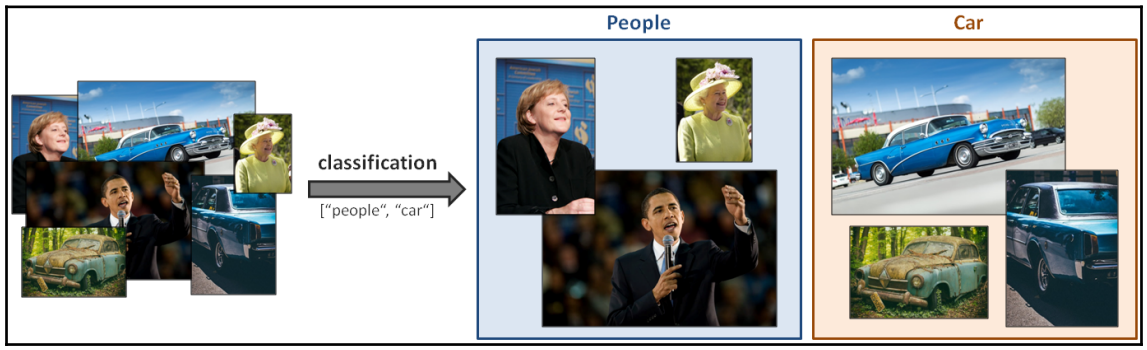
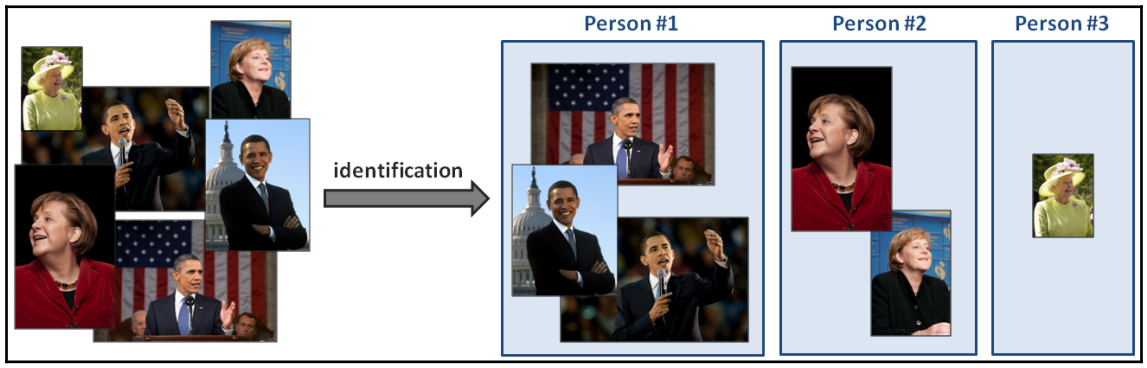
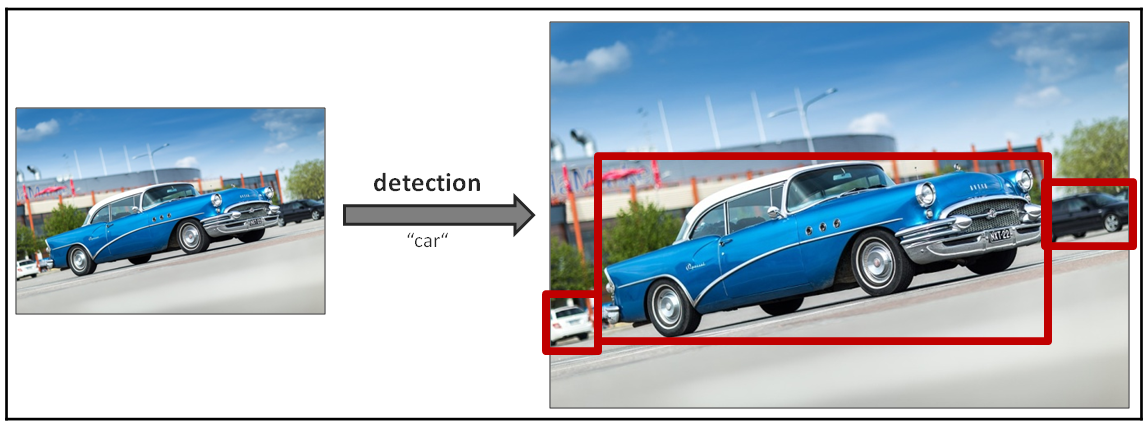
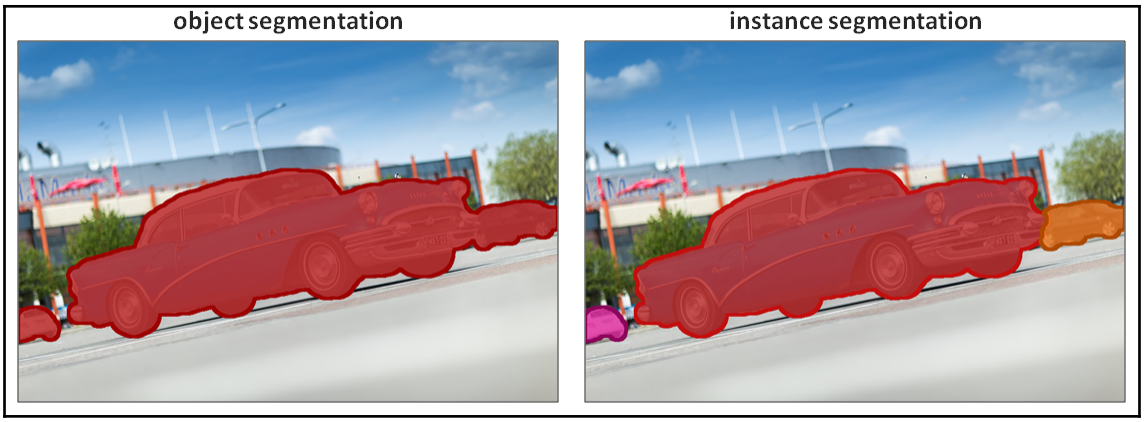
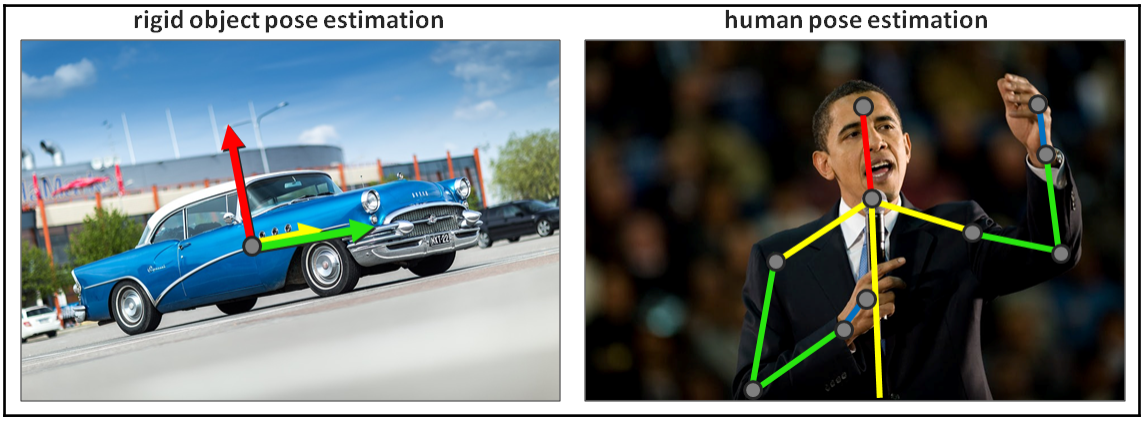
Chapter 1: Computer Vision and Neural Networks

