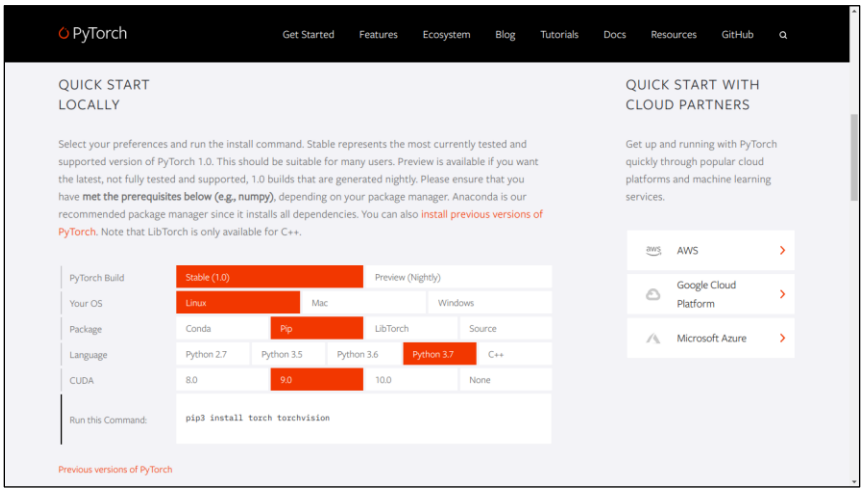
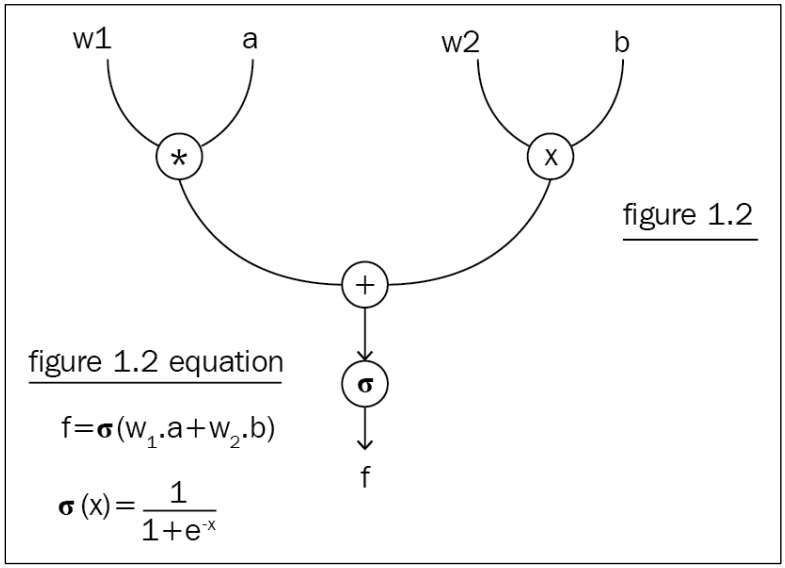
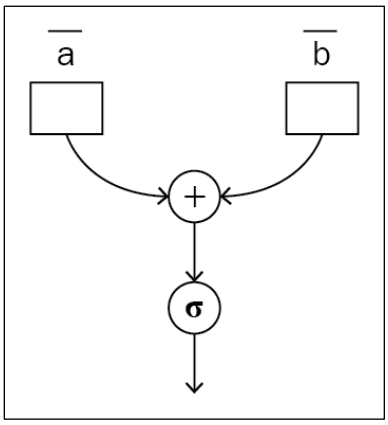
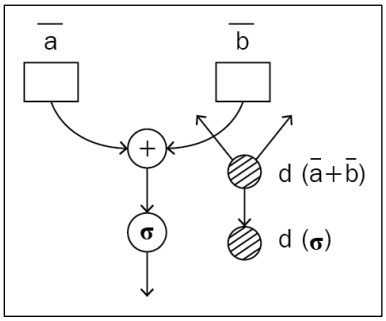
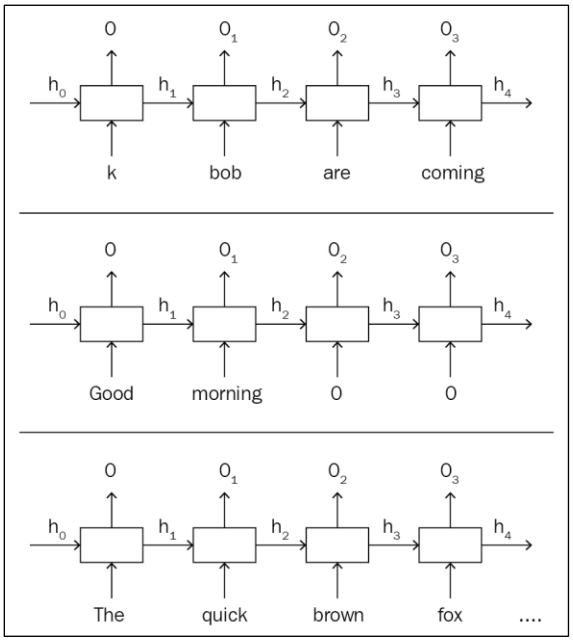
Chapter 1: Deep Learning Walkthrough and PyTorch Introduction

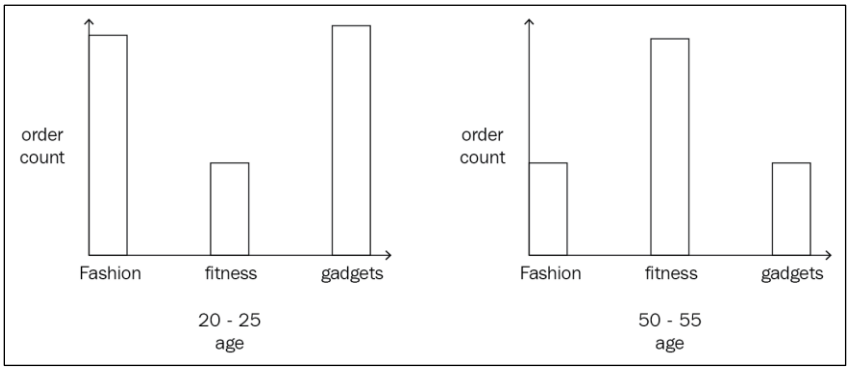


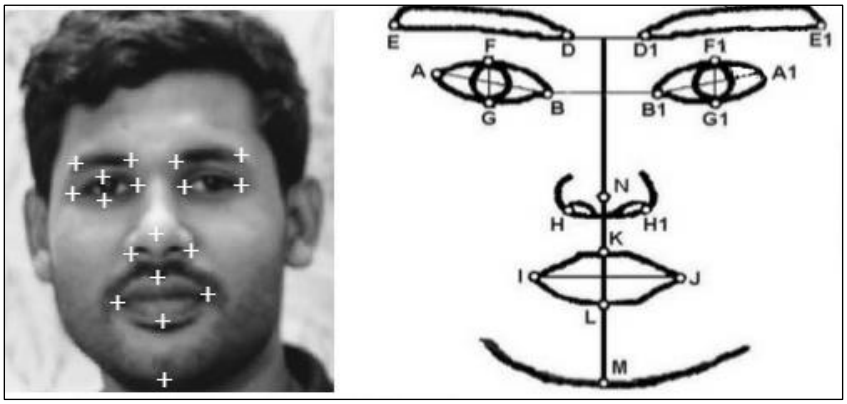


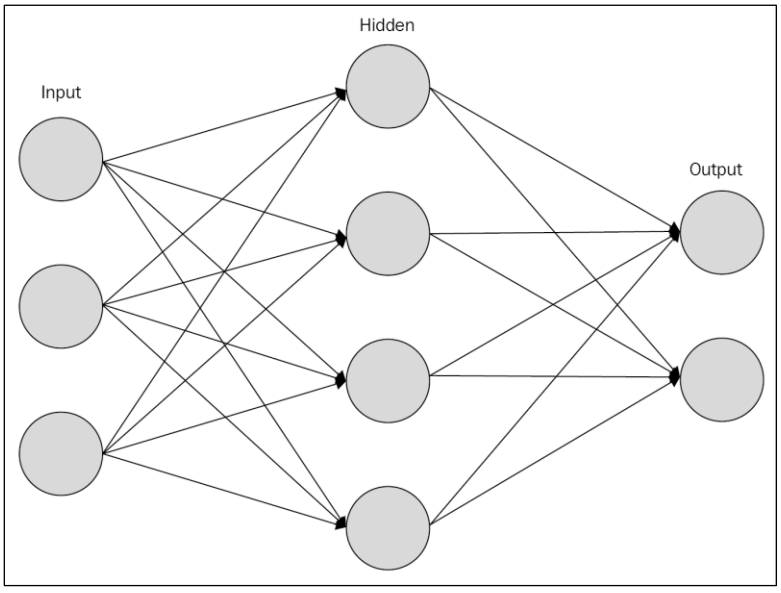

a
b
+
σ

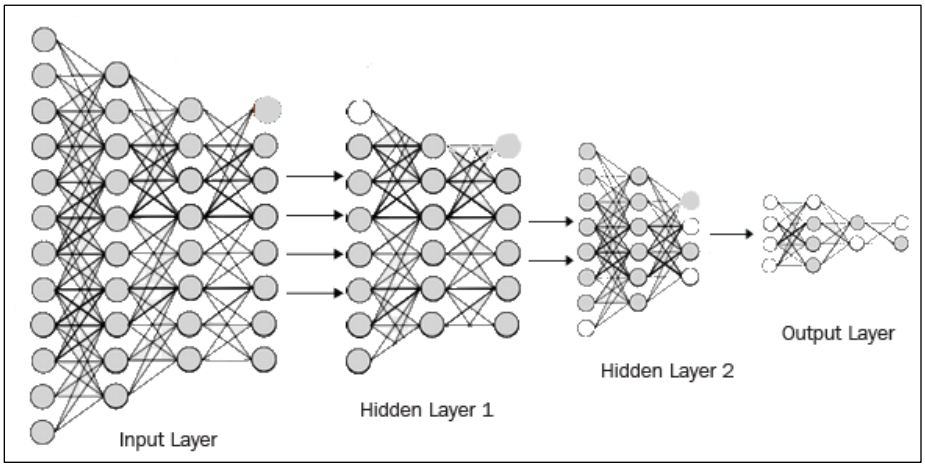

a
b
+
σ
d (a+b)
d (σ)

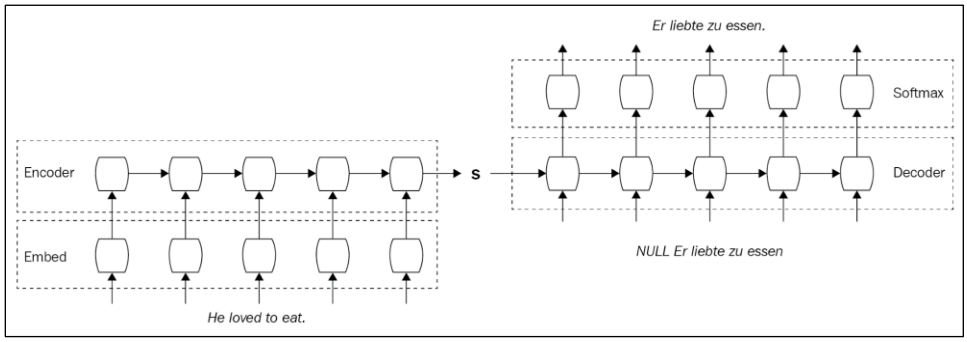

0
0 1
0 2
0 3
h0
h1
h2
h3
h4
k
bob
are
coming
Good
morning
0
0
The
quick
brown
fox
....

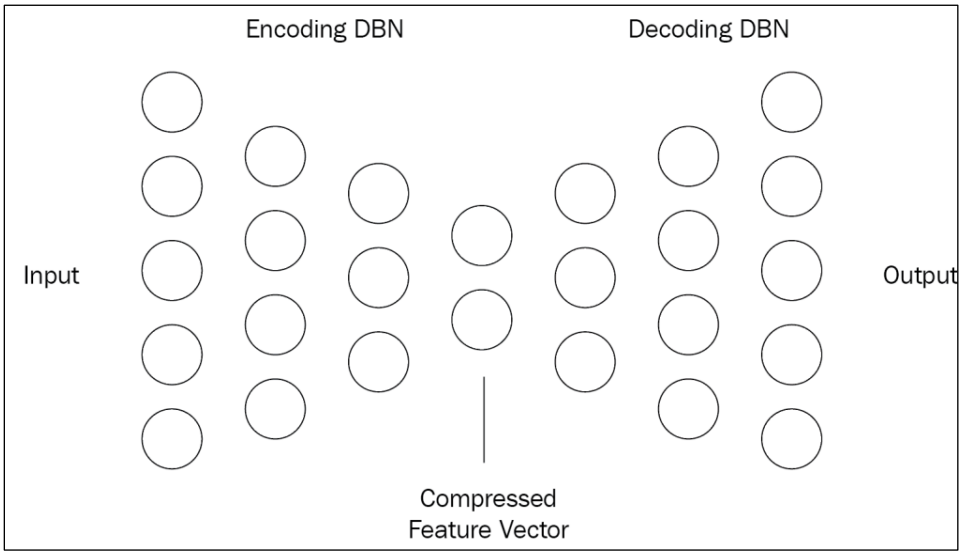

order
count
Fashion
fitness
gadgets
20 - 25
age
order
count
Fashion
fitness
gadgets
50 - 55
age

