SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Deep Learning Object Detector Using a Combination of Convolutional Neural Network (CNN) Architecture (MiniVGGNet) and Classic Object Detection Algorithm

Asmida Ismail[1, 3]*, Siti Anom Ahmad[1,4]*, Azura Che Soh[1], Mohd Khair Hassan[1] and Hazreen Haizi Harith[2]

[1]*Department of Electrical and Electronic Engineering, Faculty of Engineering, Universiti Putra Malaysia, 43400, Serdang, Malaysia*
[2]*Department of Biological and Agricultural Engineering, Faculty of Engineering, Universiti Putra Malaysia, 43400, Serdang, Malaysia*
[3]*Department of Engineering Technology, Faculty of Technical and Vocational, Universiti Pendidikan Sultan Idris, 35900, Tanjung Malim, Malaysia*
[4]*Malaysia Research Institute on Ageing (MyAgeing™), Universiti Putra Malaysia, 43400, Serdang, Malaysia*

ABSTRACT

The object detection system is a computer technology related to image processing and computer vision that detects instances of semantic objects of a certain class in digital images and videos. The system consists of two main processes, which are classification and detection. Once an object instance has been classified and detected, it is possible to obtain further information, including recognizes the specific instance, track the object over an image sequence and extract further information about the object and the scene. This paper presented an analysis performance of deep learning object detector by combining a deep learning Convolutional Neural Network (CNN) for object classification and applies classic object detection algorithms to devise our own deep learning object detector. MiniVGGNet is an architecture network used to train an object classification, and the data used for this purpose was collected from specific indoor environment building. For object detection, sliding windows and image pyramids were used to localize and detect objects at different locations, and non-maxima suppression (NMS) was used to obtain the final bounding box to localize the object location. Based on the experiment result, the percentage of classification accuracy of the network is 80% to 90% and the time

ARTICLE INFO

Article history:
Received: 10 February 2020
Accepted: 13 November 2020
Published: 31 December 2020

DOI: https://doi.org/10.47836/pjst.28.S2.13

E-mail addresses:
asmida@ftv.upsi.edu.my (Asmida Ismail)
sanom@upm.edu.my (Siti Anom Ahmad)
azuracs@upm.edu.my (Azura Che Soh)
khair@upm.edu.my (Mohd Khair Hassan)
hazreen@upm.edu.my (Hazreen Haizi Harith)
* Corresponding author

ISSN: 0128-7680
e-ISSN: 2231-8526

for the system to detect the object is less than 15sec/frame. Experimental results show that there are reasonable and efficient to combine classic object detection method with a deep learning classification approach. The performance of this method can work in some specific use cases and effectively solving the problem of the inaccurate classification and detection of typical features.

Keywords: Classification, convolutional neural network, deep learning, detection, miniVGGNet

INTRODUCTION

In any task, such as image recognition, object detection and neural language processing, deep learning has recently achieved superior performance (Zhao et al., 2017). The object detection system is used to detect and locate particular substances or things by determining the type or class of the object. It is a computer technology related to computer vision and image processing that recognizes instances of semantic objects in digital images and videos. Typically, there are two steps included in an object detector system framework, which is object classification component and object detection component. Object classification and detection are applied in many computer vision fields, including image retrieval, surveillance, security, machine inspection, and automated vehicle systems (Subhi & Ali, 2019, 2018; Subhi et al., 2018, 2019a, 2019b; Vahab et al., 2019)**.**

Classification is a method of separating a target object from all the other categories and making representations more hierarchical, semantic, and informative for visual recognition. In conventional classification methods, the image characteristics are typically carefully hand-crafted to optimise the distinction capability. A variety of hand-crafted designed features have been explored beforehand such as SIFT (Lowe, 2004), the histogram of gradient oriented, HOG (Dalal et al., 2005) and Haar-like (Lienhart & Maydt, 2002). These hand-designed features, however, are not learnt from the essence of data and are subjective to designer interpretation. Due to the variety of appearances, lighting conditions and backgrounds, it is difficult to construct a comprehensive feature descriptor manually in order to define all kinds of objects perfectly. Usually, there is a classifier used to classify an object in an image such as Supported Vector Machine (SVM) (Lewes, 2015), Deformable Part-based Model (DPM) (Felzenszwalb et al., 2010) and AdaBoost (Freund & Schapire, 1997).