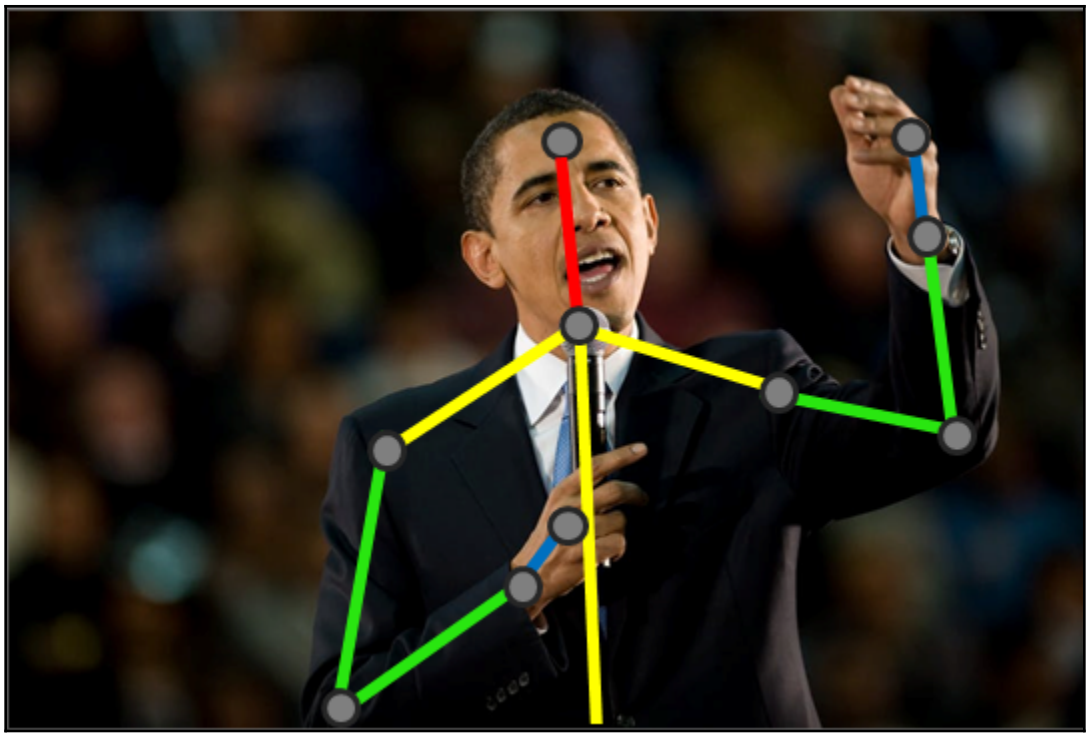


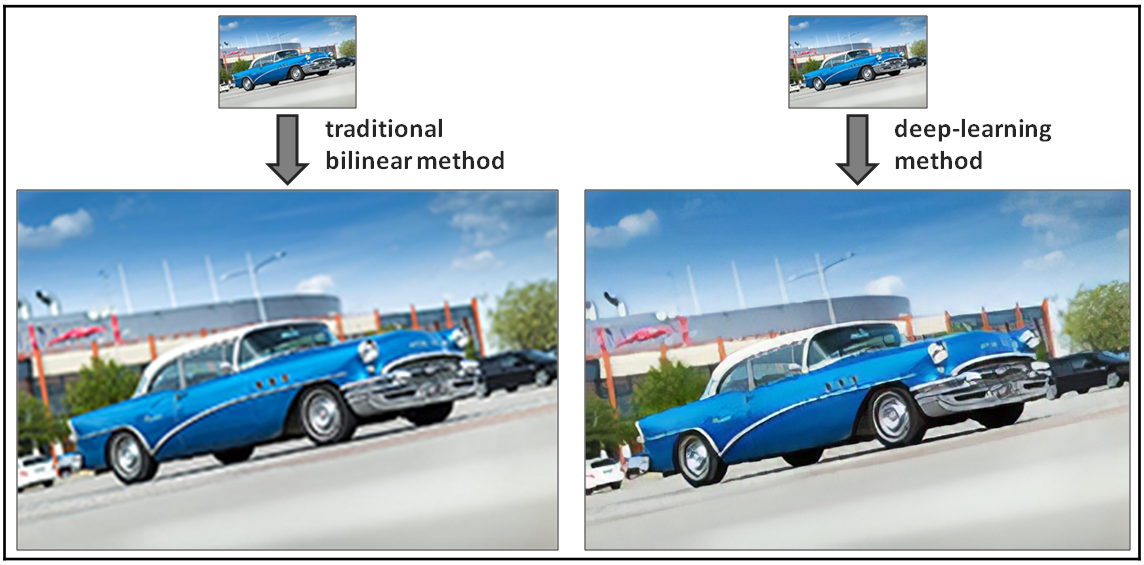


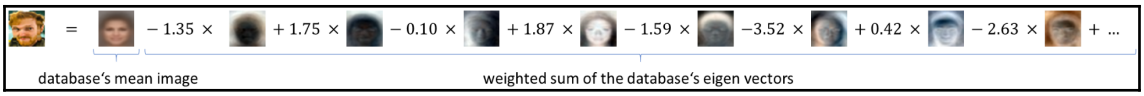


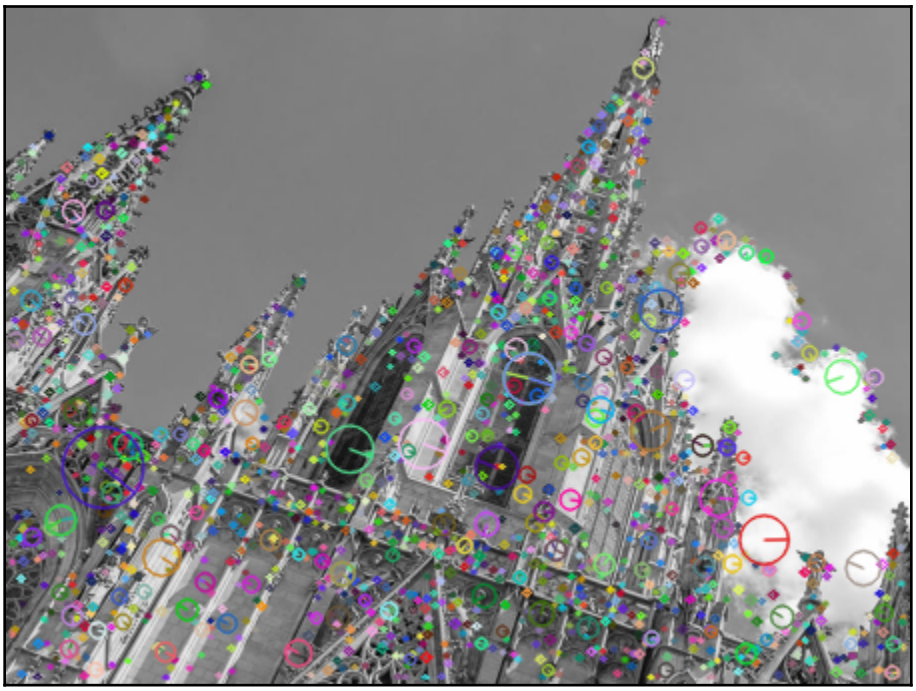

object segmentation
instance segmentation

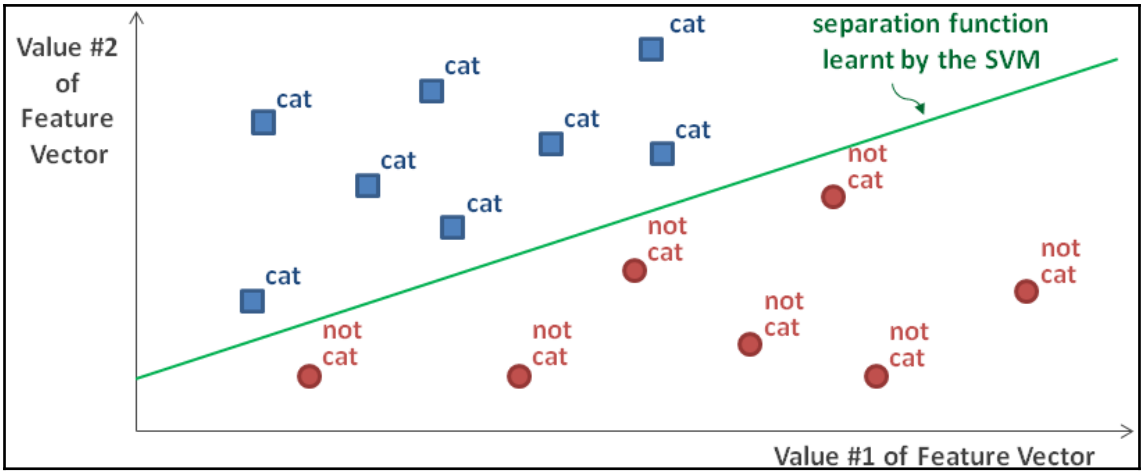

rigid object pose estimation
human pose estimation

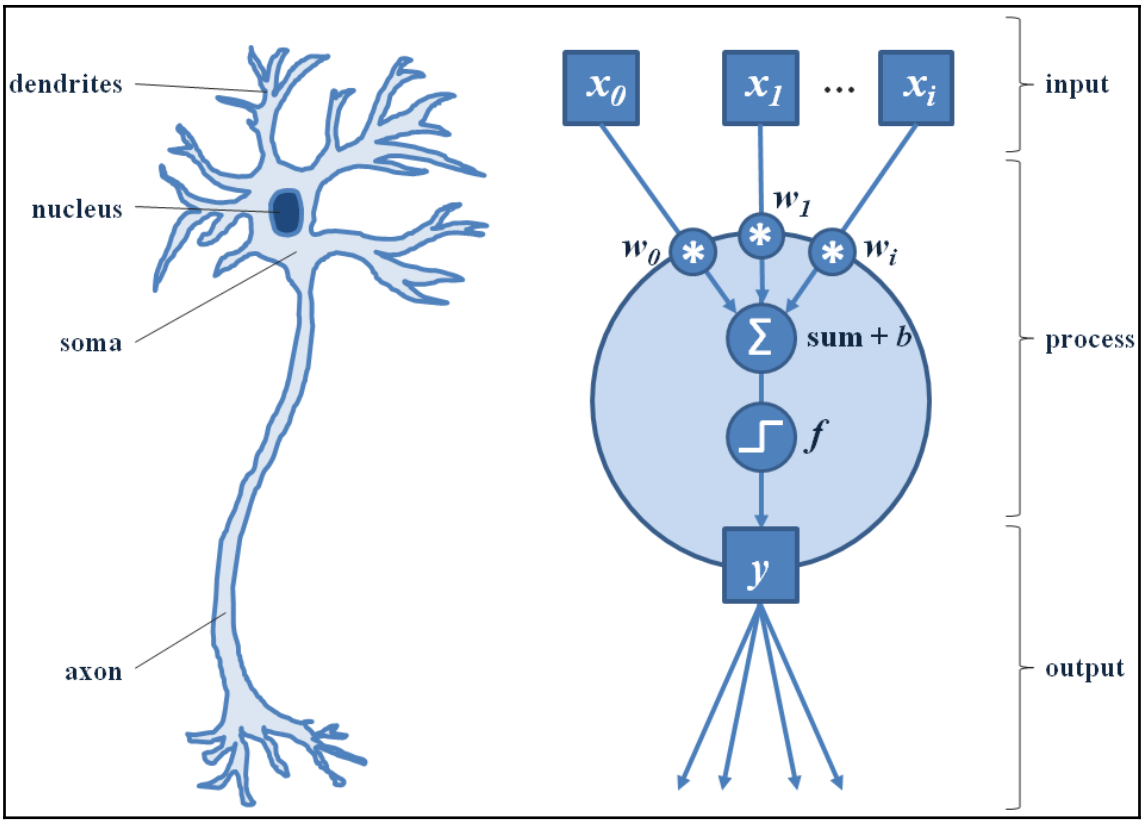


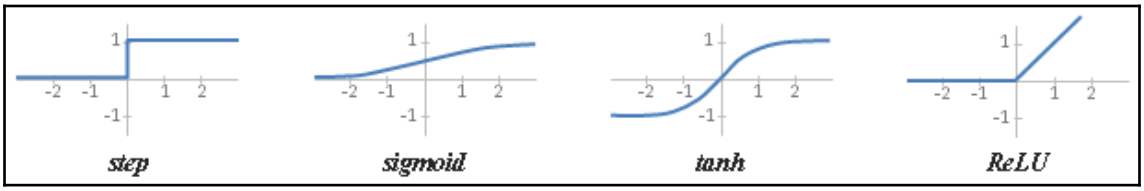

traditional
bilinear method
deep-learning
method

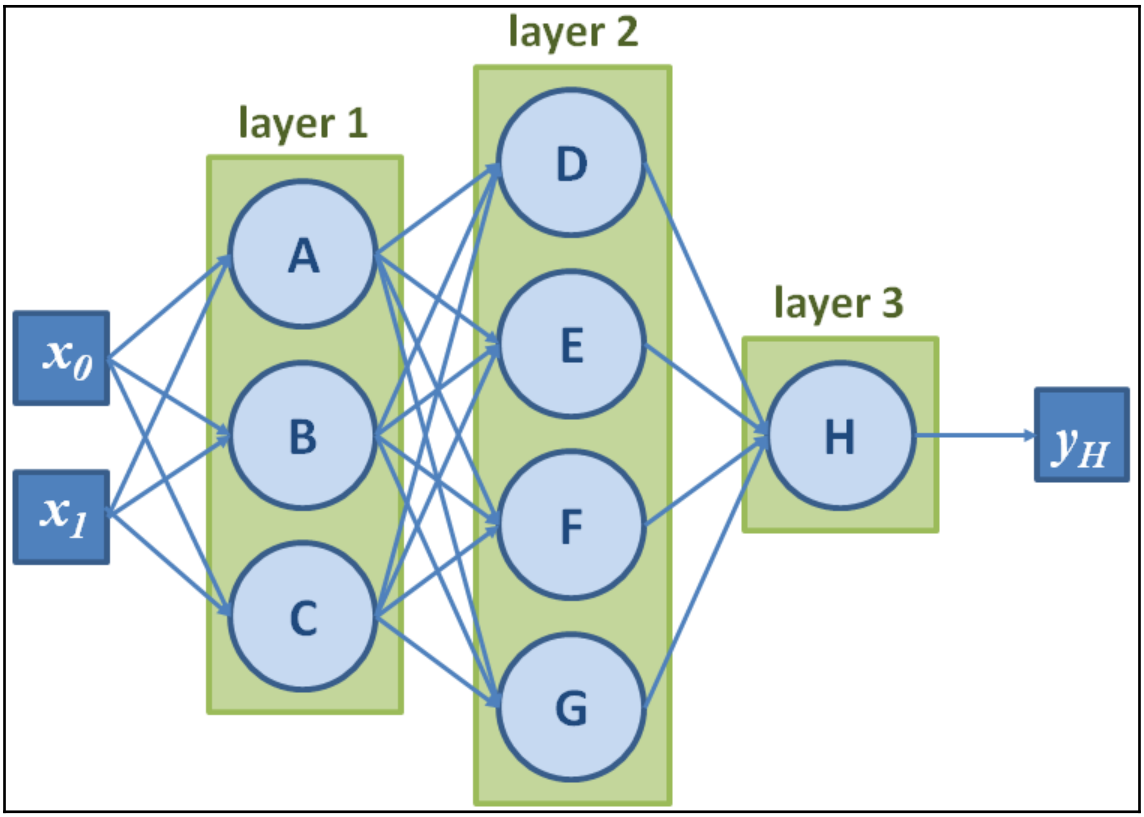

= − 1.35 × + 1.75 × − 0.10 × + 1.87 × − 1.59 × −3.52 × + 0.42 × − 2.63 × + ...
database's mean image
weighted sum of the database's eigen vectors

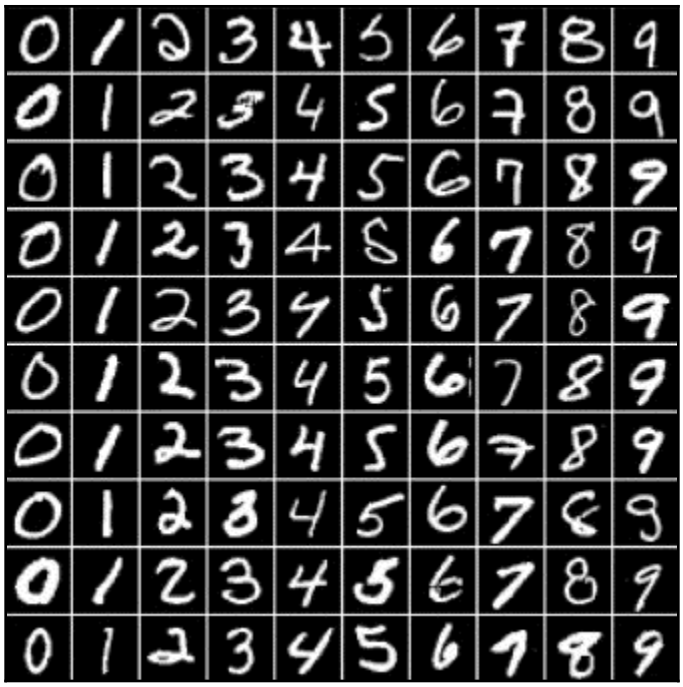


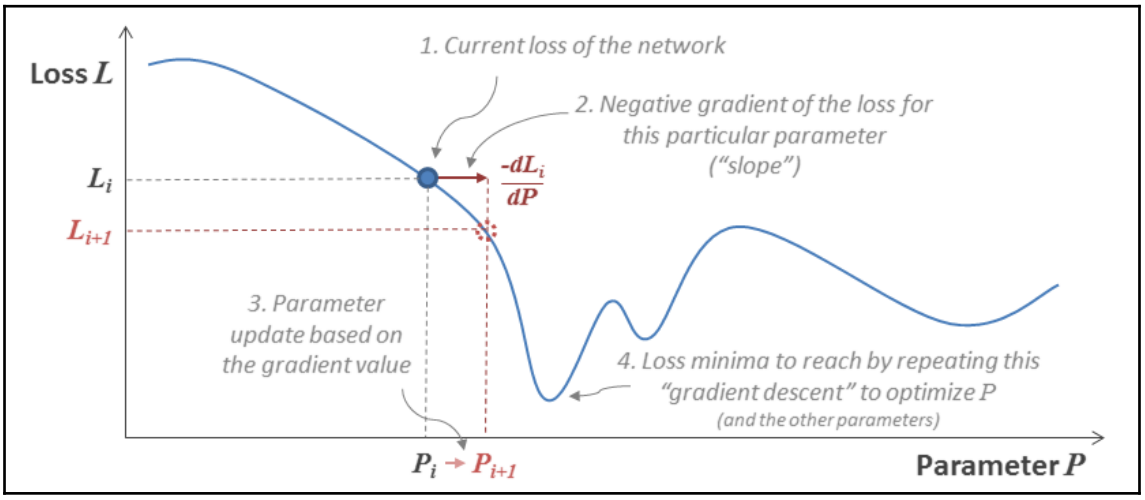

Value #2 of Feature Vector
Value #1 of Feature Vector
separation function learnt by the SVM
cat
not cat

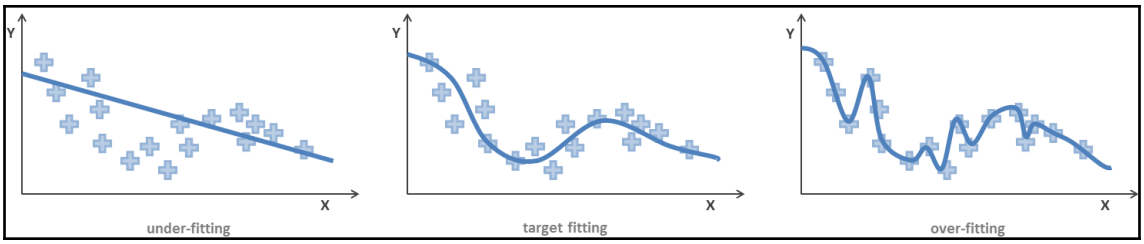

dendrites
nucleus
soma
axon
x_0
x_1
...
x_i
input
w_1
w_0
w_i
Σ
sum + b
process
f
y
output


step
sigmoid
tanh
ReLU

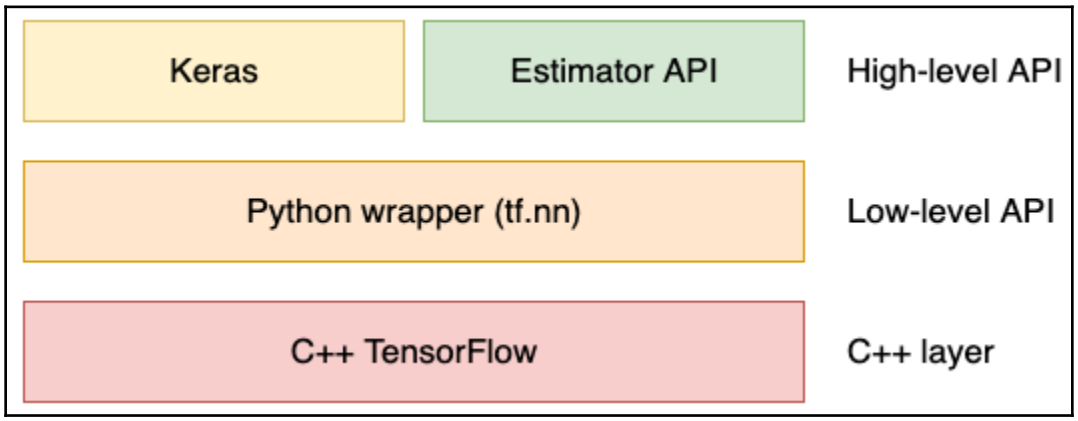

layer 2
layer 1
layer 3
D
A
E
B
F
C
G
H
x_0
x_1
y_H



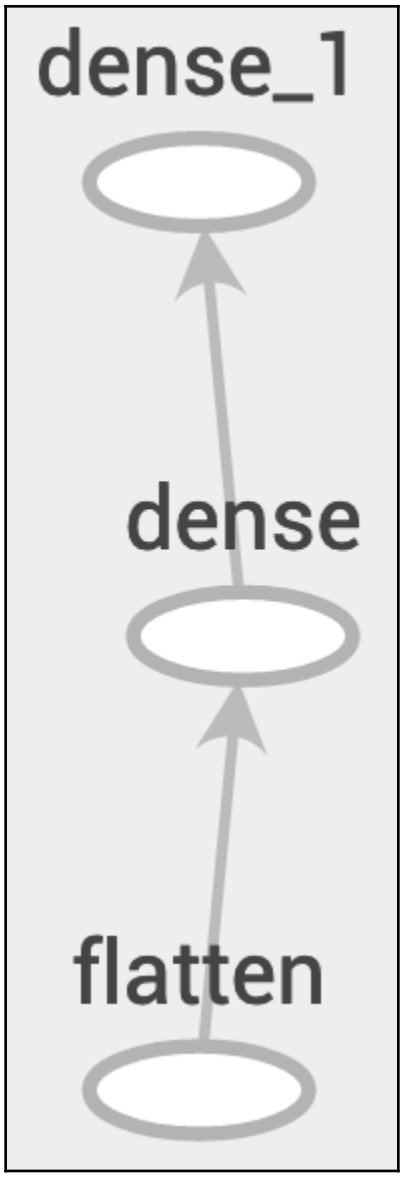

Loss L
1. Current loss of the network
2. Negative gradient of the loss for this particular parameter ("slope")
L_i
L_{i+1}
$\frac{-dL_i}{dP}$
3. Parameter update based on the gradient value
4. Loss minima to reach by repeating this "gradient descent" to optimize P (and the other parameters)
$P_i \rightarrow P_{i+1}$
Parameter P

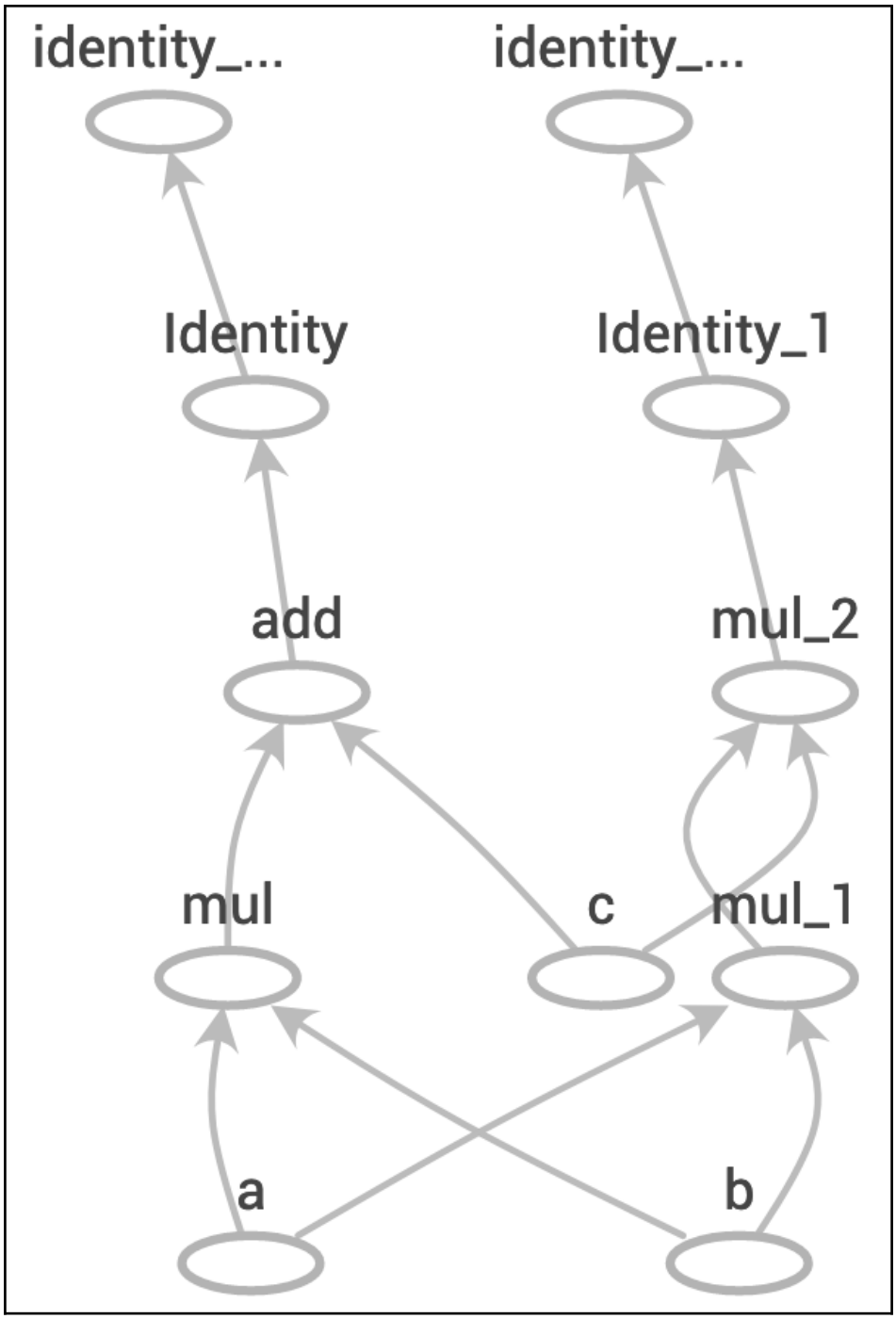

Y
X
under-fitting
Y
X
target fitting
Y
X
over-fitting

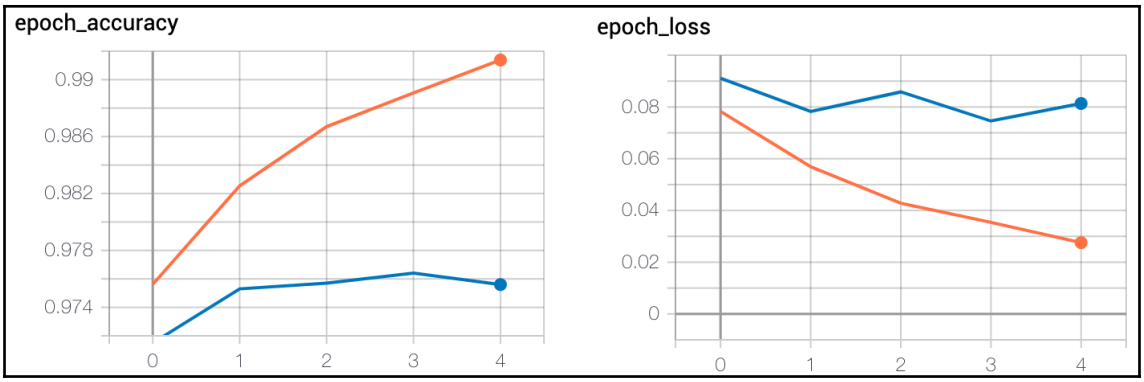
Chapter 2: TensorFlow Basics and Training a Model

