International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

Real-Time Object Detection using TensorFlow

Priyal Jawale, Hitiksha Patel Chaudhary[2], Nivedita Rajput[3]

[1]Priyal Jawale, SNDT Women's University, Mumbai, India
[2]Hitiksha Patel Chaudhary, SNDT Women's University, Mumbai, India
[3]Nivedita Rajput, SNDT Women's University, Mumbai, India
[4]Prof. Sanjay Pawar, Senior IEEE Member, Usha Mittal Institute of Technology, SNDT Women's University, Mumbai, Maharashtra, India

---

Abstract - *In recent years, deep learning has been used in image classification, object tracking, action recognition and scene labeling. Traditionally, Image Processing techniques were used to solve any Computer Vision problems occurred in an artificial intelligence system. However, in real-time identification, image processing cannot be used. This is where Deep Learning concepts are applied. We built a simple Convolutional Neural Network for object detection. The model is trained and multiple test cases are implemented in the TensorFlow environment so as to obtain accurate results*

Key Words*: *Image processing; artificial intelligence; convolutional neural network; TensorFlow

1. INTRODUCTION

The applications and widespread use of machine learning algorithms have made a significant change in the way we perceive computer vision problems. With the introduction of deep learning into the field of image classification, the dynamics of real-time object detection have faced a great impact.

In deep learning, the mapping is done by using representation-learning algorithms. These representations are expressed in terms of other, simpler representations. In other words, a deep learning system can represent the concept of an image for an object by combining simpler concepts, such as points and lines, which are in turn defined in terms of edges. By using a variety of algorithms, a benchmarking dataset and correct labeling packages a system can be trained to achieve the desired output. A fundamental aspect of deep learning in image classification is the use of Convolutional architectures.

The model is trained to detect objects in real-time. This can be best achieved through a universal and open source library-TensorFlow (TF). Within the TF environment, multiple algorithms can be used for a wide range of datasets. In this paper, we have made use of the CIFAR-10 dataset, which comprises of 5 batches each containing common objects seen on a daily basis.

2. BASIC CNN COMPONENTS

Convolutional neural network layer consists of three types of layers, namely convolutional layer, pooling layer, and fully connected layer.

2.1 Convolutional Layer

The aim of CNN is to learn feature representations of the inputs. As shown in the below image (Fig. 1), Convolutional layer has several feature maps and it is the first layer from which features are extracted. Each neuron of the same feature map is used to extract local characteristics of different positions in the former layer. In order to obtain a new feature, the input feature maps are first convolved with a learned kernel (mask) and then the results are passed into a nonlinear activation function. We will get different feature maps by applying different filter masks. The typical activation function is softmax, sigmoid, *tanh* and Relu.

2.2 Pooling Layer

Secondary feature extraction can be done within the pooling layer of a CNN. It essentially reduces the dimensions of the feature maps and increases the robustness of feature extraction. Usually placed between two Convolutional layers, the size of feature maps in the pooling layer is determined according to the moving step of the masks. Also referred to as stride of masks. The two major pooling operations are average pooling and max pooling. We can also extract the high-level characteristics of inputs by stacking several Convolutional layers and pooling layers.

2.3 Fully connected Layer

We flatten our matrix in vector form and feed it into the fully connected layer. In a fully connected layer, all the neurons in the previous layer are connected to every single neuron of the current layer. No spatial information is preserved in the fully connected layers. An output layer follows the last fully connected layer. After combining all the neurons, we can see

the entire neural network. For classification tasks, softmax regression is commonly used because it generates a well-performed probability distribution of the outputs to classify as dog, cat, car, truck, etc.

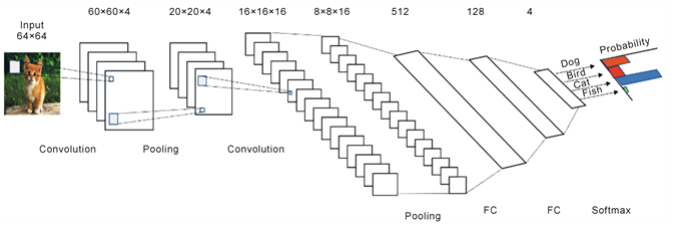

Fig -1: CNN Architecture

3. OUR SIMPLE CNN

In order for the machine to identify objects accurately, we need to train it. We adopted the CNN model to do so. The first layer is the input layer wherein the entire dataset(in batches) is fed into the network model. The process of feature extraction begins from the second layer until it reaches the last image in the last batch. A different feature is extracted in each layer as the construction of the neural network progresses. For example, first it extracts a point, then a line and then a curve. All the features are combined at the end (fully-connected layer) resulting in an output layer.

