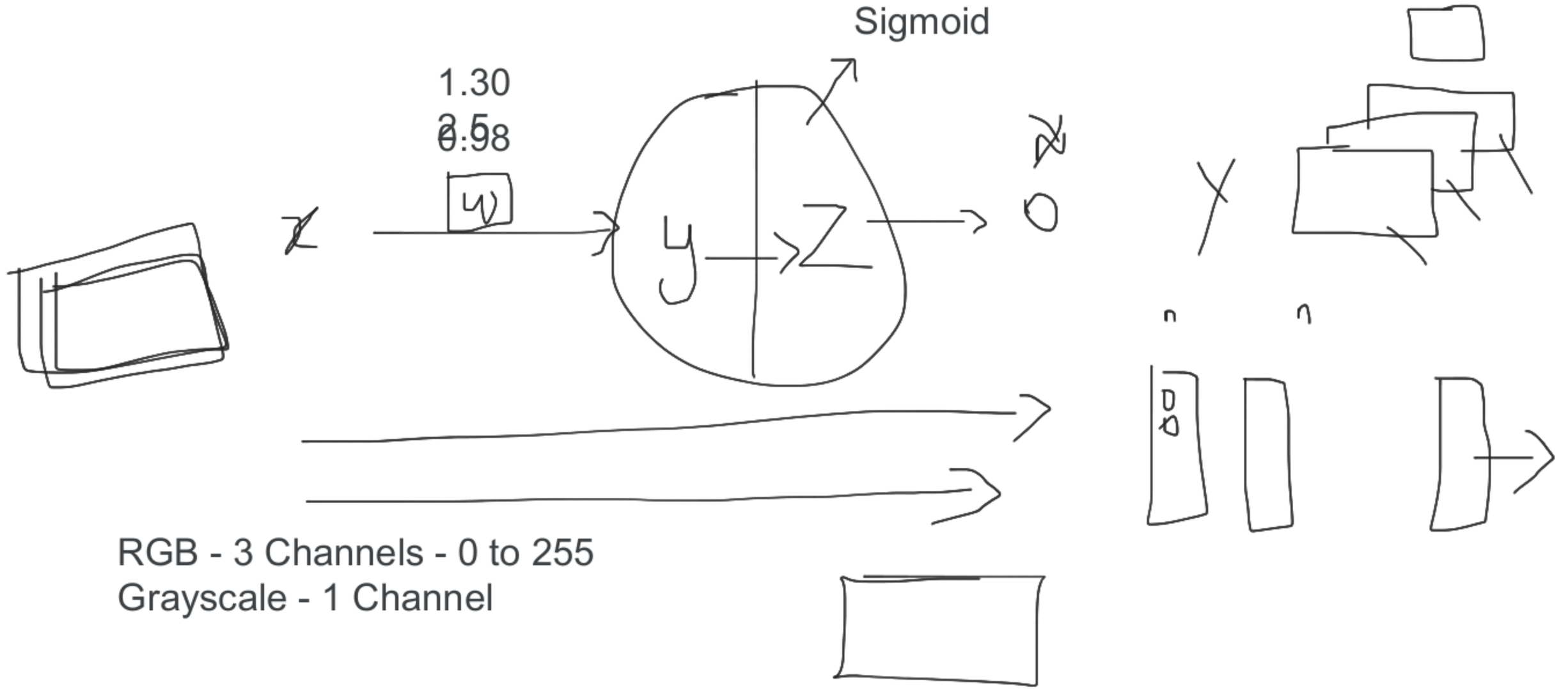
ANN

CNN - Convolutional Neural Network

Neural Network
Neural Nets



RGB - 3 Channels - 0 to 255
Grayscale - 1 Channel

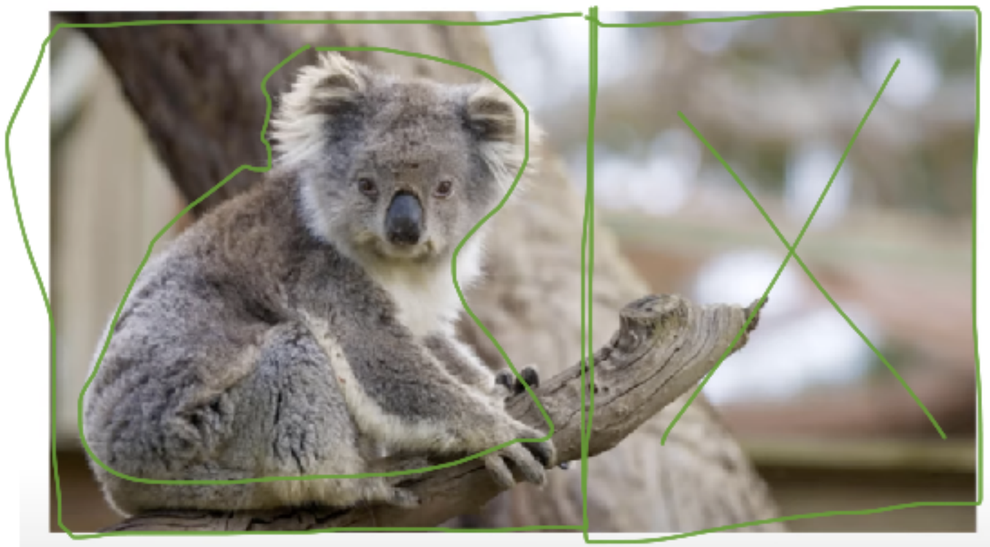


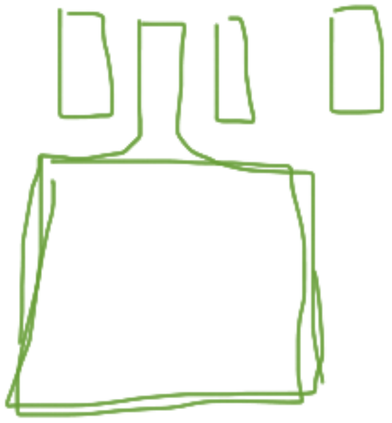
Image size = 1920 x 1080 X 3

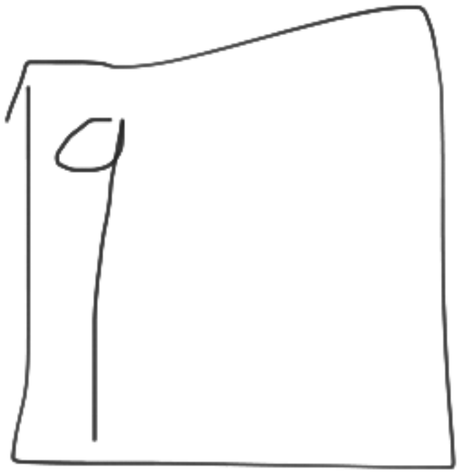