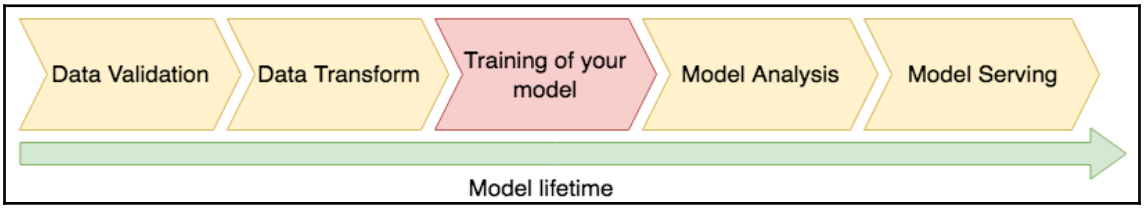



```
1952/60000 [..............................] - ETA: 6:46 - loss: 0.9248 - acc: 0.6962
```

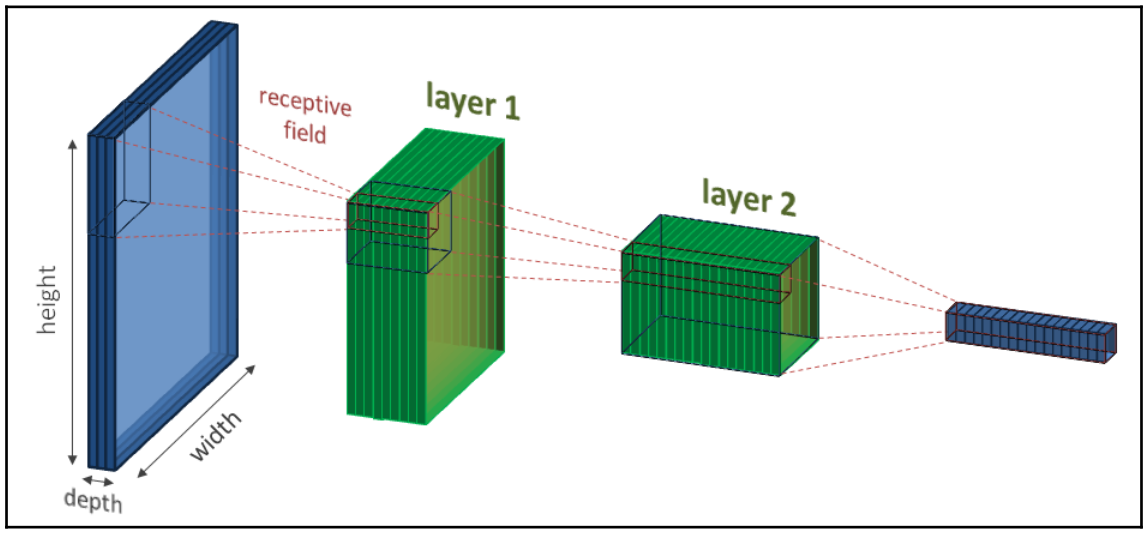

dense_1
dense
flatten

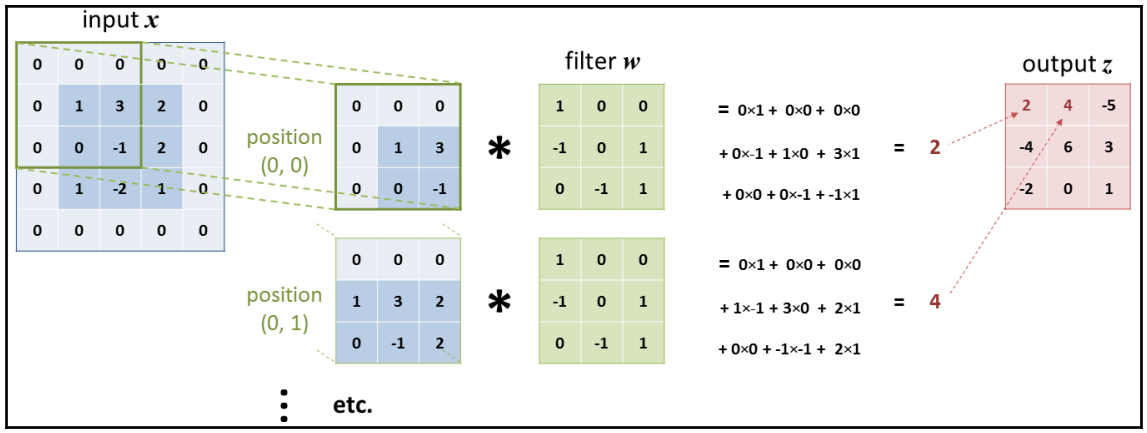

identity_...
identity_...
Identity
Identity_1
add
mul_2
mul
c
mul_1
a
b

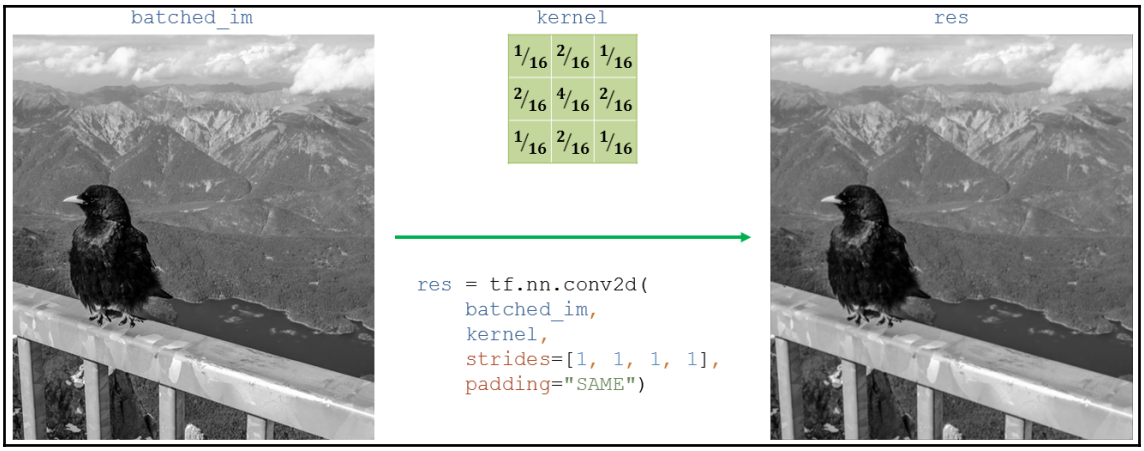

epoch_accuracy
0.99
0.986
0.982
0.978
0.974
0
1
2
3
4
epoch_loss
0.08
0.06
0.04
0.02
0
0
1
2
3
4

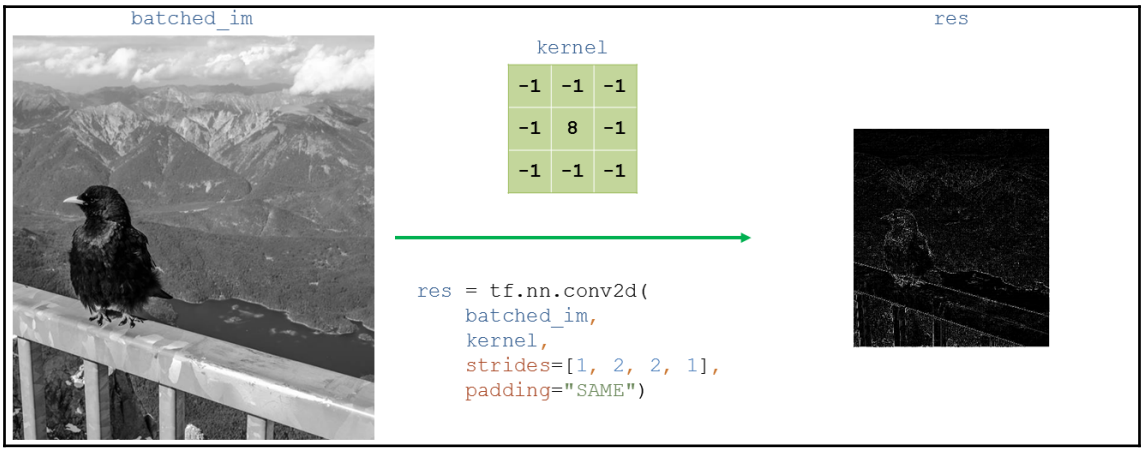

Data Validation
Data Transform
Training of your model
Model Analysis
Model Serving
Model lifetime

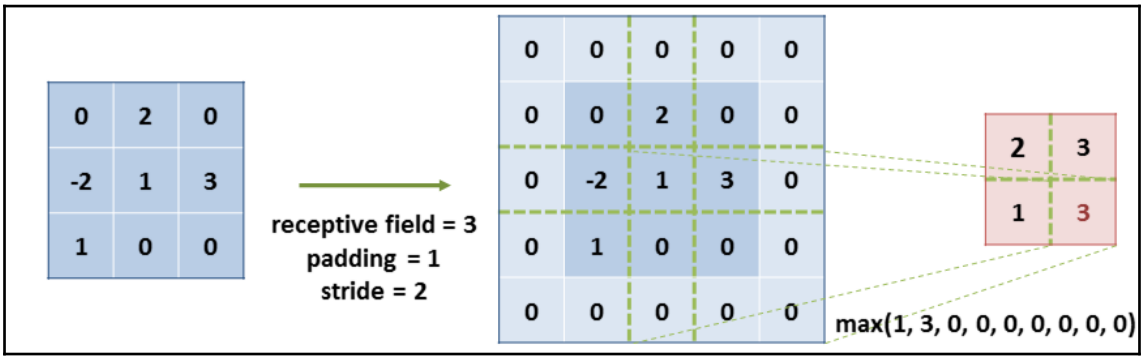
Chapter 3: Modern Neural Networks

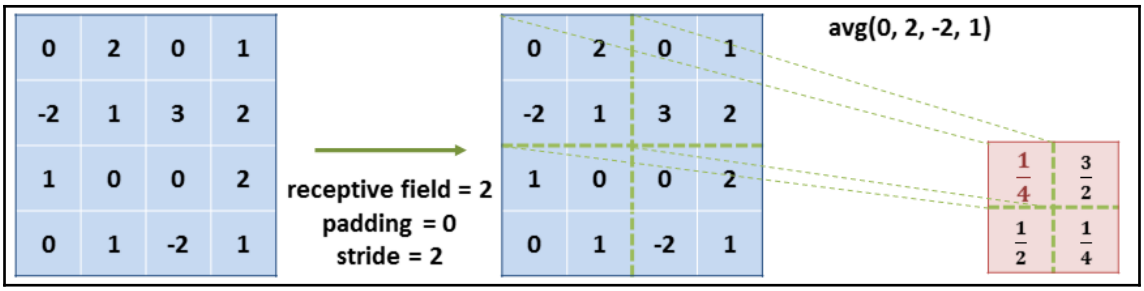





batched_im
kernel
res
1/16 2/16 1/16
2/16 4/16 2/16
1/16 2/16 1/16
res = tf.nn.conv2d(
batched_im,
kernel,
strides=[1, 1, 1, 1],
padding="SAME")

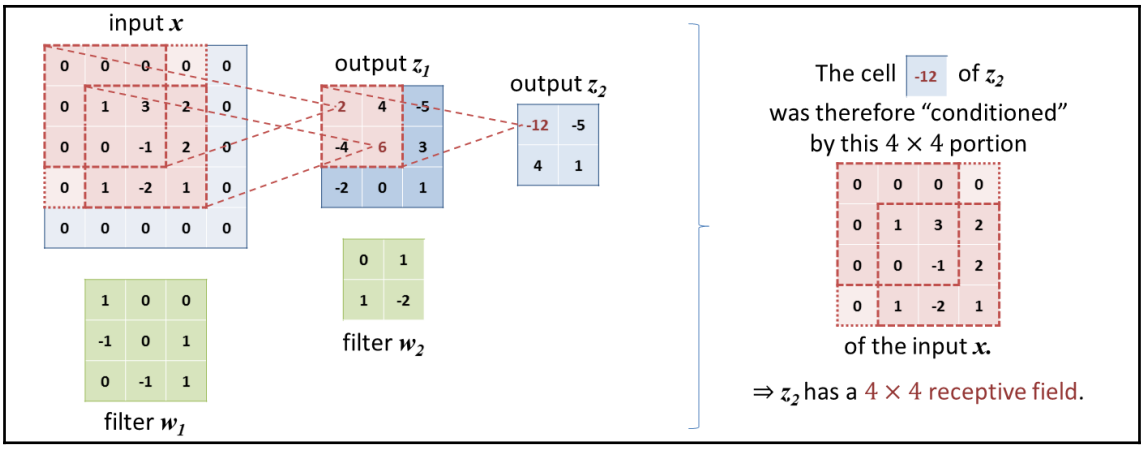

batched_im
kernel
res
-1 -1 -1
-1 8 -1
-1 -1 -1
res = tf.nn.conv2d(
batched_im,
kernel,
strides=[1, 2, 2, 1],
padding="SAME")

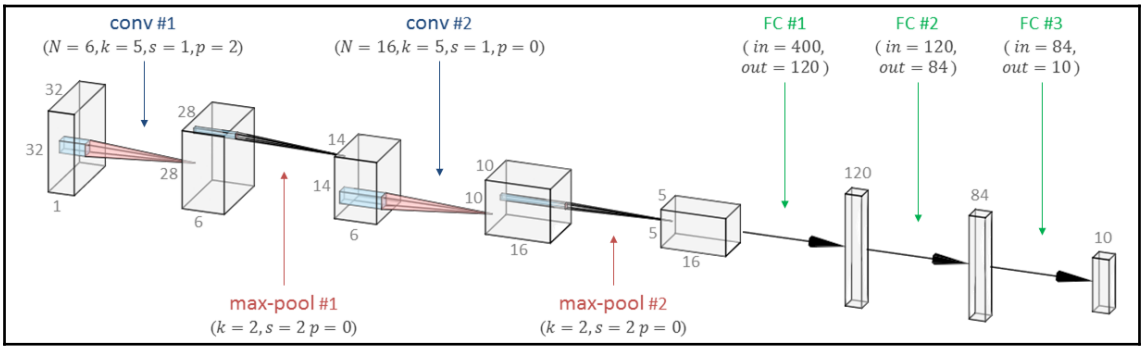

0 2 0
-2 1 3
1 0 0
receptive field = 3
padding = 1
stride = 2
0 0 0 0 0
0 0 2 0 0
0 -2 1 3 0
0 1 0 0 0
0 0 0 0 0
2 3
1 3
max(1, 3, 0, 0, 0, 0, 0, 0, 0)

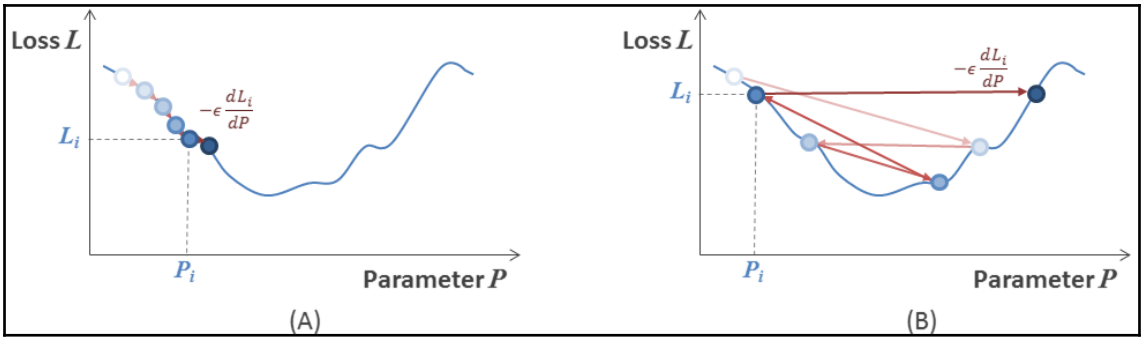

0 2 0 1
-2 1 3 2
1 0 0 2
0 1 -2 1
receptive field = 2
padding = 0
stride = 2
avg(0, 2, -2, 1)
1/4 3/2
1/2 1/4

