

#### **ABSTRACT**

Object detection is one of the exemplary issues in computer vision. It is very significant for computers/machines to perceive the common objects in life like the human mind and the human eye can. This article makes sense of the working of an object dientification framework that uses both pretrained and custom trained models that have a place within the group of most advanced Convolutional Neural Networks [3] there are to recognize and detect objects. Some of the ever so growing applications of object detection problem are also discussed in this article.

#### **OBJECTIVES**

The development of this system occurred considering the accompanying targets:

- To comprehend the use of most recent pretrained Convolutional Neural Network architectural models.
- To perform object detection using pretrained EfficientDet-D0 512x512 and mobilenet v2 fpnlite 320x320 architectural models
- To perform object detection using uniquely prepared EfficientDet-D0 model and mobilenet v2 fpulite 320x320 architectural models.

### INTRODUCTIO

Object recognition and detection is a critical computer vision task used to recognize instances of visual articles of explicit classes (for example, individuals, animals, vehicles, or designs) in advanced pictures, for instance, photos or video outlines. The target of object detection is to encourage computational models that give the most focal information expected by computer

vision applications.

Object discovery is inseparably connected to other comparative computer vision strategies and problems like image recognition and image segmentation, in that

it helps us comprehend and investigate scenes/frames in pictures or videos. In any case, there are significant contrasts between image recognition, image detection and image segmentation. image recognition just results a class mark for a distinguished

its extraordinary capacity to find objects inside a picture

or video. This then permits us to count and afterward

track those objects.

isolates object detection from these different errands is

evel comprehension of a picture's components. What

item/object, and image segmentation makes a pixel-

# **OBJECTIFIER**

# Muhammad Abdullah Usmani | Usama Saif National University Of Computer And Emerging Sciences, Karachi

## METHODOLOGY



# Pre-trained models setup Applications:

To comprehend the use of pretrained object detection models, we, utilizing python language and its ever so developing arrangement of functionalities, figured out how to concoct a python script, upon providing the link of a particular CNN model from TensorFlow model zoo[], downloads, loads and sets up that model for sure fire use, all in no time.

# Displaying annotations:

Utilizing cv2, a well-known python computer vision package, we figured out how to draw annotation boxes over objects detected by the models.

#### • Dataset:

We utilized the notorious PASCAL VOC2011 [1] dataset to prepare our Efficient Net and MobileNet models. This dataset contains 14,000+ pictures with 20 different article classes.

## · Data cleaning:

We fostered a python content to clean the PASCAL VOC2011 dataset

# · Training models:

We utilized EfficientNet-D0 512x512 and mobilenet\_v2\_fpulite\_320x320 architectures given in TensorFlow zoo [2] as a base for making our own special modified Efficient Net and MobileNet models.

We arranged the previously mentioned architectures in a manner so that they becomes viable with our chosen dataset and environment.

#### RESULTS

Four different types of Convolutional Neural Networks were used in this system with each of them having their own precision, accuracy and loss.

- When it comes to pre-trained Efficient Net D0 512x512, its mean average Precision is 33.6%.
- When it comes to pre-trained SSD MobileNet V2
   FPNLite 420x320, its mean average precision is 22.2
   The above stated facts can be verified from Tensor Flow
   Company of the content of

00 [2]
Whom it comes to me tenined Efficient Net

When it comes to pre-trained Efficient Net
 D0 512x512 and SSD MobileNet VZ FPNLite 420x320
 models, their precision and accuracy is not the best due to the fact that they were trained for only 10,000 steps in average due to the unavailability of GPUs.
 Still, these models showed somewhat promising results and gave us hope that with GPUs, they can perform much better. The Efficient Net model was evaluated on 3665 images and the results are presented below:

0/77	.473	.281	.633	.139	365	326	484	.518	.111	.381	.617
	00	(S)	00	63	en .	- cts	ca	CD.	co.	co.	- CD
1	_	4		4	-	_		_			
maxDets-166	maxDets=100	maxDets=100	maxDets=100	maxDets=100	maxDets=100	maxDets= 1 ] = 8.326	maxDets= 10	maxDets=100	maxDets=100	maxDets=100	maxDets=100
	311		Small s	molpe	= large	71	411	a11	Small.	mile	large
-gale	area=	area	areas	area-m	area	area	area	area			
1001E0.0816.95	ToU=8.5e	IoU=8.75	IoU=8.58:8.95	ToU-8.58:8.95	IoU=0.50:0.95	@[ IoU=8.50:0.95	IoU=6.58:0.95	IoU=8.58:8.95	IoU=8.58:8.95	ToU=8.58:8.95	ToU=8.50:0.95
8	100	000	79	000	700	8	08	100	18	100	00.
(AP)	(AP)	(AP)	(AP)	(AP)	(AP)	(AR)	(AR)	(AR)	(AR)	(AR)	(AR)
Precision.	Precision	Precision	Precision	Precision	Precision	Recall	Recall	leca1		Recall	
AVETABE	Average	Average	Average F	Average	Average	Average R	Average	Average	Average	Average	Average

