A CENTER FOR INTER-DISCIPLINARY RESEARCH
2021-22

TITLE

"MUSIC SHEET TRANSCRIBER"

SUPERVISED BY

PARDHU NADELLA & ADITYA KAMAT

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS

Advanced AcademicCenter

(ACenter For Inter-Disciplinary Research)

This is to certify that the project titled

“MUSIC SHEET TRANSCRIBER”

is a bonafide work carried out by the following students in partial fulfilment of the requirements for Advanced Academic Center intern, submitted to the chair, AAC during the academic year 2021-22.

NAME	ROLL NO.	BRANCH
C SREEKA PARINISHTA	21241A6615	CSE(AI&ML)
AMRUTHA VARSHINI CHEBOLU	21241A6618	CSE(AI&ML)
BALA KRISHNA SURYA	21241A6609	CSE(AI&ML)

This work was not submitted or published earlier for any study

Dr/Ms./Mr.

	Dr.B.R.K.Reddy	Dr. G. Ramesh
Project Supervisor	Program Coordinator	Dean, AAC

ADVANCED ACADEMIC CENTER

ACKNOWLEDGEMENTS

We express our deep sense of gratitude to our respected Director, Gokaraju Rangaraju Institute of Engineering and Technology, for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we extend our appreciation to our respected Principal, for permitting us to carry out this project.

We are thankful to the Associate Dean, Advanced Academic Centre, for providing us an appropriate environment required for the project completion.

We are grateful to our project supervisor who spared valuable time to influence us with their novel insights.

We are indebted to all the above mentioned people without whom we would not have concluded the project.

ABSTRACT

A beautiful piece of music can move us, and very often we wish that we could play it ourselves. Many of us started our journey in music by going through the slow and painful process of learning how to read notes, even after years of training, some still find it difficult decipher various notes and their count. This was the inspiration behind our project.

The aim of our project is to build a music sheet transcriber using techniques in computer vision which automatically translate symbolic music notation into western music notation while also telling the user how many counts they need to hold the note for. We have analysed and built a model using machine learning algorithms implemented by neural networks - A Sequential CNN Keras model along with Sliding Window technique of OpenCV. We found the presence of the note using the SCNN model and then trained it to find the count of the note. We applied the sliding window Technique of OpenCV and found the pixel coordinates of a note in the particular window to determine the name of the note.

The final output gives us the name of the note and the count of the note in a readable format which allows the user to comprehend the notes in a more efficient manner.

INTRODUCTION

Computer vision is the field of computer science that focuses on replicating parts of the human vision .

It enables computers to identify and process objects in images and videos in the same way that humans do. The recent advancements and innovations in artificial intelligence has greatly improved the scope of computer vision and one of the key driving factors behind the growth of computer vision is the amount of data we generate today. It is this data that is used to train and improve computer vision.

Applications of computer vision:

Self-Driving Cars: With the use of computer vision, autonomous vehicles can understand their environment.

Facial Recognition: Facial recognition programs, which use computer vision to recognize individuals in photographs, rely heavily on this field of study.

Healthcare: Computer vision has contributed significantly to the development of various new technologies related to healthcare.

Manufacturing: Many automation solutions have been introduced with computer vision at its center to help in automating quality control, minimize safety risks, and increase production efficiency.

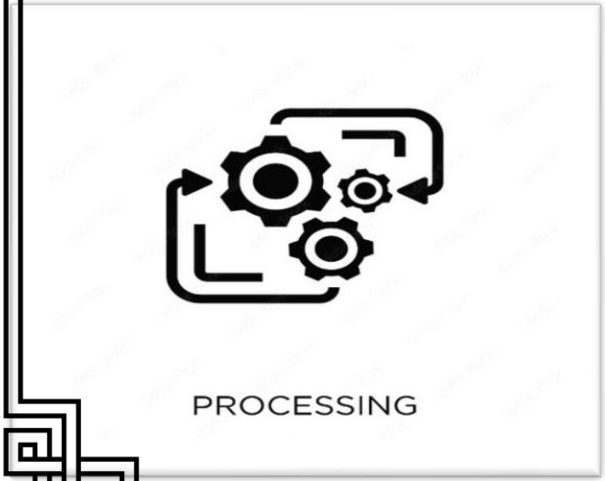
How computer vision works?

Computer vision works in three basic steps:

1. Acquiring an Image:

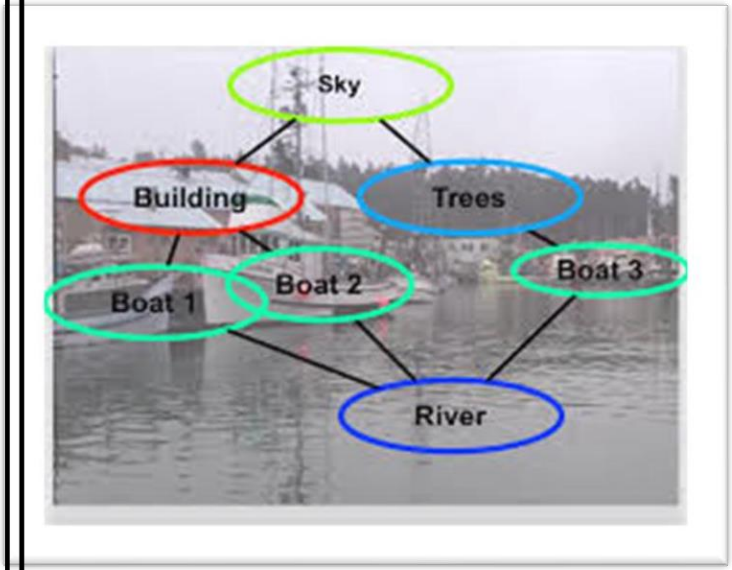
Images, even large sets, can be acquired in real-time through video, photos or 3D technology for analysis.

2.Processing the image



Deep learning models automate much of this process, but the models are often trained by first being fed thousands of labelled or pre-identified image.

3.Understanding the Image



The final step is the interpretative step, where an object is identified or classified.

What is OMR?

Optical music recognition (OMR) is a field of research that investigates how to computationally read musical notation in documents using computer vision. The goal of OMR is to teach the computer to read and interpret sheet music and produce a machine-readable version of the written music score. the transcribed copy should allow musicians to compose, play and edit music by taking picture of the music sheet. complete transcription of sheet music would also enable more efficient archival.