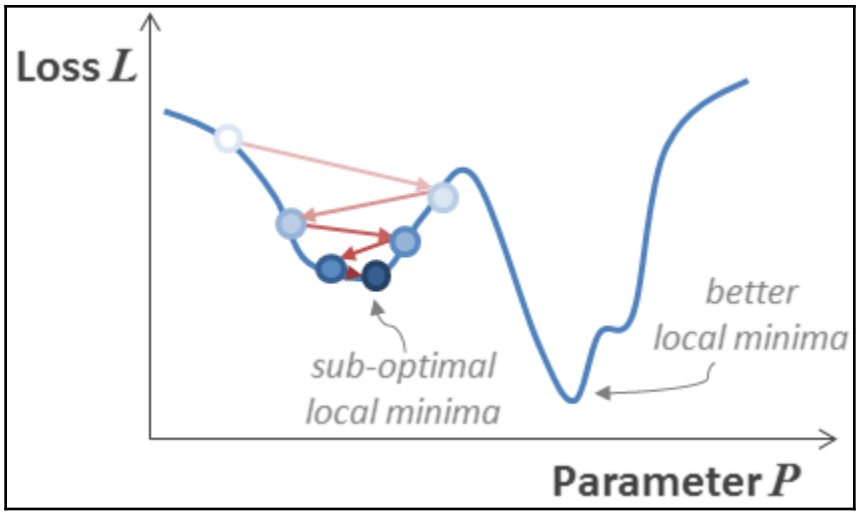
batched_im
res
res = tf.nn.avg_pool(
batched_im,
ksize=[1, 2, 2, 1],
strides=[1, 2, 2, 1],
padding="SAME")

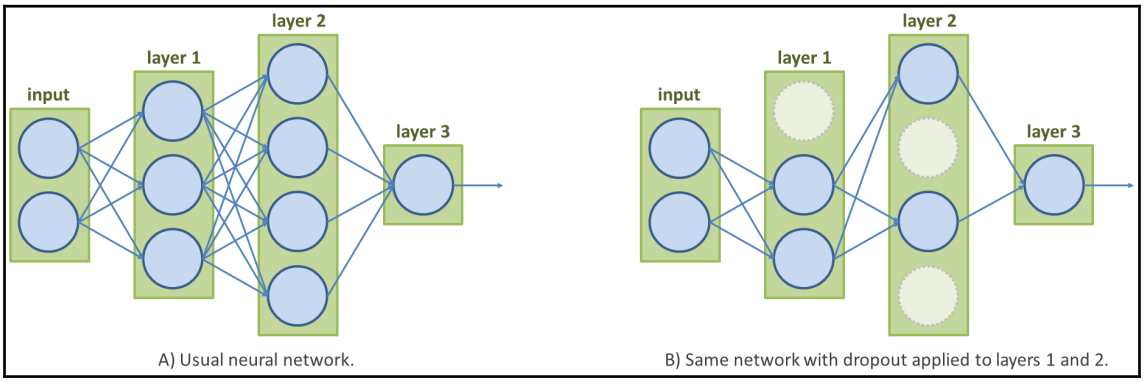
batched_im
res
res = tf.nn.max_pool(
batched_im,
ksize=[1, 10, 10, 1],
strides=[1, 2, 2, 1],
padding="SAME")


input x
output z_1
output z_2
filter w_1
filter w_2
The cell -12 of z_2
was therefore "conditioned"
by this 4 × 4 portion
of the input x.
⇒ z_2 has a 4 × 4 receptive field.


conv #1
(N = 6, k = 5, s = 1, p = 2)
conv #2
(N = 16, k = 5, s = 1, p = 0)
FC #1
(in = 400, out = 120)
FC #2
(in = 120, out = 84)
FC #3
(in = 84, out = 10)
max-pool #1
(k = 2, s = 2 p = 0)
max-pool #2
(k = 2, s = 2 p = 0)


Loss L
L_i
$-\epsilon \frac{dL_i}{dP}$
P_i
Parameter P
(A)
Loss L
L_i
$-\epsilon \frac{dL_i}{dP}$
P_i
Parameter P
(B)



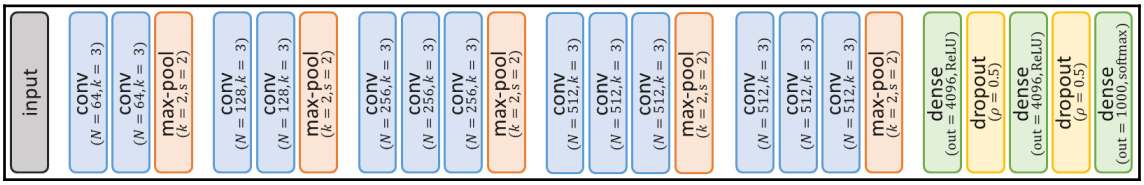


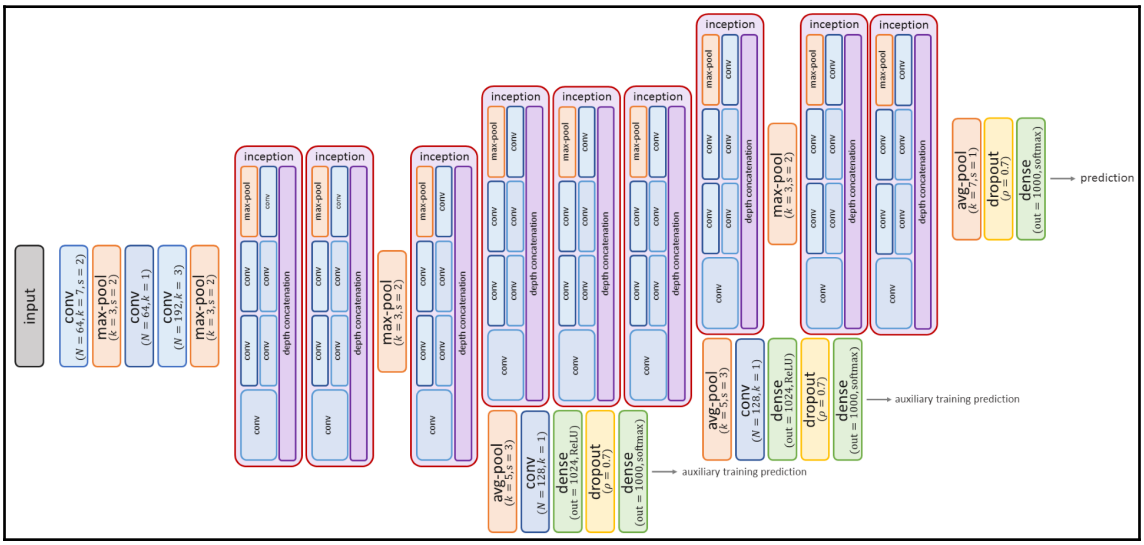
A) Usual neural network.

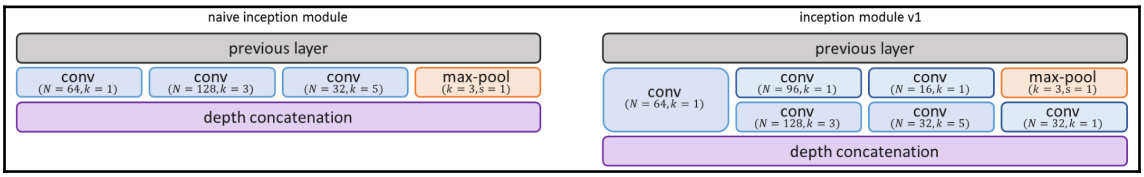
B) Same network with dropout applied to layers 1 and 2.

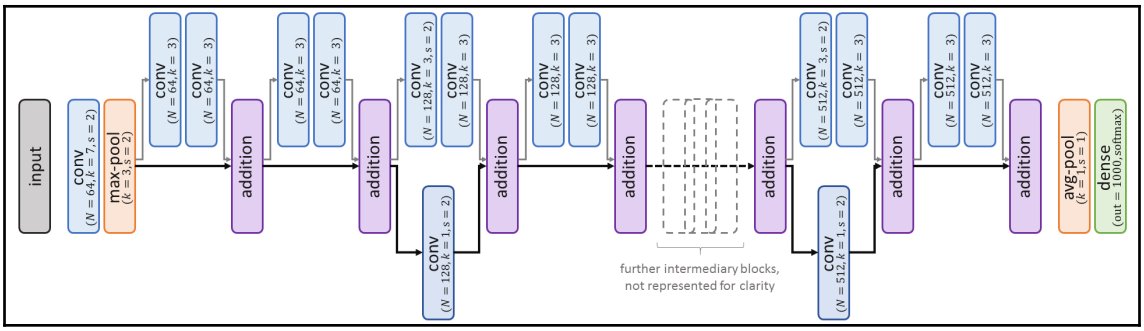
$$\rho = 0.2$$

Chapter 4: Influential Classification Tools

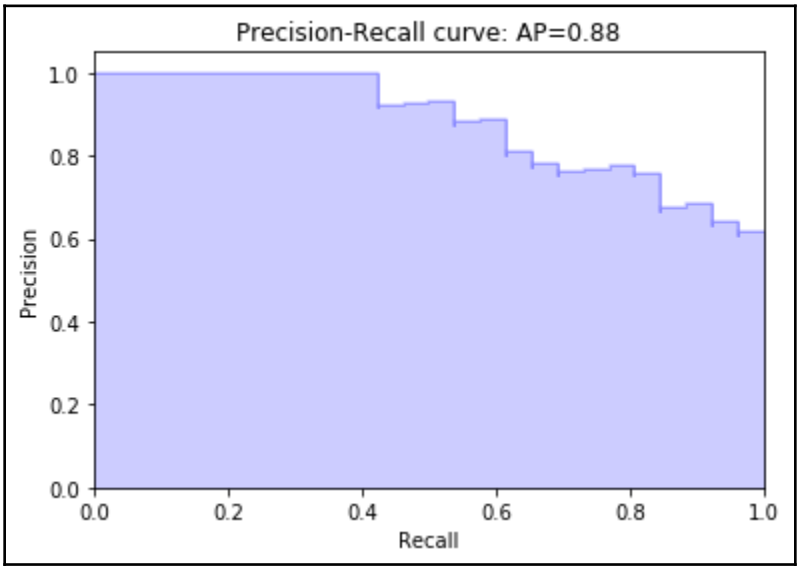


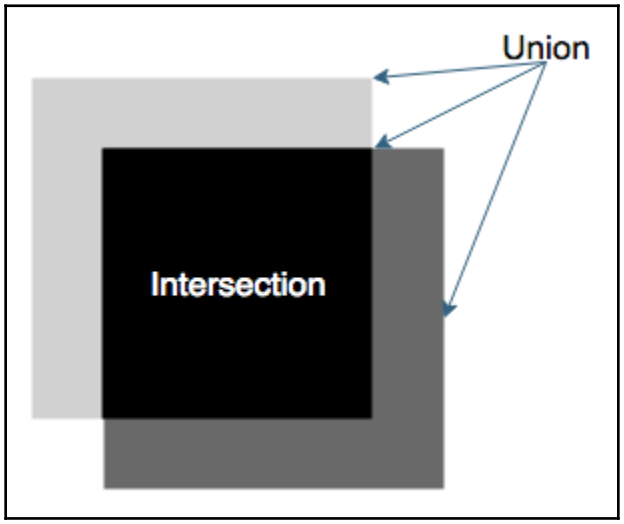


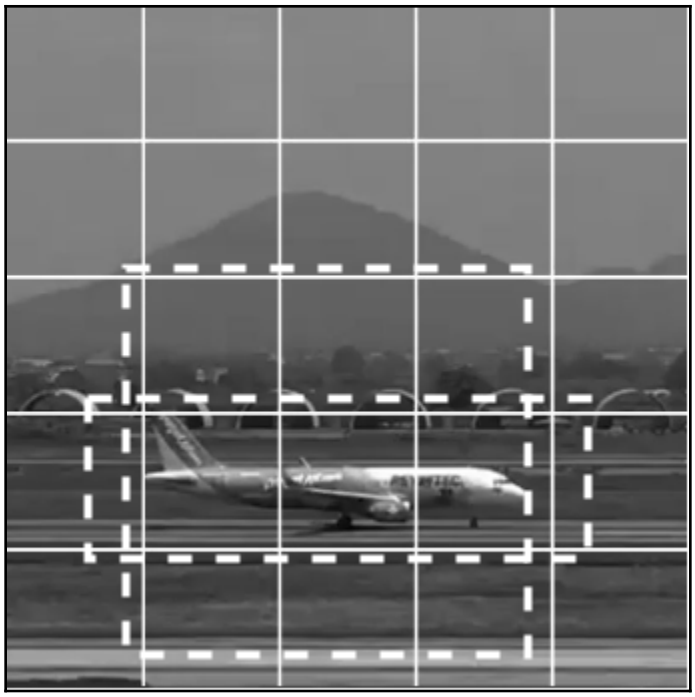


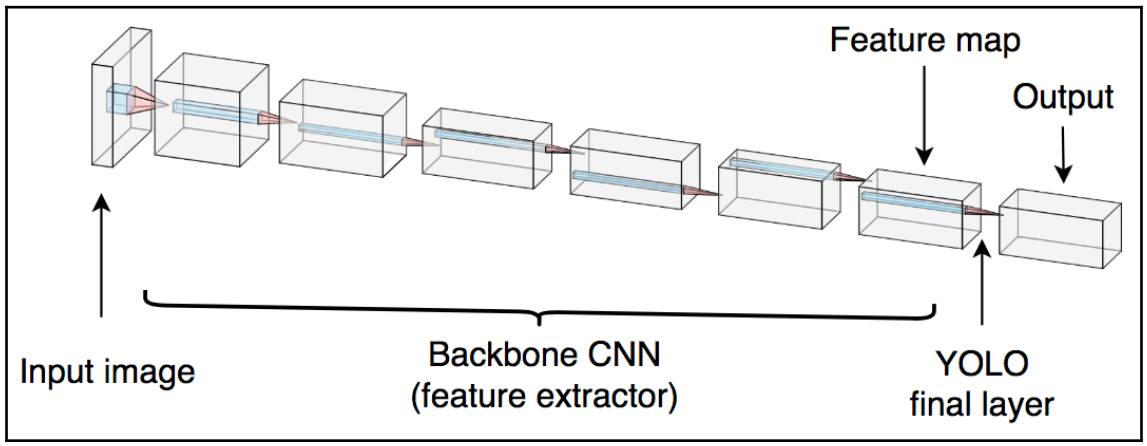

input
conv (N = 64, k = 7, s = 2)
max-pool (k = 3, s = 2)
conv (N = 64, k = 3)
conv (N = 64, k = 3)
addition
conv (N = 64, k = 3)
conv (N = 64, k = 3)
addition
conv (N = 128, k = 3, s = 2)
conv (N = 128, k = 3)
conv (N = 128, k = 1, s = 2)
addition
conv (N = 128, k = 3)
conv (N = 128, k = 3)
addition
further intermediary blocks, not represented for clarity
addition
conv (N = 512, k = 3, s = 2)
conv (N = 512, k = 3)
conv (N = 512, k = 1, s = 2)
addition
conv (N = 512, k = 3)
conv (N = 512, k = 3)
addition
avg-pool (k = 1, s = 1)
dense (out = 1000, softmax)