3.1 Tensor reshaping and transpose

Original one batch of data is 10000 x 3072 matrix in the form of an array. As mentioned in CIFAR-10/CIFAR-100 dataset, the number of columns represents the number of sample data and the row vector indicates a color image of 32 x 32 pixels. Since the dimension of the input vector must be either (width x height x num_channel) or (num_channel x width x height) to feed an image data into a CNN model, we have to reshape and transpose the input vector of CIFAR-10. An image of the row vector consists of 32x32x3(width x height x num_channels) = 3072 elements. The logical concept of reshaping an image is described below:

1. Divide the row vector into 3 frames, where each frame is denoted as color channel resulting in (10000x3x1024) tensors.
2. Further, divide 3 frames by 32, 32 is the width and height of an image which results into (10000x3x32x32) tensors.

This shape (num_channels x width x height) is not accepted in TensorFlow; therefore transpose of the reshaped image is taken.

3.2 Normalize

Each batch within the dataset is broadly divided into two parts. The first part being the training data. This is the largest part and forms 80% of the total data residing in each batch. The remaining 20% is known as the validation data and it used to validate the data once it is trained. After training and validation of each batch, the entire dataset is tested as a whole. There is a separate testing dataset for this purpose.

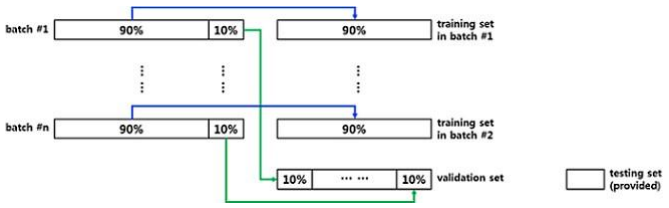

Fig -2: Division of dataset

3.3 One hot encoding

The process by which categorical variables are transformed into a format that could be provided to machine learning algorithms to do a better job in prediction is known as one hot encoding. The label data consists of a list of 10000 numbers ranging from 0 - 9 which corresponds to each of the 10 classes in CIFAR-10. We create N new features, where N is the number of unique values. One hot encode function, takes the input as a list of labels. The total number of elements in the list is the total number of samples in the batch. One hot encode function returns two-dimensional tensor- row vector that represents the size of the batch and the number of columns that represents a number of image classes.

index	label
0	airplane (0)
1	automobile (1)
2	bird (2)
3	cat (3)
4	deer (4)
5	dog (5)
6	frog (6)
7	horse (7)
8	ship (8)
9	truck (9)
...	...
...	...

original label data

label	index										...	...
	0	1	2	3	4	5	6	7	8	9		
airplane	1	0	0	0	0	0	0	0	0	0	...	...
automobile	0	1	0	0	0	0	0	0	0	0	...	...
bird	0	0	1	0	0	0	0	0	0	0	...	...
cat	0	0	0	1	0	0	0	0	0	0	...	...
deer	0	0	0	0	1	0	0	0	0	0	...	...
dog	0	0	0	0	0	1	0	0	0	0	...	...
frog	0	0	0	0	0	0	1	0	0	0	...	...
horse	0	0	0	0	0	0	0	1	0	0	...	...
ship	0	0	0	0	0	0	0	0	1	0	...	...
truck	0	0	0	0	0	0	0	0	0	1	...	...