First layer neurons = 1920 x 1080 X 3 ~ 6 million

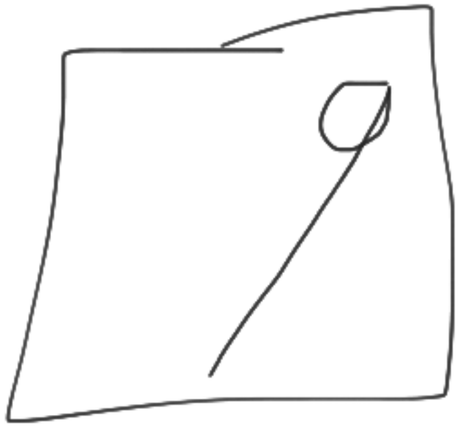
Hidden layer neurons = Let's say you keep it ~ 4 million

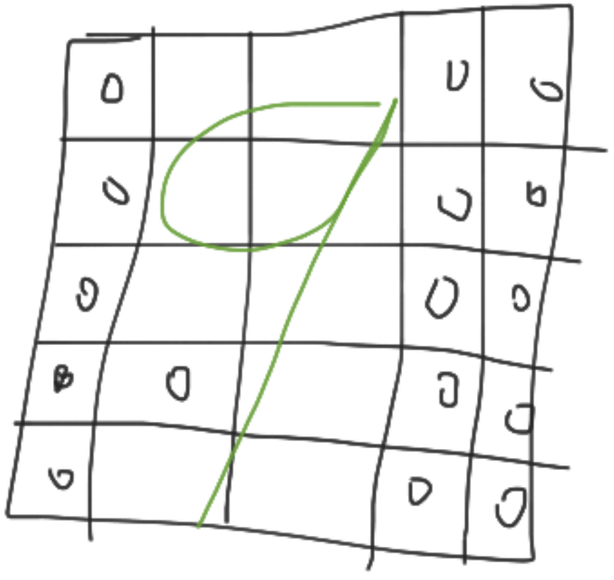
Weights between input and hidden layer = ~~6 mil * 4 mil~~
= 24 million

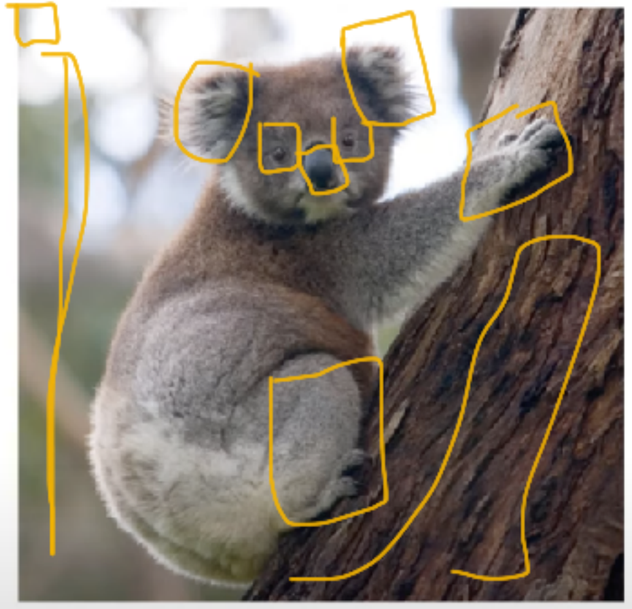
1. Too much computation
2. Time consuming











