Real-time Food-Object Detection and Localization for Indian Cuisines using Deep Neural Networks

Abhinaav Ramesh
BE Biomedical Engineering
PSG College of Technology
Coimbatore, India
contactabhinaav@gmail.com

Aswath Sivakumar
BE Biomedical Engineering
PSG College of Technology
Coimbatore, India
aswathpro7@gmail.com

Sherly Angel S
BE Biomedical Engineering
PSG College of Technology
Coimbatore, India
Sherlyangel257@gmail.com

2020 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT) | 978-1-7281-8885-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICMLANT50963.2020.9355987

***Abstract*— This paper is aimed at developing an application that automatically detects food objects in real-time scenes and localizes them within the image, which can be used in a standalone or connected application framework. A dataset compiled from multiple online sources was used to train a Single Shot Detector (SSD) configuration. The object detection model and various convolutional network architectures were paired with the Single Shot Detector and the most efficient approach was identified as InceptionV2 convolutional neural network architecture paired with a Single Shot Detector.**

Keywords—Real-time Food-Object Detection, SSD Detector, SSD_InceptionV2, 60 classes, Tensorflow

I. INTRODUCTION

Humans can perceive the three-dimensional structures of objects around us with ease. Researchers in computer vision have been developing mathematical techniques and models for mimicking human perception of vision. There are reliable techniques for computing a partial 3D model of an object or environment from thousands of partially overlapping photographs. Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world by using digital images and deep learning models to accurately identify and classify objects[1].

Early experiments in computer vision that took place in the 1950s, used some of the first neural networks to detect the edges of an object or to sort simple objects into categories like circles and squares. Mobile technology with built-in cameras has eased the process of data collection (images and videos). Processing power and computing technologies have become more affordable and easily accessible. Hardware and Software have been designed especially for computer vision and analysis. Algorithms like convolutional neural networks and recurrent neural networks can take advantage of such hardware and software capabilities.

This paper concentrate on object detection and classification specifically targeting applications in the food industry. Good nutrition is an important part of leading a healthy lifestyle. In the field of nutrition, food industries have been constantly booming with multitudes of gadgets and smartphone applications. A variety of applications are currently present in the market used for nutrient monitoring, recipe search, food delivery, and choosing good restaurants. This paper aims at providing a framework for auto-detecting food scenes, classifying, and localizing objects.

II. LITERATURE SURVEY

Ashutosh Singla et al., 2016 have given a system using a pre-trained GoogleNet classifier model based on a deep convolutional neural network for identification and segregating food images from non-food images and also to distinguish the food item into its class.

They have trained their model on two publicly available datasets and also data gathered from imaging devices and social media. They have further fine-tuned and made improvements to the existing model for achieving 99.2% on food/non-food classification[2].

Kuang-Huei Lee et al., 2018 have provided selected examples of CleanNet (a joint neural embedding network), results on the Food-101 dataset. They have also integrated CleanNet and conventional convolutional neural network classifiers into one framework for image classification learning. In this paper, the difficulties in scalability and classification of data with high noise have been illustrated. A transfer-learning based approach has been applied for the training of the CNN model[3].

Stephen Hoff et al., 2018 have proposed a system that identifies food based on the picture of the user's plate. This system holds an advantage in image recognition and the internal camera of Android phones. They have trained the system using the SSD Mobilenet V1 object detection model. In this paper, they have done two levels of testing on camera variations such as ideal positions, angled camera, distance, and low lighting[4].

Zifeng Wu et al., 2019 have analyzed the architecture ResNet model in terms of the ensemble classifiers and the residual unit. A system which is spatially and performance-wise more efficient has been proposed by them for large networks. They have used a bunch of shallow networks over deeper networks for this and have shown that the shallow network achieves better performance over the latter and also evaluated the impact of using different networks on the performance of semantic image segmentation and shows that these networks, as pre-trained features can boost existing algorithm[5].

Marc Bolaños and Petia Radeva,2016 have proposed a system that performs both localization and recognition of the food item with high precision and reasonable recall levels with only a few bounding boxes. The system has been proven applicable for both conventional and noisy images with considerable accuracy achieved for both[6].

Kiyoharu Aizawa et al., 2019 have proposed a personalized system for the classification of food image recognition in a real-world scenario. To achieve better performance, a weight optimization-based algorithm has been employed. The intra-class diversity and similarity during image recognition are also addressed in this paper[7].

