iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** IX **Month of publication:** September 2021

DOI: https://doi.org/10.22214/ijraset.2021.37920

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue IX Sep 2021- Available at www.ijraset.com

Identification of Microcontroller Based Objects Using Image Classification in Python

Zenith Nandy
Electronics and Communication Engineering Department, Narula Institute of Technology

Abstract: ***In this paper, I built an AI model using deep learning, which identifies whether a given image is of an Arduino, a Beaglebone Black or a Jetson Nano. The identification of the object is based on prediction. The model is trained using 300 to 350 datasets of each category and is tested multiple times using different images at different angles, background colour and size. After multiple testing, the model is found to have 95 percent accuracy. Model used is Sequential and uses Convolution Neural Network (CNN) as its architecture. The activation function of each layer is RELU and for the output layer is Softmax. The output is a prediction and hence it is of probability type. This is a type of an application based project. The entire scripting is done using Python 3 programming language.***
Keywords: image classification, microcontroller boards, python, AI, deep learning, neural network

I. INTRODUCTION

In today's world, automation plays a huge role in making processing and time management easier than before where everything was done with manual intervention. This paper puts up a small example of what is considered today as one of the main subjects to implement automation, the Artificial Intelligence. Here a model is built which is based on AI and deep learning which identifies images of Arduino, Jetson Nano and Beaglebone Black. This will help to group images of these three kinds from a cluster of images having these three objects only in no time. The identification of the particular category is based on prediction which comes upon training the model with proper amount of training datasets. In simple words of understanding, it is like teaching someone to identify three different types of objects and then asked to identify each of them by themselves upon seeing images which were not provided while training.

A. System Requirements and Platform Used

A minimum of 6GB RAM with any Operating System that supports Python is desirable. However, Google Colab is a cloud based platform that supports this entire process smoothly. The time taken by the entire process is dependent on the RAM and GPU of that particular system. Here Google Colab is free and provides descent amount of requirements. The Pro version of Google Colab can be used for even better performance. The programming language used is Python 3.

II. PROCESS

The processing is comprised of various steps. Root processes can be termed here as: training dataset collection, pre-processing the data, training the model, testing the model and drawing conclusions.

A. Training Dataset Collection

Around 350 images of Arduino, Beaglebone Black and Jetson Nano have been provided each. These images are collected by manual download and placed in three different folders for category classification. The training dataset carries out key role for the model accuracy and prediction. It is always advised to have a good amount of dataset for a desirable performance.

B. Pre-processing the Data

The raw data has to be processed in a way such that it is acceptable by the model. Hence the dimension of every image is changed to the dimension of 400 by 400 pixels. After the dimension adjustment, every image is converted to array whose values are the RGB values of every pixel. The array formed of each category is hence of 4 dimensions: total number of images, height, width and number of colour channels.

C. Training the Model

After the sufficient collection and pre-processing, the model is trained using the training image dataset as the feeder. The batch size is 20 and the validation split is 10% or 0.1 of the training dataset. The total number of epochs it has to undergo is 15 which can also be extended or reduced to avoid overfitting or underfitting of the model.

The accuracy of last epoch gives us the accuracy of the model. The loss function used here is 'sparse categorical crossentropy' and the optimizer is 'Adam', which is the go to optimizer for many cases. The learning rate is set to 0.001.

D. Testing the Model

After the training is done, the working efficiency is tested by observing the predictions on many random images belonging to any of the three kinds. After multiple testing, a conclusion is reached that whether the particular model is acceptable. In general, models with greater than 80% accuracy can be recognised as acceptable. If the results aren't satisfactory, the model is trained again and tried again.

