**DEEP LEARNING**

TASKS

1. **detect ripe and unripe Apple:**

**TASK STEPS:**

**1- input will be pictures it is not real time task and no physical actions.**

**#2- generalize on any types of fruits or vegetables if want**

**3- real time task using web camera, input will be video.**

**4- hard camera +SOM + physical action by a robot as picking ripe apple.**

**1- input will be pictures it is not real time task and no physical actions.**

**1-Data:-**

* **680 Ripe apples:**

**410 Multi hang +noise**

**270 Solo + no noise**

* **380 Unripe apples**

**All multi hang + noise**

**Multi hang +noise  multi hang + noise**

****

**Solo + no noise**

**Problems with data:**

**All dataset allowed for solo types, so we could solve this problem by collecting different data from many websites using web scraping technique.**

**Web scraping by automation application: UIPATH**

****

**2-Achievements:-**

**Understanding this concepts:**

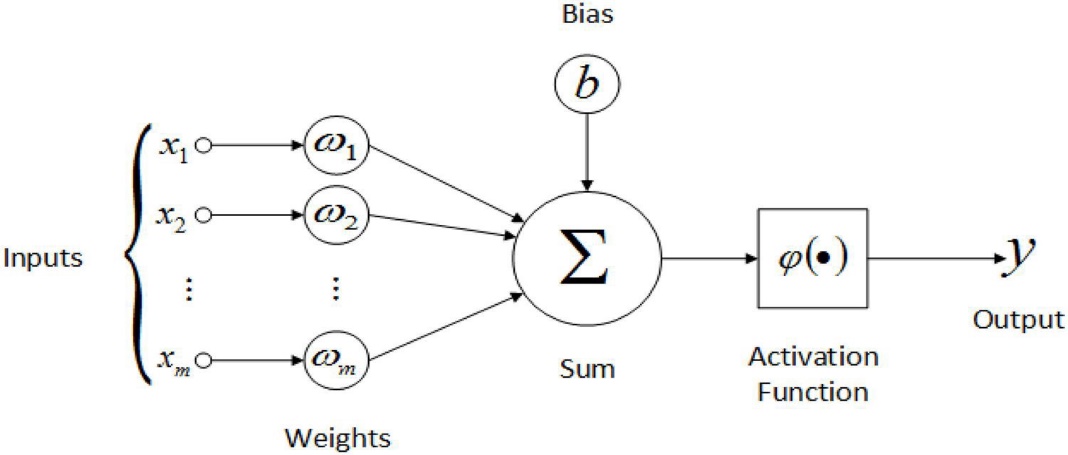
1. **CNN**
2. **Neuron / Units**
3. **Layer / Dense**
4. **Conv / Conv2D**
5. **Maxpooling / Maxpool2D**
6. **Activation Function: Relu & Sigmoid**
7. **Flatten**
8. **Softmax**
9. **crossentropy**

**10-Optimizer: Adam**

**11-Metrices**

**12-Epochs**

**13-Batch size**

**14-understand learning mean values of Weights & Biases changes Depend on inputs data**

**Problems with Achievements:**

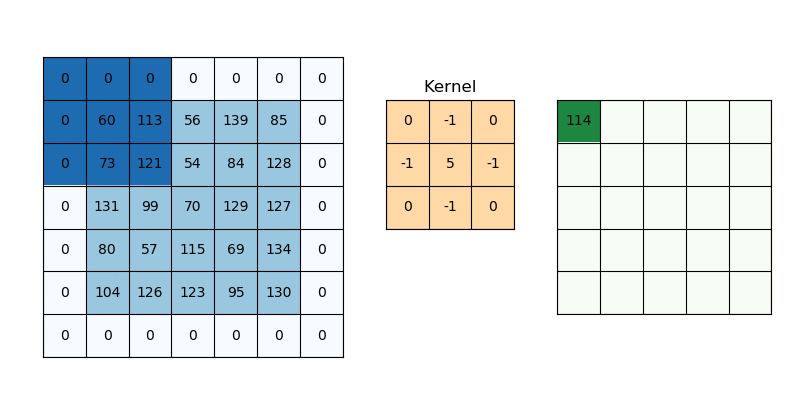
**Some of above Concepts we understood why we are using**