978-1-7281-8885-0/20/$31.00 ©2020 IEEE

Takuya Akiba et al., 2017 in their paper, demonstrate that training the ImageNet dataset on a Resnet-50 DCNN model can take as little as 15 minutes. They achieved with the help of a 1024 Tesla p100 GPU's, training the model for a total of 90 epochs with an image batch size of thirty-two thousand images. They have used several techniques such as a slow start learning rate scheduler along with batch normalization and RMSprop warm-up [8].

Krzysztof Wołk et al., 2019 have proposed a system that uses the Resnet-34 neural network model for the classification of images. They have made of the use of Google's cloud platform along with fast.AI making it easy to train a model. They have trained their model on a dataset containing images of 7000 od polish cars and have managed to get an accuracy of 99.18% [9].

III. DATASET

The dataset consists of a total of 4200 images categorized into 60 classes listed in Table 1 with each class consisting of 70 samples. The data used in the paper is collected from various online resources and labeled manually. The resolution of the images varies between 200 x 200 and 512 x 512 and all images are of the JPEG format. The images considered for the dataset belong strictly to Indian cuisine and the most common dishes were chosen based on consumer popularity across India [12]. Part of the UEC-Food-256 dataset was also used for training a few classes. The UEC Dataset is part of the Food Recognition Research Group. [23]

A. Labeling of Dataset

An open-source graphical annotation tool, label-IMG, was used for manual labeling the images. The tool allows bounding boxes to be graphically drawn on the image to annotate the objects of interest. The coordinates of the bounding boxes in the image are stored using the Pascal VOC standard in an XML file consisting of the image information and object information.

B. Data Augmentation

Data Augmentation is the technique by which additional samples are created using the original images by slightly varying the original image like flipping, zooming, blurring, and mirroring. A total of 4 different functions were used to augment the data which resulted in a total of over 15000 images to be used as data for the model. The augmentations used to increase the dataset include horizontal flip, random scaling of image, random shearing of image, random rotation across the central axis [14].

SNO	CLASS NAME	SNO	CLASS NAME
1	gulab_jamun	31	ven_pongal
2	paniyaram	32	chilly
3	vada	33	halwa
4	idli	34	pickle
5	dhokla	35	juice
6	chicken_tandoori	36	rubber_halwa
7	poori	37	milk_tea
8	samosa	38	uthapam
9	naan	39	soup
10	kati_roll	40	poriyal
11	chappati	41	omlet
12	gravy	42	milk
13	coconut chutney	43	raitha
14	chaat	44	black tea
15	sambar	45	aloo_masala
16	mint_chutney	46	pasta
17	fish fry	47	rice plain
18	lemon slice	48	ketchup
19	brownie	49	cake slice
20	biryani	50	boiled egg
21	dosa	51	pizza
22	chicken	52	cucumber
23	papad	53	tomato
24	tomato chutney	54	mayo
25	ice-cream	55	rice tomato
26	sandwich	56	rice curd
27	bisibelebath	57	rice lemon
28	onion sliced	58	rice fried
29	upma	59	water
30	noodles	60	carrot_halwa

Table1: List of Classes Labelled for Object Detection

IV. OBJECT DETECTION FRAMEWORK

The Object detection approach has two major setups, one using Faster RCNN and others using Single Shot MultiBox Detector (SSD). An R-CNN is a special type of CNN that can locate and detect objects in images: the output is generally a set of bounding boxes that closely classify each object detected, as well as a class output for each detected object. Though faster RCNN could yield the highest accuracy, the hardware requirements to tackle the computing complexity are very high and the training timeline is in days [15]. Any changes made to model configuration will take a longer time to be studied for any impact on the model [20]. In our case for 60 classes, the Faster RCNN was way too complex and required around 3 hours to compute for one batch of size 2.

The SSD configuration is highly preferred for lite mobile applications because of the lighter architecture in the number of parameters in comparison with the faster RCNN, Moreover, the prediction is in a single pass. Object localization and classification are done in a single forward pass of the SSD network. An object detector network that also classifies those detected objects is present. The proposed methodology used has been represented in Fig.1. MultiBox's loss function also combined two critical components that made their way into SSD [9]. Confidence loss measures the confidence level in percentage for an object to be inside a detected bounding box. Categorical cross-entropy is used to compute this loss. Location Loss measures how far away the network's predicted bounding boxes are from the ground truth ones from the training set. L2-Norm is used here. Alpha term refers to the learning parameter balancing the localization loss.

multibox_loss = confidence_loss + alpha * location_loss

For the SSD Configuration chosen, there were multiple CNNs available. This research work includes building and fine-tuning two CNN architectures, MobileNetV2 and InceptionV2. Besides, Resnet50 was also tried and the projected accuracy was less than that of inception or mobile net. The following explains the implementation tried out concerning this paper.