Object detection is one of the computer vision's fundamental problems. It classifies each object in an image and uses 2D rectangular bounding box to label the position of the object. There are some classical algorithm for object detection used to detect an object such as Viola-Jones object detection (Wang, 2014), SVM classification with HOG features (Sugiarto et al., 2017), image segmentation and blob analysis (Patil et al., 2015) and image segmentation using background subtraction algorithms (Shaikh et al., 2014). Most of these approaches, however, are standardised and not robust for the various data varieties. By extracting feature descriptors using the traditional vision algorithm, the system needs to process an object in the camera feed for about 30 seconds to 1 minute and just allowing not more than 70% classification accuracy (Lee, 2015). With the advent of deep learning technology, algorithms for image-based object detection such as Faster R-CNN (Ren et al., 2016), SSD (Liu et al., 2016), YOLO (Redmon & Farhadi, 2017), and Mask R-CNN (He et al., 2018) were used to achieve extremely high accuracy in an object detection system. However, this end-to-end object detection algorithm is computational complex and required a powerful GPU for training the data.

This paper will combine a deep learning algorithm for image classification and classic object detection algorithm to devise our own deep learning object detector. CNN architecture was used to train a network for object classification and a combination of sliding windows, image pyramids and NMS was used for object detection purposes.

The paper is structured as follows: the next section will explain the materials and overall methods followed by results analysis and discussions, and the paper will be concluded in the final section.

MATERIALS AND METHODS

Overall System

Figure 1 shows the overall object detector system using a combination of the convolutional neural network (CNN) architecture (miniVGGNet) and classical object detection algorithm. The processes consist of multiple steps, which are the combination of the building image dataset, followed by an object classification algorithm and ended with an object detection method.

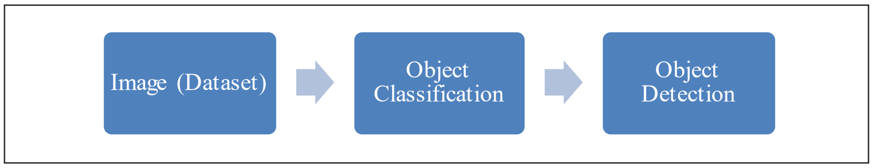


Figure 1. Overall System of an object detector

Building Dataset

Gathering an initial dataset is the first step to creating a deep learning network. The images must be labeled and connected to each others in the same classes. The number of images should be roughly standardised for each class. In this research, data were collected from the real environment field for classification purposes. The data was collected in the fully furnished student residence and was named as an INDOOR dataset.

Figure 2 indicates the random images of INDOOR datasets. The INDOOR dataset consists of 840 images, 32 x 32 x 3 (RGB) in size, resulting in 3072 elements of vector dimensionality. It consists of 14 classes, including bed, chair, cupboard, door, fan, kitchen cabinet, microwave, rack, refrigerator, shower, sink, table study, toilet bowl and washing machine. The dataset contains of 630 training images and 210 test images. From each class, the train and test images are randomly selected.

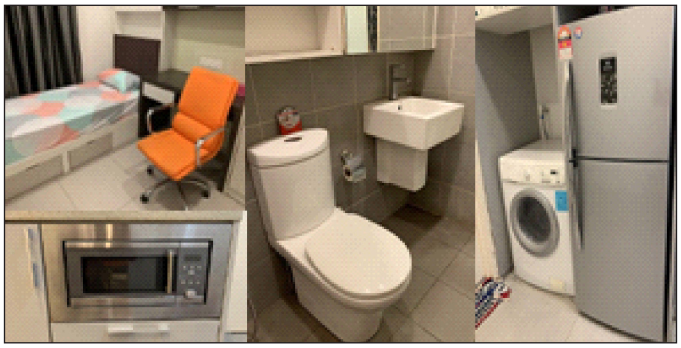


Figure 2. INDOOR dataset

Object Classification

Figure 3 displays the process of building an image classification model using deep learning CNN. The processes are made up of several phases which are pre-processing of images, dataset splitting, network training and tuning hyperparameter value.

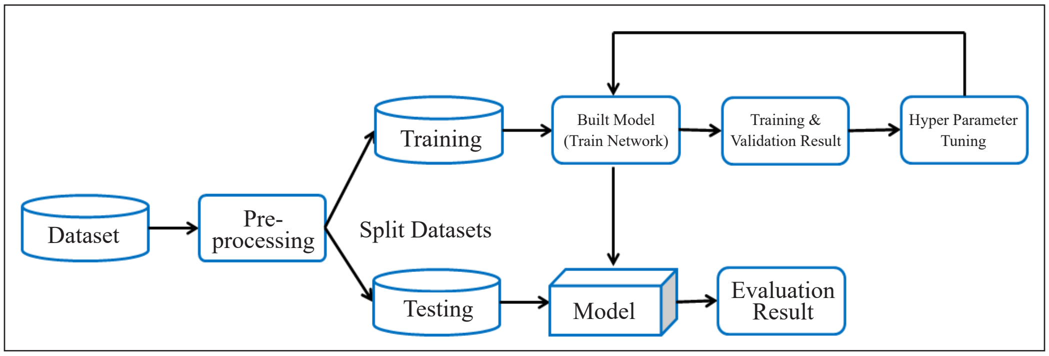


Figure 3. Object classification algorithm – method for building object classification model using deep learning CNN

Pre-processing

Some pre-processing steps could be performed before the model of deep learning is equipped. The images must be scaled to 32 x 32 pixels to ensure the dimensions and aspect ratio is the same. The images provided to the input layer should be square before the model construction. The next step, after the image has been resized is to apply channel ordering. The depth is the number of channels on the image or the number of filters in the layers of CNN architecture. The ordering of dimensions defined the form of the input and used it in the image to learn multi-level features.