Koala's eye? = Y

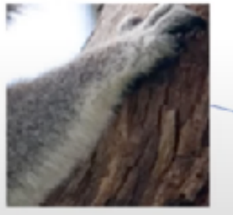
Koala's nose? = Y

Koala's ears? = Y

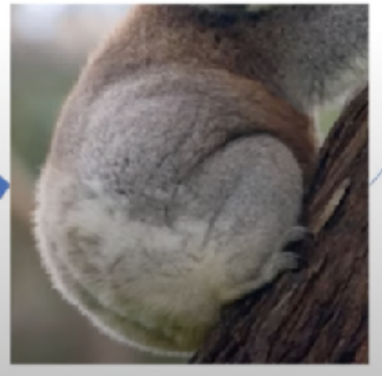
Koala's head? = Y

Is it Koala? = Y

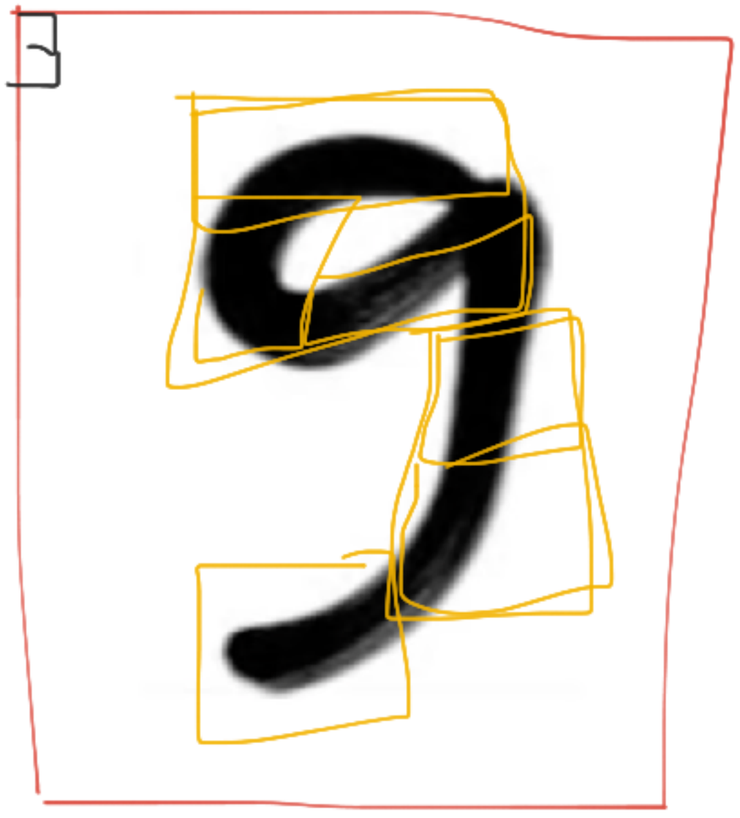
Koala's hands? = Y

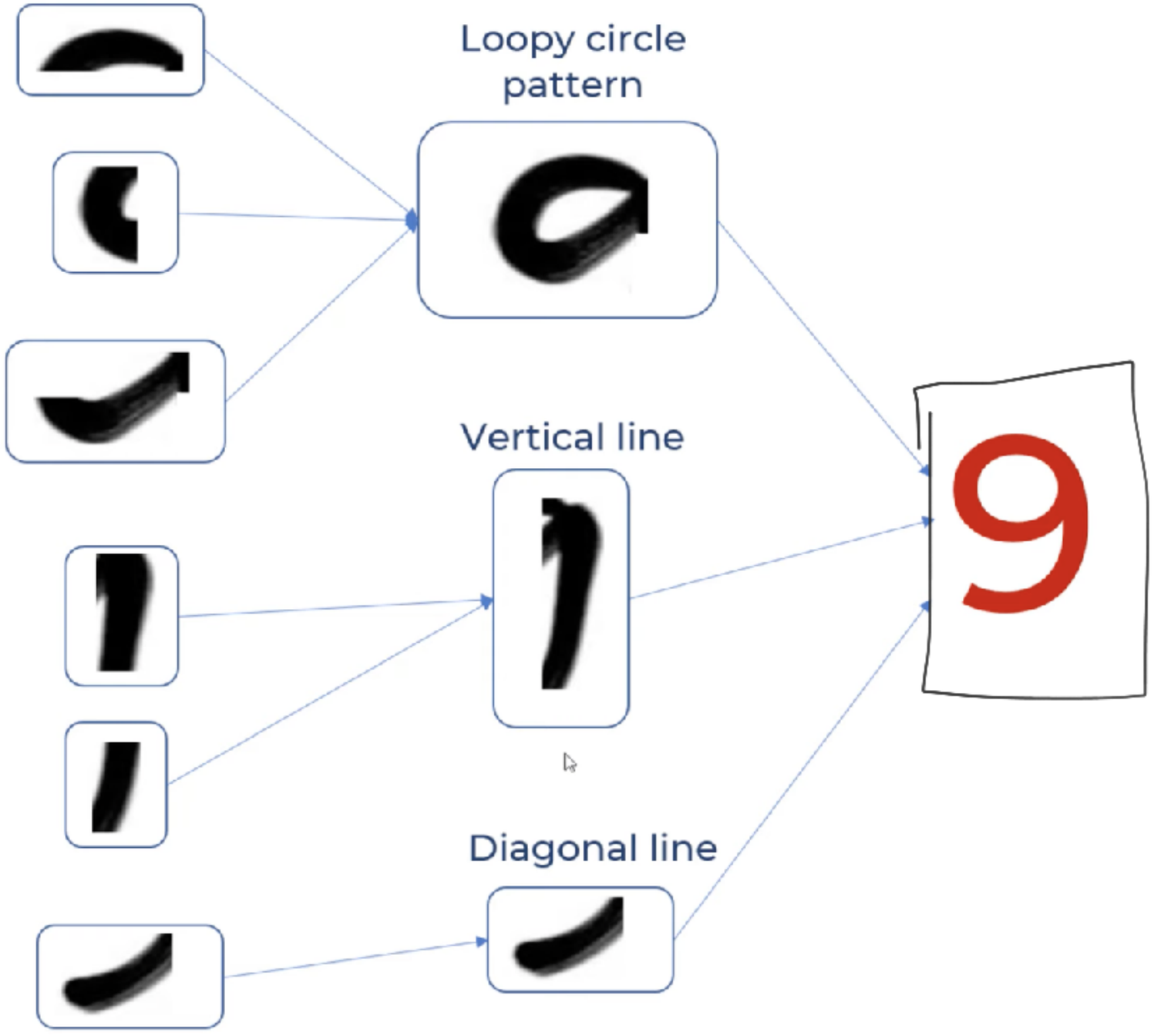


Koala's body? = Y



Koala's legs? = Y




Loopy circle pattern
Vertical line
