The effect of changing training data on a fixed deep learning detection model

A N Nasser[1], A Alsabbagh[1] and H Géza[1]

[1]Mechatronics Department, Debrecen University, Ótemető utca 2–4, 4028, Debrecen, Hungary

aramoo4@gmail.com

Abstract. Within the lack of accurate data, for some computer vision applications, researchers usually use other pictures collected from different sources for the training. To know the effect of these added data, we compare the detection results of a customized dataset of objects, using the same detection model, while changing the training data fed into the network. For our work, we run the detection on images captured by the Microsoft Kinect sensor after training the network on different combinations of training data. The first part of the training data is captured by the Kinect itself, and the second is collected from several sources from the internet, referred to as collected images. We then change the distribution of these images between training and validation to feed them into the fixed training model. The results prove that this distribution of data can considerably affect training and detection results under the same model parameters. In addition, mixing the captured images with other collected ones can improve these results.

1. Introduction

The recent decade has witnessed a dramatic increase in the ability to classify, localize and detect objects in images. This success is not only the result of the advent of powerful General Purpose Unites (GPUs,) but also designing deep structures of convolutional neural networks, in addition to the availability of large datasets.

In our work, we focus on the detection problem, where the model has to decide what objects are in the image, as well as where do they appear. Our system depends on an object detection API [1], published by Google as an open source code for researchers, after making several changes to it.

The purpose of our work is to know how to tune the dataset to get better results under a fixed detection model. To make work easier, we use images of fruits as they are available and cheap. The detection runs on a picture captured by the Microsoft Kinect sensor. We first train and validate the network on images captured by the Kinect. Second, we replace the training set with collected images and move some of the previously captured images to the validation. Third, we enrich the original dataset of captured images by collected images. Different number of images is used to keep a balance among instances of each class in the three experiments.

2. Literature review

2.1. Datasets

Large datasets, such as Common Objects in COntexts (MS COCO) [2], ImageNet [3] and PASCAL Visual Object Classes (VOC) [4] have enabled significant advances in image classification, object detection, and object segmentation to be achievable. Each of these datasets varies significantly in size, list of labelled categories and types of images.

In this paper, we will focus on the MS COCO dataset, which is used in the pre-trained model. MS COCO is designed for the detection and segmentation of objects occurring in their natural context, with 91 categories, and a total of 2.5 million labelled instances in 328000 images. While MS COCO has fewer categories than ImageNet, it has more instances per category. In comparison to PASCAL VOC, MS COCO has both more categories and instances [2].

2.2. Classification Models

One of the most successful and remarkable algorithms that won the 2012 ImageNet Large Scale Visual Recognition Competition, ILSVRC, is the AlexNet [5]. This was the first successfully applied convolutional neural network at that time, which has changed the future of computer vision algorithms.

In 2014, there were two remarkable models, GoogleNet, also known as Inception V1[6], which won the competition, and VGGNet [7], which was ranked second in the contest but first in the localization task. VGGNet has 19 layers and 7.3% top-5 error rate in classification performance while GoogleNet has 22 layers with 6.7%. It is noteworthy to mention that the number of parameters used in GoogleNet, 5 million, is 12 times less than the ones used in AlexNet, 60 million, yet deeper and more efficient. It also introduced the concept of Inception model, that apply several filters of different sizes on the input from the previous layer producing not only deeper, yet wider layers without significant performance penalty [6]. To reduce dimensionality and remove computational bottlenecks, 1x1 convolutional layers followed typically by the rectified linear activation were added to the inception model.

In [8], the authors argue that adding the Batch Normalization to a state-of-the-art classification model can significantly increase its learning speed. Although, in some publications, this paper is referred to as Inception v2, such as in [10], it was not officially introduced as Inception v2.

The inception modules v2 and v3 were officially introduced in one paper [9]. In this paper, the authors added to [8] and replaced the 5x5 convolution by two 3x3 convolution operations after noticing that the 5x5 convolution is 2.78 times more expensive with the same number of filters. This architecture and the naïve Inception model are explained in figures 2 and 1 respectively.