Dataset Splitting

The dataset must be divided into two parts: a training set, and a test set. The classifier uses a training set to "learn" what each class looks likes by predicting the input data, and then correcting the data on its own when predictions are incorrect. Once the classifier model has been trained, a test set is used to assess the classifier performance. It is important to ensure that the training set and test set are separate and not overlap with each other.

Network Training

The network will be trained from the images training dataset. The network must learn how to classify each of the classes in the labeled data from the training process. When the network makes an error it learns from the mistakes and improves itself. A simple CNN that accepts an input uses a convolution layer, then an activation layer, followed by fully connected layer, and finally a softmax classifier to obtain classification probabilities. The CNN network architecture used in this paper to create a classification model is miniVGGNet. MiniVGGNet generally consists of two sets of CONV => ACT => CONV => ACT => POOL layers, followed by a set of FC => ACT => FC => SOFTMAX layers. Of these layer types, the only layers containing parameters that are learned during the course of training are CONV and FC. ACT layers are also used in the network to make the architecture explicit. POOL layers are important as CONV and FC because they have a major impact on the spatial dimensions of the image as it passes through CNN.

The detailed architecture of the network is shown in Table 1, where the size of the initial input image is 32 x 32 x 3 and will be training on INDOOR dataset.

Based on Table 1, 32 filters with each of size 3 x 3 will learn on the first two CONV layers. The following layers of CONV will learn 64 filters, with each 3 x 3 in size. Over a 2 x 2 window with a 2 x 2 stage, POOL layers perform max pooling. RELU is an activation element (ACT) used in this architecture.

Table 1
MiniVGGNet architecture design review

Layer Type	Output Size	Filter Size/ Stride
INPUT IMAGE	32 × 32 × 3	
CONV	32 × 32 × 32	3 × 3, K=32
ACT	32 × 32 × 32	
CONV	32 × 32 × 32	3 × 3, K=32
ACT	32 × 32 × 32	
POOL	16 × 16 × 32	2 × 2
DROPOUT	16 × 16 × 32	
CONV	16 × 16 × 64	3 × 3, K=64
ACT	16 × 16 × 64	

Table 1 (*continue*)

Layer Type	Output Size	Filter Size/ Stride
CONV	16 × 16 × 64	3 × 3, K=64
ACT	16 × 16 × 64	
POOL	8 × 8 × 64	2 × 2
DROPOUT	8 × 8 × 64	
FC	512	
ACT	512	
DROPOUT	512	
FC	10	
SOFTMAX	10	

Note. Sizes of output volume are included for each layer, together with convolutional filter size/pool size.

Hyperparameter Tuning

To extrapolate different data patterns, the machine learning model may require different constraints, weights, or learning rates. These measures are called hyperparameters, and these parameters need to be tuned to enhance the model's accuracy and solve the machine learning problem optimally. Hyperparameter tuning uses an ordered list of hyperparameter components - optimal model, which reduces the predefined loss function of independent data. In this paper, the different value of learning rate decay was modified, and batch normalization layers were applied to the networks to compare the network performance.

Based on Figure 4, the coloured boxes indicate the final value that had been choosing for the classification model. The value of 07.5 was chosen as a learning rate decay factor value because it was based on the highest classification accuracy of training and validation results.