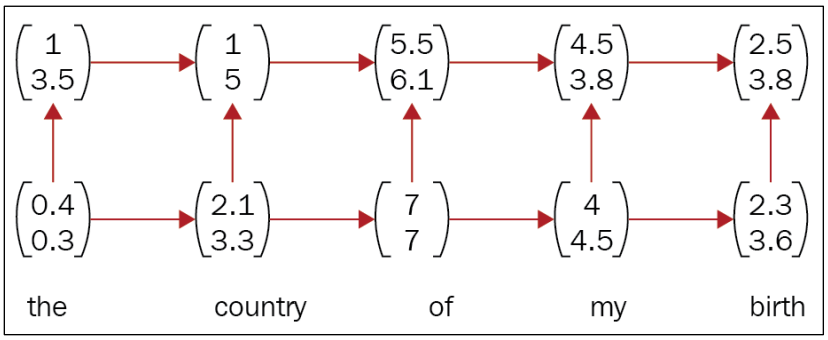

E
F
D
D1
F1
E1
A
A1
G
B
B1
G1
N
H
H1
K
I
J
L
M

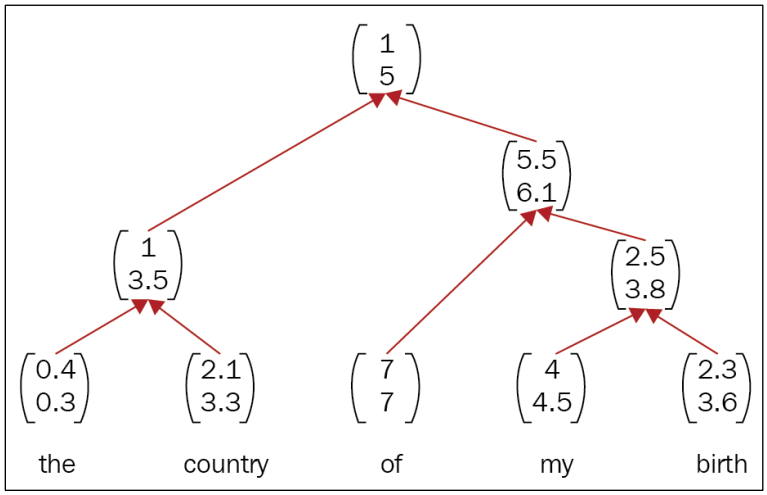

Hidden
Input
Output

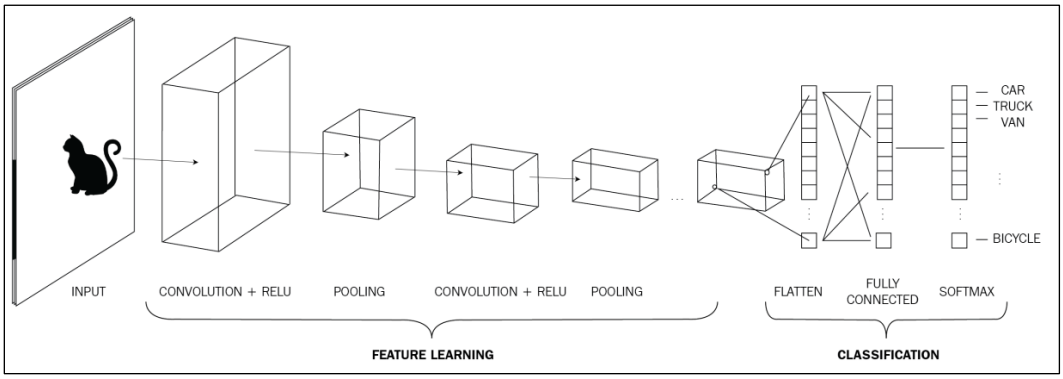

Input Layer
Hidden Layer 1
Hidden Layer 2
Output Layer

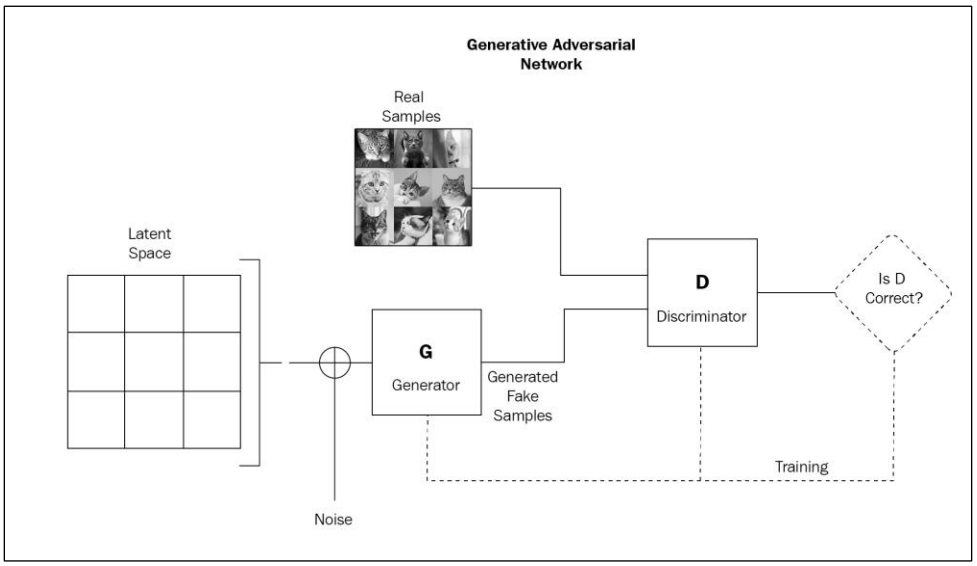

Er liebte zu essen.
Softmax
Encoder
s
Decoder
Embed
NULL Er liebte zu essen
He loved to eat.

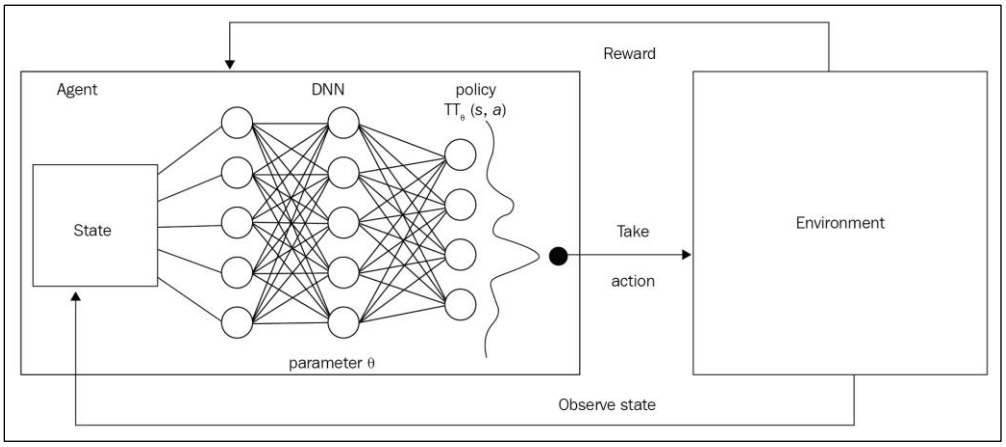

Encoding DBN
Decoding DBN
Input
Output
Compressed
Feature Vector

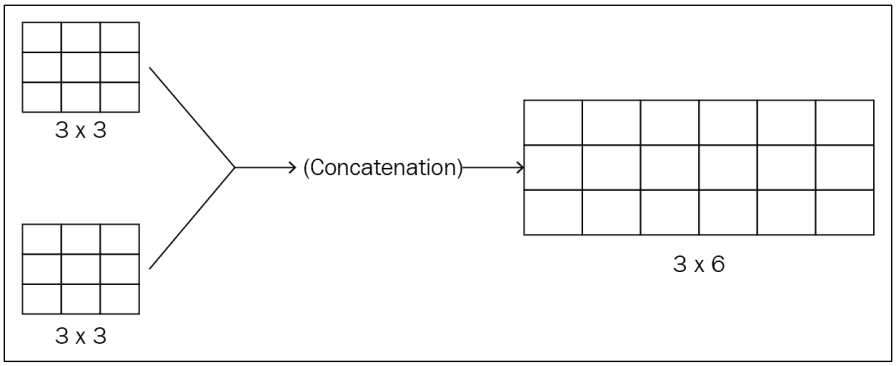

1
3.5
1
5
5.5
6.1
4.5
3.8
2.5
3.8
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth

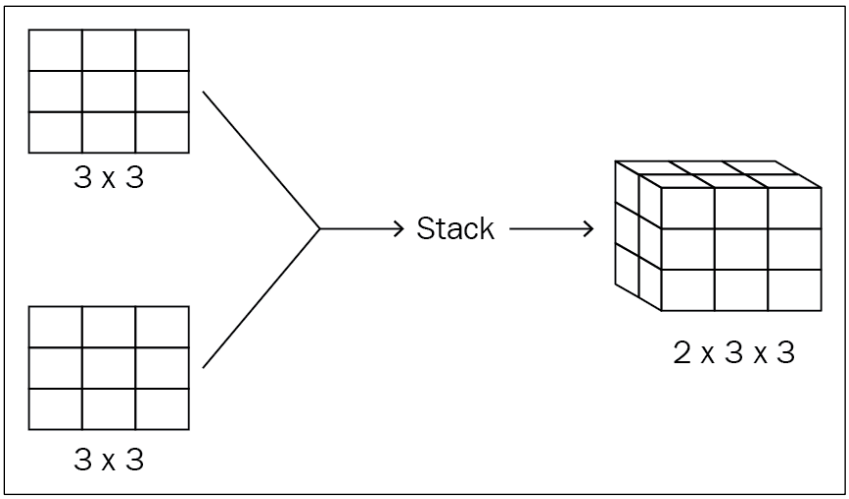

1
5
5.5
6.1
1
3.5
2.5
3.8
0.4
0.3
2.1
3.3
7
7
4
4.5
2.3
3.6
the
country
of
my
birth

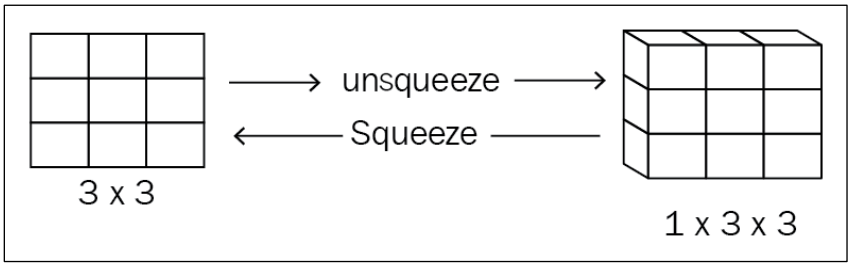

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

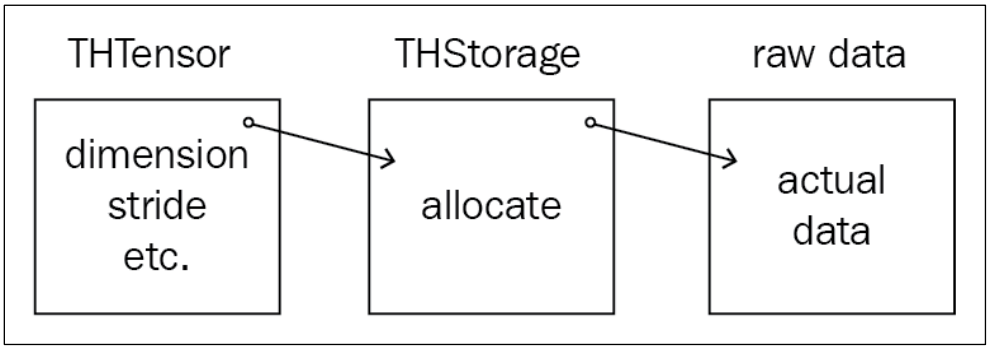

Generative Adversarial Network
Real Samples
Latent Space
D
Discriminator
Is D Correct?
G
Generator
Generated Fake Samples
Training
Noise


Reward
Agent
DNN
policy
TT_θ (s, a)
State
Take
action
Environment
parameter θ
Observe state

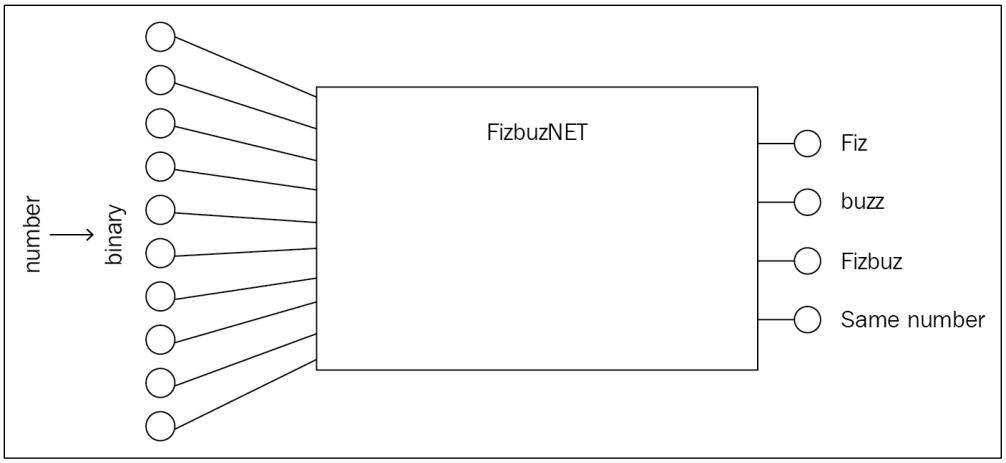

3 x 3
(Concatenation)
3 x 6
3 x 3


3 x 3
Stack
2 x 3 x 3
3 x 3


unsqueeze
Squeeze
3 x 3
1 x 3 x 3

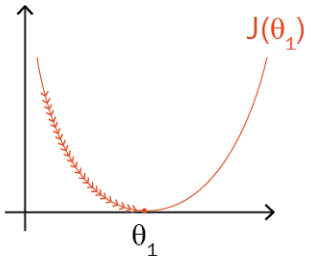

THTensor
THStorage
raw data
dimension
stride
etc.
allocate
actual
data

Chapter 2: A Simple Neural Network

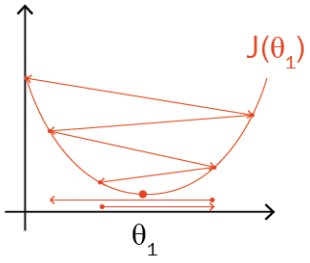


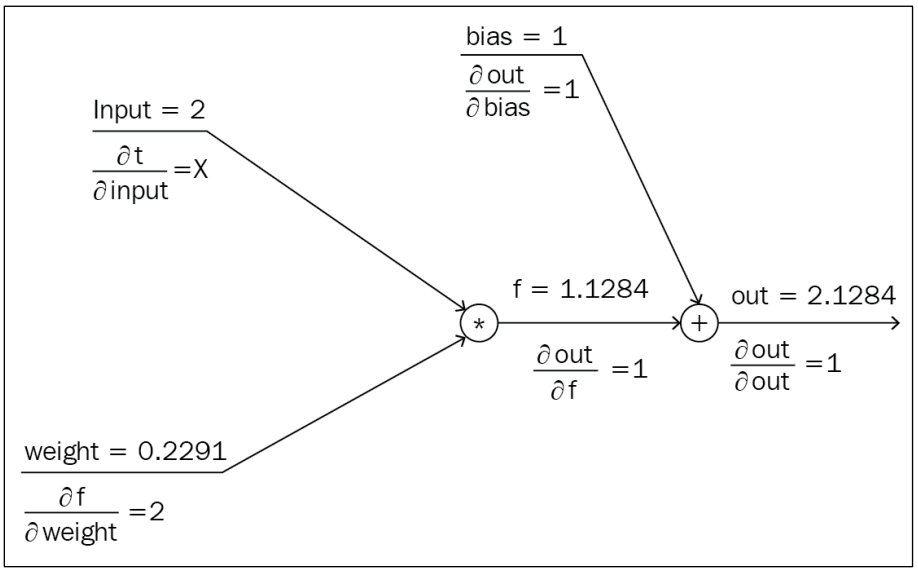
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

if α is too small, gradient decent can be slow.

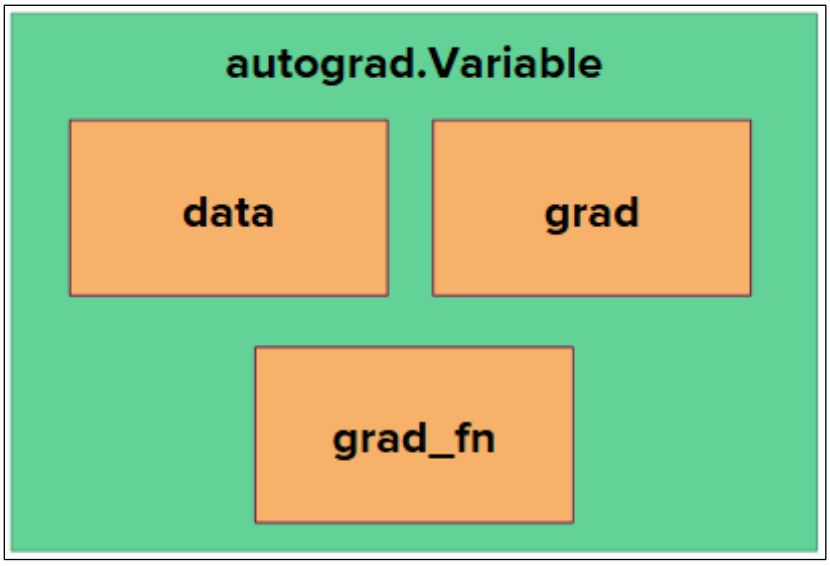


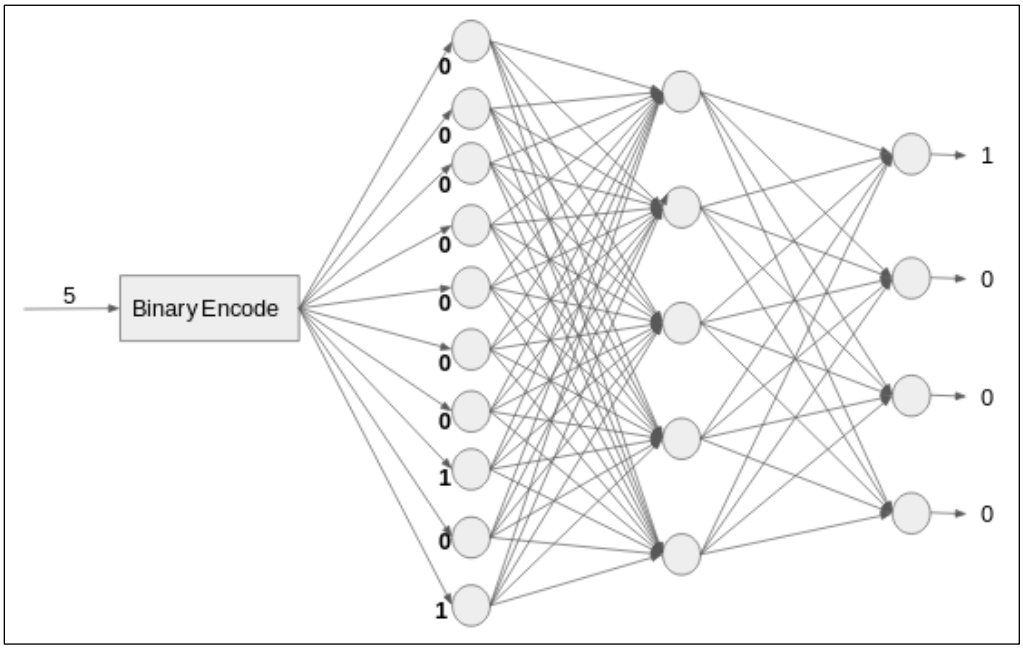
if α is too large, gradient decent can overshoot the minimum. It may fail to converge, or even diverge.