one-hot-encoded label data

Fig -3: One hot encoding

4. IMPLEMENTATION

4.1 TensorFlow workflow

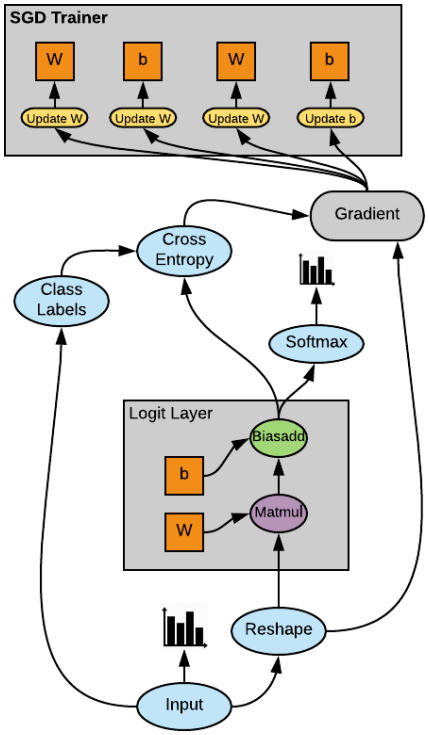

Fig -4: TensorFlow dataflow

The model is provided with two kinds of data for training the model. The image data is fed into the model, which learns and predicts the output accordingly. The other kind of data is label data, which is provided at the end of the model to be compared with the predicted output.

1. Convolution with 64 different filters in size of 3 x3.
2. Max pooling by 2 (Batch Normalization)
3. Convolution with 128 different filters in size of 3 x 3.
4. Max pooling by 2.
5. Convolution with 256 different filters in size 3 x 3
6. Max pooling by 2.
7. Flatten the 3D output of the convolving operations.
8. Fully connected layer with 128 units
    - Dropout
    - Batch Normalization
9. Fully connected with 256 units
    - Dropout
    - Batch Normalization
10. Fully connected layers of 10 units

4.1.1 Hyperparameters

The parameters in any deep learning algorithm should be initialized before training a model. 'keep_probability' is a parameter, which defines the probability of how many units of each layer, should be kept and specifies the dropout technique.

4.1.2 Cost function and Optimizer

The input tensor gets reduced which results in loss of function between the predicted output and label data. CIFAR-10 has to measure loss over 10 classes and it uses

softmax cross entropy function to do so. While training the network, in order to minimize the cost, apply Adam Optimizer algorithm.

4.2 Application snapshots

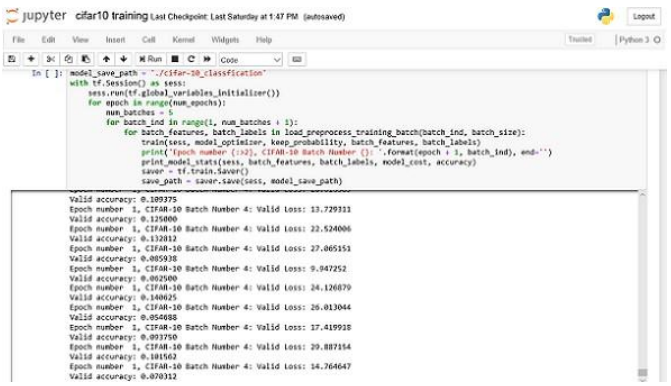

Fig -5.1: Training of the data

Fig -5.2: Output

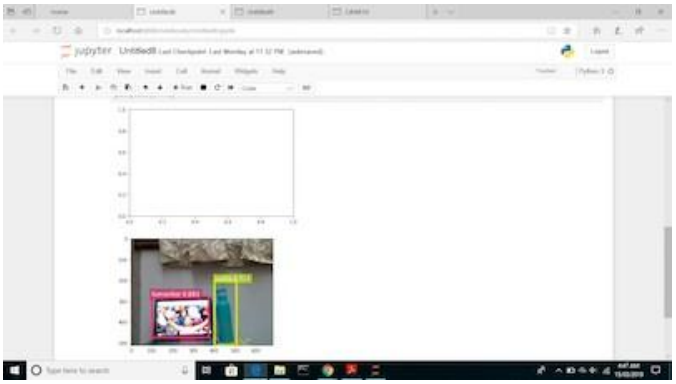

Fig -5.3: Bounding box around detected objects

5. CHALLENGES AND FUTURE WORK

The computational cost and time in a neural network is higher as compared to any other network models (R-CNN, Boltzmann machines, etc.) The most crucial requirement necessary to train CNN is a GPU (graphical processing unit). If the desktop/laptop used for training does not contain a GPU, the processing required for training a model increases which affects the performance. Therefore, it is imperative that the computer we use for training must have a GPU.

The more, the model is trained the accurate it is. Hence, a huge amount of training data is required. This sometimes leads to the slow processing speed of the computer. Despite the shortcomings there is no limit to where a CNN can be used. From developing a Facial recognition software to using it in the advancement of Self Driving Cars. A voice-over interface along with the object detection model can prove to be a boon in the everyday lives of visually impaired people.

6. CONCLUSION

This paper presents a comprehensive review of deep learning and convolutional neural network architecture. For the applications in the computer vision domain, the paper mainly explains how the advancements of CNN based schemes have made it most suitable for images. Despite the assuring results recorded so far, there is significant room for further advances. For example, the theoretical foundation does not yet explain under what conditions they will perform in the desired manner or outperform other approaches. TensorFlow is used to achieve object detection with maximum accuracy for a live scene. A bounding box is created around each object detected which displays the class label and the percentage of accuracy.

REFERENCES

1. Fu-lian Yin, Xing-Yi Oan, Xian-Wei Liu, Hui-Xin Liu, “Deep Neural Network Model Research and Application Overview”, Department of Information and Engineering, faculty of Science and Technology, communication University of China, Beijing.

2. Shin-Jye, Tonglin Chen, Lun Yu, Chin-Hui Lai, “Image Classification based on Boost Convolution neural Network”, Institute of Technology Management, National Chiao Tung University, Hsinchu, Taiwan; National Pilot Scool of Software, Yunnan university, Kunming, China; Department of Information Management, Chung Yuan Christian University, Chungli, Taiwan.

3. “Image Classification using Deep Neural Networks- A beginner friendly approach using TensorFlow”, Medium

4. “Real-Time Object detection API using TensorFlow and OpenCV”, Towards Data Science

5. Ladikos, A., Benhimane, S., and Navab, N. (2007). A real-time tracking system combining template-based and feature-based approaches. In *VISAPP*.

6. Boffy, A., Tsin, Y., and Genc, Y. (2006). Real-time feature matching using adaptive and spatially distributed classification trees. In *BMVC*.

7. Krizhevsky A, Sutskever I, Hinton G E. ImageNet Classification with Deep Convolutional Neural Networks [J]. Advances in Neural Information Processing Systems, 2012, 25(2): 2012. !

8. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D.Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with Convolutionals,” Co RR, vol. abs/1409.4842, 2014.

9. Michael A. Nielson, “Neural Networks and Deep learning”

10. “TensorFlow Tutorial 2: image classifier using convolutional neural network”, CV Trick Bay, H., Tuytelaars, T., and Van Gool, L. J. (2006). Surf: Speeded up robust features. In *ECCV*, pages 404–417.

11. Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2): 91–110.

12. Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*.

13. Matthews, I., Ishikawa, T., and Baker, S. (2003). The tem- plate update problem. In *BMVC*.

14. Mikolajczyk, K. and Schmid, C. (2004). Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86.

15. Sepp, W. and Hirzinger, G. (2003). Real-time texture- based 3-d tracking. In *DAGM Symposium*, pages 330– 337.
16. Najafi, H., Genc, Y., and Navab, N. (2006). Fusion of 3D and appearance models for fast object detection and pose estimation. In *ACCV*, pages 415–426.

17. Vijayalaxmi, K.Anjali, B.Srujana, P.Rohith Kumar "OBJECT DETECTION AND TRACKING USING IMAGE
18. PROCESSING" Global Journal of Advanced Engineering Technologies, ISSN (Online): 2277-6370 & ISSN (Print): 2394-0921-2014.

19. "Automated Driving Vehicle Using Digital Image Processing" IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 9, September 2015, ISSN 2348 – 7968.

20. Prof. D. S. Pipalia, Ravi D. Simaria " Real Time Object Detection Tracking System (locally and remotely) With Rotating Camera", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 3 Issue: 5 3058 – 3063

21. Prof. D. S. Pipalia, Ravi D. Simaria " Real Time Object Detection Tracking System

22. "Working with Advanced Views in Android-Edureka", Edureka

23. "Real Object Detection with TensorFlow Detection Model", Towards Data Science

24. CIFAR 10 and CIFAR-100 datasets, Canadian Institute for Advanced Research, Toronto University