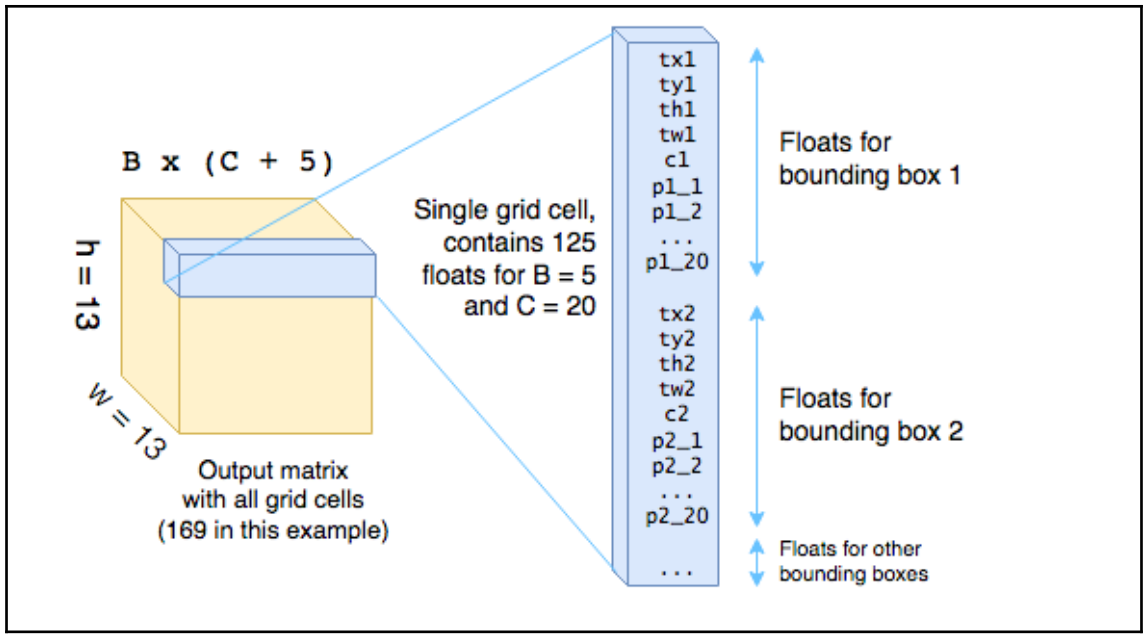
Chapter 5: Object Detection Models

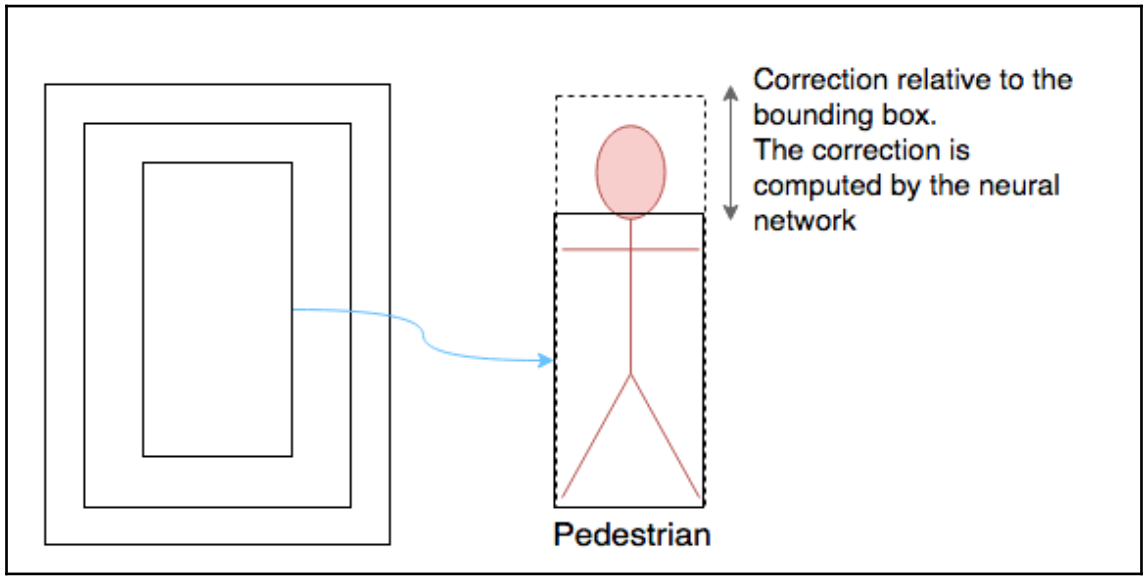


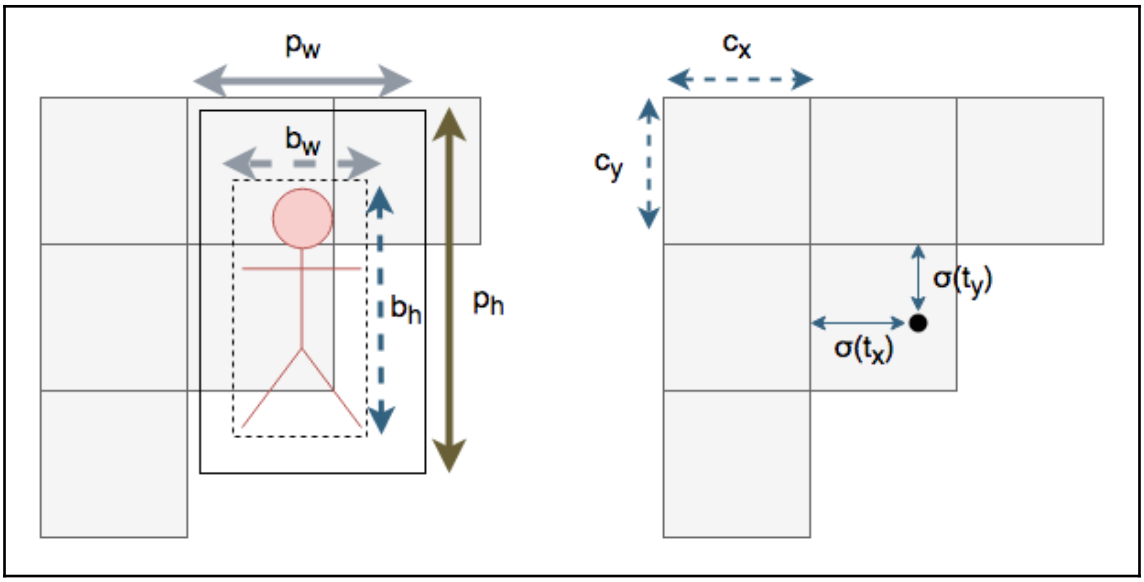


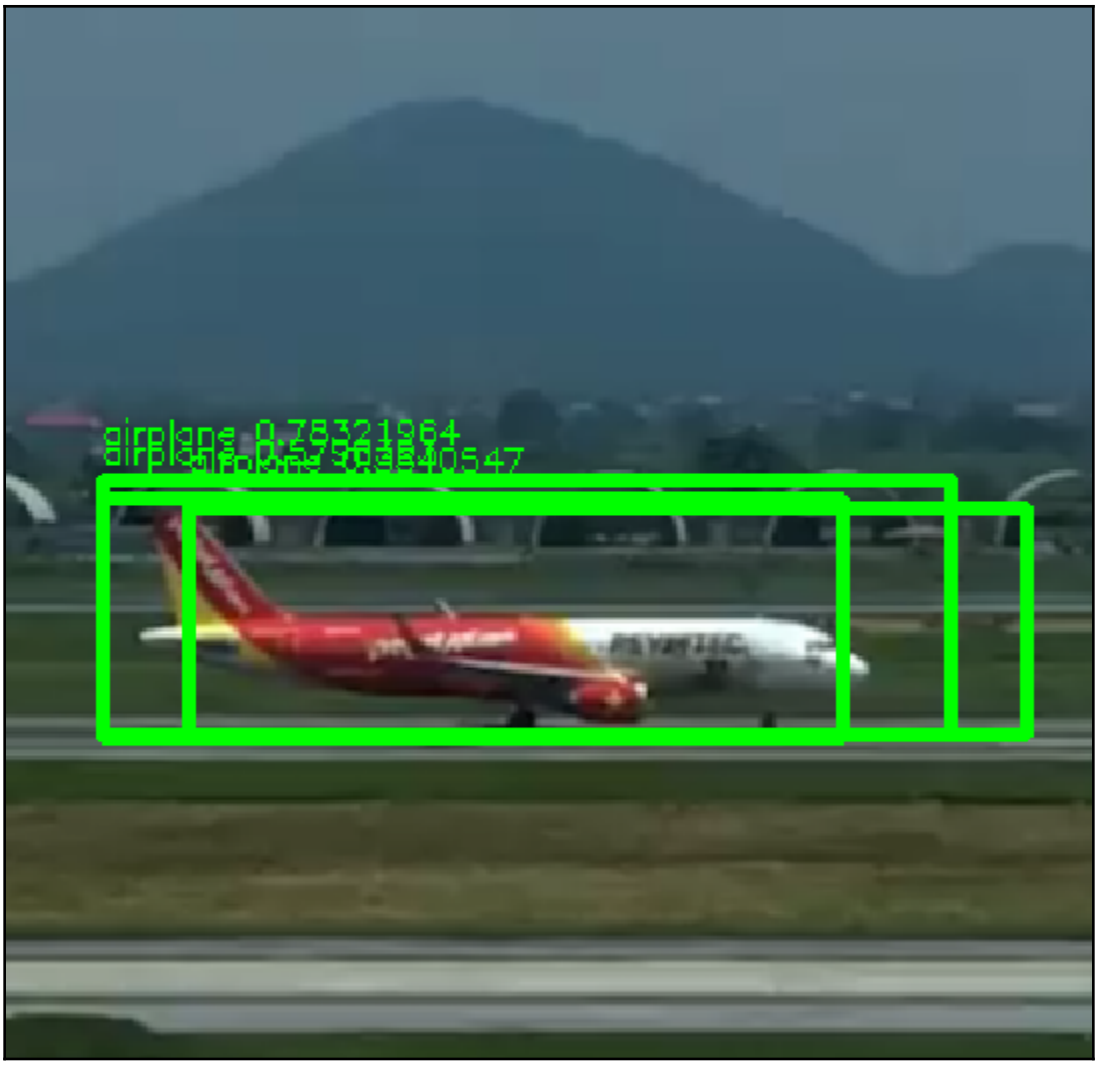



Feature map
Output
Input image
Backbone CNN
(feature extractor)
YOLO
final layer

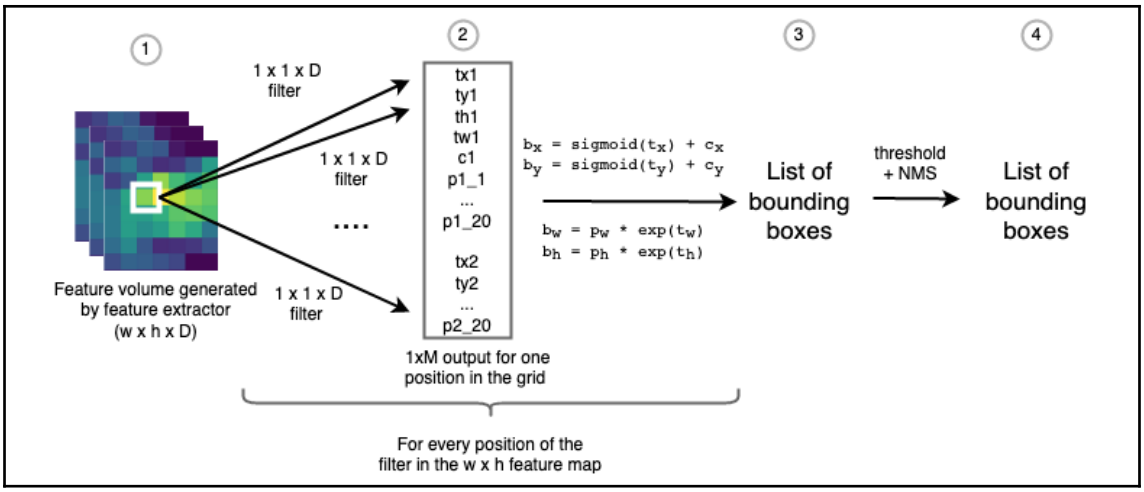

B x (C + 5)
h = 13
w = 13
Output matrix
with all grid cells
(169 in this example)
Single grid cell,
contains 125
floats for B = 5
and C = 20
tx1
ty1
th1
tw1
c1
p1_1
p1_2
...
p1_20
tx2
ty2
th2
tw2
c2
p2_1
p2_2
...
p2_20
...
Floats for
bounding box 1
Floats for
bounding box 2
Floats for other
bounding boxes

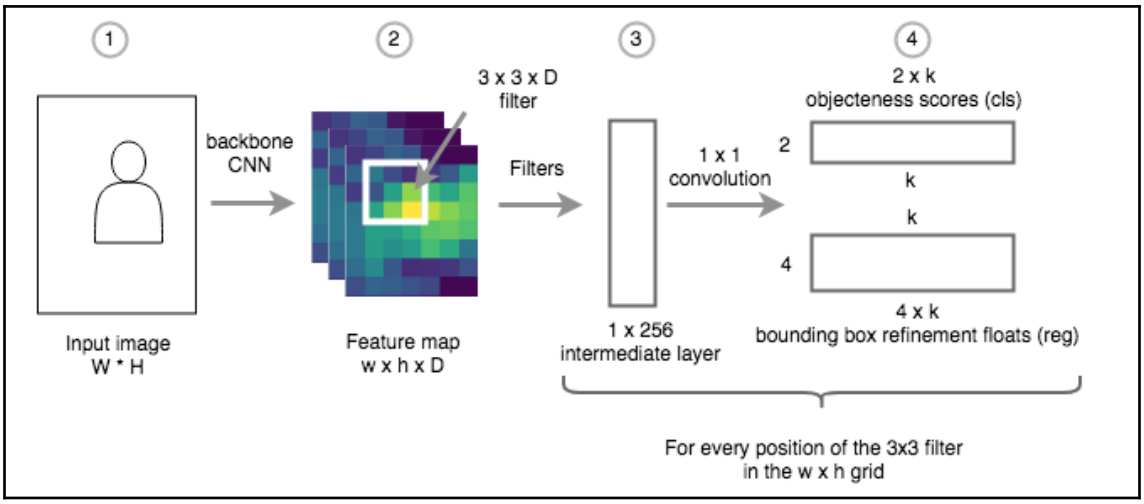

Correction relative to the
bounding box.
The correction is
computed by the neural
network
Pedestrian

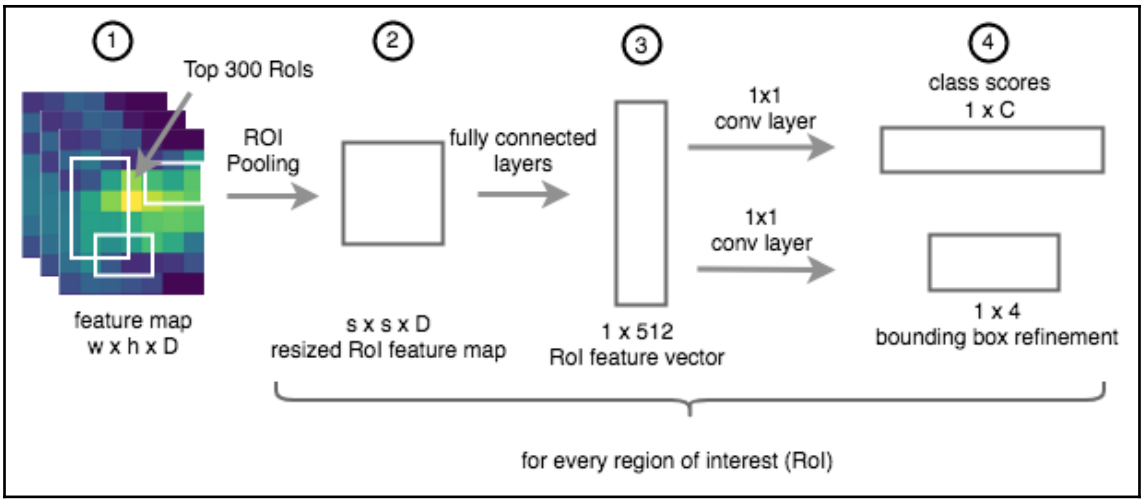

p_w
b_w
b_h
p_h
c_x
c_y
$\sigma(t_y)$
$\sigma(t_x)$


airplane 0.78321964

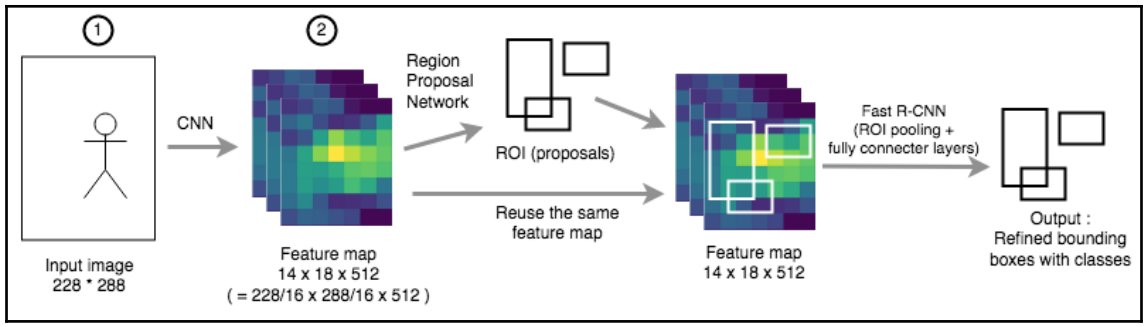
airplane 0.78321964


1
2
3
4
1 x 1 x D filter
1 x 1 x D filter
1 x 1 x D filter
Feature volume generated by feature extractor (w x h x D)
tx1
ty1
th1
tw1
c1
p1_1
...
p1_20
tx2
ty2
...
p2_20
1xM output for one position in the grid
bx = sigmoid(tx) + cx
by = sigmoid(ty) + cy
bw = pw * exp(tw)
bh = ph * exp(th)
List of bounding boxes
threshold + NMS
List of bounding boxes
For every position of the filter in the w x h feature map

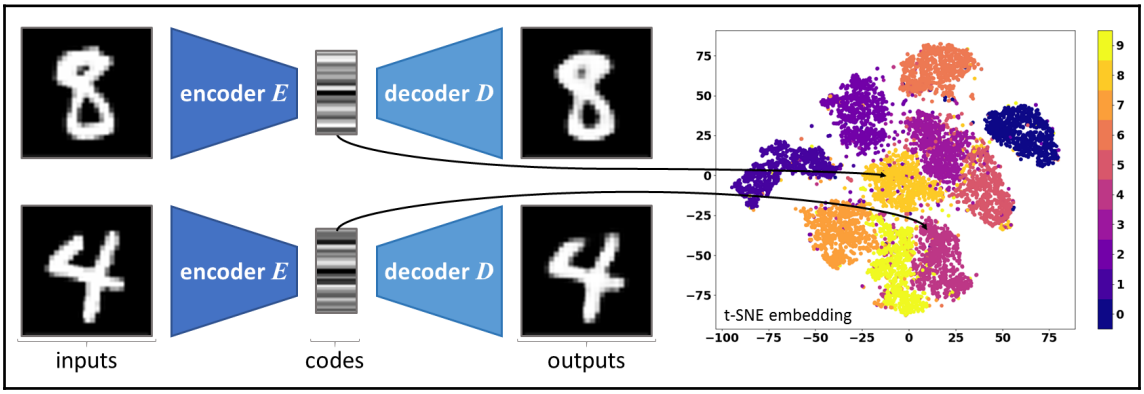

1
2
3
4
Input image W * H
backbone CNN
Feature map w x h x D
3 x 3 x D filter
Filters
1 x 256 intermediate layer
1 x 1 convolution
2 x k objecteness scores (cls)
2
k
k
4
4 x k bounding box refinement floats (reg)
For every position of the 3x3 filter in the w x h grid