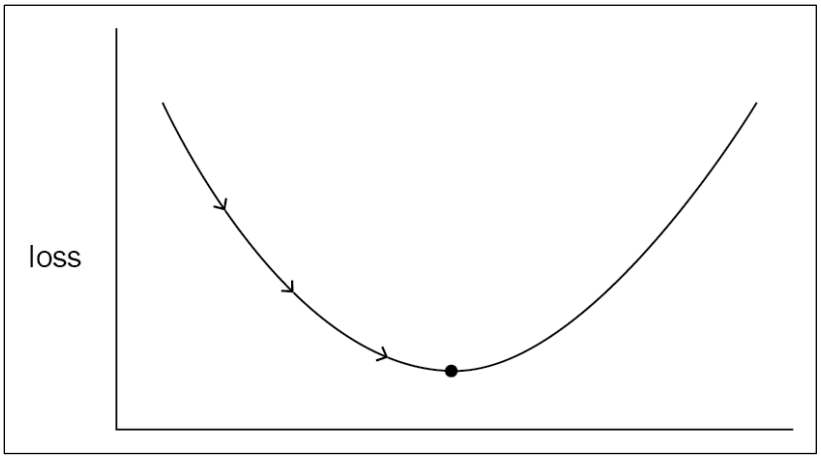


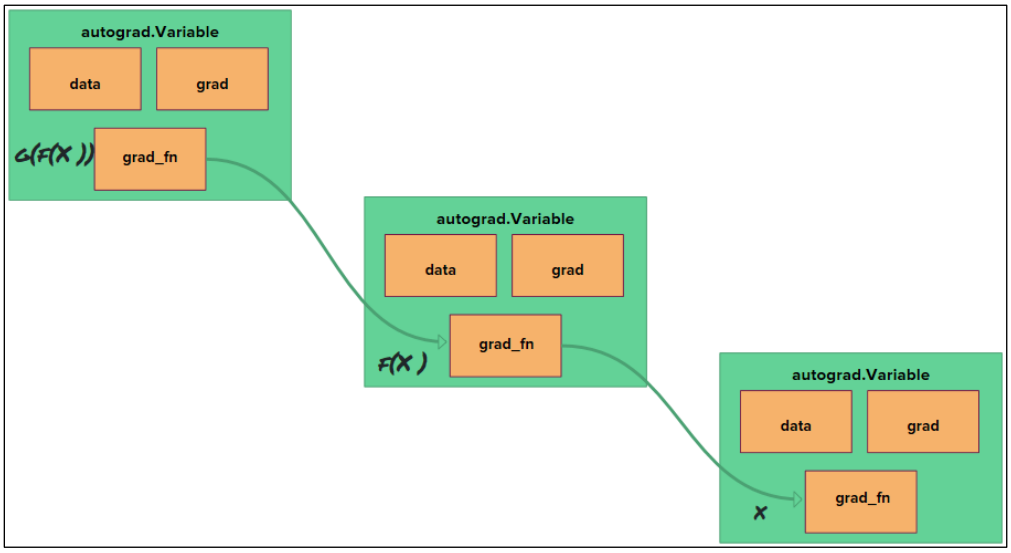


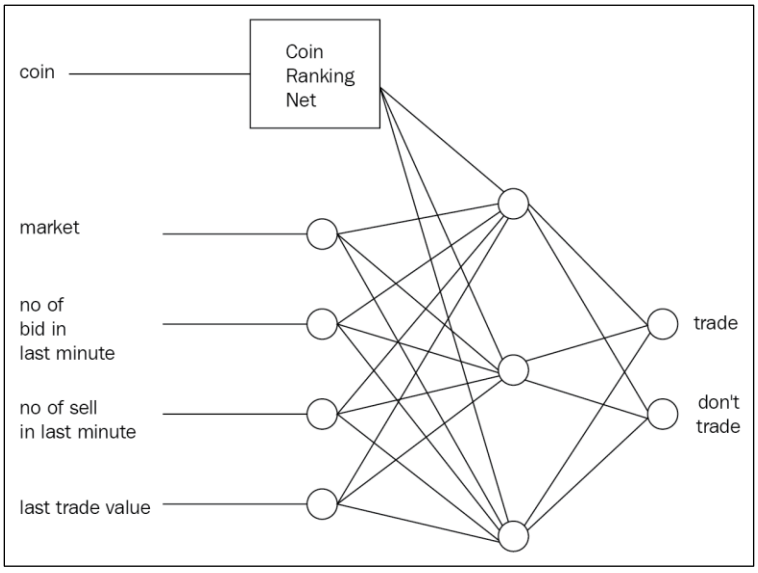
$$\text{weight gradient} = \frac{\partial t}{\partial \text{weight}} = \frac{\partial t}{\partial \text{weight}} * \frac{\partial \text{out}}{\partial t} \times \frac{\partial \text{out}}{\partial \text{out}}$$

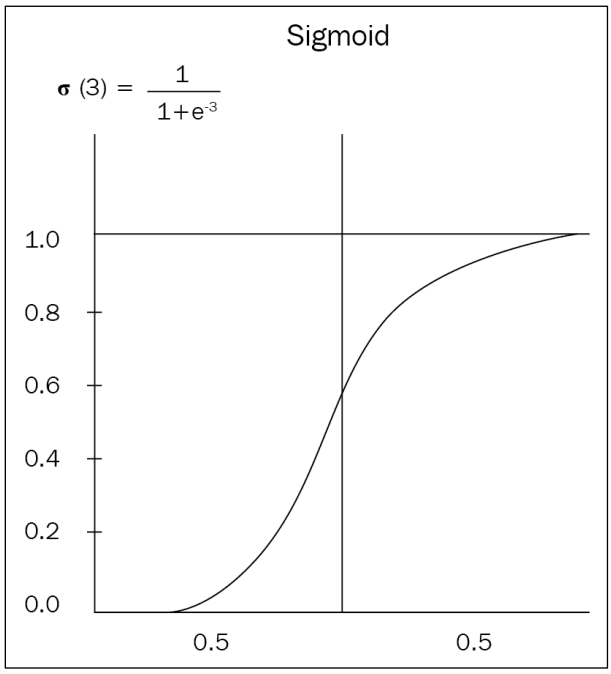



5
Binary Encode
0
0
0
0
0
0
0
1
0
1
1
0
0
0

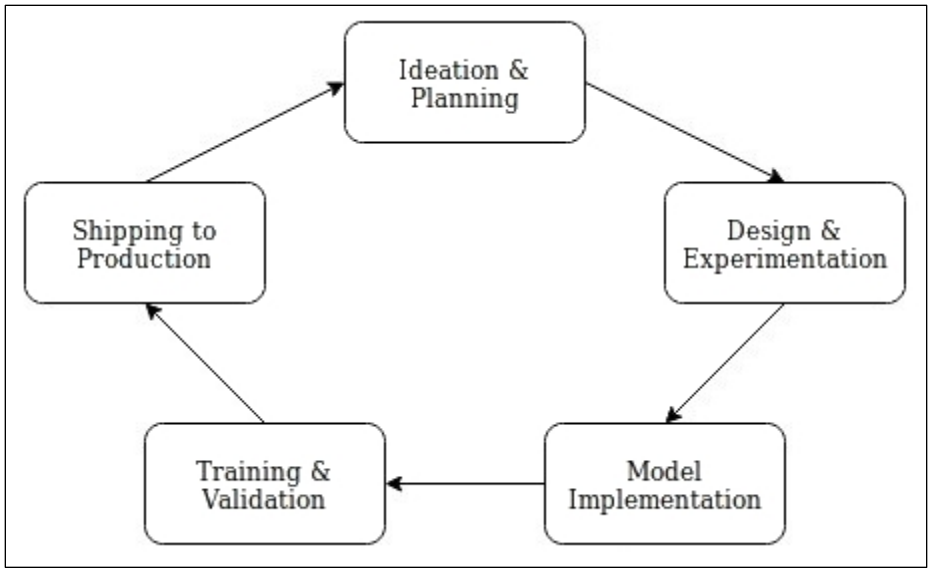

loss


autograd.Variable
data
grad
G(F(X))
grad_fn
autograd.Variable
data
grad
F(X)
grad_fn
autograd.Variable
data
grad
X
grad_fn


coin
Coin
Ranking
Net
market
no of
bid in
last minute
no of sell
in last minute
last trade value
trade
don't
trade

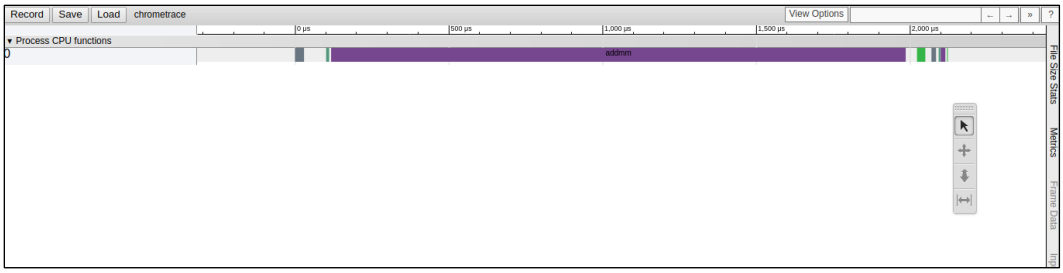

Sigmoid
σ (3) = 1 / 1+e-3
1.0
0.8
0.6
0.4
0.2
0.0
0.5
0.5

Chapter 3: Deep Learning Workflow



Name	CPU time	CUDA time	Calls	CPU total	CUDA total
t	15.758us	0.000us	2	31.516us	0.000us
expand	6.203us	0.000us	2	12.405us	0.000us
addmm	898.371us	0.000us	2	1796.742us	0.000us
sigmoid	14.462us	0.000us	2	28.923us	0.000us

Name	CPU time	CUDA time	Calls	CPU total	CUDA total
sigmoid	2.524us	0.000us	1	2.524us	0.000us
expand	4.325us	0.000us	1	4.325us	0.000us
t	5.140us	0.000us	1	5.140us	0.000us
expand	9.168us	0.000us	1	9.168us	0.000us
addmm	16.646us	0.000us	1	16.646us	0.000us
sigmoid	29.335us	0.000us	1	29.335us	0.000us
t	30.314us	0.000us	1	30.314us	0.000us
addmm	1858.968us	0.000us	1	1858.968us	0.000us

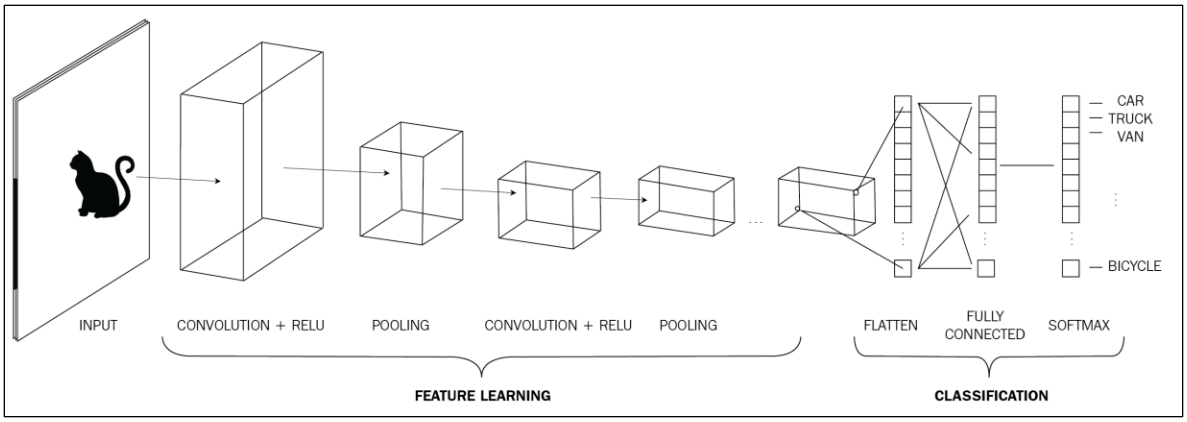


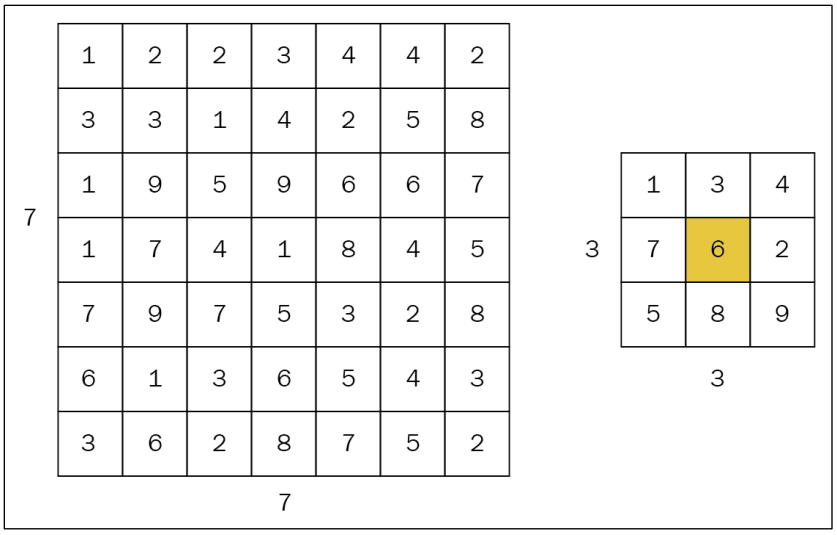
```
--------------------------------------------------------------------------------
  Environment Summary
--------------------------------------------------------------------------------
PyTorch 2018.05.07 compiled w/ CUDA 8.0.61
Running with Python 3.6 and

`pip3 list` truncated output:
msgpack-numpy (0.4.1)
numpy (1.14.2)
torch (2018.5.7, /home/sherin/miniconda3/lib/python3.6/site-packages)
torchaudio (0.1, /home/sherin/mypro/audio)
torchtext (0.2.3)
torchvision (0.2.1)
```