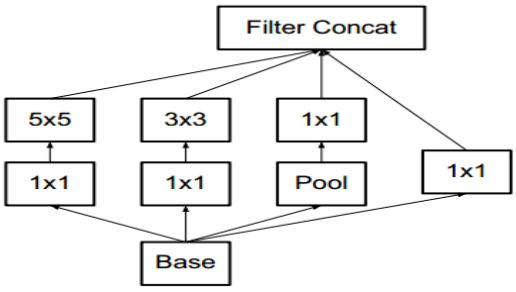

Figure 1. Original Inception module as described in [9]

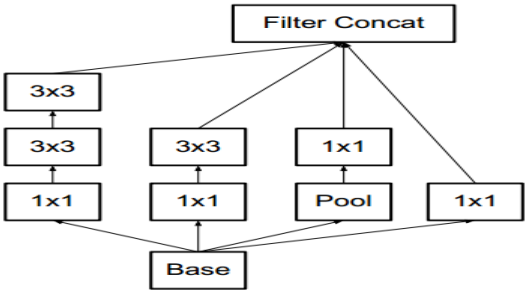

Figure 2. Inception modules where each 5 x 5 convolution is replaced by two 3 x 3 convolutions [9]

2.3. Detection models

For the detection problem, it is not enough to know whether an image contains an object or not. The detection model, however, has to know where the instances of the detected objects appear in the image.

Several detection models are existed in literature to solve this problem. Some of them use a sliding window, such as OverFeat [11]. Other models deal with the entire image during training and test time like the You Only Look Once (YOLO) [12] [13].

Other detection models run the classification on parts of an image proposed by a first stage region proposal process. The most common models that use this algorithm are the R-CNN [14] and Fast-RCNN [15], where the Selective Search algorithm [16] is used to predict region proposals. An updated version of [14] and [15] is the Faster-RCNN [17] had introduced a Region Proposal Network (PRN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals and improving the model's speed and accuracy. [17] also introduced anchor boxes the solve the multi-scale and size problem.

3. Detection methodology

3.1. Transfer Learning

Training models from scratch is time and resource consuming. Some models need days and maybe weeks to converge. Because of this, researchers usually benefit from others' previously trained models in order to initialize their models.

To explore the speed/accuracy trade-off of some of the modern detection systems, researchers at Google had published an object detection API, as an open source code [1]. In this module, the authors tried to apply the meta-architectures of object detection, described in [1], to conduct their experiments and comparisons. In our work, we depend and build on this API to train the detection model on our own dataset. The result of this training is a frozen inference graph, that can be used for the test purposes.

3.2. Used model's parameters

For the detection, we choose a detector Faster-RCNN with Inception v2 as its feature extractor. This model was pre-trained on COCO dataset, where it achieved 28 mean average precision (mAP) detector performance on a subset of the COCO validation set and a running time of 58 (milliseconds) per 600x600 image (including all pre and post-processing).

We choose this model because its accuracy and speed are suitable for our application. L2 regularizer is used in addition to the Truncated normal as the model's initializer. For optimization, we use the Momentum optimizer. We start the training with an initial learning rate of 0.0002, then we make it smaller to be 0.00002 after 5000 steps. The learning rate is set to 0.000002 if the learning exceeds 10000 steps.

3.3. Customized dataset

We created our own datasets to train a fixed model on. To run the first experiment, we captured 160 images using the Kinect and distribute them as 142 training and 18 validation images. Then, we collected 425 pictures from [18], as well as several websites, such as Adobe Stock and Shutter Stock, while keeping 72 images taken by the Kinect. For this experiment, we run the training on the collected images and validate the captured. For the last experiment, we used 385 collected images and 37 captured ones to train the network. 46 collected and 36 captured images were used for the validation process.