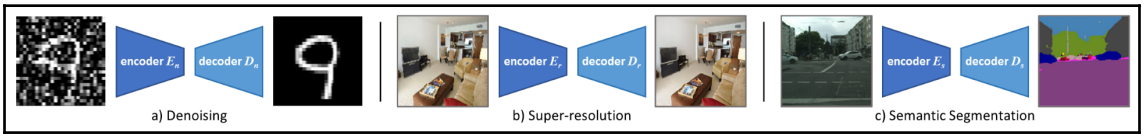


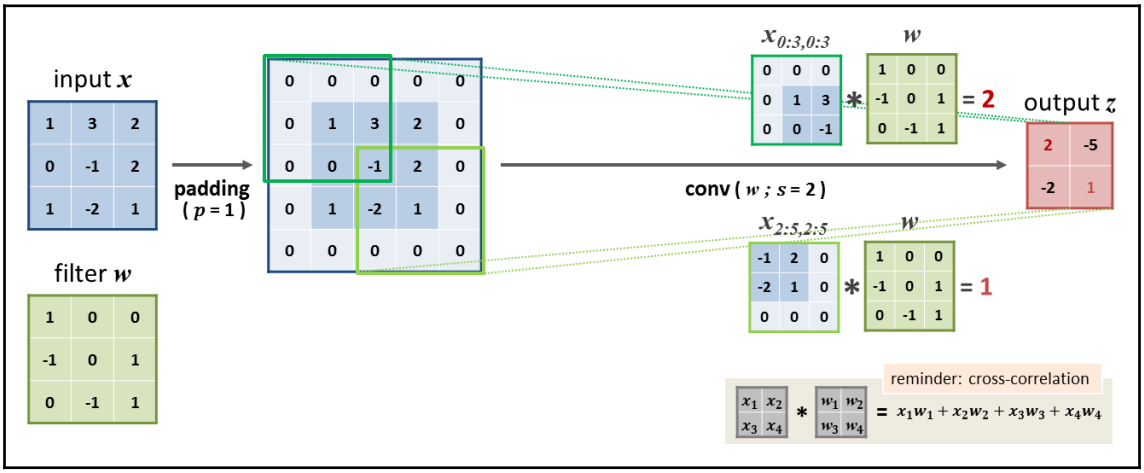
	A	B	C	D	E	F	G	H	I	J	K
1	0.16	1.00	0.26	0.11	0.14	0.90	0.06	0.15			
2	0.13	0.05	0.58	0.34	0.58	0.13	0.78	0.53			
3	0.89	0.35	0.38	0.65	0.01	0.56	0.97	0.06			
4	0.47	0.97	0.78	0.99	0.82	0.90	0.32	0.89		0.97	0.99
5	0.58	0.13	0.12	0.50	0.99	0.35	0.83	0.39		0.96	0.93
6	0.21	0.59	0.96	0.93	0.08	0.55	0.13	0.89			
7	0.03	0.83	0.63	0.46	0.09	0.03	0.68	0.13			
8	0.82	0.35	0.20	0.48	0.80	0.41	0.46	0.08			

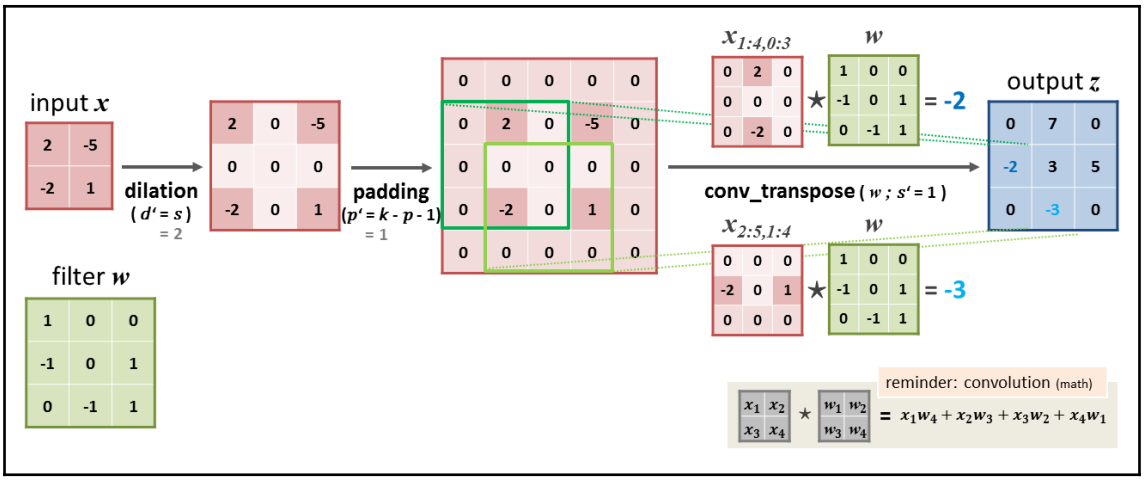

1
2
Input image
228 * 288
CNN
Feature map
14 x 18 x 512
(= 228/16 x 288/16 x 512)
Region
Proposal
Network
ROI (proposals)
Reuse the same
feature map
Feature map
14 x 18 x 512
Fast R-CNN
(ROI pooling +
fully connecter layers)
Output :
Refined bounding
boxes with classes

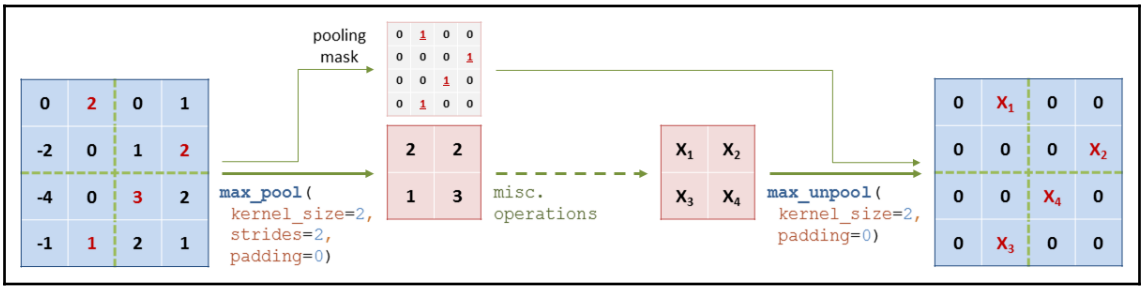
Chapter 6: Enhancing and Segmenting Images

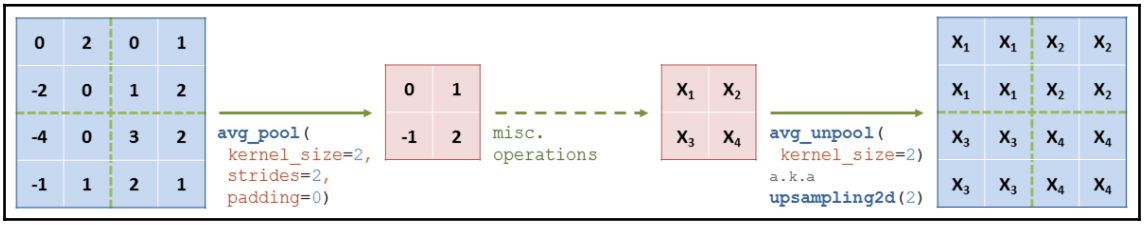


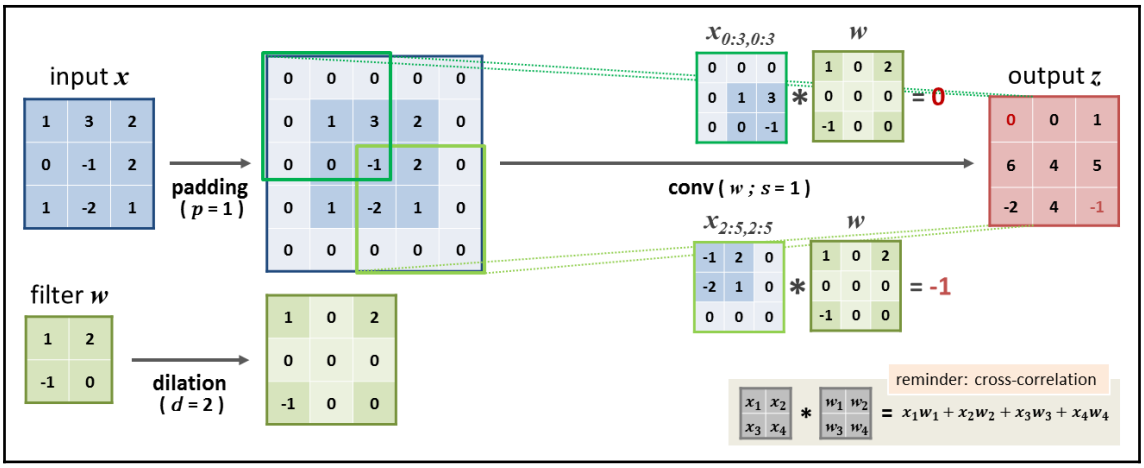


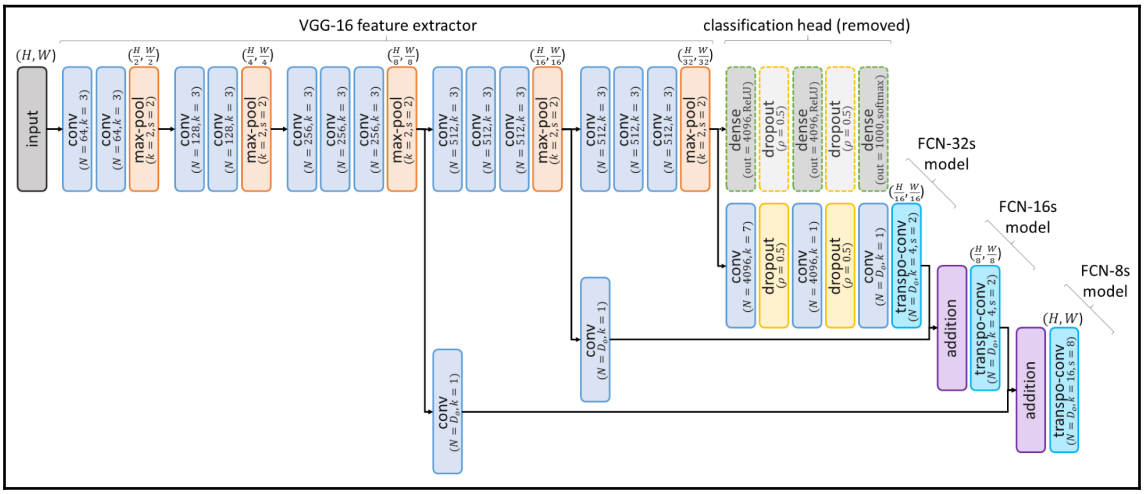


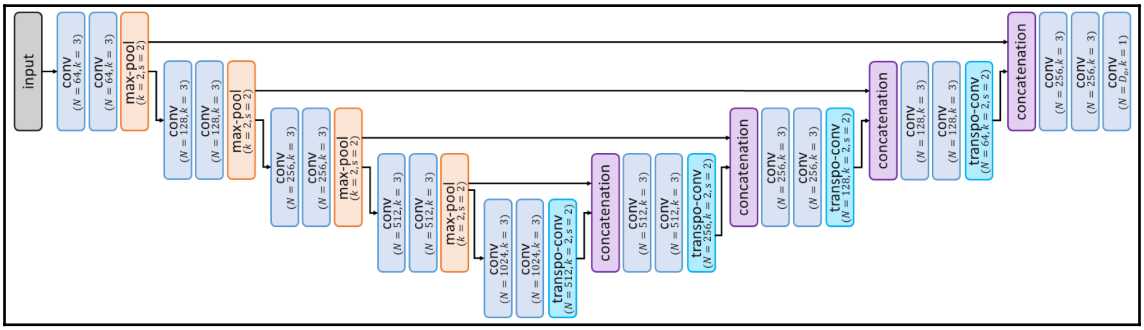

input x
dilation
(d' = s)
= 2
padding
(p' = k - p - 1)
= 1
filter w
conv_transpose (w ; s' = 1)
output z
reminder: convolution (math)

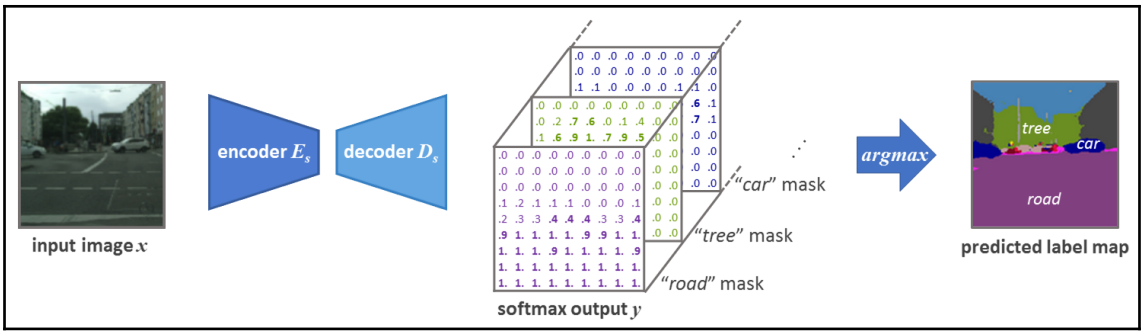

pooling
mask
max_pool(
kernel_size=2,
strides=2,
padding=0)
misc.
operations
max_unpool(
kernel_size=2,
padding=0)

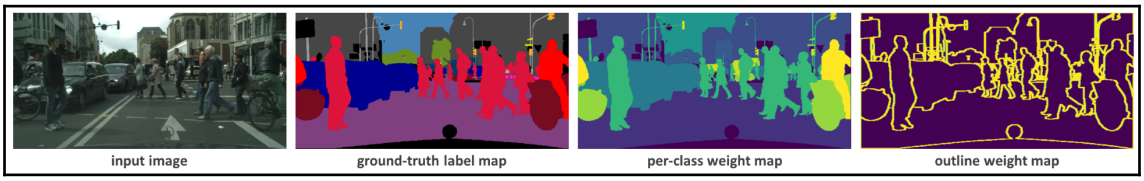

avg_pool(
kernel_size=2,
strides=2,
padding=0)
misc.
operations
avg_unpool(
kernel_size=2)
a.k.a
upsampling2d(2)

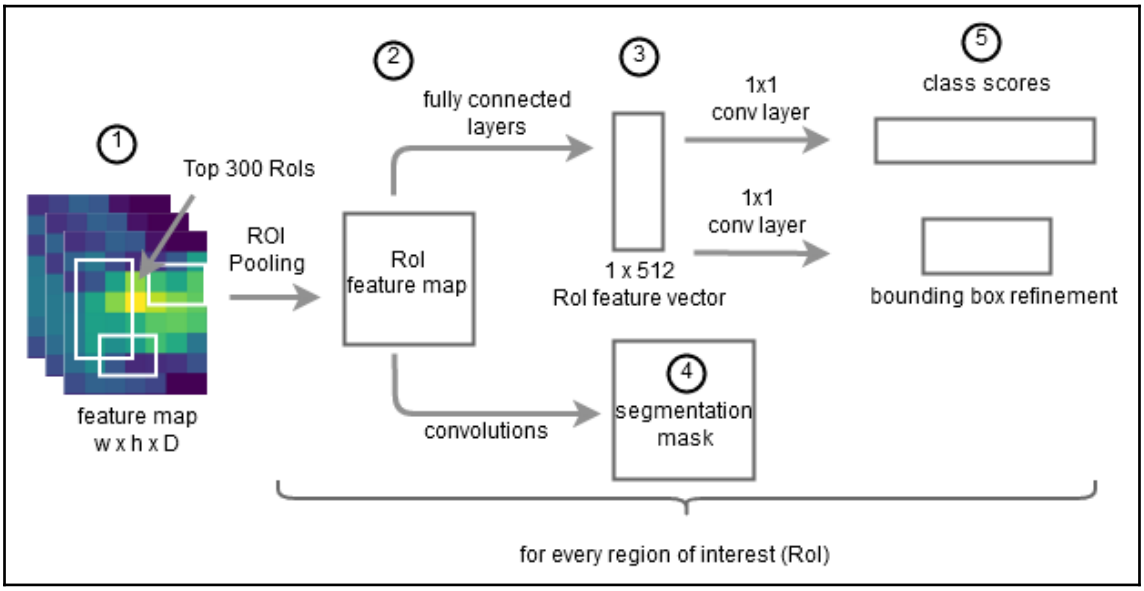

input x
padding ($p = 1$)
filter w
dilation ($d = 2$)
$x_{0:3,0:3}$
w
= 0
output z
conv (w ; $s = 1$)
$x_{2:5,2:5}$
w
= -1
reminder: cross-correlation
$x_1 x_2 x_3 x_4 * w_1 w_2 w_3 w_4 = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4$


VGG-16 feature extractor
classification head (removed)
FCN-32s model
FCN-16s model
FCN-8s model
input
addition
concatenation