Selected common music notation symbols.

(a) Clefs.

(b) Staff.

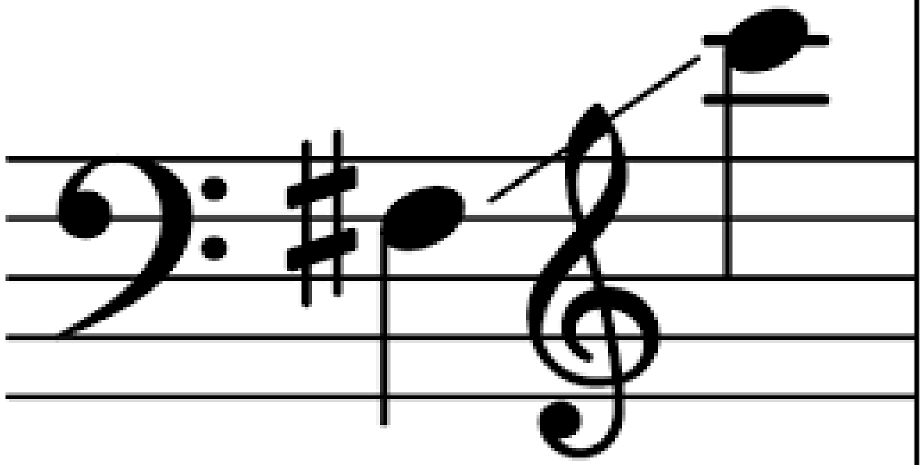
(c) Notes.

(d) Rests.

(e) Accidentals.

(f) Ornaments.

(g) Articulations.



Automatic recognition of music scores is a complex task affecting many areas of computer science. Different OMR systems use various strategies, but the most common algorithms decompose the problem into four smaller tasks .

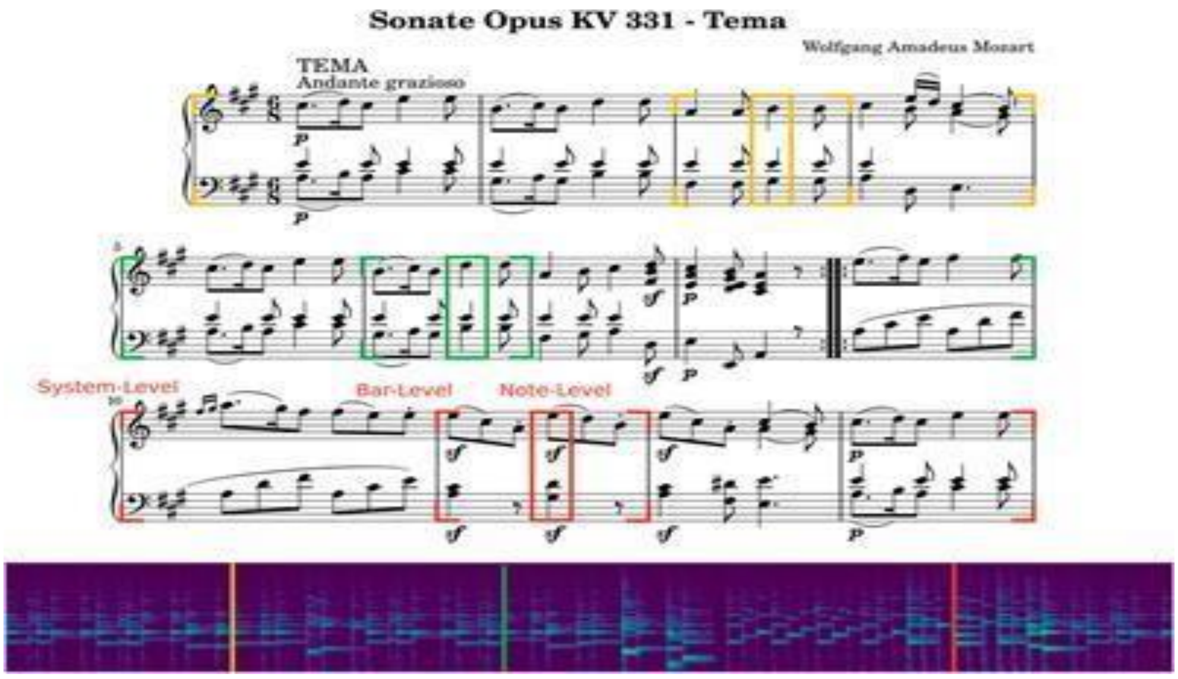
1. Image Pre-processing

2. Segmentation

3. Object Recognition

4. Semantic

Reconstruction Terminology is not always the same: the segmentation is also called primitive detection or musical object location, and the recognition phase is sometimes called musical feature classification .



Each every note is identified with the help of techniques like sliding window, as shown above. The count or time signature is determined by the colour filled inside the note and the note class is determined by the coordinates of the line it touches.

MUSIC SHEET TRANSCRIBER

Music sheet transcriber is an OMR model used to transcribe a music sheet into a ABC notation and the note count is also detected(time signature). counting is a system of regularly occurring sounds that serve to assist with the performance or audition of music by allowing the easy identification of the beats. there are 12 notes in western music!

How is this transcriber built?

Using computer vision with deep learning technique like sequential computational neural network and split window, the music sheet which is given as input is decoded, various notes are identified along with their count.

A sequential computational neural network (SCNN) is created using a high level Application Programming Interface (API), keras. using SCNN a training model (mymodel in our code) is created, to which 80 percent of the data is provided as input. the data is then trained and the accuracy is tested by giving the remaining 20% as the test inputs.

The musical sheet is then taken as input using imread, which is in cv2 library. The sheet is further split into multiple parts using sliding window technique which finds the position of note on the sheet. These notes are sent to the model recursively, the count of each note is determined by mymodel. The class and name of the note are identified by locating the pixel coordinates of each note.

Project workflow

Data set including musical notes sorted into their classes is taken from

:https://www.kaggle.com/datasets/kishanj/music-notes-datasets .

