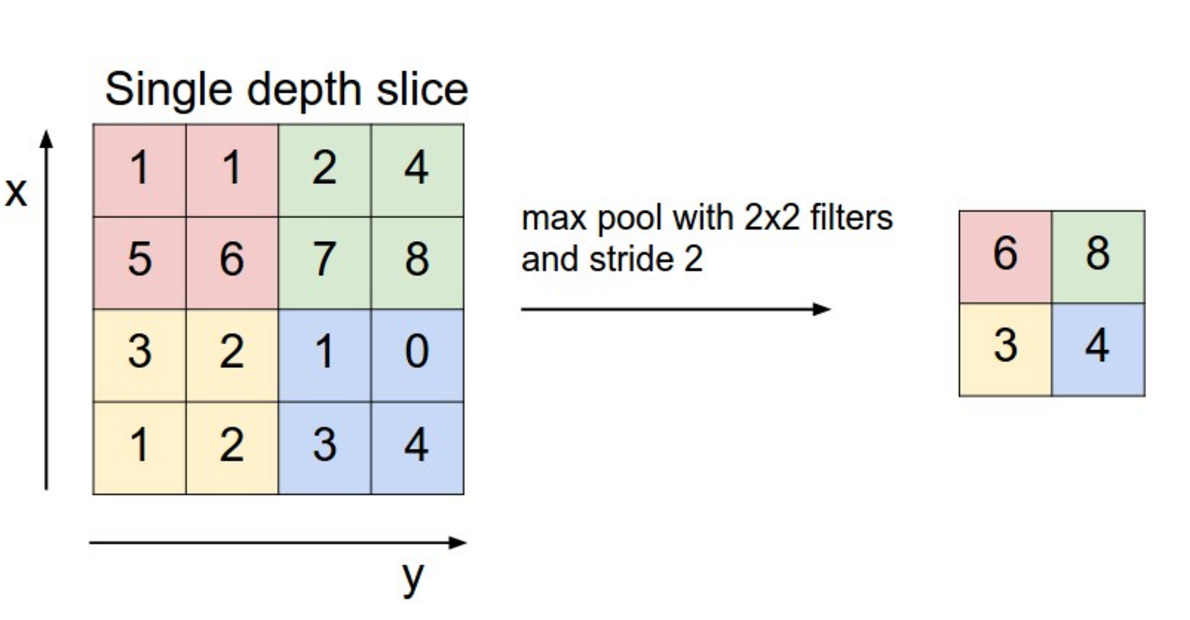
**Convolutional Neural Network**

**Short note about Neural Network**:

In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.Invloves Object identification.







