Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

Real Time Bird Detection and Recognition using TINY YOLO and GoogLeNet

Santhosh Kumar V
VIT University
Software Engineering Dept.
Vellore, Tamilnadu

Anupriya K
VIT University
Software Engineering Dept.
Vellore, Tamilnadu

Hari Balaji S
VIT University
Software Engineering Dept.
Vellore, Tamilnadu

Prof. Prabhavathy P
VIT University
Software Engineering Dept.
Vellore, Tamilnadu

Abstract: **Real time bird detection and recognition is a pre-eminent task in image processing. Till now many people used various deep learning algorithms for real time bird detection such as SSD, F-CNN, etc. In this paper we are going to use TINY YOLO along with the GoogLeNet architecture for real time bird detection.**

Detecting and recognizing of birds consists of three stages that include training, inference, and deployment. Training stage usually takes longer time and it is a complex process too. To overcome this problem, we make use of the previously trained datasets. This can be done using a hardware called Neural Compute Stick (NCS) where billions of parameters of images are loaded in this hardware (built-in training stage).

In this paper we are going to use two NCS, one along with TINY YOLO to take an image and get bounding boxes of the input image and another one with GoogLeNet for further classification of the birds found. The proposed method applies a few enhancements such as default boxes, multi scale features and depthwise separable convolution [11]. These enhancements permit the proposed system to get a high accuracy, high FPS in detection and recognition of birds and to use this in real time applications.

Keywords: Bird detection and recognition, TINY YOLO, GoogLeNet, NCS, FPS.

I. INTRODUCTION

Real time bird detection and recognition consists of two phase's identification and classification. Identification phase means identifying whether the bird is present or not. In classification phase we classify the detected birds e.g. bald eagle, etc. This real time bird detection can be used in various areas to save lives of many birds. We can implement this in airplane by detecting the birds continuously during flight. When birds detected, the bird repellents (like pyrotechnics, cannons) are automatically turned on preventing from the situation of bird strikes. Moreover it not only saves the lives of birds, during severe bird strikes the entire engine may get collapse leading to the death of the passengers can also be solved. Bird strikes can also happens in other man made things like wind turbines, vehicles, power lines leading to the death of the many birds. This can also be solved by implementing the idea presented in this paper.

Not only for saving lives of birds and humans, we can also implement our idea in various areas such as, a photographer (Alan McFadyen) waited for 6 years to take a photo of kingfisher bird diving straight into a water. He took 7, 20,000 attempts and 4,200 hours to take the photo. By implementing our idea we can automatically take photo as per our requirements. Likewise the proposed methodology can be used in various real time applications.

II. IDEA BEHIND OUR PROJECT

This idea of real time bird detection consists of three stages. Initial stage is called training stage [11]. In training stage we need to train the model by giving many bird images as input. The model will analyze each and every image and extract the parameters of those images. Training stage is a complex process since we need to train each and every image, extract the parameters in such a way that model should understand and store those parameters. Training a single image take more time. So to overcome this complexity, in this project we used a hardware called Neural Compute Stick (NCS). NCS was introduced by Intel which has built-in training data for billions of images. We can make use of these built-in training data just by invoking the libraries. Along with these we used OpenCV's deep neural network library and caffe deep learning library to perform bird detection and recognition. There are many deep learning libraries available but we used caffe because of its better performance.

Second stage is called Inference [11]. In inference stage the model will compare the input images with the trained sample images. Moreover this proposed method has an advantage compare with previous method (training each and every image) because previous method detect image by comparing input image with the sample image directly which produce less accuracy result. Because of this reason can't be applied in real time applications.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

The existing method produce less accuracy because input image of the eagle that are found in India and the sample image (image that is trained) of the eagle found in America, even though they are eagle they won't produce high accuracy. The reason is it won't match perfectly while comparing.

Whereas proposed method detect image by comparing input image with already trained parameters which produce high accuracy, probability and reduced processing time. Therefore with these qualities, the proposed method can be applied in real time applications. In proposed method everything (including detection and classification) is done in the form of graph files. Last stage is called deployment where we implement the model in real time applications.

Our major contribution in this project is using the NCS (built-in training data) which uses the previously trained parameter and eliminates the complex training stage which in turn produce high accuracy result and less processing time.

