

Real-Time Object Detection Using TensorFlow

Ningangouda Patil¹, Sonashree TR², R Gautam³, DR. K B Shivakumar⁴

^{1,2,3}Student, DBIT, Bengalore

⁴Professor & Head, DBIT, Bengalore

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

Keywords: 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. LITERATURE SURVEY

Krishna Sai and sasikala (1) focused on detecting harmful objects like threatening objects, To ease object detection for threatening objects, Tensorflow object detection API has been used to train model and implemented faster R-CNN algorithm.

Ross Girshick et al. (2) described a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. Object Recognition is a technique used in the field of computer. It is assumed to be one of the most difficult and challenging tasks in computer.

Shaoqing Ren et al. (3) introduced Faster R-CNN, a method for object detection that utilizes region proposal networks to improve speed and accuracy.

Wei Liu et al. (4) proposed method drew its strength from making normalization a part of the model architecture and performing the normalization for each training mini-batch.

Ross Girshick (5) proposed a Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks.

Kaiming He et al. (6) , conducted a comprehensive study on multiple real-time detection networks (anchor-based, keypoint-based, and transformer-based) on a wide range of datasets and report results on an extensive set of metrics.

Joseph Redmon and Ali Farhadi (7) introduced YOLO9000, a state-of-the-art, real-time object detection system that can detect over 9000 object categories. proposed various improvements to the YOLO detection method, both novel and drawn from prior work.

Sergey Lofe and Christian Szegedy (8) introduced Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training.

Raghunandan Apoorva et al. (9) proposed an application in various fields such as defence, security, and healthcare. various Object Detection Algorithms such as face detection, skin detection, colour detection, shape detection, target detection Kim Jung et al. (10) proposed a novel object detection network DBIT Page 1 which is robust in occlusions. For effective object detection even with occlusion.

3. PROPOSED MODEL

BASIC CNN COMPONENTS

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

Convolutional Layer

The aim of CNN is to learn feature representations of the inputs. As shown in the below image (Fig. 1), Convolution 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.

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.

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.

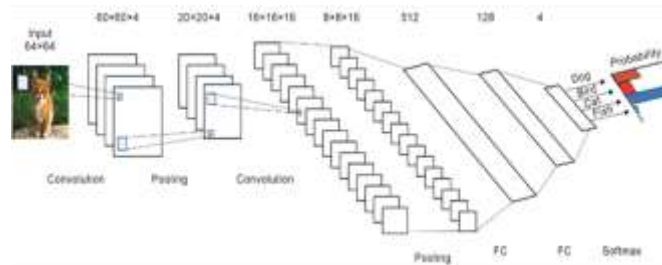


Fig -1: CNN Architecture

3.1 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 I 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.

3.2 IMPROVED CNN

The proposed object detection model leveraging TensorFlow entails a systematic approach encompassing data collection, preprocessing, model architecture selection, implementation, training, evaluation, fine-tuning, deployment, and ongoing monitoring. Initially, a diverse dataset comprising annotated images with bounding boxes around objects of interest is collected. Subsequently, these images undergo preprocessing steps including resizing, normalization, and augmentation to enhance variability. A suitable model architecture, whether pre-existing (e.g., Faster R-CNN, SSD, YOLO) or custom- designed, is selected based on specific requirements. Using TensorFlow's APIs, the chosen architecture is implemented, followed by training on the annotated dataset utilizing techniques such as transfer learning or training from scratch, with hyperparameter optimization to refine performance. Evaluation metrics like mAP are employed to assess accuracy and performance on a separate validation dataset. Fine-tuning and optimization are then conducted based on evaluation results to enhance performance further. The trained model is deployed for inference in production environments, with continuous monitoring and maintenance to ensure sustained accuracy and reliability over time. Thorough documentation and consideration of factors like computational resources and deployment requirements are integral throughout the process.

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:

Divide the row vector into 3 frames, where each frame is denoted as color channel resulting in

(10000x3x1024) tensors.