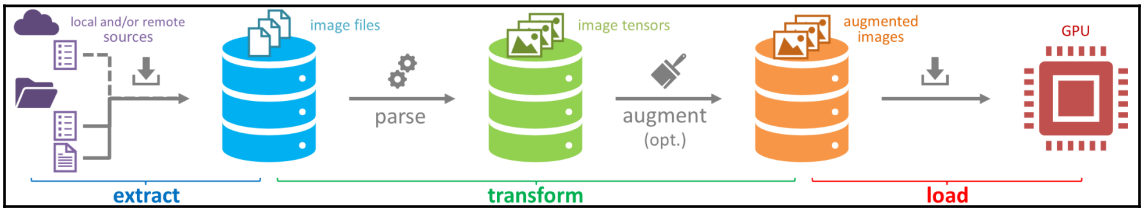

input
concatenation

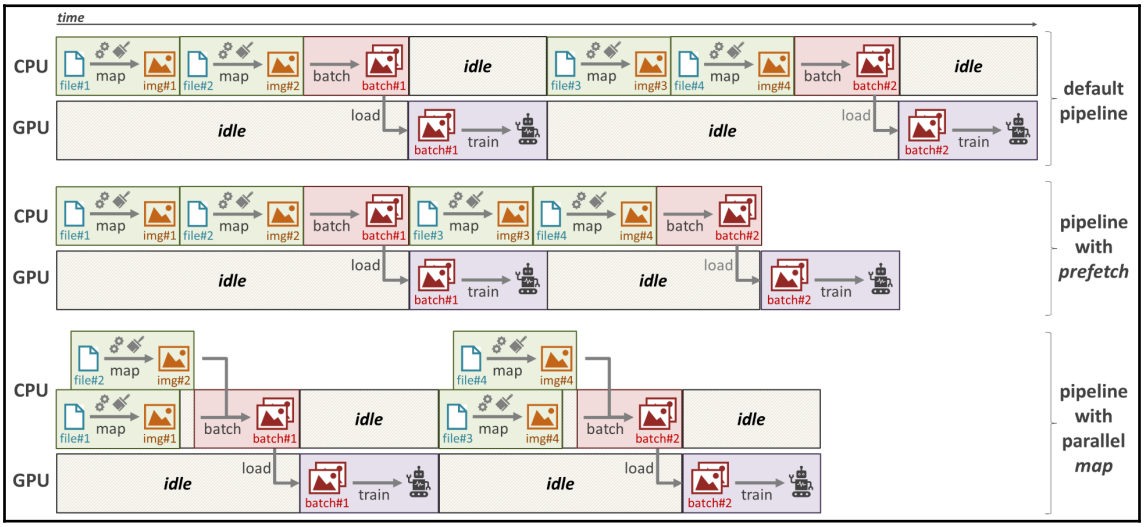

input image x
encoder E_s
decoder D_s
"car" mask
"tree" mask
"road" mask
softmax output y
argmax
tree
car
road
predicted label map

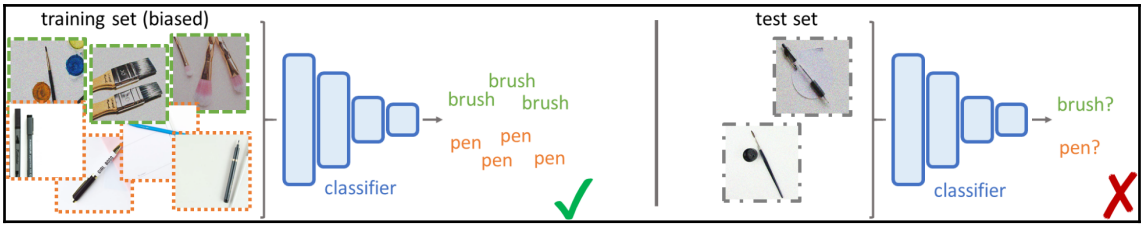

input image
ground-truth label map
per-class weight map
outline weight map

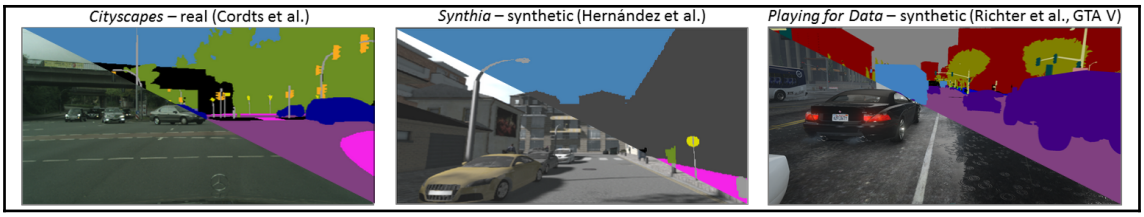

1
Top 300 RoIs
feature map
w x h x D
ROI
Pooling
2
fully connected
layers
RoI
feature map
convolutions
3
1 x 512
RoI feature vector
1x1
conv layer
1x1
conv layer
4
segmentation
mask
5
class scores
bounding box refinement
for every region of interest (RoI)

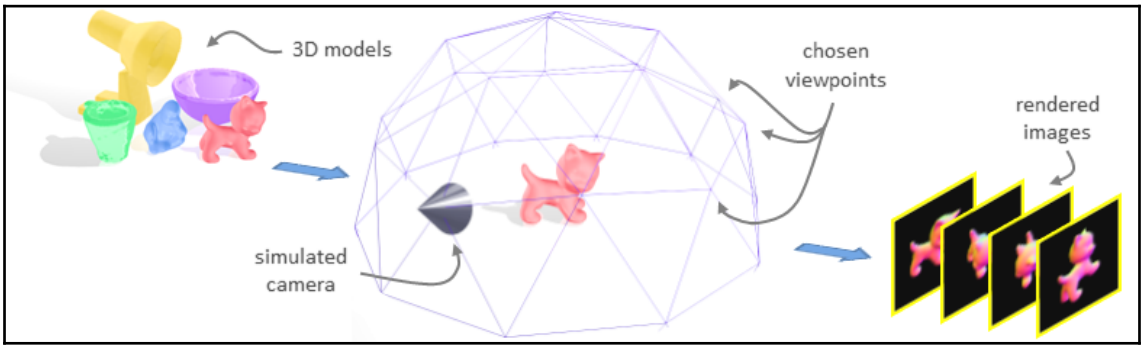
Chapter 7: Training on Complex and Scarce Datasets

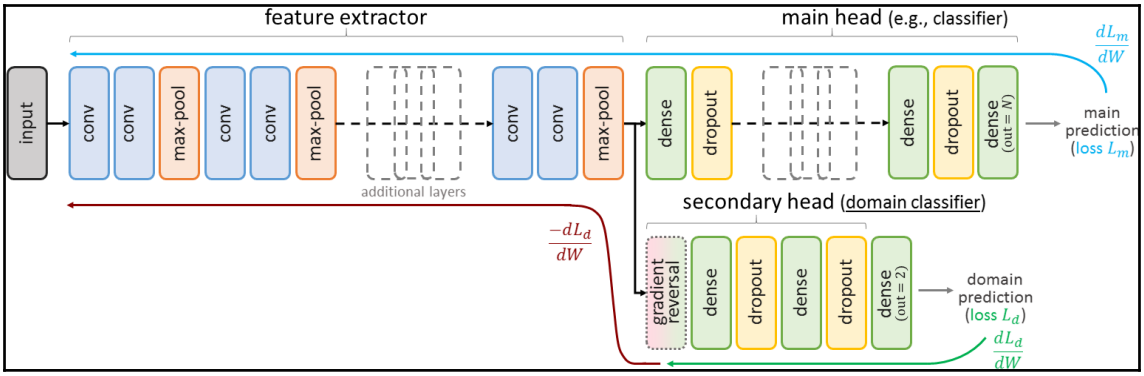


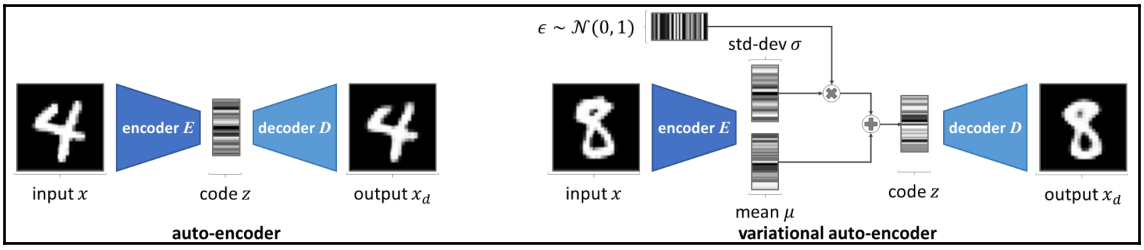


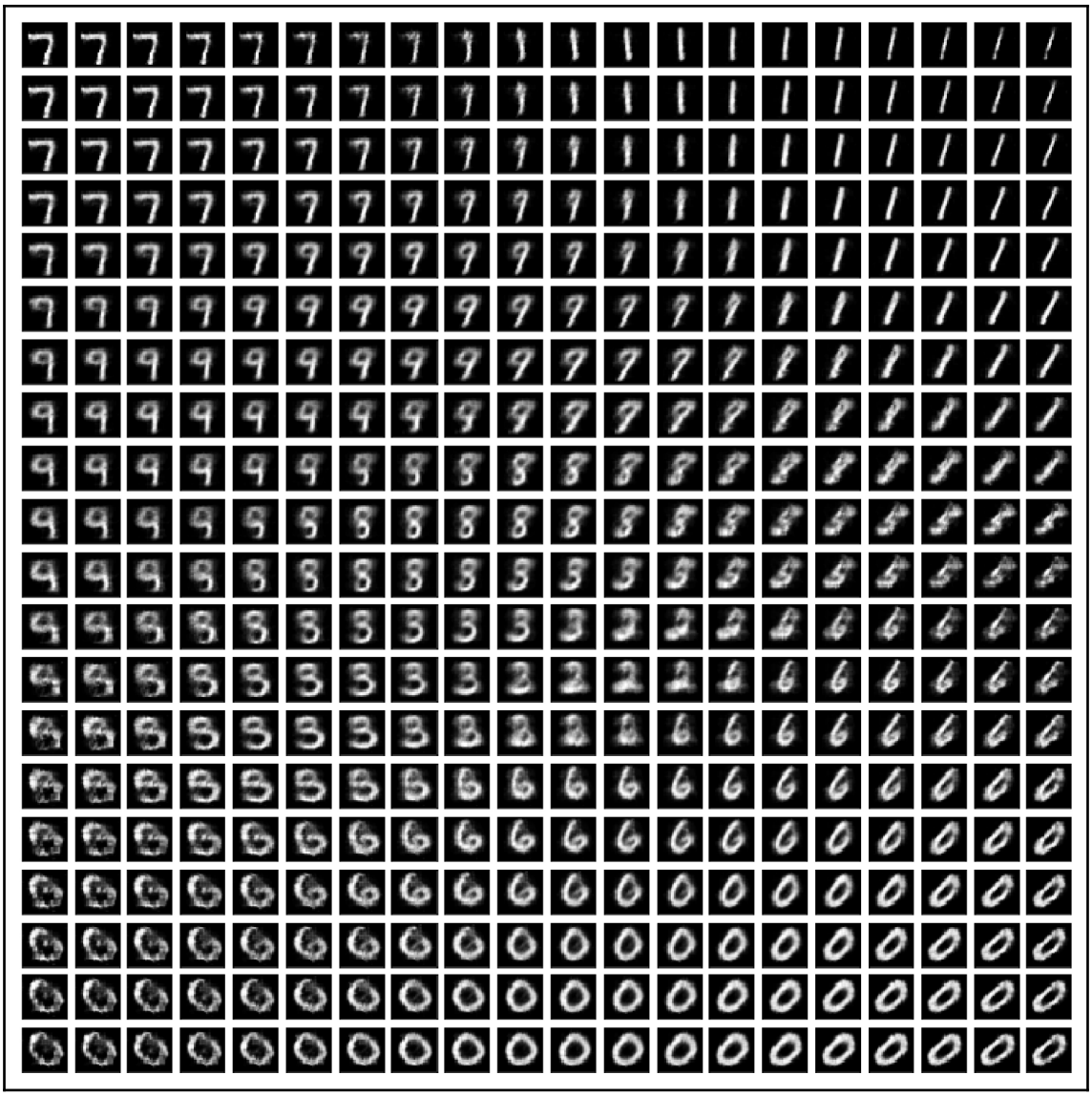


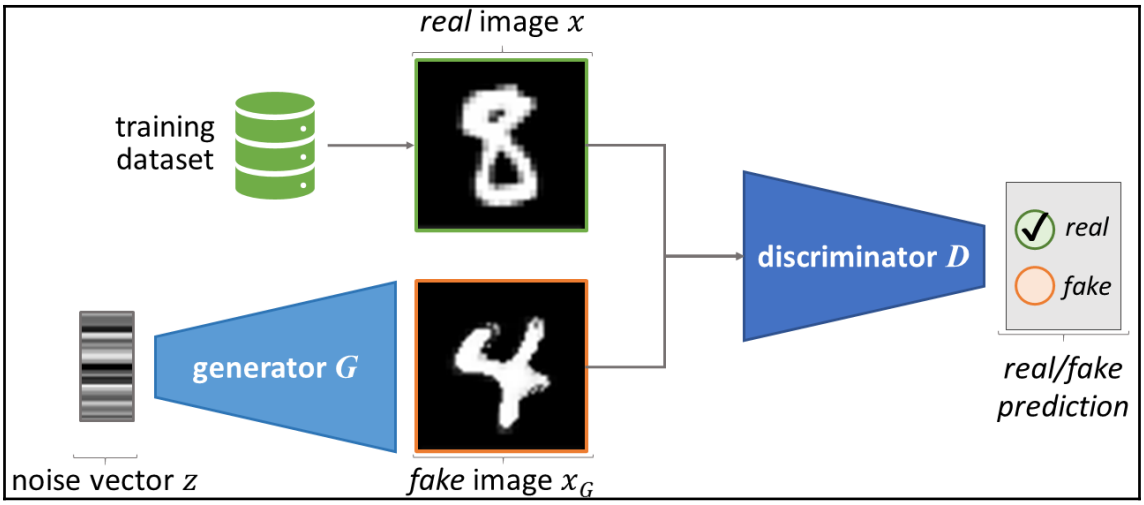

Cityscapes – real (Cordts et al.)
Synthia – synthetic (Hernández et al.)
Playing for Data – synthetic (Richter et al., GTA V)


3D models
chosen viewpoints
rendered images
simulated camera

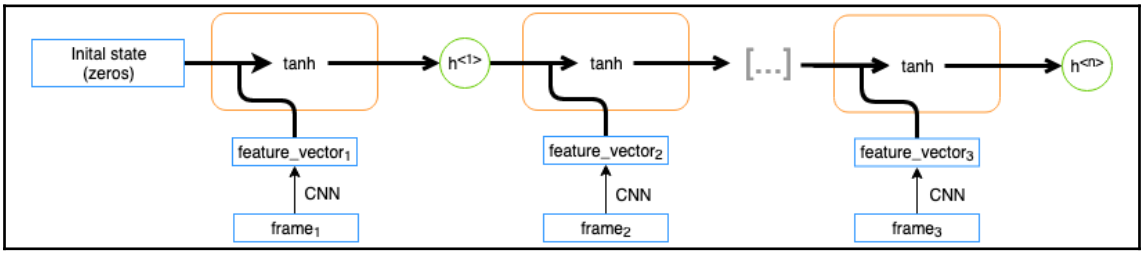

feature extractor
main head (e.g., classifier)
input
conv
conv
max-pool
conv
conv
max-pool
additional layers
conv
conv
max-pool
dense
dropout
dense
dropout
dense (out = N)
main prediction (loss L_m)
$\frac{dL_m}{dW}$
secondary head (domain classifier)
$\frac{-dL_d}{dW}$
gradient reversal
dense
dropout
dense
dropout
dense (out = 2)
domain prediction (loss L_d)
$\frac{dL_d}{dW}$

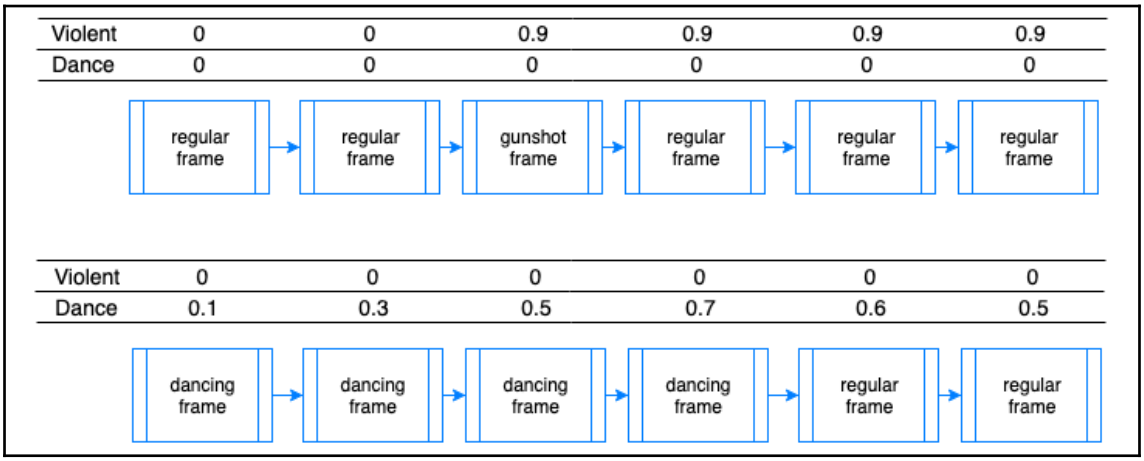

$\epsilon \sim \mathcal{N}(0,1)$
std-dev σ
encoder E
decoder D
input x
code z
output x_d
auto-encoder
encoder E
decoder D
input x
mean μ
code z
output x_d
variational auto-encoder

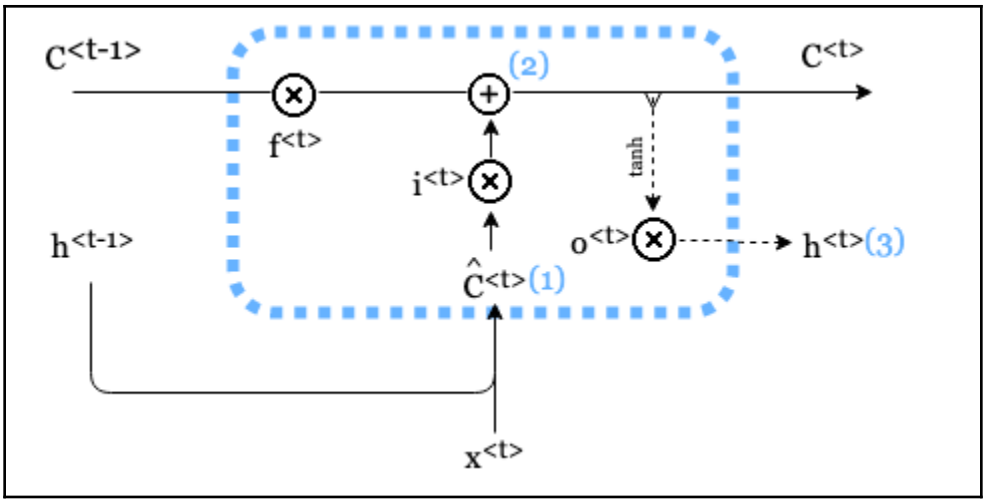



real image x
training dataset
discriminator D
real
fake
real/fake prediction
generator G
noise vector z
fake image x_G

