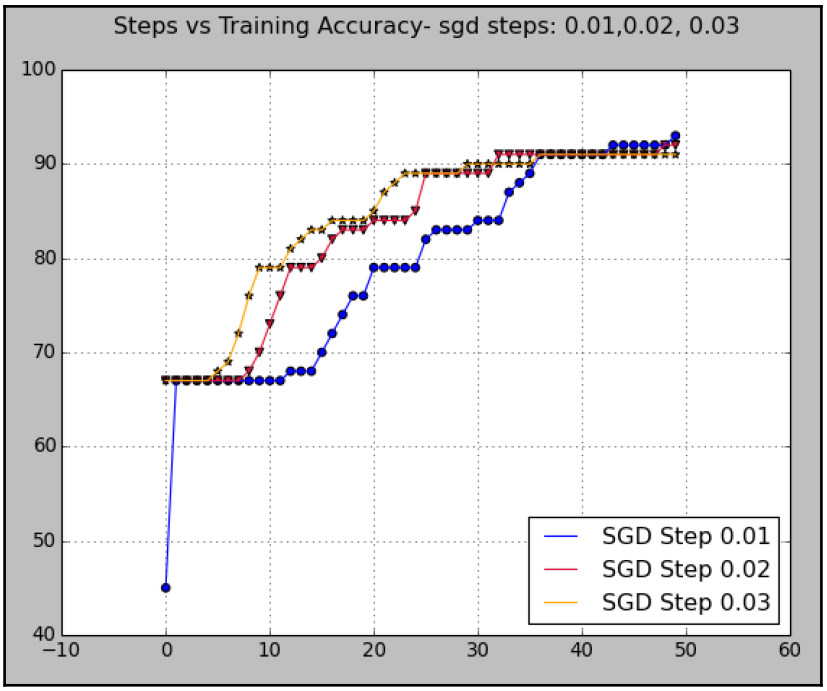


Iris Setosa

Iris Versicolor

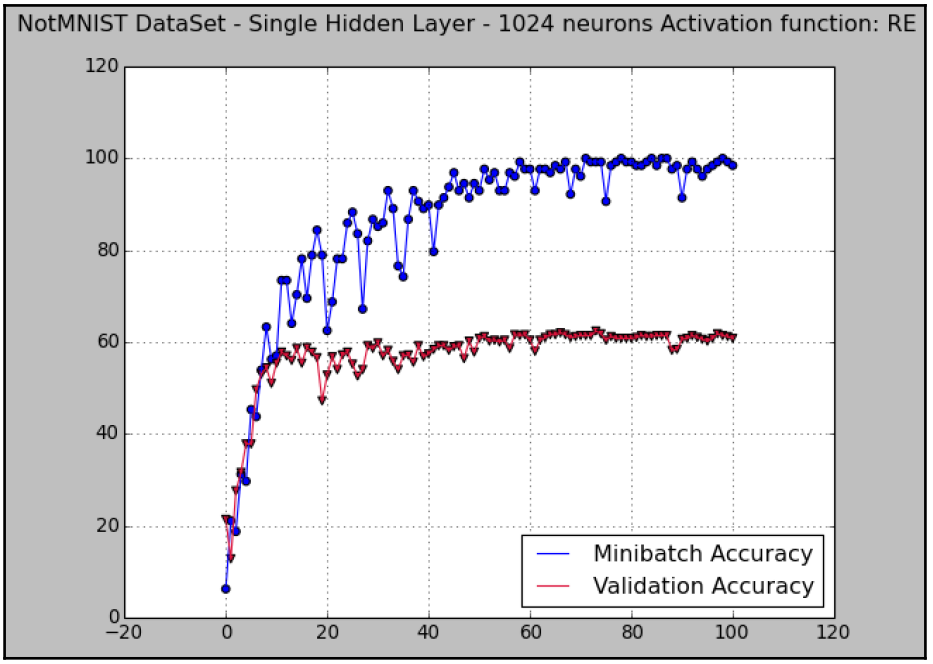
Iris Virginica

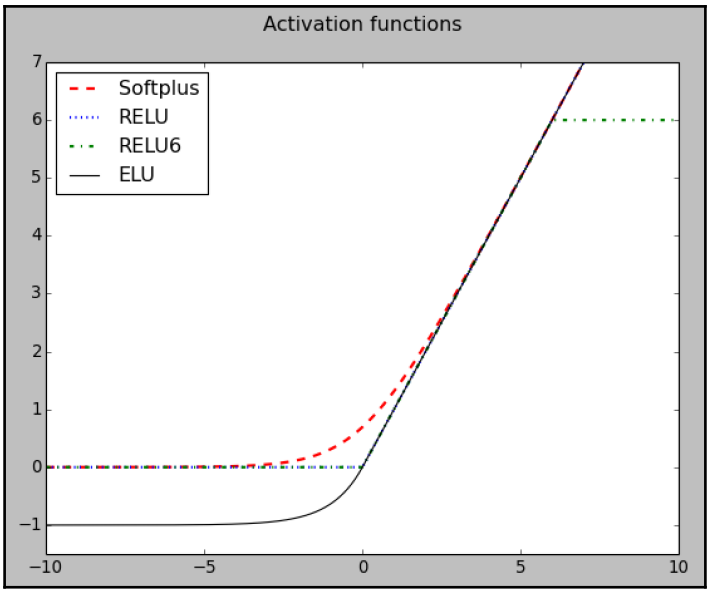

Steps vs Accuracy-No of hidden nodes: 128, sgd step:0.01, std dev:0.1
100
90
80
70
60
50
40
30
20
−10
0
10
20
30
40
50
60
Test Accuracy
Train Accuracy

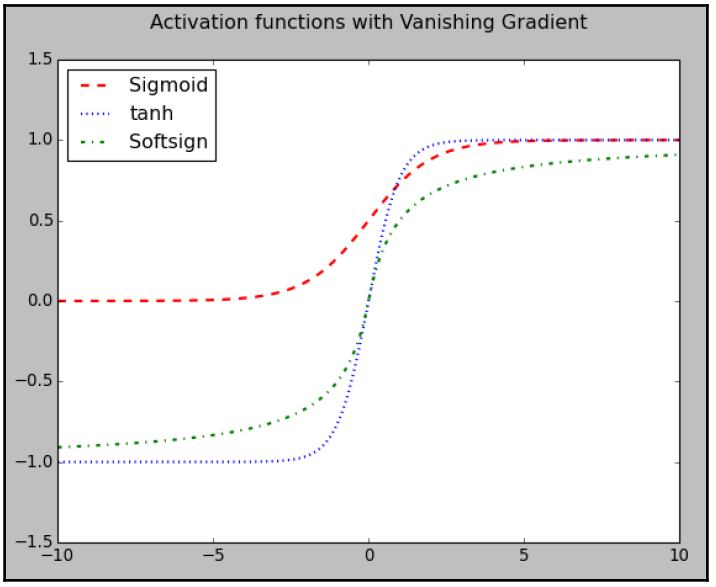


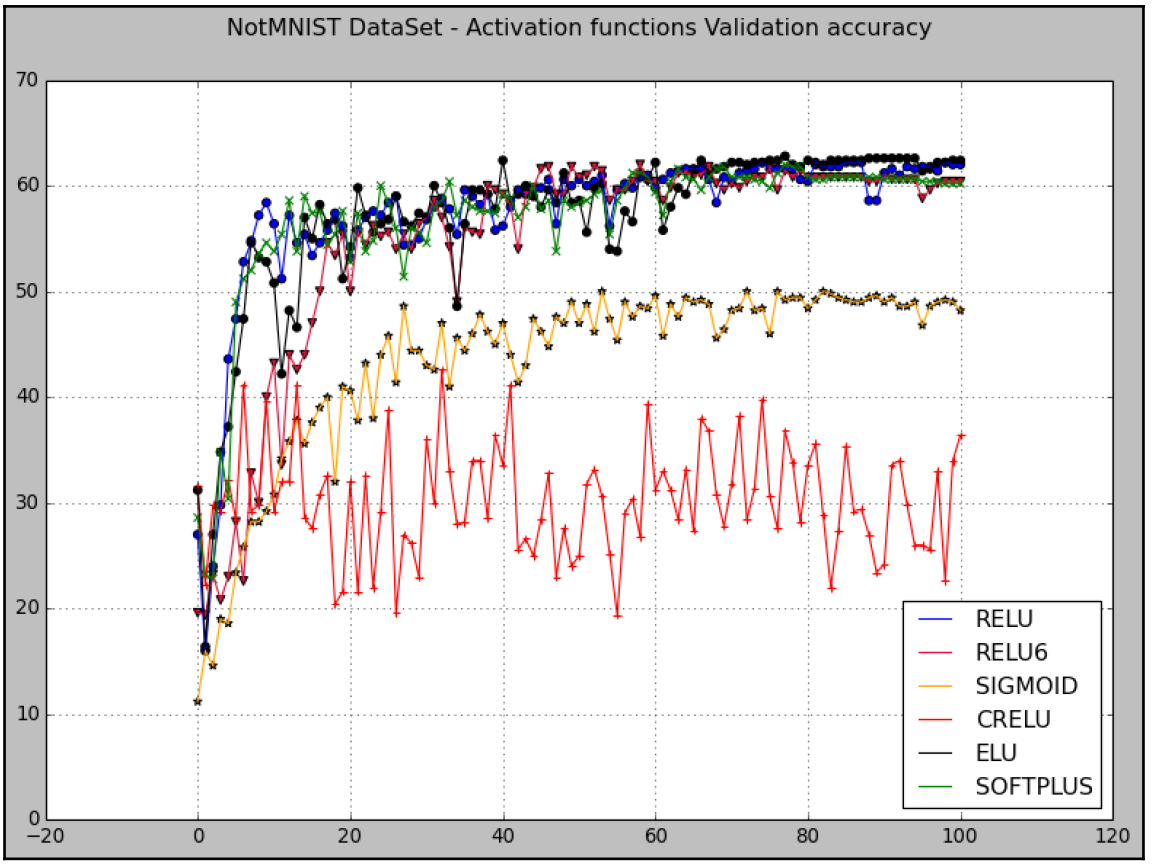

```
dataset_names = {list} <type 'list'>: ['./notMNIST_
  __len__ = {int} 10
  00 = {str} './notMNIST_large_v2/A.pickle'
  01 = {str} './notMNIST_large_v2/B.pickle'
  02 = {str} './notMNIST_large_v2/C.pickle'
  03 = {str} './notMNIST_large_v2/D.pickle'
  04 = {str} './notMNIST_large_v2/E.pickle'
  05 = {str} './notMNIST_large_v2/F.pickle'
  06 = {str} './notMNIST_large_v2/G.pickle'
  07 = {str} './notMNIST_large_v2/H.pickle'
  08 = {str} './notMNIST_large_v2/I.pickle'
  09 = {str} './notMNIST_large_v2/J.pickle'
```

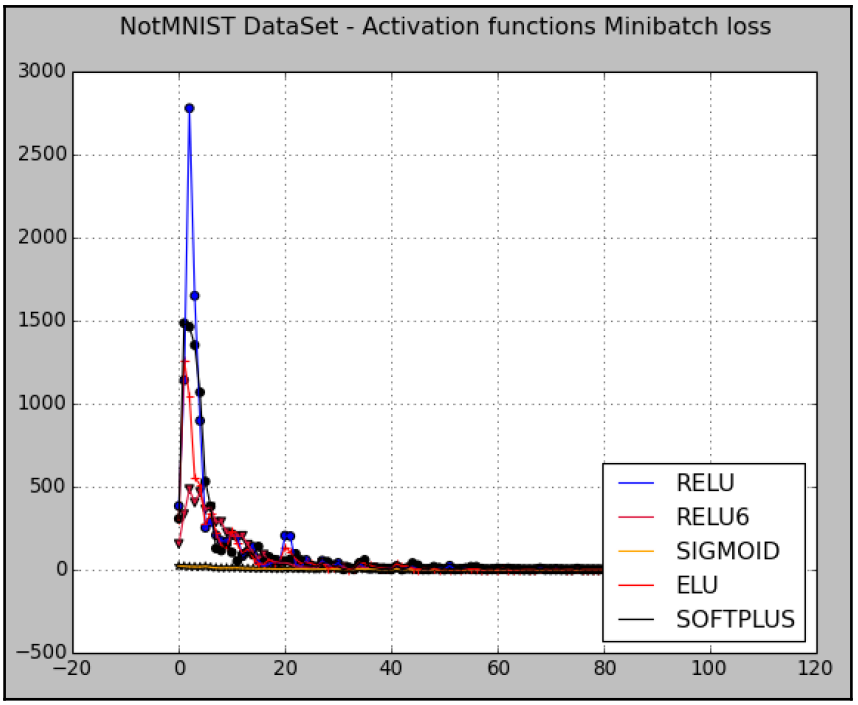
```
dataset_names = {list} <type 'list'>: ['./not
  __len__ = {int} 10
  00 = {str} './notMNIST_small/A.pickle'
  01 = {str} './notMNIST_small/B.pickle'
  02 = {str} './notMNIST_small/C.pickle'
  03 = {str} './notMNIST_small/D.pickle'
  04 = {str} './notMNIST_small/E.pickle'
  05 = {str} './notMNIST_small/F.pickle'
  06 = {str} './notMNIST_small/G.pickle'
  07 = {str} './notMNIST_small/H.pickle'
  08 = {str} './notMNIST_small/I.pickle'
  09 = {str} './notMNIST_small/J.pickle'
```



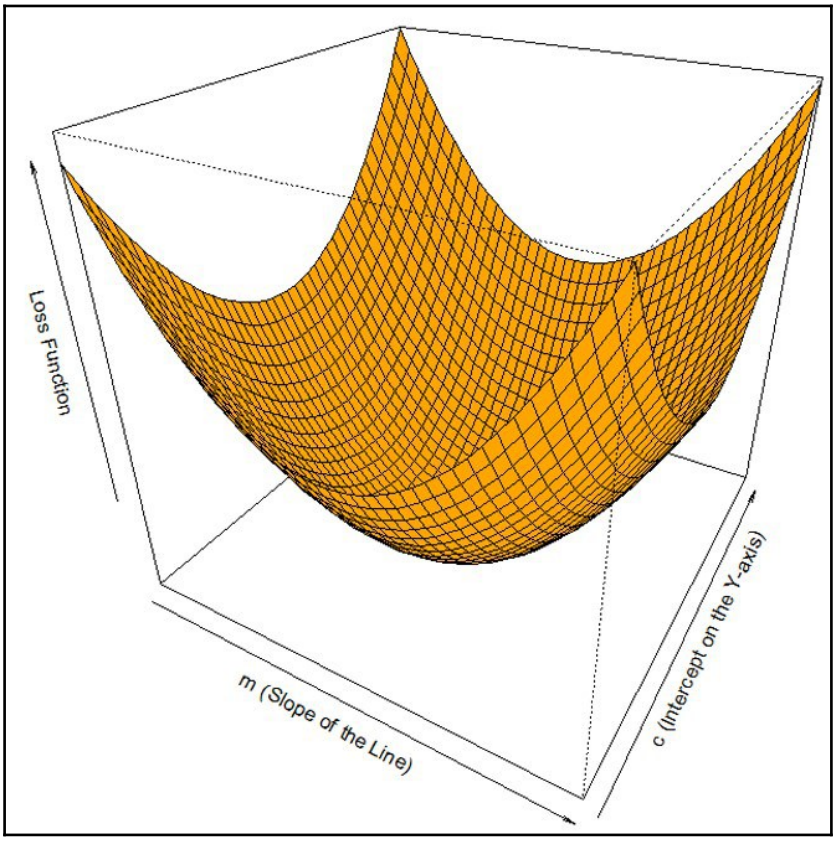

Activation functions
Softplus
RELU
RELU6
ELU
7
6
5
4
3
2
1
0
−1
−10
−5
0
5
10

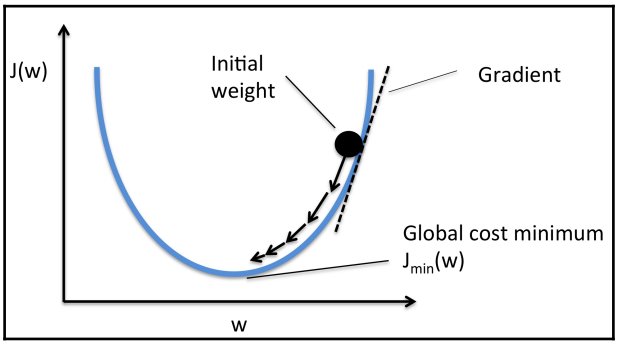

Activation functions with Vanishing Gradient
Sigmoid
tanh
Softsign
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−10
−5
0
5
10

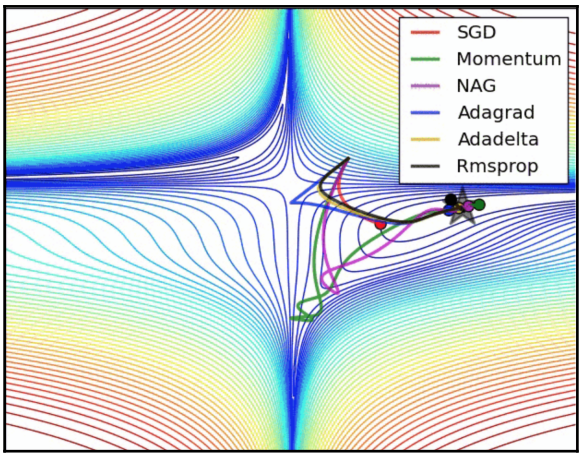

NotMNIST DataSet - Activation functions Validation accuracy
70
60
50
40
30
20
10
0
−20
0
20
40
60
80
100
120
RELU
RELU6
SIGMOID
CRELU
ELU
SOFTPLUS

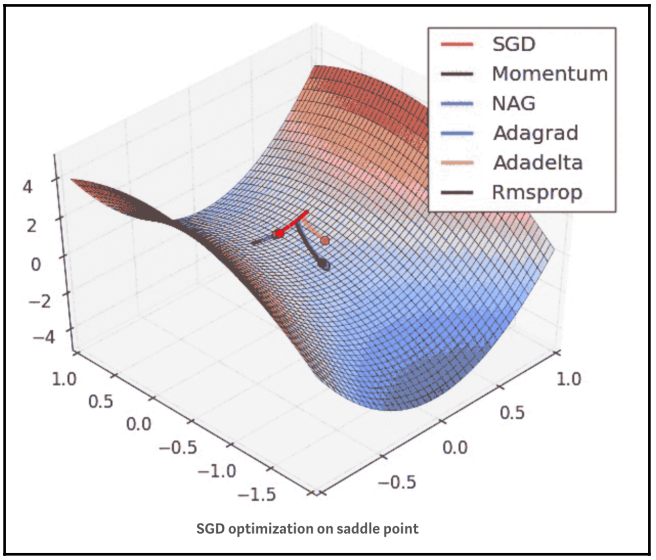

NotMNIST DataSet - Activation functions Minibatch loss
3000
2500
2000
1500
1000
500
0
−500
−20
0
20
40
60
80
100
120
RELU
RELU6
SIGMOID
ELU
SOFTPLUS

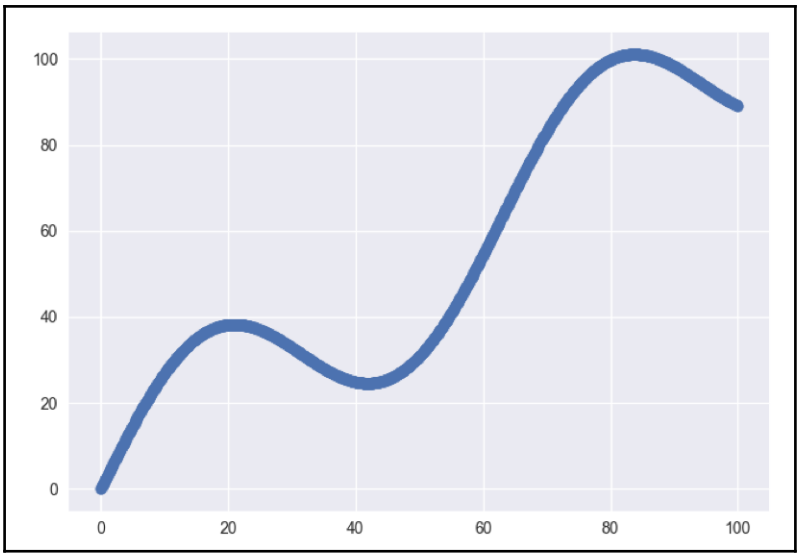
Chapter 3: Optimization for Neural Networks

