**Scalable Object Detection & Model Comparison**

Dharamendra Kumar

Master’s in Computer Science, University of Georgia, Athens,

GA-30606

Rakesh Bhavsar

Master’s in Computer Science, University of Georgia, Athens,

GA-30606

Ayush Jaiswal

Master’s in Computer

Science, University of

Georgia, Athens,

GA-30606

**ABSTRACT**

We report our work on scalable object detection and deep learning model comparison using AWS serverless component i.e. Lambda function. We used Faster Region based Convolutional Neural Network method (Faster R-CNN) as well as Single Shot detection (SSD) model for detection and then match the object with features from both neural network and compare both models based on the metrics like bounding co-ordinates and mean average precision(mAP). We were able to achieve real-time performance and satisfactory model comparison results.

#### Keywords

**AWS Lambda, Faster-RCNN, SSD, AWS API Gateway,**

**DynamoDB, Keras**

# INTRODUCTION:

Object detection is an important problem in computer vision. Detection is not only about finding the class of object but also localizing the extent of an object in the image. The object can be lying anywhere in the image and can be of any size. Following are Object Detection techniques:

* Region Based CNNs
* R-CNN - 2013
* Fast R-CNN - 2015
* Faster R-CNN – 2015
* Single Shot Detection
* YOLO

R-CNN one of the most impactful advancements in computer vision. As evident by their titles, Fast R-CNN and Faster R-CNN worked to make the model faster and better suited for modern object detection tasks.

SSD is first deep network based object detector that does not resample pixels or features for bounding box hypotheses & is very accurate. This results in a significant improvement in speed for high-accuracy detection (59 FPS with mAP 74.3% on VOC2007 test, vs. Faster R-CNN 7 FPS with mAP 73.2% or YOLO 45 FPS with mAP 63.4%).

Integration of these well-trained model to real world application is challenging, an end-user interface for such use case has not been much prevalent in the web world. There is also deficit of an interface where we can simultaneously compare different algorithms and decide which model weights can be helpful to train further on the new dataset. Also, real-time feedback to such trained model can further improve weights over the time.

AWS Lambda can be one of the option for such integration. AWS Lambda functions provides serverless architecture along with scalability. Using this architecture any trained deep learning model can be used for scalable prediction.

Architecture has support for many object detection model, and this requires a mechanism to support routing based on the selected model and datasets. API Gateway does help in routing the request and pass the required parameters from user interface to lambda functions.

# PROBLEM STATEMENT

In this project, we are performing Object Detection by implementing Faster R-CNN & SSD using Keras Deep learning framework. We are using Server-less architecture (AWS Lambda) for deep learning model evaluation. Web UI where user uploads an image, then can see the result evaluated by models deployed on AWS Lambda function. Thus, users can then perform model comparison based on the results.

# PROPOSED METHODS

Faster RCNN: Faster RCNN workflow is shown in figure 1. Following steps are performed:

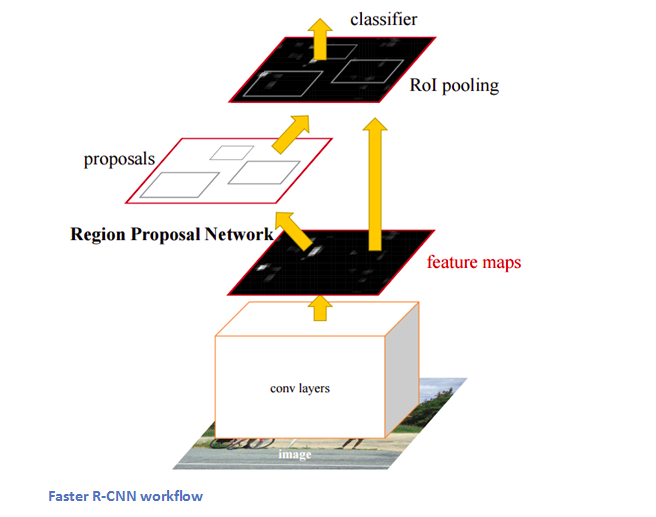
1. Insert a region proposal network (RPN) after the last convolutional layer.
2. This network was able to just look at the last convolutional feature map and produce region proposals from that.
3. From that stage, the same pipeline as R-CNN is used (ROI pooling, FC, and then classification and regression hea

Figure 1: Faster R-CNN

SSD: Single Shot Detection workflow is shown in figure 2. Following steps are performed:

1. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location.
2. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape.
3. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes.