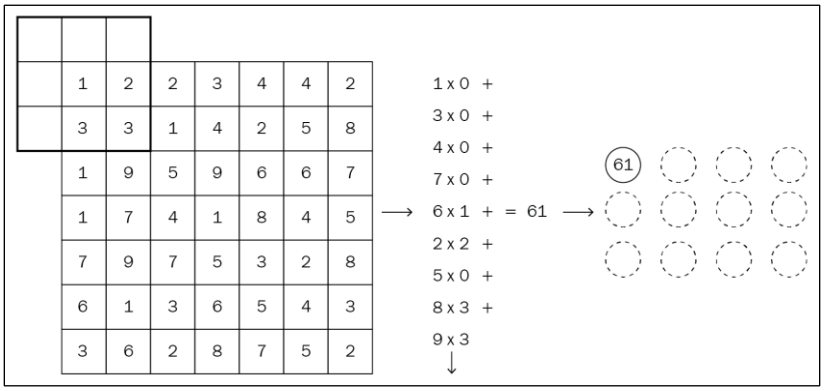
```
--------------------------------------------------------------------------------
  autograd profiler output (CPU mode)
--------------------------------------------------------------------------------
        top 15 events sorted by cpu_time_total

------------------  ------------  ------------  ------------  ------------  ------------
Name                CPU time      CUDA time     Calls         CPU total     CUDA total
------------------  ------------  ------------  ------------  ------------  ------------
AddmmBackward       233.993us     0.000us       1             233.993us     0.000us
t                   100.119us     0.000us       1             100.119us     0.000us
addmm               79.638us      0.000us       1             79.638us      0.000us
_cast_float         70.043us      0.000us       1             70.043us      0.000us
mm                  61.899us      0.000us       1             61.899us      0.000us
_mm                 53.449us      0.000us       1             53.449us      0.000us
AddmmBackward       47.908us      0.000us       1             47.908us      0.000us
zeros_like          46.894us      0.000us       1             46.894us      0.000us
ExpandBackward      46.219us      0.000us       1             46.219us      0.000us
uniform_            44.824us      0.000us       1             44.824us      0.000us
addmm               42.784us      0.000us       1             42.784us      0.000us
sqrt                40.520us      0.000us       1             40.520us      0.000us
sqrt                39.984us      0.000us       1             39.984us      0.000us
sum                 39.054us      0.000us       1             39.054us      0.000us
MseLossBackward     39.017us      0.000us       1             39.017us      0.000us
```

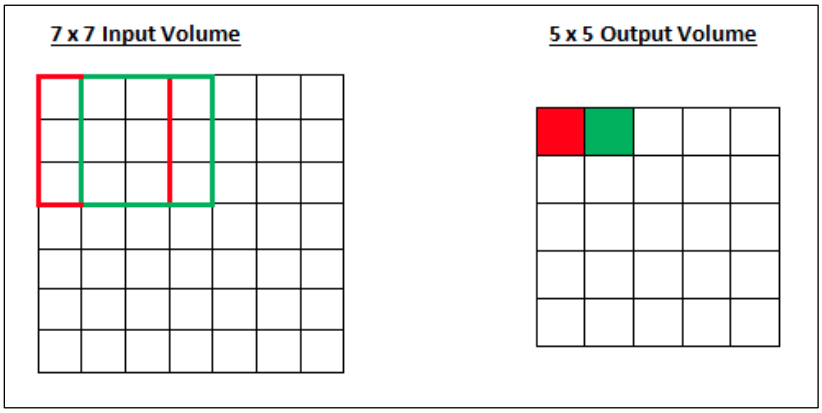
```
--------------------------------------------------------------------------------
  cProfile output
--------------------------------------------------------------------------------
         6532 function calls (6524 primitive calls) in 0.149 seconds

   Ordered by: internal time
   List reduced from 81 to 15 due to restriction <15>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.071    0.071    0.071    0.071 {method 'run_backward' of 'torch._C._EngineBase' objects}
        2    0.015    0.008    0.015    0.008 {method 'sigmoid' of 'torch._C._TensorBase' objects}
        8    0.015    0.002    0.015    0.002 {method 'add_' of 'torch._C._TensorBase' objects}
        1    0.011    0.011    0.011    0.011 {built-in method ones_like}
     1000    0.008    0.000    0.008    0.000 bottleneck_support.py:16(<listcomp>)
        2    0.005    0.003    0.005    0.003 {built-in method addmm}
        2    0.004    0.002    0.004    0.002 {built-in method numpy.core.multiarray.array}
     1000    0.004    0.000    0.013    0.000 bottleneck_support.py:15(wrapper)
        1    0.003    0.003    0.003    0.003 {built-in method torch._C._nn.mse_loss}
        1    0.003    0.003    0.021    0.021 bottleneck_support.py:39(get_data)
        4    0.002    0.001    0.002    0.001 {method 'sqrt' of 'torch._C._TensorBase' objects}
     1004    0.001    0.000    0.001    0.000 {method 'format' of 'str' objects}
        2    0.001    0.000    0.001    0.000 {method 'type' of 'torch._C._TensorBase' objects}
        1    0.001    0.001    0.149    0.149 bottleneck_support.py:1(<module>)
        1    0.001    0.001    0.001    0.001 {method 'permutation' of 'mtrand.RandomState' objects}
```

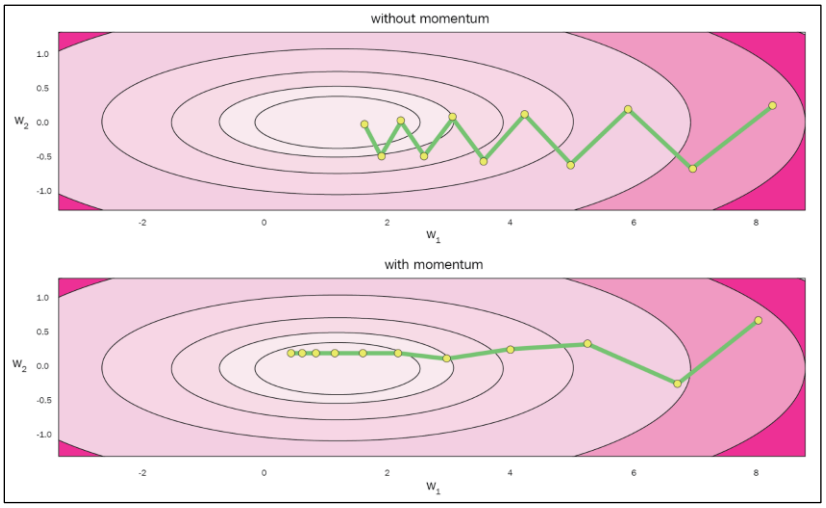
Chapter 4: Computer Vision

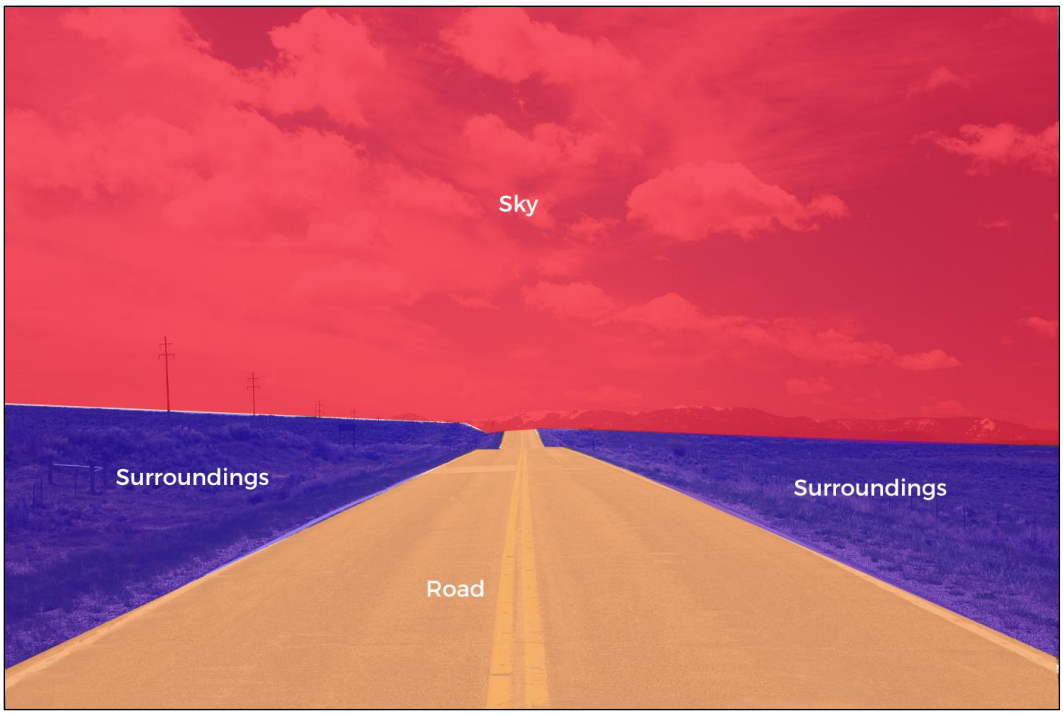


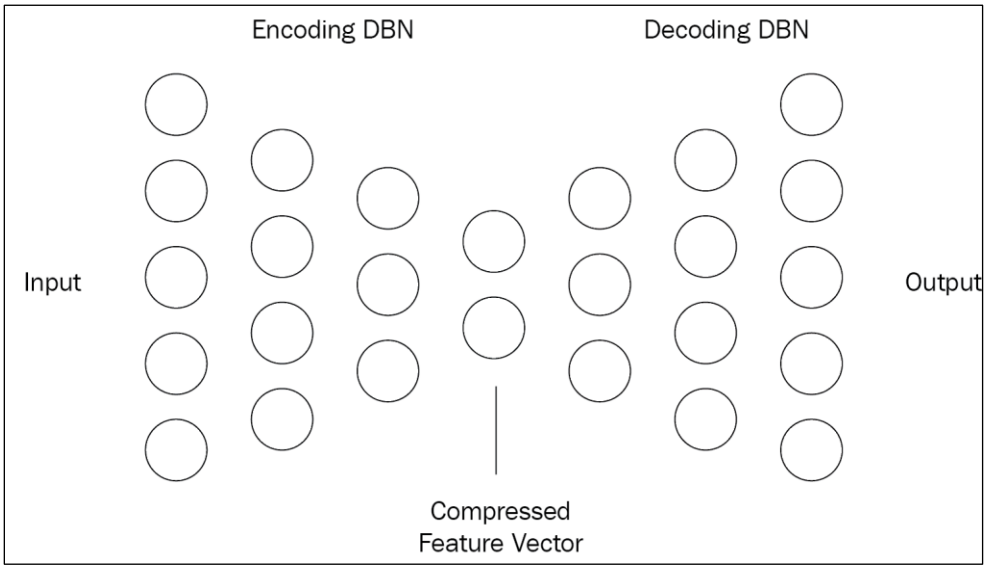


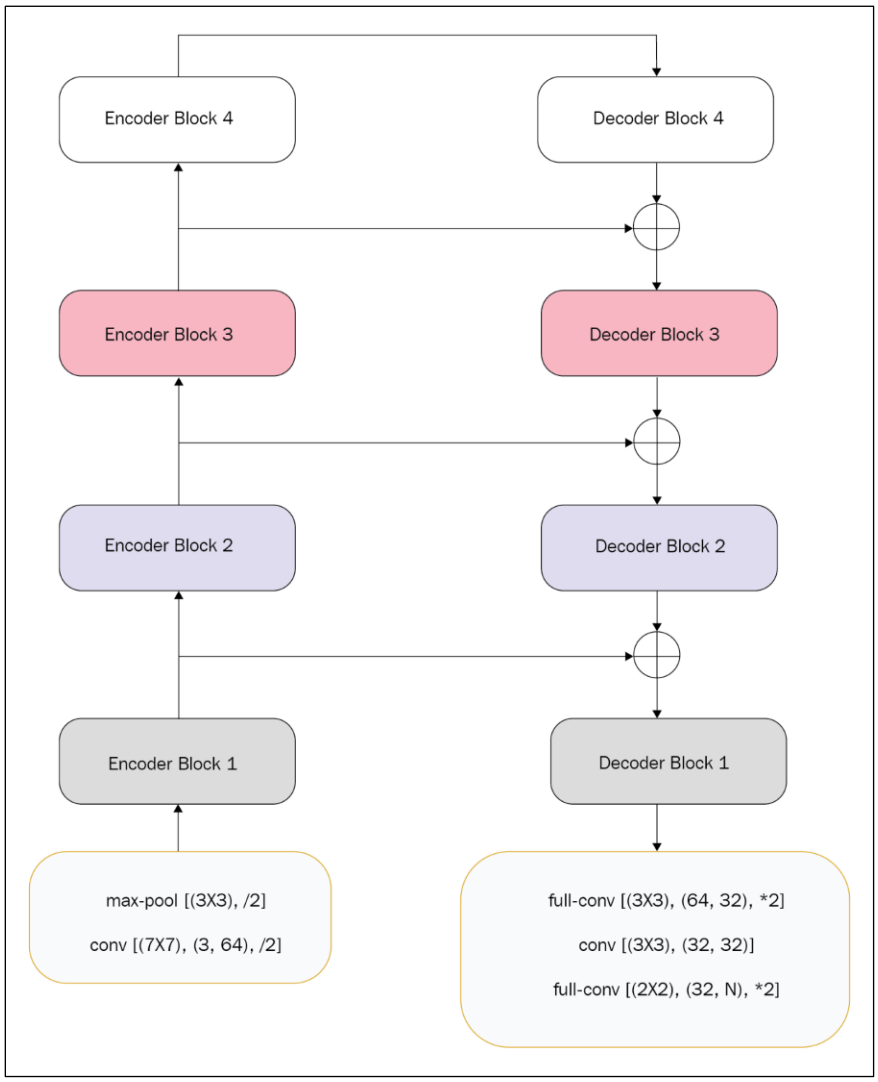

1 2 2 3 4 4 2
3 3 1 4 2 5 8
1 9 5 9 6 6 7
1 7 4 1 8 4 5
7 9 7 5 3 2 8
6 1 3 6 5 4 3
3 6 2 8 7 5 2
1 x 0 +
3 x 0 +
4 x 0 +
7 x 0 +
6 x 1 + = 61
2 x 2 +
5 x 0 +
8 x 3 +
9 x 3
61