E. Flowchart of the Process

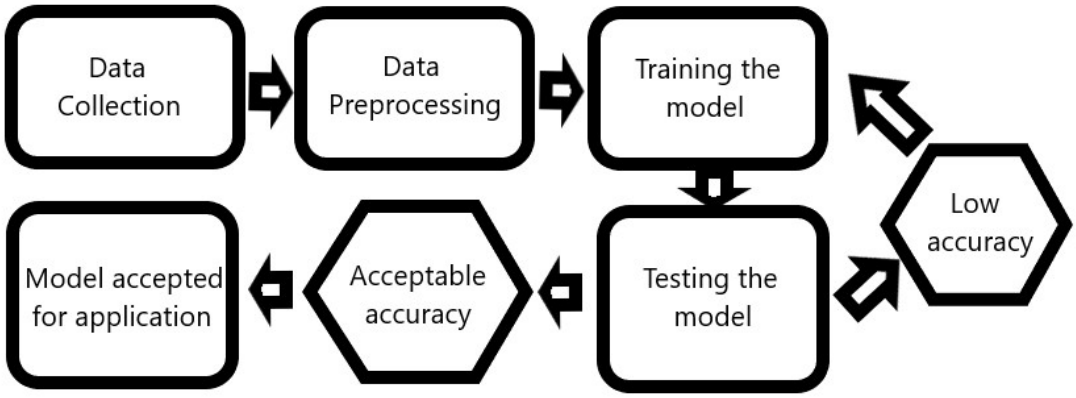


Fig. 1 Flowchart Demonstrating the Entire Process

III.MODEL SPECIFICATIONS AND ANALYSIS

The model used here is a deep learning model having a CNN (Convolution Neural Network). It is based on keras interface, which comes under tensorflow module. The model architecture goes as:

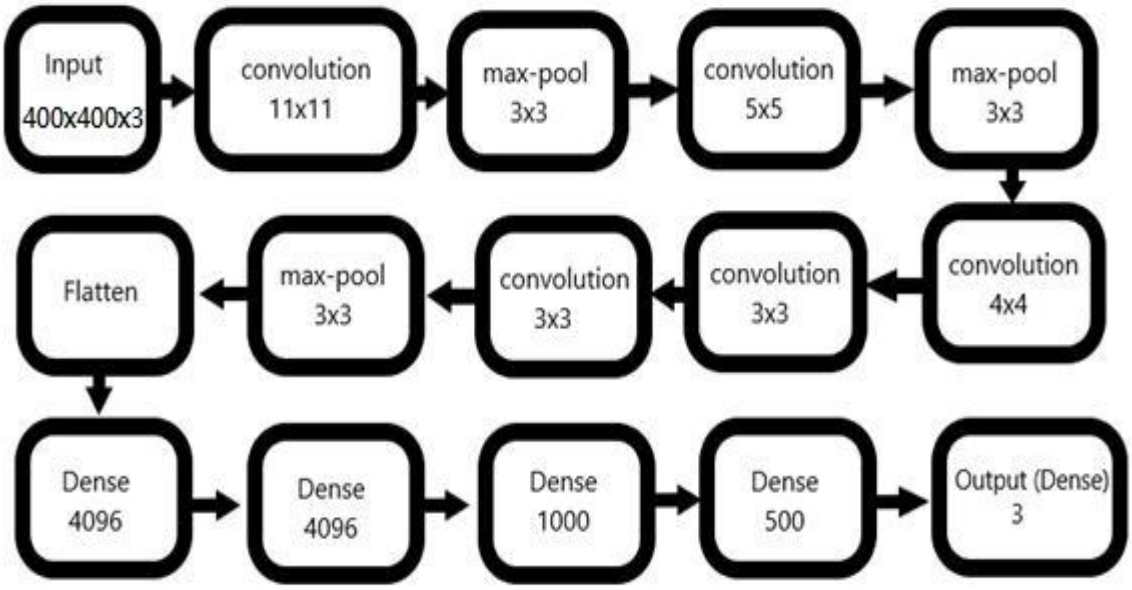


Fig. 2 Model Architecture

This architecture is modified from the AlexNet (2012) architecture of CNN for image classification. Below is the python code for the model with the above structure. This is a sequential model.

```
model=Sequential()
model.add(Conv2D(64,(11,11),activation='relu',input_shape=x.shape[1:]))
model.add(MaxPooling2D(3,3))
model.add(Conv2D(64,(5,5),activation='relu'))
model.add(MaxPooling2D(3,3))
model.add(Conv2D(64,(4,4),activation='relu'))
model.add(Conv2D(30,(3,3),activation='relu'))
model.add(Conv2D(30,(3,3),activation='relu'))
model.add(MaxPooling2D(3,3))
model.add(Flatten())
model.add(Dense(4096,activation='relu'))
model.add(Dense(4096,activation='relu'))
model.add(Dense(1000,activation='relu'))
model.add(Dense(500,activation='relu'))
model.add(Dense(3,activation='softmax'))
```

Fig. 3 Python Code for the Model Architecture

A. Model Accuracy and Training Efficiency

The model is made to undergo 15 epochs to train itself. Number of epochs, batch size, learning rate and sometimes the model architecture needs to be changed to avoid overfitting or underfitting of the model. An overfit model does not gives enough accuracy in test predictions inspite of a healthy training accuracy. An underfit model doesn't give either accuracy as desired. In both cases, it needs to train itself again after making some changes to the model specifications. In this case, we find that the training accuracy is 0.9538 or 95.38%.

```
Epoch 1/13
33/33 [==============================] - 464s 14s/step - loss: 20.2694 - accuracy: 0.4569 -
Epoch 2/13
33/33 [==============================] - 464s 14s/step - loss: 0.5317 - accuracy: 0.7815 -
Epoch 3/13
33/33 [==============================] - 463s 14s/step - loss: 0.1485 - accuracy: 0.9508 -
Epoch 4/13
33/33 [==============================] - 468s 14s/step - loss: 0.1040 - accuracy: 0.9708 -
Epoch 5/13
33/33 [==============================] - 464s 14s/step - loss: 0.3827 - accuracy: 0.8723 -
Epoch 6/13
33/33 [==============================] - 466s 14s/step - loss: 0.3210 - accuracy: 0.9246 -
Epoch 7/13
33/33 [==============================] - 463s 14s/step - loss: 0.3927 - accuracy: 0.8785 -
Epoch 8/13
33/33 [==============================] - 470s 14s/step - loss: 0.7310 - accuracy: 0.6785 -
Epoch 9/13
33/33 [==============================] - 468s 14s/step - loss: 0.5108 - accuracy: 0.7646 -
Epoch 10/13
33/33 [==============================] - 468s 14s/step - loss: 0.3574 - accuracy: 0.8662 -
Epoch 11/13
33/33 [==============================] - 468s 14s/step - loss: 0.4907 - accuracy: 0.8600 -
Epoch 12/13
33/33 [==============================] - 468s 14s/step - loss: 0.2508 - accuracy: 0.9123 -
Epoch 13/13
33/33 [==============================] - 467s 14s/step - loss: 0.1446 - accuracy: 0.9538 -
<keras.callbacks.History at 0x7f43cf6fb190>
```

Fig. 4 Final Accuracy of the Model is Rectangled in Red

The optimizer used here is 'Adam'. Customized learning rate is 0.001 and the loss function used is 'Sparse Categorical Cross Entropy'. Accuracy is the metrics or label that is shown at the end of each epoch.

```
from keras.optimizers import Adam
model.compile(Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Fig. 5 Python Code for the Model Compilation Using the Above Stated Parameters and Values

B. Testing the Model

After the successful completion of the training, it is time to test it with some unseen inputs. Ten random images, all belonging to each of a kind has been used as a testing dataset. The amount of training dataset is kept small as it is the first dry run of the model. The testing function returns the maximum prediction value of the three outputs which are in the order of 'Beaglebone Black', 'Arduino' and 'Jetson Nano'. The function here looks as:

```
def prediction(pred):
 if pred[0][0]==max(pred[0][0],pred[0][1],pred[0][2]):
    return 'Beaglebone Black'
 elif pred[0][1]==max(pred[0][0],pred[0][1],pred[0][2]):
    return 'Arduino'
 else:
    return 'Jetson Nano'

test_img=os.path.join(#--image url--#)
j=load_img(test_img)
plt.figure()
plt.imshow(j)
i=load_img(#--image url--#)
i=img_to_array(i)
i=cv2.resize(i,(400,400))
input_image=i[np.newaxis,...]
pred=model.predict(input_image)
print(pred[0])
plt.title(prediction(pred))
```

Fig. 6 Python Code for the Model Output Prediction Function

When the above code is executed using valid image URLs, the model predicts which category image it is and it goes somewhat like this:

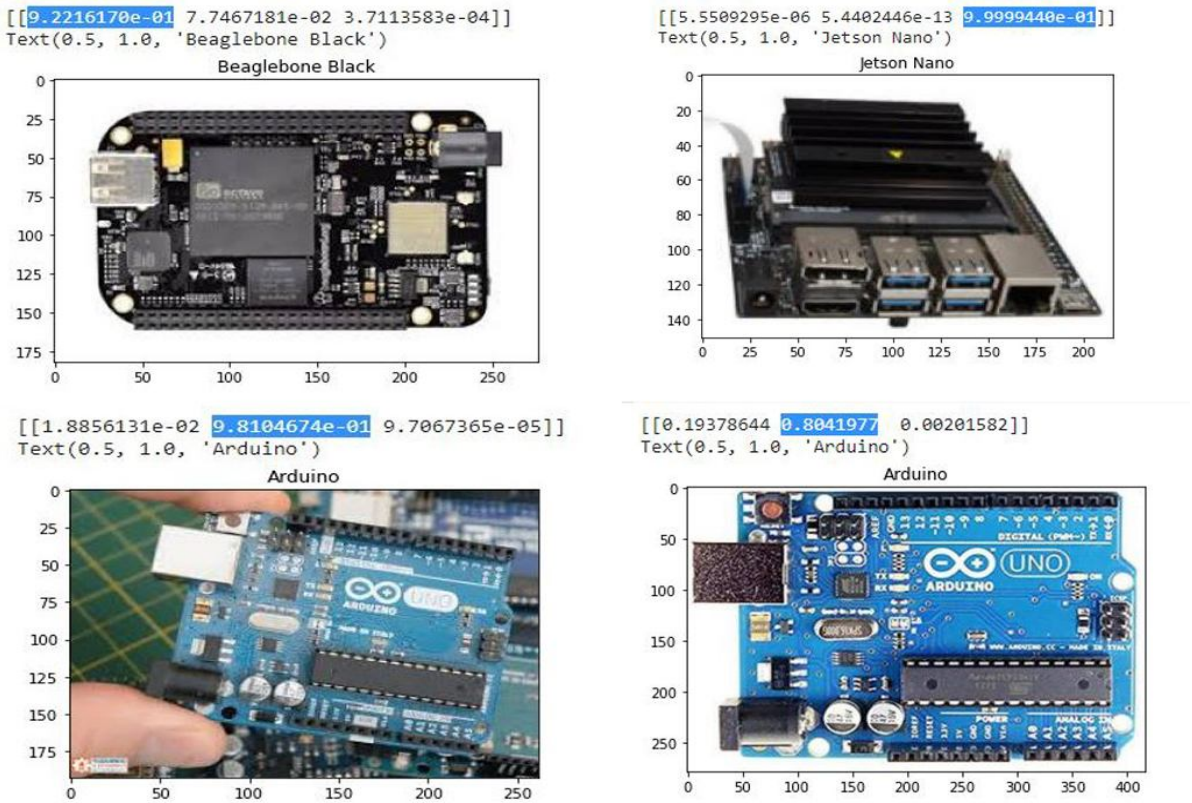


Fig. 7 Prediction of First Four Testing Input Images

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue IX Sep 2021- Available at www.ijraset.com

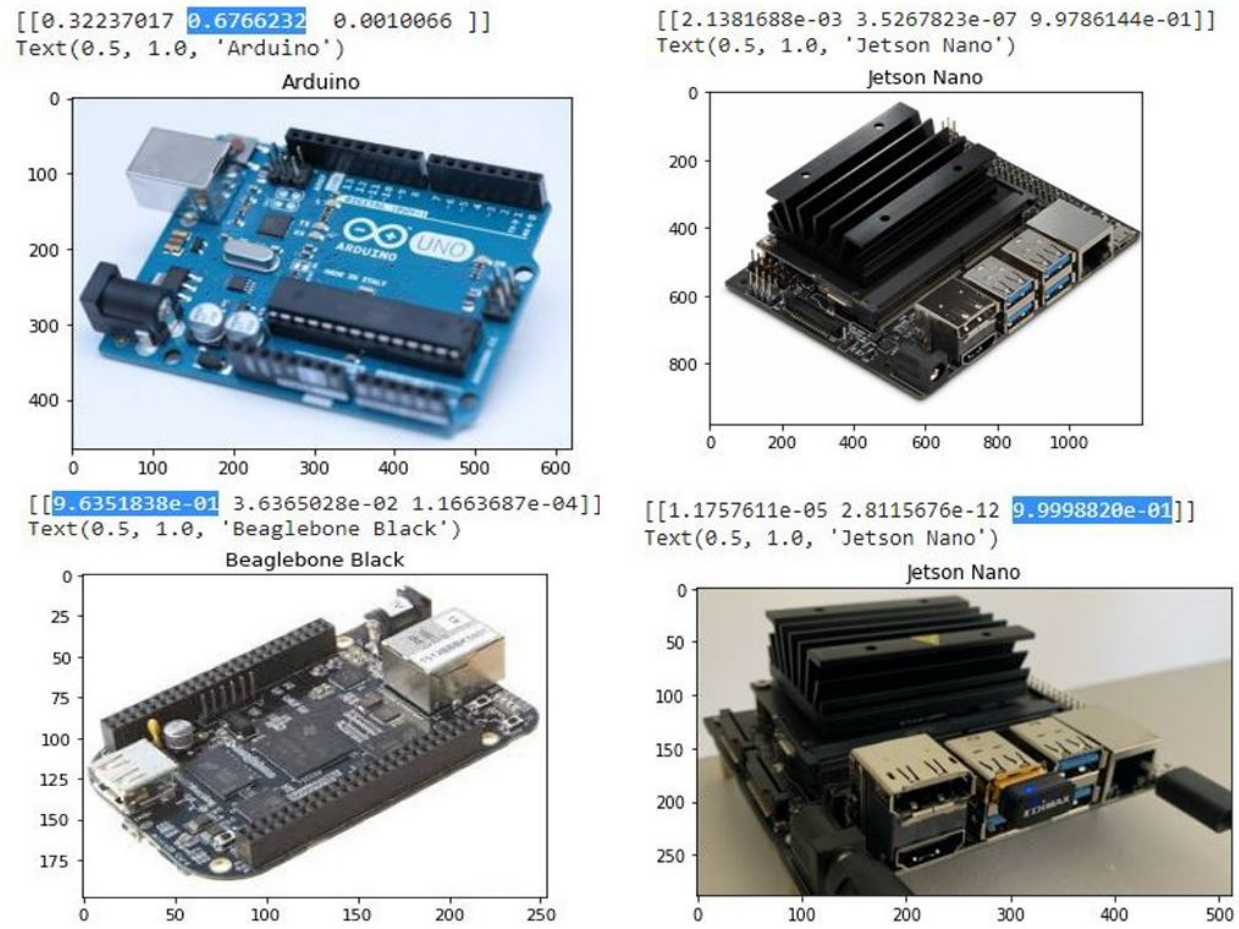