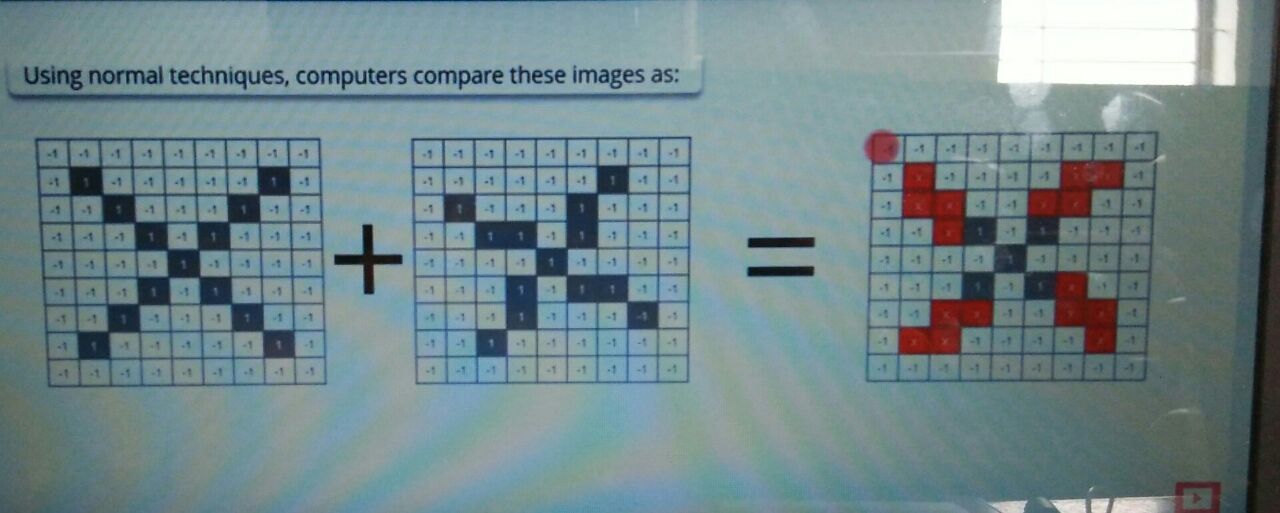
Convolutional Network involves 3 layers

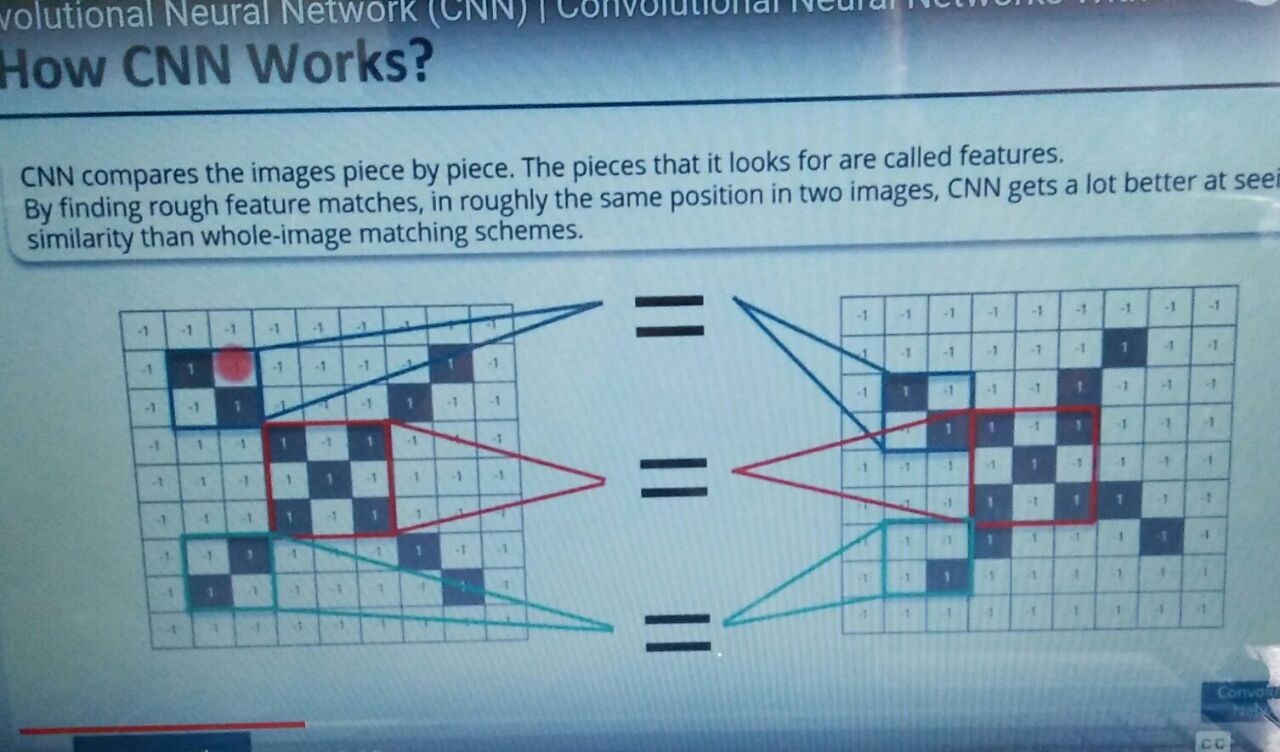
1.Convolution layer

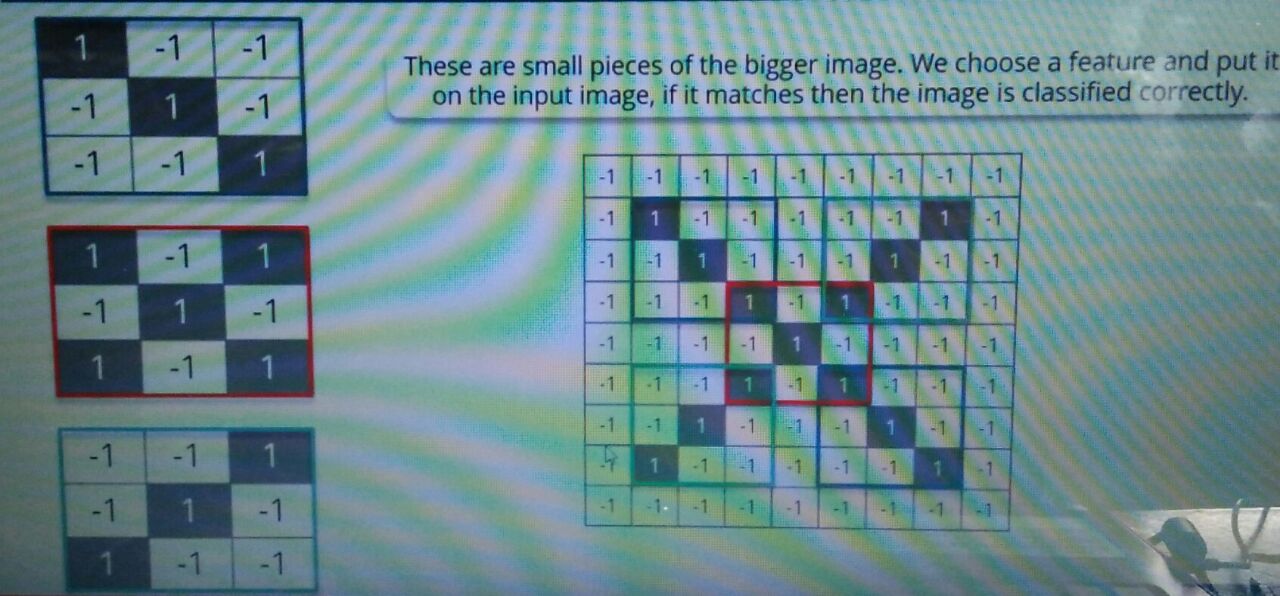
2.ReLu Layer

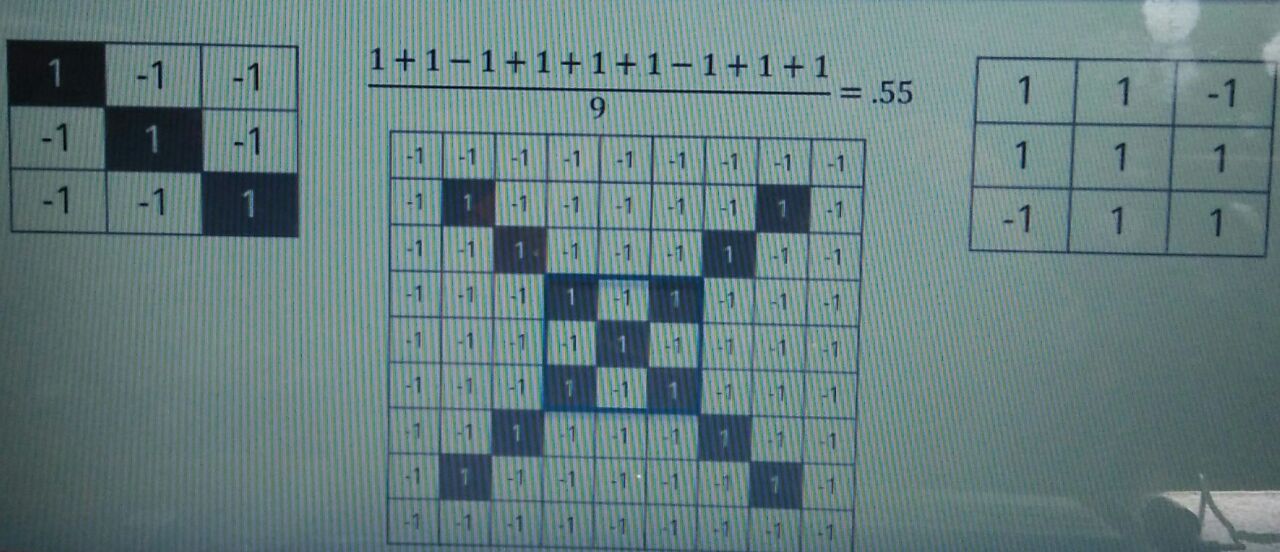
3.Pooling Layer

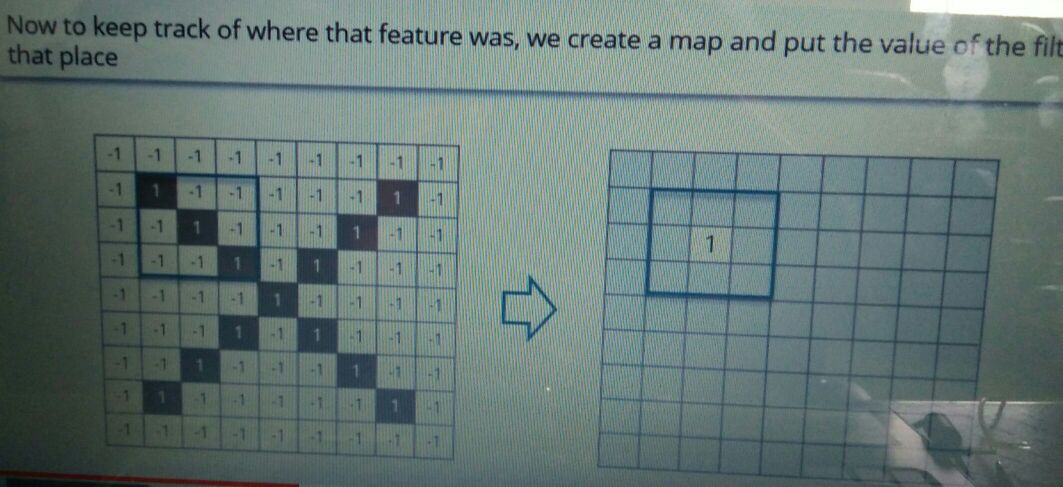
Convolution Layer-