Diagonal line
9

Remini

CNN - Convolutional Neural Network

OpenCV

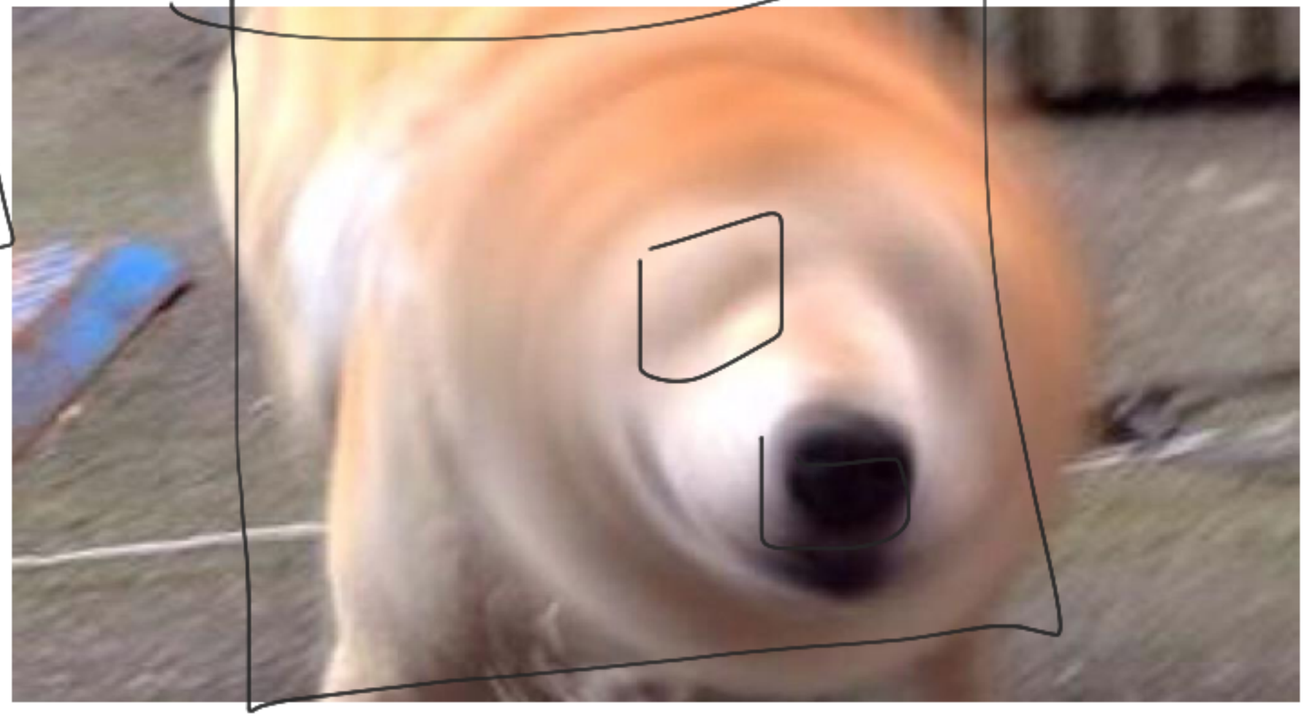
Image Processing

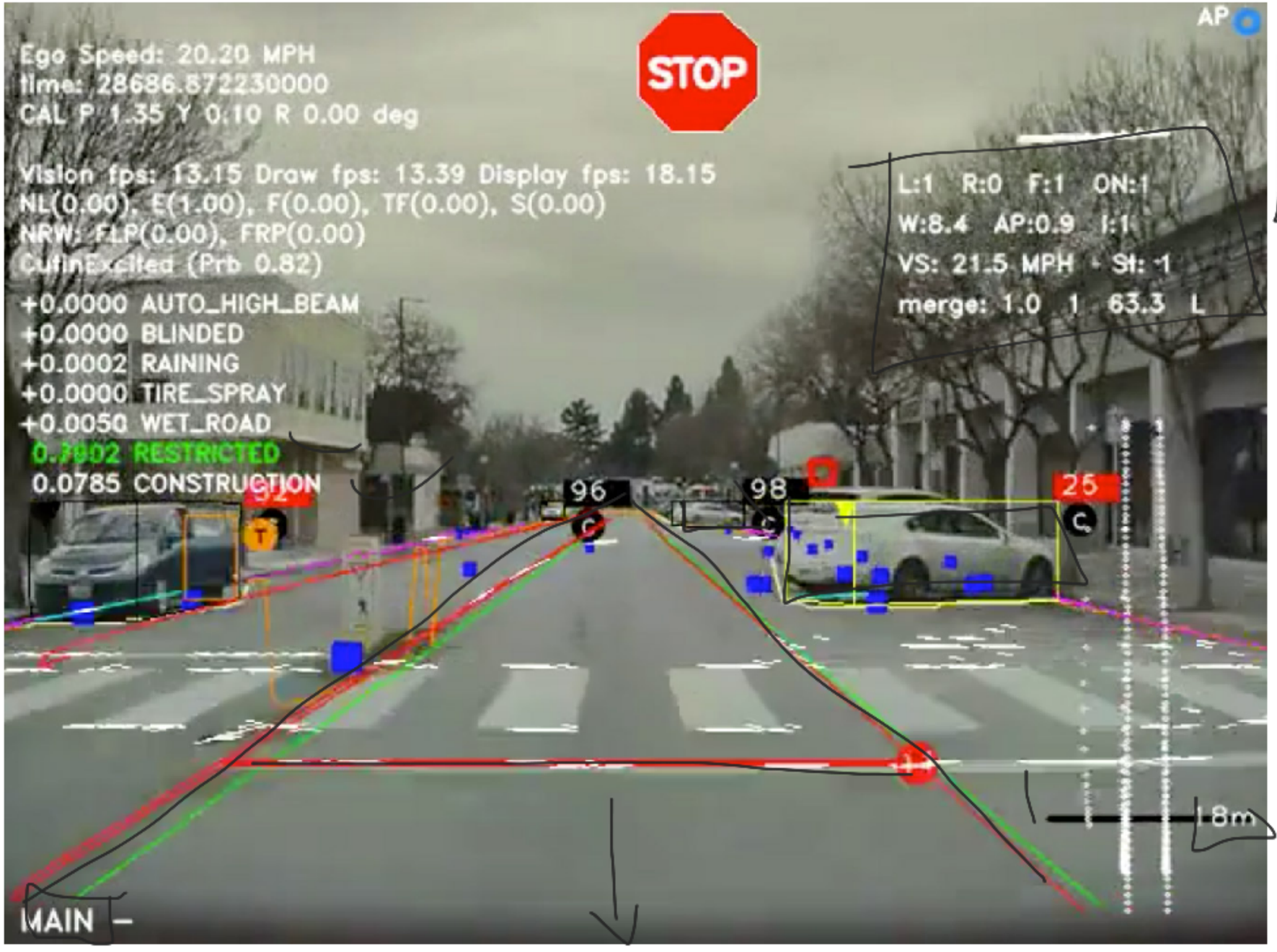
Computer Vision

Camera

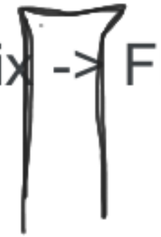
Optical Zoom

Digital Zoom




Ego Speed: 20.20 MPH
time: 28686.872230000
CAL P 1.35 Y 0.10 R 0.00 deg
Vision fps: 13.15 Draw fps: 13.39 Display fps: 18.15
NL(0.00), E(1.00), F(0.00), TF(0.00), S(0.00)
NRW: FLP(0.00), FRP(0.00)
CutinExcited (Prb 0.82)
+0.0000 AUTO_HIGH_BEAM
+0.0000 BLINDED
+0.0002 RAINING
+0.0000 TIRE_SPRAY
+0.0050 WET_ROAD
0.7802 RESTRICTED
0.0785 CONSTRUCTION
STOP
AP
L:1 R:0 F:1 ON:1
W:8.4 AP:0.9 I:1
VS: 21.5 MPH St: 1
merge: 1.0 1 63.3 L
96
98
25
8m
MAIN