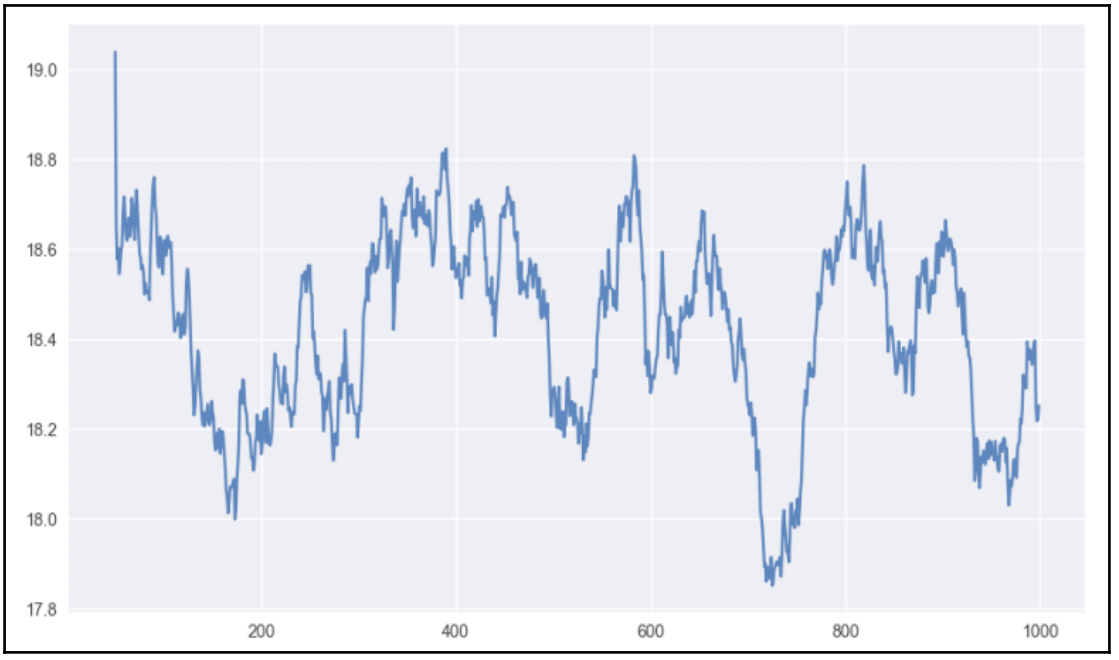


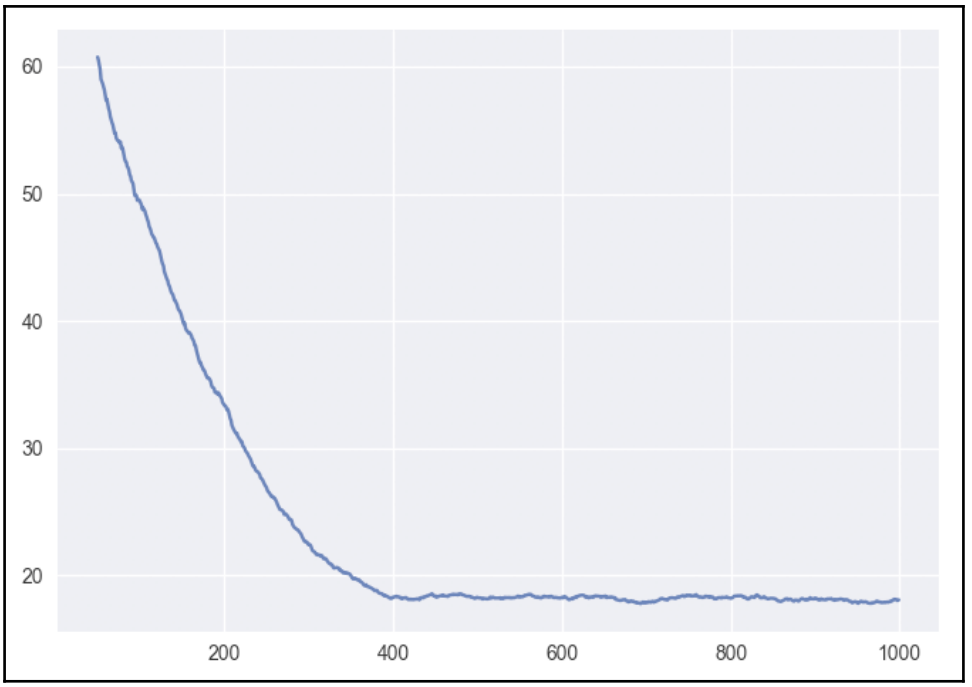



SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop

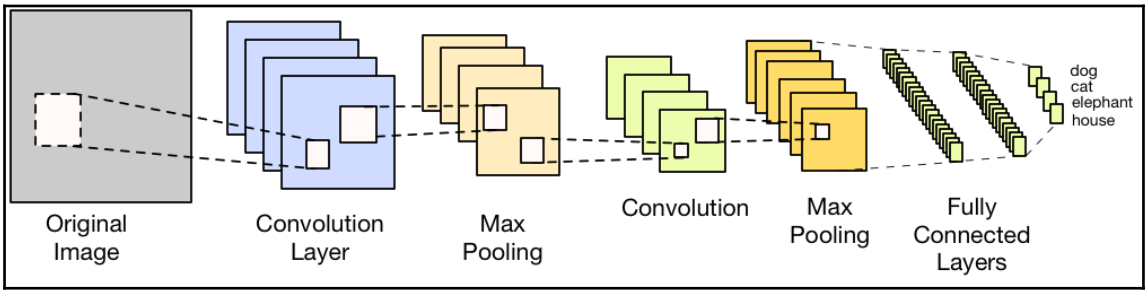

SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5
SGD optimization on saddle point

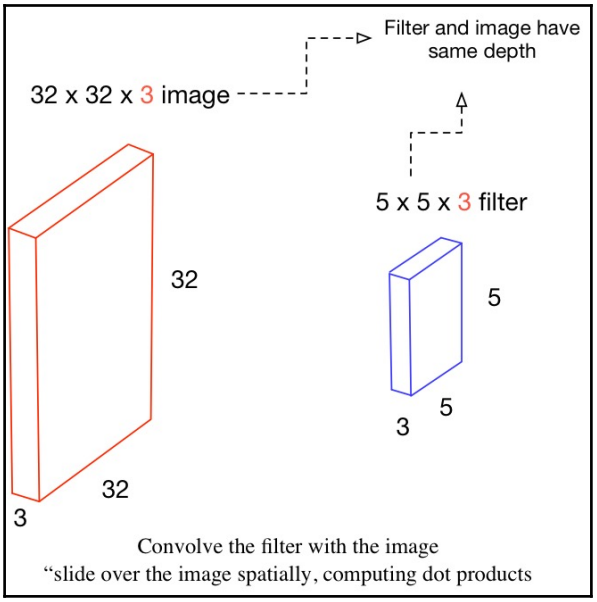

100
80
60
40
20
0
0
20
40
60
80
100

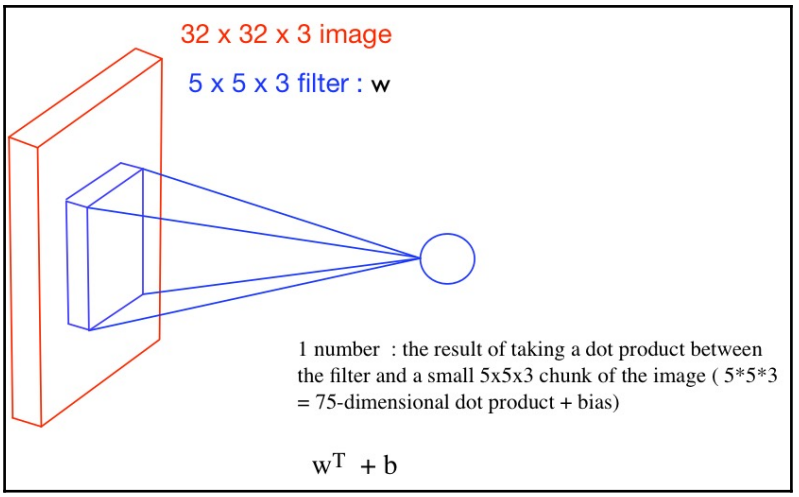

19.0
18.8
18.6
18.4
18.2
18.0
17.8
200
400
600
800
1000

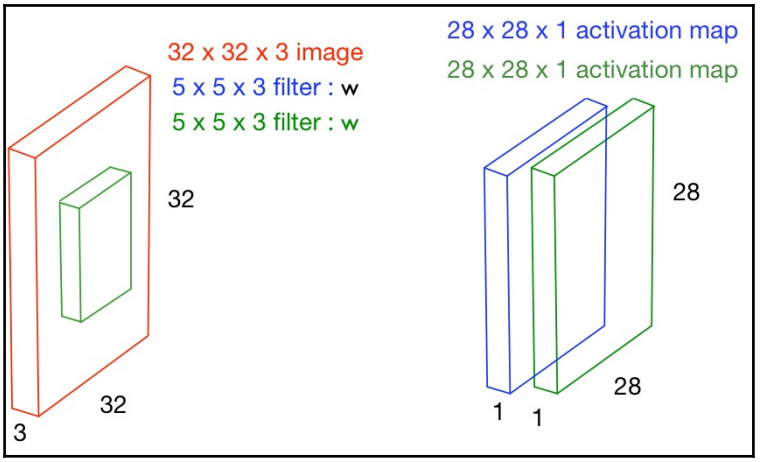

60
50
40
30
20
200
400
600
800
1000

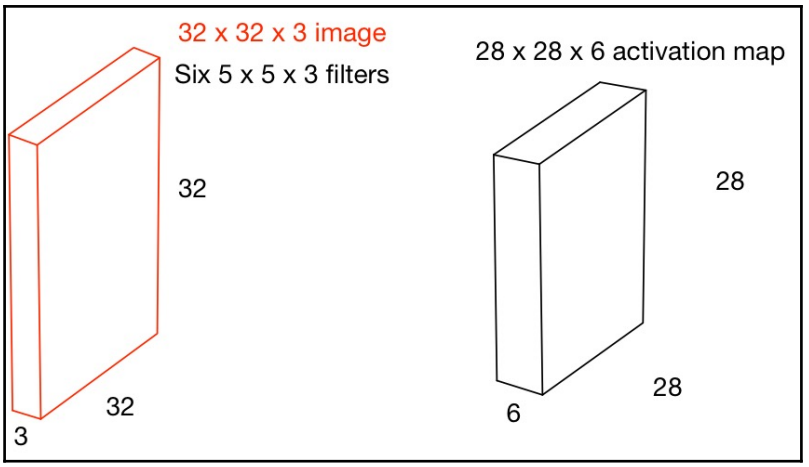
Chapter 4: Convolutional Neural Networks

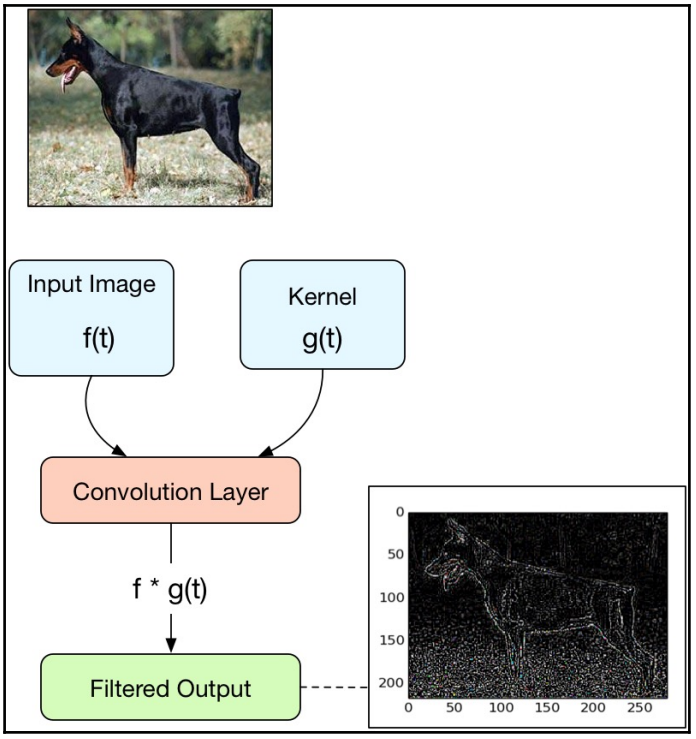





32 x 32 x 3 image
5 x 5 x 3 filter : w
1 number : the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (5*5*3 = 75-dimensional dot product + bias)
$w^T + b$


32 x 32 x 3 image
5 x 5 x 3 filter : w
5 x 5 x 3 filter : w
32
32
3
28 x 28 x 1 activation map
28 x 28 x 1 activation map
28
28
1
1


32 x 32 x 3 image
Six 5 x 5 x 3 filters
32
32
3
28 x 28 x 6 activation map
28
28
6


Input Image
f(t)
Kernel
g(t)
Convolution Layer
f * g(t)
Filtered Output
0
50
100
150
200
0
50
100
150
200
250

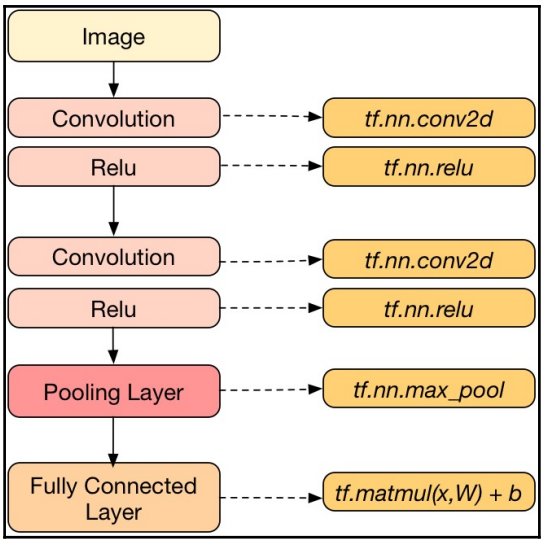
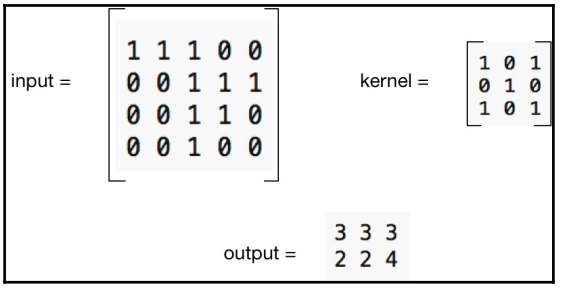
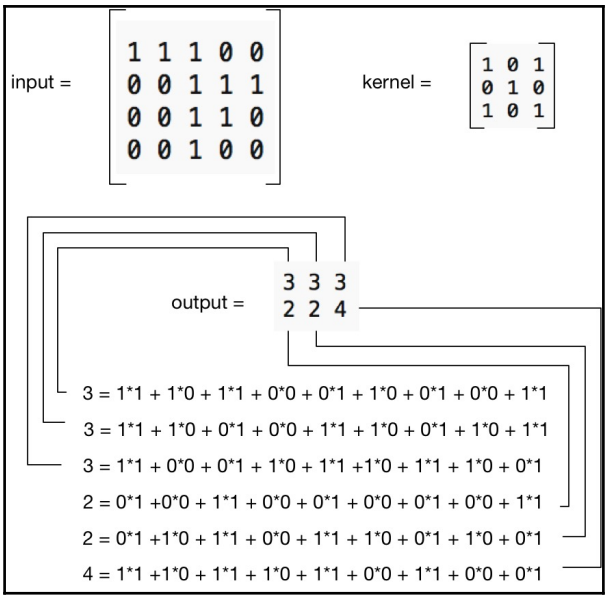


Image
Convolution
tf.nn.conv2d
Relu
tf.nn.relu
Convolution
tf.nn.conv2d
Relu
tf.nn.relu
Pooling Layer
tf.nn.max_pool
Fully Connected Layer
tf.matmul(x,W) + b


input =
1 1 1 0 0
0 0 1 1 1
0 0 1 1 0
0 0 1 0 0
kernel =
1 0 1
0 1 0
1 0 1
output =
3 3 3
2 2 4


input =
1 1 1 0 0
0 0 1 1 1
0 0 1 1 0
0 0 1 0 0
kernel =
1 0 1
0 1 0
1 0 1
output =
3 3 3
2 2 4
3 = 1*1 + 1*0 + 1*1 + 0*0 + 0*1 + 1*0 + 0*1 + 0*0 + 1*1
3 = 1*1 + 1*0 + 0*1 + 0*0 + 1*1 + 1*0 + 0*1 + 1*0 + 1*1
3 = 1*1 + 0*0 + 0*1 + 1*0 + 1*1 +1*0 + 1*1 + 1*0 + 0*1
2 = 0*1 +0*0 + 1*1 + 0*0 + 0*1 + 0*0 + 0*1 + 0*0 + 1*1
2 = 0*1 +1*0 + 1*1 + 0*0 + 1*1 + 1*0 + 0*1 + 1*0 + 0*1
4 = 1*1 +1*0 + 1*1 + 1*0 + 1*1 + 0*0 + 1*1 + 0*0 + 0*1

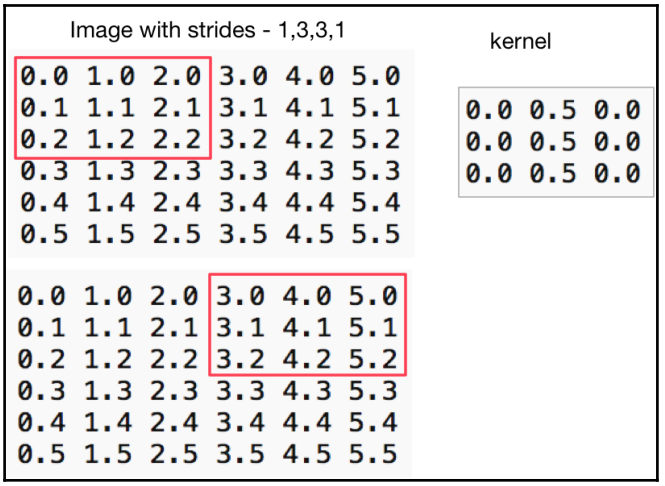
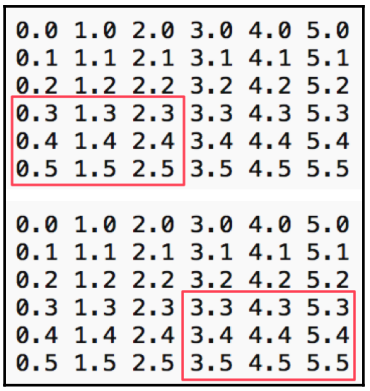


Image with strides - 1,3,3,1
kernel
0.0 1.0 2.0 3.0 4.0 5.0
0.1 1.1 2.1 3.1 4.1 5.1
0.2 1.2 2.2 3.2 4.2 5.2
0.3 1.3 2.3 3.3 4.3 5.3
0.4 1.4 2.4 3.4 4.4 5.4
0.5 1.5 2.5 3.5 4.5 5.5
0.0 0.5 0.0
0.0 0.5 0.0
0.0 0.5 0.0
0.0 1.0 2.0 3.0 4.0 5.0
0.1 1.1 2.1 3.1 4.1 5.1
0.2 1.2 2.2 3.2 4.2 5.2
0.3 1.3 2.3 3.3 4.3 5.3
0.4 1.4 2.4 3.4 4.4 5.4
0.5 1.5 2.5 3.5 4.5 5.5



- `"VALID"` = without padding:

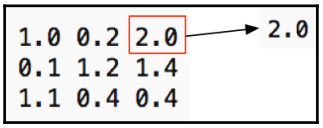
```
   inputs:         1  2  3  4  5  6  7  8  9  10 11 (12 13)
                  |_____________________|                dropped
                                        |_____________________|
```