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.

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.

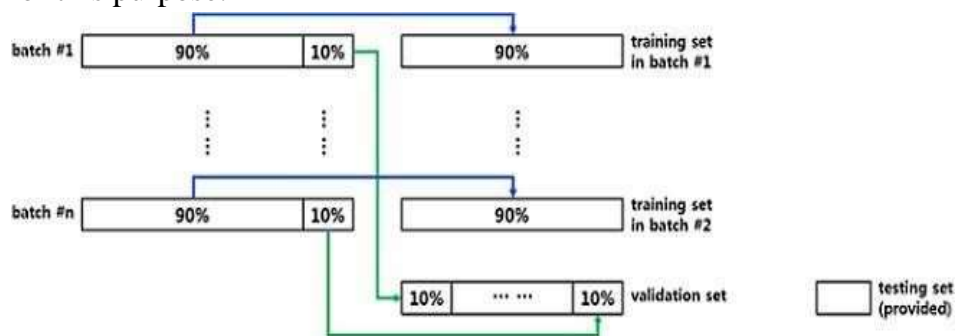



Fig -2: Division of dataset

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)
...	...
...	...



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

Fig -2: Division of dataset

4. IMPLEMENTATION

a. TensorFlow workflow

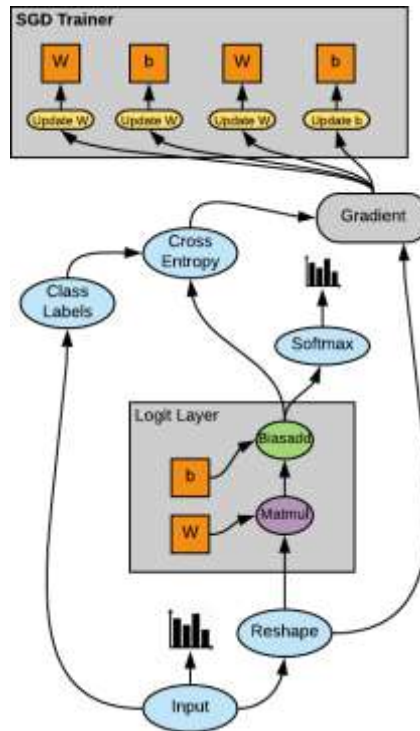


Fig -4: TensorFlow dataflow

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 x3.
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
 1. Dropout
 2. Batch Normalization
9. Fully connected with 256 units
 1. Dropout
 2. Batch Normalization
10. Fully connected layers of 10 units

Hyperparameters

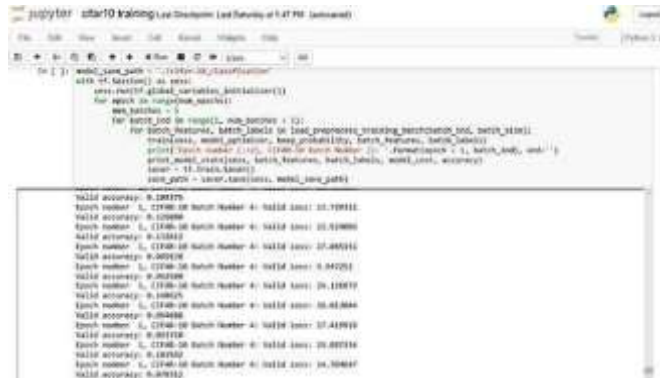
The parameters in any deep learning algorithm should be initialized before training a model. ‘keep_probability’ is a parameter, which defines the probability of how many units of each layer, should be kept and specifies the dropout technique.