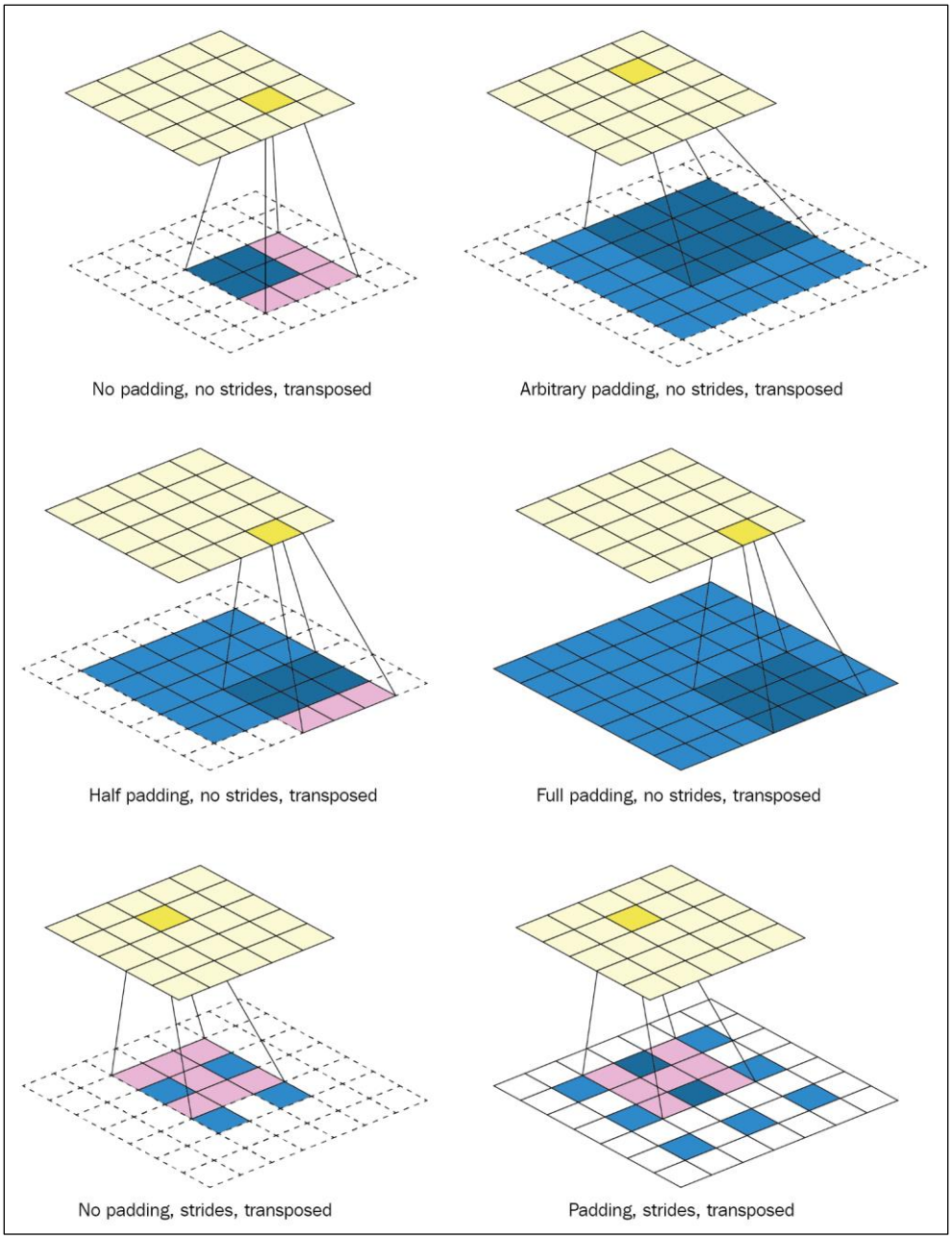

7 x 7 Input Volume
5 x 5 Output Volume

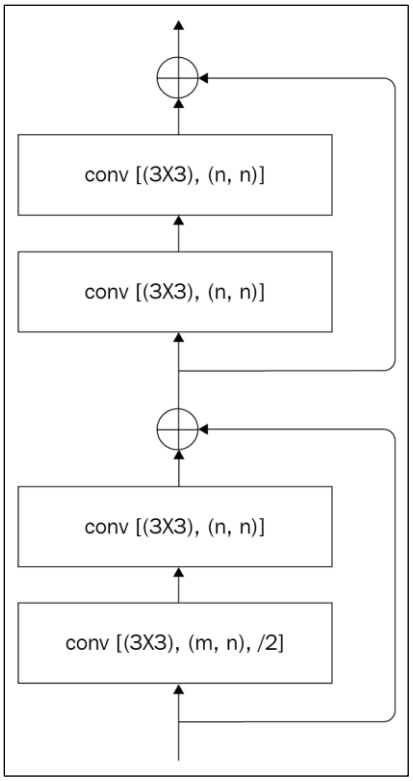

without momentum
with momentum
w_1
w_2

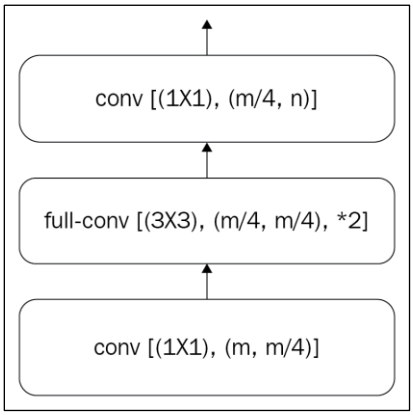

Sky
Surroundings
Surroundings
Road


Encoding DBN
Decoding DBN
Input
Output
Compressed
Feature Vector

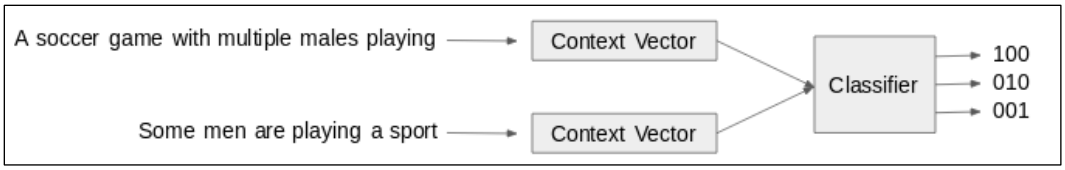

Encoder Block 4
Decoder Block 4
Encoder Block 3
Decoder Block 3
Encoder Block 2
Decoder Block 2
Encoder Block 1
Decoder Block 1
max-pool [(3X3), /2]
conv [(7X7), (3, 64), /2]
full-conv [(3X3), (64, 32), *2]
conv [(3X3), (32, 32)]
full-conv [(2X2), (32, N), *2]

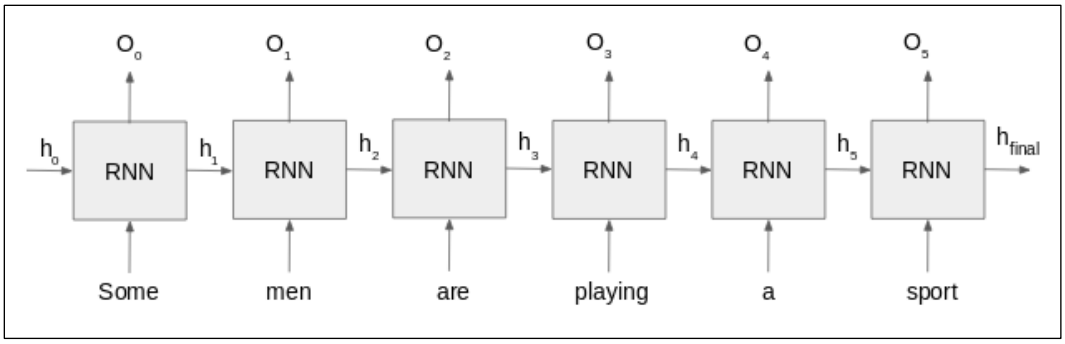

No padding, no strides, transposed
Arbitrary padding, no strides, transposed
Half padding, no strides, transposed
Full padding, no strides, transposed
No padding, strides, transposed
Padding, strides, transposed

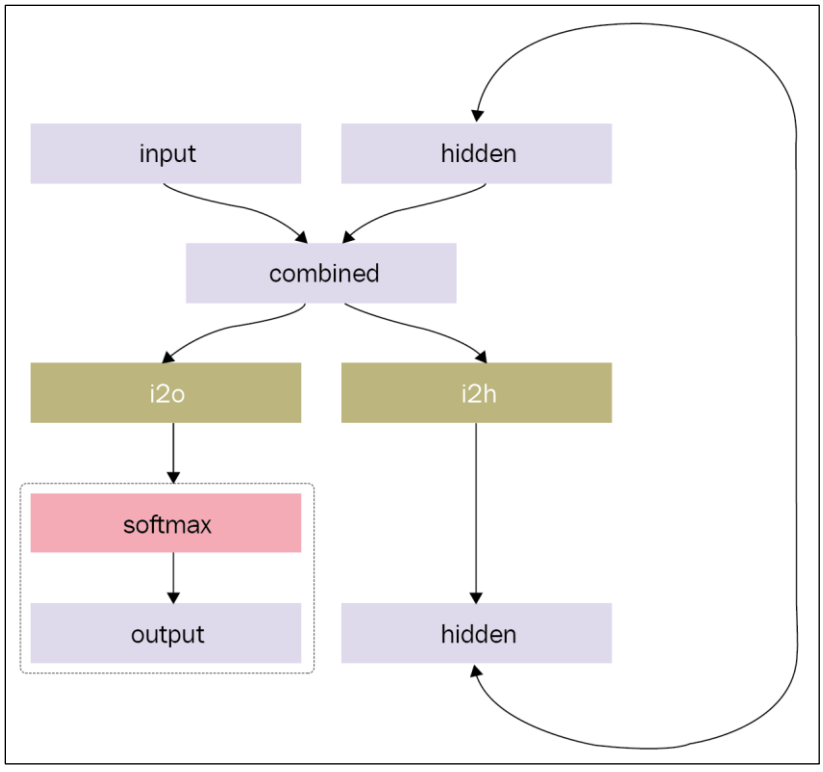

conv [(3X3), (n, n)]
conv [(3X3), (n, n)]
conv [(3X3), (n, n)]
conv [(3X3), (m, n), /2]

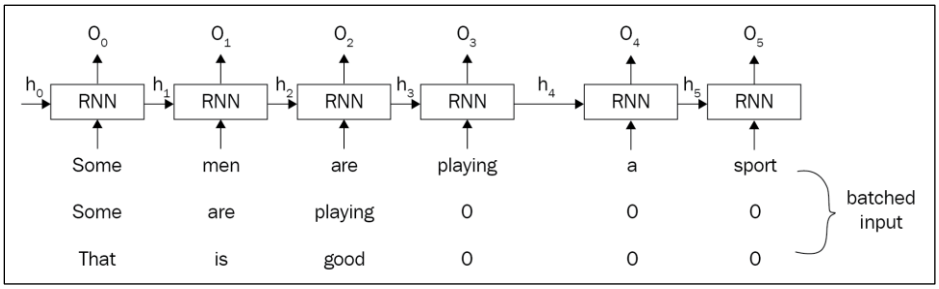

conv [(1X1), (m/4, n)]
full-conv [(3X3), (m/4, m/4), *2]
conv [(1X1), (m, m/4)]