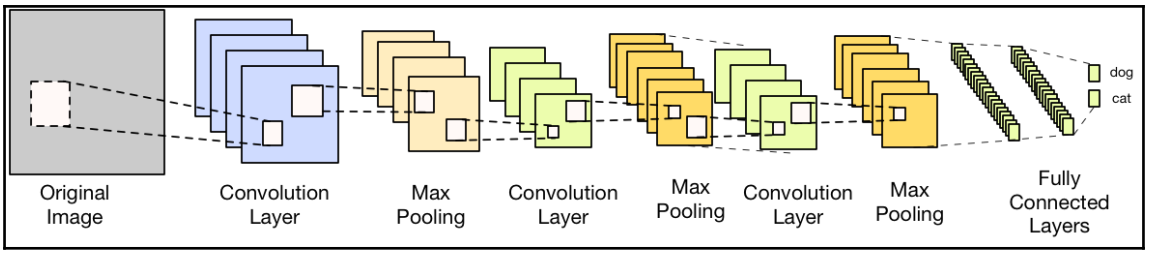
- `"SAME"` = with zero padding:

```
                pad|                                        |pad
   inputs:       0 |1  2  3  4  5  6  7  8  9  10 11 12 13|0  0
                  |_____________________|
                                        |_____________________|
                                                              |_____________________|
```

In this example:

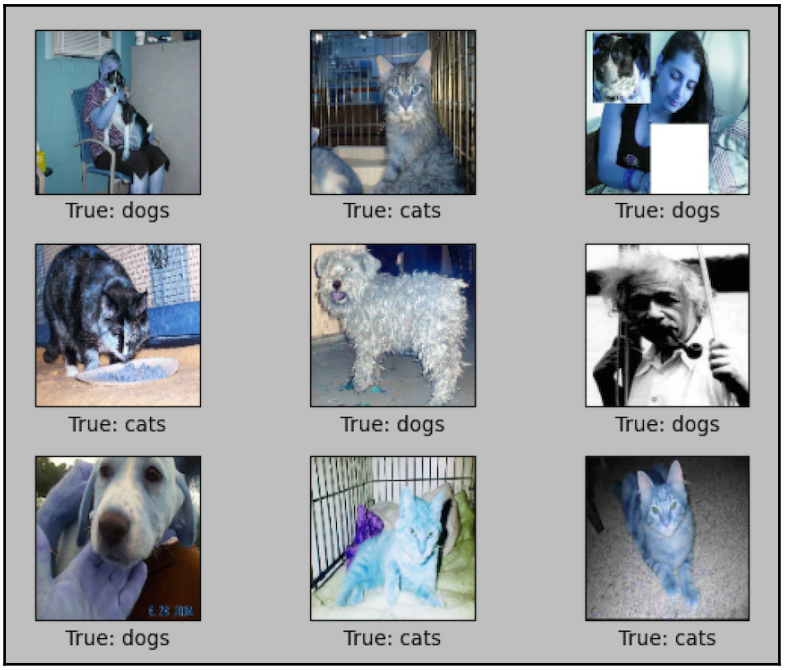
- Input width = 13
- Filter width = 6
- Stride = 5



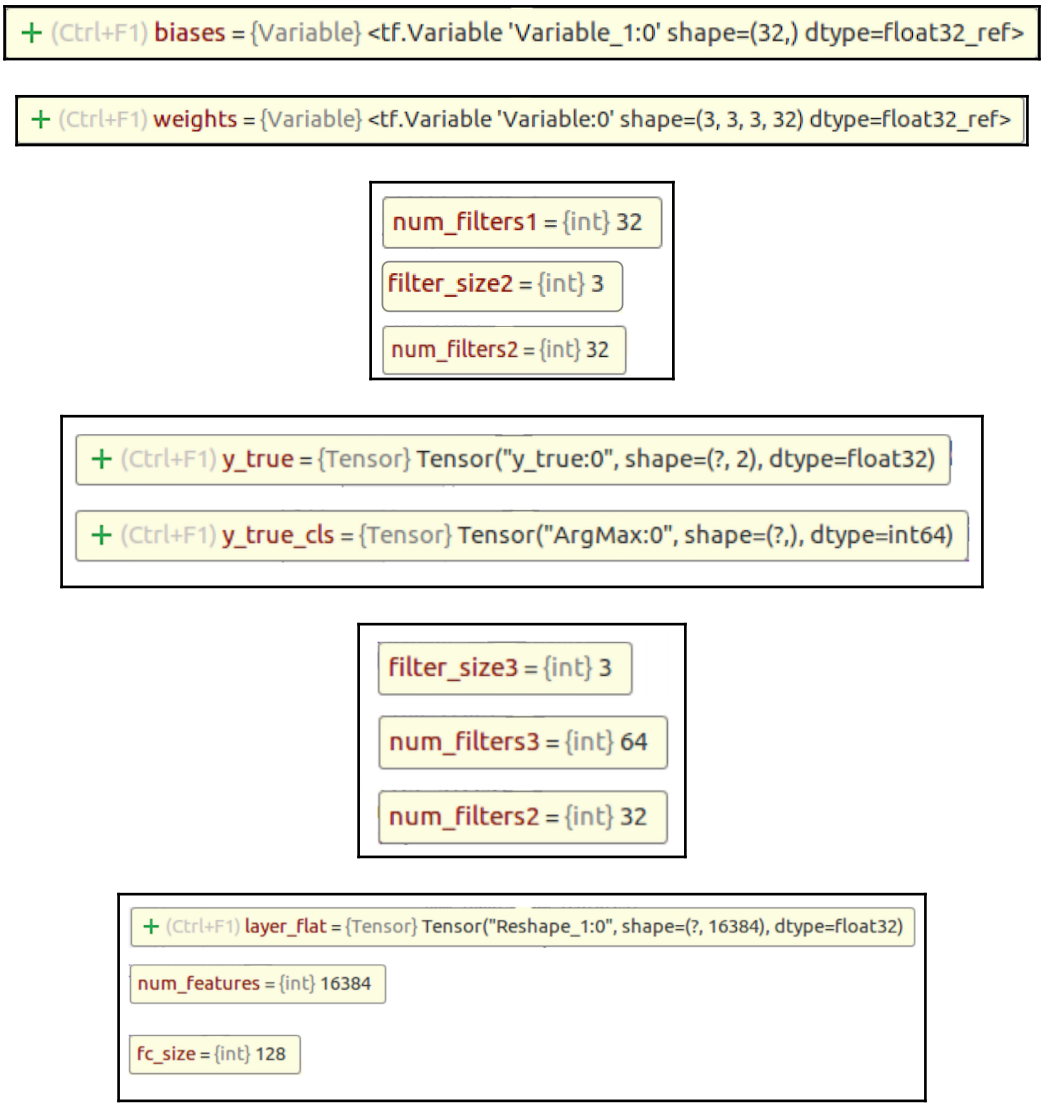


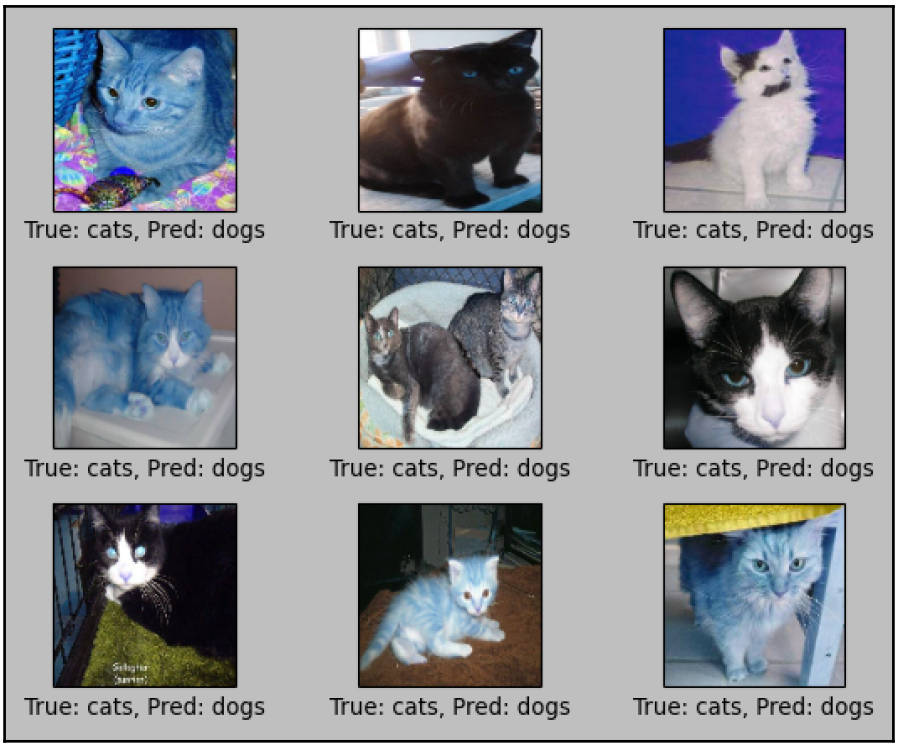

```
+ (Ctrl+F1) images.shape = {tuple} <type 'tuple'>: (291, 128, 128, 3)
```

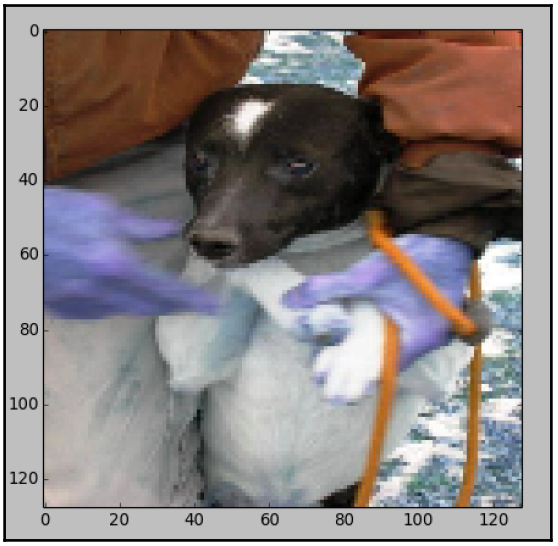
```
validation_size = {int} 58
```

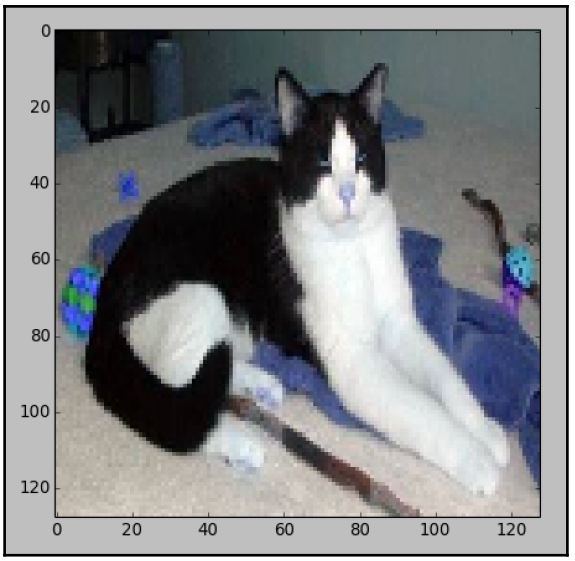


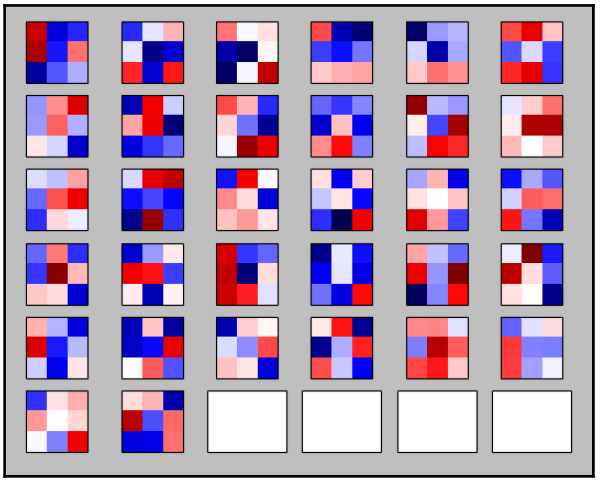

```
num_channels = {int} 3
filter_size1 = {int} 3
num_filters = {int} 32
```

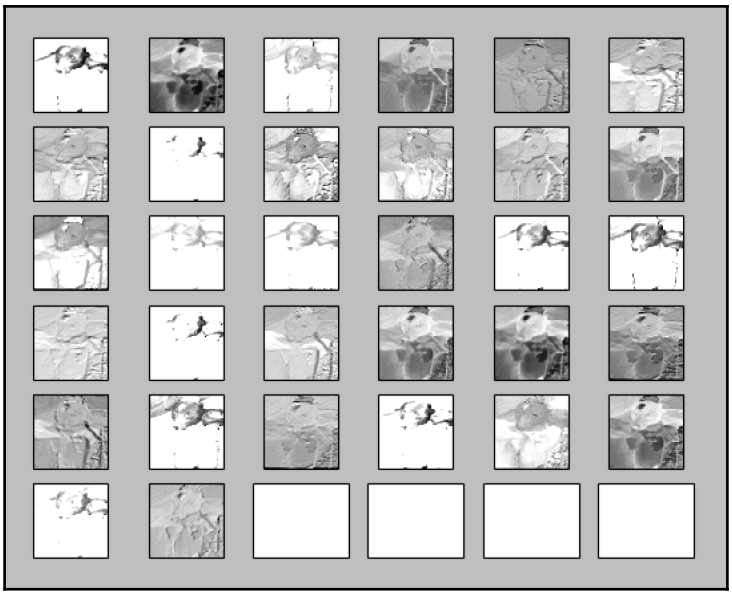

+ (Ctrl+F1) biases = {Variable} <tf.Variable 'Variable_1:0' shape=(32,) dtype=float32_ref>
+ (Ctrl+F1) weights = {Variable} <tf.Variable 'Variable:0' shape=(3, 3, 3, 32) dtype=float32_ref>
num_filters1 = {int} 32
filter_size2 = {int} 3
num_filters2 = {int} 32
+ (Ctrl+F1) y_true = {Tensor} Tensor("y_true:0", shape=(?, 2), dtype=float32)
+ (Ctrl+F1) y_true_cls = {Tensor} Tensor("ArgMax:0", shape=(?,), dtype=int64)
filter_size3 = {int} 3
num_filters3 = {int} 64
num_filters2 = {int} 32
+ (Ctrl+F1) layer_flat = {Tensor} Tensor("Reshape_1:0", shape=(?, 16384), dtype=float32)
num_features = {int} 16384
fc_size = {int} 128

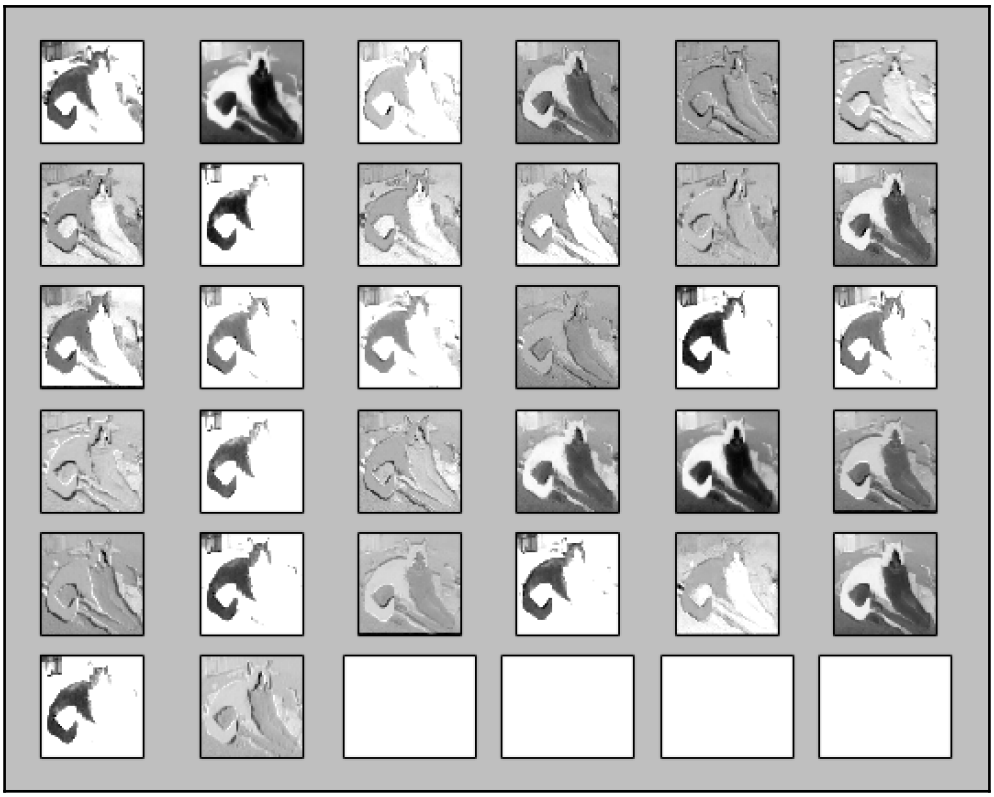

True: cats, Pred: dogs
True: cats, Pred: dogs
True: cats, Pred: dogs
True: cats, Pred: dogs
True: cats, Pred: dogs
True: cats, Pred: dogs
True: cats, Pred: dogs
True: cats, Pred: dogs
True: cats, Pred: dogs

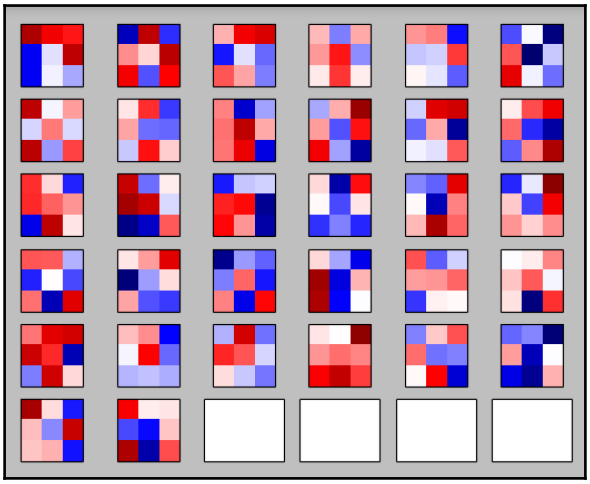

0
20
40
60
80
100
120
0
20
40
60
80
100
120

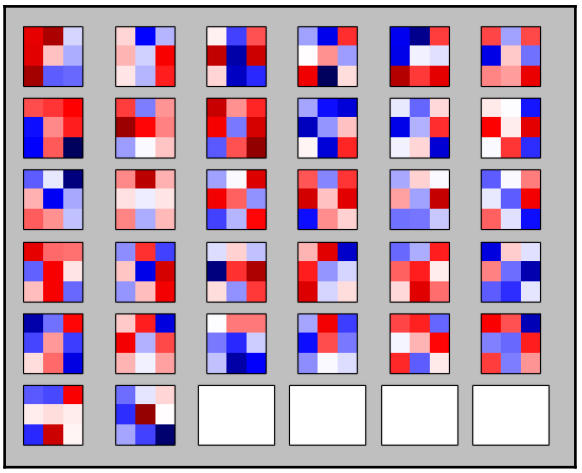

0
20
40
60
80
100
120
0
20
40
60
80
100
120

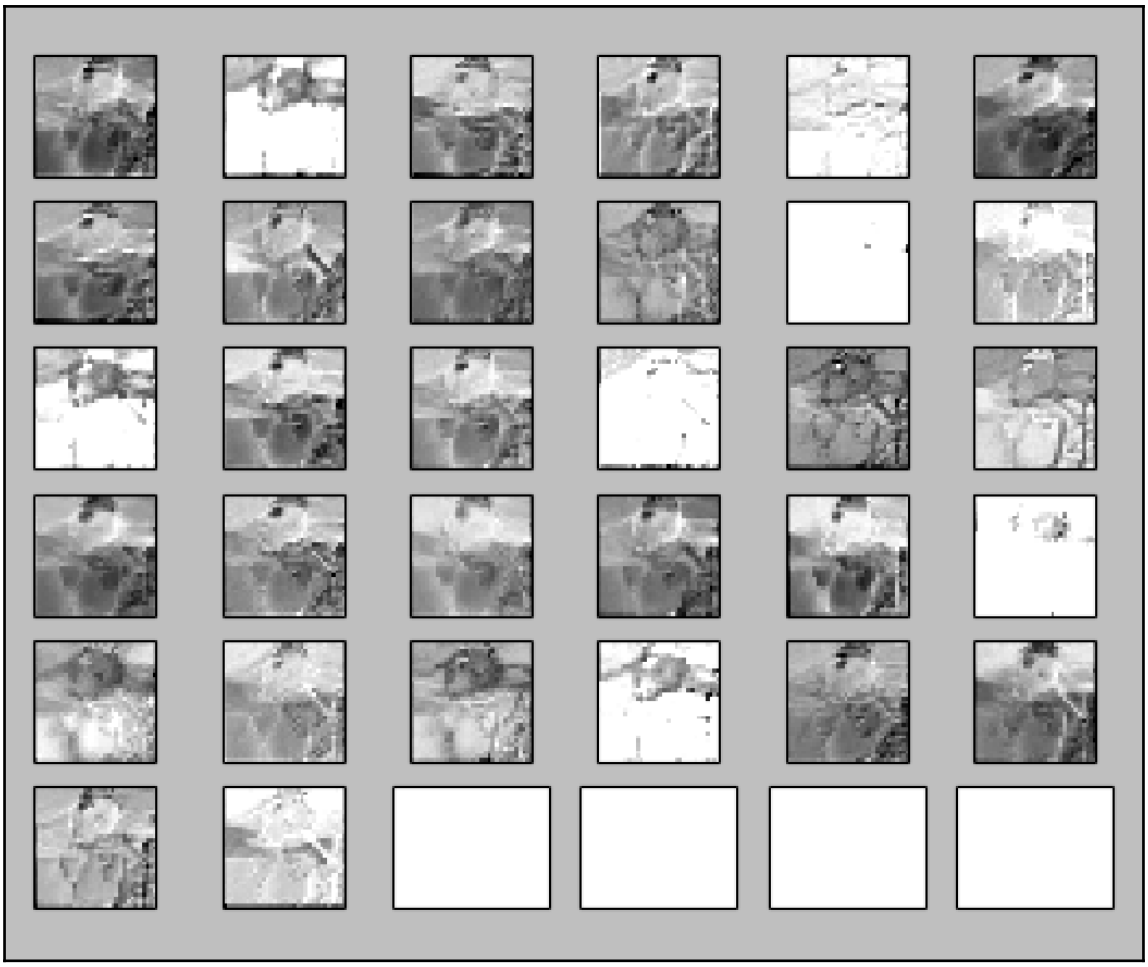


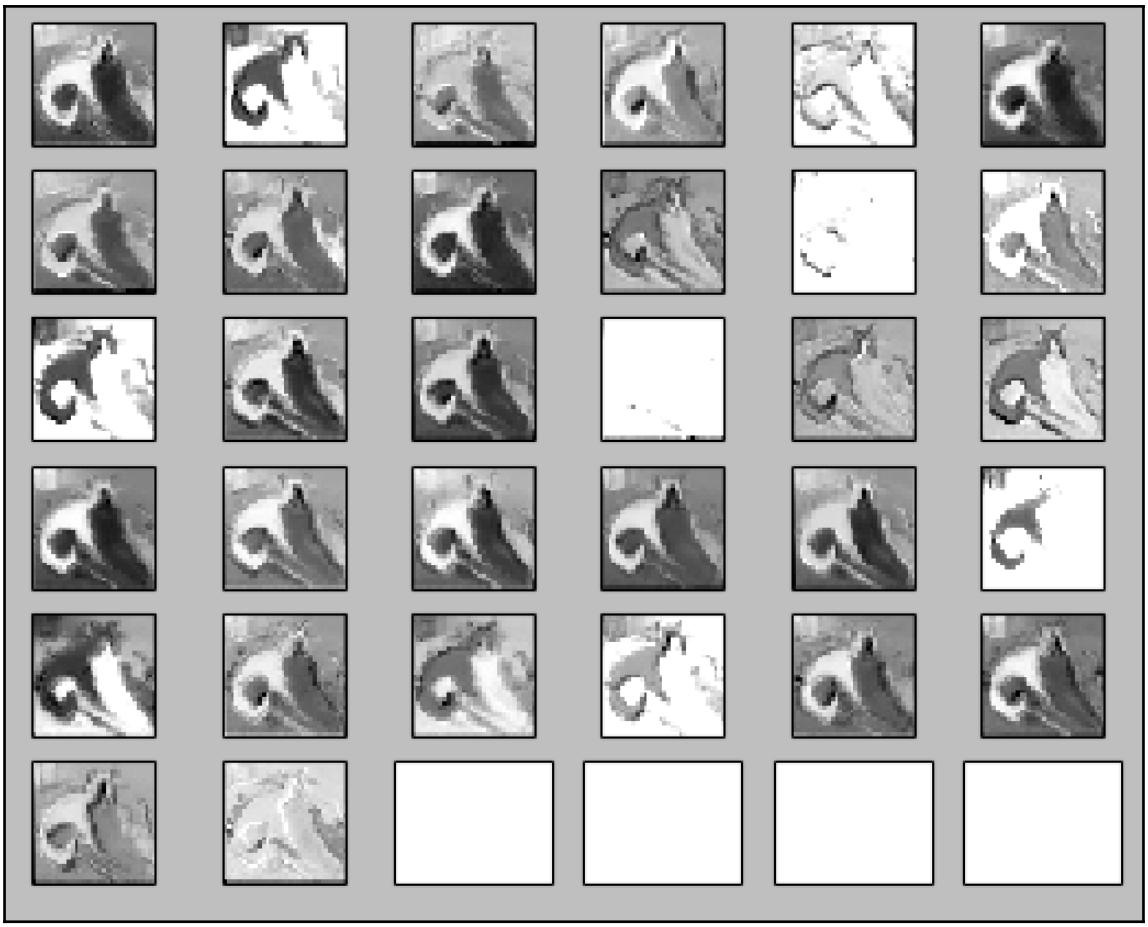


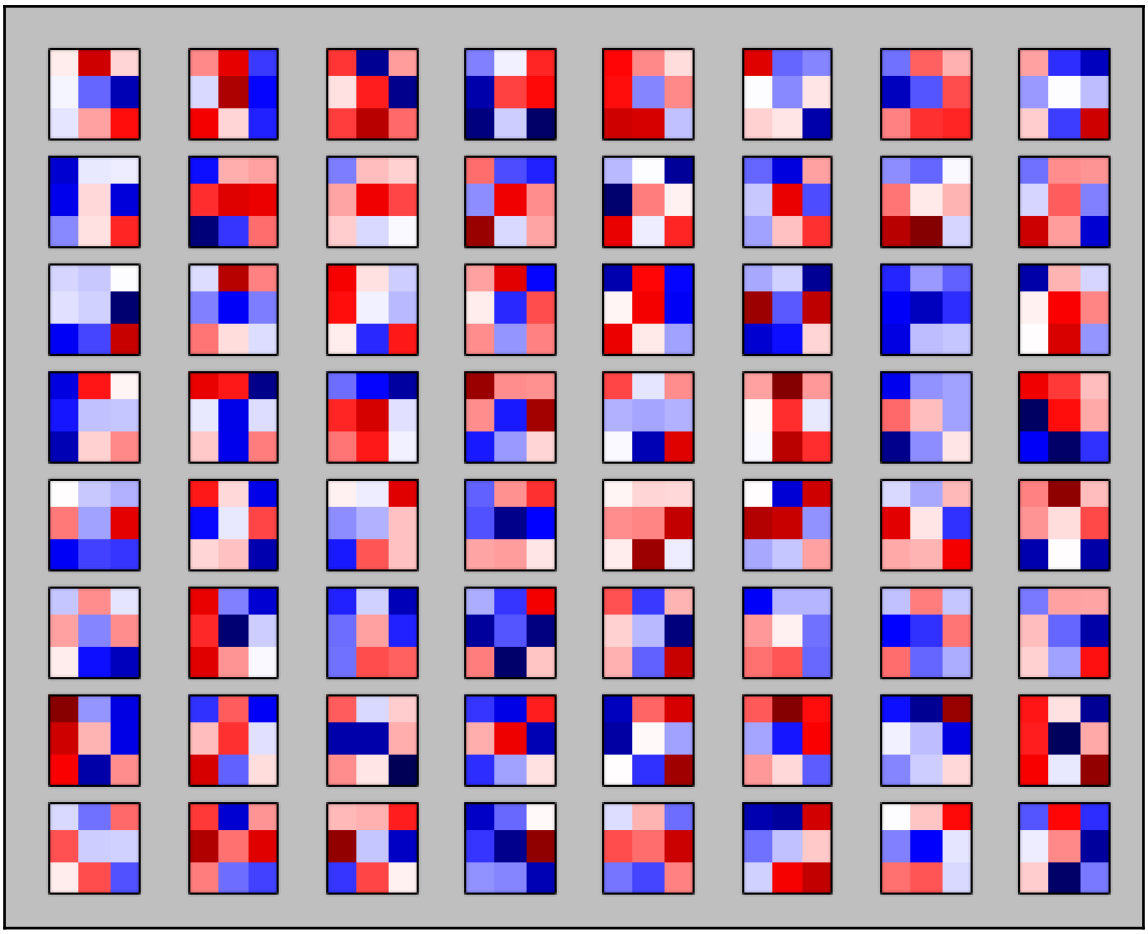


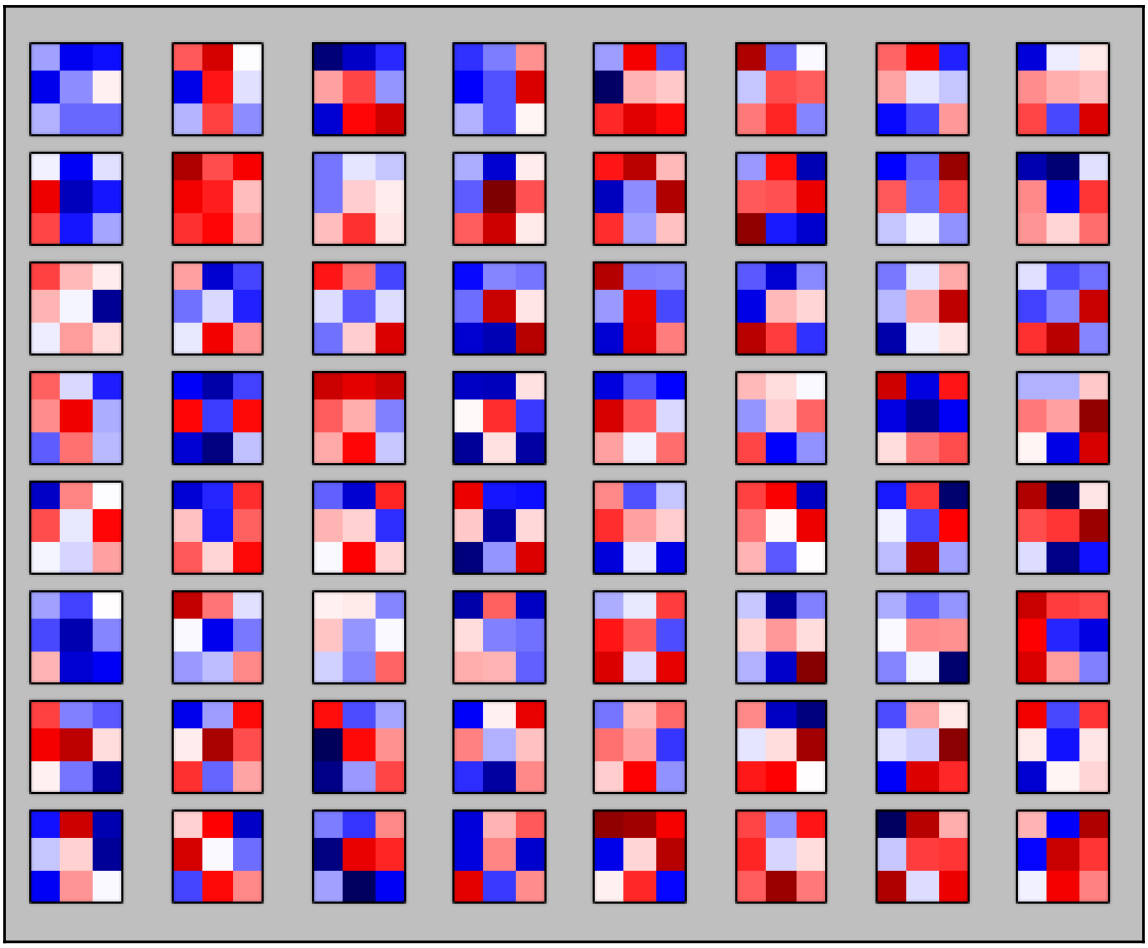


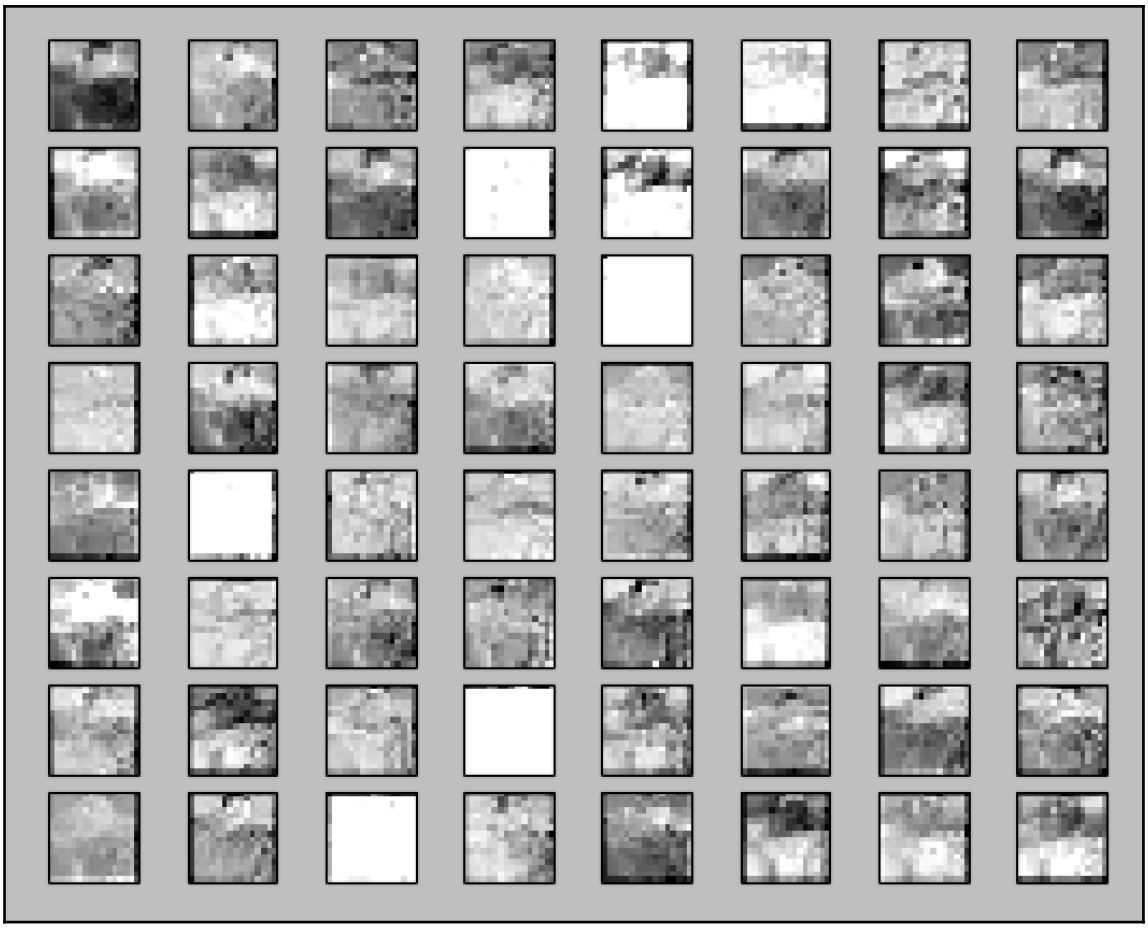


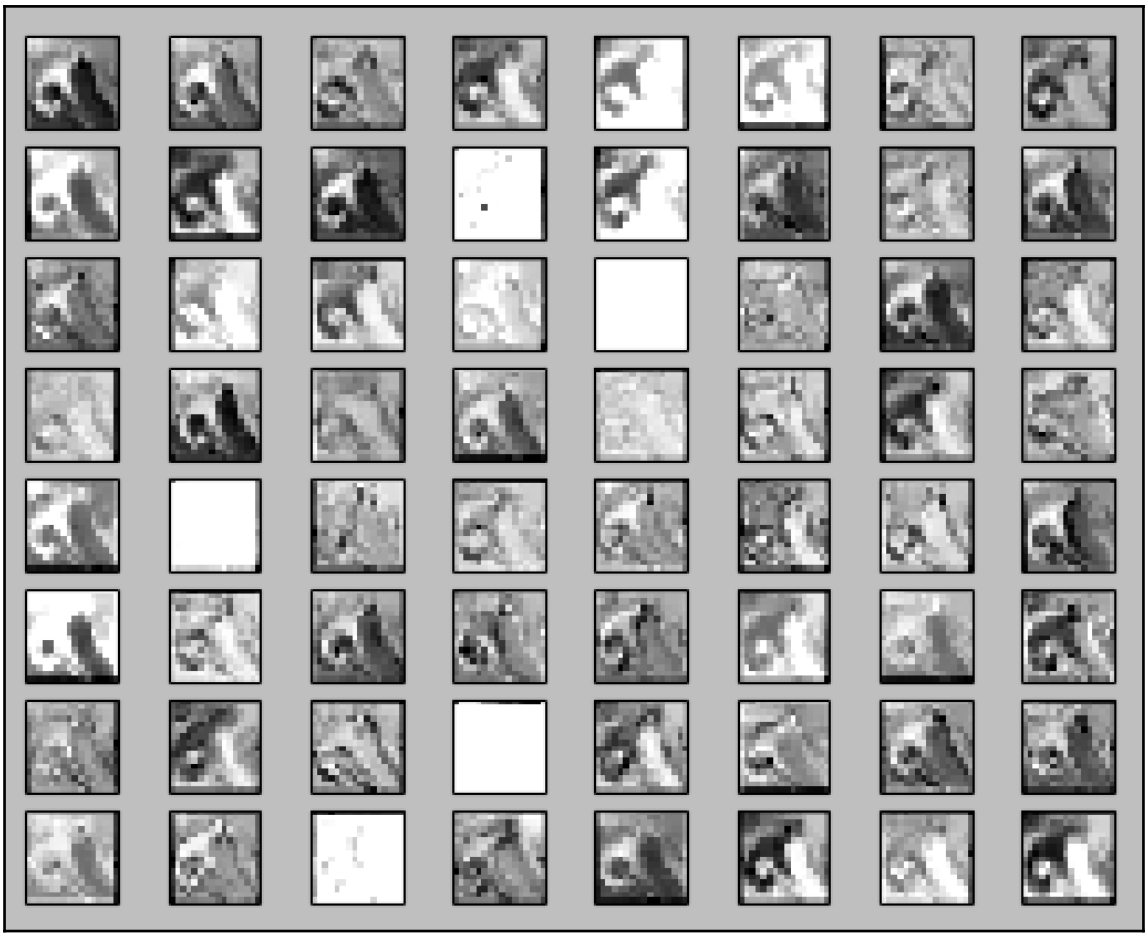




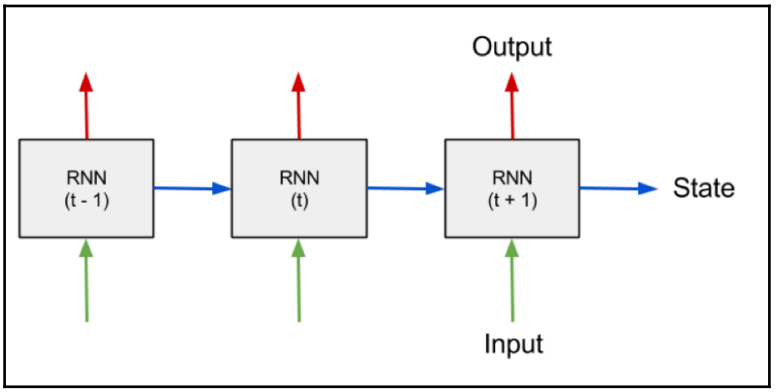


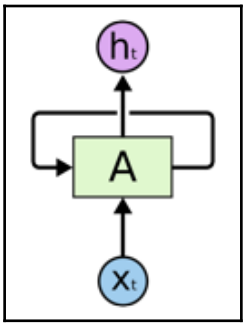


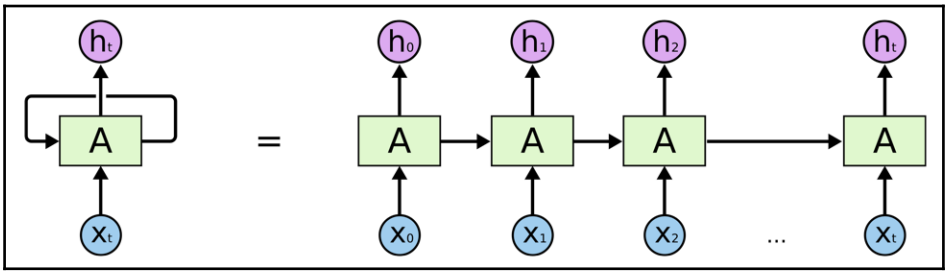


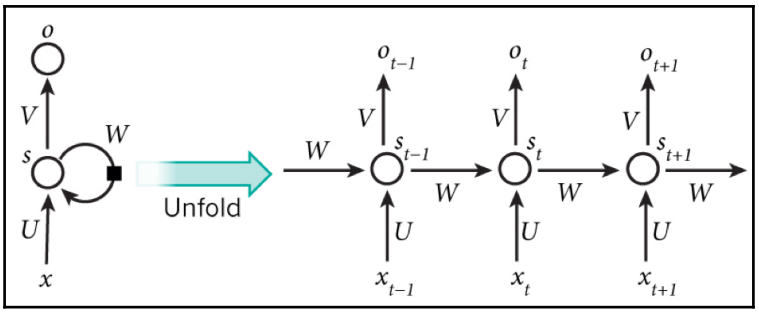


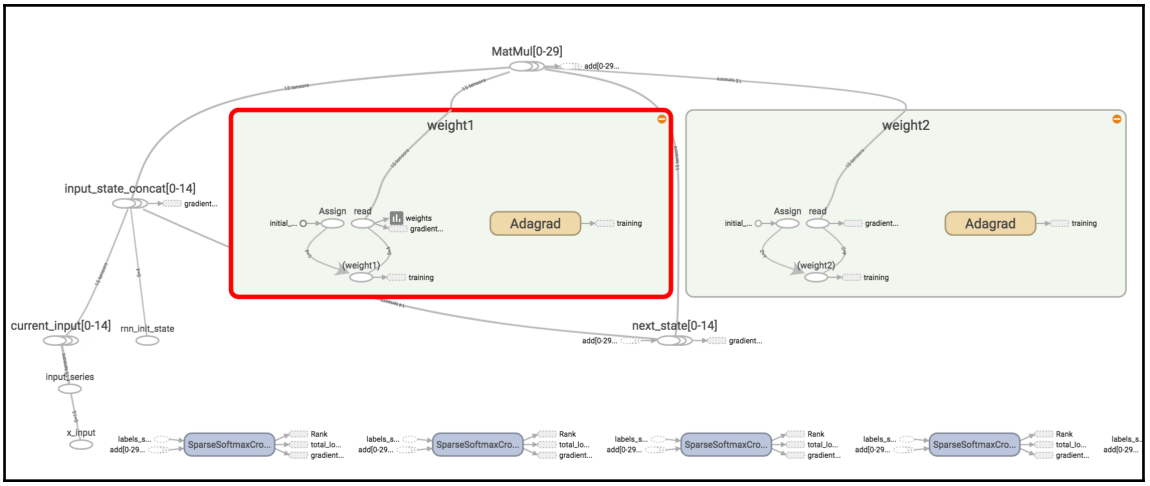
Chapter 5: Recurrent Neural Networks

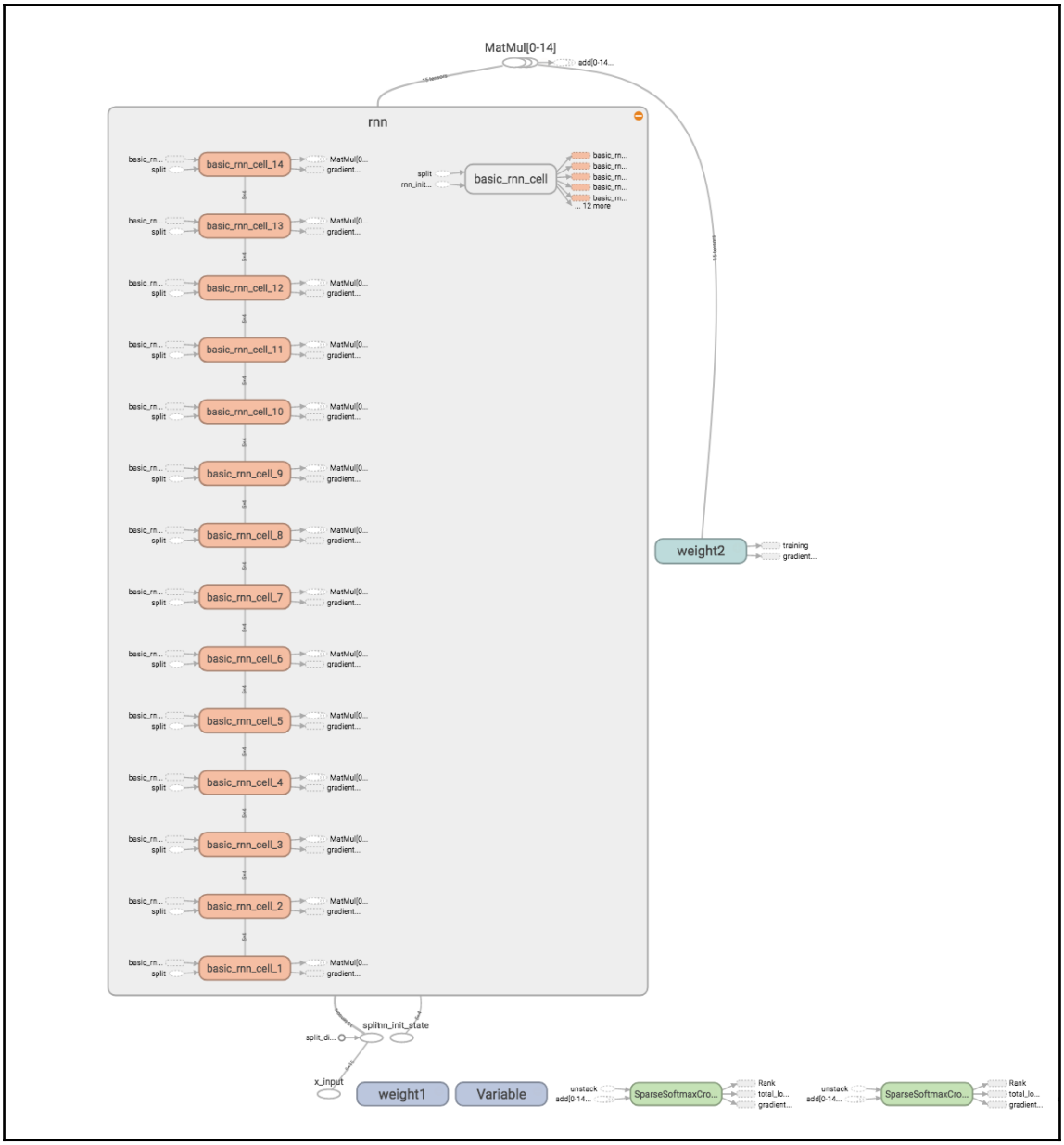


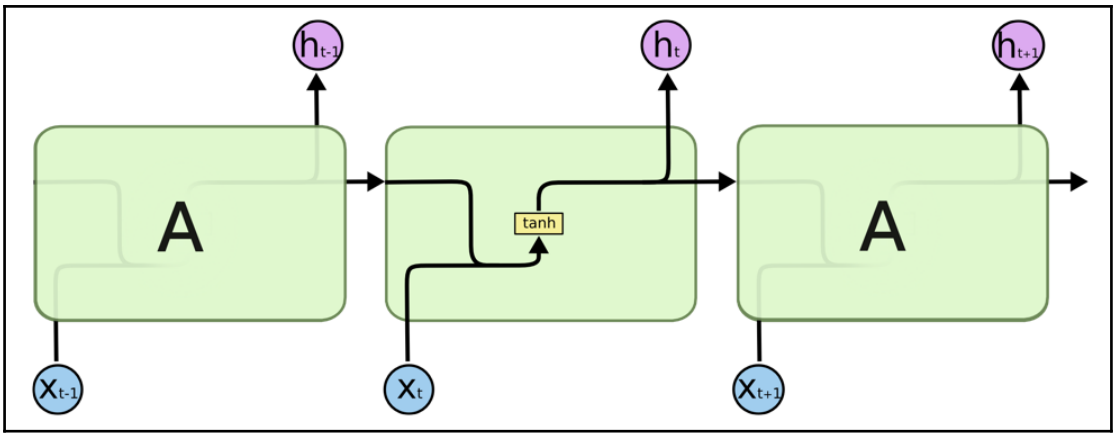


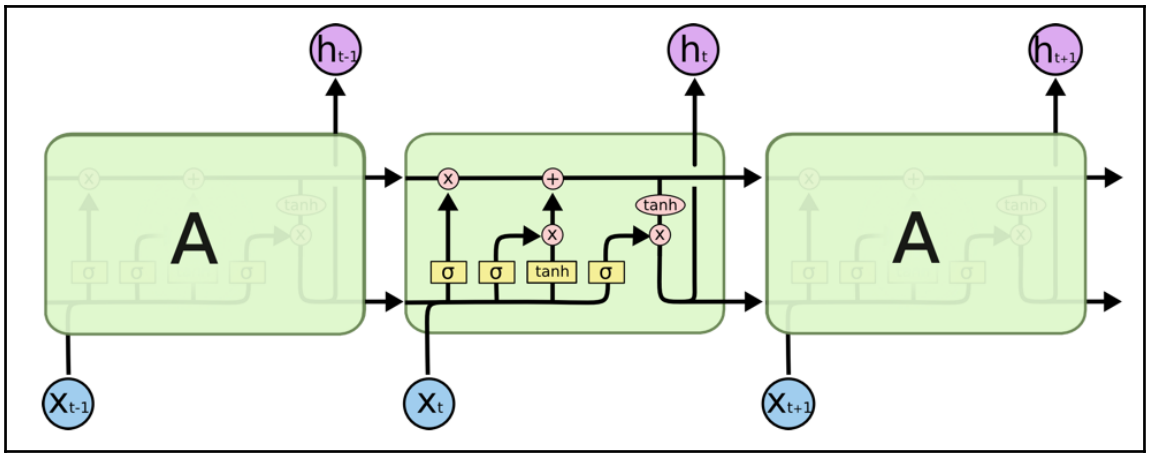


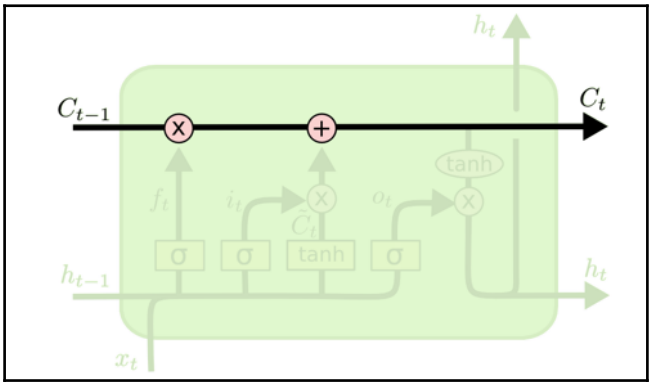

o
V
W
s
U
x
Unfold
o t−1
o t
o t+1
s t−1
s t
s t+1
x t−1
x t
x t+1

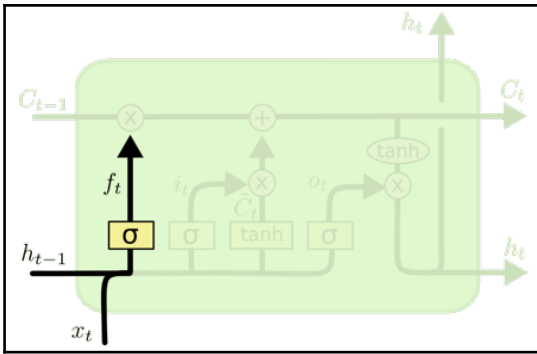

MatMul[0-29]
weight1
weight2
input_state_concat[0-14]
Assign
read