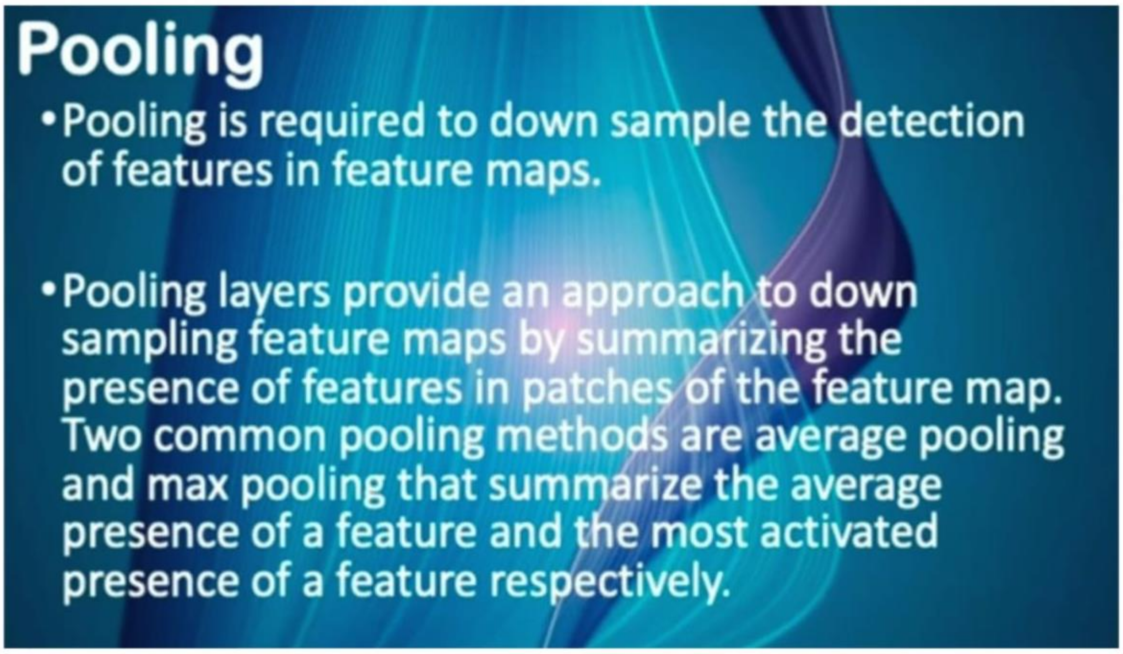
Reducing image dimensionality :

Input image is reduced by using different building blocks like kernel,pooling,flatten etc.and given as input to the connected network.

Why reducing image dimensionality?

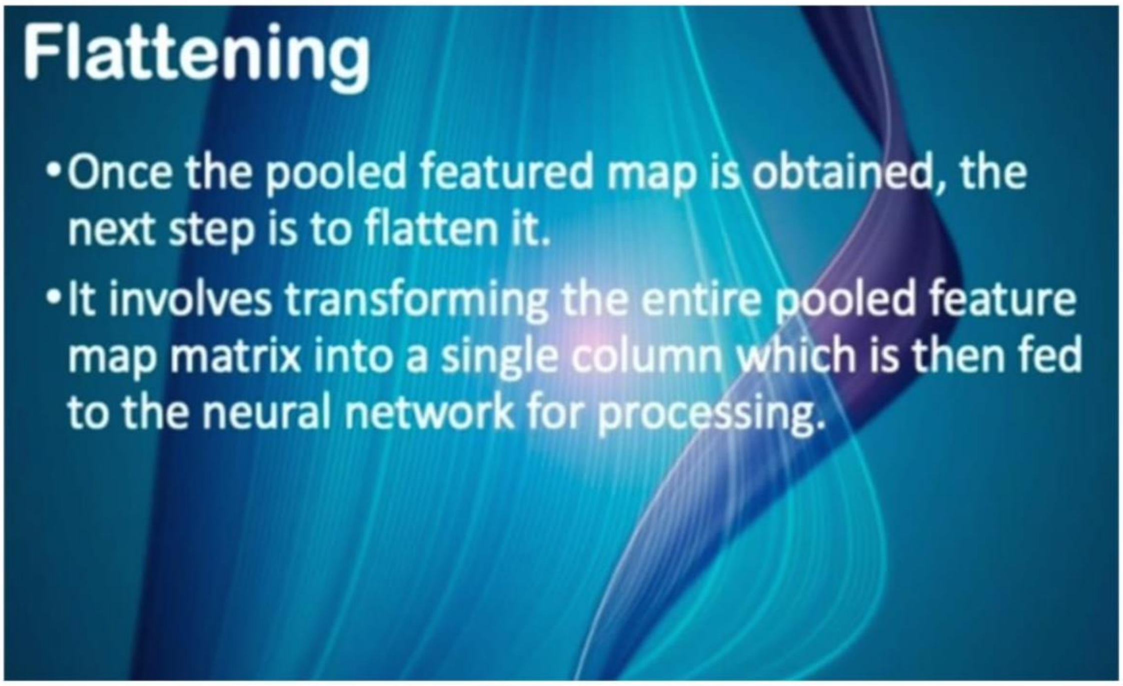
Since neural networks receive inputs of the same size, all images need to be resized to a fixed size before inputting them to the CNN . The larger the fixed size, the less shrinking required. Less shrinking means less deformation of features and patterns inside the image.

Different techniques available to reduce image are:



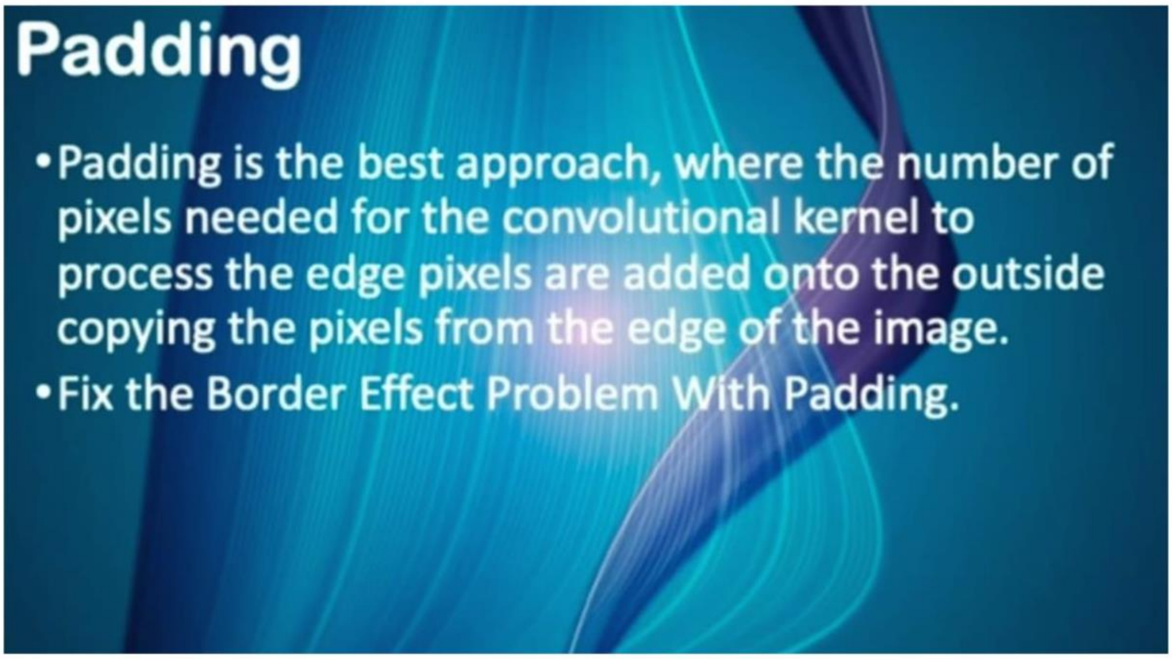
Pooling

- Pooling is required to down sample the detection of features in feature maps.
- Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively.



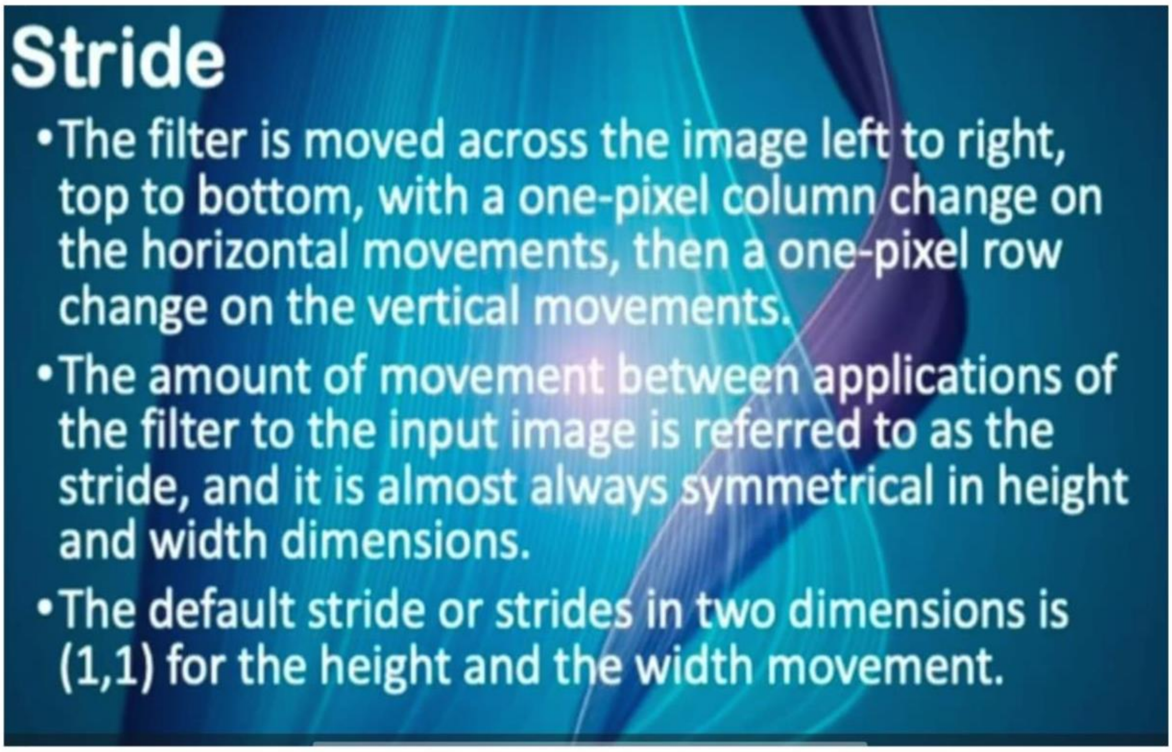
Flattening

- Once the pooled featured map is obtained, the next step is to flatten it.
- It involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.



Padding

- Padding is the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image.
- Fix the Border Effect Problem With Padding.



Stride

- The filter is moved across the image left to right, top to bottom, with a one-pixel column change on the horizontal movements, then a one-pixel row change on the vertical movements.
- The amount of movement between applications of the filter to the input image is referred to as the stride, and it is almost always symmetrical in height and width dimensions.
- The default stride or strides in two dimensions is (1,1) for the height and the width movement.

Data augmentation:

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model.

keras is also used to Augment the Dataset using function Called ImageDataGenerator

(from keras.preprocessing.image import ImageDataGenerator)

Many optimizer subclasses, such as Adam and Adagrad allocate and manage additional variables associated with the variables to train. These are called Slots.

(from tensorflow.keras.optimizers import Adam)

All the augmented images are reduced to same dimensions (32,32,1) and are given as **input to mymodel.**

Working of CNN:

1.We are Using Sequential Convolutional Neural Network model to train the model

2.We use a high level application programming interface ,keras to implement sequential CNN. keras serves as a front end to the model. While TensorFlow acts as a back end.

What are APIs…?

An Application Programming Interface can be thought of as the way in which programs communicate with each other, accessing data and functionality despite being separate systems. This kind of integration allows for the easy extension of application functionality such as, for example, adding multiple payment methods to a site

High-level APIs are fairly abstracted, meaning that they are more generic and therefore limited in functionality. Low-level APIs are much more detailed and specific due to a low level of abstraction. Low-level APIs allow for finer control over application functions.