Fig. 8 Prediction of Next Four Testing Input Images

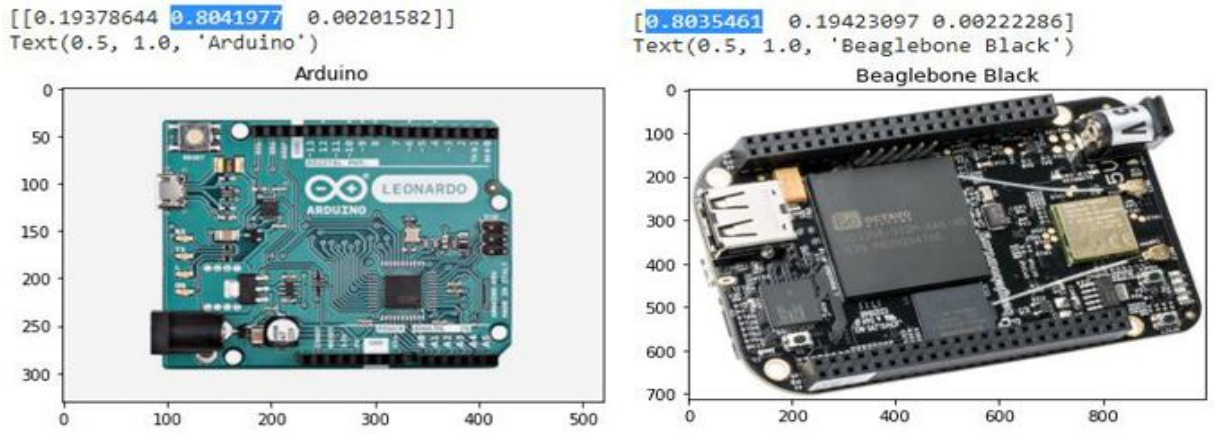


Fig. 9 Prediction of Last Two Testing Input Images

The value of the maximum prediction value is highlighted with blue and hence the output is the corresponding category for the maximum value. Apart from these several other images were also tested. Most of them produced a desired result. This is a demonstration of how Artificial Intelligence is extremely helpful to reduce the human workload. Application of such models includes data segregation and automatic identification of various objects as well as human peripherals.

IV. CONCLUSIONS

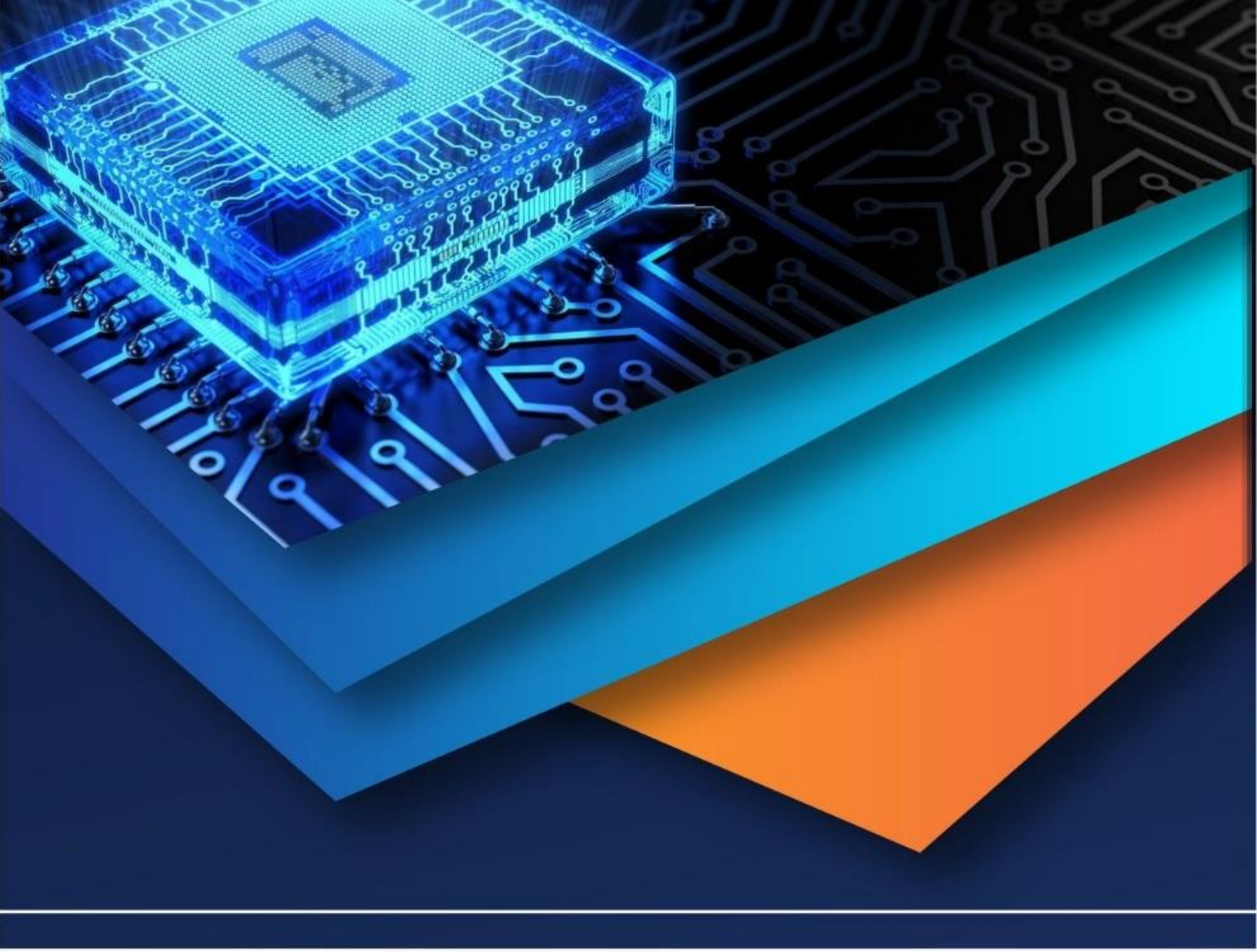
This is an example of how a deep learning model which uses neural network can be beneficial for use in various activities without manual intervention. Speaking of such models, if to be applied on a large scale, should undergo multiple testing