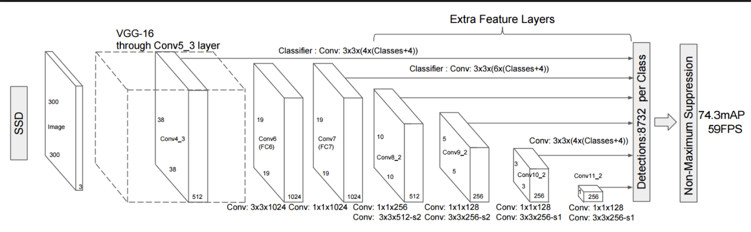


Figure 2: Single Shot Detection

To achieve the scalability for large number of users we designed our architecture in Amazon Cloud. Several services which makes such requirements as scalable for the end users. The model comparison is one of the key feature for this system. Various state of the art model has been proposed in object detection and taking decision on the model for end-user is sometime challenging, Architecture designed in such way that it can be extended to not only object detection models as well as models in segmentation, and classification can also be evaluated.

# SYSTEM ARCHITECTURE

In our proposed system, we have following important modules:

1. User Interface
2. Preprocessor Lambda
3. Model Lambda

# Object Detection Lambda

# Simple Storage Service

# API Gateway

# AWS Dynamo DB

# Working of each module is explained in

# following section (Functional Specification)

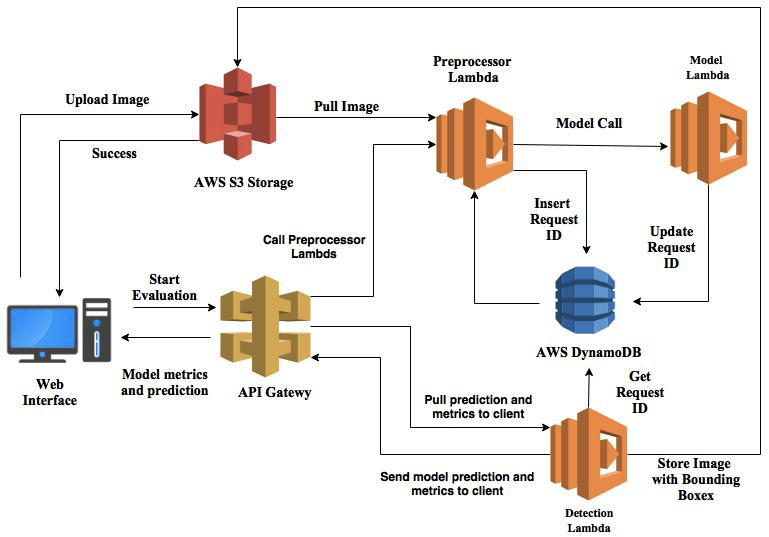


Figure 3: System Architecture

# FUNCTIONAL SPECIFICATION

# *User Interface*: We have designed a user interface using Nodejs where user can upload an image. User is provided with a Upload button as shown in the figure.

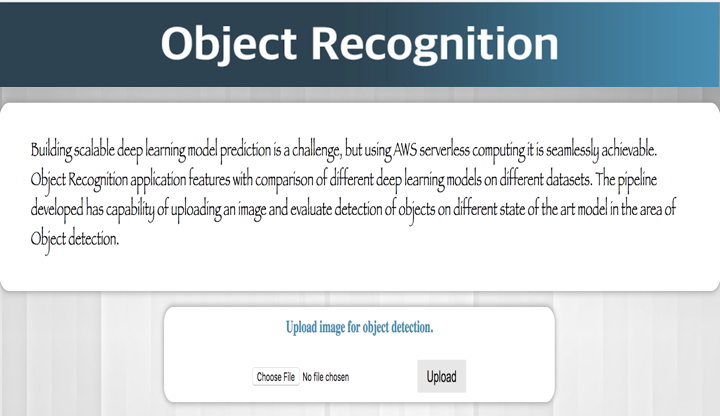


Figure 3: User Interface

# *Pre-processor Lambda:* The module has responsibilities to do pre-processing like scaling, normalization using OpenCV library and prepare the image for trained model to predict.