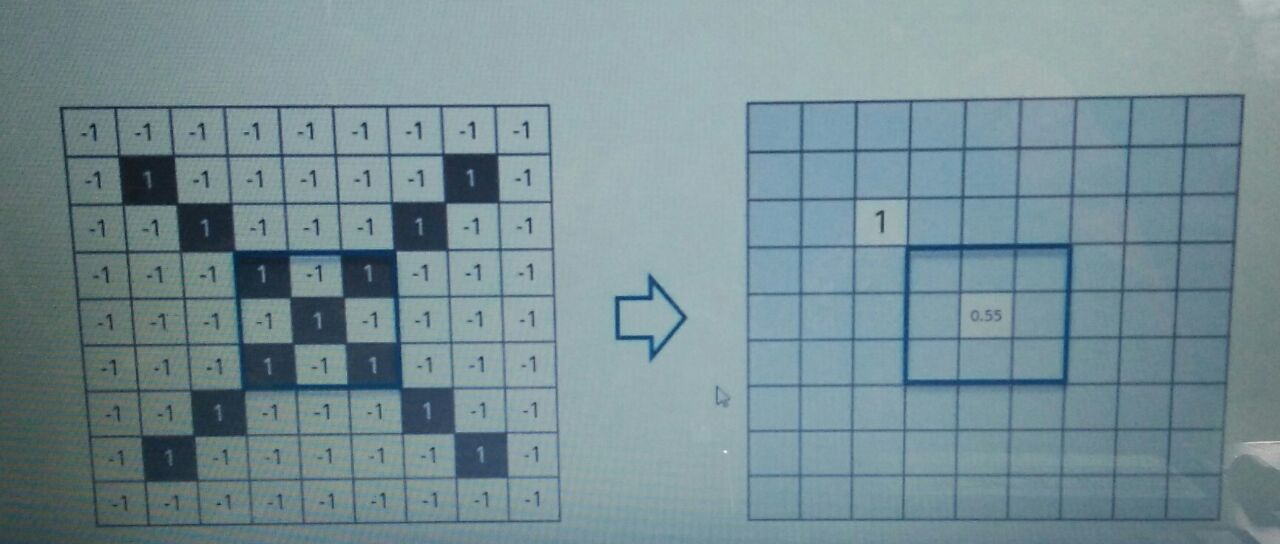


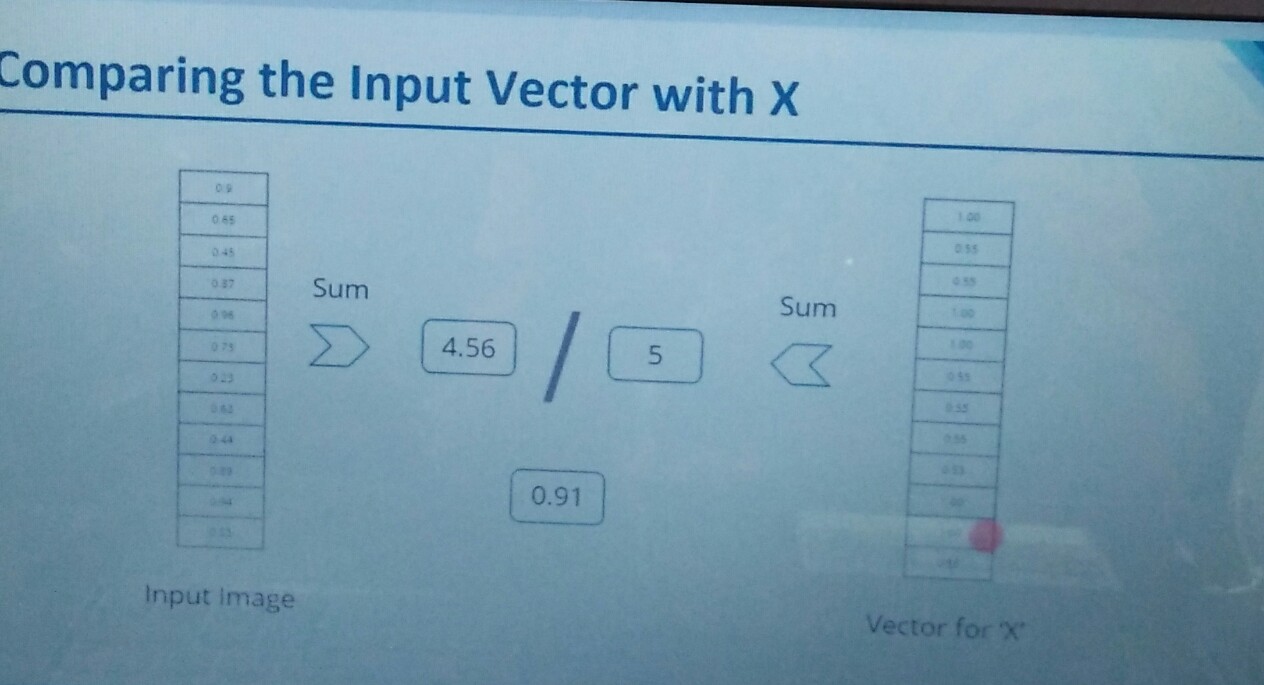


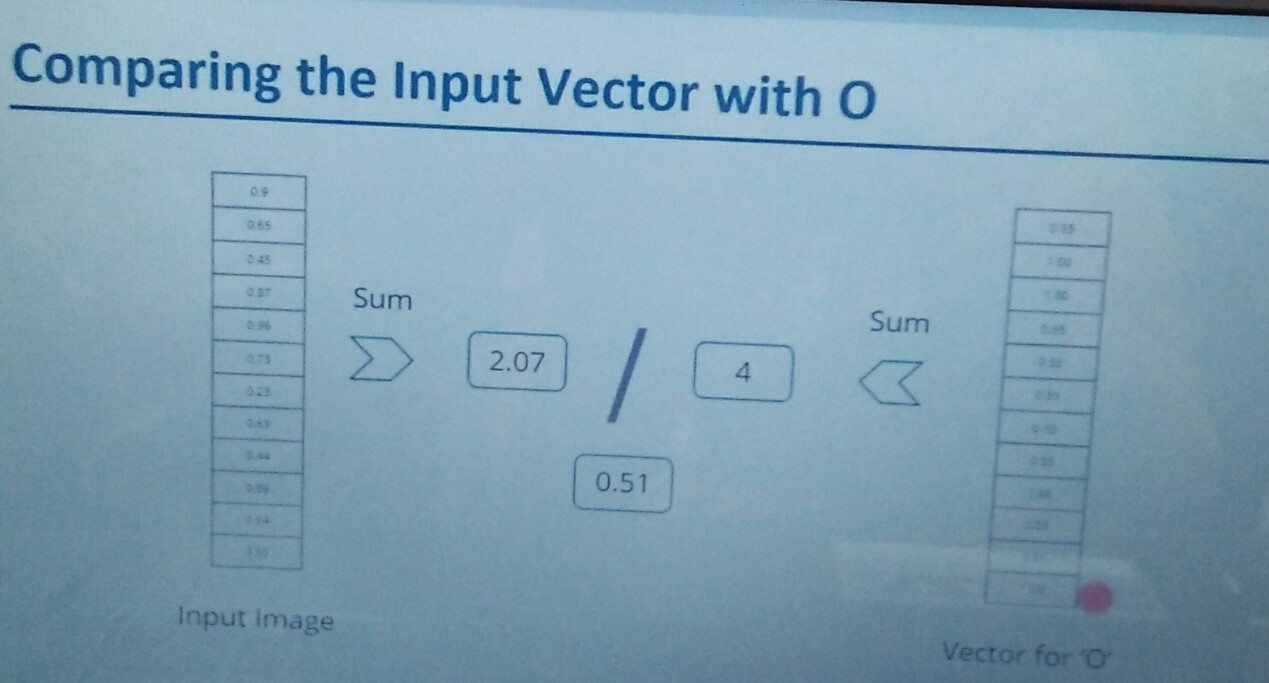


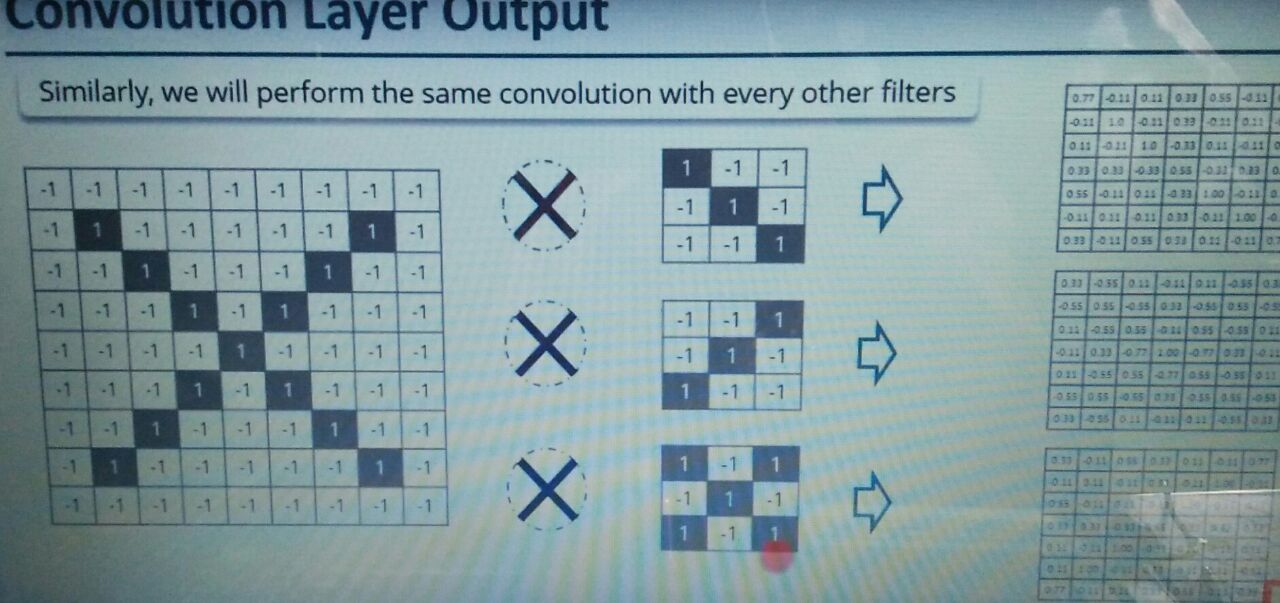


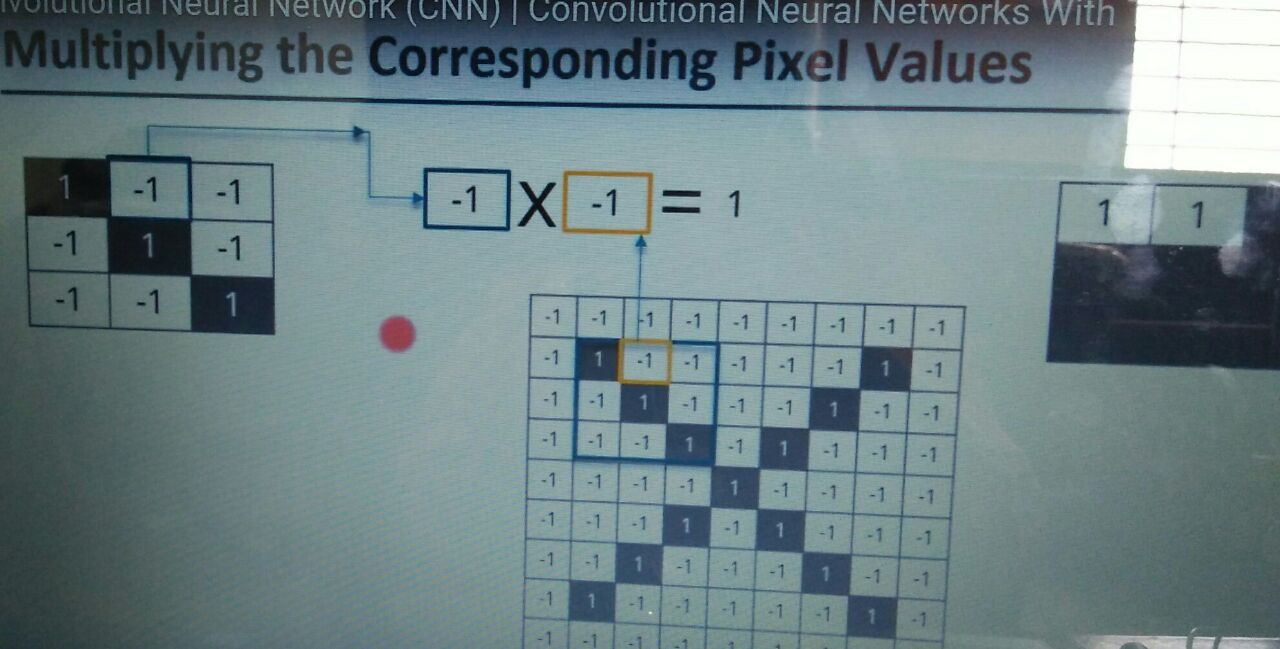


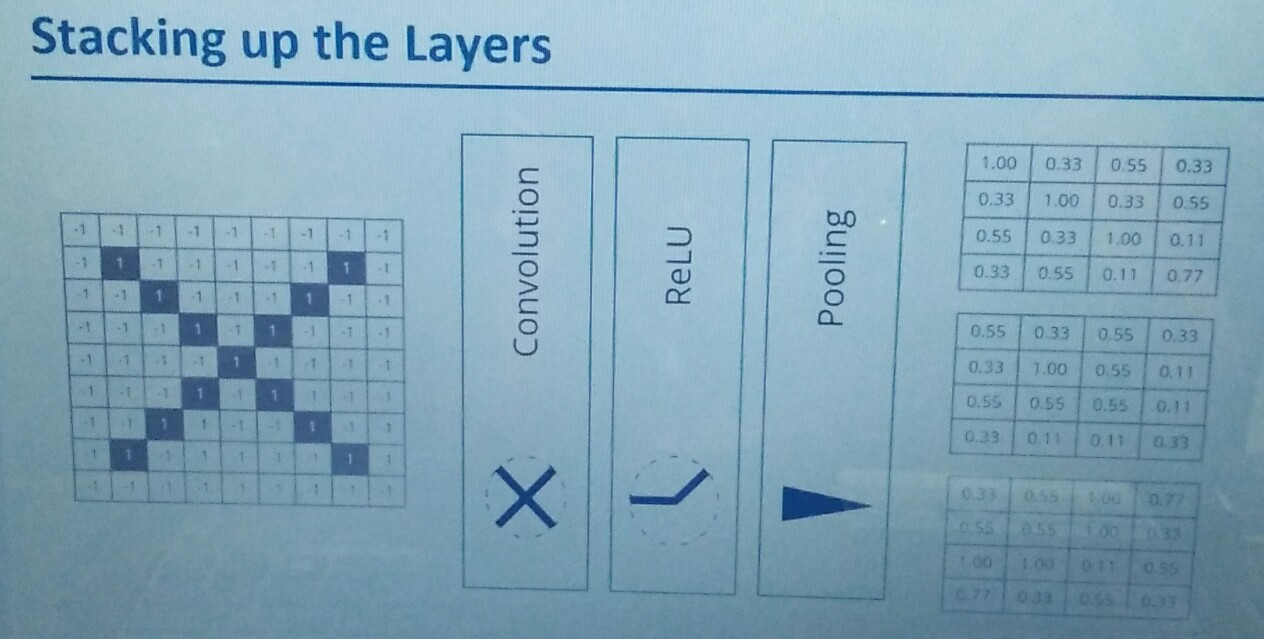


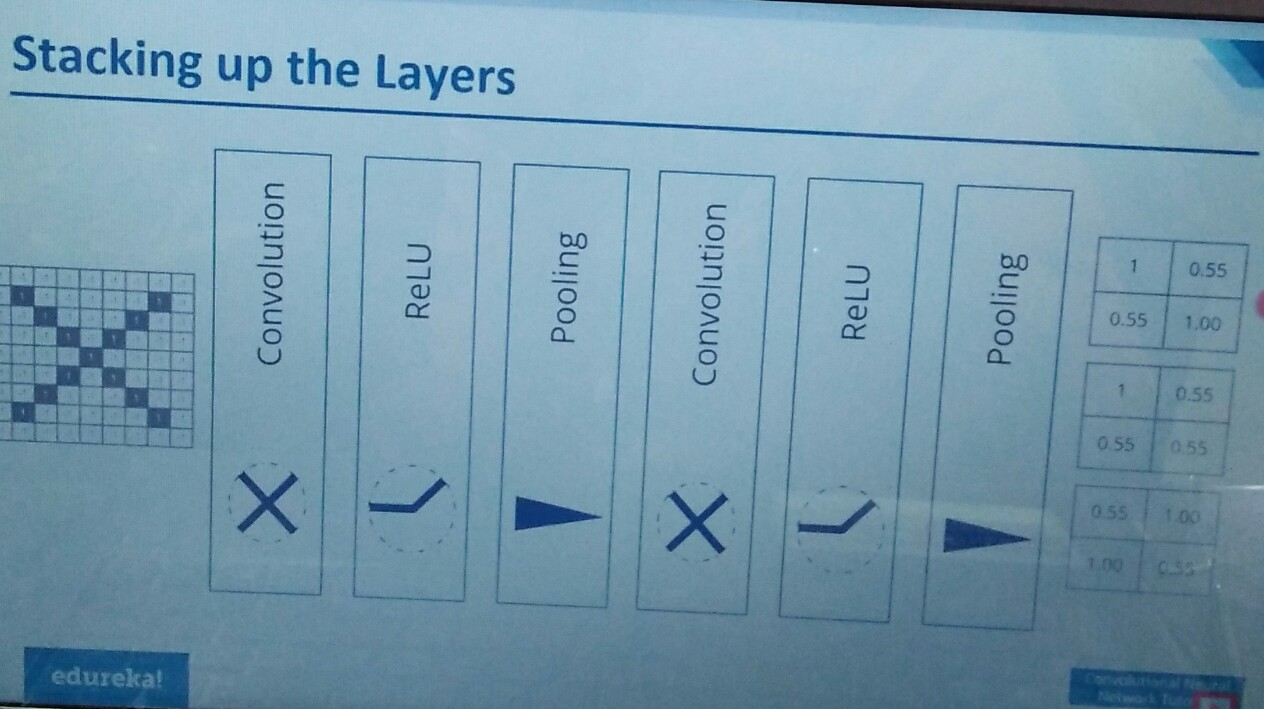


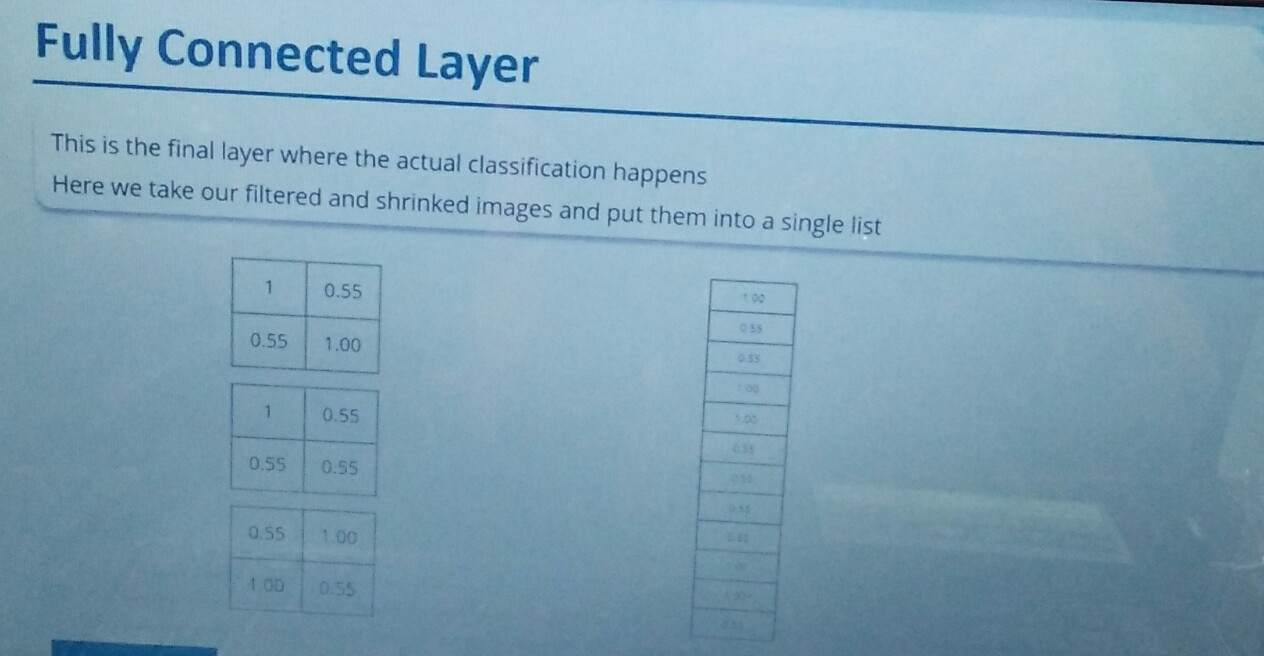