3.KERAS

keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library

Various functions imported from keras include:

**** (from keras.models import Sequential)****

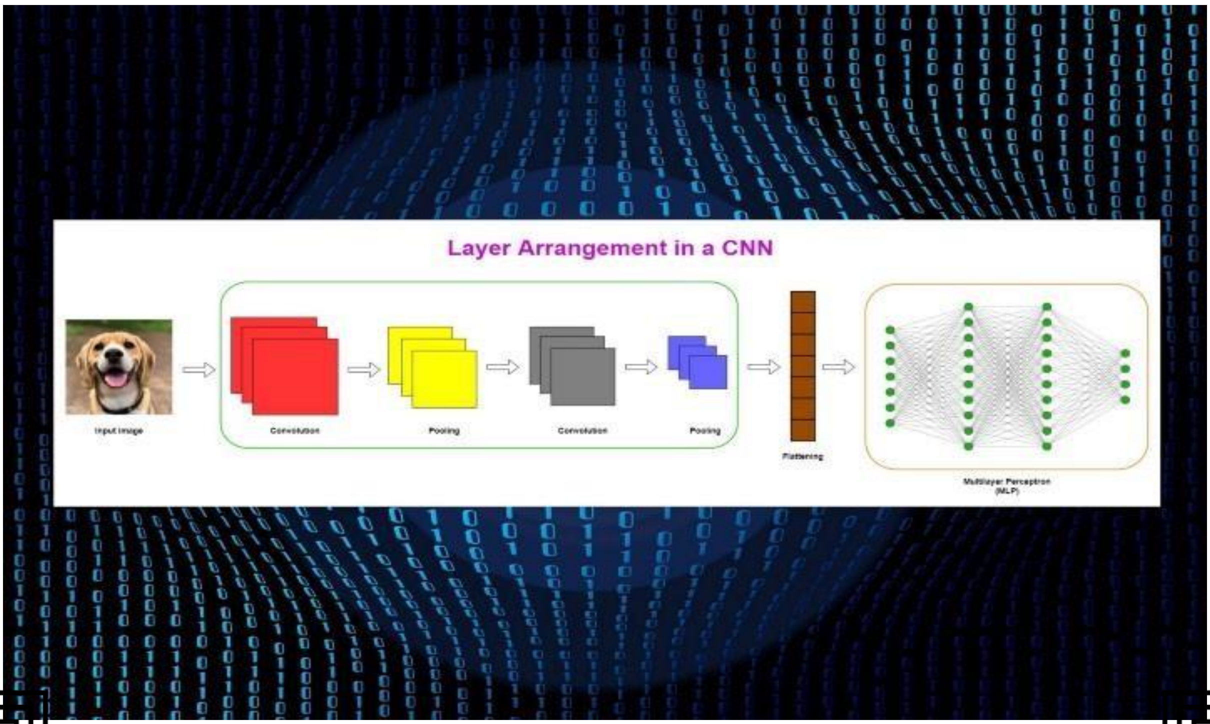
****(from keras.layers import Dense, Dropout, Activation, Flatten, Softmax (2 Pooling layers)****

4. TENSORFLOW

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow. Both provide high-level APIs used for easily building and training models.

****(from tensorflow.keras.layers import Conv2D , MaxPooling2D)****



Inside Convolution Layer.:

Keras Conv2D is a 2D Convolution Layer, this layer creates a convolution kernel that is wound with layers input which helps produce a tensor of outputs.

Kernel: In image processing kernel is a convolution matrix or masks which can be used for blurring, sharpening, embossing, edge detection, and more by doing a convolution between a kernel and an image.

The Keras Conv2D class constructor has the following arguments:

****(Conv2D(noOfFilters,sizeOfFilter1,input_shape=(imageDimensions[0],imageDimensions[1],1), activation='relu')))****

In Keras, a Sequential model can be built by using the Sequential()class. Here, we sequentially add layers to the model using the add()method. According to the Keras documentation,

Layer arrangement in cnn:

Convolutional-Pooling pair → Convolutional-Pooling pair → A flattened layer → Multiple dense layers

Few other libraries we have used:

MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots

(import matplotlib.pyplot as plt)

SKLEARN

****(from sklearn.model_selection import train_test_split)****
Split arrays or matrices into random train and test subsets.

PICKLE

Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling.

****(import pickle)****

****(from keras.utils.np_utils import to_categorical)****

Using the method to_categorical(), a numpy array (or) a vector which has integers that represent different categories, can be converted into a Numpy array Which is further Used in the program.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

(import numpy as np)

OS

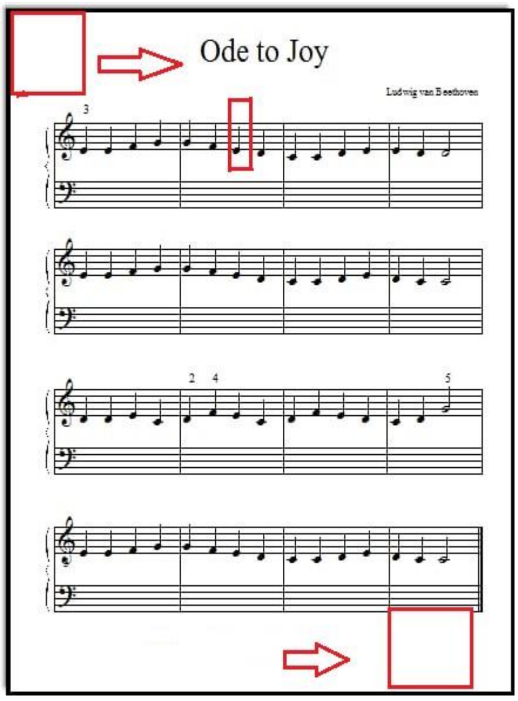
The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules.

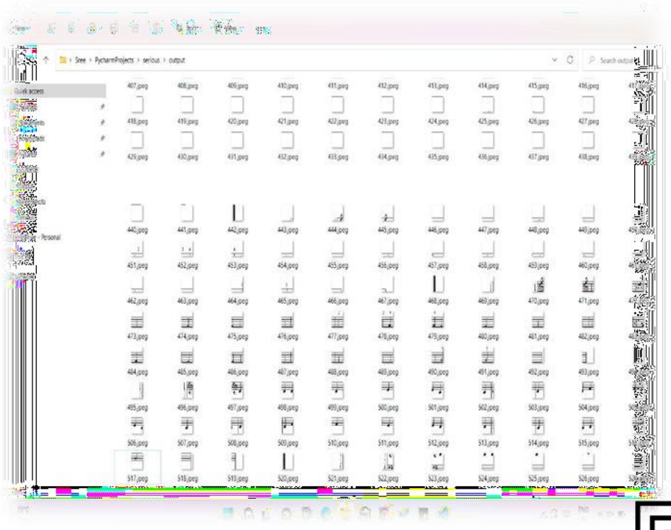
****(import os)****

Sliding window

Sliding window Technique is a computational technique which aims to reduce the use of nested loop and replace it with a single loop, thereby reducing the time complexity. the sliding window technique is useful when you need to keep track of a **contiguous** sequence of elements, such as summing up the values in a subarray. By using the sliding window technique, we are able to solve the problem above with O(n) time complexity, eliminating the need for duplicate iterations.

