Image credit: https://mathematicaforprediction.files.wordpress.com/2013/08/digitimageswithzenbrush-testset.jpg


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1

Location shifted

Location shifted

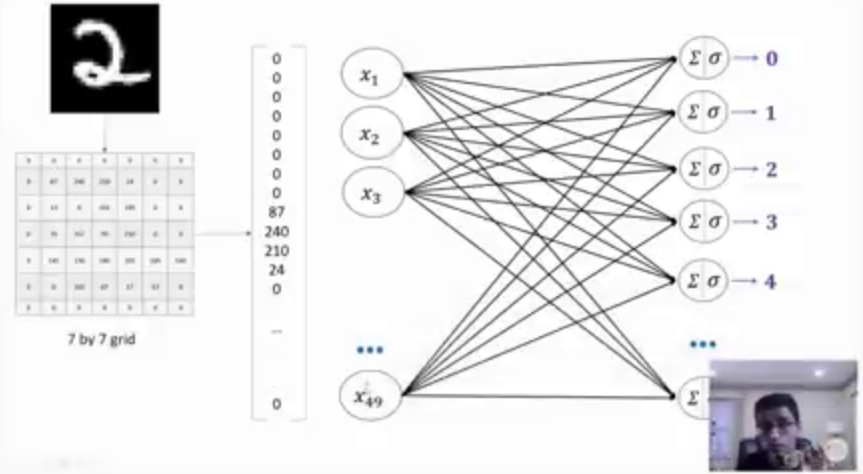
Variation 1


-1 -1 1 -1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1


-1 -1 1 -1 -1
-1 1 -1 1 -1
-1 1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1

To handle variety in digits we can use simple artificial neural network (ANN)



Neural Network For Handwritten Digits Classification | Deep Learning Tutorial 7 (Tensorflow2.0)

10,172 views • Premiered Jul 19, 2020

codebasics
1756 subscribers

In this video we will build our first neural network in tensorflow and python for handwritten digits classification. We will first build a very simple neural network with only input and output layer. After that we will add a hidden layer and check how the performance of our model changes.

SHOW MORE

ANALYTICS EDIT VIDEO

Deep Learning With Tensorflow 2.0, Keras and Python

codebasics

Neural Network | Deep Learnin...

Install tensorflow 2.0 | Deep Learning Tutorial 5...

Pytorch vs Tensorflow vs Keras | Deep Learning Tutorial 6...

Neural Network For Handwritten Digits Classification | Deep...

Activation Functions | Deep Learning Tutorial 8...

Derivatives | Deep Learning Tutorial 9 (Tensorflow2.0, Kera...

Matrix Basics | Deep Learning Tutorial 10 (Tensorflow2.0...

Data Management

LET'S TALK

Build a campaign in 5 steps

VISIT SITE

2020 Machine Learning Roadmap

Daniel Bourke

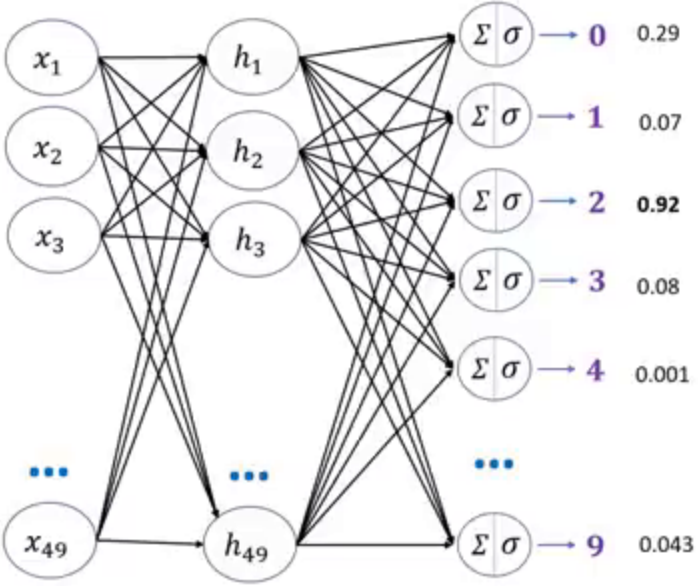
178K views

To handle variety in digits we can use simple artificial neural network (ANN)

0	0	0	0	0	0	0
0	87	240	210	24	0	0
0	13	0	101	195	0	0
0	35	167	99	210	0	0
0	145	230	240	201	189	140
0	0	102	67	17	13	0
0	0	0	0	0	0	0

7 by 7 grid

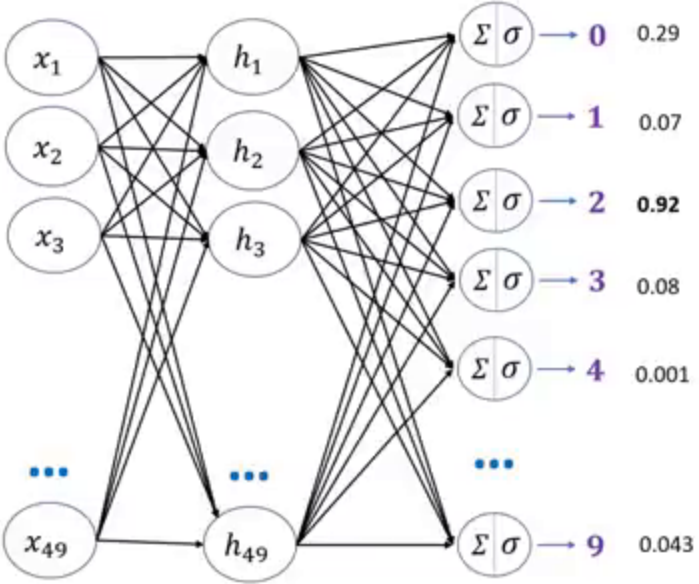
0
0
0
0
0
0
0
0
0
87
240
210
24
0
...
0



0	0	0	0	0	0	0
0	87	240	210	24	0	0
0	13	0	101	195	0	0
0	35	167	99	210	0	0
0	145	230	240	201	189	140
0	0	102	67	17	13	0
0	0	0	0	0	0	0

7 by 7 grid

0
0
0
0
0
0
0
0
0
87
240
210
24
0
...
0



49*49 = 2401

49*10 = 490

Image size = 1920 x 1080 X 3

Image size = 1920 x 1080 X 3

First layer neurons = 1920 x 1080 X 3 ~ 6 million

Image size = 1920 x 1080 X 3

First layer neurons = 1920 x 1080 X 3 ~ 6 million

Hidden layer neurons = Let's say you keep it ~ 4 million

Weights between input and hidden layer = 6 mil * 4 mil
= 24 million

Image size = 1920 x 1080 X 3

First layer neurons = 1920 x 1080 X 3 ~ 6 million

Hidden layer neurons = Let's say you keep it ~ 4 million

Weights between input and hidden layer = 6 mil * 4 mil
= 24 million

Image credit: https://giphy.com/ (@rabbids)

Disadvantages of using ANN for image classification

1. Too much computation
2. Treats local pixels same as pixels far apart
3. Sensitive to location of an object in an image

Disadvantages of using ANN for image classification

1. Too much computation
2. Treats local pixels same as pixels far apart
3. Sensitive to location of an object in an image

Disadvantages of using ANN for image classification

1. Too much computation
2. Treats local pixels same as pixels far apart
3. Sensitive to location of an object in an image

Disadvantages of using ANN for image classification

1. Too much computation
2. Treats local pixels same as pixels far apart
3. Sensitive to location of an object in an image


HOW DOES HUMANS
RECOGNIZE IMAGES SO EASILY?