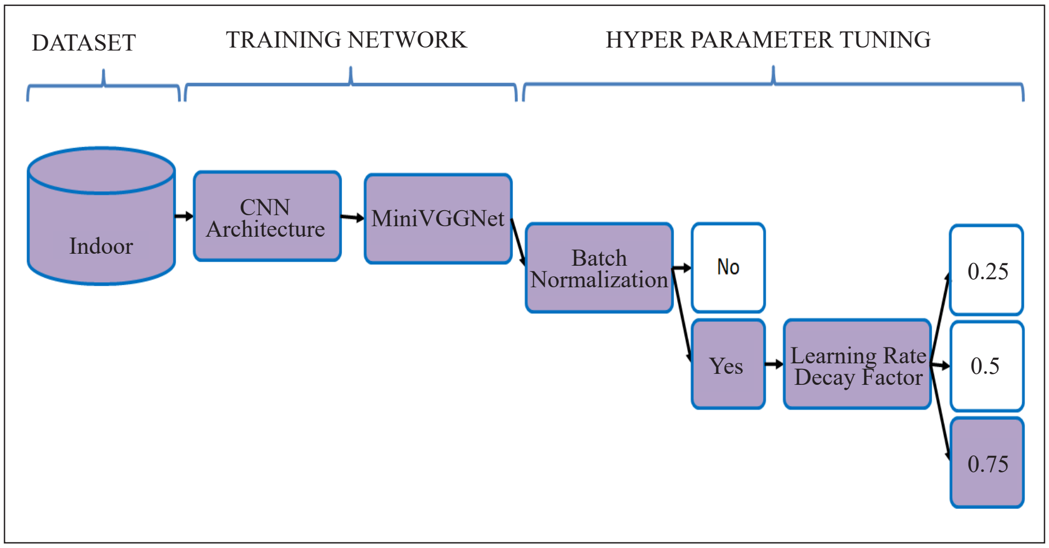


Figure 4. Object classification pipeline

Object Detection

There are four fundamental steps needed when building a deep learning object detector using classic object detection, as shown in Figure 5. The first step is scanning an image at all scales and locations using an image pyramid. An image pyramid is used to reduce (or in some cases, increases) the size of the input image. It is called as an image pyramid because the image was stack from largest to smallest and it looks like a pyramid. The second step is to extract image features by using sliding windows. A sliding window that sits on top of the image will slides from left-to-right and top-to-bottom, classifying each region of interest (ROI) along the way. The sliding window will enable the system to detect precisely where an object is located in the image. The sliding window will runs on each scale of the image pyramid and enabling the system to detect objects that are both closer and farther away from the image. At each stop of the sliding window and image pyramid, the ROI will be extracted and fed into a CNN classification model. That is in the third step where CNN trained model will be used to classify extracted features from each window. When using a sliding window and image pyramid implies to trained classification model, it will report multiple bounding boxes for the same object. By applying non-maxima suppression (NMS) in step 4, the problem of overlapping bounding boxes can be solved, where NMS will keep only the largest confident prediction and obtain only one final bounding box for the object detection.

Figure 6 shows the object detection pipeline of the system. This method of object detection is a better approach, as it uses a combination of a deep learning model for object classification and classical object detection algorithm, which avoids the need to train end to end deep learning object detection frameworks. However, this method has some drawbacks such as slower response and tedious process.

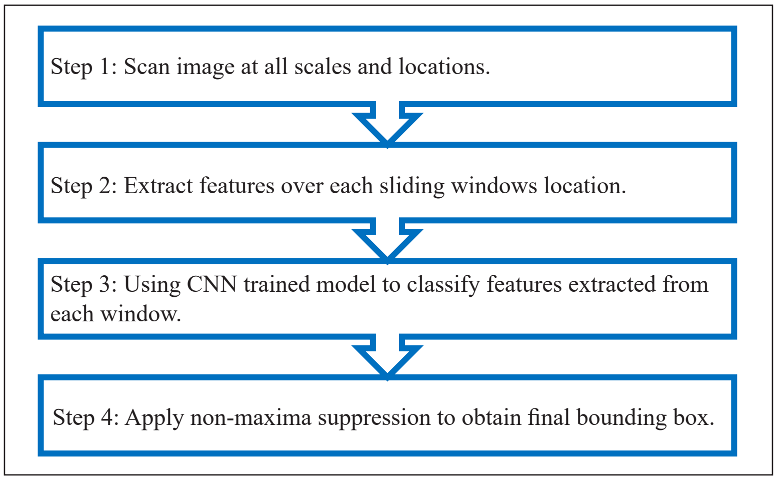

Step 1: Scan image at all scales and locations.
Step 2: Extract features over each sliding windows location.
Step 3: Using CNN trained model to classify features extracted from each window.
Step 4: Apply non-maxima suppression to obtain final bounding box.