In our model the window slides the music sheet that is given as input. the duplicates are removed. Overlapping is avoided by eliminating the similar number in the array.

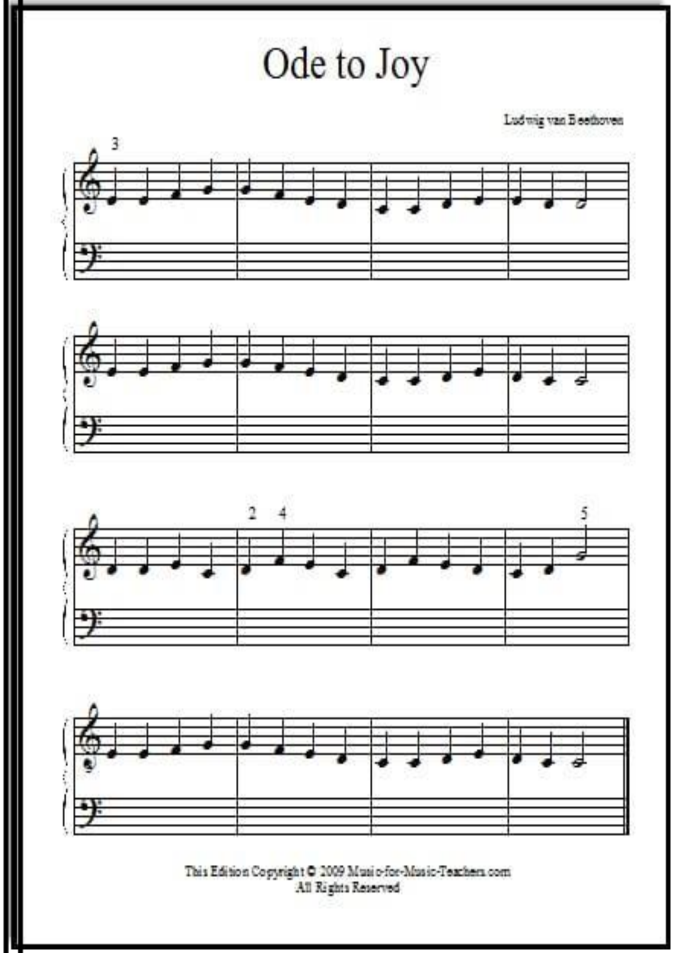


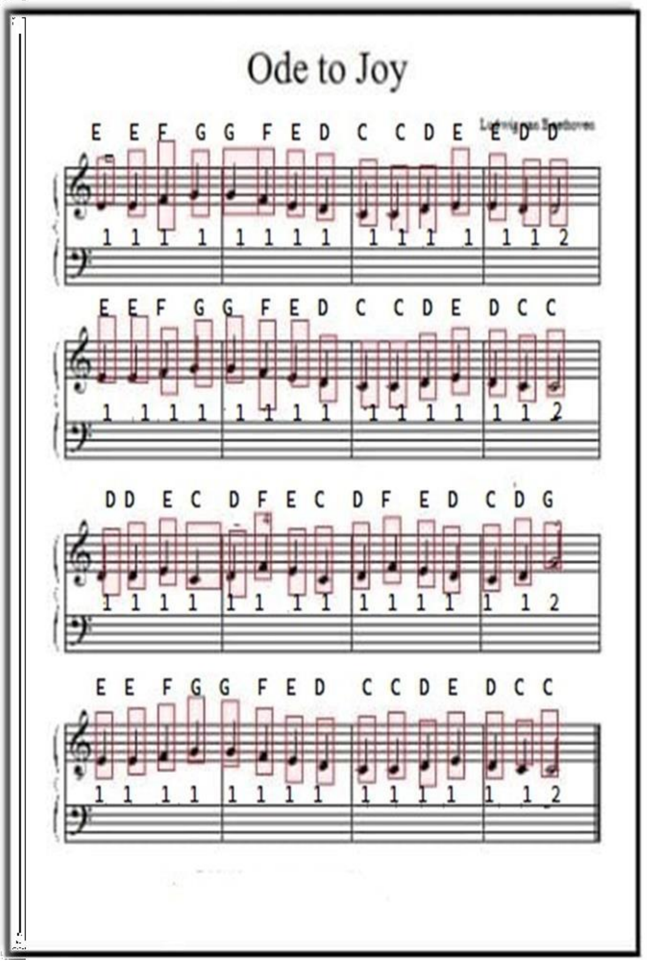


Output:

The model is tested and the precision of the model is give as output .

The note is compared with every class and the probability of the note matching to the compared class is calculated and an array is made. The class with highest probability is consider as the accurate class for the note.time signature is also identified.





