

PROFESSIONAL TRAINING REPORT

at

Sathyabama Institute of Science and Technology

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Technology in Information Technology

By

Shilp Gohil

(REG NO : 39120112)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND

TECHNOLOGY JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI-600119, TAMILNADU

NOVEMBER 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI– 600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of SHILP GOHIL (Reg . No: 39120112) who carried out the project entitled “AUTOMATED IDENTIFICATION OF FISH SPECIES USING DEEP LEARNING AND NEURAL NETWORKS” under the Supervision of Dr. Ms. Brumencia E and Dr. Bevish Jinila. Y respectively from June 2021 to November 2021.

Internal Guide

Dr. Bevish Jinila. Y

Head of the Department

Dr. R. Subhashini

DECLARATION

I Shilp Gohil hereby declare that the project report entitled "AUTOMATED IDENTIFICATION OF FISH SPECIES USING DEEP LEARNING AND NEURAL NETWORKS" done by us, under the guidance of Dr. Bevish Jinila. Y and is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Information Technology .

DATE:- 29-04-2021

SIGNATURE OF CANDIDATE



(Shilp Gohil)

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr. T. Sasikala M.E., Ph.D, Dean, School of Computing, Dr. R. Subhashini Heads of the Department of Information technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to our respective Project Guides Dr. Bevish Jinila. Y for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Information Technology who were helpful in many ways for the completion of the project.

I gratefully acknowledge the support of Tensorflow cooperation to provide us with repositories like label img and protobuf.

Abstract

Real-time recognition of dynamic gestures from video streams is a challenging task since there is no indication when a gesture starts and ends in the video, performed gestures should only be recognized once, and

the entire architecture should be designed considering the memory and power budget. In this work, we address these challenges by proposing a hierarchical structure enabling offline-working convolutional neural network (CNN) architectures to operate online efficiently by using sliding window approach. The proposed architecture consists of two models: A detector which is a lightweight CNN architecture to detect gestures, a classifier which is a deep CNN to classify the detected gestures. In order to evaluate the single-time activations of the detected gestures. We evaluate our architecture on our own datasets - which require temporal detection and classification of the performed gestures. In real-time detection and classification, we obtain considerable early detections while achieving performances close to offline operation. The codes and pretrained models used in this work are publicly available.

Table of content

TITLE	PAGE NO.
CHAPTER 1: Introduction.	7
CHAPTER 2: Related Work	10
CHAPTER 3: Aim and Scope of the present investigation	12
CHAPTER 4: Experiments or materials and methods; Algorithm used	13
CHQPTER 5: Results and discussion, performance analysis.	19
CHAPTER 6: Summary and Conclusion.	21

CHAPTER 1: Introduction

Computers and computing devices are becoming an essential part of our lives day by day. The increasing demand for such computing devices increased the necessity of easy and practical computer interfaces. For this reason, systems using vision-based interaction and control are becoming more common, and as a result of this, gesture recognition is getting more and more popular in research community due to various application

possibilities in human machine interaction.

Compared to mouse and keyboard, any vision based interface is more convenient, practical and natural because of the intuitiveness of gestures.