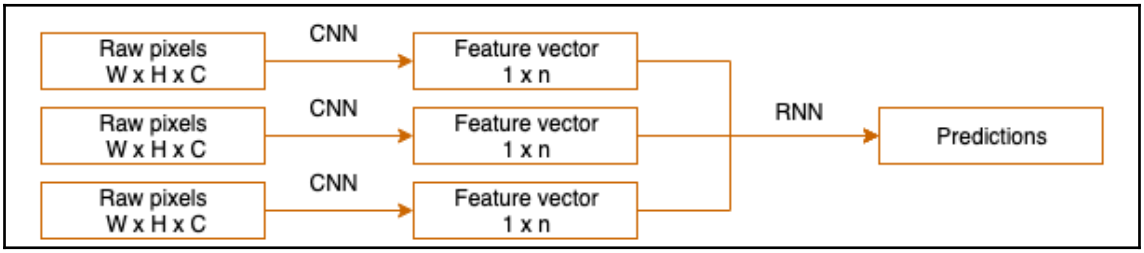
Chapter 8: Video and Recurrent Neural Networks

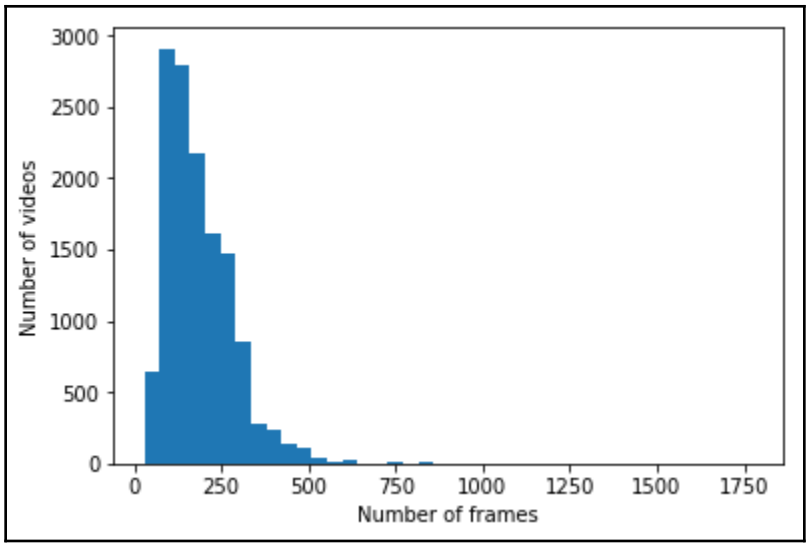


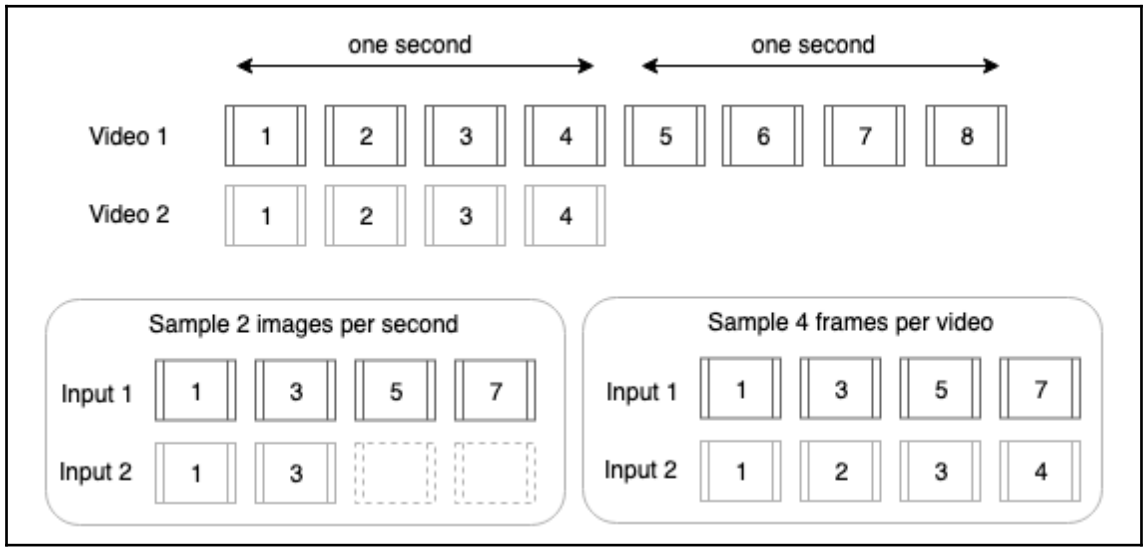


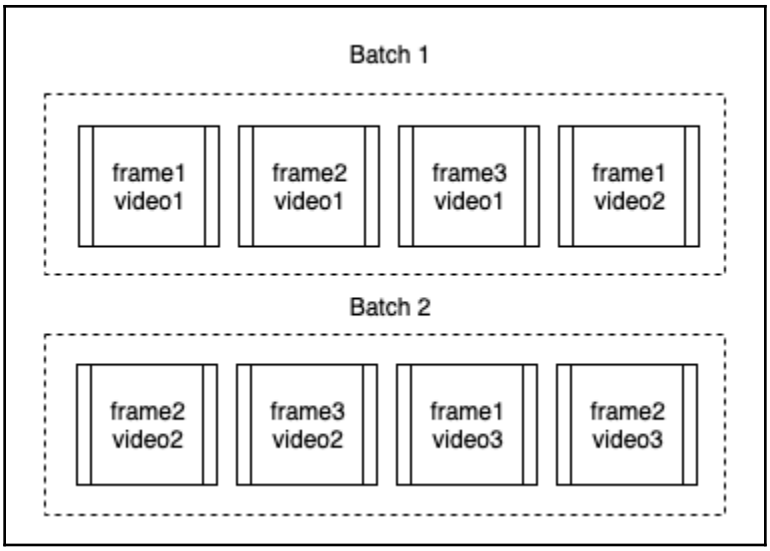

C<t-1>
C<t>
(2)
f<t>
i<t>
tanh
h<t-1>
o<t>
h<t>(3)
Ĉ<t>(1)
x<t>

Long Jump
Parallel Bars
Pole Vault
Pommel Horse
Punch
Rafting
Rowing
Jetski
Sky Diving
Soccer Penalty
Still Rings
Sumo Wrestling
Surfing
Table Tennis Shot

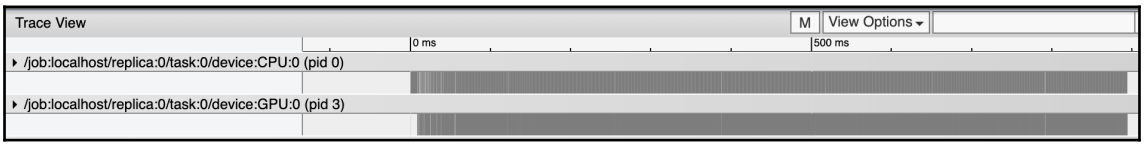

Raw pixels
W x H x C
CNN
Feature vector
1 x n
Raw pixels
W x H x C
CNN
Feature vector
1 x n
Raw pixels
W x H x C
CNN
Feature vector
1 x n
RNN
Predictions


3000
2500
2000
1500
1000
500
0
Number of videos
0
250
500
750
1000
1250
1500
1750
Number of frames

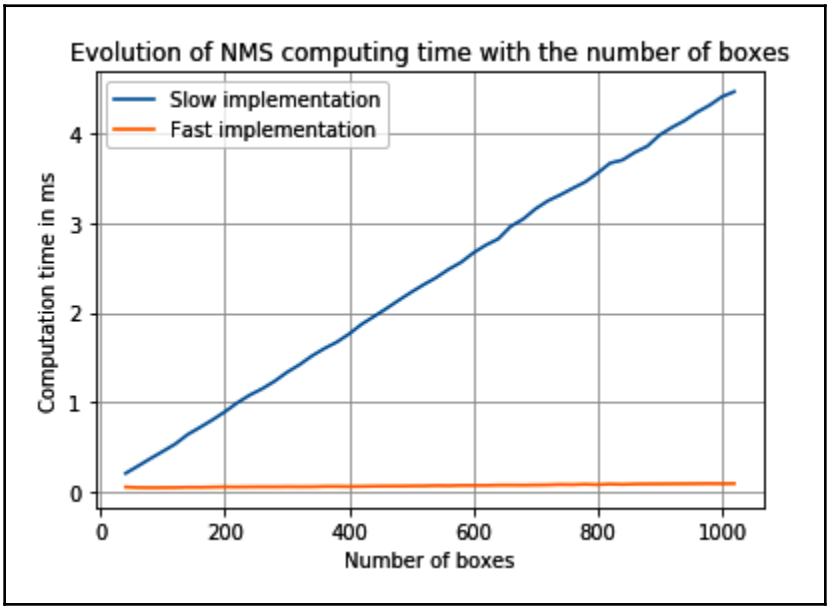

one second
one second
Video 1
1
2
3
4
5
6
7
8
Video 2
1
2
3
4
Sample 2 images per second
Input 1
1
3
5
7
Input 2
1
3
Sample 4 frames per video
Input 1
1
3
5
7
Input 2
1
2
3
4


Batch 1
frame1 video1
frame2 video1
frame3 video1
frame1 video2
Batch 2
frame2 video2
frame3 video2
frame1 video3
frame2 video3

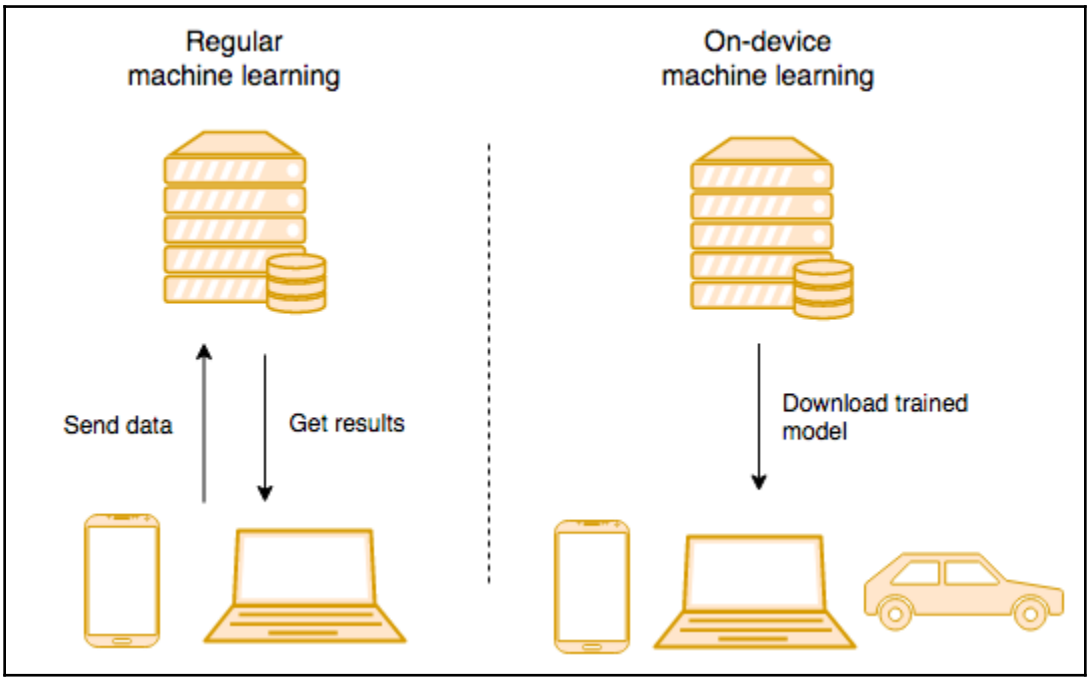
Chapter 9: Optimizing Models and Deploying on Mobile Devices



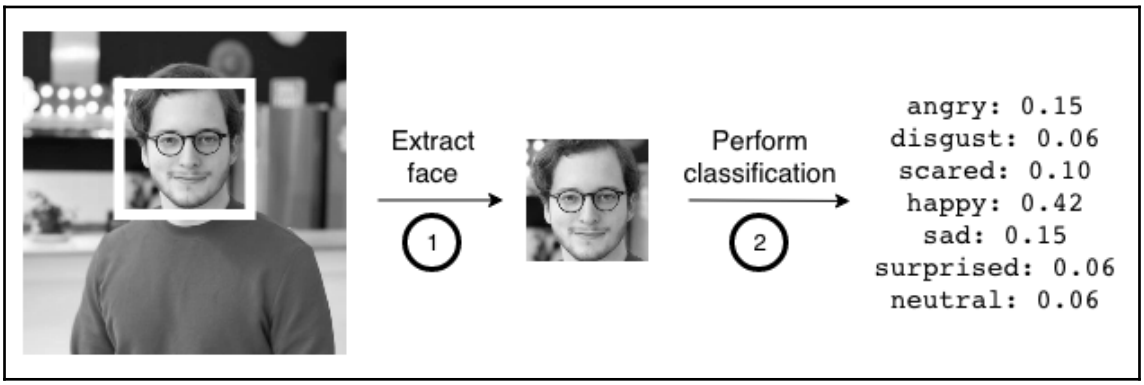
Title	dense/MatMul:MatMul#id=0#
User Friendly Category	other
Start	17,582,000 ns
Wall Duration	16,000 ns

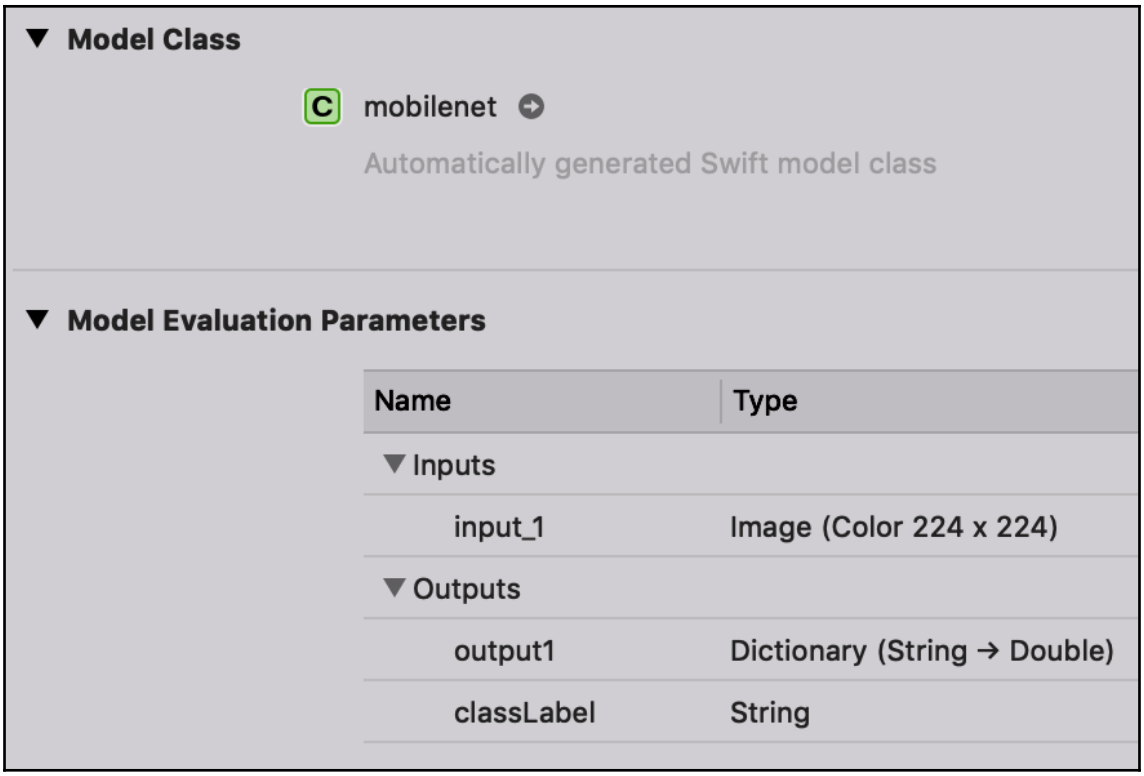


	output	label	output	label	output	label
Golden Retriever:	0.9	1	0.7	0	0.2	0
Husky:	0.7	0	0.9	1	0.1	0
Tabby cat:	0.1	0	0.1	0	0.9	1



223	135	186	Divide by 127	1.76	1.06	1.46	Subtract 1	0.76	0.06	0.46
101	93	110	→	0.80	0.73	0.87	→	-0.20	-0.27	-0.13
129	156	189		1.02	1.23	1.49		0.02	0.23	0.49


Extract face
1
Perform classification
2
angry: 0.15
disgust: 0.06
scared: 0.10
happy: 0.42
sad: 0.15
surprised: 0.06
neutral: 0.06


Model Class
mobilenet
Automatically generated Swift model class
Model Evaluation Parameters
Name
Type
Inputs
input_1
Image (Color 224 x 224)
Outputs
output1
Dictionary (String → Double)
classLabel
String