CODE

```
from tensorflow.keras.optimizers import Adam
import numpy as np
import cv2
import os
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.layers import Dense, Dropout, Activation, Flatten,  Softmax
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from tensorflow.keras.layers import Conv2D ,  MaxPooling2D
import pickle
from skimage.morphology import binary_opening
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from skimage.filters import threshold_otsu, gaussian, median
#####################################
path= 'Notes'
testRatio=0.2
valRatio=0.2
imageDimensions=(32, 32, 3)

batchSizeVal= 50
epochsVal= 30
stepsPerEpoch = 20

#######################################
images=[]
count=0
classNo =[]
List= os.listdir(path)
print("Total no of clases detected",len(List))
noOfClasses =len(List)
print("Importing classes....")
for x in range (0,noOfClasses):
    PicList =os.listdir(path+"/"+str((x)))

    for y in PicList:
        curImg =cv2.imread(path+"/"+str(x)+"/"+y)
        curImg =cv2.resize(curImg,(32,32))
        images.append(curImg)
        classNo.append(count)
    print(count,end= " ")
    count +=1
print(" ")
images=np.array(images)
```

```
classNo = np.array(classNo)
print(images.shape)
## Spliting the data
x_train, x_test,y_train,y_test =
train_test_split(images,classNo,test_size=0.2)
x_train,x_validation,y_train,y_validation=train_test_split(x_train,y_train,tes
t_size=valRatio)
print(x_train.shape)
print(x_test.shape)
print(x_validation.shape)


numOfSamples= [ ]
for x in range(0,noOfClasses):
    ##print(len(np.where(y_train==0)[0]))
    numOfSamples.append(len(np.where(y_train==x)[0]))
print(numOfSamples)

plt.figure(figsize=(10,5))
plt.bar(range(0,noOfClasses),numOfSamples)
plt.title("No of Images for each Class")
plt.xlabel("Class ID")
plt.ylabel("Number of Images")
plt.show()

def preProcessing(img):
        img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        img = cv2.equalizeHist(img)
        img = img/255
        return img

#img = preProcessing(x_train[30])
#img=cv2.resize(img,(300,300))
#cv2.imshow("PreProcessed",img)
#cv2.waitKey(0)

x_train=np.array(list(map(preProcessing,x_train)))
x_test=np.array(list(map(preProcessing,x_test)))
x_validation=np.array(list(map(preProcessing,x_validation)))

x_train =
x_train.reshape(x_train.shape[0],x_train.shape[1],x_train.shape[2],1)
x_test = x_test.reshape(x_test.shape[0],x_test.shape[1],x_test.shape[2],1)
x_validation =
x_validation.reshape(x_validation.shape[0],x_validation.shape[1],x_validation.
shape[2],1)

dataGen = ImageDataGenerator(width_shift_range=0.1,
                             height_shift_range=0.1,
zoom_range=0.2,
                                        shear_range=0.1,
                                        rotation_range=10)

dataGen.fit(x_train)

y_train=to_categorical(y_train,noOfClasses)
y_test=to_categorical(y_test,noOfClasses)
y_validation=to_categorical(y_validation,noOfClasses)

def myModel():
     noOfFilters= 60
     sizeOfFilter1=(5,5)
     sizeOfFilter2= (3,3)
     sizeOfPool= (2,2)
     noOfNode = 500

     model = Sequential()
     model.add((Conv2D(noOfFilters,sizeOfFilter1,input_shape=(imageDimensions[0
],imageDimensions[1],1), activation='relu')))
     model.add((Conv2D(noOfFilters, sizeOfFilter1, activation='relu')))
     model.add(MaxPooling2D(pool_size=sizeOfPool))
     model.add((Conv2D(noOfFilters//2, sizeOfFilter2, activation='relu')))
     model.add((Conv2D(noOfFilters//2, sizeOfFilter2, activation='relu')))
     model.add(MaxPooling2D(pool_size=sizeOfPool))
     model.add(Dropout(0.5))

     model.add(Flatten())
     model.add(Dense(noOfNode,activation='relu'))
     model.add(Dropout(0.5))
     model.add(Dense(noOfClasses,activation='softmax'))
     model.compile(Adam(lr=0.001),loss='categorical_crossentropy',metrics=['acc
uracy'])
     return  model

model = myModel()
print(model.summary())

history =
model.fit(dataGen.flow(x_train,y_train,batch_size=batchSizeVal),steps_per_epoc
h=stepsPerEpoch,
                           epochs=epochsVal,validation_data=(x_validation,y_validatio
n),
                           shuffle=1)

"""
new_img = cv2.imread()
pred = model.predict(new_img)
```