III. METHODOLOGY

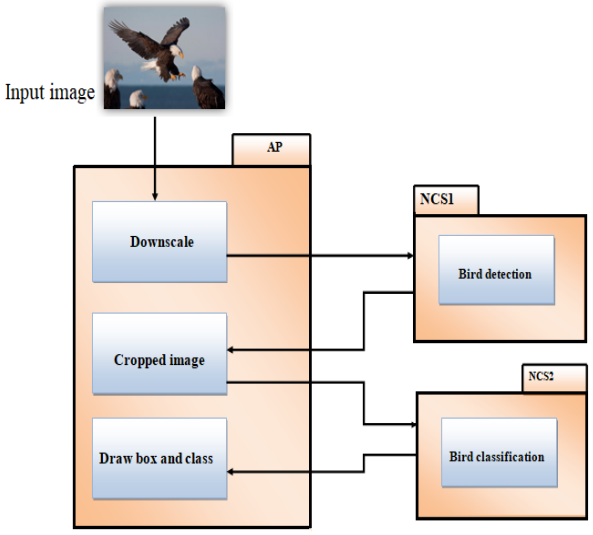


Figure 1: Entire process that demonstrates our work.

Bird detection and recognition is a general term commonly used in computer vision techniques to analyze and identify the birds present in an input image or videos. Various deep learning algorithms (such as R-CNN, Focal loss) have been used in detecting birds. These algorithms won't be suitable for the real time applications because processing time of each image (it takes around 47 seconds) is high during training and during inference they can recognize the image at the maximum of only 10FPS (Frames per second). But we require less processing time and at least 24 FPS for recognition of images in order to use it in real time applications. So to recognize at faster FPS and train images at less processing time in this paper we proposed a new detection methodology. We used a hardware device called NCS which achieve faster FPS even on low power devices. In this project we used two NCS for bird detection and classification. One for analyze and detect (TINY YOLO) and another for classification of the detected bird (GoogLeNet). Pre-training for detection of birds (i.e. trained dataset) is done using TINYYOLO.

TINY YOLO:

Advantage over other algorithm is, in other algorithm detection occurs by reprocessing the classifiers, therefore many neural networks will be used and result won't be found within a single evaluation. Therefore it takes more processing time and won't provide more accuracy. To satisfy these conditions we go for TINY YOLO in which detection occur through a single neural network by using bounding boxes and associated class probabilities. Since we use single evaluation, processing time will be greatly reduced and we get accurate probability. Moreover TINY YOLO able to achieve 244 FPS (training stage) on a computer with a GPU which is comparatively high compared to existing method [2].

GoogLeNet Architecture:

In our project we used GoogLeNet Architecture only during inference stage. We used this architecture along with TINY YOLO. TINY YOLO helps to determine the bird and further classification of the bird is done using GoogLeNet Architecture. We used GoogLeNet architecture for classification of the detected birds because of the presence of inception layer. This inception layer allows us to determine more accurately even if the size of the bird (face) is different for each input. Inception layer is the combination of 1x1, 3x3, 5x 5 Convolutional layers, with their result combined and provided as a input to the next stage. Because of the presence of inception layer in GoogLeNet architecture we used this architecture during inference stage.

To use these algorithms we need to train the Convolutional neural network with loss function and bounding boxes. Generally this loss function is used to compute the error that occurs during prediction therefore it is a complex process [1].

Due to the presence of varying parameters in our model we can't get complete match between ground-truth and predicted bounding boxes. So to get the accurate match, Intersection over Union (IOU) matching strategy is used during the training stage.

a) Training stage

Bird can be detected by training the collected datasets with bounding boxes and given class label for each input.

1) Data augmentation:

During training stage we need huge amount of data to train a single class (e.g. eagle). To recognize a single class we need to train at least 359 images. Collecting a similar data at different locations is a difficult task. So we used the concept of data augmentation instead of collecting image at different locations.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

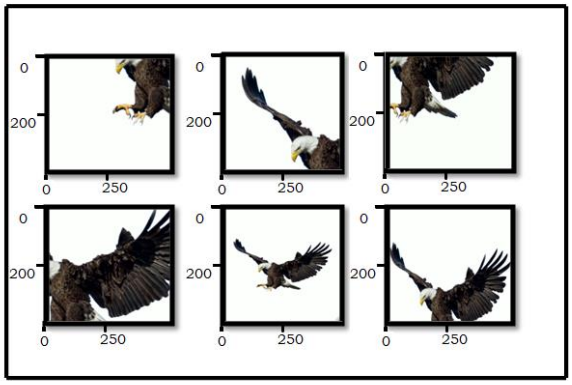


Figure 2: Data Augmentation.

2) Loss function:

TINY YOLO predicts multiple bounding boxes within a grid cells. Loss function is used to compute the error that occurs during prediction. TINY YOLO uses the error sum of squares method to find the error between the ground truth and prediction.

3) Multi-Scale feature maps for bounding boxes:

With the help of data augmentation, images with diverse size and locations were made and these images were processed and those results are then combined [1]. Along with that we used feature maps to a single neural network. As a result we can obtain coordinates for each bird located in the image. These coordinates form the bounding boxes for each bird located in the image. If we take local features (features in each Convolutional layer) we can't predict the correct class, but if we add global context along with that we can predict class correctly, so we used Multi-Scale feature maps for bounding boxes.

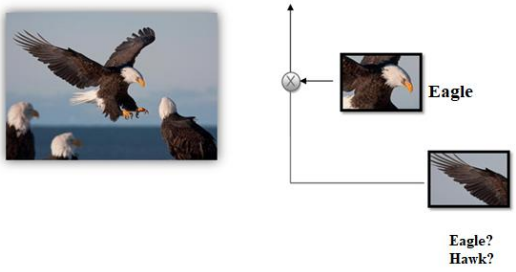


Figure 3: Multi scale feature maps, by combining feature from each Convolutional layer we can exactly predict the output.

4) Matching strategy:

During prediction if the bird is detected, multiple boxes will be created on the input at different scales and spatial locations, these boxes act as default boxes [1]. These default boxes have IOU value, if the value is greater than 0.5 it is considered for a match. The default box with maximum IOU is selected as our match [1].

b) Inference stage

1) Detection:

After training the entire model, we go for detection stage i.e. Inference stage. We use TINY YOLO algorithm for this stage. Detecting the bird occur within a single neural network. Initially the given input image is split into 7×7 grid cells. The input image is processed by 9 Convolutional layers and 3 fully connected layers. While processing the input image the stride value = 1 and the kernel size = 3, these value remains constant throughout entire process of bird detection [3]. The grid cells were divided in such a way the middle points of the bird falls within the grid cells.

Two bounding boxes are predicted by each grid cells. These bounding boxes can be described by five parameters which includes x and y coordinates, height and width of bounding box, and confidence (whether the bounding box contains the bird or not). The parameter confidence is used to determine how precise the bird is located at center of bounding box i.e. confidence = IOU × Pr (bird) [3].

Finally the outcome of above process is grouped into 7×7 segments. Therefore each segment will have one to one relation between each grid cell (both are 7×7). Each segment contains 30 values; these values were divided and distributed for each grid cells. It is distributed in such a way 20 values for class probabilities and 10 values for two bounding boxes.

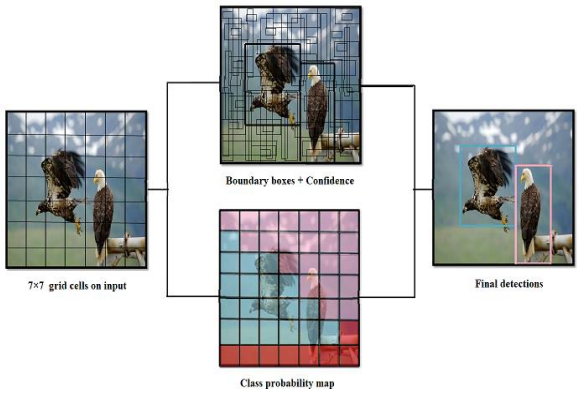


Figure 4: TINY YOLO divides the input into 7×7 grid cells; each grid cell has 2 bounding cells and each bounding box have five parameters.

2) Further classification:

Output of the detected bird image from the TINY YOLO is cropped out and given as input to GoogLeNet architecture for further classification. GoogLeNet architecture is 27 layers deep CNN. The presence of the inception layer (among 27 layers) helps us to focus on the individual parts of the face in the cropped image and identify the different parts of the bird face. They have different filters to identify different parts of the bird face. In this way the detected bird is classified further.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

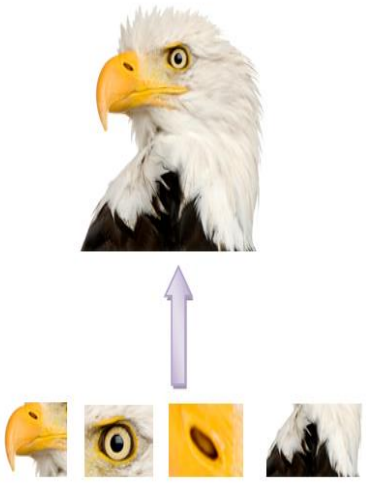


Figure 5: To recognize the face of the bird, inception layer works accordingly in this way.

c) Non maximum suppression:

When bird is detected more prediction's boxes (which have IOU > 0.5) will be produced around the detected bird [1]. These prediction boxes were arranged based on their confidence score. To get the single prediction box (high IOU) we apply a method called Non maximum suppression [1]. This method removes the prediction's boxes with lower IOU (duplicate boxes) that are around the detected bird. This indicates only the good predictions were kept and remaining predictions were ignored. Applying this method is the last step of detection process.

IV. ARCHITECTURE

Neural compute stick consists of two microprocessor (Vision processing unit and Leon Microprocessor).Execution flow is controlled by Leon microprocessor and computing part is taken care by SHAVE processors (a part of Vision Processing Unit).Vision Processing Unit (VPU) consists of 4 Gbit LPDDR3 Memory and SHAVE (Streaming Hybrid Architecture Vector Engine) programmable DL Engines which include 4*3(12) vector processors (also known as Shave Processor). These 12 Vector processors make every parts of neural network continuously run parallel to accelerate them.

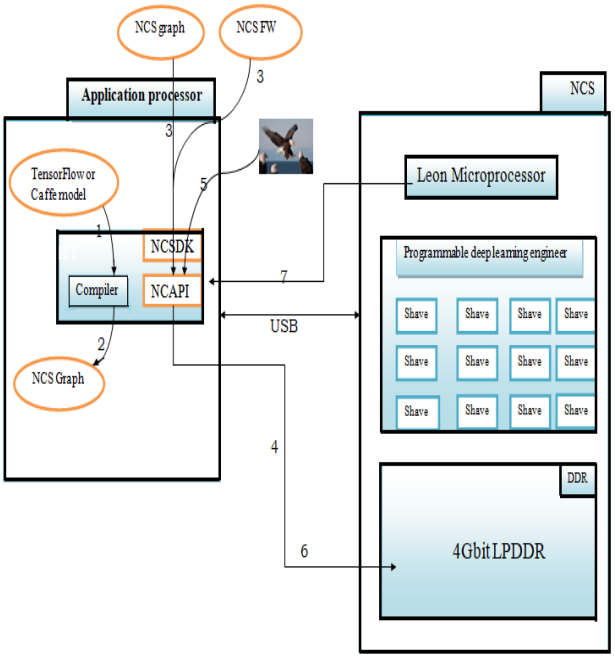


Figure 6: NCS Workflow

Step 1: Converting the model to graph file:

We need to connect NCS to host machine through USB interface. In the application processor (host machine) the developed model (which is in TensorFlow or Caffe) is processed by the compiler. Once the model is processed by compiler it is automatically converted to NCS graph file.

Step 2: Loading the Firmware to NCS:

When we connect NCS to host machine for the first time there will be no loaded firmware available on NCS. But during the program execution the host machine makes contact with VPU through USB using Neural Compute API (NCAPI).During this process NCAPI initializes and the firmware is loaded into the NCS from the NCSDK. Once the firmware is loaded to NCS, it is ready to accept the input in the form of neural network graph files. Generally the graph file and firmware is loaded into the 4Gbit LPDBR through NCAPI.

Step 3: Inference stage:

The Leon Microprocessor coordinates the images and receives graph file through NCAPI for inference. During this process, computing will take place in SHAVE processor simultaneously. Finally output of the neural network and associated calculations is send to application processor through USB and received by application processor through MCAPI.

There are three command line tools available for the process of inference. They are mvNCCompile, mvNCCheck, mvNCProfile

- mvNCCompile: Converts the given input (image/video) to graph file format [11].
- mvNCCheck: Compares the graph file from the NCS and the graph file is converted by the command line tool [11].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 11, November-2019

- mvNCProfile: It converts the graph file to text file. The text file contains the data of each layer (i.e. in each layer how the conversation happened) [11].

V. RESULT AND ANALYSIS

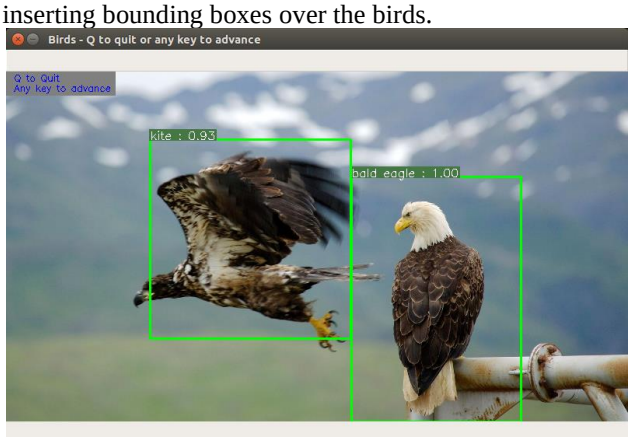
The proposed method of detection using NCS performs four operations which include capture frames, detect birds, classify birds, and calculating mean average precision for all detected birds.

Input:

The input is given in the form of image or video.

Output:

If bird is presented in the given input then the bird is detected and classified. The output will be displayed by inserting bounding boxes over the birds.



Network	Speed	Inference time	High accuracy
TINY YOLO	204	115.697 ms	8.64 fps

Table 1: Performance of the proposed method

Table 1 shows the processing speed of bird detection and recognition for the proposed methodology. For a maximum (Top) accuracy we obtained result at 8.64 fps.

Network	High accuracy
NCS + TINY YOLO	8.64 fps
Other methodology	2.34 fps

Table 2: FPS obtained from high accuracy

Table 2 implies proposed methodology of using NCS helps to reach more than 3.5 times faster than any other normal method of bird detection and recognition. Based on the accuracy level the detection of image at each frame (fps) varies. For instance, for the accuracy of 78% we can able to detect the bird at 30 fps. Through our proposed methodology we are able to obtain mean average precision.

VI. CONCLUSION

In this paper we proposed a new method of real time bird detection using hardware called Neural Compute Stick (NCS). Existing method of bird detection can't be used in real time application because the processing time of the image is high and moreover they detect birds at low frames per second with less accuracy. We require at least 24 FPS for implementing in real time application. In the proposed method we used NCS along with TINY YOLO which can overcome the above mentioned problems. Therefore by using the proposed methodology we can save lives of many birds from bird strike and also we can monitor the birds, identify their habitat and estimate the size of their population [4]. By implementing this proposed method all the requirements such as high accuracy, low processing time and detection of bird in real time application is satisfied. The final output we obtained is bird detected in bounding boxes (default boxes) with class probabilities.

REFERENCES

[1] Othman, N. A., & AYDIN, I. (2018, October). A New Deep Learning Application Based on Movidius NCS for Embedded Object Detection and Recognition. In *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)* (pp. 1-5). IEEE.

[2] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. in *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[3] Ma, J., Chen, L., & Gao, Z. (2017, November). Hardware implementation and optimization of tiny-yolo network. in *International Forum on Digital TV and Wireless Multimedia Communications* (pp. 224-234). Springer, Singapore.

[4] Hong, S. J., Han, Y., Kim, S. Y., Lee, A. Y., & Kim, G. (2019). Application of Deep-Learning Methods to Bird Detection Using Unmanned Aerial Vehicle Imagery. *Sensors*, *19*(7), 1651.

[5] Huang, R., Pedoeem, J., & Chen, C. (2018, December). YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 2503-2510). IEEE.

[6] Dean, T., Ruzon, M. A., Segal, M., Shlens, J., Vijayanarasimhan, S., & Yagnik, J. (2013). Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1814-1821).

[7] Intel Movidius Neural Compute Stick, 2017. https://developer.movidius.com/.(accessed September 28, 2017).

[8] MvNCCompile,2017.https://movidius.github.io/ .(accessed 2017)

[9] Blaschko, M. B., & Lampert, C. H. (2008, October). Learning to localize objects with structured output regression. In *European conference on computer vision* (pp. 2-15). Springer, Berlin, Heidelberg.

[10] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: single shot multibox detector, pp. 21–37 (2015)

[11] K. Banumathi, B. B. (2018). Real Time Vehicle Detection using Movidius Neural Compute Stick . *International Journal of Engineering Research & Technology (IJERT)* , 5.

[11] Mehta, R., & Ozturk, C. (2018). Object detection at 200 frames per second. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 0-0).