HOW DOES HUMANS
RECOGNIZE IMAGES SO EASILY?
imgflip.com

Koala's eye? = Y

Koala's nose? = Y

Koala's ears? = Y

Koala's eye? = Y

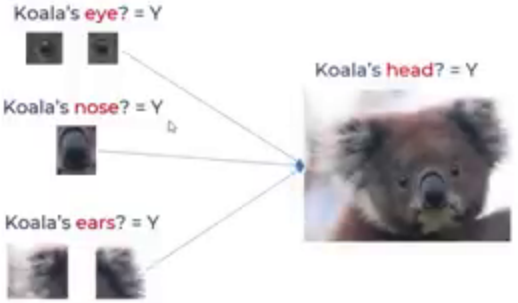
Koala's nose? = Y

Koala's ears? = Y

Koala's eye? = Y


Koala's nose? = Y

Koala's ears? = Y



Koala's eye? = Y
Koala's nose? = Y
Koala's ears? = Y
Koala's head? = Y

Koala's eye? = Y

Koala's nose? = Y

Koala's ears? = Y

Koala's head? = Y

Koala's hands? = Y

Koala's legs? = Y

Koala's eye? = Y

Koala's nose? = Y

Koala's ears? = Y

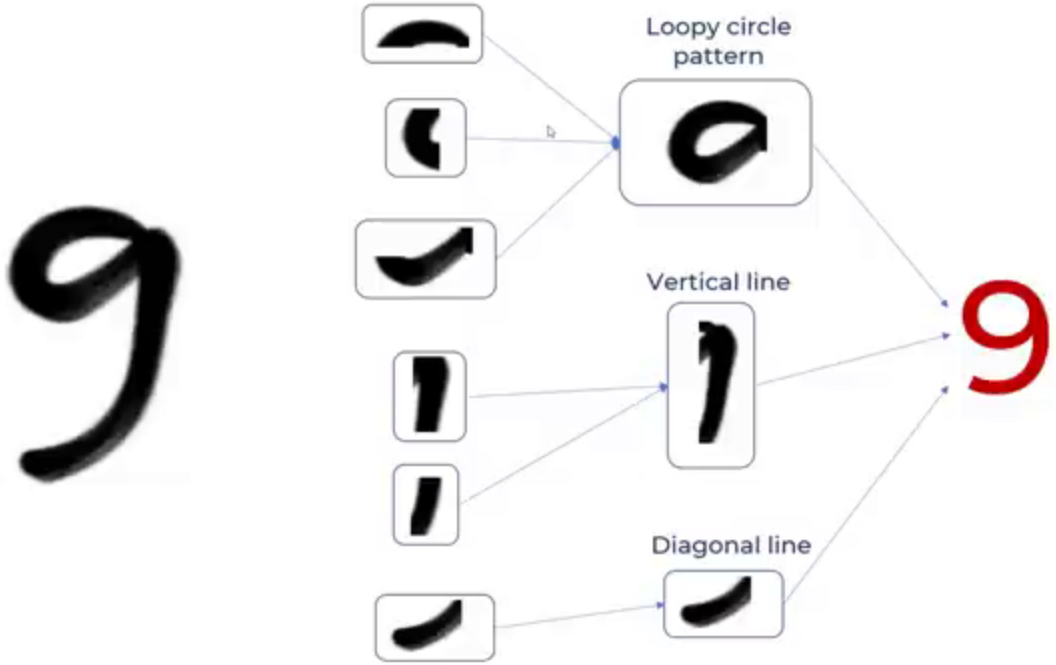
Koala's head? = Y

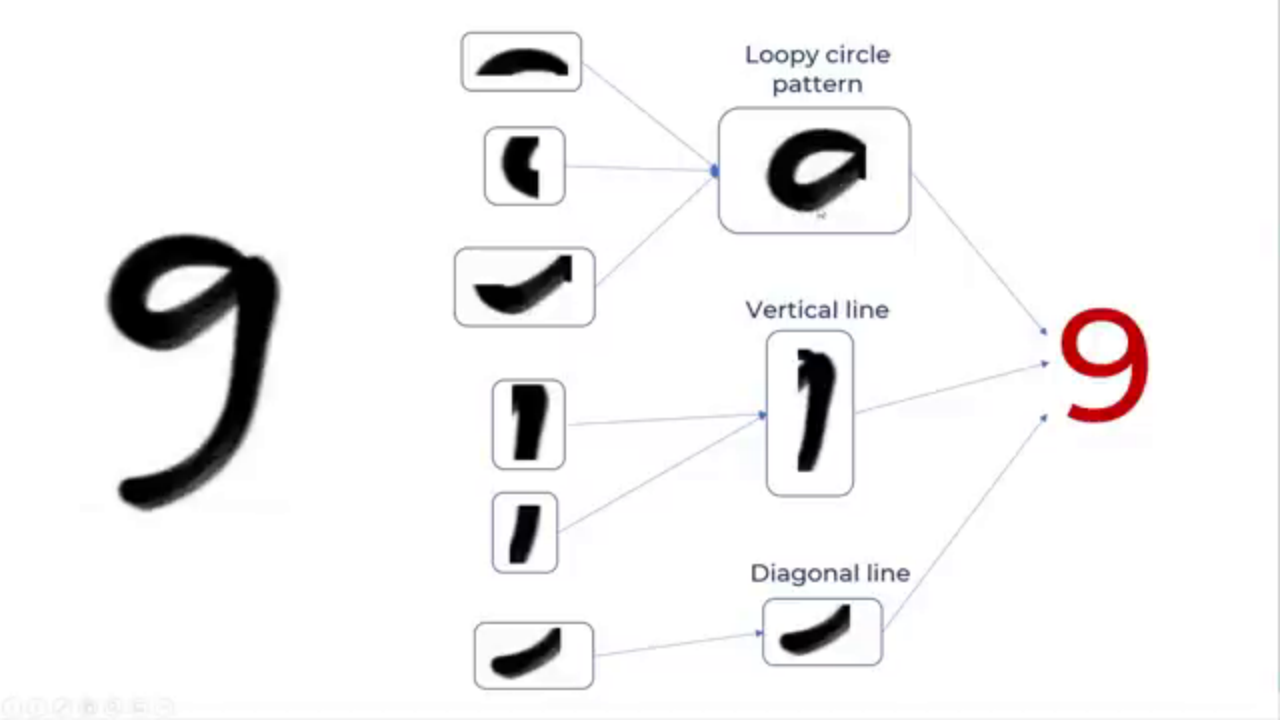
Is it Koala? = Y

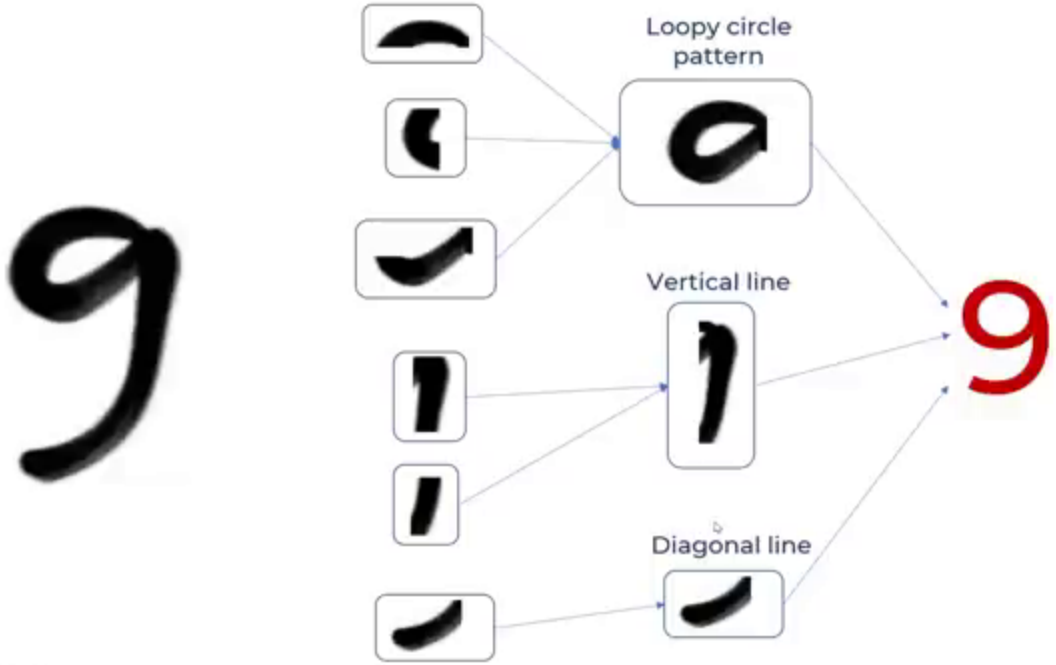
Koala's hands? = Y

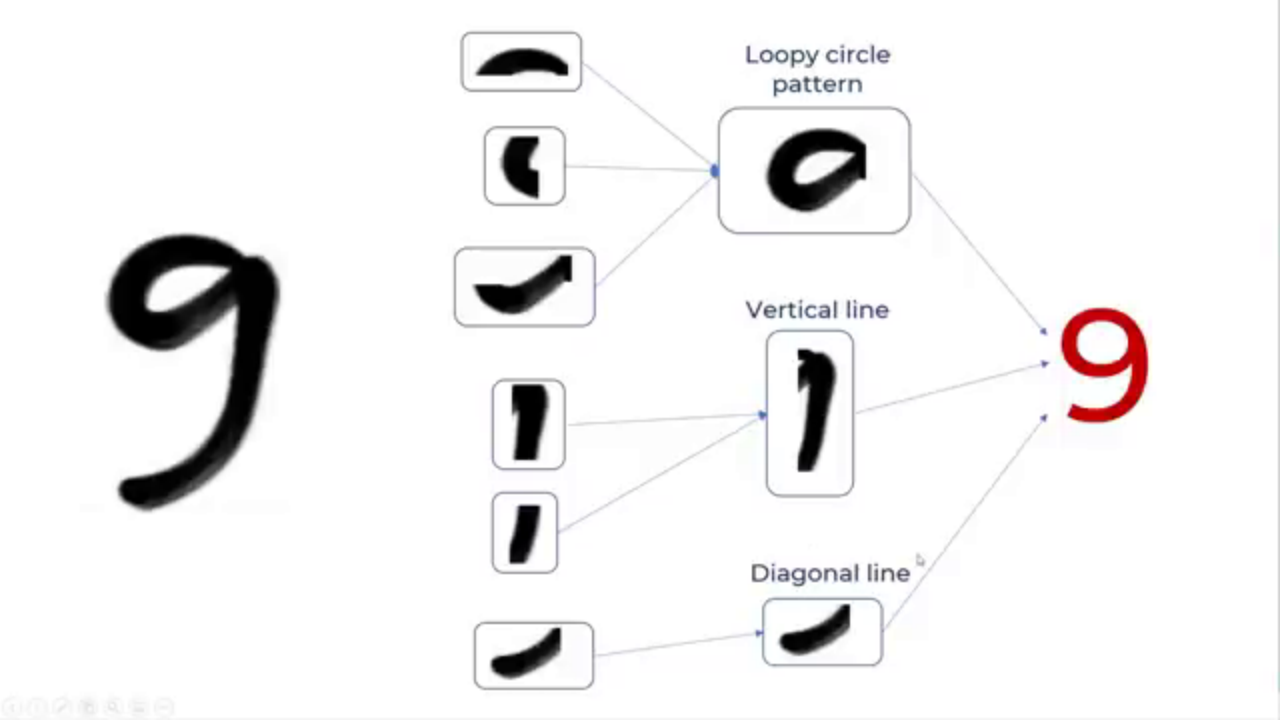
Koala's legs? = Y

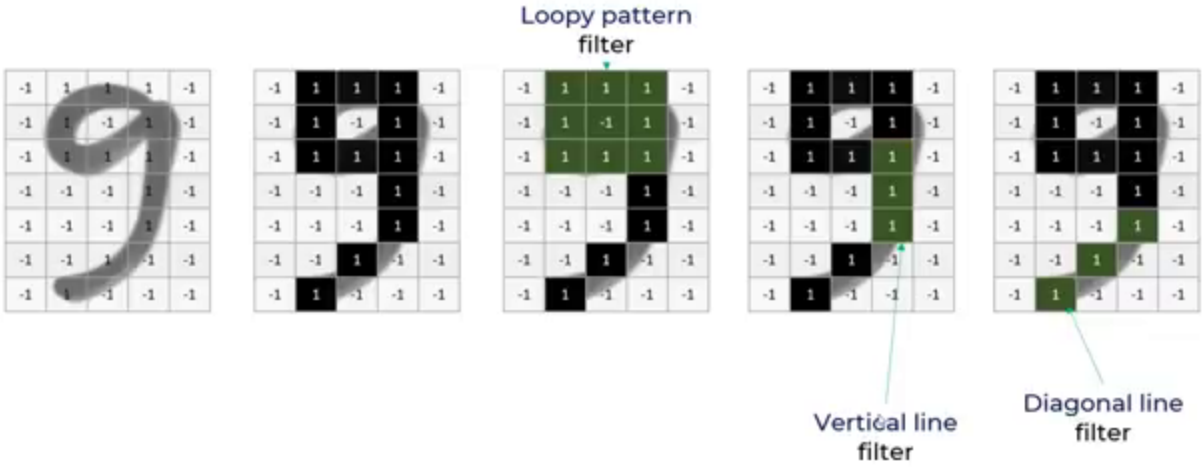
Koala's body? = Y

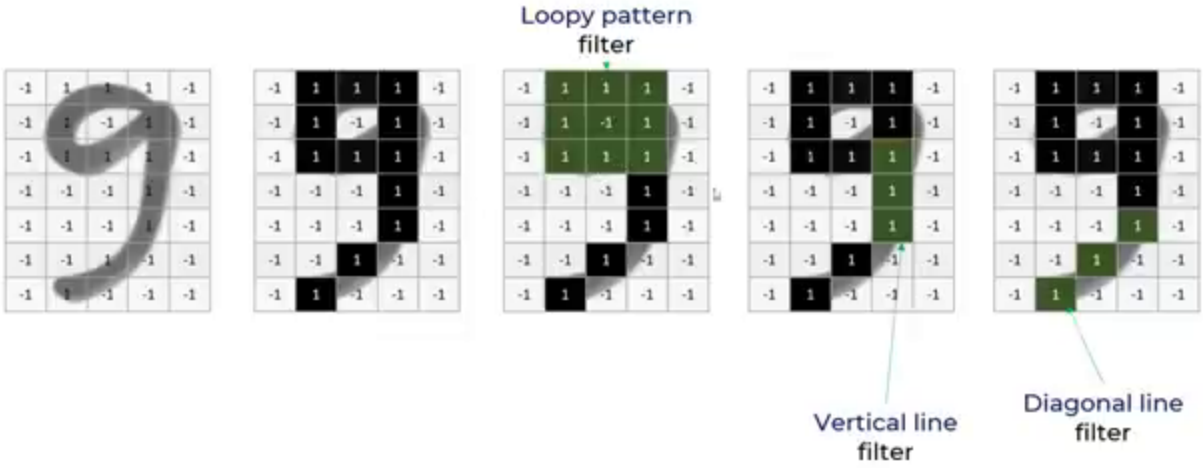

Loopy circle pattern
Vertical line
Diagonal line
9


Loopy circle pattern
Vertical line
Diagonal line
9


Loopy circle pattern
Vertical line
Diagonal line
9


Loopy circle pattern
Vertical line
Diagonal line
9


Loopy pattern filter
Vertical line filter
Diagonal line filter


Loopy pattern filter
Vertical line filter
Diagonal line filter

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

1	1	1
1	-1	1
1	1	1

-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

-1	1	1
1	-1	1
1	1	1

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

1	1	1
1	-1	1
1	1	1

-1+1+1-1-1-1-1+1+1 = -1 → -1/9 = -0.11

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

*

1	1	1
1	-1	1
1	1	1

-0.11		

-1+1+1-1-1-1-1+1+1 = -1 → -1/9 = -0.11

-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

*

1	1	1
1	-1	1
1	1	1

-0.11		

-1+1+1-1-1-1-1+1+1 = -1 → -1/9 = -0.11

-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

*

1	1	1
1	-1	1
1	1	1

-0.11		

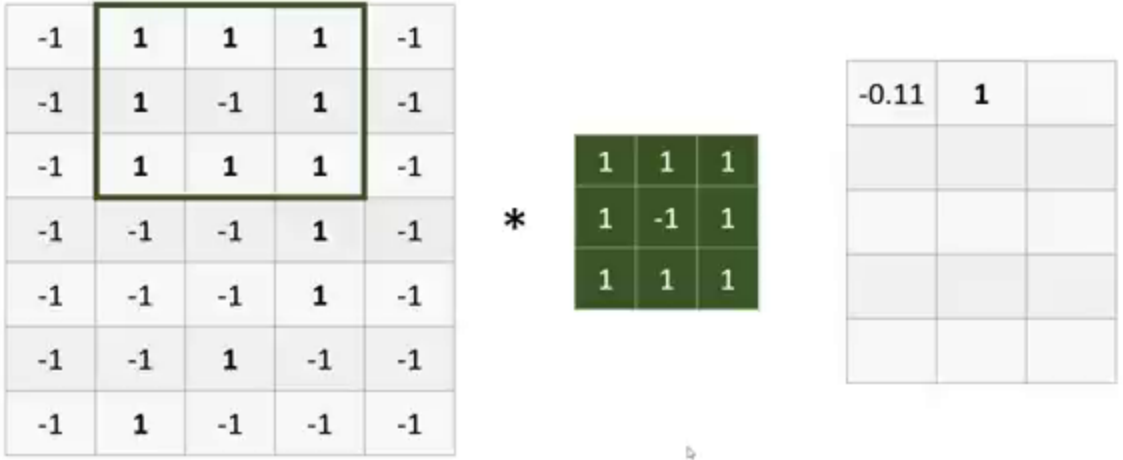
-1+1+1-1-1-1-1+1+1 = -1 → -1/9 = -0.11

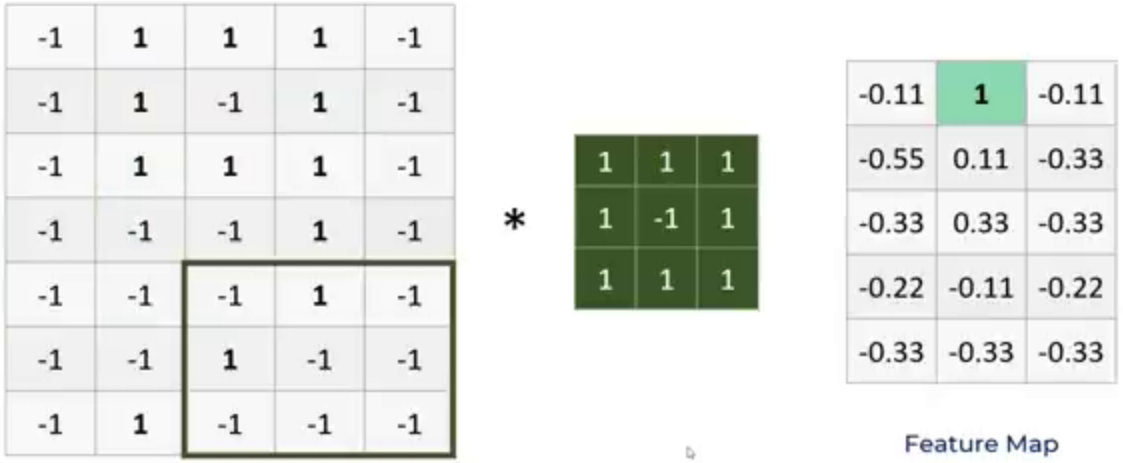
-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