Adagrad
training
current_input[0-14]
rnn_init_state
next_state[0-14]
input_series
x_input
SparseSoftmaxCro...


MatMul[0-14]
add[0-14...
rnn
basic_rnn_cell_14
basic_rnn_cell_13
basic_rnn_cell_12
basic_rnn_cell_11
basic_rnn_cell_10
basic_rnn_cell_9
basic_rnn_cell_8
basic_rnn_cell_7
basic_rnn_cell_6
basic_rnn_cell_5
basic_rnn_cell_4
basic_rnn_cell_3
basic_rnn_cell_2
basic_rnn_cell_1
basic_rnn_cell
... 12 more
weight2
training
gradient...
rnn_init_state
x_input
weight1
Variable
SparseSoftmaxCro...

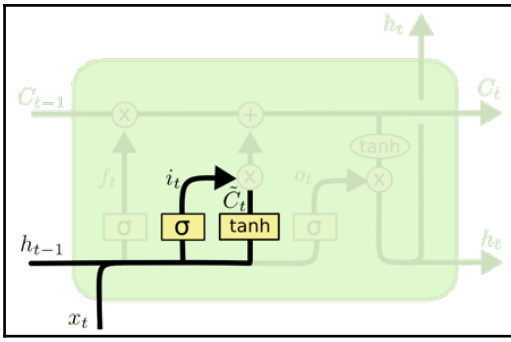

h_{t-1}
h_t
h_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}


h_{t-1}
h_t
h_{t+1}
A
σ
σ
tanh
σ
tanh
A
X_{t-1}
X_t
X_{t+1}

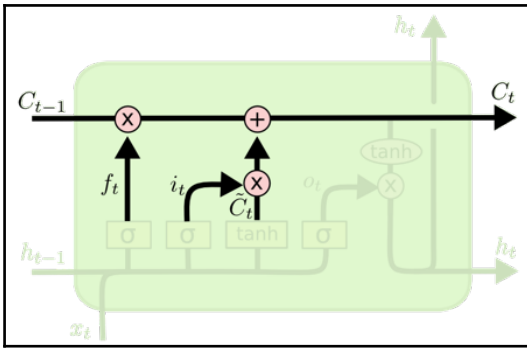

h_t
C_{t-1}
C_t
tanh
f_t
i_t
o_t
$\tilde{C}_t$
σ
σ
tanh
σ
h_{t-1}
h_t
x_t



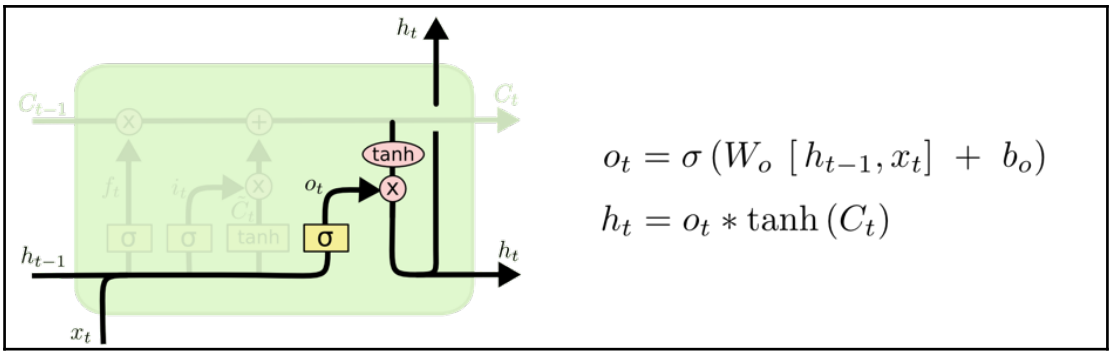
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

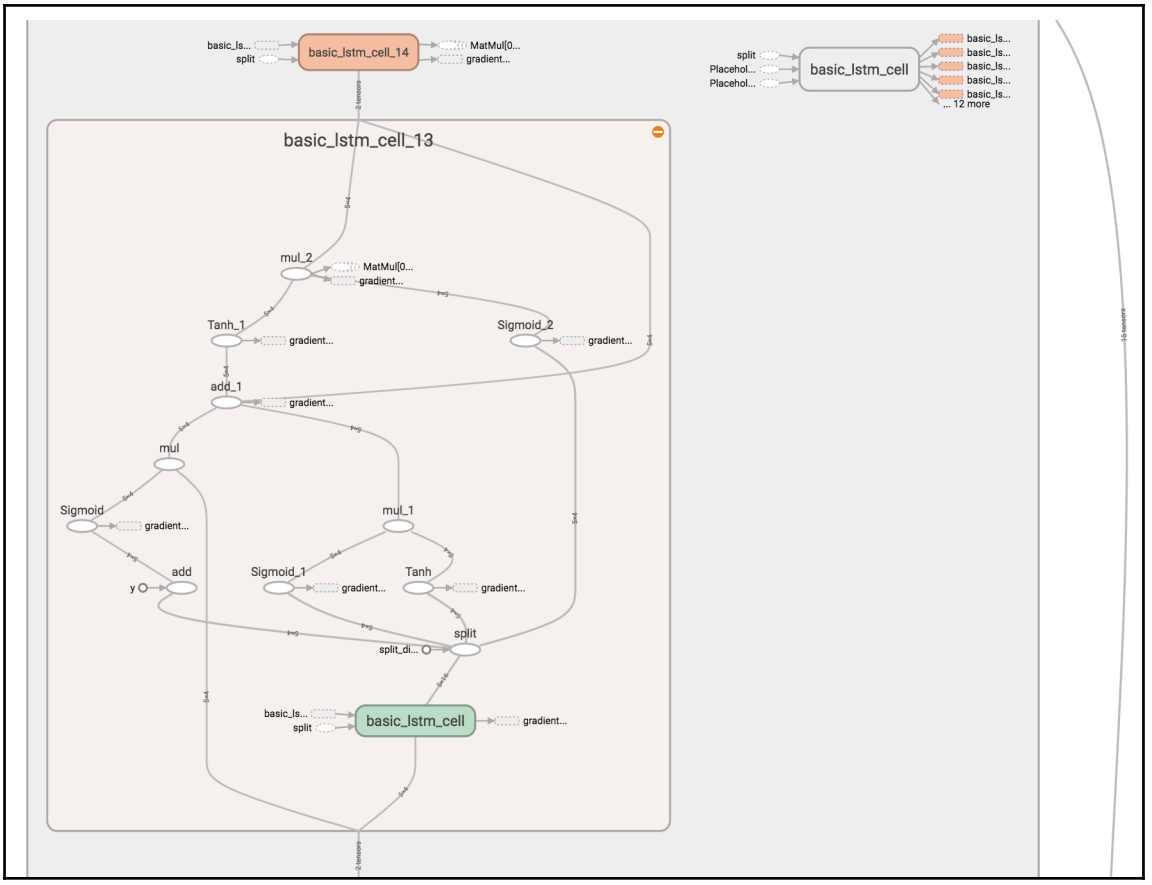


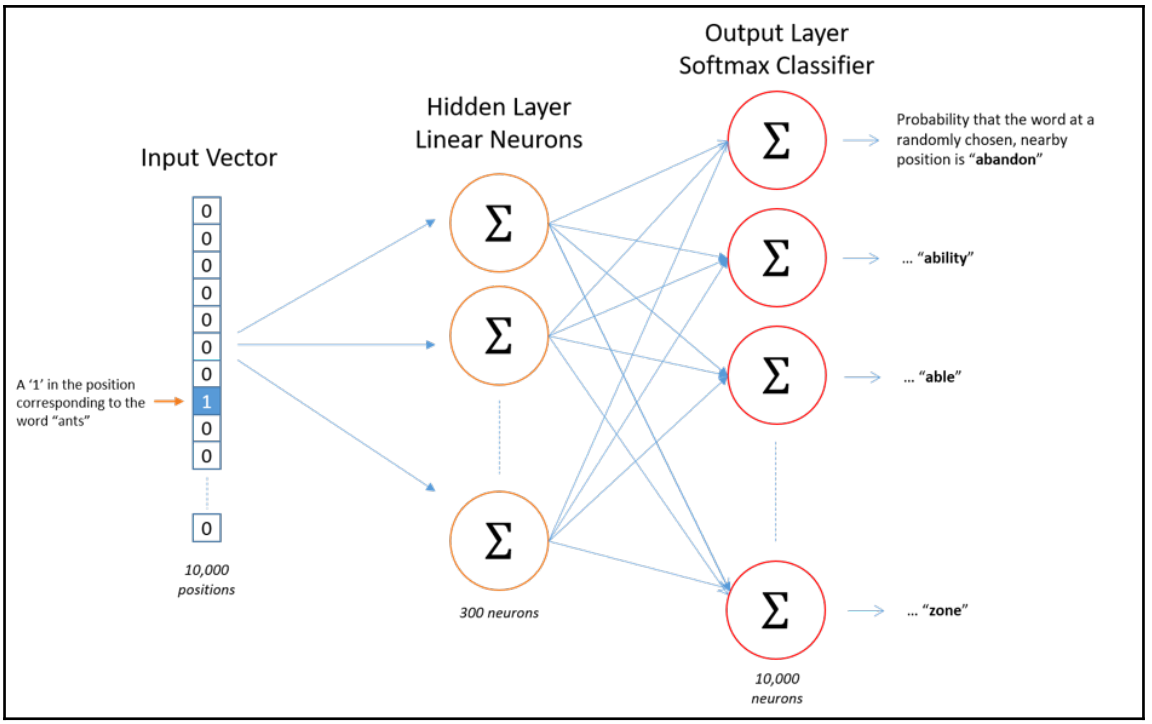
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