**It in this sequence but we are trying to find how actually its work .**

**AS example:**

**We understood we are using Conv to capture features from data .**

**Features like : pixels value & Edges & …..**

**But we don’t know How Filters actually work , and values in matrix represent filter how change?**

**Hint:**

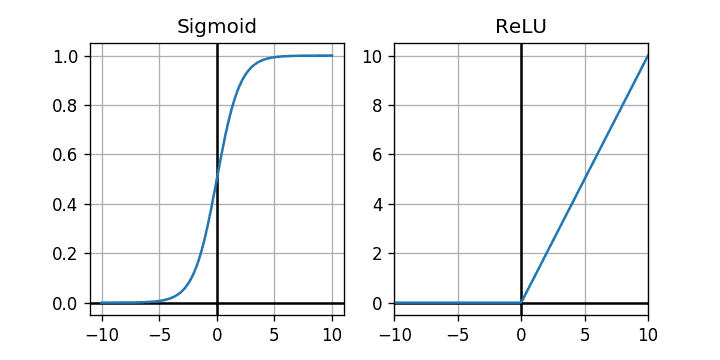
**In this case there is padding :**

**Padding uses to maintain the size of original matrix**

**Another Example:**

**we are using activation function RELU**

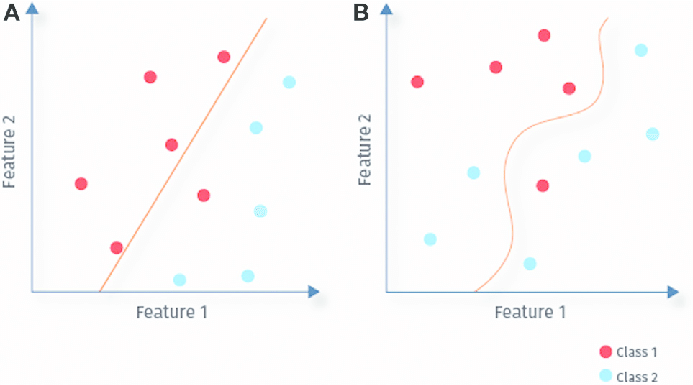
**To find Nonlinearity of inputs but we don’t know how actually its work with inputs**

****

**Hint:**

**Sigmoid: useful for finally output decision because it can separate between high and low values so it is useful for binary decisions**