4. Results

The result of running the detection after training on images purely captured by the Kinect is shown in figure 3. As can be seen, there are three misclassified classes, in addition to the low probability for the detected objects. In addition, as we notice in figure 4, the training process takes a lot of time and the model is unstable. These results prove that the limited captured images are not enough for the training and more data need to be fed into the model.

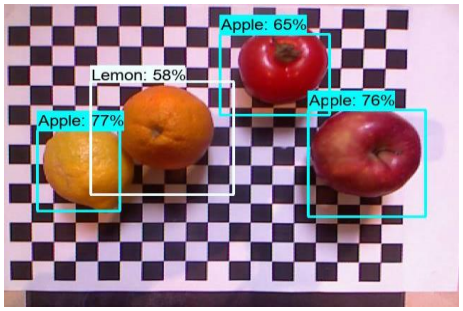

Figure 3. Training and validating on images captured by the Kinect, (limited number of images)

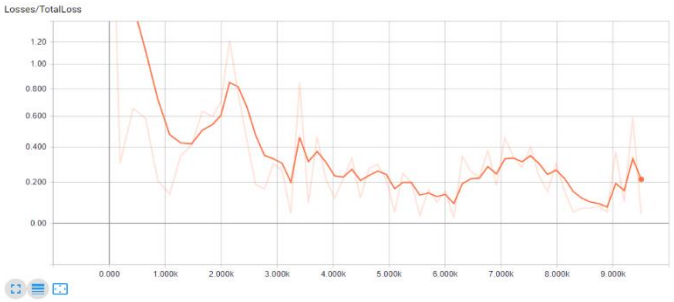

Figure 4. Training process when the model is trained and evaluated on captured.

In the second experiment, we change the training data with images collected from different resources as illustrated in 3.3. We can notice, in figure 5, that the detection results are improved, but still have two misclassified classes. Additionally, the training processed is improved but still showing instability between steps 6000 and 8000.

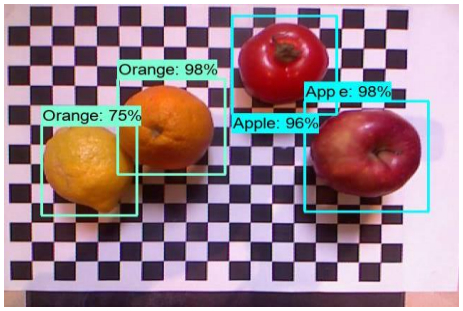

Figure 5. Training on collected images and validating on captured

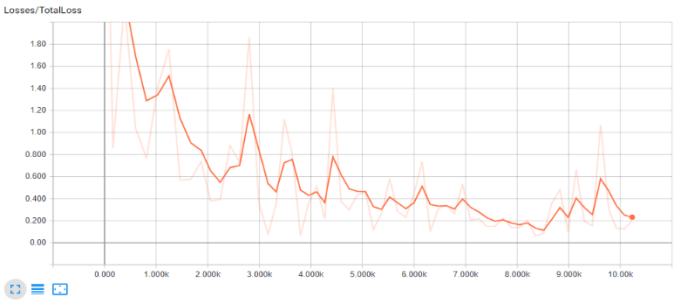

Figure 6. Training process when the model is trained on collected images and evaluated on captured

Finally, figure 7 shows more accurate detection results by only manipulating the dataset. It can be noticed in figure 8 that the training process becomes relatively stable after being unstable for the first 5000 steps.

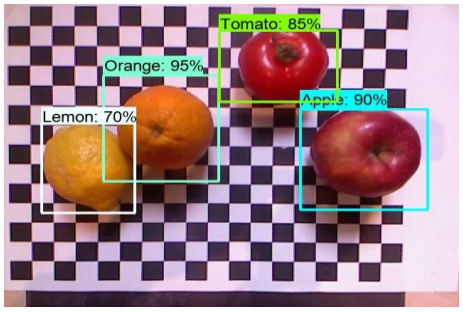

Figure 7. Training and validating on mixed images.