978-1-7281-8885-0/20/$31.00 ©2020 IEEE

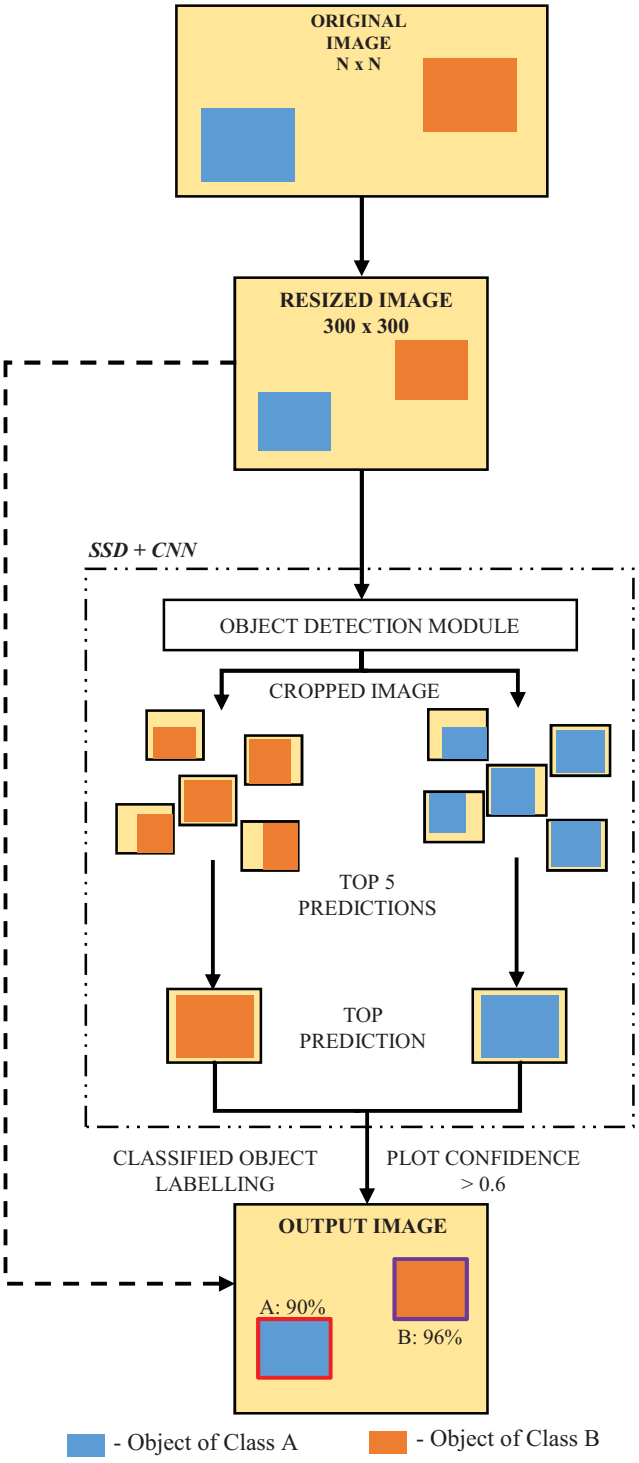

Fig. 1: Framework of Proposed Methodology

A. *MobileNet_V2*

In MobileNetV2, there are two types of blocks. One is the residual block with the stride of 1 and another one is a block with a stride of 2 for downsizing. There are 3 layers for these two types of blocks. The first layer is 1×1 convolution with ReLU6 and the second layer is the depthwise convolution. The third layer is a linear 1×1 convolution. The network computational cost up to 585M Adds, while the model size varies between 1.7M and 6.9M parameters. 16 GB Tesla GPU with a batch size of 96 was used to train the network [16].