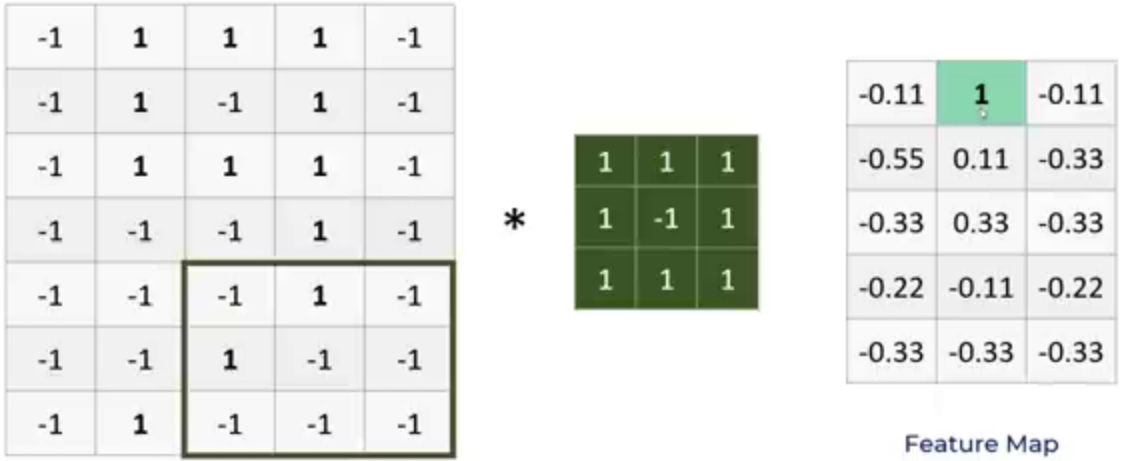
*

1	1	1
1	-1	1
1	1	1

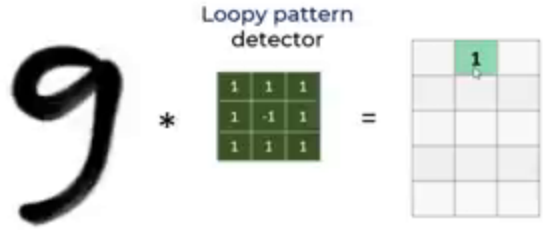
-0.11		

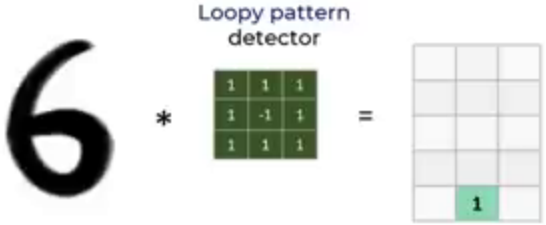

-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
1 1 1
1 -1 1
1 1 1
-0.11 1

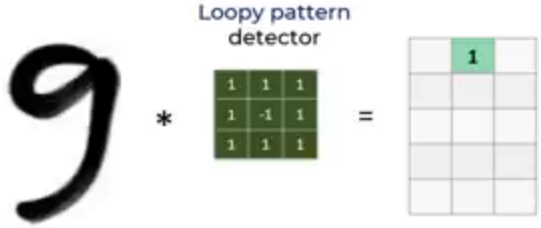

-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
1 1 1
1 -1 1
1 1 1
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
Feature Map

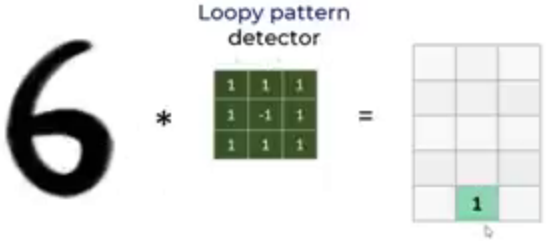

-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
1 1 1
1 -1 1
1 1 1
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
Feature Map

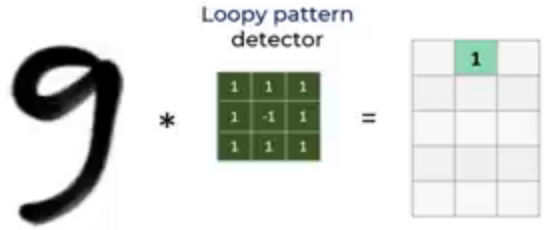

-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
1 1 1
1 -1 1
1 1 1
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
Feature Map