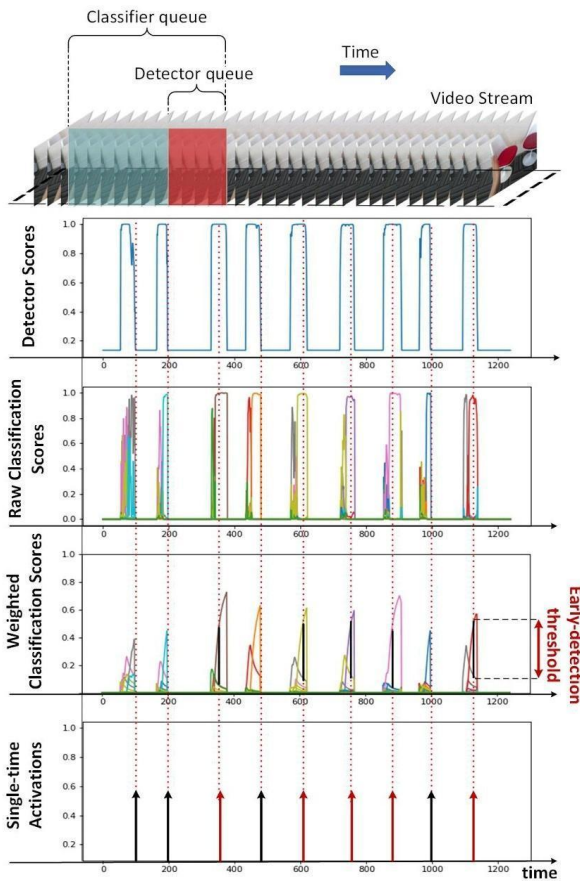


Fig. 1: Illustration of the proposed pipeline for real-time gesture recognition. The video stream is processed using a sliding window approach with stride of one. Top graph shows the detector probability scores which is activated when a gesture starts and kept active till it ends. The second graph shows classification score for each class with a different color. The third graph applies weighted-average filtering on raw classification scores which eliminates the ambiguity between possible gesture candidates. The bottom graph illustrates the single-time activations such that red arrows represent early detections and black ones represent detections after gestures finalize.

• Gesture recognition can be practiced with mainly three methods: Using

- glove-based wearable devices,
- 3- dimensional locations of keypoints,
- raw visual data.

The first method comes with the obligation of wearing an additional device with which lots of cables come even though it provides good results in terms of both accuracy and speed. The second, on the other , requires an extra step of -keypoints extraction, which brings additional time and computational cost. Lastly, for raw visual data only an image capturing sensor is required such as camera, infrared sensor or depth sensor, which are

independent of the user. Since the user does not require to wear a burdensome device to achieve an acceptable accuracy in recognition and sufficient speed in computation, this option stands out as the most practical one. It is important for the infrastructure of any gesture recognition system to be practical. After all, we aim to use it in real life scenarios.

In this work, in order to provide a practical solution, we have developed a vision based gesture recognition approach using deep convolutional neural networks (CNNs) on raw video data. Currently, CNNs provide the state-of-the-art results for not only image based tasks such as object detection, image segmentation and classification, but also for video based tasks such as activity recognition and action localization as well as gesture recognition.

In real-time gesture recognition applications, there are several characteristics that the system needs to satisfy:

- An acceptable classification accuracy,
- fast reaction time,
- resource efficiency and
- single-time activation per each performed gesture.

All these items contain utmost importance for a successful real-time vision based gesture recognition application. However, most of the previous research only considers and tries to increase the offline classification accuracy in gesture recognition disregarding the remaining items. Some proposed approaches are even impossible to run in real-time since they consist of several deep CNNs on multiple input modalities, which is forcing the limits of memory and power budget.

In this paper, we propose a hierarchical architecture for the task of real-time gesture detection and classification that allows us to integrate offline working models and still satisfy all the above-mentioned attributes. Our system consists of an offline-trained deep 3D CNN for gesture classification (classifier) and a light weight, shallow 3D CNN for

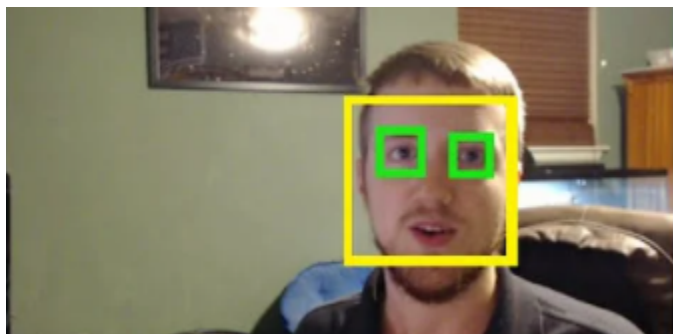
gesture detection (detector). Fig. 1 illustrates the pipeline of the proposed approach. A sliding window is used over the incoming video stream feeding the input frames to the detector via detector queue. Top graph in Fig. 1 shows the detector probability scores which become active when the gestures are being performed, and remain inactive for the rest of the time. The classifier becomes active only when the detector detects a gesture. This is very critical since most of the time, no gesture is performed in real time gesture recognition applications. Therefore, there is no need to keep the high-performance classifier always active, which increases the memory and power consumption of the system considerably. The second graph shows the raw classification scores of each class with a different color. As it can be seen from the graph, scores of similar classes become simultaneously high especially at the beginning of the gestures. In order to resolve these ambiguities, we have weighted the class scores to avoid making a decision at the beginning of the gestures (third graph in Fig. 1). Lastly, the bottom graph illustrates the single-time activations, where red arrows represent the early detections and black ones represent the detections after gestures end. Our system can detect gestures earlier in their nucleus part, which is the part distinguishing the gesture from the rest. We propose to use the Levenshtein distance as an evaluation metric to compare the captured single-time activations with ground-truth labels. This metric is more suitable and evaluative since it can measure misclassifications, multiple detections and missing detections at the same time.

We evaluated our approach on fish species data set prepared and labeled by us. For the classifier of the proposed approach, any offline working CNN architecture can be used. For our experiments, we have used well-known C3D and mobilenetXt-101. We have achieved the state-of-the-art offline classification accuracies of 94.03% and 83.82% on depth modality with mobile_ne_v2 architecture on our datasets, respectively. For real-time detection and classification, we achieve considerable early detections by relinquishing little amount of recognition performance.

CHAPTER 2: Related Work

Object-detection api has been used across various platforms by many users. A similar work was done by sentdex for facial recognition.

The success of CNNs in object detection and classification tasks, has created a



growing trend to apply them also in the other areas of computer vision. For video analysis tasks, CNNs have been initially extended to be applied for video action and activity recognition and they have achieved state-of-the-art performances.

There have been various approaches using CNNs to extract spatio-temporal information from video data. Due to the success of 2D CNNs in static images, video analysis based approaches initially applied 2D CNNs. In [18], [8], video frames are treated as multi-channel inputs to 2D CNNs. Temporal Segment Network (TSN) [22] divides video into several segments, extracts information from color and optical flow modalities for each segment using 2D CNNs, and then applies spatio-temporal modeling for action recognition. A convolutional long short-term memory (LSTM) architecture is proposed in [4], where the authors extract first the features from video frames by a 2D CNN and then apply LSTM for global temporal modeling. The strength of all these approaches comes from the fact that there are plenty of very successful 2D CNN architectures, and these architectures can be pre trained using the very large-scale ImageNet dataset [3].

Although 2D CNNs perform pretty well on video analysis tasks, they are limited to model temporal information and motion patterns. Therefore 3D CNNs have been proposed in [20], [21], [6], which use 3D convolutions and 3D pooling to capture discriminative features along both spatial and temporal dimensions. Different from 2D

CNNs, 3D CNNs take a sequence of video frames as inputs. In this work, we also use the variants of 3D CNNs.

The real-time systems for gesture recognition requires to apply detection and classification simultaneously on continuous stream of video. There are several works addressing detection and classification separately. In [15], authors apply histogram of oriented gradient (HOG) algorithm together with an SVM classifier. The authors in [12] use a special radar system to detect and segment gestures. In our work, we have trained a light weight 3D CNN for gesture detection. Moreover, in human computer interfaces, performed gestures must be recognized only once (i.e. single time activations) by the computers. This is very critical and this problem has not been addressed well yet. In [13], the authors apply connectionless temporal classification (CTC) loss to detect consecutive similar gestures. However, CTC does not provide single time activations. To the best of our knowledge, in this study, it is the first time single time activations are performed for deep learning based gesture recognition.

CHAPTER 3: Aim and Scope of the present investigation

- Often species level catch reporting is having hindrances because of several reasons pertaining to manual efforts, which result in low or erroneous reporting. So the problem statement requires an AI based model for Image classification of Species.
- The solution we offered is to classify different fish species found in the Indian ocean and perform image recognition using Convolution Neural Networks(CNN)
- We made use of a pre-trained CNN model known as MOBILE_NET_V2_320x320. This model architecture contains different hyper-parameters that can be transferred and customised according to our dataset.

1) Detector: The purpose of the detector is to distinguish between gesture and no gesture classes by running on a sequence of images, which detector queue masks. Its main and only role is to act as a switch for the classifier model, meaning that if it detects a gesture, then the classifier is activated and fed by the frames in the classifier queue. Since the overall accuracy of this system highly depends on the performance of detector, we require the detector to be (i) robust, (ii) accurate in detection of true positives (gestures), and (iii) lightweight as it runs continuously. For the sake of (i), the detector runs on smaller number of frames than the classifier to which we refer as detector and classifier queues. For (ii), detector queue is placed on the very beginning of classifier queue as shown in Fig. 2, and this enables the detector to activate the classifier whenever a gesture starts regardless of the gesture duration. Moreover, the detector model is trained with a weighted cross entropy loss in order to decrease the likelihood of false positives (i.e., achieve higher recall rate). The class weights for no gesture and gesture classes are selected as 1 and 3, respectively as our experiments showed that this proportion is sufficient to have 98+% and 97+% recall rates in our datasets, respectively. Besides that, we postprocess the output probabilities, and set a counter for the consecutive number of no gesture predictions in decision of deactivating classifier. For (iii), M-10 architecture is constructed using the mobilenet block in Fig. 3 with very small feature sizes in each layer as given in Table I, which results in less than 1M ($\approx 862K$) parameters. F and N correspond to the number of feature channels and the number blocks in corresponding layers, respectively. BN, ReLU and group in Fig. 3 refers to batch

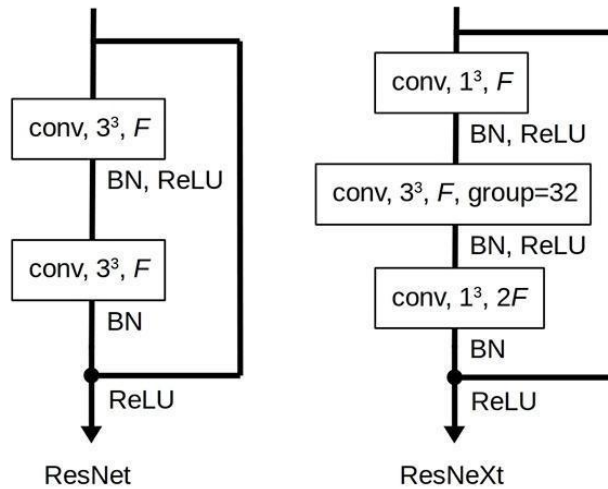


Fig. 3: ResNet and ResNeXt blocks used in the detector and classifier architectures.

normalization, rectified linear unit nonlinearities and the number of group convolutions, respectively.

2) Classifier: Since we do not have any limitation regarding the size or complexity of the model, any architecture providing a good classification performance can be selected as classifier. This leads us to use two recent 3D CNN architectures (C3D [19] and mobilenetxt-101 [7]) as our classifier model. However, it is important to note that our architecture is independent of the model type. For C3D model, we have used the exact same model as in [19], but only changed the number of nodes in the last two fully connected layers from 4096 to 2048. For mobilenet-101, we have followed the guidelines of [6] and chosen the model parameters as given in Table I with mobilenet block as given in Fig. 3.

Since the number of parameters for 3D CNNs are much more than 2D CNNs, they require more training data in order to prevent overfitting. Because of this reason, we pretrained our classifier architectures first on Jester dataset [1], which is the largest publicly available gesture dataset, and then fine tune our model on our datasets. This approach increased the accuracy and shortened the training duration drastically.

Training Details: We use stochastic gradient descent (SGD) with Nesterov momentum =

Layer	Output Size	ResNeXt-101	ResNet-10
conv1	L x 56 x 56	conv(3x7x7), stride (1, 2, 2)	
pool	L/2 x 28 x 28	MaxPool(3x3x3), stride (2, 2, 2)	
conv2_x	L/2 x 28 x 28	N:3, F:128	N:1, F:16
conv3_x	L/4 x 14 x 14	N:24, F:256	N:1, F:32
conv4_x	L/8 x 7 x 7	N:36, F:512	N:1, F:64
conv5_x	L/16 x 4 x 4	N:3, F:1024	N:1, F:128
	<i>NumCls</i>	global average pooling, fc layer with softmax	

TABLE I: Detector (ResNet-10) and Classifier (ResNeXt-101) architectures. For ResNet-10, max pooling is not applied when input of 8-frames is used.

0.9, damping factor = 0.9, and weight decay = 0.001 as optimizer. After pretraining on Jester dataset, the learning rate is started with 0.01, and divided by 10 at 10th and 25th epochs, and training is completed after 5 more epochs. For regularization, we used a weight decay ($\gamma = 1 \times 10^{-3}$), which is applied on all the parameters of the network. We also used dropout layers in and

several data augmentation techniques throughout training. For data augmentation, three methods were used:

(1) Each image is randomly cropped with size 112×112 and scaled randomly with one of $\{1, 1 \frac{1}{4}, 1 \frac{3}{4}, 2\}$ scales.

(2) Spatial elastic displacement [17] with $\alpha = 1$ and $\sigma = 2$ is applied on the cropped and scaled images. For temporal augmentation,

(3) we randomly select consecutive frames according to the size of input sample duration from the entire gesture videos. If the sample duration is more than the number of frames in target gesture, we append frames starting from the very first frame in a cyclic fashion. We also normalized the images into 0-1 scale using mean and standard deviation of the whole training sets in order to force models to learn faster. The same training details are used for the detector and classifier models.

During offline and online testing, we scale images and apply center cropping to get 112×112 images. Then only normalization is performed for the sake of consistency between training and testing.

Post-processing: In dynamic gestures, it is possible that the gesture gets out of the camera view while performing gestures. Even though the previous predictions of the detector are correct, any misclassification reduces the overall performance of the proposed architecture. In order to make use of previous predictions, we add the raw softmax probabilities of the previous detector predictions into a queue (qk) with size k, and apply filtering on these raw values and obtain final detector decisions. With this approach, detector increases its confidence in decision making, and clears out most of the misclassifications in consecutive predictions. The size of the queue (k) is selected as 4, which achieved the best results for stride s of 1 in our experiments.

4) Single-time Activation: In real-time gesture recognition systems, it is extremely important to have smaller reaction time and single-time activation for each gesture. Pavlovic et al. states that dynamic gestures have preparation, nucleus (peak or stroke [11]) and retraction parts [16]. Out of all parts, nucleus is the most discriminative one, since we can decide which gesture is performed in nucleus part even before it ends. Single-time activation is achieved through two level control mechanism. Either a gesture is detected when a confidence measure reaches a threshold level before the gesture actually ends (early-detection), or the gesture is predicted when the detector deactivates the classifier (late-detection). In late-detection, we assume that the detector should not miss any gesture since we assure that the detector has a very high recall rate. The most critical part of the early-detection is that, the gestures should be detected after their nucleus parts for a better recognition performance. Because several gestures can contain a similar preparation part which creates an ambiguity at the beginning of the gestures, as can be seen on the top graph of Fig. 5. Therefore, we have applied weighted averaging on class scores with a weight function. preparation part of the gestures is successfully resolved with this approach.

With this weighted-averaging strategy, we force our single time activator to make decision at mid-late part of the gestures after capturing their nucleus parts. On the other , we need a confidence measure for early-detections in real-time since the duration of gestures varies. Hence, we decided to use the difference between weighted average scores of each classes as our confidence measure for early detection. When the detector switches the classifier on, weighted average probabilities for each class is calculated at each iteration. If the difference between two highest average probabilities is more than a threshold τ_{early} , then early detection is triggered; otherwise, we wait for the detector to switch off the classifier and the class with the highest score above τ_{late} (fixed to 0.15 as it showed the

best results in our experiments) is predicted as late-detection. Details for this strategy can be found in Algorithm 1.

5) Evaluation of the Activations: As opposed to offline testing which usually considers only about class accuracies, we must also consider the following scenarios for our real time evaluation:

- Misclassification of the gesture due to the classifier,
- Not detecting the gesture due to the detector,
- Multiple detections in a single gesture.

Considering these scenarios, we propose to use the Levenshtein distance as our evaluation metric for online experiments. The Levenshtein distance is a metric that measures distance between sequences by counting the number of item-level changes (insertion, deletion, or substitutions) to transform one sequence into the other. For our case, one video and the gestures in this video correspond to a sequence and the items in this sequence, respectively. For example, lets consider the following ground truth and predicted gestures of a video:

GroundTruth ***[1,2,3,4,5,6,7,8,9]***

Predicted ***[1,2,7,4,5,6,6,7,8,9]***

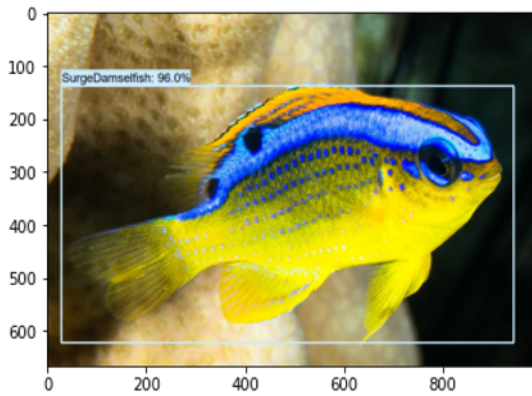
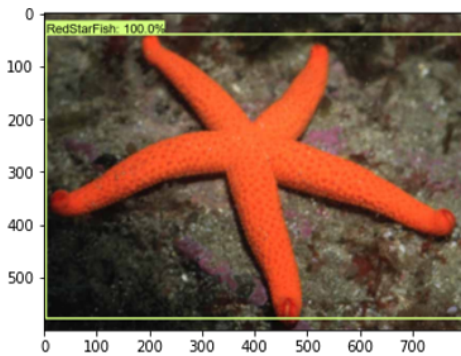
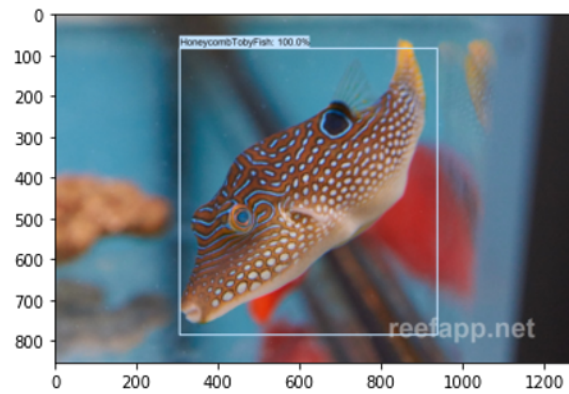
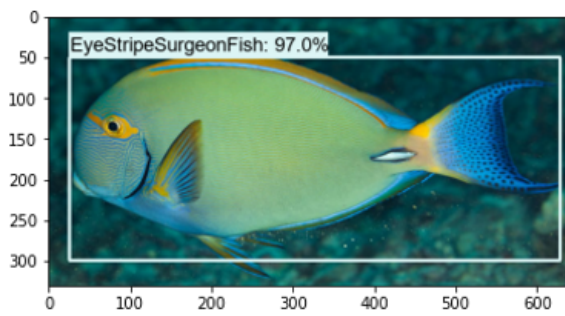
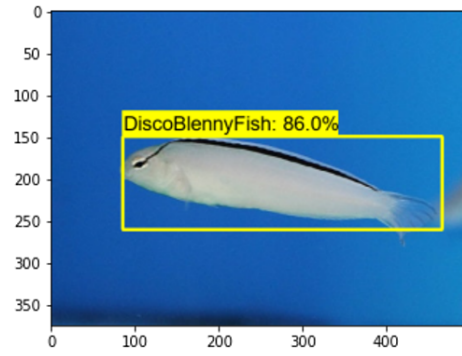
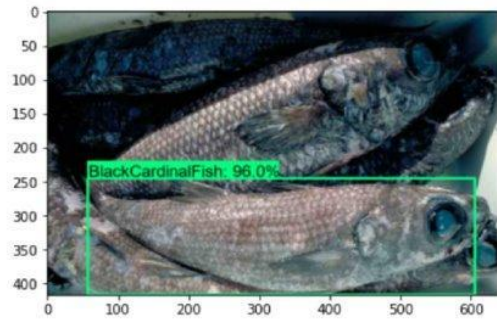
For this example, the Levenshtein distance is 2: The deletion of one of "6" which is detected two times, and the substitution of "7" with "3". We average this distance over the number of true target classes. For this case, the average distance is $2/9 = 0.2222$ and we subtract this value from 1 since we want to measure closeness (in this work it is referred as the Levenshtein accuracy) of our results, which is equal to $(1-0.2222) \times 100 = 77.78\%$.

CHAPTER 5: Results and discussion , performance analysis

- Images of different species are collected and classified into different folders.
- These images are then annotated using LABEL_IMG(annotation tool) to generate labels for different species in xml format
- The images are then split into train and test folders(80-20 split) and are fed to the CNN model.
- The CNN model we chose is MOBILE_NET.V2_320X320. The dataset fed to this model goes through pre-processing and is ready to be trained.
- We used over 500 images for 7 different species
- After the dataset is taken as the input. The model is then trained for over 8000 epochs/Iterations.
- Once the training is done. The model is then executed.
- Detection and bounding boxes across the fishes are made possible using OPENCV
- The entire code is then converted into a web-app using flask, HTML and CSS
- We received an accuracy between 90-99%

About the Convolution Neural network model used:

- We are using a pre-trained model called Mobile_net_v2_320x320 that comes with pre-trained weights and hyper parameters that are then transferred to our own custom model using transfer learning.
- Weights can be changed to fit our own model and dataset



CHAPTER 6: Summary and Conclusion

- The model accuracy is 80% and can be increased by adding more datasets and by performing more training
- There is no delay in the detection process
- Live detections are done using a webcam

- In conclusion, our model with a minimum accuracy of 90% performs image recognition and classifies different species across the Indian Ocean. The image of the fish or the fish itself can be shown to the camera in order to detect the species. Our model can classify different species if shown together, which shows better classification.

LIBRARIES USED:-

- TENSOR FLOW
- OPENCV
- NUMPY
- PROTOBUF

REPOSITORIES USED:-

- OBJECT DETECTION
- LABELIMG
- MOBILENET.V2