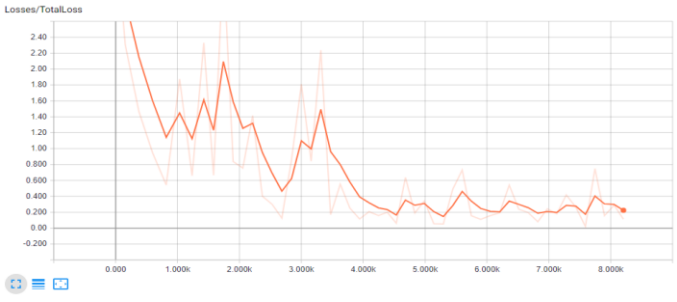

Figure 8. Training process when the model is trained and validated on mixed images.

5. Conclusion

In conclusion, this work shows the importance of choosing the right dataset for the training and validation processes under limited training data. We show the detection results of three different situations where only the dataset is changing while maintaining the model's parameters fixed. As a

result, we see that when we have a lack of information, the best way is to use mixed images to train and validate the network.

Acknowledgment
The work/publication is supported by the EFOP-3.6.1-16-2016-00022 project. The project is co-financed by the European Union and the European Social Fund.

References

[1] Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S and Murphy K 2017 Speed/accuracy trade-offs for modern convolutional object detectors *Proceedings of the IEEE conference on computer vision and pattern recognition* 7310-7311

[2] Lin T Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P and Zitnick CL 2014 September. Microsoft coco: Common objects in context *Springer, Cham* (pp. 740-755)

[3] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M and Berg A C 2015 Imagenet large scale visual recognition challenge *International journal of computer vision* **115(3)** 211-252

[4] Everingham M, Van Gool L, Williams C K, Winn J and Zisserman A 2010 The pascal visual object classes (voc) challenge *International journal of computer vision* **88(2)** 303-338

[5] Krizhevsky A, Sutskever I and Hinton G H 2012 Imagenet classification with deep convolutional neural networks *Advances in neural information processing systems* 1097-1105

[6] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A 2015 Going deeper with convolutions *Proceedings of the IEEE conference on computer vision and pattern recognition* 1-9

[7] Simonyan K and Zisserman A 2014 Very deep convolutional networks for large-scale image recognition *preprint* arXiv/1409.1556

[8] Ioffe S and Szegedy C 2015.Batch normalization: Accelerating deep network training by reducing internal covariate shift *preprint* arXiv/1502.03167

[9] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016

[10] Szegedy C, Ioffe S, Vanhoucke V and Alemi A A 2017 Inception-v4, inception-resnet and the impact of residual connections on learning *Thirty-First AAAI Conference on Artificial Intelligence*

[11] Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R and LeCun Y 2013 Overfeat: Integrated recognition, localization and detection using convolutional networks *preprint* arXiv:1312.6229

[12] Redmon J, Divvala S, Girshick R and Farhadi A 2016 You only look once: Unified, real-time object detection *Proceedings of the IEEE conference on computer vision and pattern recognition* 779-788

[13] Redmon J and Farhadi A 2017 YOLO9000: better, faster, stronger *Proceedings of the IEEE conference on computer vision and pattern recognition* 7263-7271

[14] Girshick R. Donahue J, Darrell T and Malik J, 2014 Rich feature hierarchies for accurate object detection and semantic segmentation *Proceedings of the IEEE conference on computer vision and pattern recognition* 580-587

[15] Girshick R 2015 Fast R-CNN object detection with Caffe *Microsoft Research*

[16] Uijlings J R, Van De Sande K E, Gevers T and Smeulders AW 2013 Selective search for object recognition *International journal of computer vision* **104(2)** 154-171.

[17] Ren S, He K, Girshick R and Sun J 2015 Faster r-cnn: Towards real-time object detection with region proposal networks *Advances in neural information processing systems* 91-99

[18] Hussain I, He Q and Chen Z 2018 Automatic fruit recognition based on DCNN for commercial source trance system