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.

b. Application snapshots



```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
data_dir = 'data'
train_data_gen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    crop_shift_range=0.1,
)
train_data_gen.flow_from_directory(
    data_dir,
    target_size=(32, 32),
    class_mode='categorical',
)

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 3)),
    Conv2D(32, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, kernel_size=(3, 3), activation='relu'),
    Conv2D(128, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(256, kernel_size=(3, 3), activation='relu'),
    Conv2D(256, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(512, kernel_size=(3, 3), activation='relu'),
    Conv2D(512, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(10, kernel_size=(1, 1), activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

train_data_gen.fit(train_data_gen.flow_from_directory(
    data_dir,
    target_size=(32, 32),
    class_mode='categorical',
)))

model.fit_generator(
    train_data_gen.flow_from_directory(
        data_dir,
        target_size=(32, 32),
        class_mode='categorical',
    ),
    validation_data=train_data_gen.flow_from_directory(
        data_dir,
        target_size=(32, 32),
        class_mode='categorical',
    ),
    epochs=10,
)

model.save('model.h5')
```

Epoch 1/10: 0.0000 accuracy: 0.0000
Epoch 2/10: 0.0000 accuracy: 0.0000
Epoch 3/10: 0.0000 accuracy: 0.0000
Epoch 4/10: 0.0000 accuracy: 0.0000
Epoch 5/10: 0.0000 accuracy: 0.0000
Epoch 6/10: 0.0000 accuracy: 0.0000
Epoch 7/10: 0.0000 accuracy: 0.0000
Epoch 8/10: 0.0000 accuracy: 0.0000
Epoch 9/10: 0.0000 accuracy: 0.0000
Epoch 10/10: 0.0000 accuracy: 0.0000

Fig -4.1: Training of the data



Fig -4.2: Output

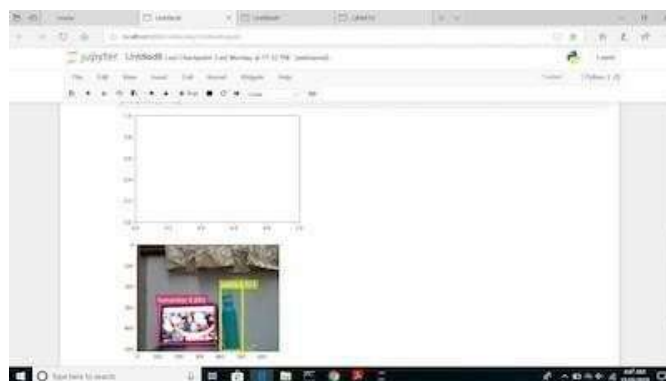


Fig -5.3: Bounding box around detected objects

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. Krishna Sai and Sasikala ,”Deep Learning Object Detection API” , Second International Conference on Smart Systems and Inventive Technology (ICSSIT) pp.679-728 2019.
2. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. ”Rich feature hierarchies for accurate object detection and semantic segmentation”. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 509-522, 2014.
3. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. ”Faster R-CNN: Towards object detection with region proposal networks”. In Advances in Neural Information Processing Systems (NIPS), pp. 349-369 , 2015.
4. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, ChengYang Fu, and Alexander C. Berg. SSD:” Single shot multibox detector”. In ECCV, pp.827-891, 2016.
5. Ross Girshick. “Fast R-CNN”. In: Proceedings of the IEEE international conference on computer vision., pp. 1440– 1448, 2015.
6. Kaiming He et al. “Deep residual learning for image recognition”. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition , pp. 770–778, 2016.
7. Joseph Redmon and Ali Farhadi. “YOLO9000: Better, faster, stronger”.In: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, pp. 6517–6525, .
8. Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: 32nd International Conference on Machine Learning, ICML 2015. Vol. 1.International Machine Learning Society (IMLS), pp. 448–456, 2015.
9. Raghunandan, Apoorva, Pakala Raghav, and HV Ravish Aradhya."Object Detection Algorithms for video surveillance applications".International Conference on Communication and Signal Processing (ICCSP), pp. 0563- 0568. IEEE, 2018
10. Kim, Jung Uk, Jungsu Kwon, Hak Gu Kim, Haesung Lee, and Yong Man Ro. "Object Bounding Box-Critic Networks for Occlusion-Robust Object Detection in Road Scene." In 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 1313-1317. IEEE, 2018