**Nonlinearity of inputs:**

****

**3-Future keywords:-**

**Learning rate**

**Gradient decent**

**Back propagation**

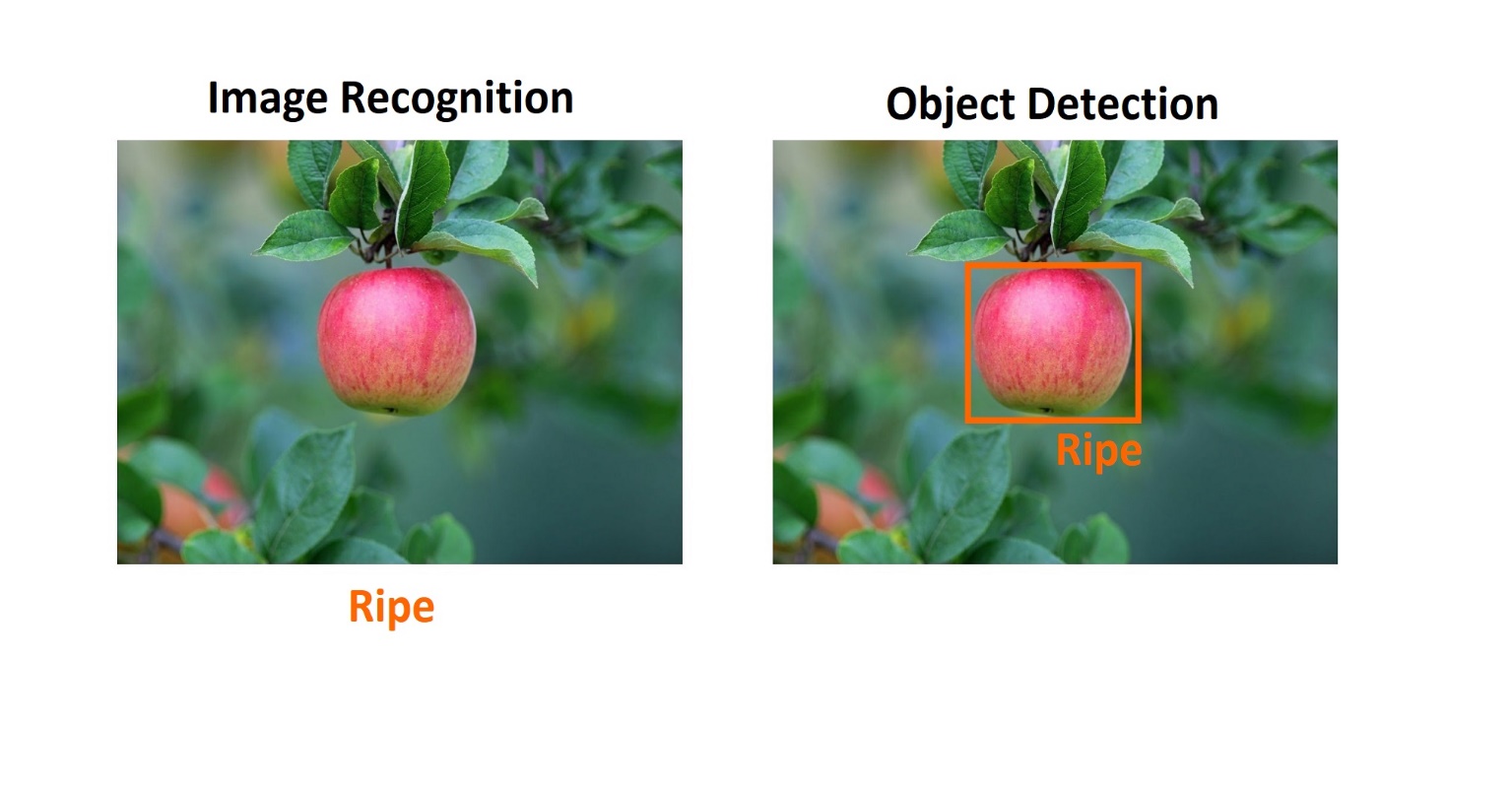
**Loss function**

**Verbose**

**Flops**

**3-Future TASKs:-**

**Until now we didn’t use detection in our project, our code can only classify the input picture and we are seeking to find best way to build our detection model.**

****

**4-CODE USING COLAB ENVIROMENT:-**

**Importing libraries and packages**

**import cv2                 # working with, mainly resizing, images**

**import numpy as np         # dealing with arrays**

**import os                  # dealing with directories**

**from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.**

**from tqdm import tqdm      # Faster iteration**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**import tensorflow as tf**