ANN -> Digital Foramat -> Matrix -> Flatten the matrix -> Neural Nets -> Output



CNN -> Input -> CNN Layers -> ANN

CNN Layers/Steps

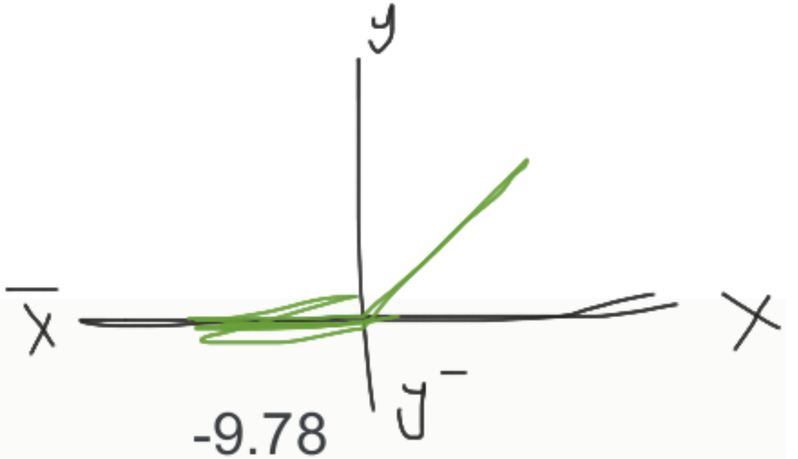
1. Convolution (Filter/Kernel) + RelU
2. Pooling - Max or Avg
3. Convolution (Filter) + RelU
4. Pooling - Max or Avg
5. Convolution (Filter) + RelU
6. Pooling - Max or Avg

