# Convolutional Neural Network

# Installing Theano

# pip install --upgrade --no-deps git+git://github.com/Theano/Theano.git

# Installing Tensorflow

# Install Tensorflow from the website: https://www.tensorflow.org/versions/r0.12/get\_started/os\_setup.html

# Installing Keras

# pip install --upgrade keras

# Part 1 - Building the CNN

#we do not classification because its already done manually

#training and test section also segregated

# Importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

# Initialising the CNN

classifier = Sequential()#secquential class by object classifier

# ---------Step 1 - Convolution

#here we dont apply directly input image into flatten image because since each node represent one pixel, so after flattening only we will get information regarding one pixel, we wont get information of other connected nodes.

#if apply convulation and flatten to input images into huge size one dimentional vector since each feature map is corresponding one specific size of images then each node contain huge information of each vector represent the specific detail of input images.

#high no does not represent unique pixel by itself, it represent standard specific feature that feature detector extract from input images through convulation operation. so we keep special structure feature of convulation process.

#so first layer is convolutional layer

classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

#32 feature detector of dimension 3\*3 matrix(row and column)

#(2-channel for balck and 3-channel for colour) mean here using scolour images of 64 and 64 formet dimension, 3 -channel

#we remove -ve pixel in order to have non linearity because classifing images is non linear problem so we need to have non linearity in model so here using rectifier function for non linearity

#(64, 64, 3) in tensor flow and in theano (3, 64, 64)

# ----------Step 2 - Pooling

# it just reducing the size of feature map

#taking maximum is called max polling in each block(size is half of original size + 1)

#here max polling reduce the size in order to reduce complaxity otherwise no of node will be more and problem will be more complex

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

# most time we use 2\*2 without lossing information

# -------------Adding a second convolutional layer

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

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

# --------------Step 3 - Flattening

classifier.add(Flatten())

#since we have taken classifier, it understand that it need to be flatten previous pooled map.

#flatten all feature image into single vector

# ---------------Step 4 - Full connectionw

#we need to create hidden layer i.e fully connected layer

classifier.add(Dense(output\_dim = 128, activation = 'relu'))

#by experiment output\_dim was found near to 100.

#output\_dim is no of node in hidden layer(it should be between no of input node and output node) by thub of rule and be experiment it should be fined. but it should be in power of 2.

classifier.add(Dense(output\_dim = 1, activation = 'sigmoid'))

#sigmoid function because binary outcome but if outcome is in two categories then we need to use softmax activation function.

# --------------Compiling the CNN

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

#"adam" is stotastic gradient algorithm optimiser

#here we have binary outcome (cat or dog), if we have three outcome then we will use "categorical\_crossentropy".

# ================Part 2 - Fitting the CNN to the images

#Fit CNN to image

#https://keras.io/ and go processing::use for image preprocessing for image augmentation to remove overfitting

#it will create many batches of images and in each batch it will apply some random transformation or random selection of images by rotating them,fliping them, shifting them or sharing them so finally we will get many more diverse images inside batches therefore lot more material to train.

#image augmentation because amount of training material augmented because our model is random model , our model never find same images across batches.so all image augmentation only remove overfitting.

# So in summary, image augmentation is technique that allow us to enrich in dataset or training set without adding more images that allow us to get good result without overfitting even with small images.

#great result in trainning set but bad result in test set due to overfitting so use image augmentation process before.

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)#use to preprocess the images of test set

training\_set = train\_datagen.flow\_from\_directory('dataset/training\_set',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

# ( batch\_size = 32 )create batches of 32 images

test\_set = test\_datagen.flow\_from\_directory('dataset/test\_set',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

classifier.fit\_generator(training\_set,

samples\_per\_epoch = 8000, #8000 image in training set

nb\_epoch = 25,

validation\_data = test\_set,

nb\_val\_samples = 2000) #no of image in test set

#RESULT::so we achieve accuracy of 84% in training set and accuracy of 75% in test set