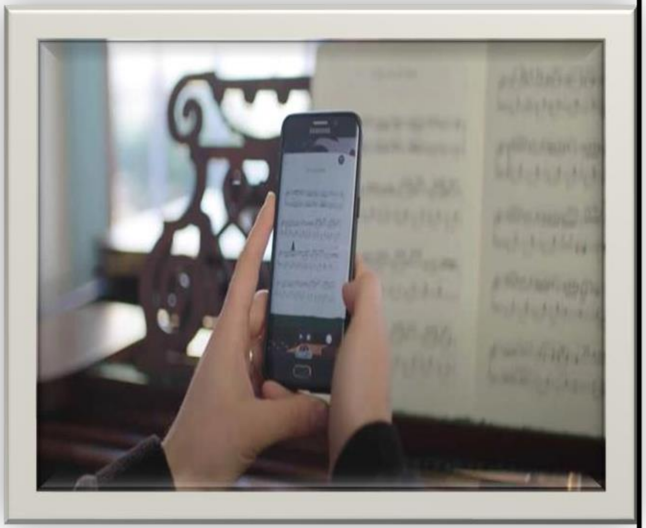
```
"""

#print(history)
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('Loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training','validation'])
plt.title('Accuracy')
plt.xlabel('epoch')
plt.show()

score =model.evaluate(x_test,y_test,verbose=0)
print('Test Score =',score[0])
print('Test Accuracy =', score[1])

pickle_out =open("model_trained.pkl","wb")
pickle.dump(model,pickle_out)
pickle_out.close()
#############################################
width = 640
height = 480
threshold =0.85
#############################################
cap=cv2.VideoCapture(0)
cap.set(3,width)
cap.set(4,height)

#pickle_in =open("model_trained.pkl","rb+")
#model=pickle.loads(pickle_in)

def preProcessing(img):
    img=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img=cv2.equalizeHist(img)
    img=img/ 255
    return img

# while True:success, imgOriginal = cap.read()
```

```
imgOriginal =
cv2.imread("C:\\Users\\Sree\\PycharmProjects\\serious\\Notes\\1\\h1.jpg")
img = np.asarray(imgOriginal)
img = cv2.resize(img,(32,32))
img = preProcessing(img)
cv2.imshow("Processed Image",img)
img = img.reshape(1,32,32,1)

#Predict
# classIndex = int(model.predict_classes(img))
#print(classIndex)
predictions = model.predict(img)[0]
print(predictions)
probVal= np.amax(predictions)
# print(classIndex,probVal)

#if  probVal> threshold:
  #      cv2.putText(imgOriginal,str(classIndex)
+"   "+str(probVal),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),1)
cv2.imshow("Original Image",imgOriginal)
cv2.waitKey(10)
#if  cv2.waitKey(1) &0xFF == ord('q'):
#    break

cv2. destroyAllWindows()
cap.release()
from tensorflow.keras.optimizers import Adam
import numpy as np
import cv2
import os
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.layers import Dense, Dropout, Activation, Flatten, Softmax
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from tensorflow.keras.layers import Conv2D ,  MaxPooling2D
import pickle
from skimage.morphology import binary_opening
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from skimage.filters import threshold_otsu, gaussian, median
row_percentage = 0.3

def  remove_staff_lines_2(thickness,  img_with_staff):
 img = img_with_staff.copy()
     projected = []
     rows, cols =img.shape
```

```
    for i  in range(rows):
        proj_sum = 0
        for j  in range(cols):
            proj_sum += img[i][j] == 1
        projected.append([1]*proj_sum + [0]*(cols-proj_sum))
        if(proj_sum <= row_percentage*cols):
            img[i, :] = 1
    closed = binary_opening(img, np.ones((3*thickness, 1)))
    return closed

def otsu(img):
    '''
    Otsu with gaussian
    img: gray image
    return: binary image, pixel values 0:1
    '''
    blur = gaussian(img)
    otsu_bin = 255*(blur > threshold_otsu(blur))
    return (otsu_bin/255).astype(np.int32)

image = cv2.imread("MUSI.jpeg")  #
img = cv2.medianBlur(img,5)
cv2.imshow('source_image', image)
cv2.waitKey(10)

ret,thresh_img = cv2.threshold(image, 0, 255,
cv2.THRESH_BINARY_INV|cv2.THRESH_OTSU)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (4,8))
morph_img = cv2.morphologyEx(thresh_img, cv2.MORPH_CLOSE, kernel)

cv2.imshow("Window", morph_img)
cv2.waitKey(0)
# def sliding_window(image, stepSize):
#    imgarr=[]
#    #xarr =[]
#    #yarr = []

#    Eline[]=[[115,][212][320][426]]
#    Gline[]=[[109][206][314][420]]
#    #Fif  condition stating that it  touches both.
#    Cline[]=[[120][216][325][432]]
#    Dline[]=[[115 &&119][212&&216][320&&324][426&&430]]
#    Fline[]=[[109 &&115][206&&212][320&&314][426&&420]]
#    #Aline[]=[[]]
#    #Bline[]=[[]]
#    ##################################################
# Cl2[]=[[115&&119],[216],[314],[430]]
# cl1[]=[[115],[115 &&119],[109],[109 &&
115],[212][320][426][206][314][420][120][216][325][432][212&&216][320&&324][42
6&&430][206&&212][320&&314][426&&420]]

image = cv2.imread("MUSI.jpeg")
image = cv2.imread("without.jpg")
image = cv2.imread("testing.png")



class0=[]
class1=[]
class2=[]
class3=[]
class4=[]


stepSize = 32
threshold = 0.85
new_arr = []
ci = []
pv = []

# slide a window across the image
cc = 0
for y in range(0, image.shape[0] - stepSize - 1, int(stepSize/2)):
    for x in range(0, image.shape[1] - stepSize - 1, int(stepSize/2)):  #
        yield the current window img=image[y:y+stepSize,x:x+stepSize]
        cv2.imwrite("output/"+str(cc)+".jpeg", img)
        cc += 1
        img = np.asarray(image)
        img = cv2.resize(img,(32,32))
        img = preProcessing(img)
        # cv2.imshow("Processed Image",img)
        img = img.reshape(1,32,32,1)

        predictions = model.predict(img)[0]
        # print(predictions)
        ind_count = 0
        probVal= np.amax(predictions)
        for tempp in predictions:
            if tempp == probVal:
                classIndex = ind_count
            ind_count+=1

        # Prediction of the class here
        # print(classIndex,probVal)
        ci.append(classIndex)
        pv.append(probVal)
        if  probVal> threshold:
            new_arr.append(classIndex)
            cv2. put Text ( image, str( classIndex) , ( x,
y),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),1)
cv2.imshow("output",  image)
cv2.waitKey(10)
cv2.destroyAllWindows()
print(new_arr)
# Append the coords in  xarr  and  yarr


#  for
#            if  probVal in  predictions[0]
#            class0.append(x,y)
#            if  probVal in  predictions[1]
#            class1.append(x,y)
#            if  probVal in  predictions[2]
#            class2.append(x,y)
#            if  probVal in  predictions[3]
#            class3.append(x,y)
#            if  probVal in  predictions[4]
#            class4.append(x,y)
#        # Return the xarr  and yarr
#            imgarr.append(img)
#    return  imgarr

print(ci)
print(pv)
import cv2
output=cv2.imread("OUTPUTF.jpg")
cv2.imshow("output",output)
cv2.waitKey(0)
testimage=cv2.imread("download.png")
imarr=sliding_window(testimage,stepSize=32)
count=0
for i  in  imarr:
    img = np.asarray(i)
    img = cv2.resize(img,(32,32))
    img = preProcessing(img)
    cv2.imshow("Processed Image",img)
    img = img.reshape(1,32,32,1)

    #Predict
    # classIndex = int(model.predict_classes(img))
    #print(classIndex)
    predictions = model.predict(img)[0]
    print(predictions)
    probVal= np.amax(predictions)#insert maxfunction
    count+=1
    #identify index of max
##################################################################################################################
########################
line_y_coords =
[[94,99,105,109,115],[192,196,202,206,212],[299,304,309,315,320],[406,410,416,
420,426]]
for a in  range(0,
    if  note in line_y_coords
    if  note in Cline
        cv2.puttext("C ")
    elif  note in Dline
        cv2.puttext("D ")
    elif  note in Eline
        cv2.puttext("E ")
    elif  note in Fline
        cv2.puttext("F ")
    print(max)
```

Future developments

This project can be liked with web or app to make a precise working model. we can also link our project with IOT and take the input with camera, make the system play the musical sheet without any human effort. We can further include notations from different styles and instruments which will be used to decipher the count of the note.



Reference

- *https://keras.io/guides/sequential_model/*
- https://levelup.gitconnected.com/understanding-the-sliding-window-technique-in-algorithms-c6c3bf8226da
- https://youtu.be/Y1qxI-Df4Lk
- https://towardsdatascience.com/coding-a-convolutional-neural-network-cnn-using-keras-sequential-api-ec5211126875
- https://towardsdatascience.com/coding-a-convolutional-neural-network-cnn-using-keras-sequential-api-ec5211126875
- https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article#:~:text=TensorFlow%20is%20an%20open%2Dsourced,because%20it%27s%20built%2Din%20Python
- https://www.google.com/amp/s/www.geeksforgeeks.org/keras-conv2d-class/amp