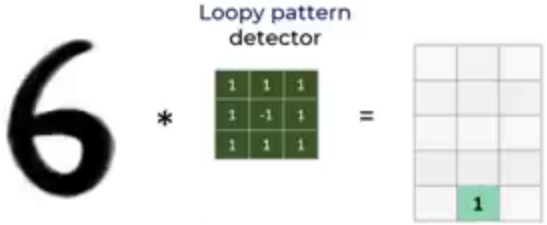

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1

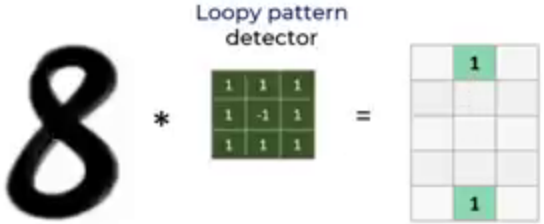

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1

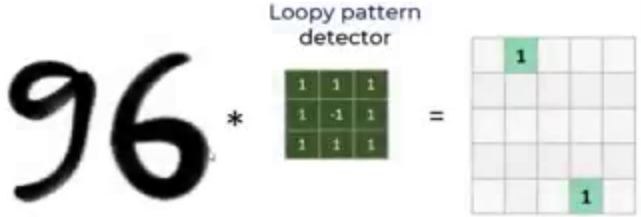

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1

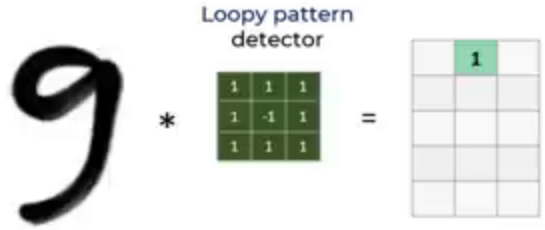

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1

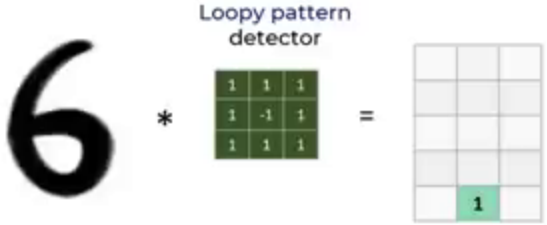

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1

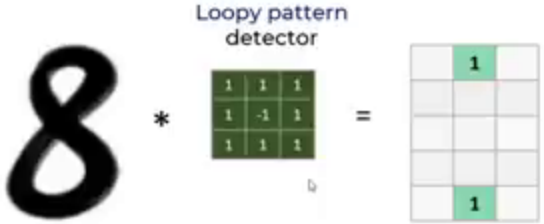

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1

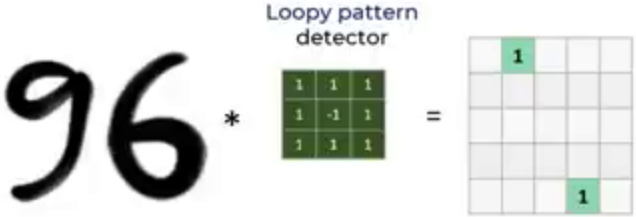

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
1


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
1


Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
1


Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
1


Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
1
1


Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
1
1

eye
detector

*

eye
detector

=

eye
detector

=

Location invariant: It can detect eyes in any location of the image

*

eye
detector

=

eye
detector

=

Location invariant: It can detect eyes in any location of the image

*

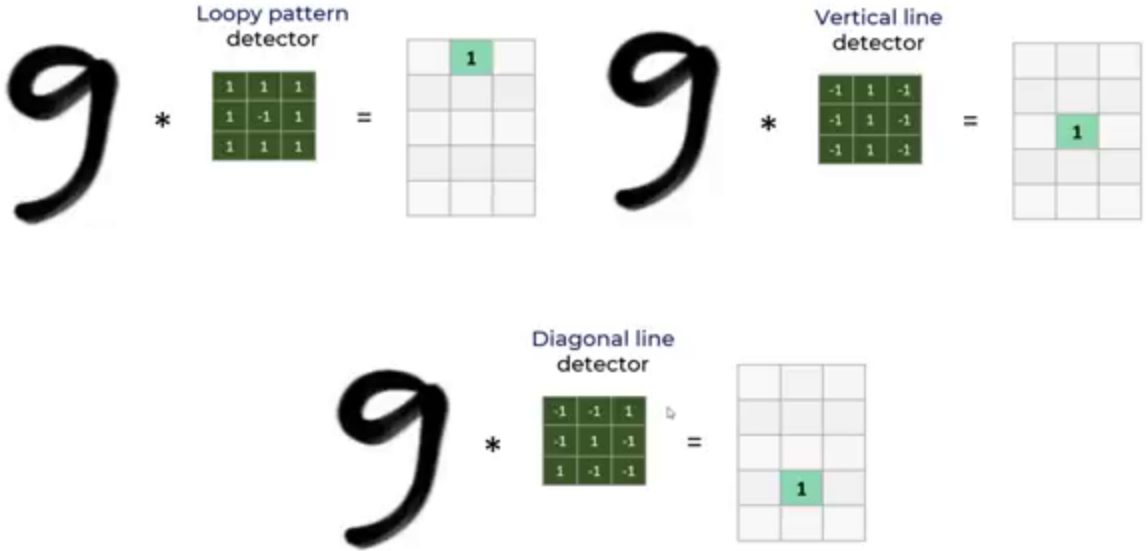
eye
detector

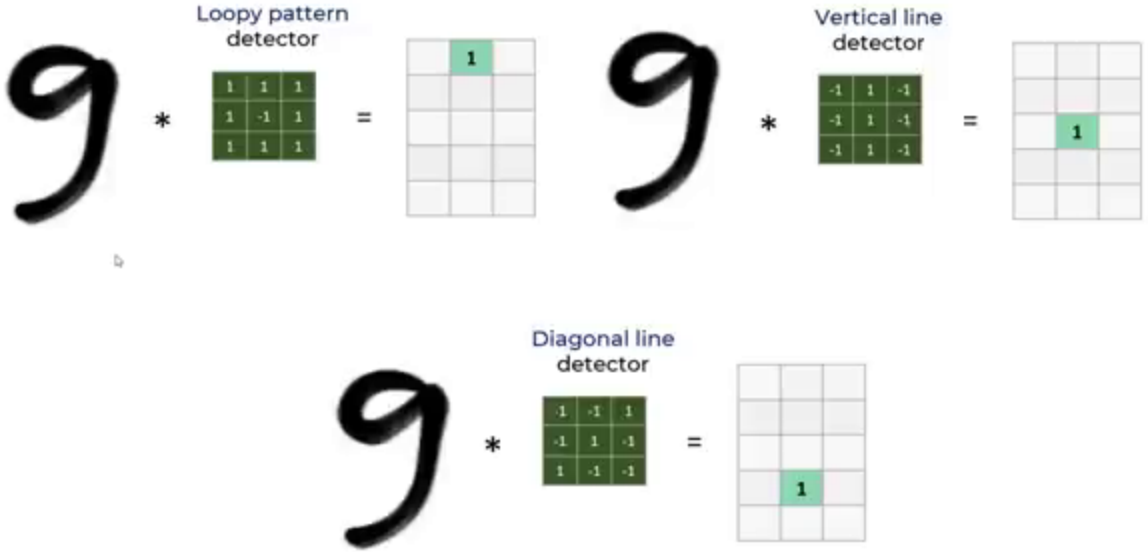
=

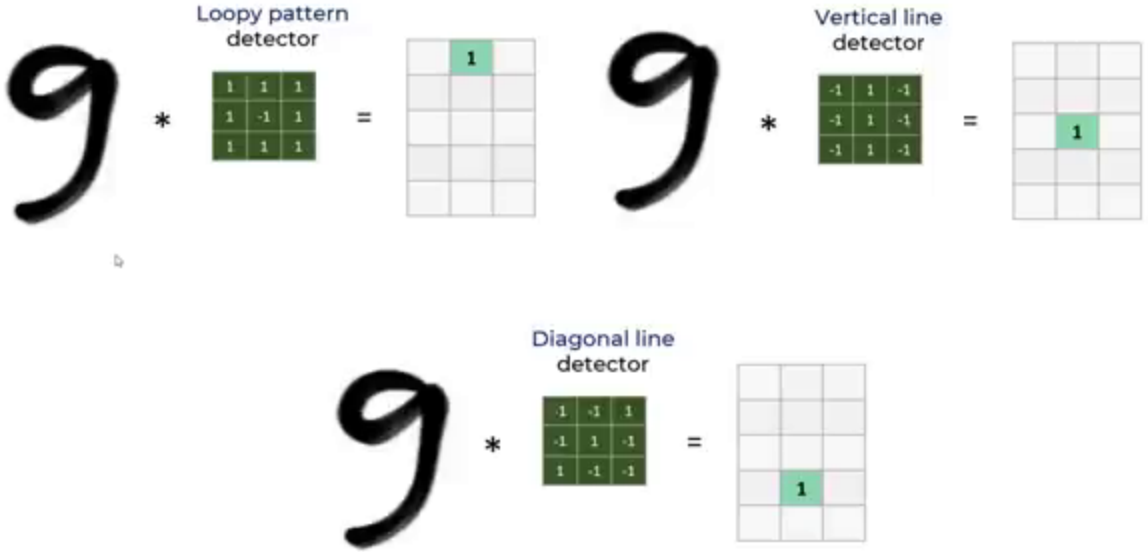
eye
detector

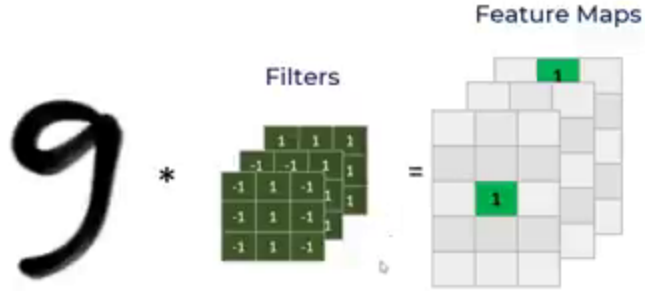
=

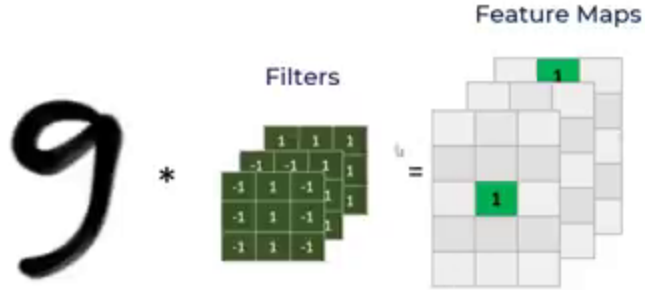
hands
detector

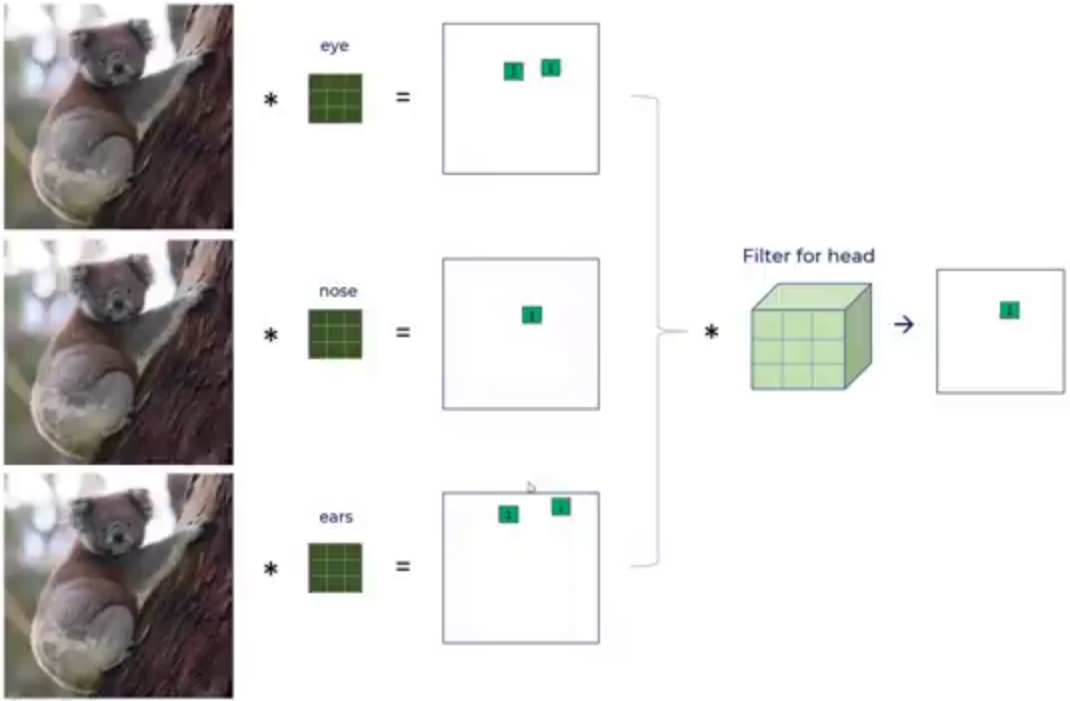

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
Vertical line detector
-1 1 -1
-1 1 -1
-1 1 -1
1
Diagonal line detector
-1 -1 1
-1 1 -1
1 -1 -1
1

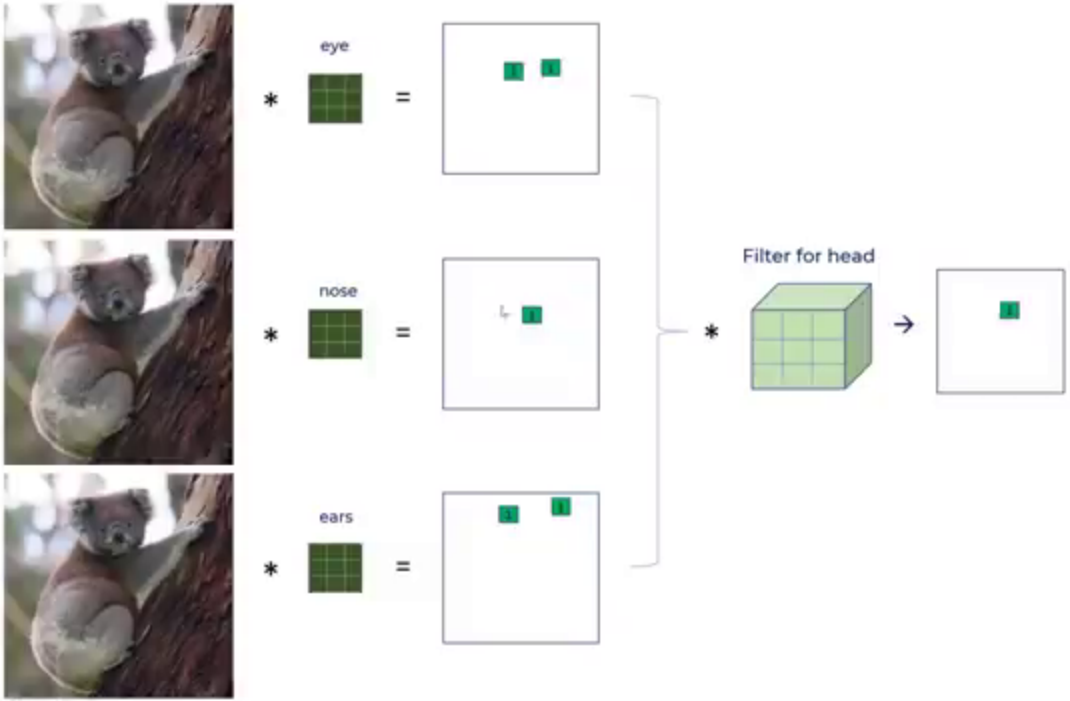

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
Vertical line detector
-1 1 -1
-1 1 -1
-1 1 -1
1
Diagonal line detector
-1 -1 1
-1 1 -1
1 -1 -1
1

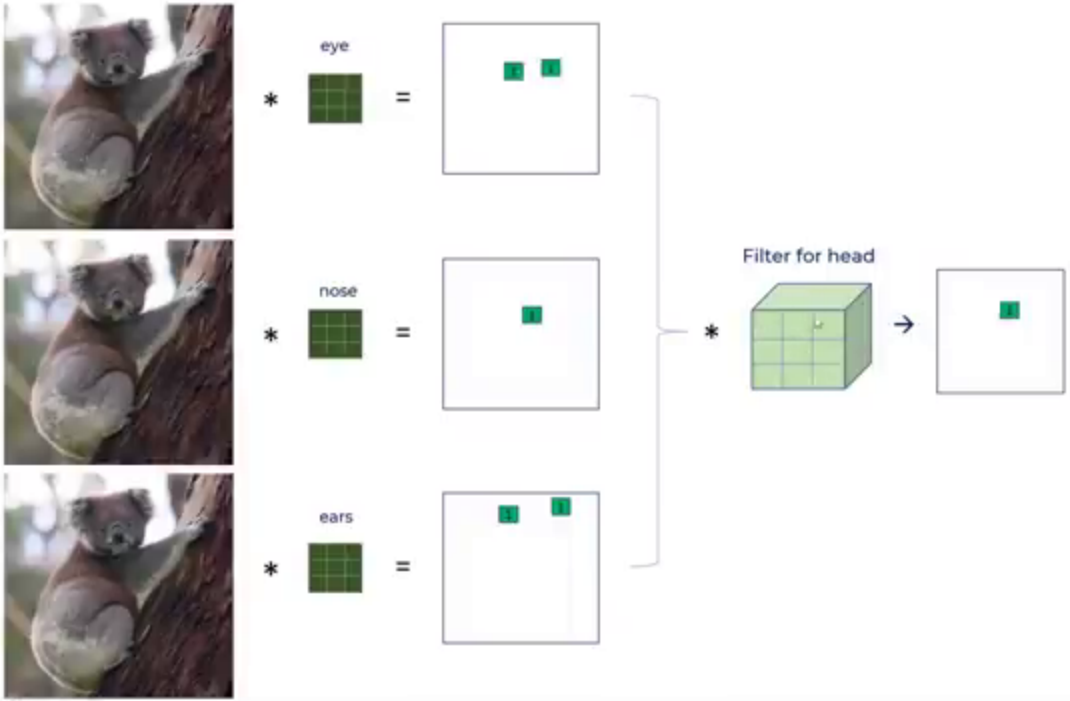

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
Vertical line detector
-1 1 -1
-1 1 -1
-1 1 -1
1
Diagonal line detector
-1 -1 1
-1 1 -1
1 -1 -1
1