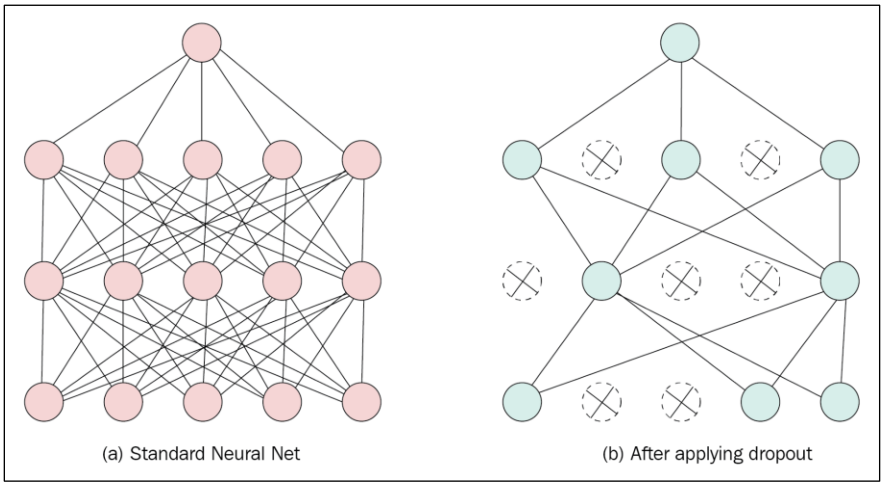
Chapter 5: Sequential Data Processing

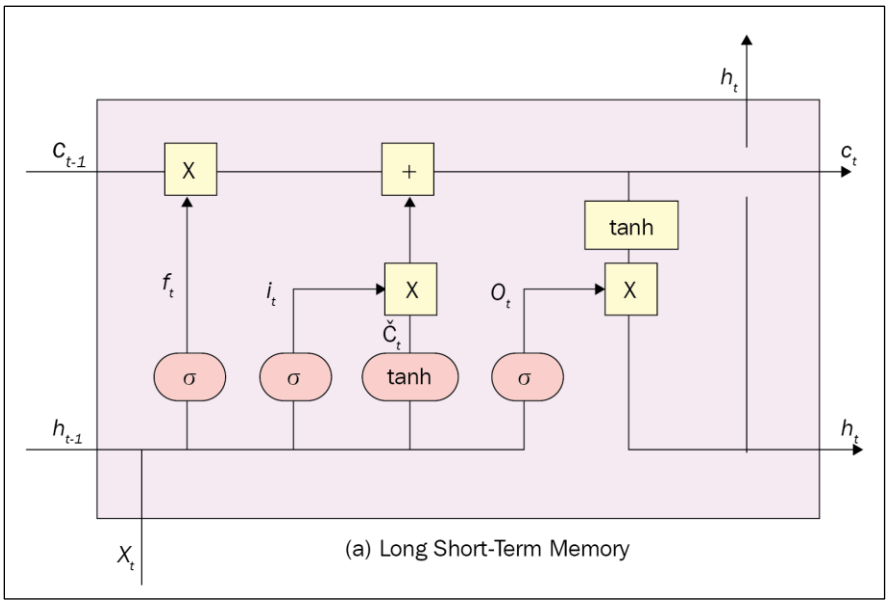


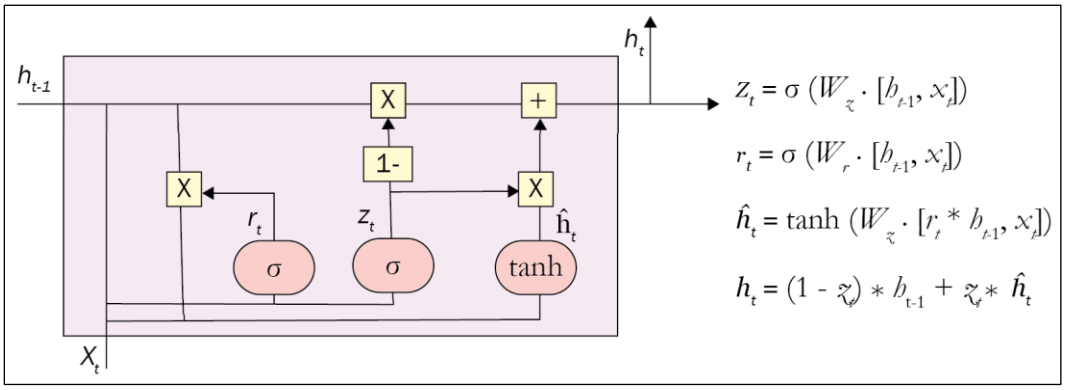


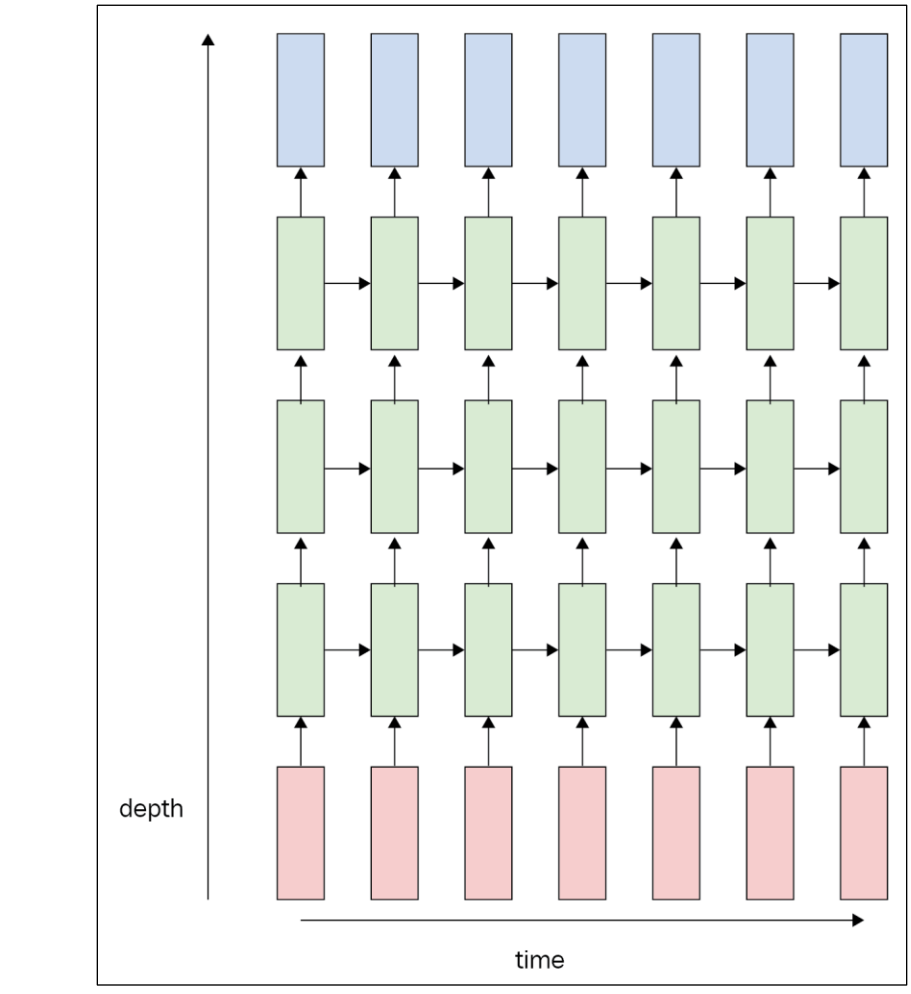


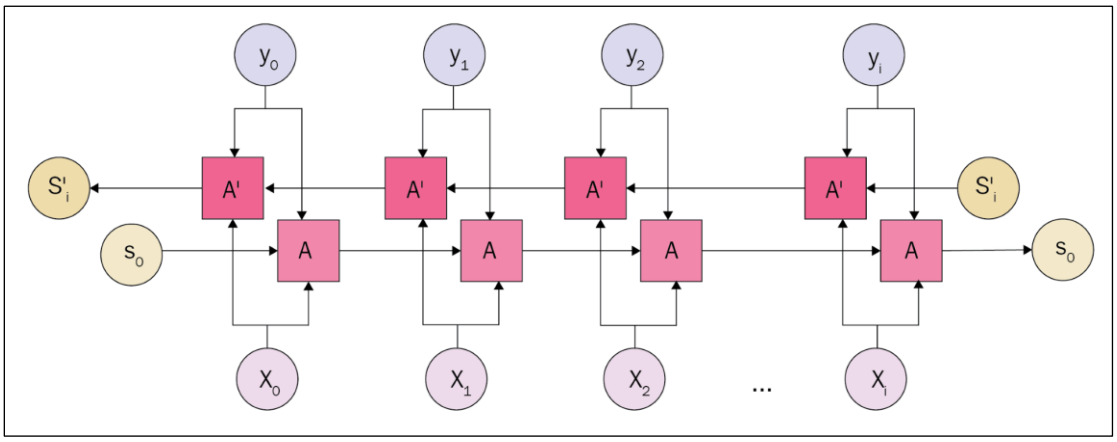

O0
O1
O2
O3
O4
O5
h0
h1
h2
h3
h4
h5
RNN
Some
men
are
playing
a
sport
Some
are
playing
0
0
0
That
is
good
0
0
0
batched input


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


ht
ct-1
X
+
ct
tanh
ft
it
X
Ot
X
Č t
σ
σ
tanh
σ
ht-1
ht
Xt
(a) Long Short-Term Memory


h_{t-1}
h_t
X
+
1-
X
X
r_t
z_t
$\hat{h}_t$
σ
σ
tanh
X_t
$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$
$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$
$\hat{h}_t = \tanh(W_z \cdot [r_t * h_{t-1}, x_t])$
$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t$


depth
time



Reduce for each blank space

Shift for each character

input	stack
H E _ L L _ O_ _ _	
E _ L L _ O_ _ _	H
_ L L _ O_ _ _	E H
L L _ O_ _ _	H
L _ O_ _ _	L H
_ O_ _ _	L L H
O_ _ _	L H
_ _ _	O L H
_ _	L H
_	H

Chapter 6: Generative Networks

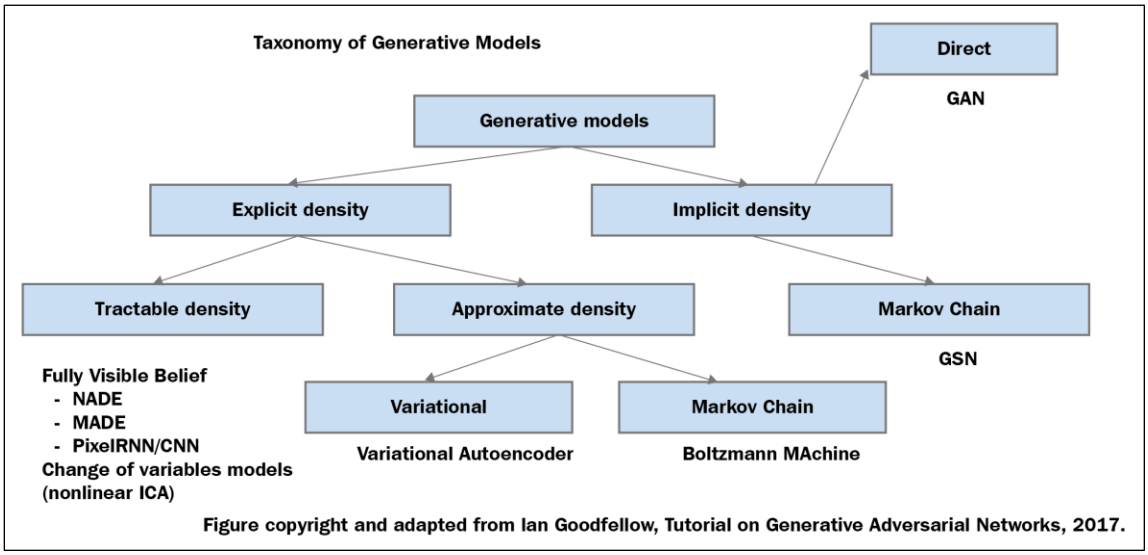
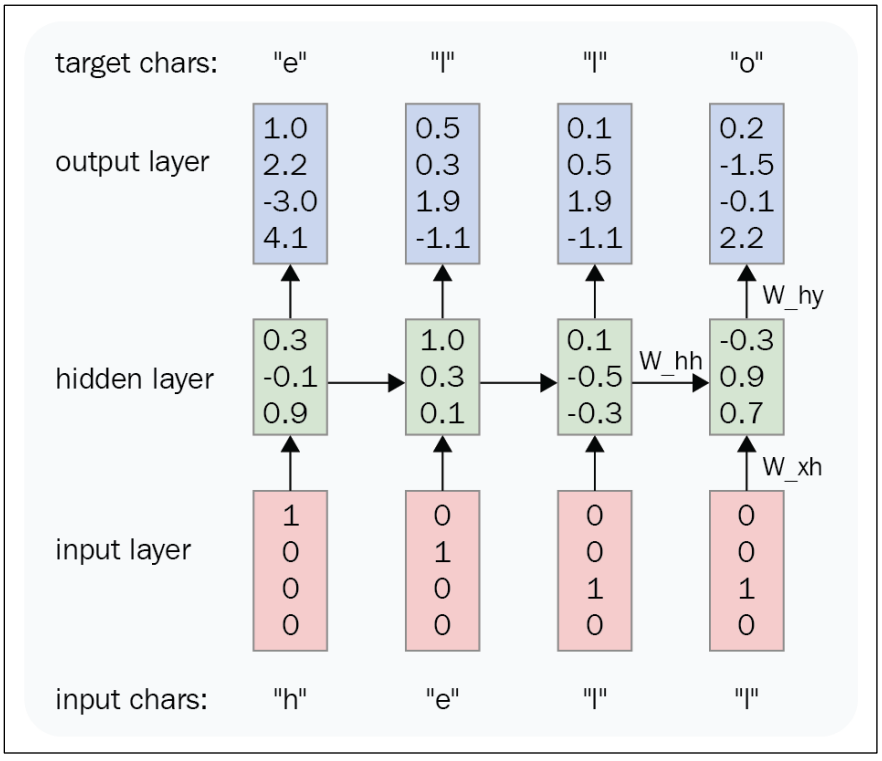
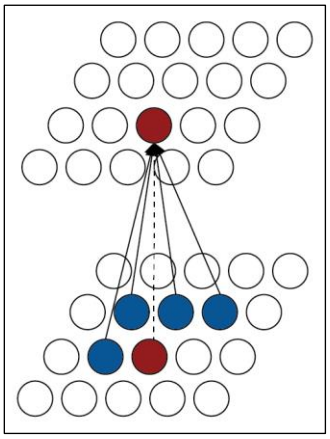
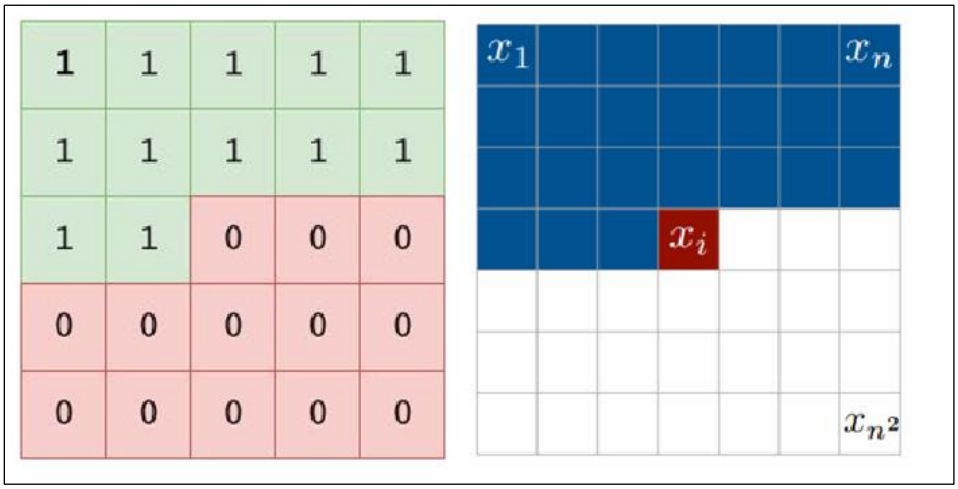
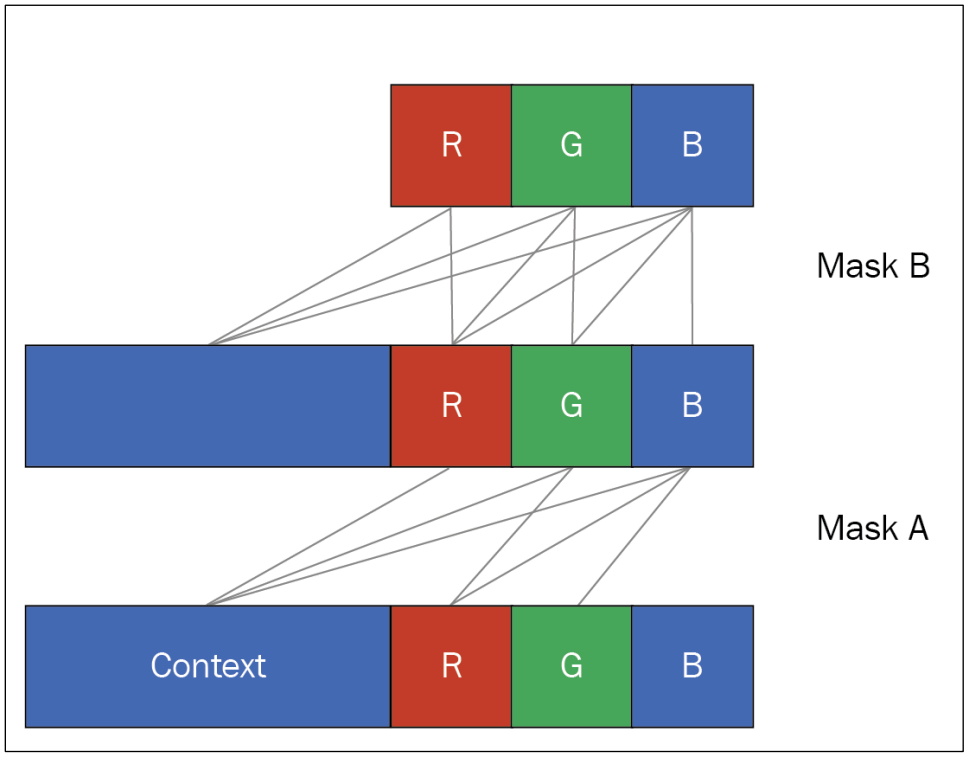


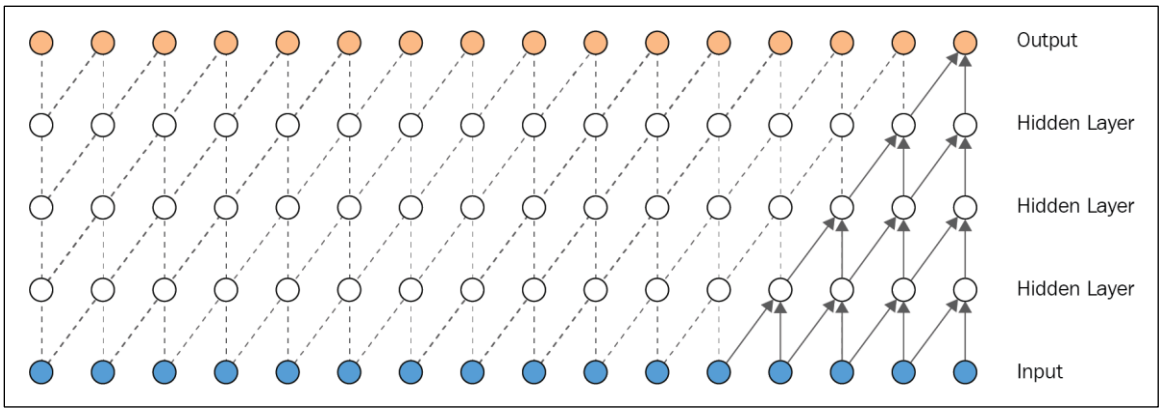
Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