Figure 5. The four fundamental steps of traditional object detection

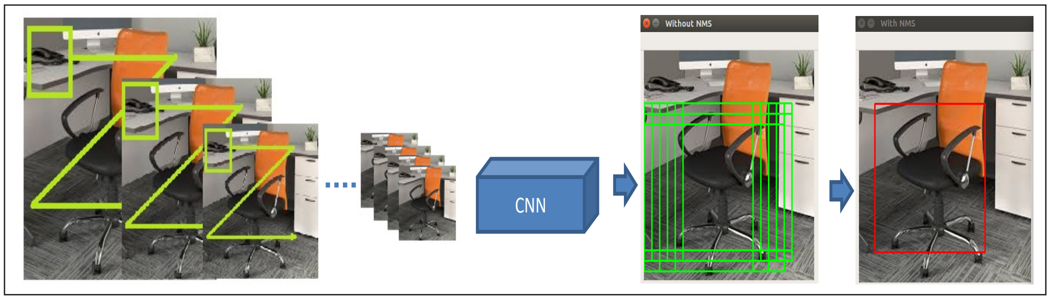
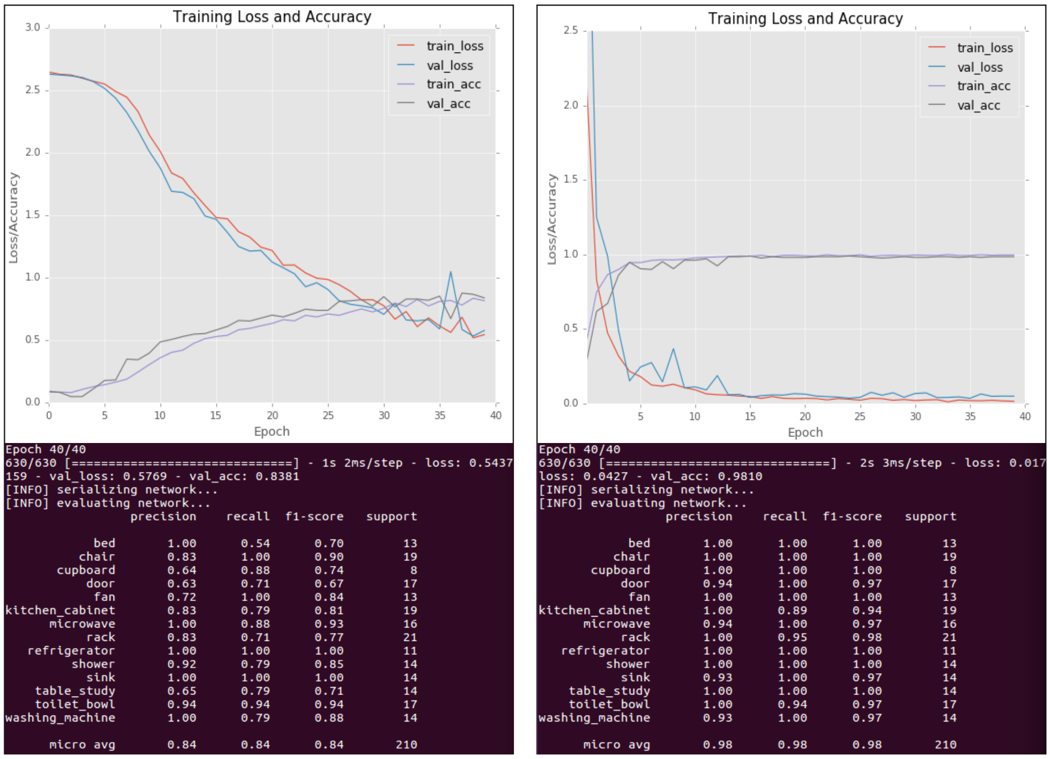


Figure 6. Object detection pipeline

RESULTS AND DISCUSSION

Classification Result

Training results and validation results will be shown in the training loss and accuracy and validation loss and accuracy graph. The python terminal will show the percentage of evaluation accuracy. The network has been trained on INDOOR dataset using miniVGGNet architecture, and the result is shown in the Figure 7.



```
Epoch 40/40
630/630 [==============================] - 1s 2ms/step - loss: 0.5437
159 - val_loss: 0.5769 - val_acc: 0.8381
[INFO] serializing network...
[INFO] evaluating network...
                precision    recall  f1-score   support

            bed       1.00      0.54      0.70        13
          chair       0.83      1.00      0.90        19
       cupboard       0.64      0.88      0.74         8
           door       0.63      0.71      0.67        17
            fan       0.72      1.00      0.84        13
kitchen_cabinet       0.83      0.79      0.81        19
      microwave       1.00      0.88      0.93        16
           rack       0.83      0.71      0.77        21
   refrigerator       1.00      1.00      1.00        11
         shower       0.92      0.79      0.85        14
           sink       1.00      1.00      1.00        14
    table_study       0.65      0.79      0.71        14
    toilet_bowl       0.94      0.94      0.94        17
washing_machine       1.00      0.79      0.88        14

      micro avg       0.84      0.84      0.84       210
```

(a)

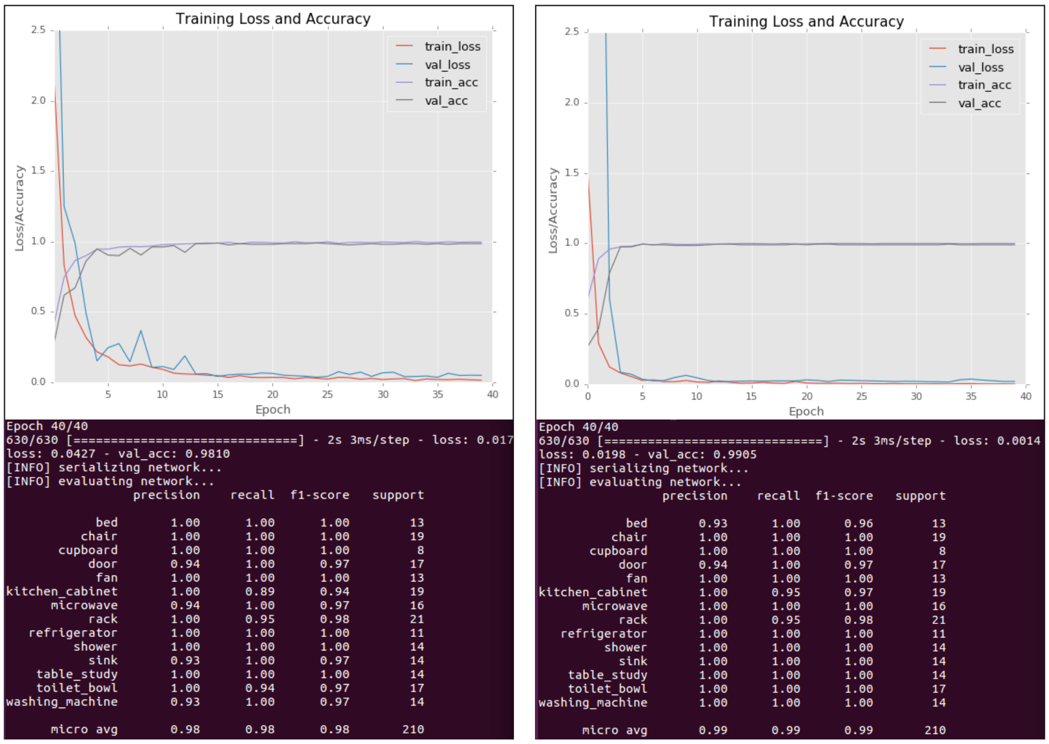
```
Epoch 40/40
630/630 [==============================] - 2s 3ms/step - loss: 0.017
loss: 0.0427 - val_acc: 0.9810
[INFO] serializing network...
[INFO] evaluating network...
                precision    recall  f1-score   support

            bed       1.00      1.00      1.00        13
          chair       1.00      1.00      1.00        19
       cupboard       1.00      1.00      1.00         8
           door       0.94      1.00      0.97        17
            fan       1.00      1.00      1.00        13
kitchen_cabinet       1.00      0.89      0.94        19
      microwave       0.94      1.00      0.97        16
           rack       1.00      0.95      0.98        21
   refrigerator       1.00      1.00      1.00        11
         shower       1.00      1.00      1.00        14
           sink       0.93      1.00      0.97        14
    table_study       1.00      1.00      1.00        14
    toilet_bowl       1.00      0.94      0.97        17
washing_machine       0.93      1.00      0.97        14

      micro avg       0.98      0.98      0.98       210
```

(b)

Figure 7. Object classification result: (a) model trained without batch normalization; (b) model trained with batch normalization

Figure 7 shows the comparison of the object classification result when adding batch normalization layer to the network. The result is shown in Figure 7(a) is the model train without batch normalization, while Figure 7(b) shows the result while the model trained with batch normalization. From the graph plot, there is a positive affect batch normalization has on the training process. The model that implements with batch normalization shows more reliable and the classification accuracy improved by 14% from 84% to 98%. Applying batch normalization helps to reduce overfitting and allows the network to achieve greater classification accuracy because the batch normalization layer are used to normalize the activations of the given input volume before transferring it to the next layer of the network.

Conveniently, the Keras library provides a LearningRateScheduler class that allows us to define a custom learning rate function and then apply it automatically during the training process. By adjusting the learning rate decay factor value, it will help to mitigate the effects of overfitting during the training process. Through epoch-to-epoch modification of the learning rate value, the loss will decrease, the accuracy will increase, and the overall amount of time takes to train a network will decrease. Based on the result shown in Figure 8(a) and Figure 8(b), adding the learning rate decay factor to the network can increase



```
Epoch 40/40
630/630 [==============================] - 2s 3ms/step - loss: 0.017
loss: 0.0427 - val_acc: 0.9810
[INFO] serializing network...
[INFO] evaluating network...
                precision    recall  f1-score   support

           bed       1.00      1.00      1.00        13
         chair       1.00      1.00      1.00        19
      cupboard       1.00      1.00      1.00         8
          door       0.94      1.00      0.97        17
           fan       1.00      1.00      1.00        13
kitchen_cabinet       1.00      0.89      0.94        19
     microwave       0.94      1.00      0.97        16
          rack       1.00      0.95      0.98        21
  refrigerator       1.00      1.00      1.00        11
        shower       1.00      1.00      1.00        14
          sink       0.93      1.00      0.97        14
   table_study       1.00      1.00      1.00        14
   toilet_bowl       1.00      0.94      0.97        17
washing_machine       0.93      1.00      0.97        14

     micro avg       0.98      0.98      0.98       210
```