B. *Inception_V2*

Inception_V2 is a reduced representational bottleneck framework, which prevents too much loss of information on increased dimensionality reduction in comparison with MobilenetV2. Using smart factorization methods, convolutions can be made more efficient in terms of computational complexity [17]. Complexity is reduced by factorizing 5x5 convolution to two 3x3 convolutions, factorizing convolutions of filter size n x n to a combination of 1 x n and n x 1 convolutions and expansion (make wider instead of deeper) of the filter banks to remove the representational bottleneck.

VI. Training Pipeline for Food Object Detection

In the previous section, various architectures were discussed. The model architecture which provided the highest was SSD_Inception_V2. To rapidly prototype a model for this customized application, the model training was initiated with a pre-trained checkpoint. This pre-trained checkpoint was trained for COCO Dataset, a large-scale open-source object detection, segmentation, and captioning dataset compiled by Microsoft [22]. In this section, the entire pipeline for training a model for a custom dataset is discussed in detail.

A. *Train-Test Split*

After augmentation, the dataset size was increased to contain more than 15000 images with a cumulative of 60 total classes of objects. Since the evaluation of the model is necessary in training, there is a need to set aside evaluation samples, and the number of times each sample will be tested. In this case, each sample tested is an image and its corresponding label XML file. The train to test ratio used to train the current model is 8:2, i.e., 20% of data was used for evaluation. The split was done randomly by sampling 20% of images and their corresponding labels from the master data.

B. *TF Record Creation*

Once the training and testing data were split, the XML files containing the bounding box and class info for each image were combined in a single CSV file for train and one for test data respectively. In the CSV file, crucial fields like image name, object name, and the coordinates of corners of the bounding box containing that object were only retained. Other additional information from the Pascal VOC labeling format were discarded at this stage. To achieve maximum efficiency in reading data, we must serialize the data such that data read will be linear. This technique has also been proven effective in caching data pre-processing. The TFRecord format is a proprietary format for storing a sequence of binary records [18].

The TFRecord is easily parsed and accessed by tensor flow avoiding excess time required in parsing data to trainer session if the data is stored conventionally. A TFRecord file holds a series of records where each record is a byte-string. The images and CSV have been used to create two TFRecords, one for training and one for testing. Once the TFRecords have been generated, the use of the Dataset in the original format is no longer required unless the train test split ratio is modified or data is added or deleted. TFRecord creation step has to be repeated for any afore-mentioned change.

978-1-7281-8885-0/20/$31.00 ©2020 IEEE

C. Label-Map Creation:

Label-Map is a ".pbtxt file" which maps each class name or object name to a non-zero integer key value. It is much similar in the naming convention to JSON. An example of an entry in the Label-Map file is as follows:

item {id: 1
name: 'idli'}

The number of items in the Label-Map file must match the number of classes entered in the model configuration file of the chosen model architecture. In this case, there are 60 items in the label-map file.

D. Model Configuration

The total number of classes is 60. The coder used was a faster RCNN box coder with 10 scaling along both x and y-axis and scaling of 5 across both height and width. The argmax matcher was used with a threshold of 0.5. The number of negatives during matching was maintained to be lower than unmatched. Each Sample will be tried to match forcefully even if the accuracy is low. The Intersection over Union(IOU) is the similarity measure used to evaluate the box localization for the object. An SSD anchor generator with 6 layers, a minimum scale of 20 %, and a maximum scale of 95% with reduced boxes in the lowest layer are used to train the model. The input image tensor is of size 300 x 300. All images were resized to match the input tensor shape. A convolutional box predictor was used with a kernel size 3 with RELU activation. This convolutional layer included L2 regularization.

Inception_V2 architecture was used to extract features from the images. Batch normalization was also used to regularize weights across images in a batch. Batch non-max suppression was also used to boost the efficiency of the model when fed with blind images. The training was monitored by an RMSprop optimizer. The learning rate of the optimizer was dynamically changed along the course of the training. The learning rate was lowered as the epochs increased to fine-tune the learning. An initial learning rate of 0.002 was used. When the training was truncated upon saturation of accuracy, the final learning rate was 0.00075. The total steps planned for this model was 150,000. Since the variance of accuracy was 0.001 for more than 2500 steps, the training was truncated at 80,000 steps. The evaluation of the samples after each epoch was done only once. All the samples in the test data were evaluated. For monitoring purposes, 15 samples were tested and their results were monitored using TensorBoard. The batch size was set concerning the amount of available VRAM. In setting up the environment, the computing resources have been discussed in detail.

E. Training Environment

a. Hardware Environment

Deep Learning requires high computing power. During the initial phases of this research work, online compute engines were used to have Intel Xeon processor and NVidia Tesla GPU. Throughout the process, as we increase the number of data points and classes, the handling of data in the cloud environment grew difficult. We used Google Colaboratory in the initial stages. Later on a high-end system with 6 core 10th Gen Intel i7 Processor, 32 Gb RAM, and 6GB NVidia Graphics with Compute Capability 7.5 were used to train the final model. The final training was performed with a batch size of 24 using the high-end system. This change had no impact on the result of the trained model. In the second setup, the training time was increased when comparing to the Google Colaboratory notebook.

b. Software Environment

Tensor Flow has an API specifically tailored to train and deploy Object Detection models. The API requires Tensorflow-GPU (1.15.0), NumPy (1.17.4), OpenCV (>3.4), and pycocotools. The environment used was Python 3.6 running anaconda with all the above-mentioned packages installed. In addition to these packages, the CUDA toolkit for the GPU and supported Graphics Driver was installed to make use of GPU resources in the training environment. The dlls required for the TensorFlow to utilize the GPU have to be fetchable in the python environment.

F. Model Training

The model was trained for 80000 steps and truncated since the variance in accuracy was less than 0.001 for more than 2000 steps before the stoppage. The configurations for training the model were passed on to the TensorFlow API through a configuration file to the training script of the API. The above-mentioned configurations were included and training was monitored using the TensorBoard visualization tool. The training approximately took less than 6 hours to complete 80000 steps for the final model. Prototyping and fine-tuning the hyperparameters consumed more than 6 hours daily for 3 weeks.

G. Prediction Environment

The trained model was exported to form an inference graph and weights were dumped to a checkpoint file. This weight was used to validate the accuracy of the model to set up a prediction environment. The prediction was quick at the average rate of 14 fps.

VII. Results and Inference

In this research paper, both SSD_MobilnetV2 and SSD_InceptionV2 were trained until the point of convergence. The point of convergence was achieved during training when the total loss (computed for validation data) had a variance of less than $1e^{-3}$ for at least 10 consecutive epochs. The MobilenetV2 architecture converged at 60000 steps and InceptionV2 architecture at 80000 steps. The metrics of both trials are tabulated in Table 2. mAP stands for Mean Average Precision, 50IOU stands for Max_50_class Intersection over Union for Bounding Box and 100IOU stands for Max_100_class Intersection over the union.

The current use case has 60 classes only, so the 100 IOU depicts the IOU score for all classes in the dataset. From the results, it is evident that InceptionV2 is having better accuracy in comparison with MobileNetV2 Architecture for CNN paired with SSD. The SSD used was similar in configuration (as discussed earlier) for both MobileNetV2 and InceptionV2 architectures. Shown in Fig.2 are some of the predictions obtained for test images using the trained SSD_InceptionV2 model.

978-1-7281-8885-0/20/$31.00 ©2020 IEEE

Fig.2: Predictions from the Trained InceptionV2 Model

978-1-7281-8885-0/20/$31.00 ©2020 IEEE

	MobilenetV2	InceptionV2
mAP	0.59	0.738
mAP(Large)	0.632	0.769
mAP(Medium)	0.308	0.575
mAP(Small)	-1	0.612
50IOU	0.909	0.9645
75IOU	0.804	0.8627
Recall	0.524	0.792
Classification Loss	1.98	1.284
Localization Loss	0.293	0.245
Regularization Loss	0.63	0.626
Total Loss	2.903	2.15

Table 2: Result Metrics of MobileNet and Inception Networks for SSD

CONCLUSION

The accuracy of the model expressed through metrics is subjective to the accuracy of labeling the dataset. Using the open-source tool, there were shortcomings in labeling overlapping objects. Thus a clear bounding box containing only the object was not possible in all cases. Comprehending the predictions in blind images, the accuracy of the prediction and localization of the bounding box by the SSD on the Image is far more accurate than it seems from the metrics. The class classification confidence was above 80% for 1900 of 2000 images predicted and above 95% for 1750 of 2000 images tested. During testing, the number of false predictions was in the ratio of 250:1, i.e., 1 false prediction per object for 250 accurate predictions. Comprehending that, the accuracy of the model can be practically estimated at 97.6%.

FUTURE SCOPE FOR THIS RESEARCH WORK

We believe that this paper has the potential to be useful not only for casual users but also in the industry lines where precision and resource management are of the utmost importance. Some of the ways that this paper can be further improved are:

- Deploying a mobile application for portability and ease of real-time operation.
- Adding a calorie counter that can predict the number of calories present in the dish by considering the volume.
- Check if the dishes cooked matches the orders in a hotel kitchen scenario and to check if all items are present in an order (combo) in a hotel.
- Increase the number of classes dynamically (by transfer learning) [19].
- End of production line checker for respective industries.
- Nutrient monitoring for different diet plans (balanced, body buffing, etc.).
- Having a Food log consisting of orders based on their popularity can help the hotel management plan the restocking of raw ingredients.

REFERENCES

[1] Computer Vision: Algorithms and Applications by Richard Szeliski [Published – 2010]

[2] Singla, A., Yuan, L., & Ebrahimi, T. (2016, October). Food/non-food image classification and food categorization using a pre-trained googlenet model. In Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management (pp. 3-11).

[3] Lee, K. H., He, X., Zhang, L., & Yang, L. (2018). Cleannet: Transfer learning for scalable image classifier training with label noise. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5447-5456).

[4] Hoff, Stephen; Jaffurs, Patterson; Enriquez, Michael; and Wilde, Quintin, "Snap-n-Snack: a Food Image Recognition Application" (2018). Computer Science and Engineering Senior Theses. 121.

[5] Wu, Z., Shen, C., & Van Den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. Pattern Recognition, 90, 119-133.

[6] Bolaños, M., & Radeva, P. (2016, December). Simultaneous food localization and recognition. In 2016 23rd International Conference on Pattern Recognition (ICPR) (pp. 3140-3145). IEEE.

[7] Yu, Q., Anzawa, M., Amano, S., & Aizawa, K. (2019). Personalized Food Image Classifier Considering Time-Dependent and Item-Dependent Food Distribution. IEICE Transactions on Information and Systems, 102(11), 2120-2126.

[8] Akiba, T., Suzuki, S., & Fukuda, K. (2017). Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes. arXiv preprint arXiv:1711.04325.

[9] Zawadzka-Gosk, E., Wołk, K., & Czarnowski, W. (2019, April). Deep learning in state-of-the-art image classification exceeding 99% accuracy. In World Conference on Information Systems and Technologies (pp. 946-957). Springer, Cham.

[10] R. Phadnis, J. Mishra and S. Bendale, "Objects Talk - Object Detection and Pattern Tracking Using TensorFlow," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2018, pp. 1216-1219, DOI: 10.1109/ICICCT.2018.8473331.

[11] Vision system development for hybrid robot-gripper to perform manipulation tasks in the food industry for object detection and localization, Yao, Lingjie, 2020, https://hdl.handle.net/10356/140456

[12] Indian Food Online Data - https://www.kaggle.com/cdart99/food20dataset

[13] TensorFlow API used -https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/index.html

[14] Fadaee, Marzieh, Bisazza, Arianna, and Monz, Christof. "Data augmentation for low-resource neural machine translation." arXiv:1705.00440 (2017).

[15] Liu, Wei, et al. "SSD: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[16] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[17] Maeda, Hiroya, et al. "Road damage detection using deep neural networks with images captured through a smartphone." arXiv preprint arXiv:1801.09454 (2018).

[18] Yoon, Heemoon, Sang-Hee Lee, and Mira Park. "TensorFlow with user-friendly Graphical Framework for object detection API." arXiv preprint arXiv:2006.06385 (2020).

[19] Talukdar, Jonti, et al. "Transfer learning for object detection using state-of-the-art deep neural networks." 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN). IEEE, 2018.

[20] Ali, Hashir, et al. "Object Recognition for Dental Instruments Using SSD-MobileNet." 2019 International Conference on Information Science and Communication Technology (ICISCT). IEEE, 2019.

[21] Gupta, M., & Dahlya, D. (2016). Performance Evaluation of Classification Algorithms on Different Datasets. Indian Journal of Science and Technology, 9(40), 1-6.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár (2015), Microsoft COCO: Common Objects in Context

[23] Kawano, Y., & Yanai, K. (2014, September). Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In European Conference on Computer Vision (pp. 3-17). Springer, Cham.

978-1-7281-8885-0/20/$31.00 ©2020 IEEE