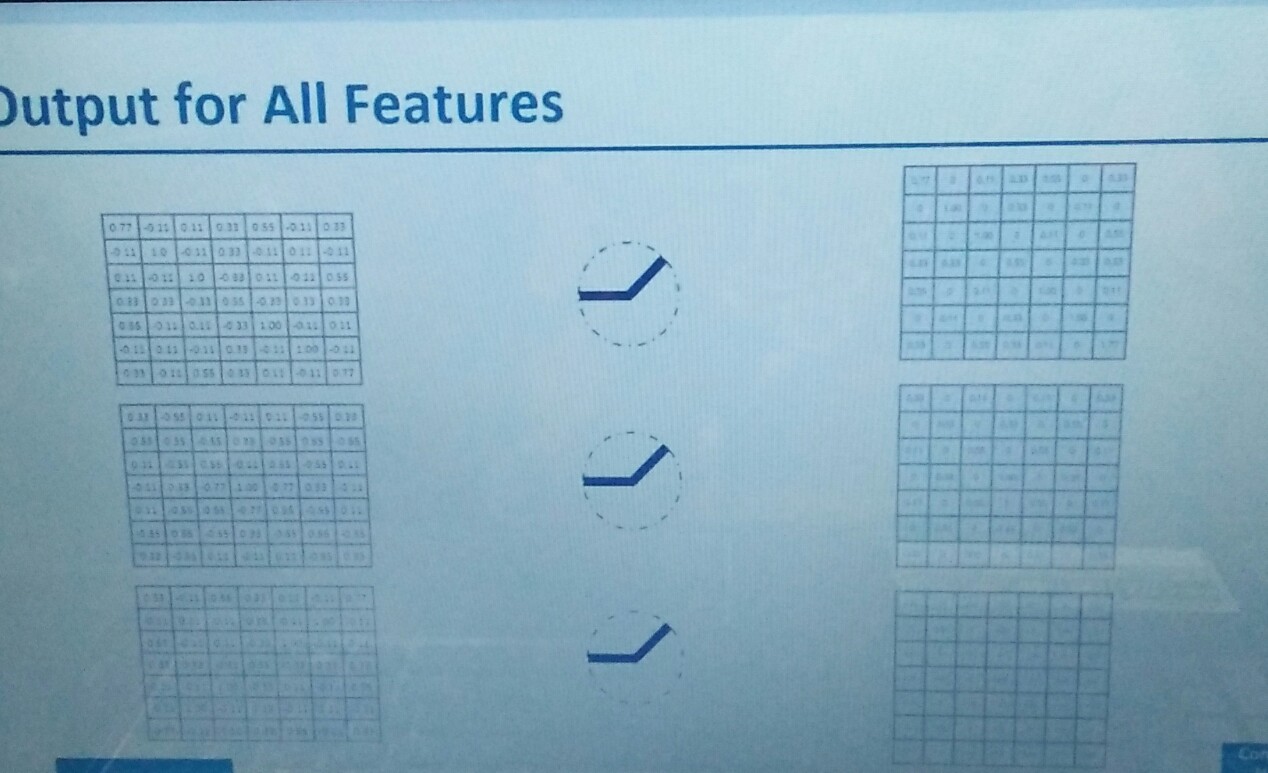


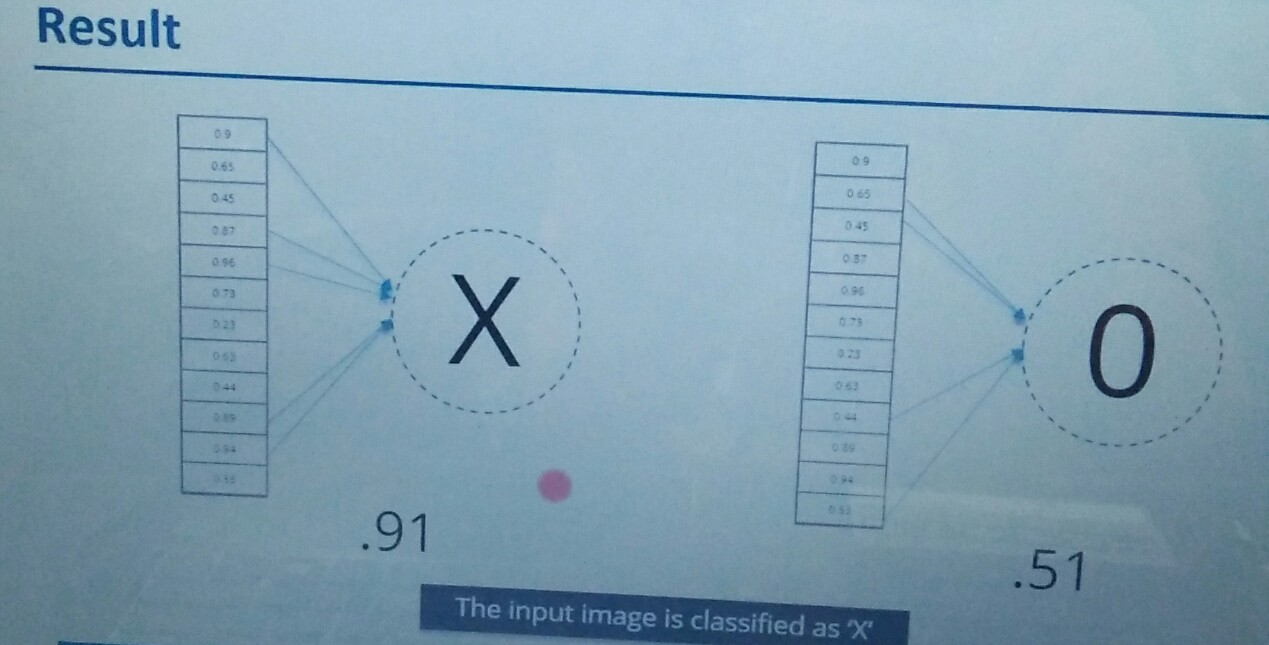


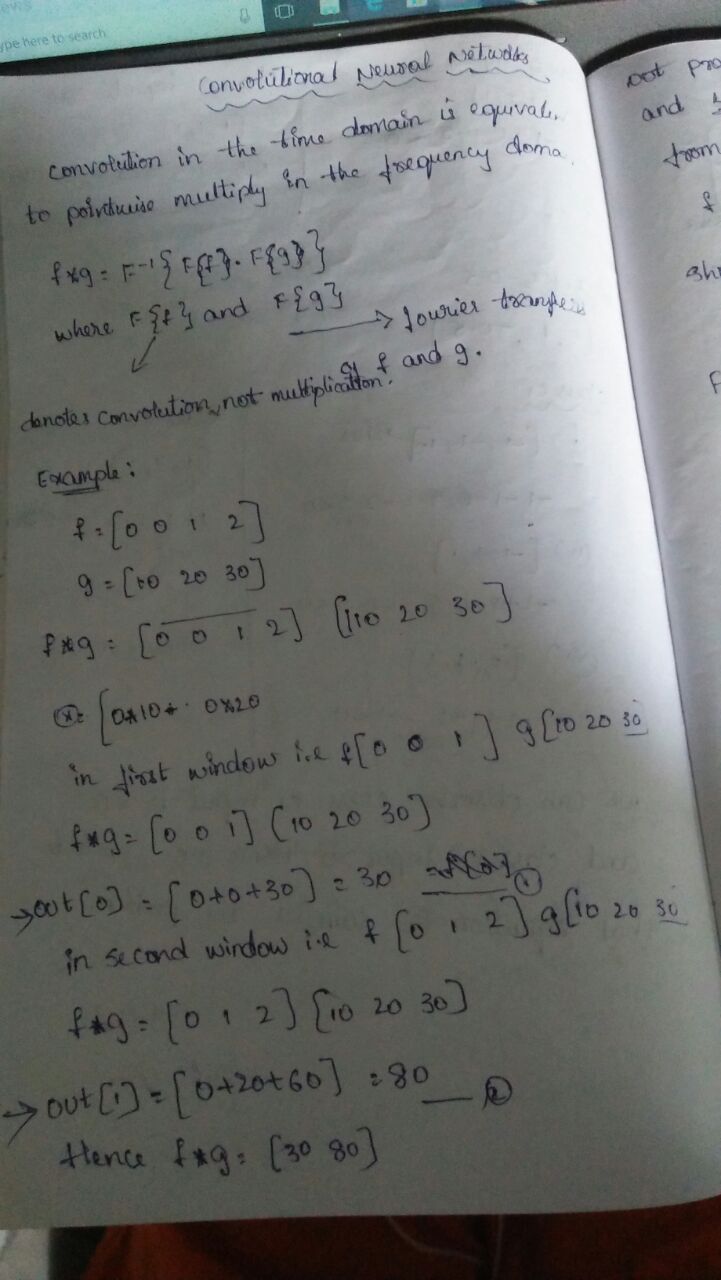


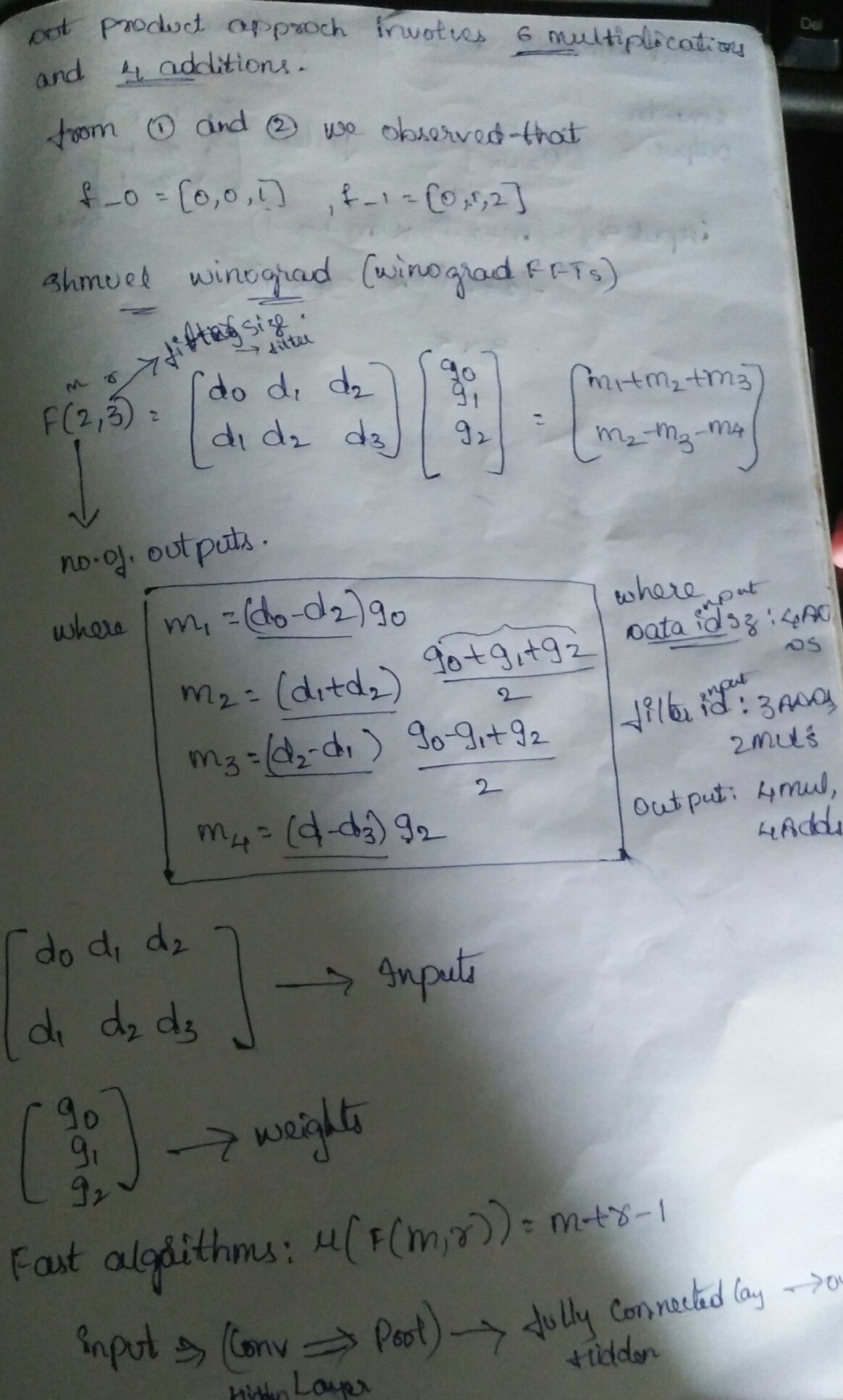










****

**Use Cases-**

* Automation driving
* Mobile
* cameras.
* To find the dimensions of the room.
* To create the objects.
* To move the objects.
* Finding the changes in Pollution
* To check the existence of the customer in the bank.
* To check the market strategy.
* To find the price fluctuation of Amazon products.

Python-

# Importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Conv2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

# Initialising the CNN

classifier = Sequential()

# Step 1 - Convolution

classifier.add(Conv2D(32, (3, 3), input\_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Adding a second convolutional layer

classifier.add(Conv2D(32, (3, 3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# Step 3 - Flattening

classifier.add(Flatten())

# Step 4 - Full connection

classifier.add(Dense(units = 128, activation = 'relu'))

classifier.add(Dense(units = 1, activation = 'sigmoid'))

# Compiling the CNN

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale = 1./255)

training\_set = train\_datagen.flow\_from\_directory(r"C:\Users\Rama\Desktop\New folder",

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

test\_set = test\_datagen.flow\_from\_directory(r"C:\Users\Rama\Desktop\New folder",

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

classifier.fit\_generator(training\_set,

steps\_per\_epoch = 8000,

epochs = 25,

validation\_data = test\_set,

validation\_steps = 2000)