(a)

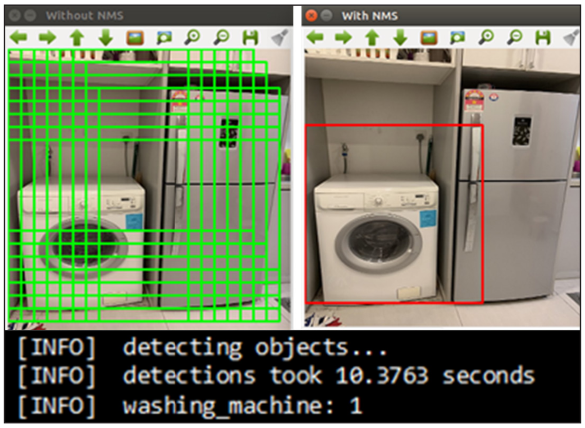
```
Epoch 40/40
630/630 [==============================] - 2s 3ms/step - loss: 0.0014
loss: 0.0198 - val_acc: 0.9905
[INFO] serializing network...
[INFO] evaluating network...
                precision    recall  f1-score   support

           bed       0.93      1.00      0.96        13
         chair       1.00      1.00      1.00        19
      cupboard       1.00      1.00      1.00         8
          door       0.94      1.00      0.97        17
           fan       1.00      1.00      1.00        13
kitchen_cabinet       1.00      0.95      0.97        19
     microwave       1.00      1.00      1.00        16
          rack       1.00      0.95      0.98        21
  refrigerator       1.00      1.00      1.00        11
        shower       1.00      1.00      1.00        14
          sink       1.00      1.00      1.00        14
   table_study       1.00      1.00      1.00        14
   toilet_bowl       1.00      1.00      1.00        17
washing_machine       1.00      1.00      1.00        14

     micro avg       0.99      0.99      0.99       210
```

(b)

Figure 8. Object classification result: (a) model trained without learning rate decay factor; (b) model trained on learning rate decay factor (f=0.75) the classification accuracy of the model and at the same time can reduce overfitting on the graph plot. By adjusting the value of the learning rate decay factor by 0.75 in Figure 8(b), the classification accuracy of the model increase by 1% from 98% to 99%. Adding the learning rate factor value will control the rate in learning rate drops that affect the classification accuracy.

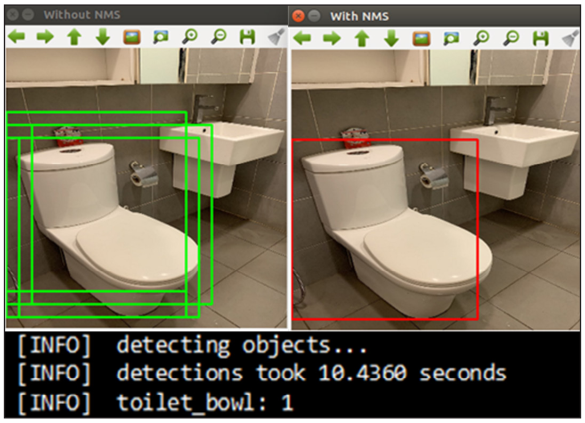
Detection Result

Figure 9 shows the object detection result by applying our deep learning object detector. The figure shows the comparison of object detection by applying NMS and without applying NMS. Results in Figure 9 (a) and (b) show object detector without NMS yields multiple bounding boxes in the image and applying non-maxima suppression (NMS), to remove overlapping bounding boxes, and keeping only the one with the largest probability/ confidence. The information of the detecting objects about time detection and object classes



```
[INFO]  detecting objects...
[INFO]  detections took 10.3763 seconds
[INFO]  washing_machine: 1
```

(a)



```
[INFO]  detecting objects...
[INFO]  detections took 10.4360 seconds
[INFO]  toilet_bowl: 1
```

(b)

Figure 9. Object detection result, comparison between applying NMS and without NMS: (a) Applying our object detector to detect the washing machine in the image; (b) Applying our object detector to detect toilet bowl in the image.

was shown in the python terminal. Based on the result, the time taken for object detection is more than 10 seconds, which means that this object detector system is incredibly slow but this is an acceptable trade-off between accuracy and detection time. This system can detect only one object that has a larger probability and cannot detect multiple objects in one image.

CONCLUSION

This paper presents an analysis performance of deep learning object detector by combining a deep learning Convolutional Neural Network (CNN) for object classifying and applies some of the classic object detection algorithms to devise our own deep learning object detector. MiniVGGNet architecture was used as a classification model and a combination of the image pyramid, sliding window and NMS was used as an object detection algorithm. These approaches have a benefit, where they can treat any deep learning model trained for classification as an object detector.