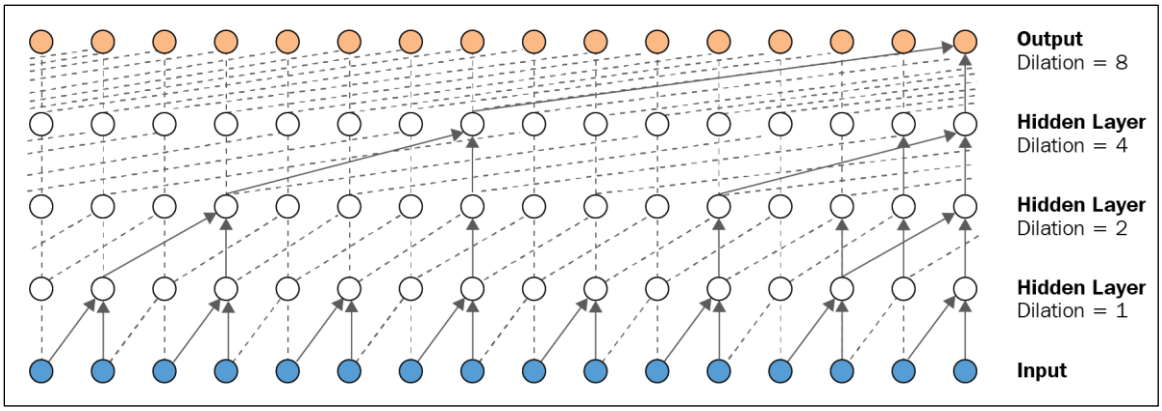


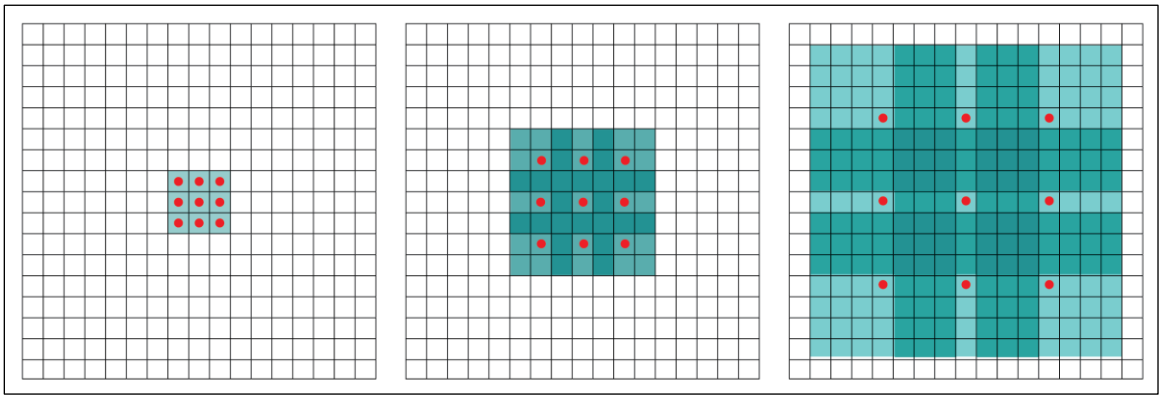


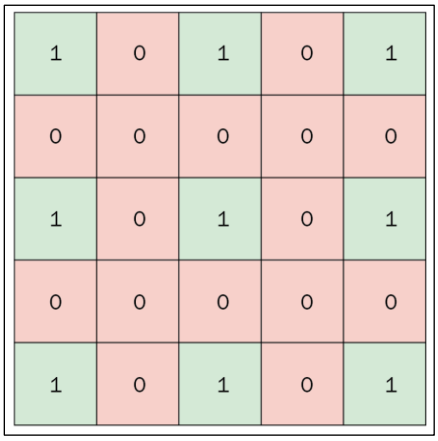

1 1 1 1 1
1 1 1 1 1
1 1 0 0 0
0 0 0 0 0
0 0 0 0 0
x_1
x_n
x_i
x_{n^2}

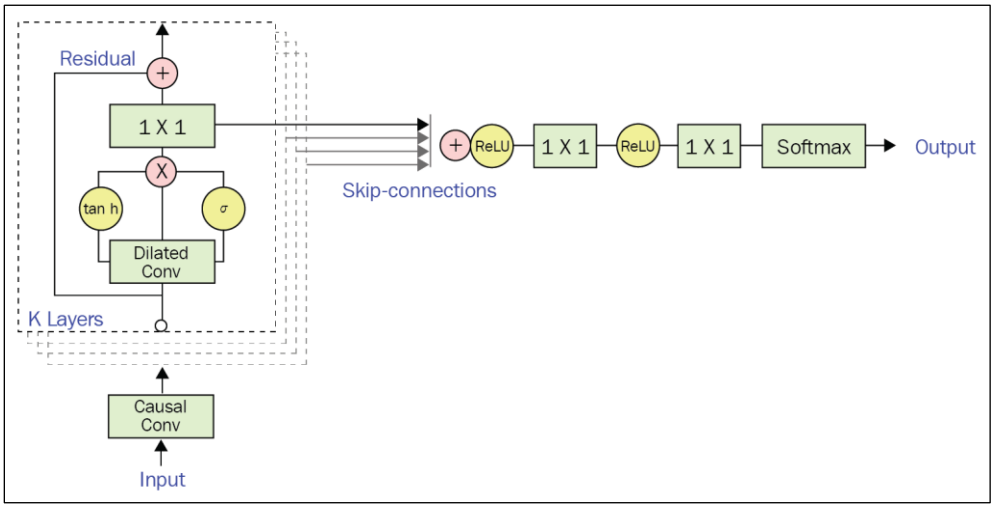

R
G
B
Mask B
R
G
B
Mask A
Context
R
G
B

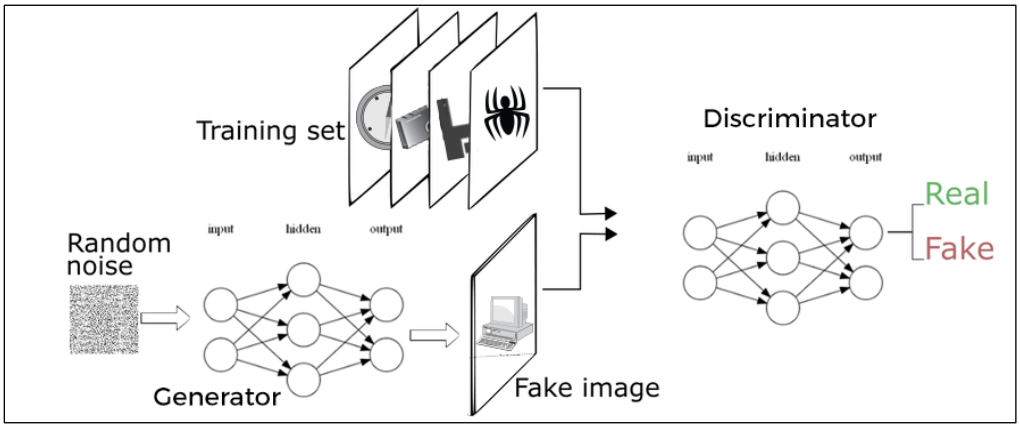

Output
Hidden Layer
Hidden Layer
Hidden Layer
Input

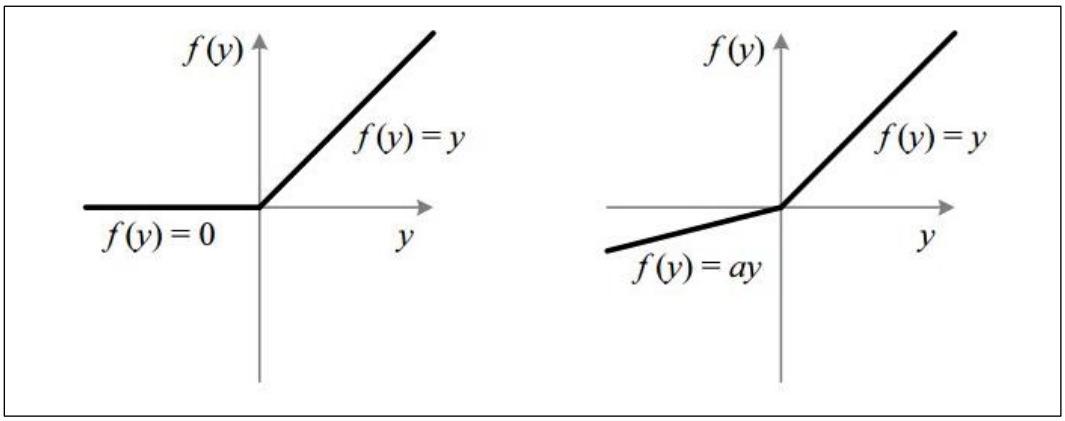

Output
Dilation = 8
Hidden Layer
Dilation = 4
Hidden Layer
Dilation = 2
Hidden Layer
Dilation = 1
Input

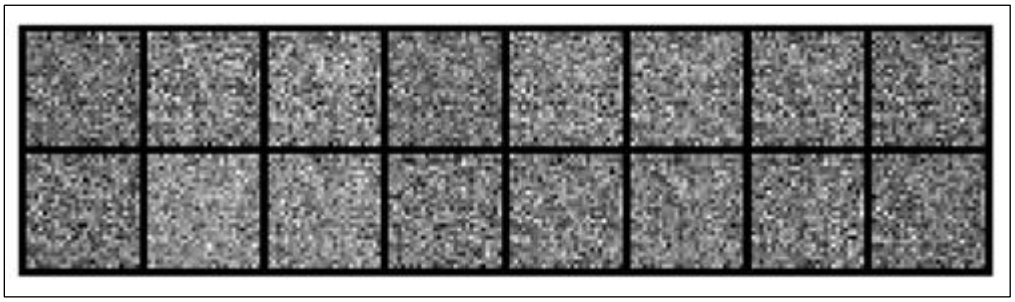


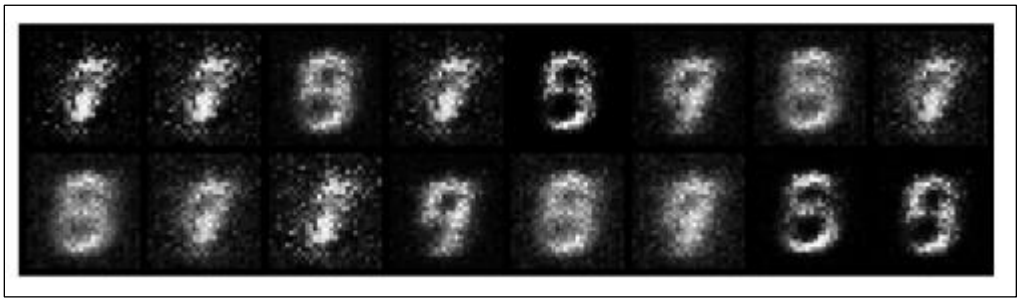

1 0 1 0 1
0 0 0 0 0
1 0 1 0 1
0 0 0 0 0
1 0 1 0 1

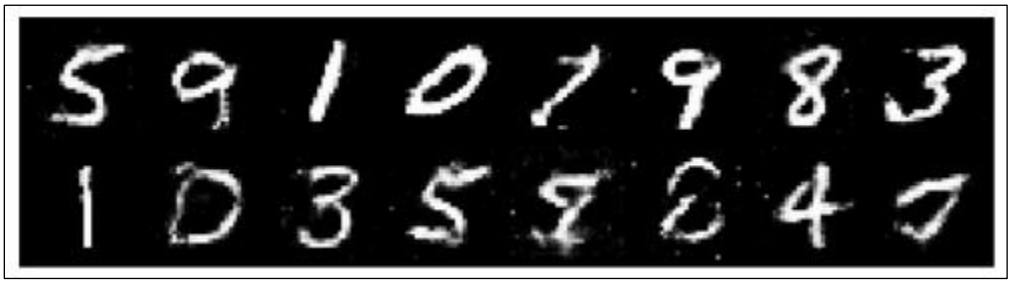

Residual
1 X 1
X
tan h
σ
Dilated Conv
K Layers
Causal Conv
Input
Skip-connections
ReLU
1 X 1
ReLU
1 X 1
Softmax
Output

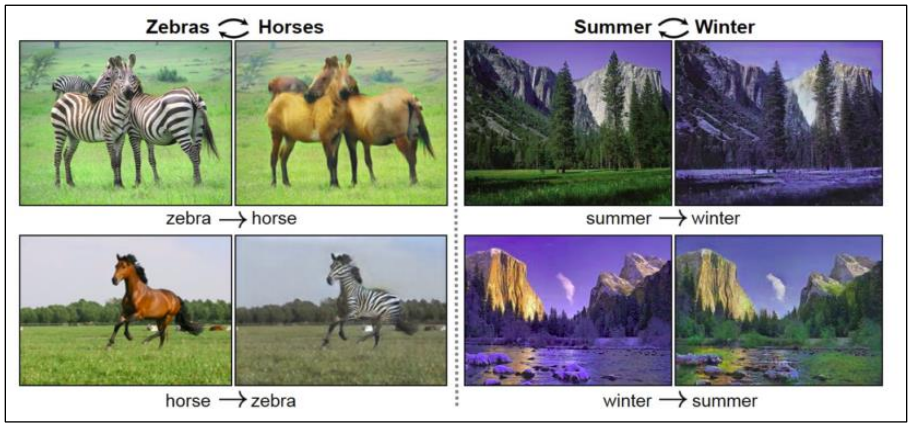

Training set
Random noise
input
hidden
output
Generator
Fake image
Discriminator
input
hidden
output
Real
Fake

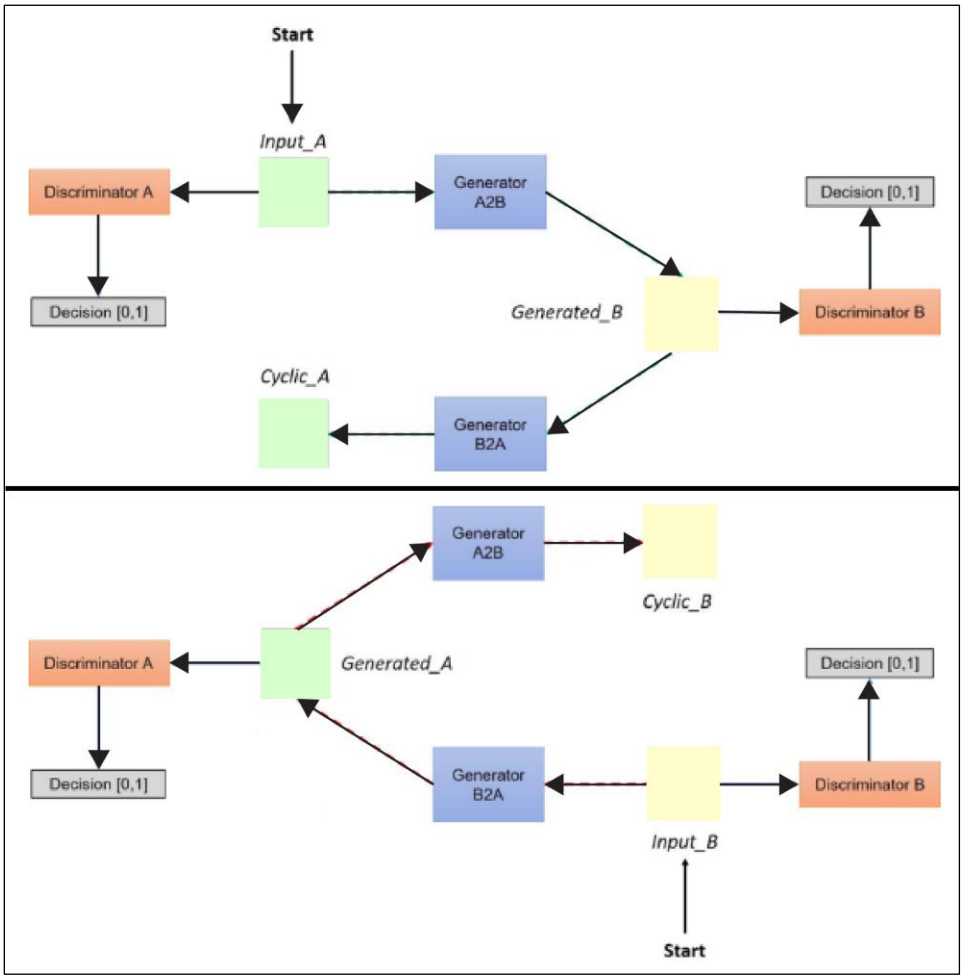

f(y)
f(y) = y
f(y) = 0
y
f(y)
f(y) = y
f(y) = ay
y

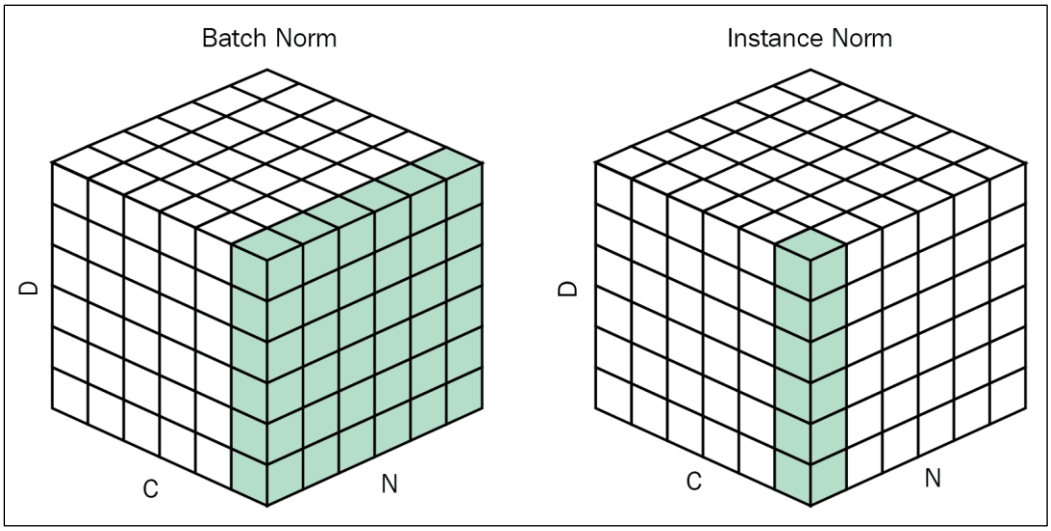




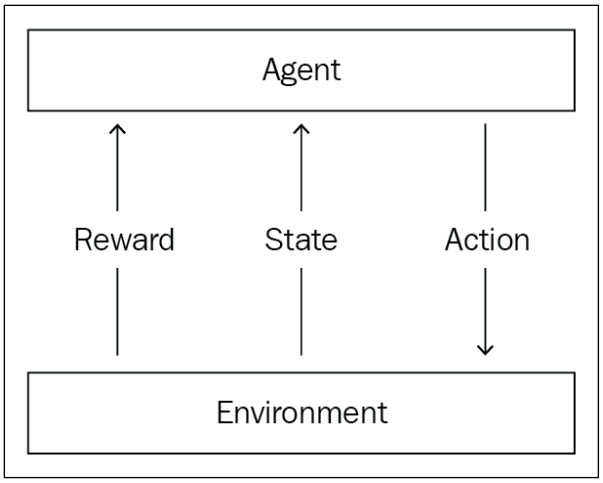


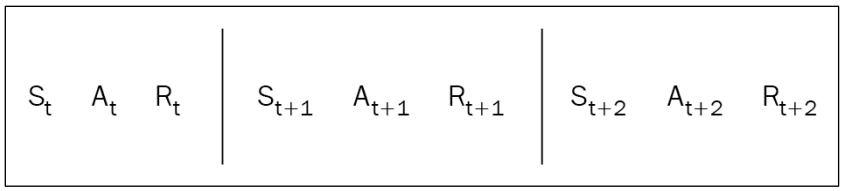

Zebras
Horses
Summer
Winter
zebra → horse
summer → winter
horse → zebra
winter → summer