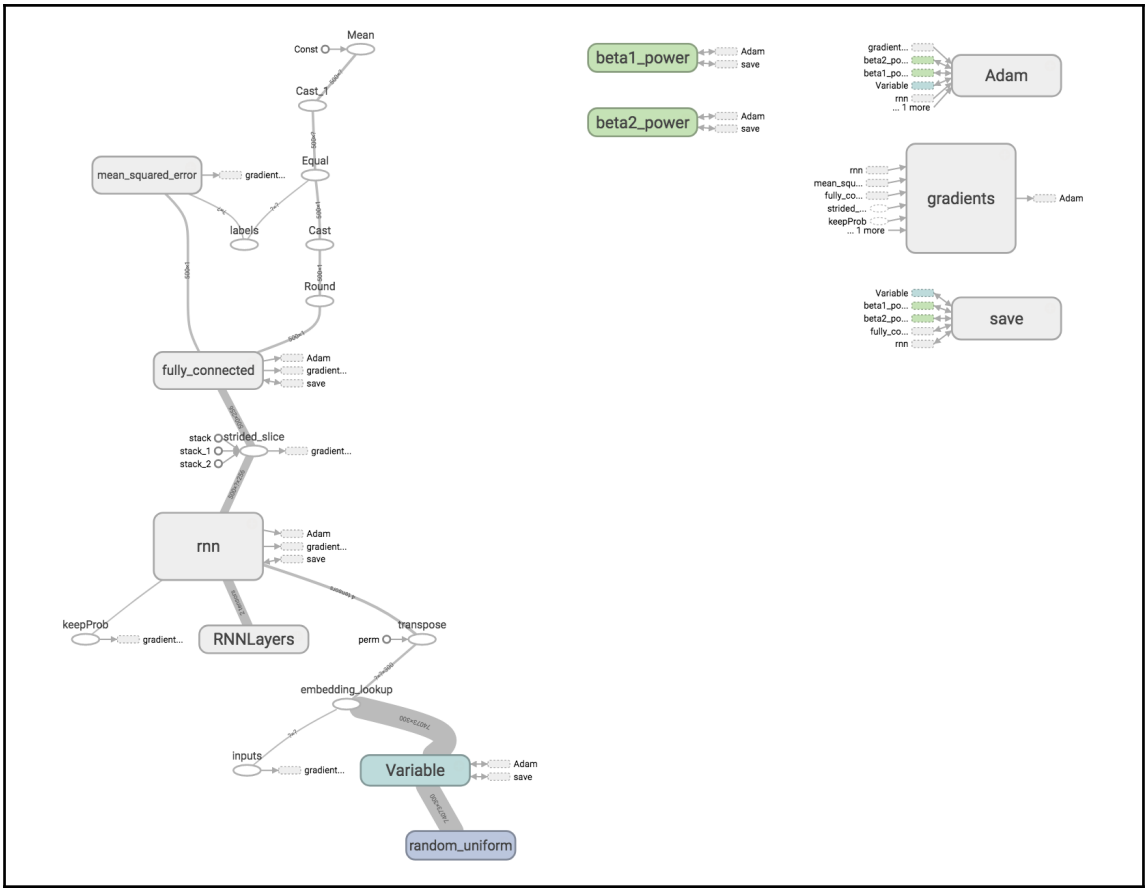


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

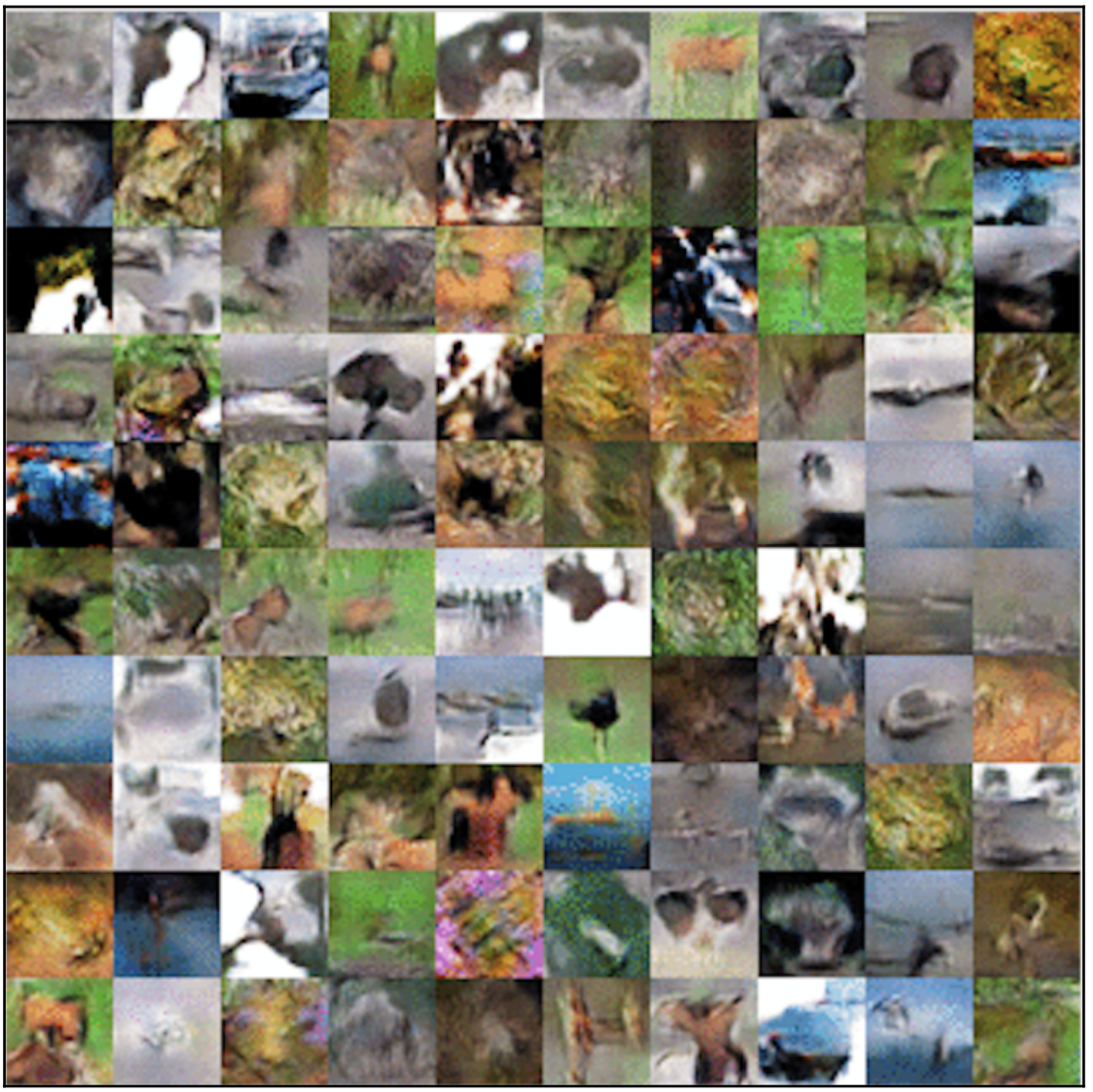

h_t
C_{t-1}
C_t
tanh
o_t
f_t
i_t
$\tilde{C}_t$
σ
σ
tanh
σ
h_{t-1}
h_t
x_t
$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$
$h_t = o_t * \tanh\left(C_t\right)$

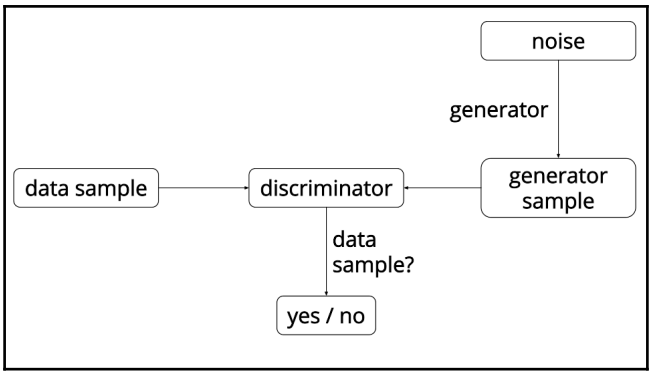

basic_ls...
split
basic_lstm_cell_14
MatMul[0...
gradient...
basic_lstm_cell_13
mul_2
MatMul[0...
gradient...
Tanh_1
gradient...
Sigmoid_2
gradient...
add_1
gradient...
mul
Sigmoid
gradient...
mul_1
add
y
Sigmoid_1
gradient...
Tanh
gradient...
split
split_di...
basic_ls...
split
basic_lstm_cell
gradient...
split
Placehol...
Placehol...
basic_lstm_cell
basic_ls...
basic_ls...
basic_ls...
basic_ls...
basic_ls...
... 12 more

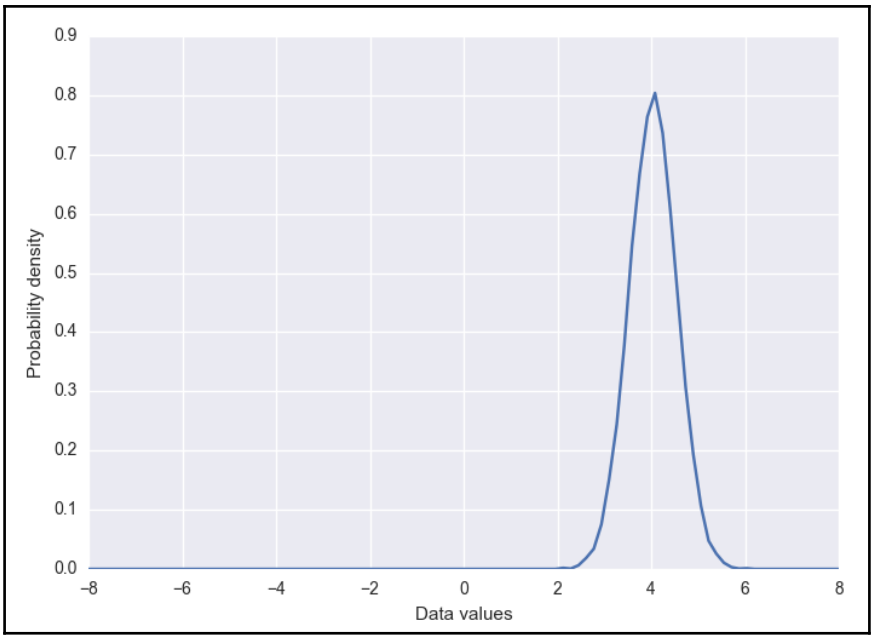

Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"
A '1' in the position corresponding to the word "ants"
0
0
0
0
0
0
0
1
0
0
0
10,000 positions
300 neurons
10,000 neurons

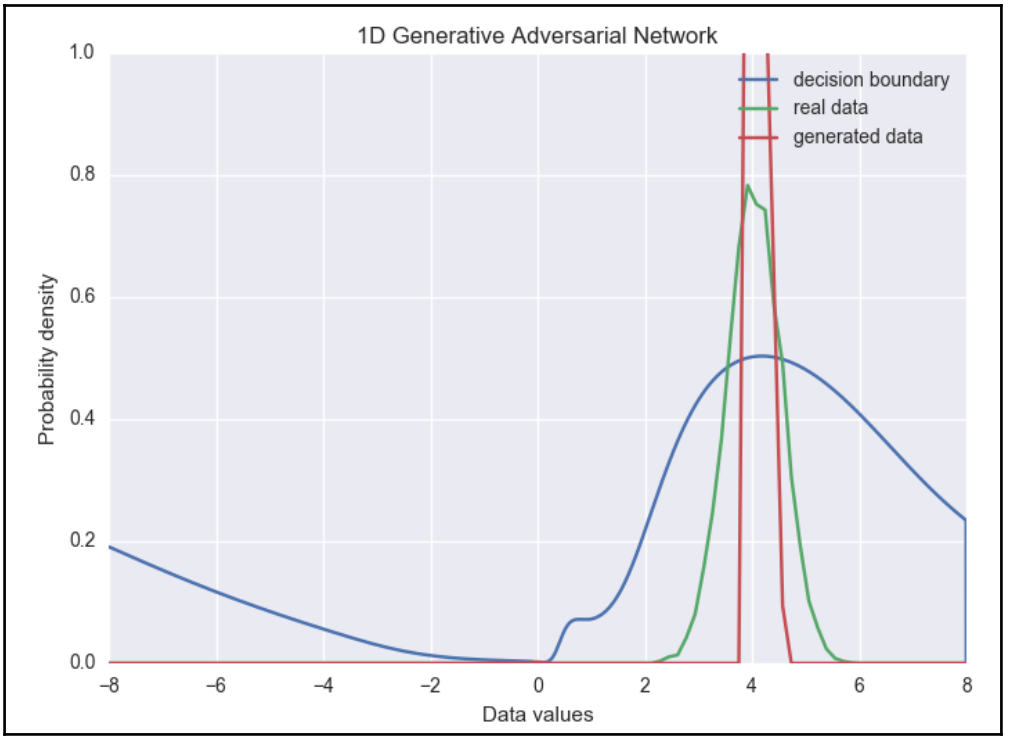

Mean
Const
Cast_1
Equal
mean_squared_error
gradient...
labels
Cast
Round
fully_connected
Adam
gradient...
save
stack
strided_slice
stack_1
stack_2
gradient...
rnn
Adam
gradient...
save
keepProb
gradient...
RNNLayers
transpose
perm
embedding_lookup
inputs
gradient...
Variable
Adam
save
random_uniform
beta1_power
Adam
save
beta2_power
Adam
save
gradient...
beta2_po...
beta1_po...
Variable
rnn
... 1 more
Adam
rnn
mean_squ...
fully_co...
strided_...
keepProb
... 1 more
gradients
Adam
Variable
beta1_po...
beta2_po...
fully_co...
rnn
save

Chapter 6: Generative Models

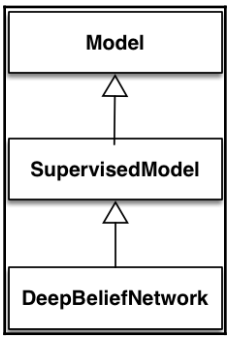


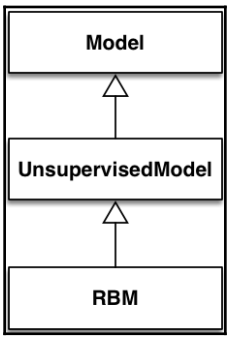

noise
generator
data sample
discriminator
generator sample
data sample?
yes / no

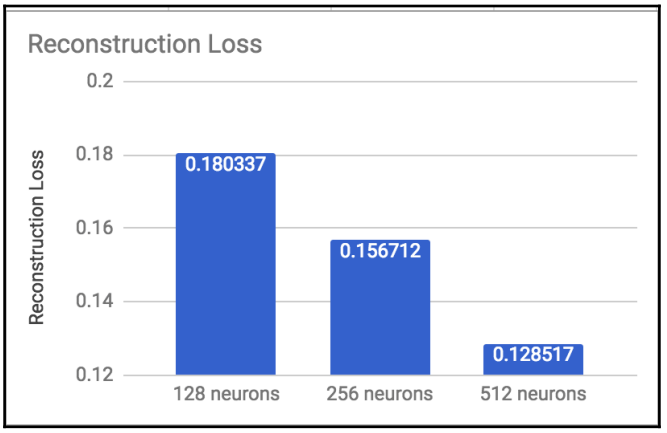

Probability density
Data values
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
−8
−6
−4
−2
0
2
4
6
8

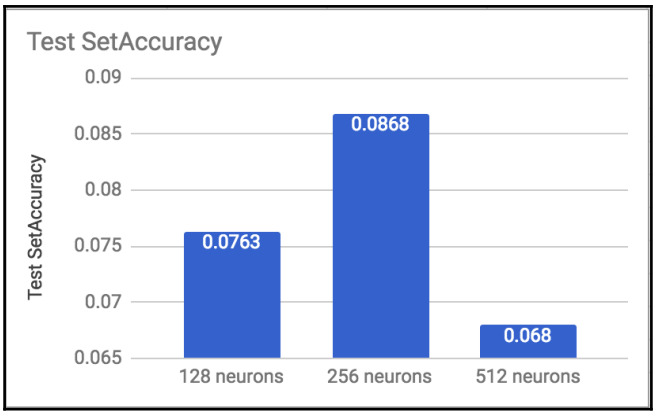

1D Generative Adversarial Network
decision boundary
real data
generated data
Probability density
Data values
0.0
0.2
0.4
0.6
0.8
1.0
−8
−6
−4
−2
0
2
4
6
8

Chapter 7: Deep Belief Networking

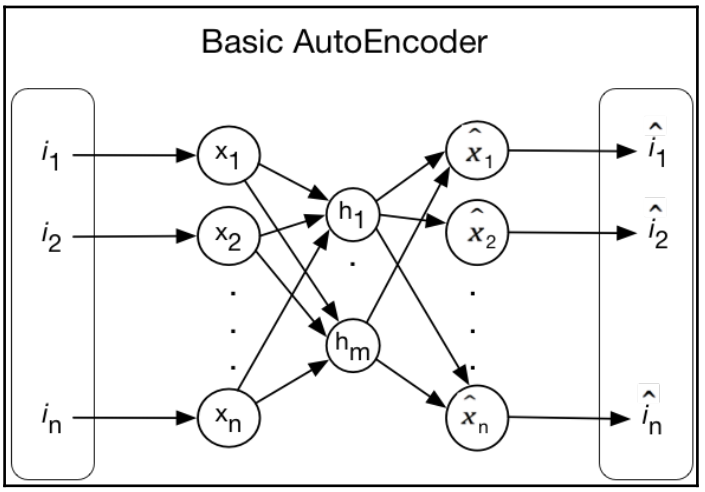


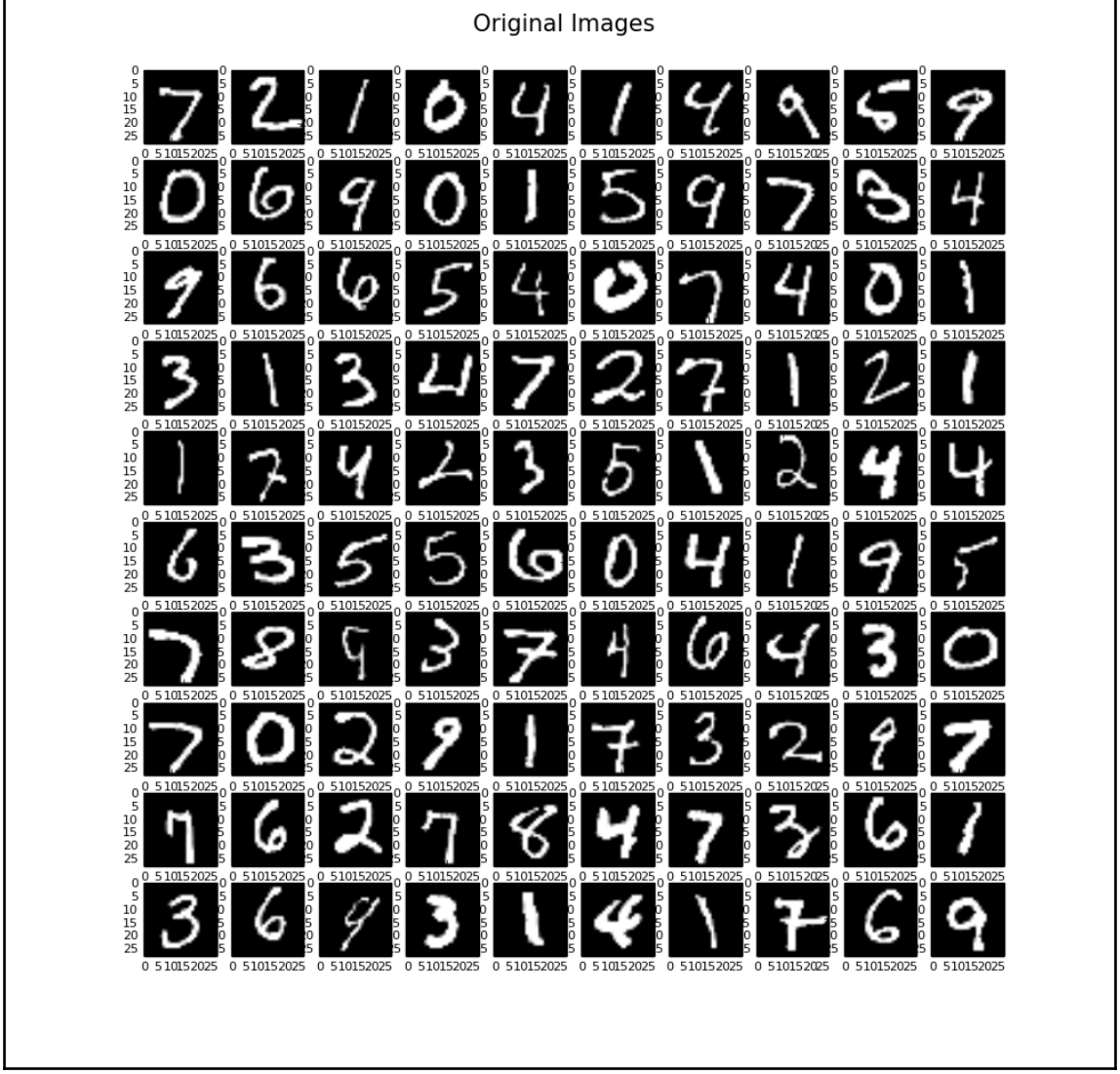


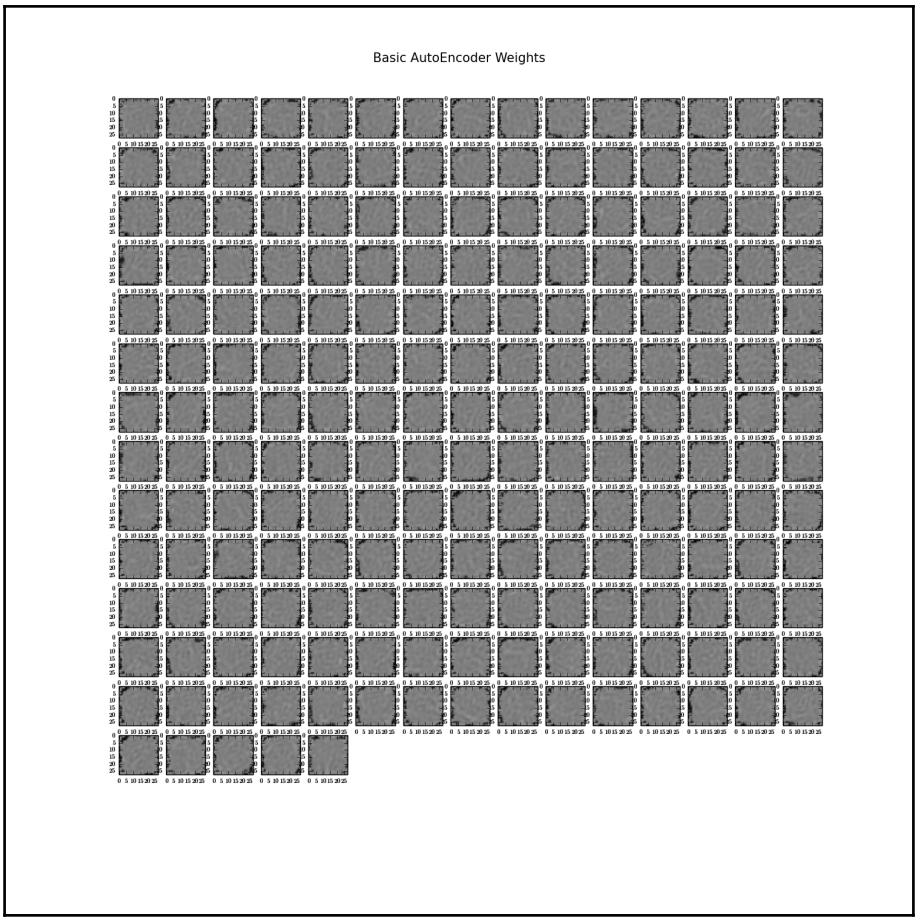


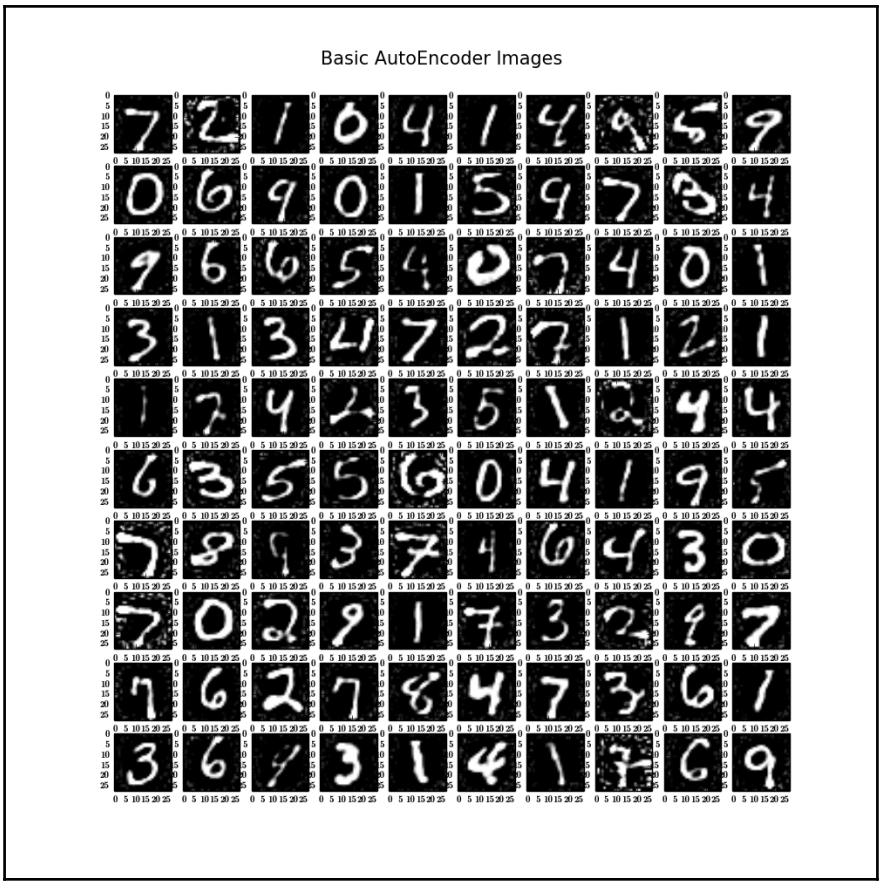

Test SetAccuracy
Test SetAccuracy
0.09
0.085
0.08
0.075
0.07
0.065
0.0763
0.0868
0.068
128 neurons
256 neurons
512 neurons

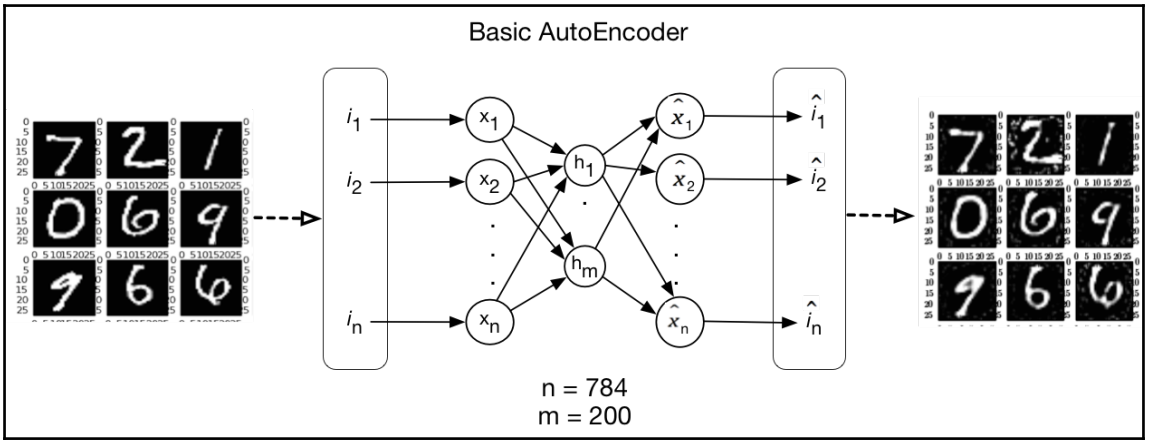
Chapter 8: Autoencoders

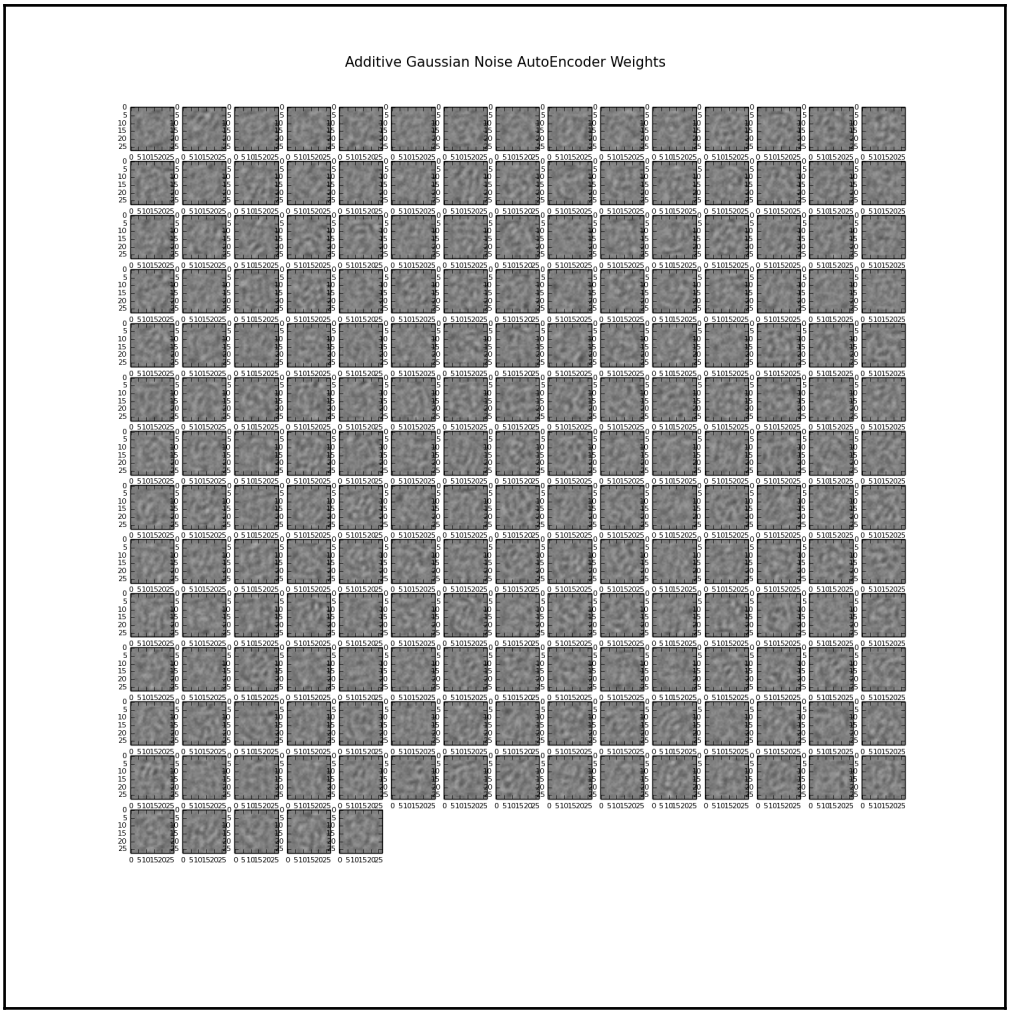


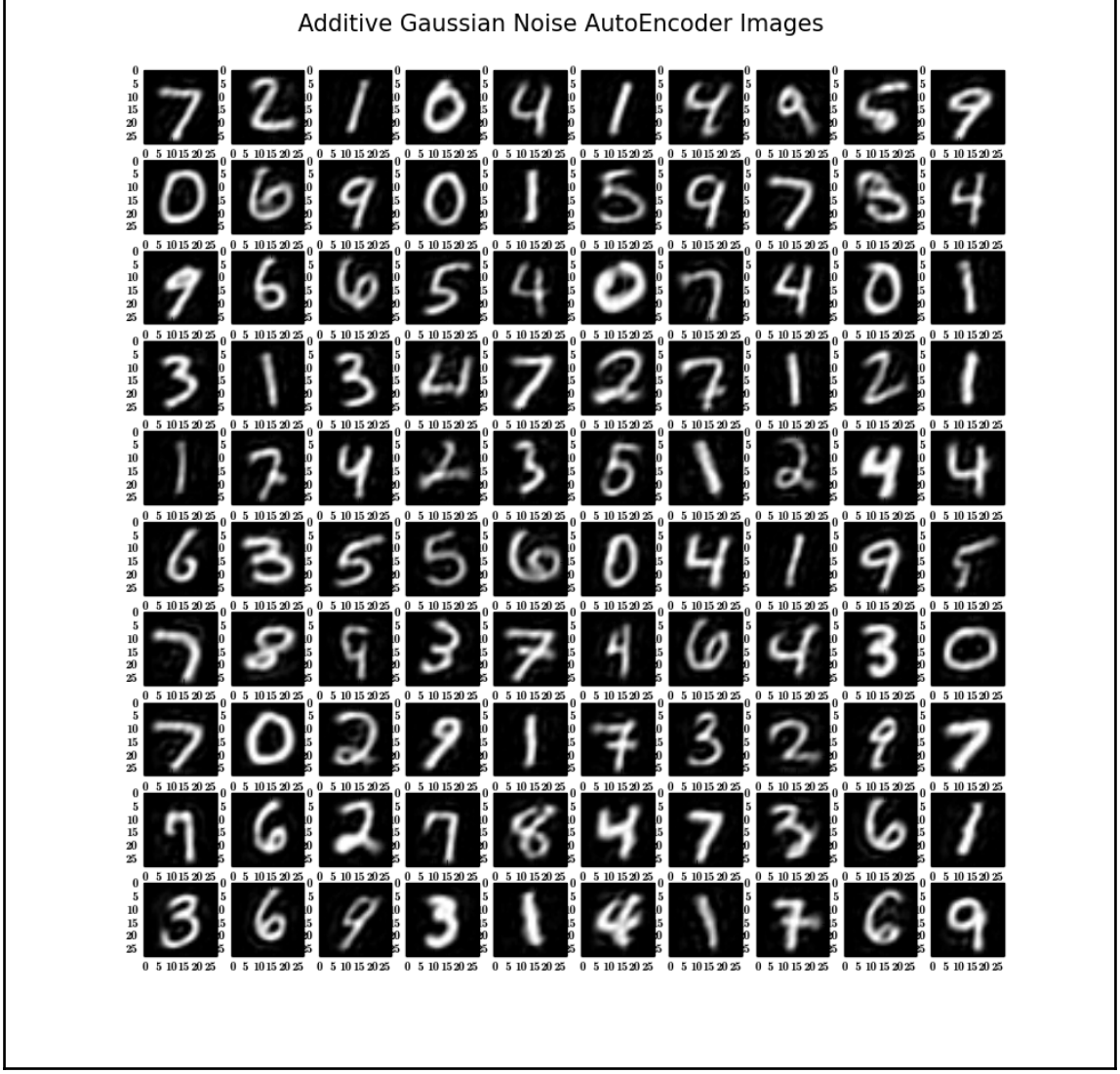

Original Images

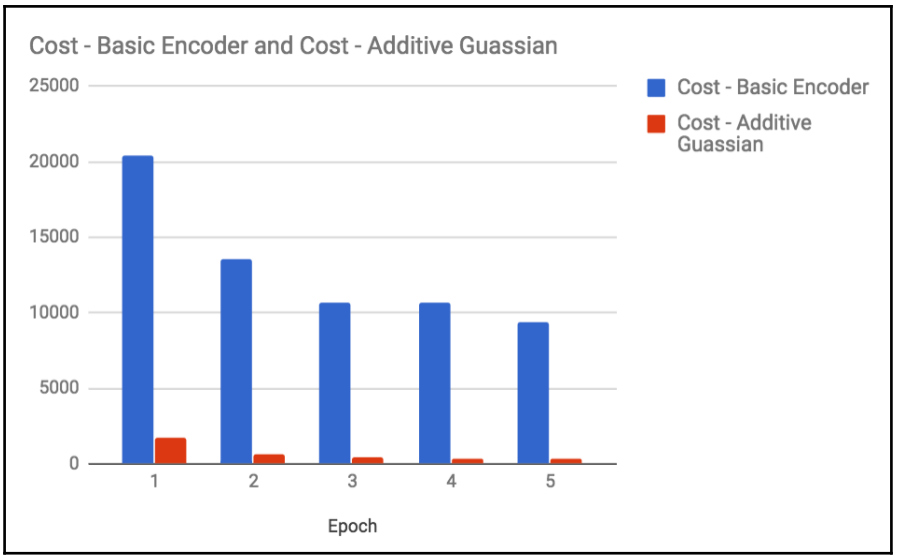

Basic AutoEncoder Weights

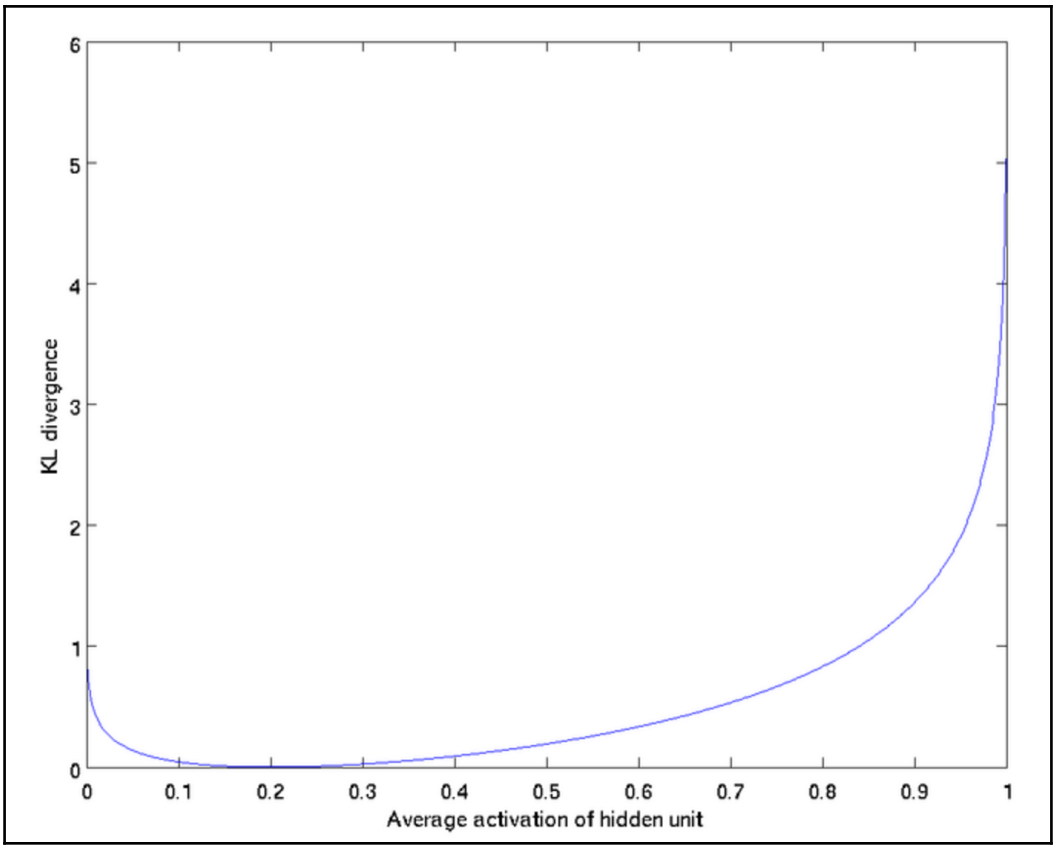

Basic AutoEncoder Images

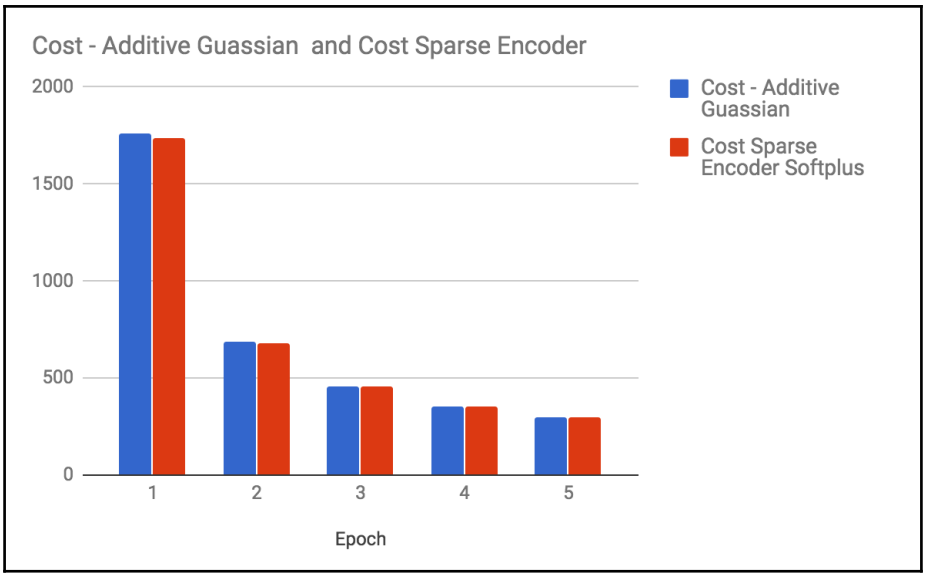

Basic AutoEncoder
i_1
i_2
i_n
x_1
x_2
x_n
h_1
h_m
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_n$
$\hat{i}_1$
$\hat{i}_2$
$\hat{i}_n$
n = 784
m = 200