ACKNOWLEDGEMENTS

The authors wish to express their sincere gratitude to the Control and Signal Processing community Faculty of Engineering, Universiti Putra Malaysia for their support and encouragement during this research. This research was funded through a Putra Graduate Initiative (IPS) UPM grant project code GP-IPS/2018/9606100.

REFERENCES

Dalal, N., Triggs, B., & Europe, D. (2005, June 20-25). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 886-893). San Diego, California. doi:10.1109/CVPR.2005.177

Felzenszwalb, P. F., Society, I. C., Girshick, R. B., Member, S., Mcallester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645. doi: 10.1109/tpami.2009.167

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119-139. doi:10.1006/jcss.1997.1504

He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2018). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(2), 386-397. doi:10.1109/TPAMI.2018.2844175

Lee, A. (2015). *Comparing deep neural networks and traditional vision algorithms in mobile robotics*. Retrieved September 16, 2020, from https://www.cs.swarthmore.edu/~meeden/cs81/f15/papers/Andy.pdf

Lewes, G. H. (2015). Efficient learning machines. In Awad, M., & Khanna, R. (Ed.), *Support vector machines for classification* (pp. 39-66). Berkeley, California: Apress. doi:10.1007/978-1-4302-5990-9_3

Lienhart, R., & Maydt, J. (2002, September 22-25). An extended set of Haar-like features for rapid object detection. In *IEEE International Conference on Image Processing* (pp. 900–903). Rochester, New York. doi:10.1109/ICIP.2002.1038171

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Computer vision - ECCV. In (Ed.), *SSD : Single shot multibox detector* (pp. 21-37). Dordrecht, Netherlands: Springer. doi:10.1007/978-3-319-46448-0_2

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110. doi: 10.1023/b:visi.0000029664.99615.94

Patil, A., Student, M. E., & Dhanvijay, M. (2015). Blob detection technique using image processing for identification of machine printed characters. *Journal of Innovations in Engineering Research and Technology [IJIERT]2*(10), 1–8. doi:10.5281/zenodo.1467487

Redmon, J., & Farhadi, A. (2017, July 21-26). YOLO9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 7263-7271). Honolulu, Hawaii. doi:10.1109/CVPR.2017.690

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transaction on Pattern Analysis and Machine, 39*(6), 1137-1149. doi:10.1109/tpami.2016.2577031

Shaikh, S. H., Saeed, K., & Chaki, N. (2014). SpringerBriefs in computer science. In Zdonik, S., Shekhar, S., Wu, X., Jain, L. C., Padua, D., Shen, X. S., ... & Lee, N. (Ed.), *Moving object detection using background subtraction* (pp. 15-23). Cham, Switzerland: Springer. doi:10.1007/978-3-319-07386-6

Subhi, M. A., Md Ali, S. H., Ismail, A. G., & Othman, M. (2018). Food volume estimation based on stereo image analysis. *IEEE Instrumentation & Measurement Magazine, 21*(6), 36-43. doi:10.1109/mim.2018.8573592

Subhi, M. A., & Ali, S. M. (2018, December 3-6). A deep convolutional neural network for food detection and recognition. In *IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)* (pp. 284-287). Sarawak, Malaysia. doi:10.1109/IECBES.2018.8626720

Subhi, M. A., Ali, S. H., & Mohammed, M. A. (2019a). Vision-based approaches for automatic food recognition and dietary assessment: A survey. *IEEE Access, 7*, 35370-35381. doi:10.1109/access.2019.2904519

Subhi, M. A., Ali, S. H., & Abdulameer, M. (2019b). Deep convolutional networks for food detection and classification. *Journal of Computational and Theoretical Nanoscience, 16*(5-6), 2433-2438. doi:10.1166/jctn.2019.7913

Sugiarto, B., Prakasa, E., Wardoyo, R., Damayanti, R., Krisdianto, Dewi, L. M, ... & Rianto, Y. (2017, November 1-2). Wood identification based on histogram of oriented gradient (HOG) feature and support vector machine (SVM) classifier. In *2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)* (pp. 337-341). Yogyakarta, Indonesia. doi:10.1109/ICITISEE.2017.8285523

Vahab, A., Naik, M. S., Raikar, P. G., & Prasad, S. R. (2019). Applications of object detection system. *International Research Journal of Engineering and Technology*, *6*(4), 4186–4192.

Wang, Y. (2014). An analysis of the Viola-Jones face detection algorithm. *Image Processing On Line, 4*, 128-148. doi:10.5201/ipol.2014.104

Zhao, B., B., Feng, J., Wu, X., & Yan, S. (2017). A survey on deep learning-based fine-grained object classification and semantic segmentation. *International Journal of Automation and Computing, 14*(2), 119-135. doi:10.1007/s11633-017-1053-3