before drawing any final conclusion about it. This specific model can be accepted as it has been validated by multiple testing of various types, of course by maintaining the category bounding. This methodology can be used for high scale predictions after fine adjustment of the model architecture and other parameters related. These tools of automation give the humans a new way of life every day.

V. ACKNOWLEDGMENT

A warm gratitude to my friends who helped me in the accomplishment, to my college Narula Institute of Technology, to our respected H.O.D (Head of Department) Dr. Anilesh Dey and to the edutech organization Verzeo who helped me to gain the basic essentials about how to build it up and make it in a working condition. To be honest, there are many surrounding help taken as well from internet and other learning resources. To sum up this is an assembled product of many contributions which are the key factors for this publication successful. Thank you everyone.

REFERENCES

[1] D. Jude Hemnath and Vania Vieira Estrela, *Deep Learning for Image Processing Applications*, IOS Press BV. Amsterdam, Netherlands : 2017. DOI: 10.3233/978-1-61499-822-8-1
[2] Licheng Jiao and Jin Zhao, *A Survey on the New Generation of Deep Learning in Image Processing*. School of Artificial Intelligence, Xidian University. Xi'an 710071, China, 12 December 2019. DOI: 10.1109/ACCESS.2019.2956508
[3] https://www.geeksforgeeks.org/opencv-python-tutorial
[4] https://www.tensorflow.org/
[5] Sulabh Srestha and Basanta Joshi, "Image Classification with Deep Convolutional Neural Network," The British College, Leeds Beckett University, Kathmandu, Nepal, May 11, 2018. Available at: https://www.researchgate.net/publication/325358907_Image_Classification_using_Deep_Convolutional_Neural_Network



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY