Easily Trainable Neural Network Using TransferLearning

Varghese P Kuruvilla[1]
Department of Electronics and Communication Engineering,Sir M. Visvesvaraya Institute of Technology Bengaluru, India

V N Naveen Raju[2]
Department of Electronics and Communication Engineering,Sir M. Visvesvaraya Institute of Technology Bengaluru, India

Vishnu Sodishetty[3]
Department of Electronics and Communication Engineering, Sir M. Visvesvaraya Institute of Technology Bengaluru, India

Phaninder Ravi Parimi[4]
Department of Electronics and Communication Engineering,Sir M. Visvesvaraya Institute of Technology Bengaluru, India phaninder

---

***Abstract*—**Convolutional neural networks are growing increas-ingly popular in the field of computer vision. One of the most common problems encountered when using convolutional neural networks for computer-vision tasks is that of transfer learning, where a pre-trained model should be adapted to the specific task at hand. This paper investigates the performance of the Tiny YOLOv3 neural network for transfer learning. Specifically, we aim to minimize the time taken for the pre-trained network to converge by varying the number of trainable layers and the number of images used for training. For this experiment, Tiny YOLOv3 was used to detect a specific window of a building after it was trained on a large and varied dataset of real-world images of windows. The resulting mean average precision scores were compared and it was found that even for a shallow networklike Tiny YOLOv3, transfer learning greatly improved the mean average precision scores and minimized the convergence time. The neural network was trained using an Nvidia Tesla K80 GPU and the model was deployed on the Nvidia Jetson TX2 embedded platform. Using transfer learning we obtained an 83% decrease in the time taken to achieve the required mean average precision. Our work could be applied in scenarios where the neural network needs to be retrained quickly with a limited number of training samples.

***Index Terms*—Transfer learning, TinyYOLOv3, Mean AveragePrecision**

I. INTRODUCTION

Traditionally, convolutional neural networks were used for object classification. However, in recent years, convolutional neural networks have been adapted for object detection tasks where localization is required in addition to classification. A typical convolutional neural network achieves this by using alternating convolutional layers for feature extraction followed by max-pooling or average pooling layers to reduce the number of parameters in the network. However, training a convolutional neural network from scratch using randomly initialized weights is impractical, as it requires considerable computational resources and takes considerable amount of time. To overcome this problem, most neural networks are initially trained on a large and varied dataset and the pre- trained weights are used as a starting point for the specific object detection task at hand.

The rest of the paper is organized as follows. In the next section, we give an overview of convolutional neural networks and transfer learning. In section III, we briefly survey some of the works which have been carried out on transfer learning. Section IV talks about the methodology used and contains the experimental results obtained. In Section V, we analyze our results and list some of the potential applications of our work. Finally, we conclude the paper in Section VI.

II. BACKGROUND

A. Convolutional Neural Networks

Convolutional neural networks have been used for computer vision-based tasks from around 1998. However, with the introduction of AlexNet [1] in 2012 which harnessed the power of GPUs during training, several novel architecturesfor object recognition have been proposed and implemented.A convolutional neural network generally used for object detection tasks is shown in Fig. 1.

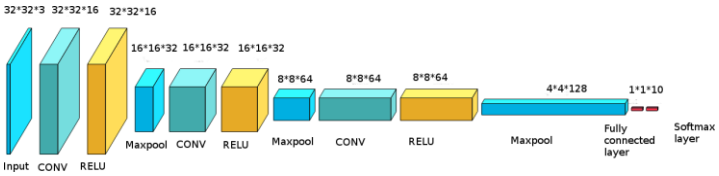

Fig. 1. Architecture of a typical convolutional neural network

The steps followed in object detection are as follows: The input image is fed into the first convolutional layer where the image is convolved with a filter of appropriate size using nec- essary stride and padding. The activation map output from the convolution layer is passed through a pooling layer to reduce the computational complexity of the model. The convolutional and downsampling (pooling) layers were initially introduced by Kunihiko Fukushima in [2]. In a typical neural network, several convolutional and pooling layers are stacked together to extract features from the image. Finally, the activation map of the last layer is flattened and passed as the input to a fully connected layer which is responsible for predicting the probability associated with each class. Using an appropriate loss function, the error is computed and is backpropagated to update the weights and bias values.

B. Tiny YOLOv3

We used the Tiny YOLOv3 neural network for our exper- iment. It is a state of the art object detection model and has achieved a throughput of 30 frames per second on the Nvidia Pascal Titan X GPU. YOLO (You only look once) worksby dividing the input image into a grid of 26 by 26 cells. Each cell predicts a certain number of bounding boxes along with a confidence score that represents the probability thatthe bounding box encloses an object. The architecture of Tiny YOLOv3 is clearly explained in [3].

C. Transfer Learning

A typical neural network has a very large number of parameters. YOLO has approximately 50,000 parameters and training the network from scratch would take several days. Transfer learning is used to avoid this problem. Transfer learning refers to a machine learning method where a model initially developed for a particular task is used as a starting point for some other specific task.This concept was initially introduced in the NIPS-95 workshop ”Learning to Learn”. A detailed explanation of transfer learning is given in [4]. Thisis the focus of this paper.

III. RELATED WORKS

The concept of Multitask learning and transfer learning was initially introduced in [5], where it was postulated that learning several tasks in parallel can be beneficial, as the features learnt in one task could help the neural network perform better in other similar tasks. This work was improved upon in [6] where it was observed that almost all tasks share certain invariances which neural networks could potentially learn and transfer across different tasks. The work in [7] investigates the structural learning problem where predictive functional structures were learnt by simultaneously considering multiple prediction problems. The paper proposed a semi-supervised learning approach to achieve the same. Transfer learning is closely linked with the generality and specificity of features in a neural network. The work carried out in [8] attempts to quantify the degree of generality and specificity associated with each layer of a neural network. It observed that the general features are learnt in the first layers while the last layers learn more specific features. It attempts to find outthe layer or layers in the neural network where this transition from general to specific features occur. This knowledge can be used to optimize the transfer learning process. More recent works such as [9] try to reduce the model selection and hyperparameter tuning that is usually required when using traditional transfer learning approaches. The work in [9] proposes a model called Easy TL (Transfer Learning) which exploits intra-domain structures to learn features.

IV. METHODOLOGY

This section outlines the methodology we use for our experiment. The aim of investigating various transfer learning approaches is to train a neural network in a limited amount of time to detect the window structure shown in Fig. 2. The neural network used for the detection task was Tiny YOLOv3. Tiny YOLOv3 was used since the model had to be deployed on a drone and carry out detections in real-time with reasonable accuracy. The neural network was trained using an Nvidia Tesla K80 GPU following which it was deployed on an Nvidia Jetson TX2 which was mounted on a drone. The experiments conducted are summarized in the following two sections

Fig. 2. Window structure to be detected

A. Model trained using ImageNet weights

The authors of Tiny YOLOv3 provide convolutional weights for the darknet network that are pre-trained on ImageNet. TinyYOLOv3 is built based on the darknet framework which it uses for feature extraction. When these convolutional weights are used for training, the additional weights specific to Tiny YOLOv3 are randomly initialized. Various trials were con- ducted where different layers of the network were frozen and the corresponding mean average precision (mAP) scores

were recorded. The results of these trials are summarized in Tab. I.

B. Model trained using COCO weights

The authors also provide yolov3-tiny.weights which contain weights for the entire Tiny YOLOv3 network trained on the COCO dataset, after being initialized with the pre-trained ImageNet weights. Weights corresponding to the first 15 layers

TABLE I

PERFORMANCE OF THE MODEL TRAINED USING IMAGENET WEIGHTS

Iterations	**Mean Average Precision**
Number of layers frozen = 0	
1000	35
1500	67
1800	92
2000	91
Number of layers frozen = 13	
1000	0.4
1500	5
1800	3
2000	2.98
Number of layers frozen = 20	
1000	0.7
1500	0.7
1800	0.7
2000	0.8

Were extracted and the model was trained for the window detection task. The results are summarized in Tab. II.

TABLE II

PERFORMANCE OF THE MODEL TRAINED USING COCO WEIGHTS

Iterations	**Mean Average Precision**
Number of layers frozen = 0	
1000	89
1500	96
1800	96.4
2000	98
Number of layers frozen = 13	
1000	60
1500	77
1800	80
2000	79
Number of layers frozen = 20	
1000	4
1500	23
1800	23
2000	27

C. Transfer Learning

The method that we followed to further improve the perfor- mance of the detector is as follows: Initially, the network is trained on a large number of images of real-world windows, these weights were then used as a starting point for training the detector to detect the specific type of window shown in Fig. 2. We made use of the Google Open Images Dataset which contains approximately 55,000 images of windows split into train, test and validation sets.

When all 50,000 images were initially used for training the network, it failed to converge. Upon further investigation, it was found that several images like the one shown in Fig. 3 contained annotations that could potentially confuse the detector. Furthermore, the dataset contained windows of different sizes and shapes which does not help our specific case. To solve the problem, we combed through the dataset manually in order to extract 2000 images of reasonably annotated windows.

Even after extracting windows of roughly similar shapes and sizes, the network failed to achieve a satisfactory mAP score

when trained on the 2000 window images using the pre-trained COCO weights as a starting point. After various trials in which different number of layers were frozen, the best mAP score obtained was around 60 percent after 1800 iterations. However,as these weights are going to be used for transfer learning, the mAP score achieved on the Open Images dataset is not of great importance. Using these weights as a starting point, transfer learning was carried out and the results obtained are tabulated in Tab. III.

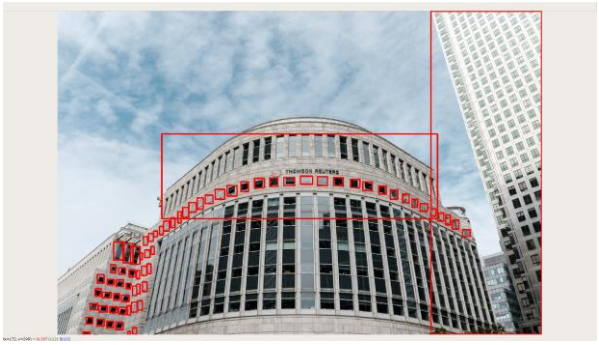

Fig. 3. Example of a bad annotation that could bring down the performance of the detector

TABLE III

PERFORMANCE OF THE PRE-TRAINED MODEL

Iterations	Mean Average Precision
Number of layers frozen = 0	
100	95
200	98.1
300	98.3
400	98.2
Number of layers frozen = 13	
100	93.08
200	94.81
300	96.8
400	95.1
Number of layers frozen = 20	
100	66.9
200	69.7
300	72.2
400	72.8

Fig. 4 is a plot of the mAP score versus the number of iterations for the models trained using ImageNet weights, COCO weights and the model pretrained on real world windowimages.

V. RESULTS AND DISCUSSIONS

From Tab. I, we can observe that the performance of the model trained using the pre-trained ImageNet weights with none of the layers frozen is reasonable, however, the performance of the model drops drastically when a few of its layers are frozen. This could imply that the pre-trained ImageNet weights have not captured all of the low-level features required for the window detection task. Tab. II shows a relative increase in the performance of the model trained using COCO weights.This increase in performance is because the authors obtained

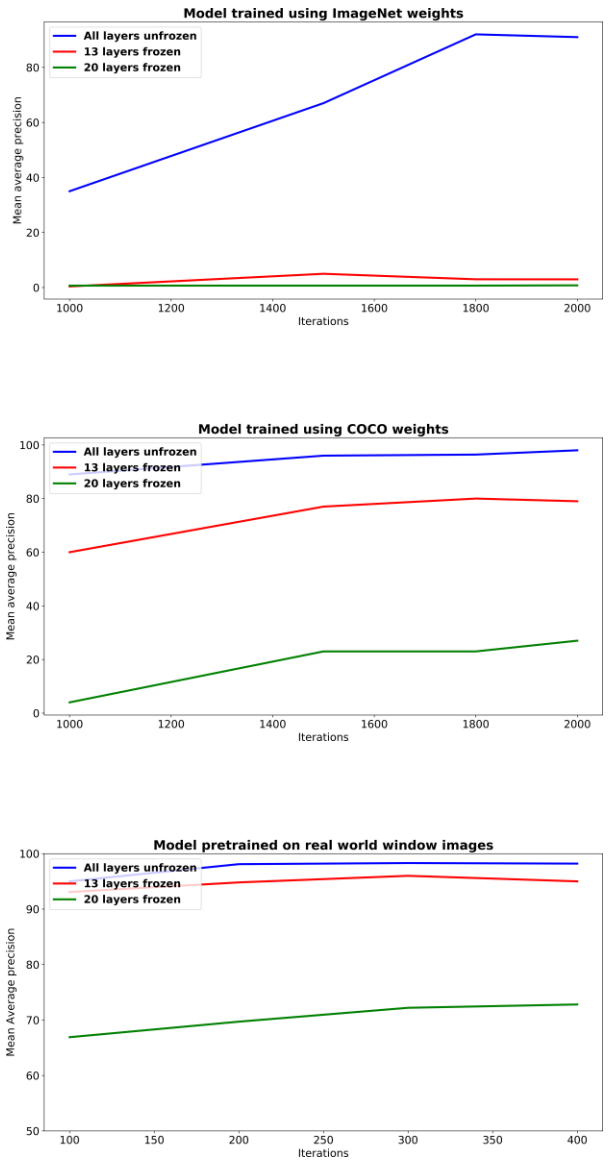

Fig. 4. Comparison of the mAP scores obtained using different models

VI. CONCLUSIONS:

The pre-trained COCO weights by training the entire network on the COCO dataset after loading it with the ImageNet weights. It is also observed that freezing a large number of layers for a shallow network such as Tiny YOLOv3 leads to a degradation in its performance. One of the drawbacks of using COCO weights is that the network requires a large number of iterations to produce a reasonable mAP score.

Further from Fig. 4, it is observed that transfer learning greatly improves the performance of the detector, even on an almost featureless object like a window. Using the same testing and training dataset, the model achieves a mAP score of 98 percent after only 300 iterations. Even after freezing 13 layers of the network, it achieves a mAP score of 96 percent after 300 iterations. A slight degradation in performance can be seen only after 20 layers of the network were frozen. Along with an increase in the mAP score, the time taken for the network to converge greatly decreased as a result of transfer learning.

We demonstrate that transfer learning can greatly aid in improving the performance of object detectors, even if the objects in question are almost featureless. Transfer learning can be used in scenarios where there is a scarcity in the amount of data available for the object detection task. If transfer learning is performed on larger models like YOLOv3, the network would learn more general features and could thereafter be used for a wide variety of object detection tasks.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," Neural Information Processing Systems, vol. 25, 01 2012.

[2] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biological Cybernetics, vol. 36, pp. 193–202, 1980.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[4] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, pp. 1345–1359, Oct 2010.

[5] R. Caruana, "Multitask learning," Machine Learning, vol. 28, pp. 41–75, Jul 1997.

[6] S. Thrun and L. Pratt, eds., Learning to Learn. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[7] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," J. Mach. Learn. Res., vol. 6, pp. 1817–1853, Dec. 2005.

[8] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks," in Advances in Neural Information Processing Systems 27 (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 3320–3328, Curran Asso- ciates, Inc., 2014.

[9] J. Wang, Y. Chen, H. Yu, M. Huang, and Q. Yang, "Easy transfer learning by exploiting intra-domain structures," 2019.