**import keras**

**TRAIN\_DIR = 'drive/MyDrive/Colab Notebooks/OurData/train'**

**TEST\_DIR = 'drive/MyDrive/Colab Notebooks/OurData/test'**

**IMG\_SIZE = 100**

**LR = 1e-3**

**MODEL\_NAME = 'ripeVSunripe-{}-{}.model'.format(LR, '2conv-basic')**

**((-$ OUR MODEL $-))**

**KerasModel = keras.models.Sequential([**

**keras.layers.Conv2D(200,kernel\_size=(3,3),activation='relu',input\_shape=(IMG\_SIZE,IMG\_SIZE,3)),**

**keras.layers.Conv2D(150,kernel\_size=(3,3),activation='relu'),**

**keras.layers.MaxPool2D(4,4),**

**keras.layers.Conv2D(120,kernel\_size=(3,3),activation='relu'),**

**keras.layers.Conv2D(80,kernel\_size=(3,3),activation='relu'),**

**keras.layers.Conv2D(50,kernel\_size=(3,3),activation='relu'),**

**keras.layers.MaxPool2D(4,4),**

**keras.layers.Flatten() ,**

**keras.layers.Dense(120,activation='relu') ,**

**keras.layers.Dense(100,activation='relu') ,**

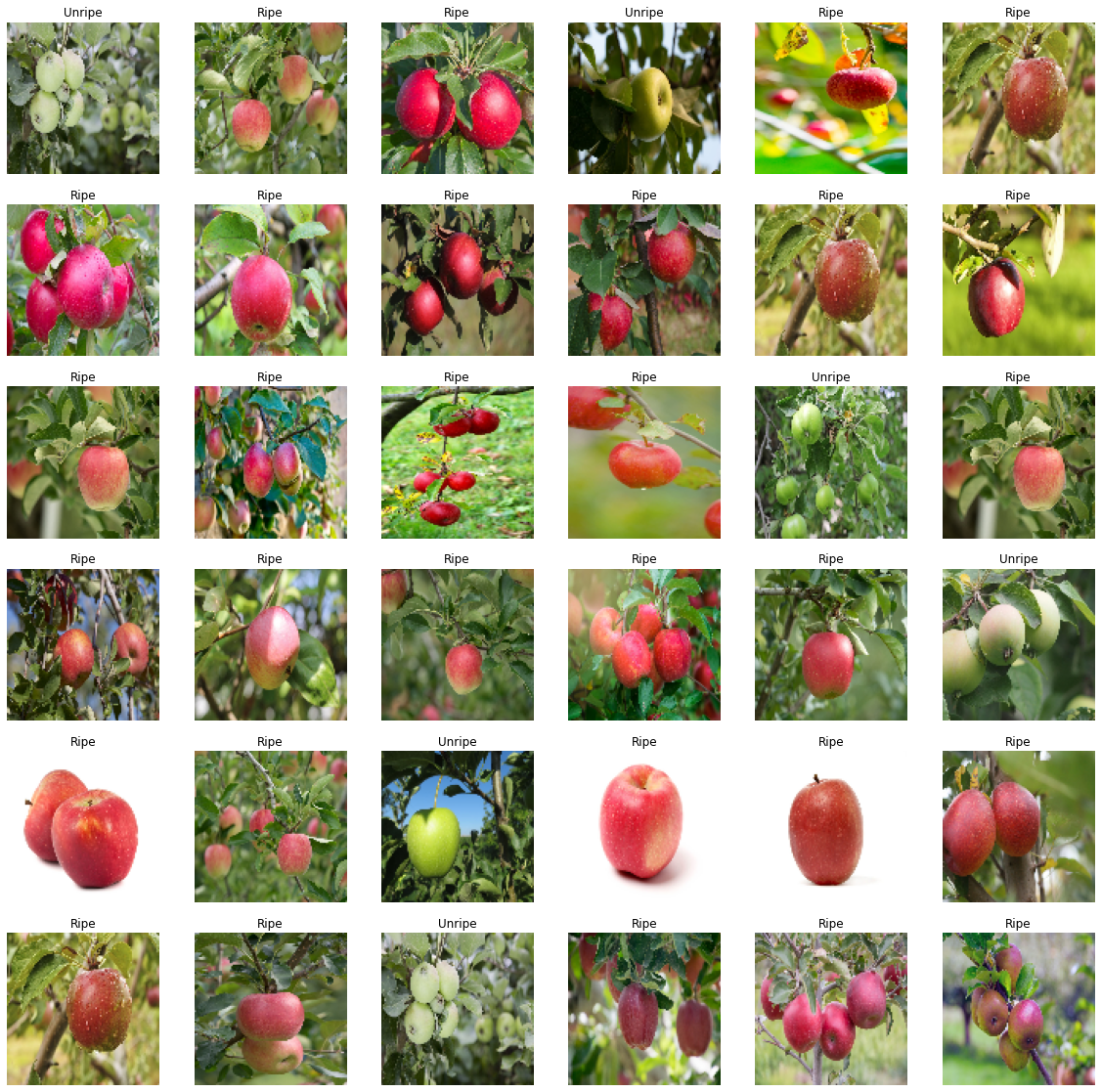
**keras.layers.Dense(50,activation='relu') ,**

**keras.layers.Dropout(rate=0.5) ,**

**keras.layers.Dense(6,activation='softmax') ,**

**])**

**KerasModel.compile(optimizer ='adam',loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])**

**OUTPUT**

**Hint:**

**MORE DETAILS WITH RUNNING CODE**

**5-Resources**

1. [**https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53**](https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53)
2. [**https://www.ibm.com/cloud/learn/convolutional-neural-networks**](https://www.ibm.com/cloud/learn/convolutional-neural-networks)
3. [**https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9**](https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9)
4. [**https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/**](https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/)
5. [**https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6**](https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)
6. [**https://www.v7labs.com/blog/neural-networks-activation-functions**](https://www.v7labs.com/blog/neural-networks-activation-functions)
7. [**https://towardsdatascience.com/overview-of-various-optimizers-in-neural-networks-17c1be2df6d5#:~:text=Optimizers%20are%20algorithms%20or%20methods,problems%20by%20minimizing%20the%20function**](https://towardsdatascience.com/overview-of-various-optimizers-in-neural-networks-17c1be2df6d5#:~:text=Optimizers%20are%20algorithms%20or%20methods,problems%20by%20minimizing%20the%20function)**.**
8. [**https://machinelearningmastery.com/cross-entropy-for-machine-learning/**](https://machinelearningmastery.com/cross-entropy-for-machine-learning/)
9. [**https://robomechjournal.springeropen.com/articles/10.1186/s40648-019-0141-2**](https://robomechjournal.springeropen.com/articles/10.1186/s40648-019-0141-2)
10. [**https://keras.io/api/optimizers/**](https://keras.io/api/optimizers/)
11. [**https://www.tensorflow.org/tutorials/images/classification**](https://www.tensorflow.org/tutorials/images/classification)
12. [**https://www.nature.com/articles/s41598-021-96103-2**](https://www.nature.com/articles/s41598-021-96103-2)
13. [**https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/**](https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/)
14. [**https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/**](https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/)

**Embedded system**