SERVICE IS:38:18.648252 139506826134168 model_lib_v3.pp;1815] Evai metrics at step 25866 FMTO:temcorphism: betterfineDecember Percelision.org 9:074801   Per	1015] fval :: 0.274582 1018]	netrics at step 25000 + DetectionBoxes_Precision/maP: 0.274602
THE OFFICE AND A STATE OF THE OFFICE AND A STATE OFFI AND A STATE	8 8	+ DetectionBoxes_Precision/mapg.58IOU: 0.473245 8.281332
<pre>COLM 15:25:40:001464 1300908402144106 model_llb v2.py.1818] ThG:tensorino.</pre>	(Smell): 1818]	+ DefectionBoxes_Precision/mAPg://SiOU: 0.281532 0.832647 DefectionBoxes_Precision/wAP (sesil): 0.032647
TMFO:tensorflow: - DetactionBoxes_Precision/map (medium): 0.139429 BSS 15758:18, 662398 13969805024168 model_llb_v_2.py;388] + Detectl MPO:tensorflow: - Detactionmap (llres): 0.302181	(medium): 1818]	8.139429 + DetectionBoxes_Precision/mAP (medium): 8.139429 8.165281
(8518 15:28:10.662942 139696826124160 model 11b.v2.py:1018] TMFO:tencorflow: + DetectionBoxes_Recall/ARR1: 8.325847	1018]	+ DetectionBoxes_Precision/mAP (large): 8.365283
t0518 15:28:10.663529 139696826124168 model_110_v2.py:1018] TMFO:tensorflow: + DetectionBoxes_Recall/ARB18: 0.484674	1018]	+ DetectionBoxes_Recall/ARB1: 0.325847
139696826124160 model 11b_v2.py:1018] + DetectionBoxes Recall/ARG108: 0.50963	1618]	+ DetectionBoxes Recall/ASSIS: 0.484874
139696826124168 model 11b v2.py:1018]		+ DetectionBoxes_Recall/ARgid8: 0.509633
+ DetectionBoxes_Metall/ANGING (SMBIL): 139696826124168 model_11b_v2.py:1018]	1018]	6.111154 + DetectionBoxes_Recall/ARgidG (small): 0.111154
INFO:tensorflow: + DetectionBoxes_Recall/ARGNBB (medium): 0.381167 10518 15:28:10.665721 139696826124160 model_lib_v2.py:1018] + Detecti	(medium):	6.381167 + DetectionBoxes_Recall/AR@166 (medium): 6.381167
+ DetectionBoxes_Recall/ASB188 (large): 139696826124168 model_lib_v2.py:1818] - Intellocalitation_loce. 0 29431	(large): 1018]	8.617112 + DetectionBoxes_Recall/AD@168 (large): 8.617112
0518 15:28:10.668049 139696820124160 model_lib_v2.py:018 NFO:tensor-low: + Loss/classification_loss: 0.447305	1018]	+ loss/localization_loss: 0.226471
139696826124160 model_llb_v2.py:1018] + Loss/regularization_loss: 0.003652	1818]	+ Loss/classification_loss: 0.447305
10518 15:28:10.669126 139696826124160 model_11b_v2.py:1018] + Loss/total_loss: 6.767427	1918]	+ Loss/regularization_loss: 0.093652

# ADDITIONS

Object detection is one of the basic issues of computer vision. It shapes the premise of numerous other downstream computer vision assignments, for example, instance segmentation, image captioning, object tracking, pedestrian detection, People counting, face detection, text detection, pose detection, number-plate recognition, Crowd counting, Self-driving cars, Video surveillance, Face detection, Anomaly detection etc.



19K-1377 19K-0248 Section: F

#### CONCLUSION

In summary, we developed a system known as Objectfler allows its users to perform object detection utilizing 2 of the latest Tensor Flow pretrained models. This system also allows users to detect objects

We achieved these features by performing the following steps:

- Cleaning the ever so popular PASCAL VOC2011 dataset [1] so that it can be used for the purpose of training our models.
- Downloading and configuring MobileNet and Efficient Net architectures from Tensor Flow 200 [2] to make them compatible with our environment
  - 3. Training our configured models.
- 4. Testing our models and analyzing their result.

#### CELEKENCES

The following sites/articles were referenced in this article:

- [1] http://host.robots.ox.ac.uk/pascal/VOC/voc2 011/index.html
- [2] https://github.com/tensorflow/models/blob/master/research/object\_detection/g3doc/tf2\_detection\_zoo.md
- [3] arXiv:1511.08458v2