# *Model Lambda:* The role of this module is to load the trained model inside the lambda, and for this paper we have used Keras with Tensorflow as backend. The input for this module is the preprocessed image from pre-processor lambda. It predicts bounding boxes and classification for all the objects.

# *Object Detection Lambda:* Lambda function has limitation on the size of the code, so decoupling needs to be taken into consideration while designing of the architecture. This module aggregates the result after model lambda predicts the bounding boxes and classify all the object found in an image. The computation over here is to use non-max suppression algorithm for bounding boxes which had predicted with confident score more than the threshold. In case the annotation file for an image is provided, it computes mAP which helps in verifying bounding boxes predicted over the ground truth and objects found after prediction.

# *Simple Storage Service:* After uploading an image, it is stored in S3 which is a storage service provided by Amazon. Screen is displayed where user can select Dataset and Model and hit Start evaluation model. We have tested our application on following three datasets: PASCALVOC 2007/2012, COCO2015 & IMAGENET.

# *API Gateway:* API Gateway is an interface where user interface passes the parameters captured from user interface and based on the input parameters gateway rout the inputs for prediction, pre-processing and model loading.

# *DynamoDB:* This is NoSQL database for storage of data in form of key-value pair. We used this module for polling the status of request which user requested for object detection.

# ANALYSIS OF OUR PROPOSED SCHEME

# The pipepline in Figure3 is explained in detail.

# After uploading an image by the user, we display screen i.e. Figure4. At this point, user can select dataset and model, and presently it supports Faster-RCNN and SSD algorithms for the model selection.



Figure 4: Interface for dataset and model selection

# After selection of model and dataset, and optional upload of annotation file, user clicks on “Start *Evaluation*” to get the result. API Gateway has a limit of timeout on sending the response to user timeout, presently it supports not more than 29 seconds. Our model has huge number of training parameters which for some cases it takes longer than that, so to handle such long running processes we took advantage of *fire-and-forget*. Using this approach, we poll the status from DynamoDB. It works in following way:

# Pre-processor lambda after pre-processing insert a row in DynamoDB which have *RequestId, and Status (In Progress)*.

# Model lambda after loading the model and prediction update the same *RequestId* in DynamoDB with *Status(Processed)* and also store predicted object classes and corresponding bounding boxes for each class.

# Object detection lambda polls the status of *RequestID* from DynamoDB and show the status on user interface (Figure 5).

# 

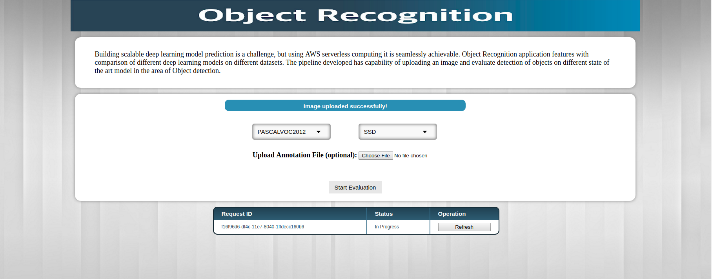


Figure 5: Interface for ’In progress’ status of the request

Figure 6 is the interface where user click on Refresh and after few seconds the status of the *RequestId*  is changed from *In Progress* to *Processed.* Figure 7 and Figure 8 shows the interface of object detection result with annotation and without annotation. In case annotation file is uploaded which have the information of bounding boxes and the number of objects present in uploaded image for evaluation, we calculate the mAP and display one of the metric for model comparison.