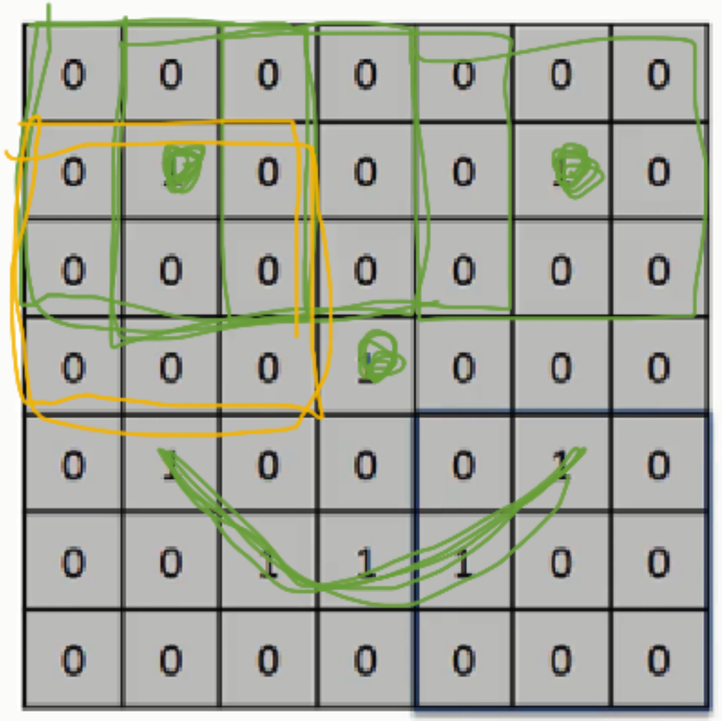
Convolutional Block

Fully Connected Block

1. Convolutional (Filter/Kernel)

7X7 -> 5X5





Input Image

Eyes

0	0	1
1	0	0
0	1	1

Feature Detector

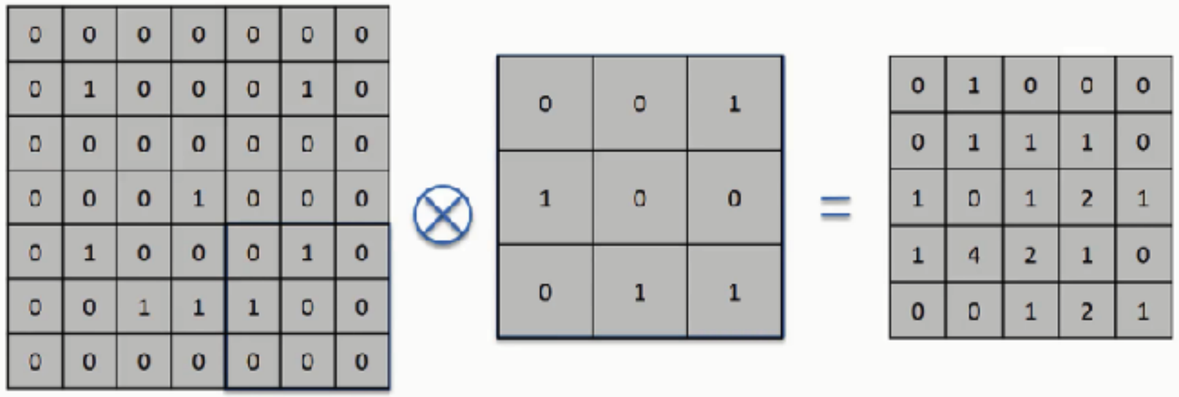
-9.78

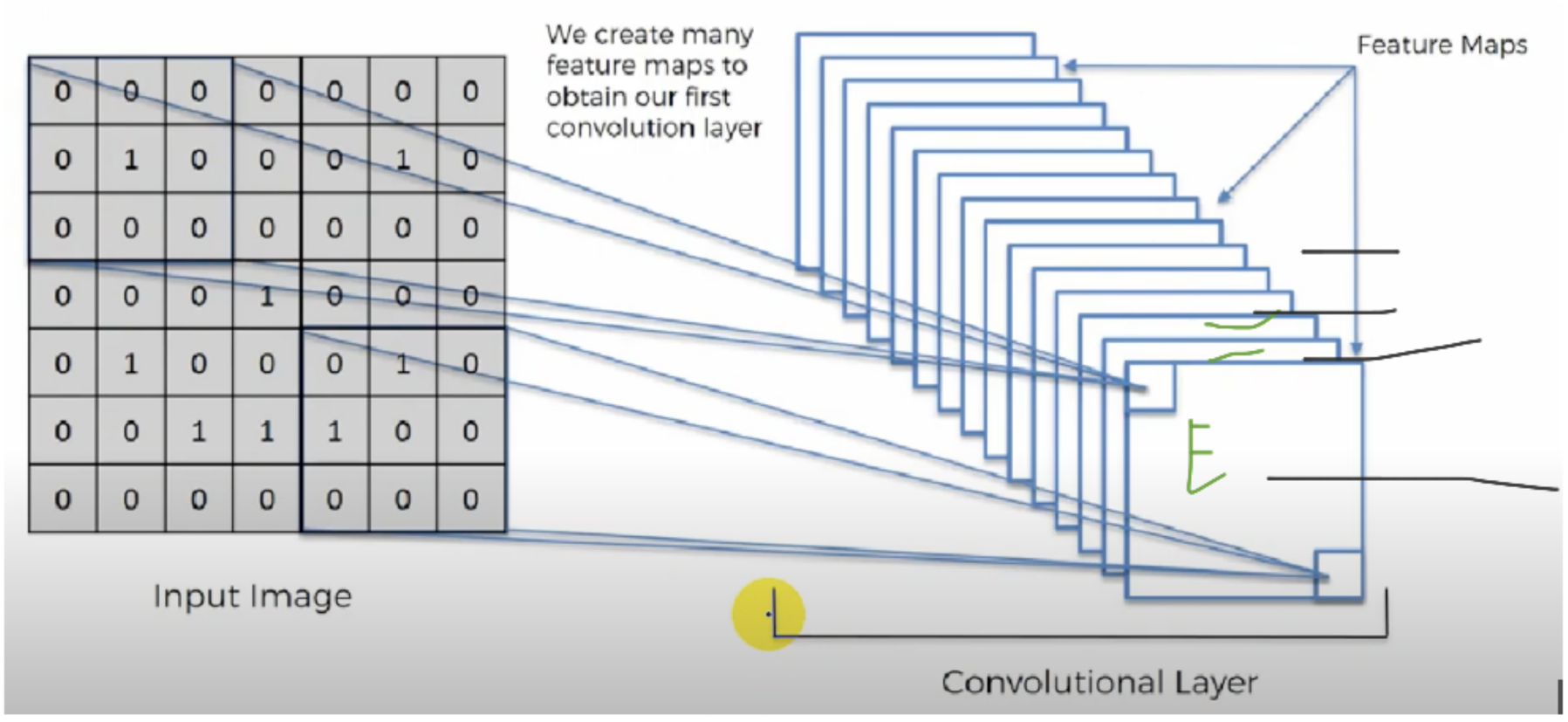
0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Convolved Feature

Feature Map

```
def relu(x):
    if x < 0:
        return 0
    else:
        return x
```


0 0 0 0 0 0 0
0 1 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 0 1 0
0 0 1 1 1 0 0
0 0 0 0 0 0 0
0 0 1
1 0 0
0 1 1
0 1 0 0 0
0 1 1 1 0
1 0 1 2 1
1 4 2 1 0
0 0 1 2 1
Input Image
Feature Detector
Feature Map


We create many feature maps to obtain our first convolution layer
Feature Maps
0 0 0 0 0 0 0
0 1 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 1 0 0 0 1 0
0 0 1 1 1 0 0
0 0 0 0 0 0 0
Input Image
Convolutional Layer

2. Pooling Layer (Max/Avg)

It will going to perform thge process of extraction of particular values from the set of values, usually tha max value or avg value is used.