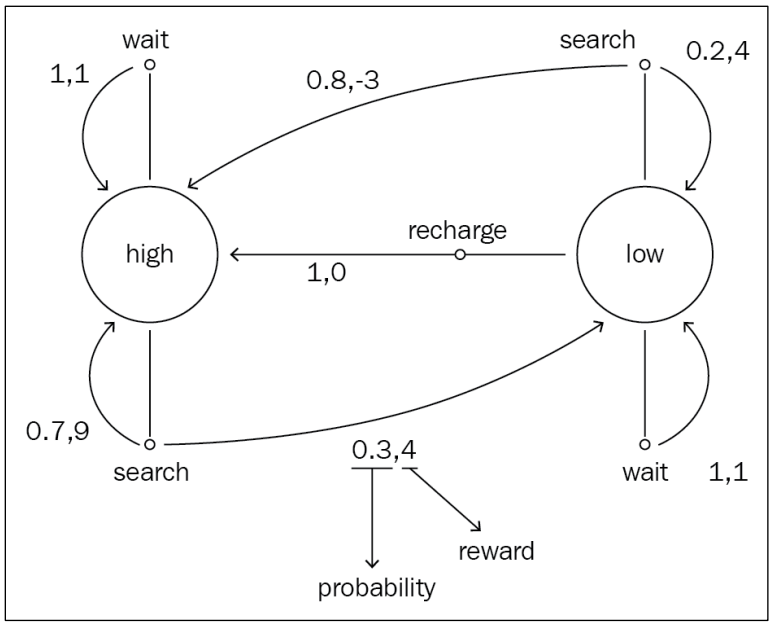

Start
Input_A
Discriminator A
Generator A2B
Decision [0,1]
Decision [0,1]
Generated_B
Discriminator B
Cyclic_A
Generator B2A
Generator A2B
Cyclic_B
Discriminator A
Generated_A
Decision [0,1]
Decision [0,1]
Generator B2A
Discriminator B
Input_B
Start

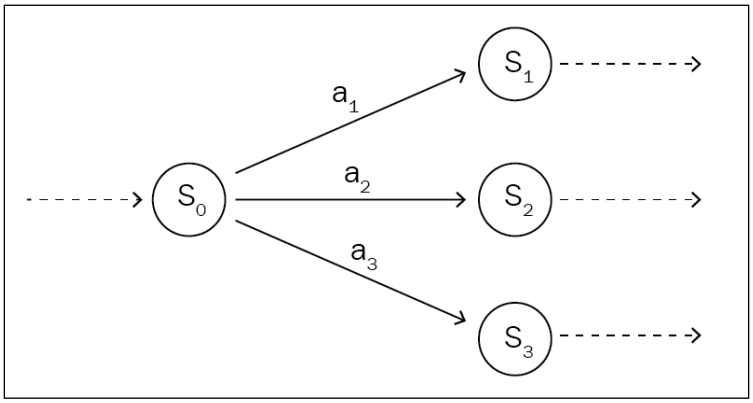

Batch Norm
D
C
N
Instance Norm
D
C
N

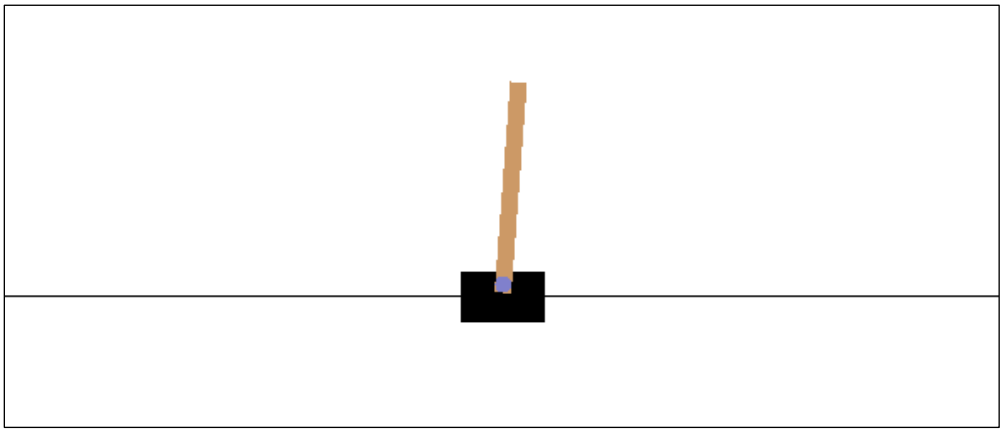
Chapter 7: Reinforcement Learning











```
class CartPoleEnv(gym.Env):
    """
    Description:
        A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum star

    Source:
        This environment corresponds to the version of the cart-pole problem described by Barto, Sutton, and Anderson

    Observation:
        Type: Box(4)
        Num     Observation               Min         Max
        0       Cart Position             -4.8            4.8
        1       Cart Velocity             -Inf            Inf
        2       Pole Angle                 -24°           24°
        3       Pole Velocity At Tip      -Inf            Inf

    Actions:
        Type: Discrete(2)
        Num     Action
        0       Push cart to the left
        1       Push cart to the right
```

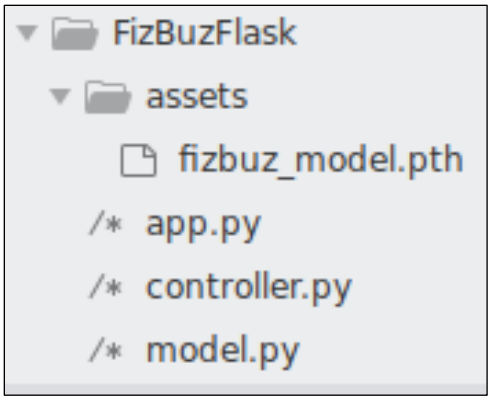
```
tensor([[[2.0429, 1.4886],
         [1.2952, 1.2798],
         [1.1960, 1.1665],
         [1.3114, 1.1780],
         [1.2970, 1.2814],
         [1.4016, 1.4096],
         [1.5460, 1.2322],
         [2.1189, 1.5717],
         [1.4563, 1.1823],
         [1.2912, 1.2759],
         [2.0797, 1.6504],
         [1.2814, 1.2050],
         [1.3184, 1.3216],
         [1.3782, 1.3824],
         [1.4194, 1.4275],
         [1.4445, 1.1700],
```

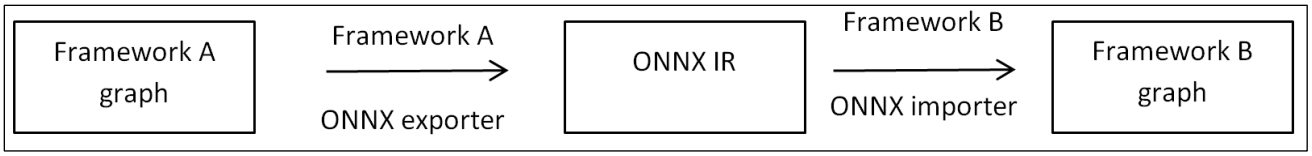
```
tensor([[[0.9818],
         [0.8832],
         [1.2682],
         [0.9230],
         [0.9572],
         [0.8275],
         [1.0659],
         [1.1392],
         [1.2381],
         [1.1048],
         [0.9397],
         [0.8558],
         [1.0015],
         [1.0669],
         [1.0863],
         [1.1538],
         [1.0786],
         [0.9248],
         [0.9540],
         [0.9916],
```

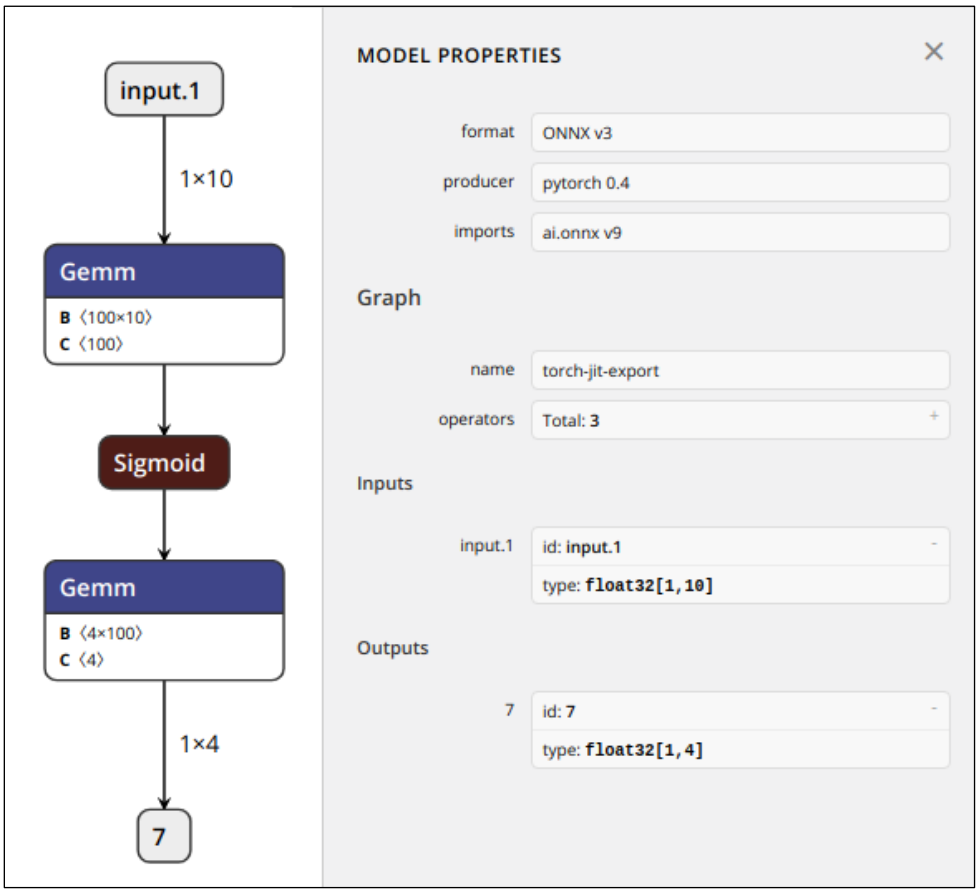
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=torch.uint8)
```

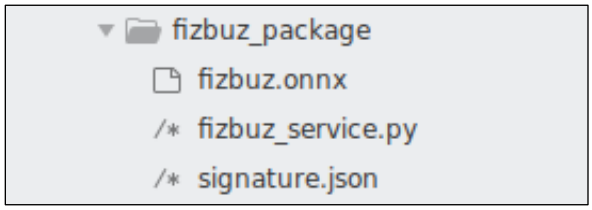
```
tensor([-0.0286, -0.0289, -0.0286, -0.0287, -0.0287, -0.0287, -0.0281, -0.0285,
        -0.0285, -0.0285, -0.0285, -0.0284, -0.0285, -0.0281, -0.0288, -0.0280,
        -0.0281, -0.0286, -0.0283, -0.0285, -0.0281, -0.0289, -0.0282, -0.0285,
        -0.0286, -0.0281, -0.0288, -0.0284, -0.0284, -0.0281, -0.0288, -0.0280,
        -0.0282, -0.0291, -0.0285, -0.0282, -0.0287, -0.0288, -0.0287, -0.0286,
         0.0000, -0.0284, -0.0285, -0.0283, -0.0289, -0.0282, -0.0286, -0.0285,
        -0.0283, -0.0286, -0.0285, -0.0284, -0.0288, -0.0287, -0.0283, -0.0280,
```

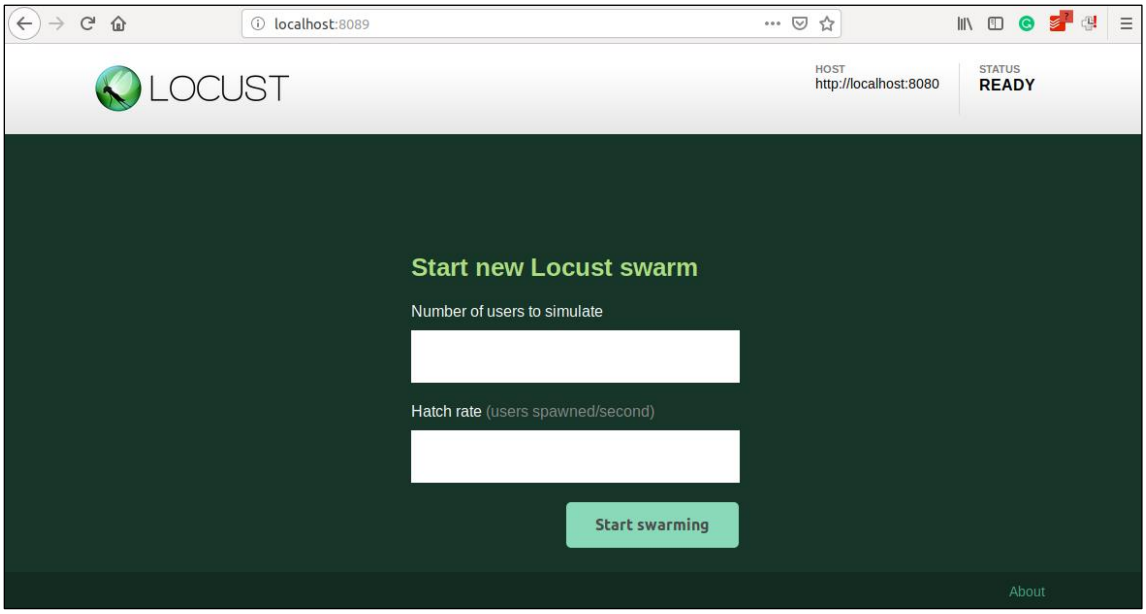
Chapter 8: PyTorch to Production






input.1
1×10
Gemm
B 〈100×10〉
C 〈100〉
Sigmoid
Gemm
B 〈4×100〉
C 〈4〉
1×4
7
MODEL PROPERTIES
format
ONNX v3
producer
pytorch 0.4
imports
ai.onnx v9
Graph
name
torch-jit-export
operators
Total: 3
Inputs
input.1
id: input.1
type: float32[1,10]
Outputs
7
id: 7
type: float32[1,4]


fizbuz_package
fizbuz.onnx
fizbuz_service.py
signature.json


localhost:8089
LOCUST
HOST
http://localhost:8080
STATUS
READY
Start new Locust swarm
Number of users to simulate
Hatch rate (users spawned/second)
Start swarming
About