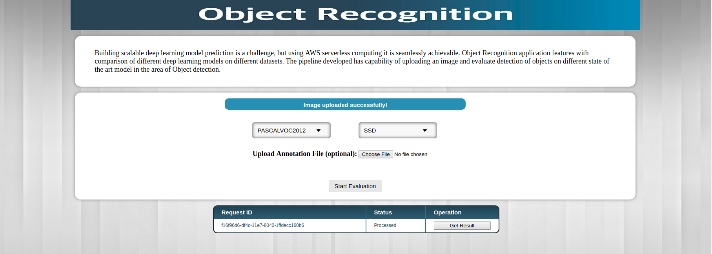


Figure 6: Interface for ’Processed’ status of the request

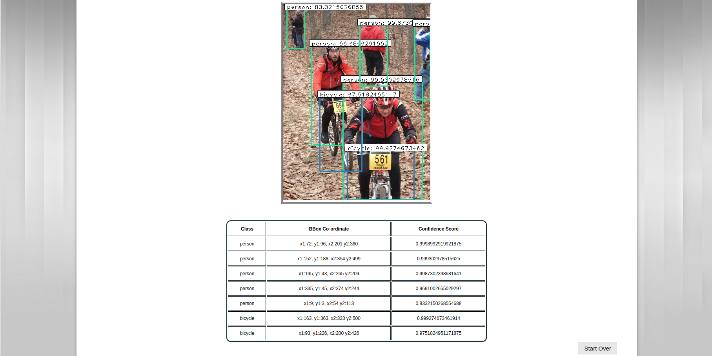


Figure 7: Interface for Object detection results

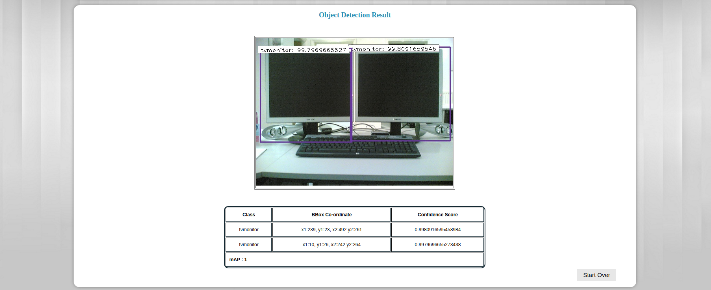


Figure 8: Interface for Object detection results with annotation

# ADVANTAGES AND BENEFITS

# Researchers often implement different object detection model to perform image analysis. Implementing various model and training each model is time consuming. A readily available model comparison web portal can help researchers by giving them holistic comparison of various model for the input and train the model. This will give them knowledge to choose the best model for their problem in hand.

# FUTURE SCOPE

# Model comparison is a vast field and a tremendous area for research and improvement. We have implemented Faster R-CNN and SSD. Another object detection algorithm can be implemented using the Lambda function. Training the model at the same time using the Lambda function will be a good addition to improve the model learning by deploying on AWS EC2.

# CONCLUSION

# In this paper, we successfully implemented object detection using Keras Deep learning network. We use Faster Region based Convolutional Neural Network method (Faster R-CNN) and Single Shot detection (SSD) model for detection. AWS- Amazon Web Services offer the server less computing which allowed us to design a scalable architecture that seamlessly provide object detection and model comparison. We implemented Faster R-CNN and SSD, among them, SSD is better in terms of accuracy and performance. For the same input image, SSD is much faster and accurate for object detection than Faster R-CNN. The mAP result of both the detection method provide the absolute difference of accuracy. The web user interface allows user to compare the model by uploading an image live and choose the best model that suits ones need. We have implemented only two detection technique (Faster R-CNN and SSD), but the AWS’s modular architecture allow as to incorporate as many models using the Lambda Function. We have used the trained model for our implementation. We can extend the model by training each model at the same time, this can be achieved by AWS’s EC2 instances and Lambda functions.

# The object detection and at the same time model comparison will help researchers to choose the best model for their problem in hand. The overhead of configuring each model and test in removed with the Web portal.

# REFERENCES

1. <https://arxiv.org/pdf/1512.02325.pdf>

# <https://arxiv.org/pdf/1506.01497.pdf>

# <https://github.com/yhenon/keras-frcnn>

# <https://github.com/rykov8/ssd_keras>

# <https://aws.amazon.com/blogs/compute/seamlessly-scale-predictions-with-aws-lambda-and-mxnet/>

# <https://aws.amazon.com/blogs/compute/seamlessly-scale-predictions-with-aws-lambda-and-mxnet/>