The motive is to reduce the size of the output matrix

2x2

7x7 -> 5x5 -> 3x3

0	1
0	0
1	1

=

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

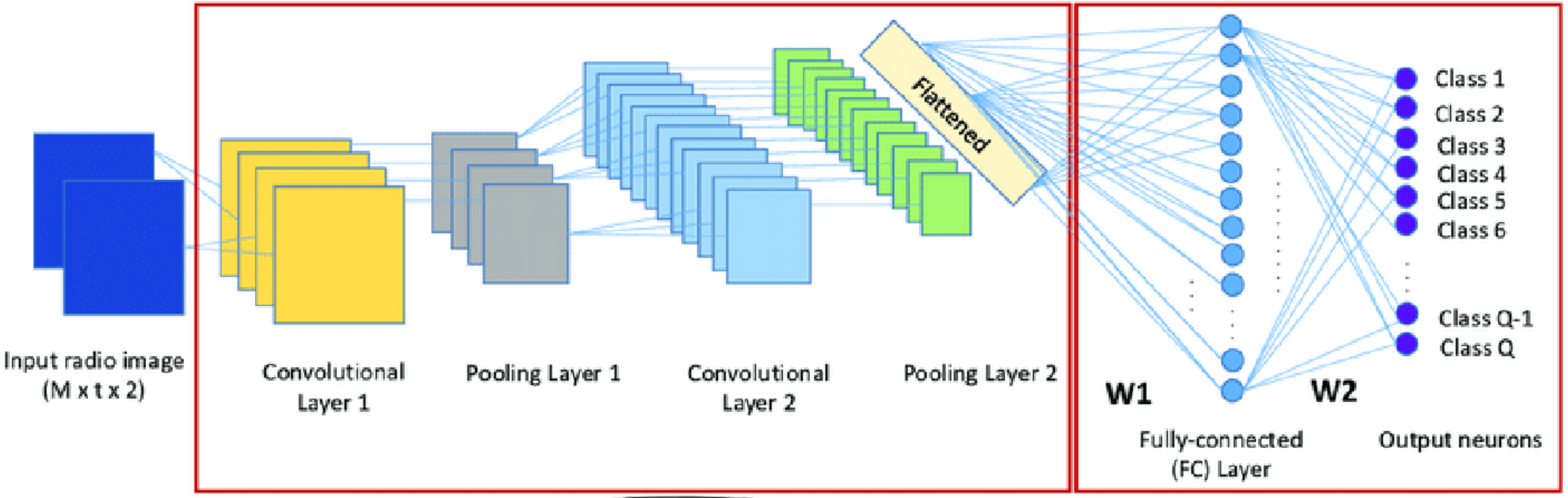
Feature Map

1	1	0
4	2	1
0	2	1

Pooled Feature Map

ure
ctor

ANN

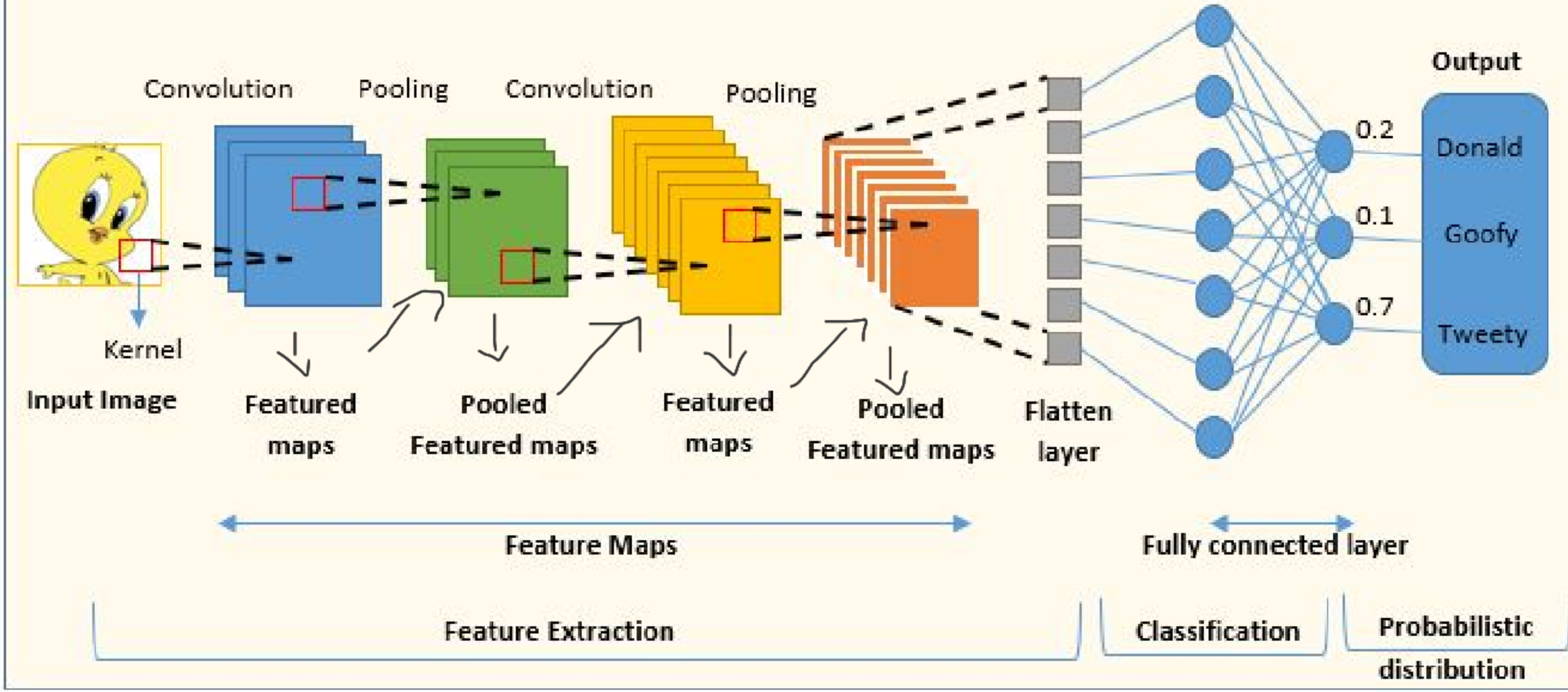


Features extraction

Classification

3

A Typical Convolutional Neural Network (CNN)



CNN Explainer: https://poloclub.github.io/cnn-explainer/

CNN Kernel: https://setosa.io/ev/image-kernels/