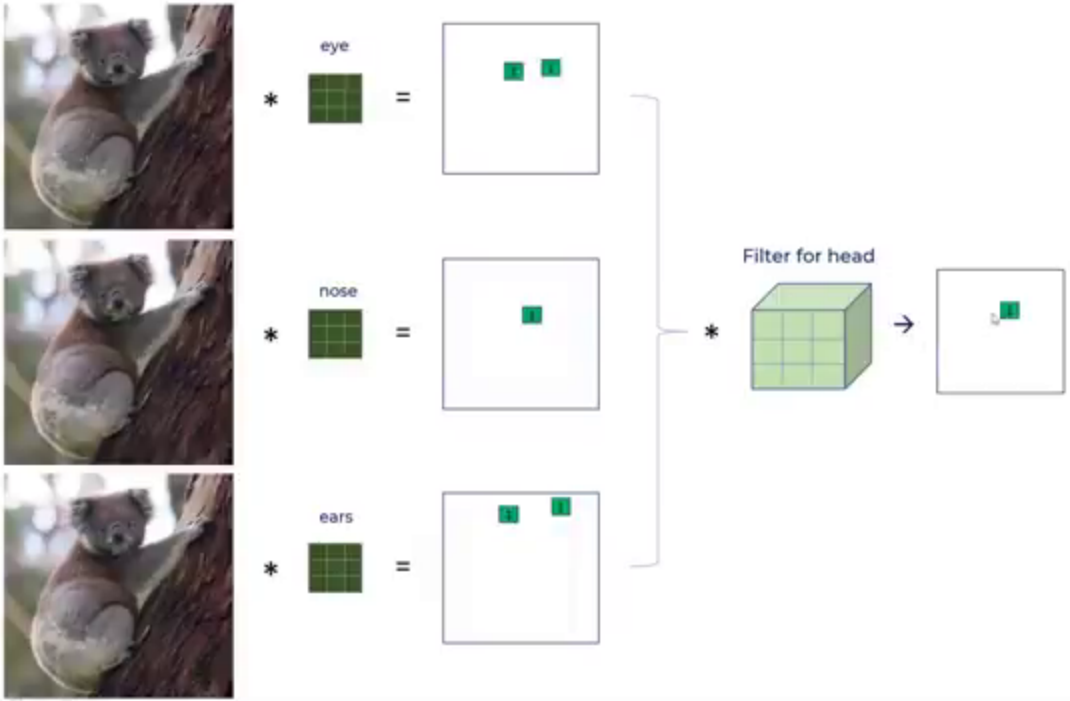

Feature Maps
Filters
1 1 1
-1 -1 1
-1 1 -1
-1 1 -1
-1 1 -1
1
1

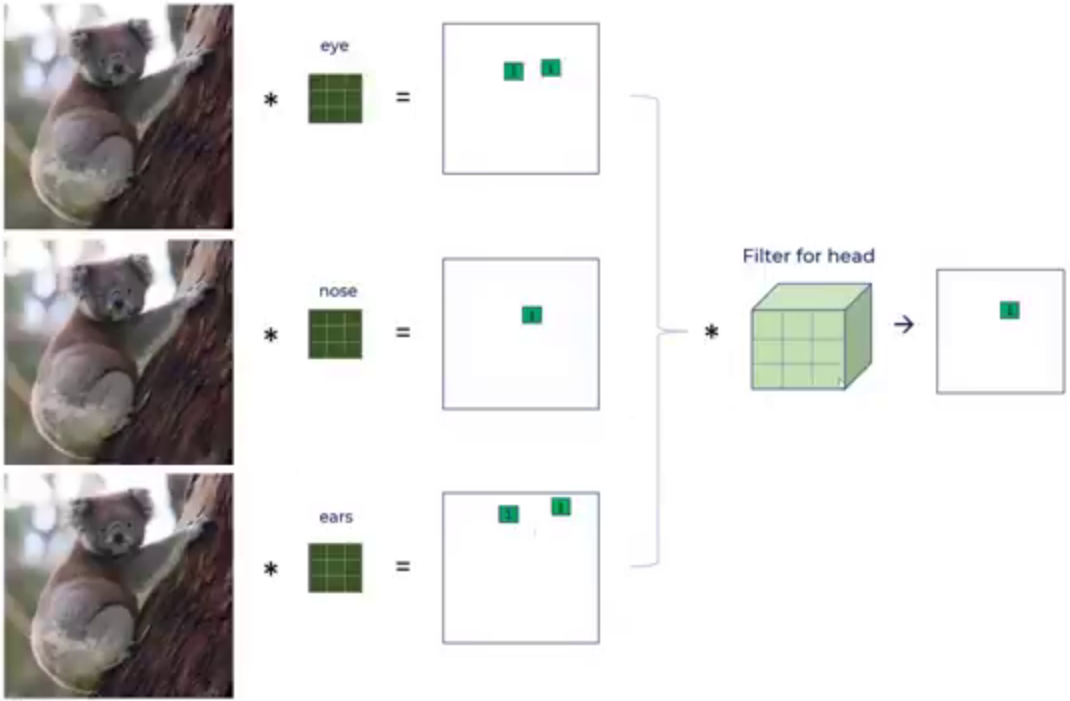

Feature Maps
Filters
1 1 1
-1 -1 1
-1 1 -1
-1 1 -1
-1 1 -1
1
1

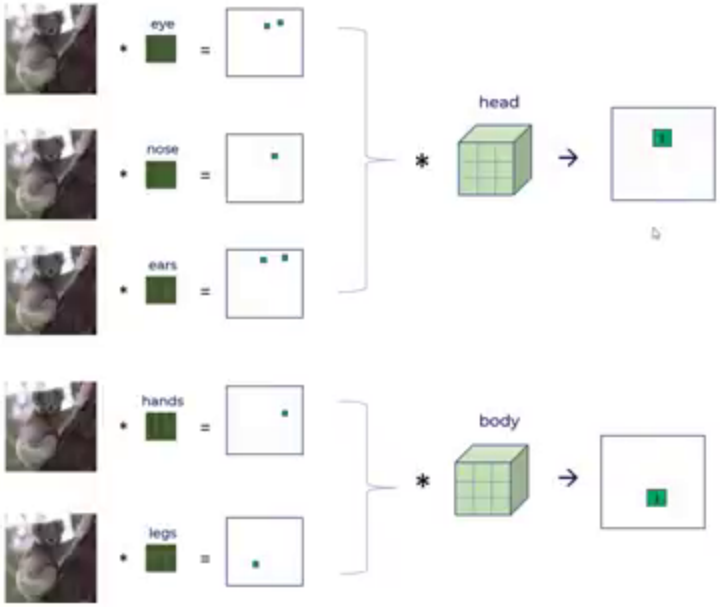

eye
nose
ears
Filter for head

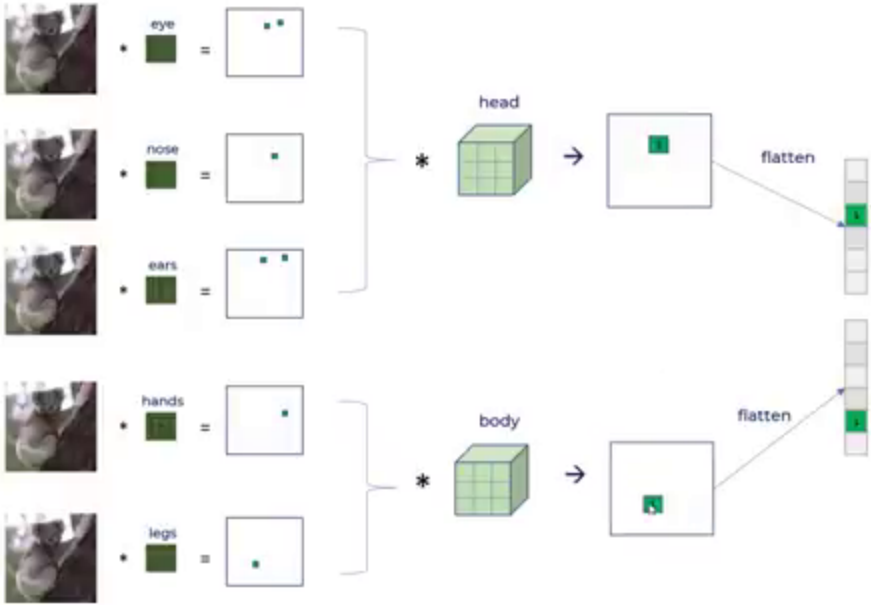

eye
nose
ears
Filter for head

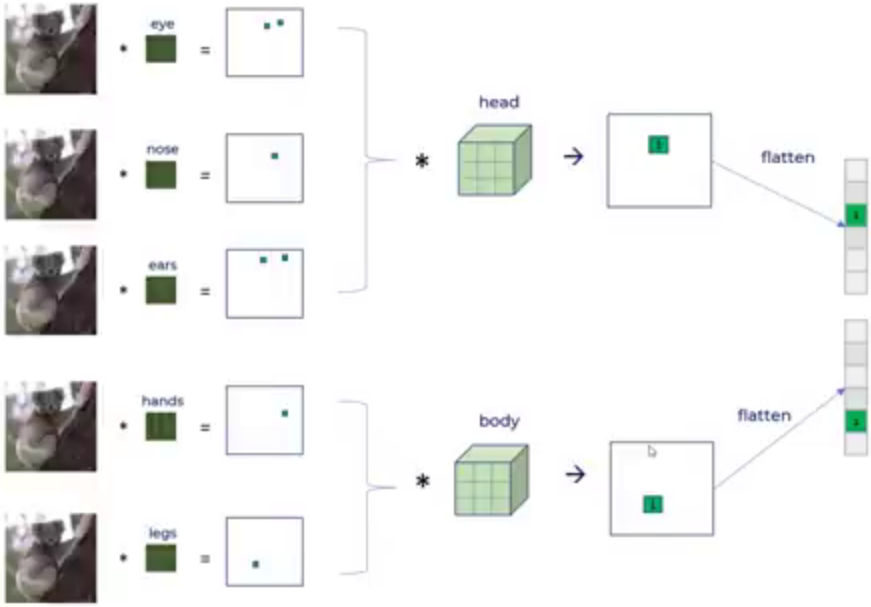

eye
nose
ears
Filter for head

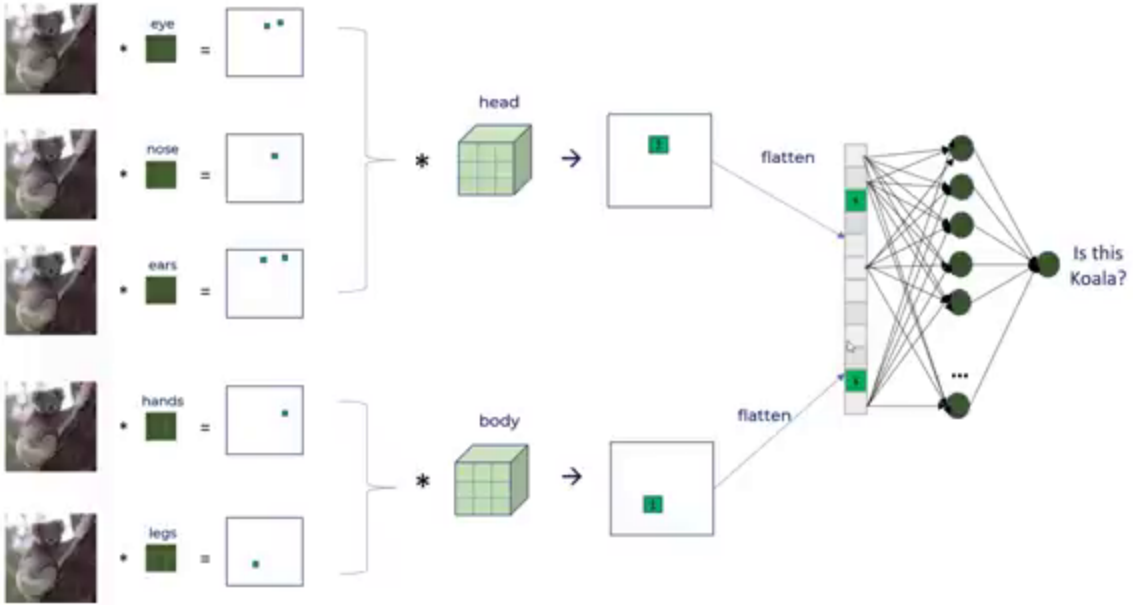

eye
nose
ears
Filter for head

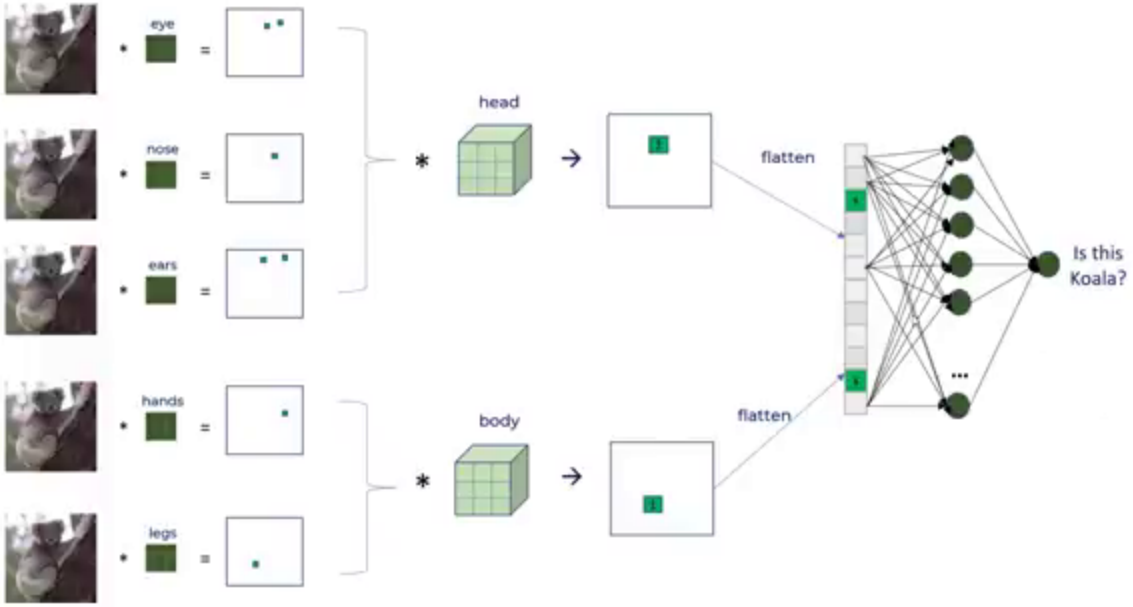

eye
nose
ears
Filter for head


eye
nose
ears
head
hands
legs
body


eye
nose
ears
hands
legs
head
body
flatten
flatten


eye
nose
ears
hands
legs
head
body
flatten
flatten


eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this Koala?


eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this Koala?

eye

nose

ears

hands

legs

head

body

flatten

flatten

Is this Koala?

eye

nose

ears

hands

legs

head

body

flatten

flatten

Is this Koala?

eye

nose

ears

hands

legs

head

body

flatten

flatten

Is this Koala?

eye

nose

ears

hands

legs

head

body

flatten

flatten

Is this Koala?

eye

nose

ears

hands

legs

head

body

flatten

flatten

Is this Koala?

Feature Extraction

eye

nose

ears

hands

legs

head

body

flatten

flatten

Is this Koala?

Feature Extraction

Classification

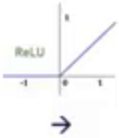
*

Loopy pattern filter

1	1	1
1	-1	1
1	1	1

→

-0.11	**1**	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33



→

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

*

Loopy pattern filter

1	1	1
1	-1	1
1	1	1

→

-0.11	**1**	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

→

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

*

Loopy pattern filter

→

-0.11	1	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

→

0	1	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

*

Loopy pattern filter

→

-0.11	**1**	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

→

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

ReLU helps with making the model nonlinear

*

Loopy pattern filter

→

-0.11	1	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

→

0	1	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

*

Loopy pattern filter

→

-0.11	**1**	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

→

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

We didn't address the issue of too much computation

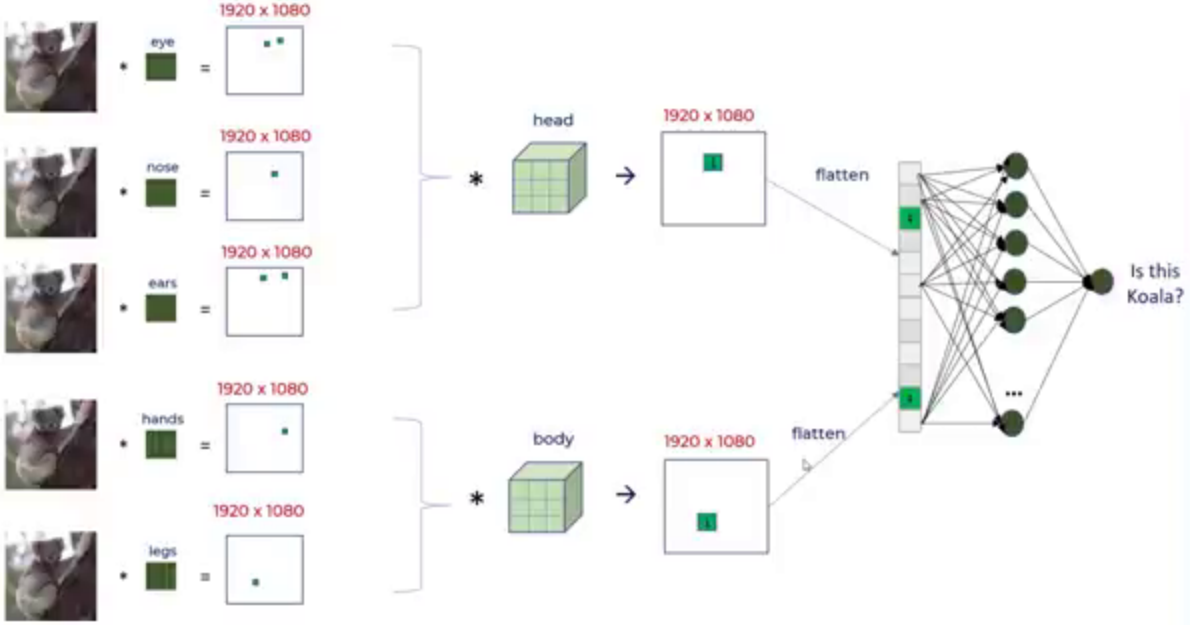
Image credit: https://giphy.com/ (@rabbids)

Image size = 1920 x 1080 X 3

First layer neurons = 1920 x 1080 X 3 ~ 6 million

Hidden layer neurons = Let's say you keep it ~ 4 million

Weights between input and hidden layer = 6 mil * 4 mil
= 24 million

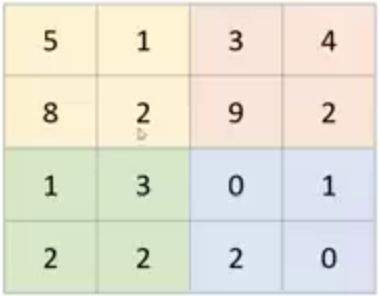

1920 x 1080
eye
1920 x 1080
nose
1920 x 1080
ears
1920 x 1080
hands
1920 x 1080
legs
head
1920 x 1080
body
1920 x 1080
flatten
flatten
Is this Koala?

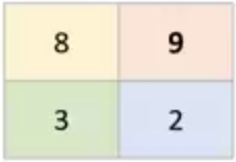
Pooling layer is used to reduce the size

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

8	**9**
3	2

2 by 2 filter with stride = 2



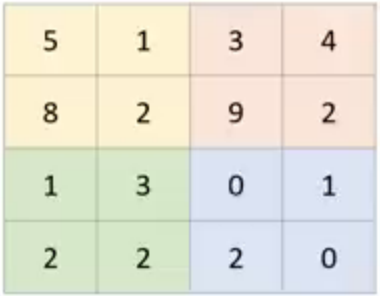


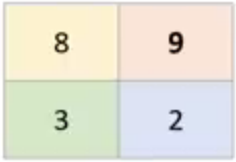
2 by 2 filter with stride = 2

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

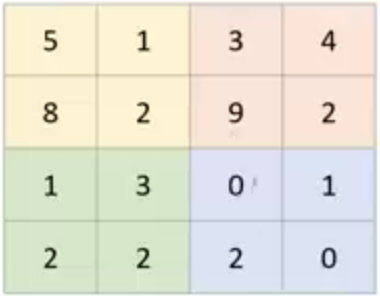
8	9
3	2

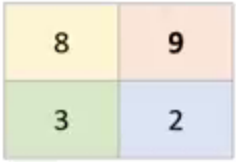
2 by 2 filter with stride = 2



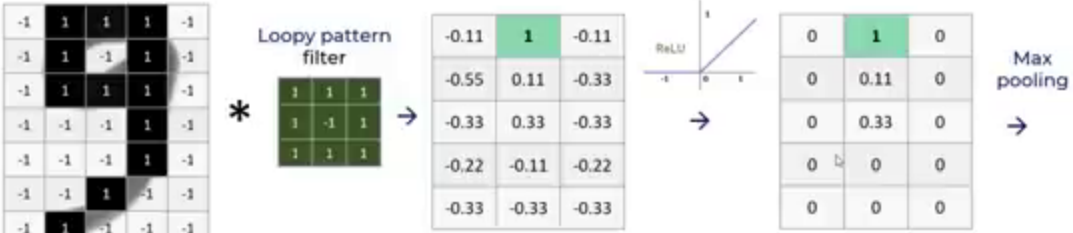


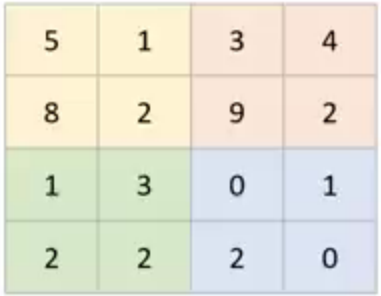
2 by 2 filter with stride = 2

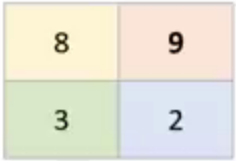




2 by 2 filter with stride = 2


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
Loopy pattern filter
1 1 1
1 -1 1
1 1 1
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
ReLU
0 1 0
0 0.11 0
0 0.33 0
0 0 0
0 0 0
Max pooling



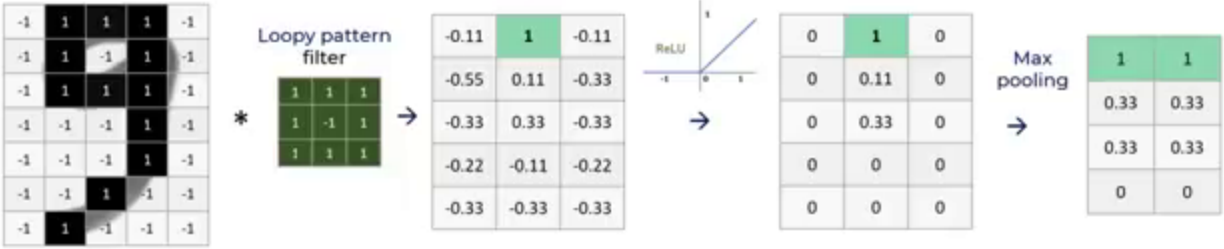


2 by 2 filter with stride = 2

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

1	1

2 by 2 filter with stride = 1


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
Loopy pattern filter
1 1 1
1 -1 1
1 1 1
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
ReLU
0 1 0
0 0.11 0
0 0.33 0
0 0 0
0 0 0
Max pooling
1 1
0.33 0.33
0.33 0.33
0 0

Shifted 9 at different position

*

Loopy pattern filter

1	1	1
1	-1	1
1	1	1

→

1	-0.11	-0.11
0.11	-0.33	0.33
0.33	-0.33	-0.33
-0.11	-0.55	-0.33
-0.55	-0.33	-0.55

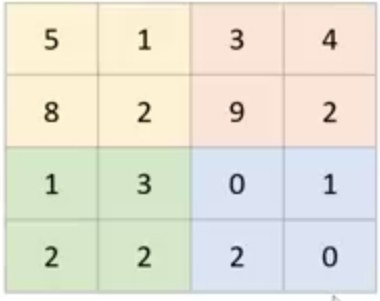
→

1	0	0
0.11	0	0.33
0.33	0	0
0	0	0
0	0	0

Max pooling →

1	0.33
0.33	0.33
0.33	0
0	0

There is average pooling also...



4	4.5
2	0.75

Shifted 9 at different position

*

Loopy pattern filter

1	1	1
1	-1	1
1	1	1

→

1	-0.11	-0.11
0.11	-0.33	0.33
0.33	-0.33	-0.33
-0.11	-0.55	-0.33
-0.55	-0.33	-0.55

→

1	0	0
0.11	0	0.33
0.33	0	0
0	0	0
0	0	0

Max pooling →

1	0.33
0.33	0.33
0.33	0
0	0

There is average pooling also...

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

4	4.5
2	0.75

There is average pooling also...

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

4	4.5
2	0.75

Benefits of pooling

Reduces dimensions & computation

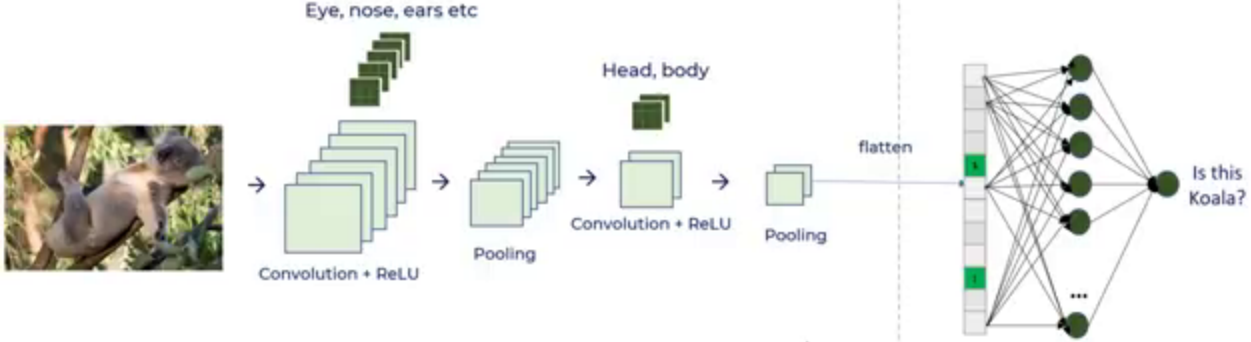
Reduce overfitting as there are less parameters

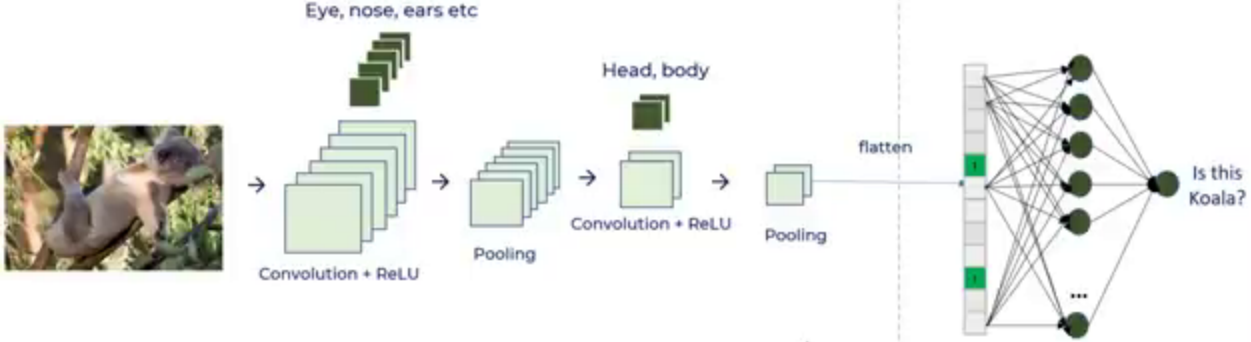
Benefits of pooling

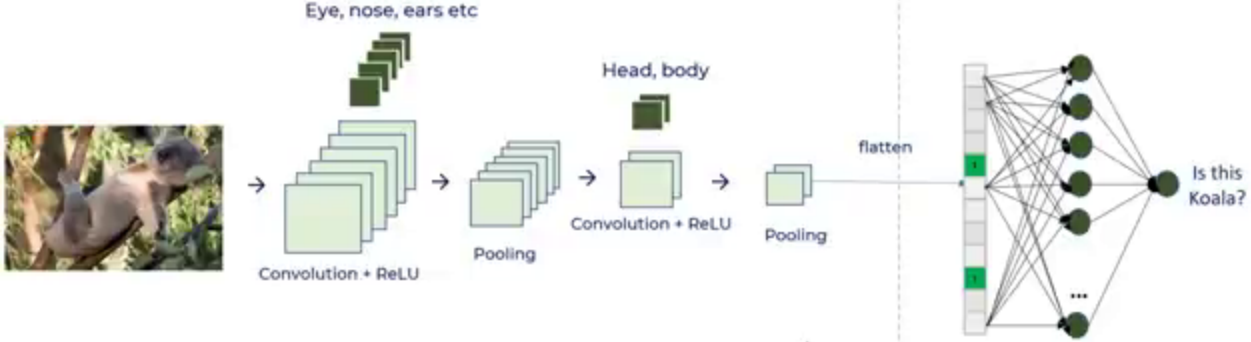
Reduces dimensions & computation

Reduce overfitting as there are less parameters

Model is tolerant towards variations, distortions


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

Eye, nose, ears etc

Convolution + ReLU

Pooling

Head, body

Convolution + ReLU

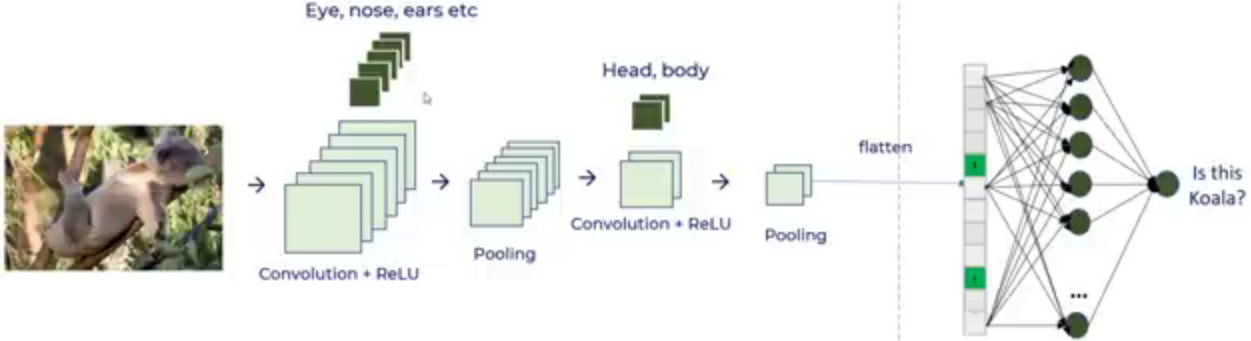
Pooling

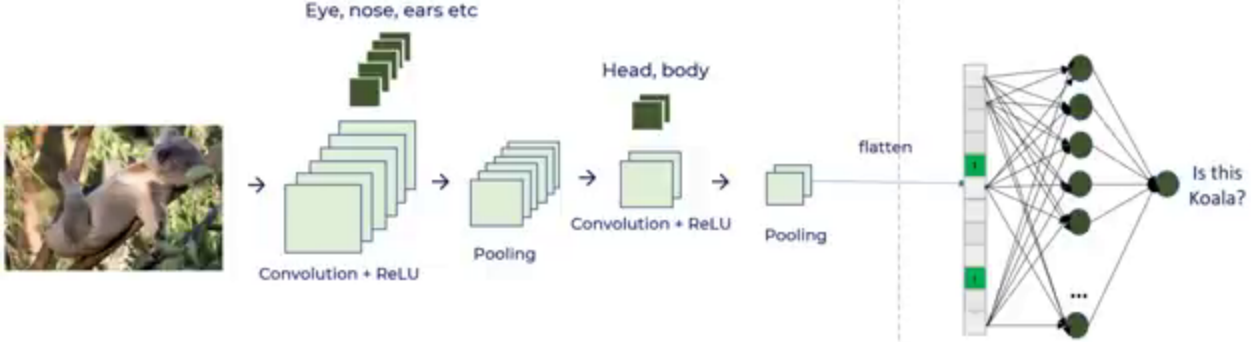
flatten

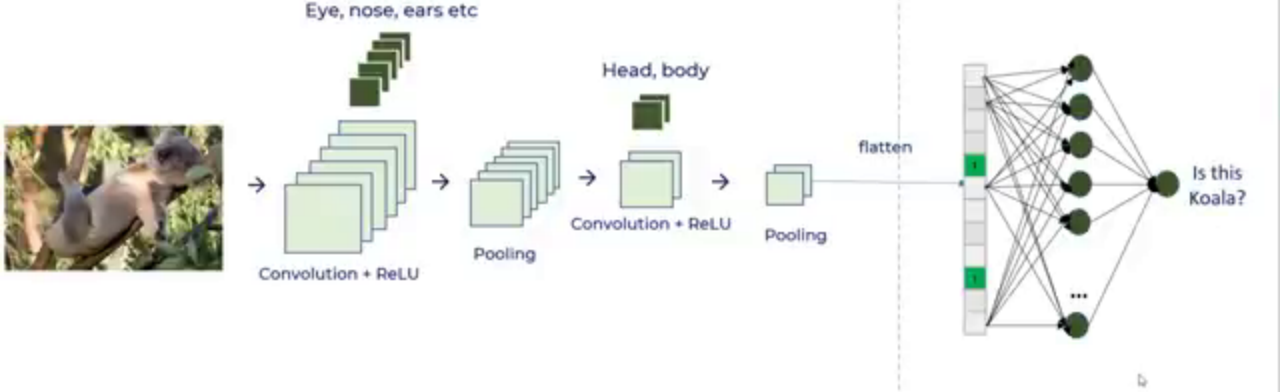
Is this Koala?

Feature Extraction

Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

ReLU

- Introduces nonlinearity
- Speeds up training, faster to compute

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

ReLU

- Introduces nonlinearity
- Speeds up training, faster to compute

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

ReLU

- Introduces nonlinearity
- Speeds up training, faster to compute

Pooling

- Reduces dimensions and computation
- Reduces overfitting
- Makes the model tolerant towards small distortion and variations

Rotation

Thickness

CNN by itself doesn't take care of rotation and scale

- You need to have rotated, scaled samples in training dataset
- If you don't have such samples than use data augmentation methods to generate new rotated/scaled samples from existing training samples

CNN by itself doesn't take care of rotation and scale

- You need to have rotated, scaled samples in training dataset
- If you don't have such samples than use data augmentation methods to generate new rotated/scaled samples from existing training samples

CNN by itself doesn't take care of rotation and scale

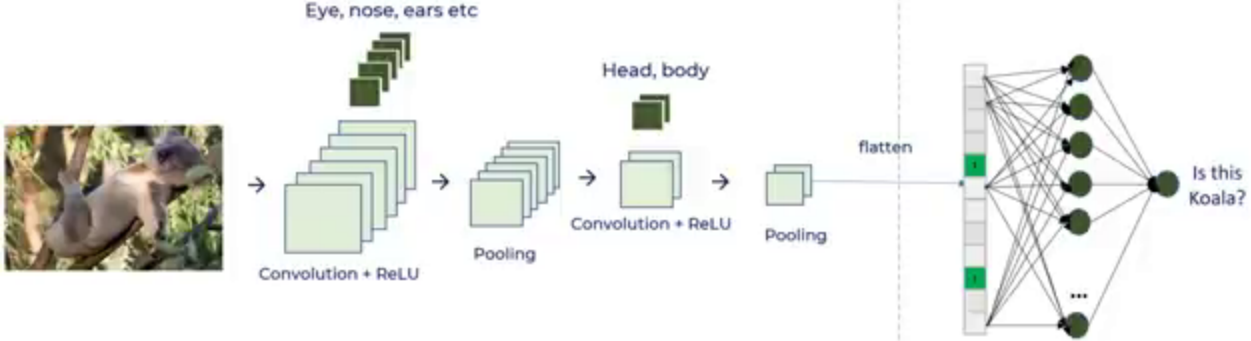
- You need to have rotated, scaled samples in training dataset
- If you don't have such samples than use data augmentation methods to generate new rotated/scaled samples from existing training samples

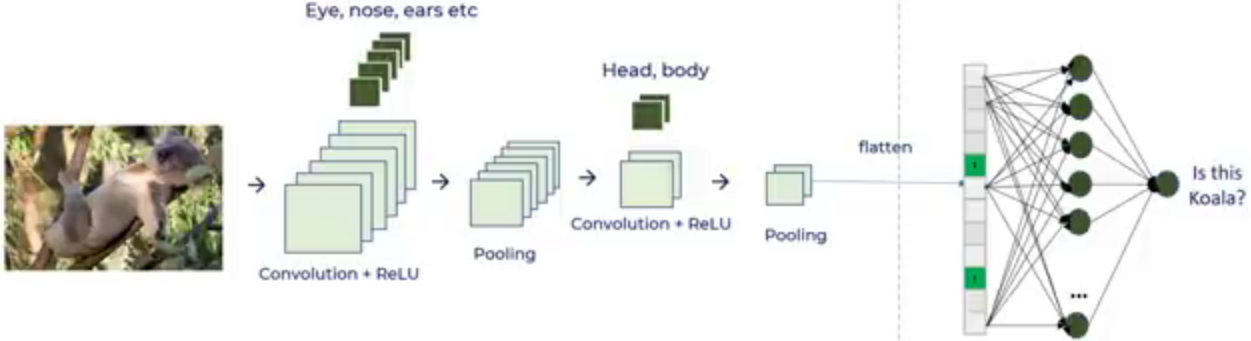
CNN by itself doesn't take care of rotation and scale

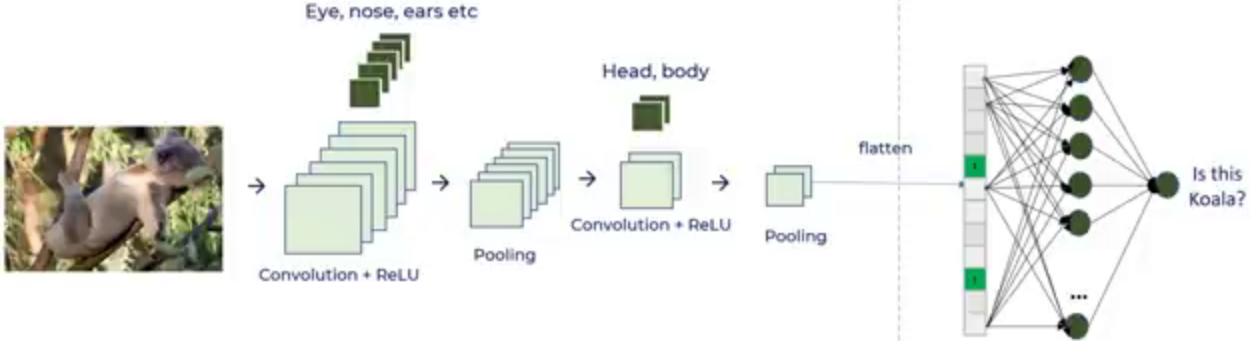
- You need to have rotated, scaled samples in training dataset
- If you don't have such samples than use data augmentation methods to generate new rotated/scaled samples from existing training samples

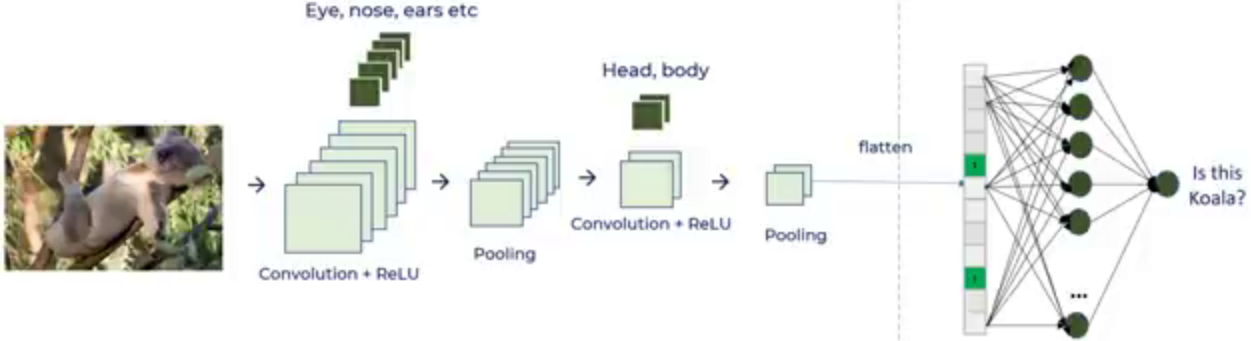
CNN by itself doesn't take care of rotation and scale

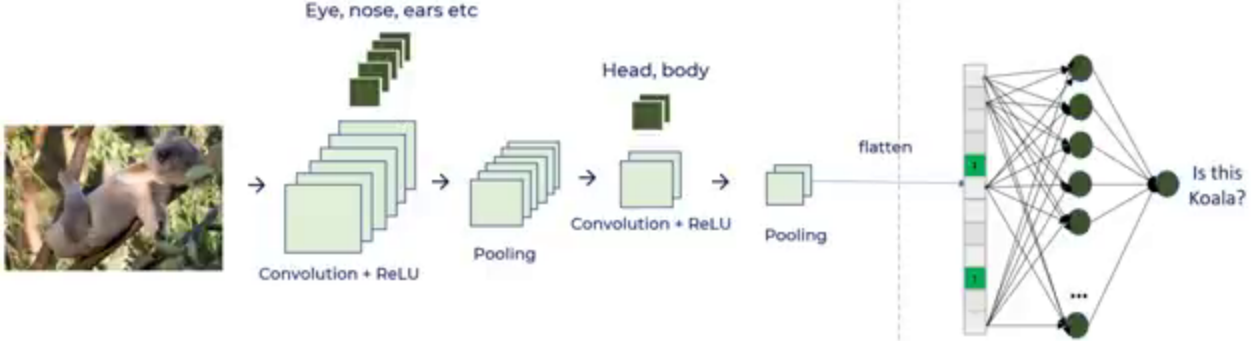
- You need to have rotated, scaled samples in training dataset
- If you don't have such samples than use data augmentation methods to generate new rotated/scaled samples from existing training samples

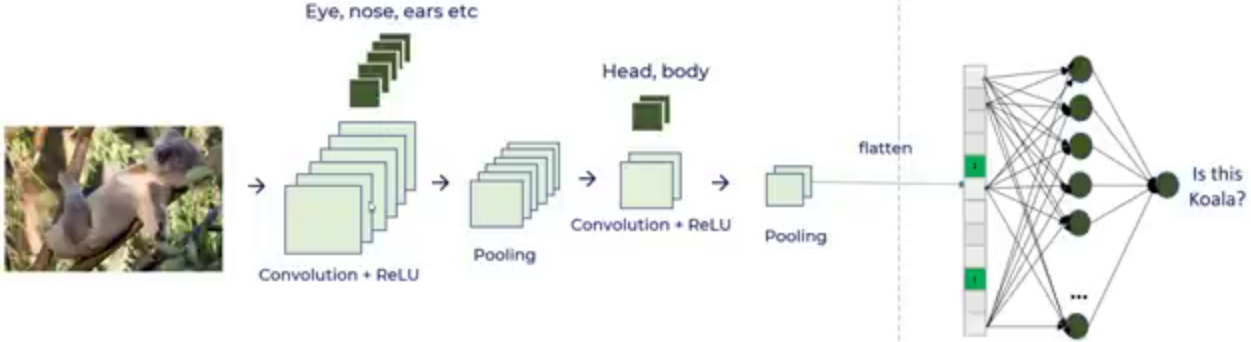

Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

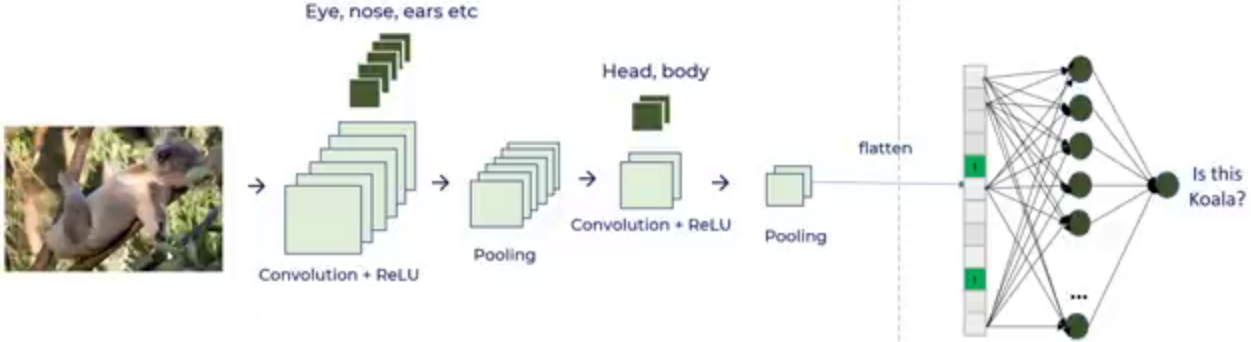

Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

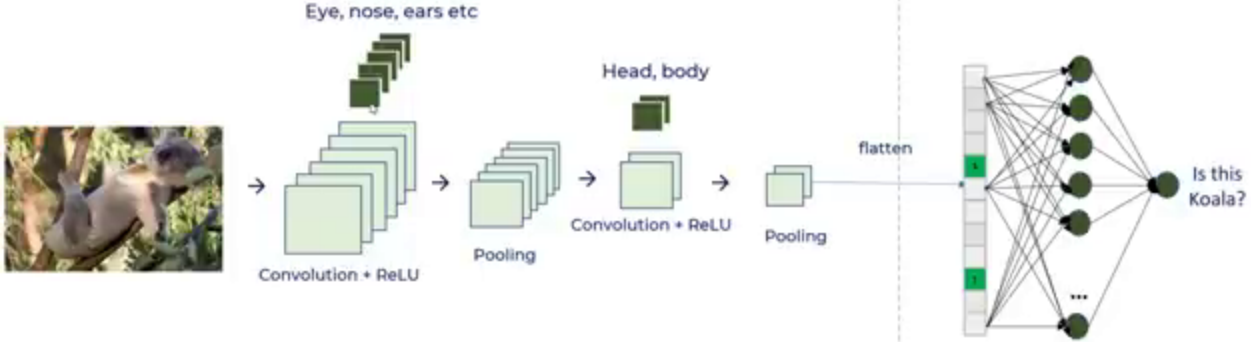

Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

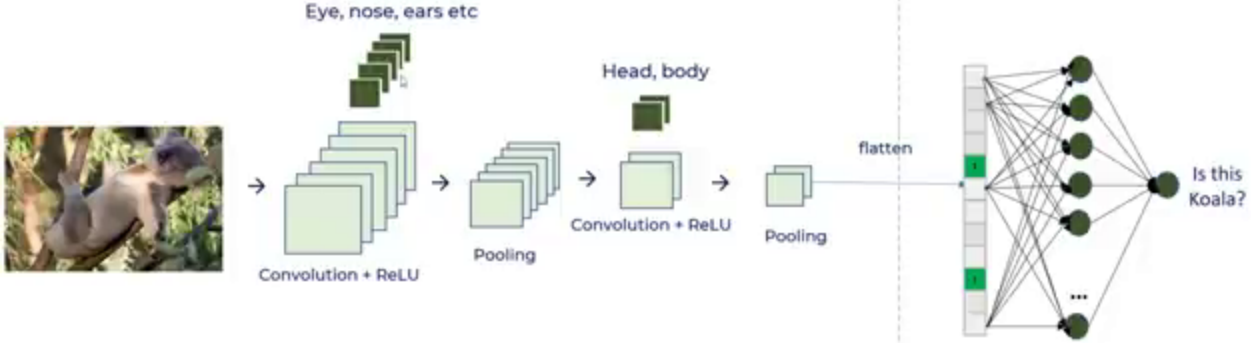

Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

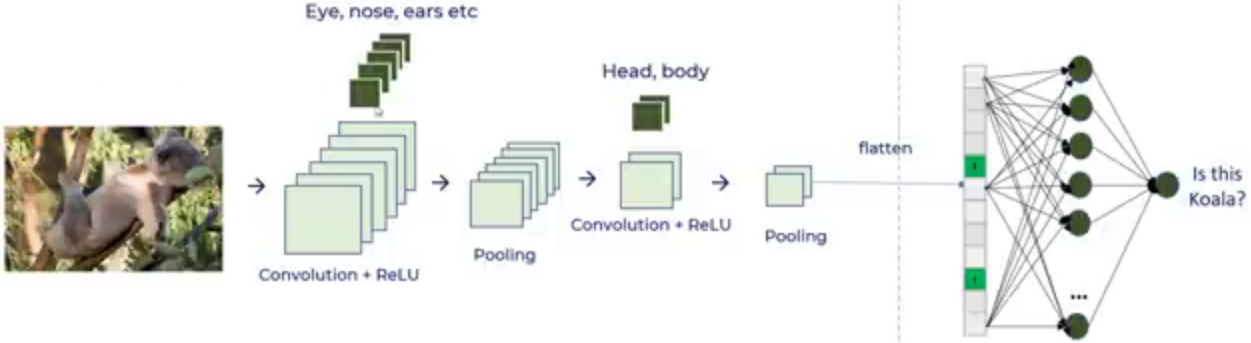

Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

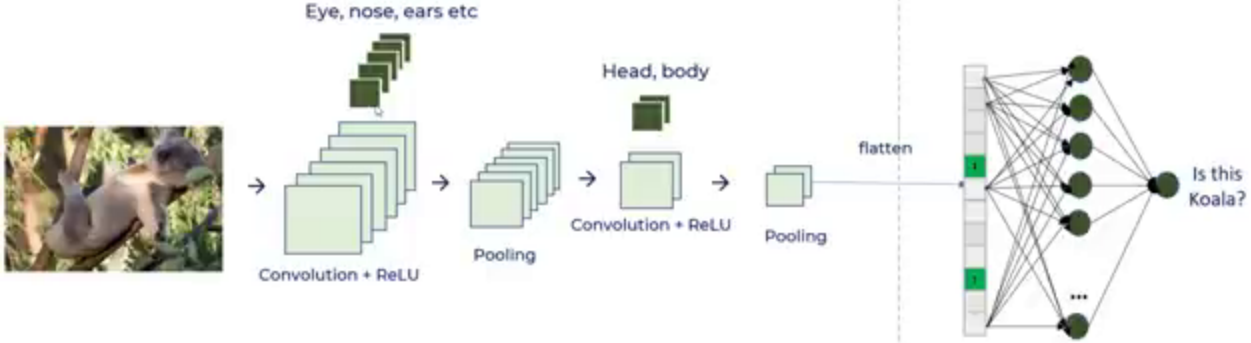

Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
...
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
...
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

Machine Learning Tutorial Python | Machine Learning For Beginners

59 videos • 560,292 views • Updated 6 days ago

Public

Machine learning tutorial playlist that is best suitable for a total beginner. It will start with basics of machine learning, cover various ML algorithms for regression and classification, feature engineering and also includes some real life exercises and end projects. In terms of technology I use these tools: sklearn,python,pandas,numpy,jupyter notebook, excel, tensorflow

codebasics

Machine Learning Tutorial Python - 9 Decision Tree

Machine Learning Tutorial Python - 10 [illegible]

Machine Learning Tutorial Python - 11 Support Vector

Machine Learning Tutorial Python - 12 [illegible]

Machine Learning Tutorial Python - 13 [illegible]

Machine Learning Tutorial Python - 14 Naive Bayes Part 1

Machine Learning Tutorial Python - 15 Naive Bayes Part 2

Machine Learning Tutorial Python - 15: Naive [illegible]

Machine Learning & Data Science Project - 1 [illegible] Price Prediction Project)

Machine Learning & Data Science Project - 2 [illegible] Price Prediction Project)

Machine Learning Tutorial Python | Machine Learning For Beginners

59 videos • 590,292 views • Updated 6 days ago

Public

Machine learning tutorial playlist true is best suitable for a total beginner. It will start with basics of machine learning, cover various ML algorithms for regression and classification, feature engineering and also includes some real life end to end projects. In terms of technology I use these tools: sklearn,python,pandas,numpy,jupyter notebook, excel, tensorflow

codebasics

Data Science & Machine Learning Project - Part 7 Build Website | Image Classification

Data Science & Machine Learning Project - Part 8 Deployment & Exercise | Image Classification

What is feature engineering | Feature Engineering Tutorial Python # 1

Outlier detection and removal using percentile | Feature engineering tutorial python # 2

Outlier detection and removal: z score, standard deviation | Feature engineering tutorial python # 3

Outlier detection and removal using IQR | Feature engineering tutorial python # 4

Introduction | Deep Learning Tutorial 1 (Tensorflow2.0, Keras & Python)

Why deep learning is becoming so popular? | Deep Learning Tutorial 2 (Tensorflow2.0, Keras & Python)

What is a neuron? | Deep Learning Tutorial 3 (Tensorflow2.0, Keras & Python)

Very Simple Explanation Of Neural Network | Deep Learning Tutorial 4 (Tensorflow2.0, Keras & Python)