Additive Gaussian Noise AutoEncoder Weights

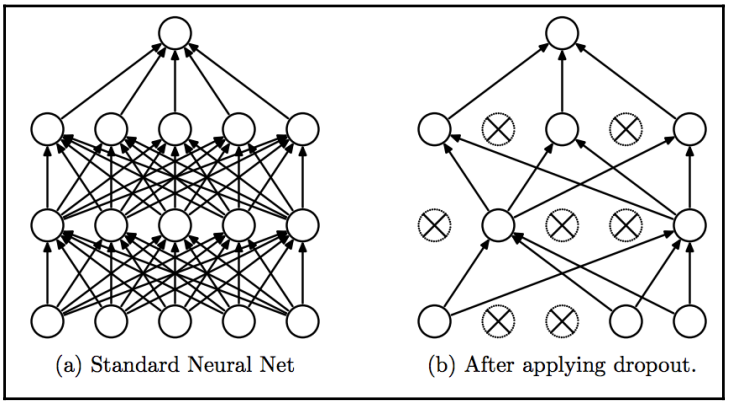

Additive Gaussian Noise AutoEncoder Images


Cost - Basic Encoder and Cost - Additive Guassian
25000
20000
15000
10000
5000
0
1
2
3
4
5
Epoch
Cost - Basic Encoder
Cost - Additive Guassian

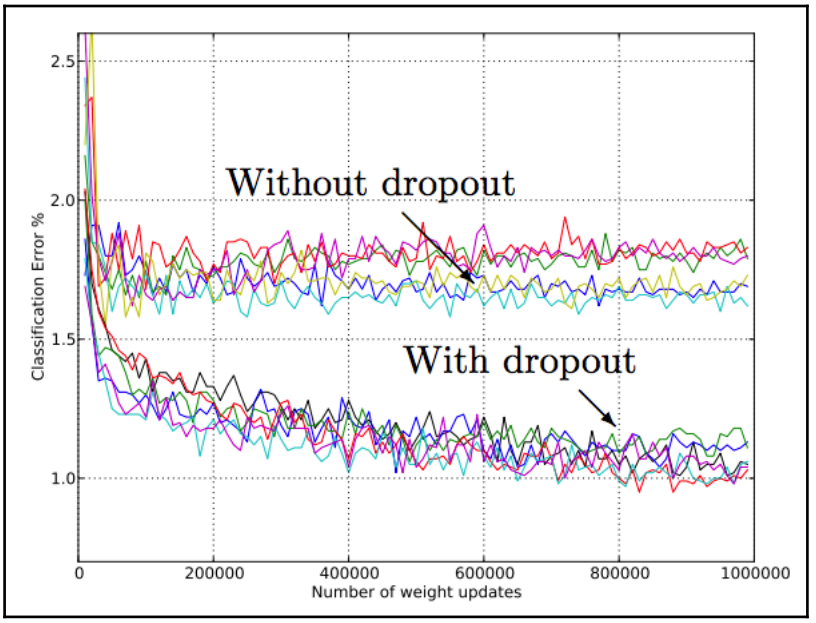

KL divergence
Average activation of hidden unit
0
1
2
3
4
5
6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

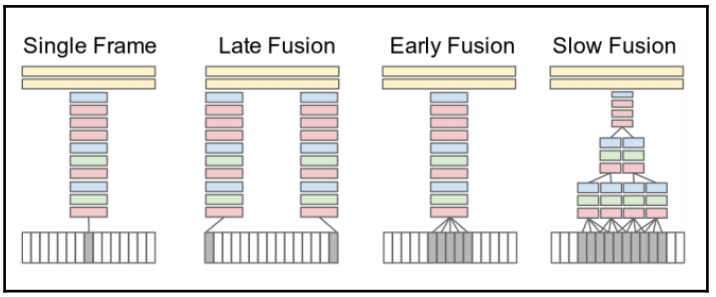

Cost - Additive Guassian and Cost Sparse Encoder
2000
1500
1000
500
0
1
2
3
4
5
Epoch
Cost - Additive Guassian
Cost Sparse Encoder Softplus

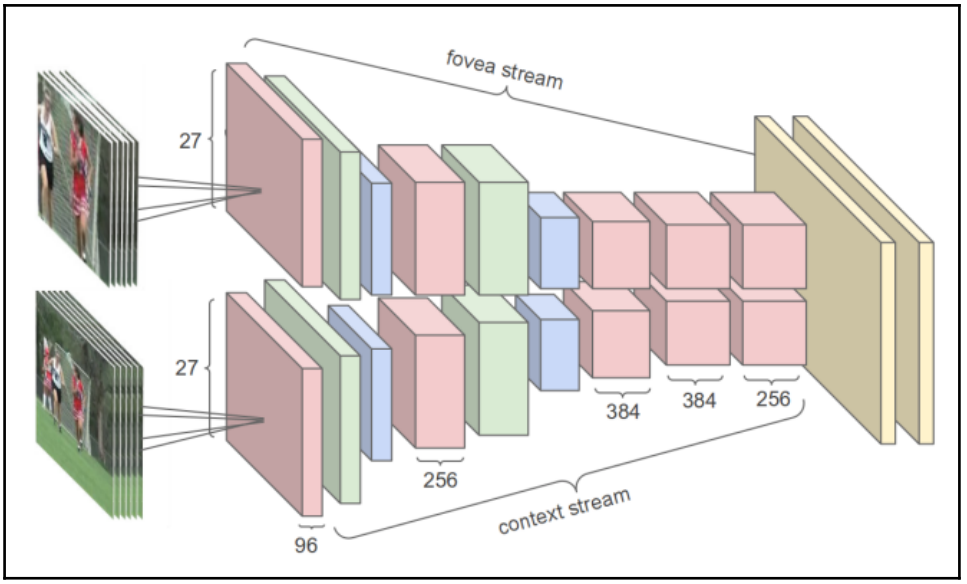
Chapter 9: Research in Neural Networks



(a) Standard Neural Net (b) After applying dropout.

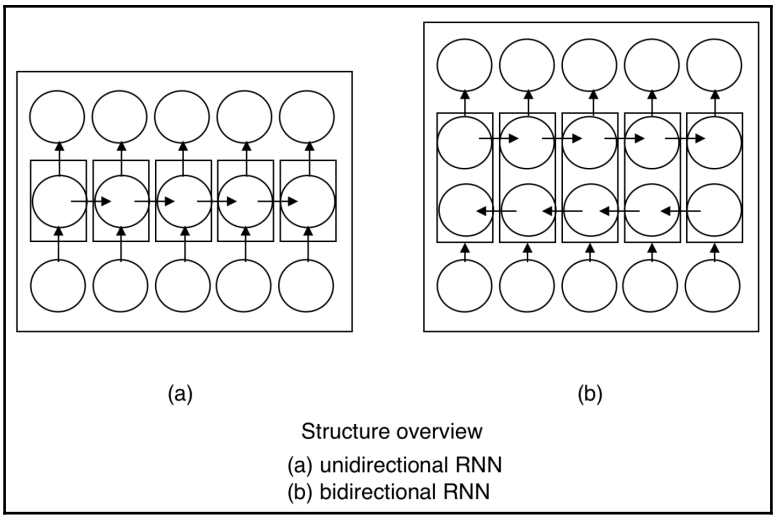




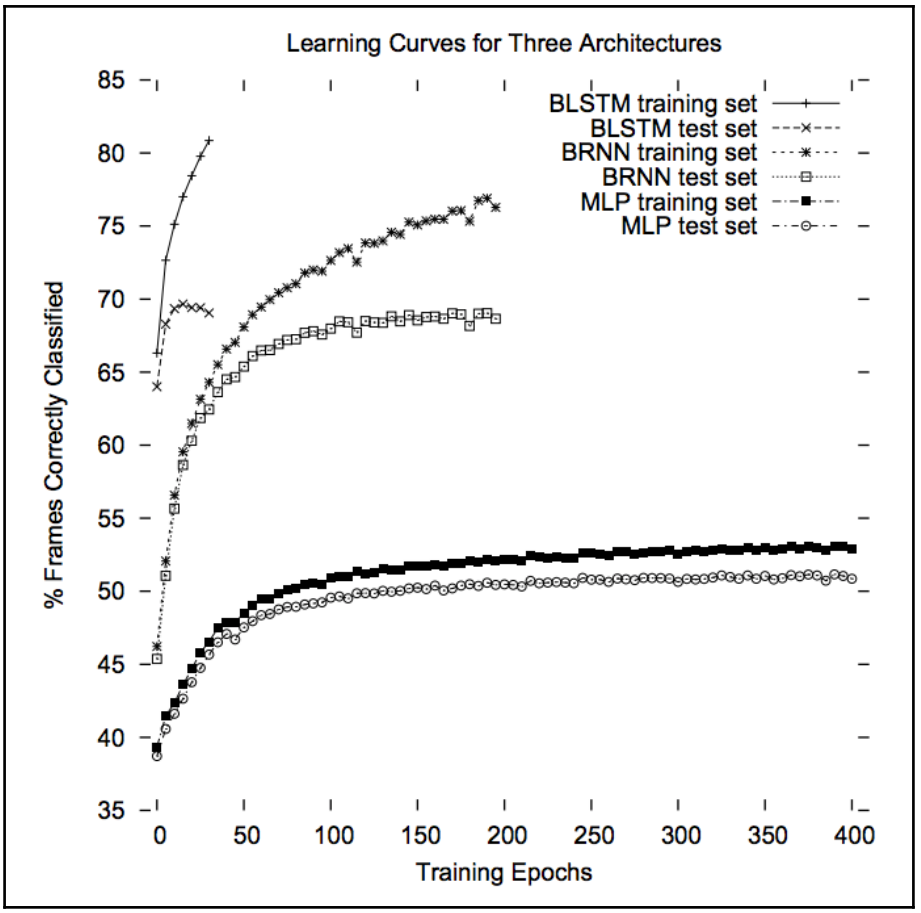


Model	Clip Hit@1	Video Hit@1	Video Hit@5
Feature Histograms + Neural Net	-	55.3	-
Single-Frame	41.1	59.3	77.7
Single-Frame + Multires	**42.4**	**60.0**	**78.5**
Single-Frame Fovea Only	30.0	49.9	72.8
Single-Frame Context Only	38.1	56.0	77.2
Early Fusion	38.9	57.7	76.8
Late Fusion	40.7	59.3	78.7
Slow Fusion	**41.9**	**60.9**	**80.2**
CNN Average (Single+Early+Late+Slow)	41.4	63.9	82.4

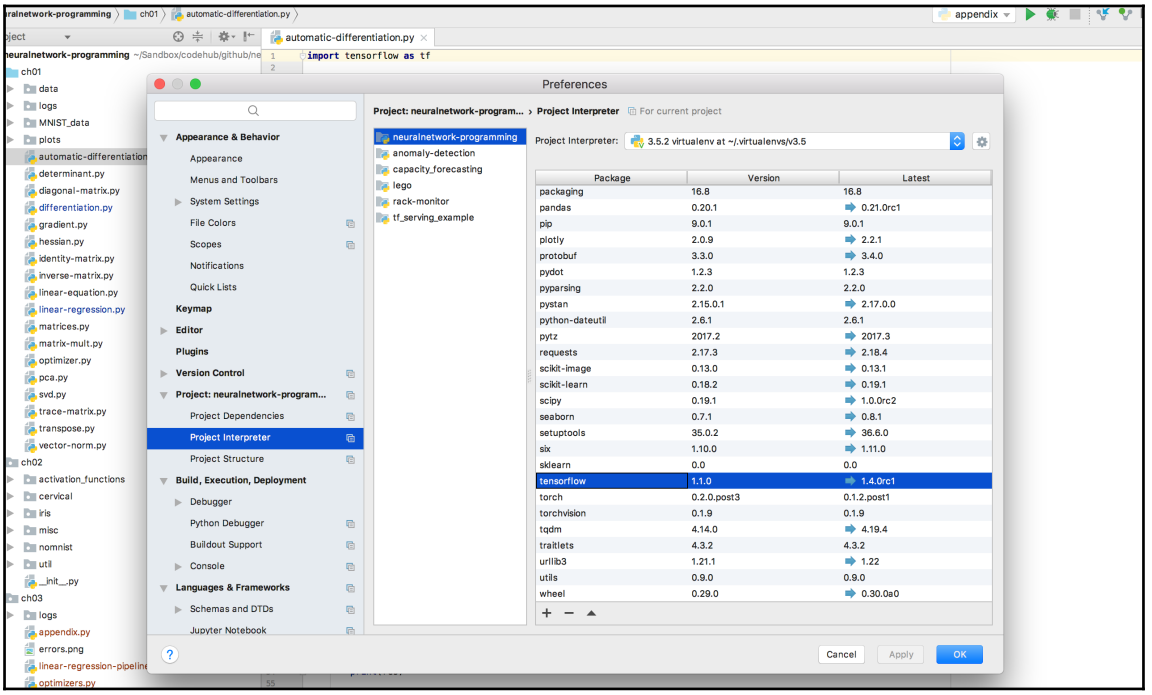
	P	R	F1
Stanford NER	84.04	80.86	82.42
Illinois NER	85.86	84.20	85.02
Twinet	86.06	86.34	**86.20**

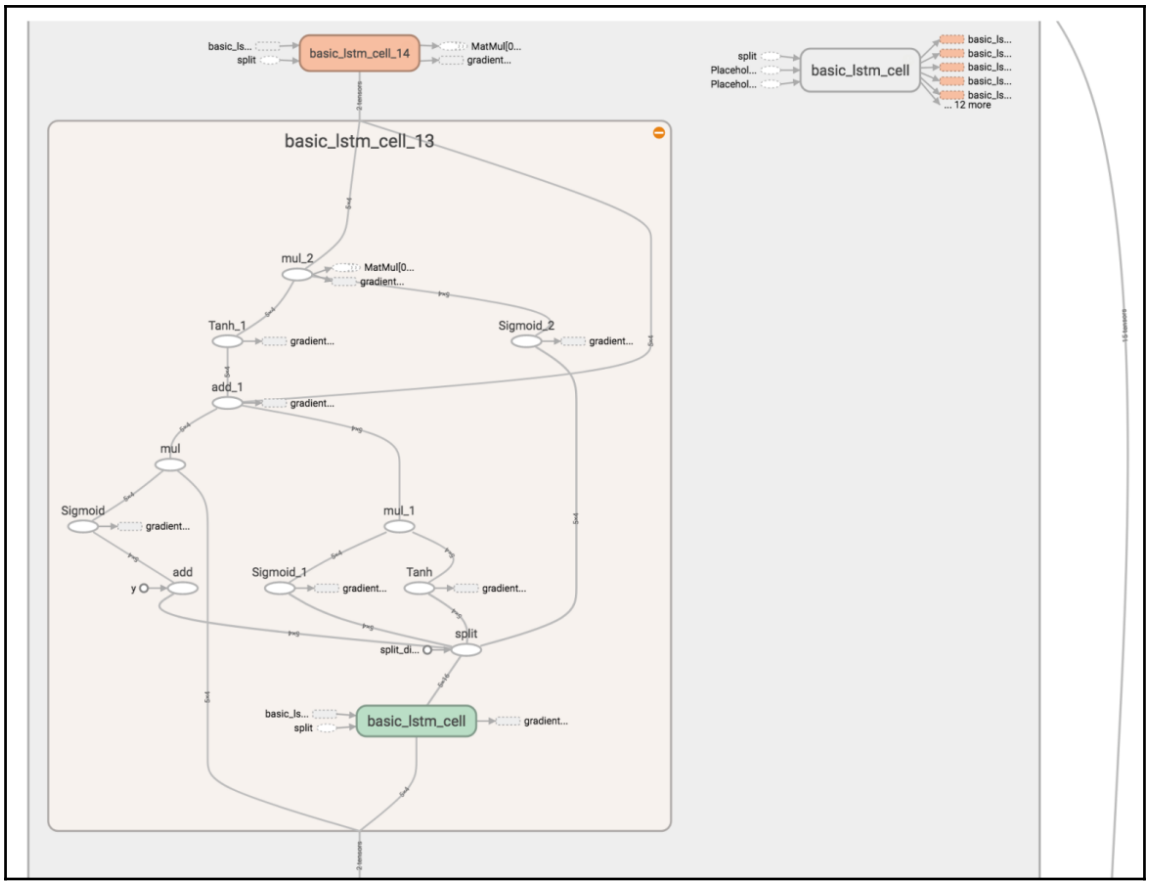


Structure overview
(a) unidirectional RNN
(b) bidirectional RNN


Learning Curves for Three Architectures
BLSTM training set
BLSTM test set
BRNN training set
BRNN test set
MLP training set
MLP test set
% Frames Correctly Classified
Training Epochs
85
80
75
70
65
60
55
50
45
40
35
0
50
100
150
200
250
300
350
400

Appendix : Getting started with TensorFlow




basic_ls...
split
basic_lstm_cell_14
MatMul[0...
gradient...
basic_lstm_cell_13
mul_2
MatMul[0...
gradient...
Tanh_1
gradient...
Sigmoid_2
gradient...
add_1
gradient...
mul
Sigmoid
gradient...
mul_1
add
y
Sigmoid_1
gradient...
Tanh
gradient...
split
split_di...
basic_ls...
split
basic_lstm_cell
gradient...
split
Placehol...
Placehol...
basic_lstm_cell
basic_ls...
basic_ls...
basic_ls...
basic_ls...
basic_ls...
... 12 more

Index