EasyChair Preprint

№ 3434

Object Detection Using Deep Learning

Mukul Bhardwaj

May 18, 2020

Object Detection Using Deep Learning

Mukul Bhardwaj

Bhardwaj.mukul1998@gmail.com

Galgotias University, Uttar Pradesh 203201, India

Abstract

Today computer is one the important part in one's life almost everyone is using a computer whether that is a pc or a smart-phone. Digital technology has seen a large boom in past two decades which is resulted in an increase in the power of modern computer devices even modern smartphones are capable of performing over billion operations per second with this increase on performance, usage of these devices has also increased rapidly which is resulted in an increase in the amount of data. With this increase in power of modern pcs concepts like Deep Learning has also seen a boom with the involvement of deep learning concepts object detection in real-time become a reality using deep learning we can detect the object very quickly and with higher accuracy.

Keywords: Deep Learning; CNN; Pooling; Flattening; Full Connection; Object Detection

1. Introduction

Pre deep learning era concepts like sliding window, selective search, cascade classifier was used these were basically based on handcrafted features these models were not that accurate and also not that fast to detect and classify an object.

So, with the evolution of deep learning the detection and classification become more and more accurate and the detection becomes very fast. In deep learning Convolutional Neural Network [1] is a subclass which is mainly used in the working with visual aspects or features. Convolutional Neural Network extracts features from the image to identify the object.

Features like edge, corner, region of interest and interest points. To identify these features; feature detectors like canny which is used to detect edges, SUSAN is used for edge and corner detection, grey-level blobs is used for blob or region of interest detection.

So, to summaries to identity the features we apply different detector or filters to obtain certain features, these features are stored in

feature vector which is used in further calculation or detection and then classification.
In this paper, we will discuss these concepts in details and we will get a better understanding of the Convolutional Neural Network and how it really works. Also, we will learn about the concepts like convoluting, pooling, flattening which are the basics of CNN.

2. Object Detection Model

For building CNN model, we need to understand certain components [2] which are to be followed

- Convolution
- Pooling
- Flattening
- Full Connection

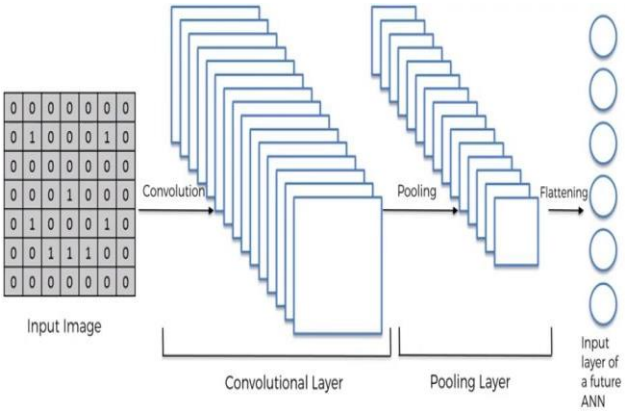

Figure 1 [6] Simple Graphical Representation of CNN

2.1 Convolution

In mathematical terms convolution is a function, which is used to find the relation, that is how the shape or pixel behave on modifying the other pixel.

Mathematical formula of Convolution is

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Equation 1 Mathematical formula for Convolution

Basically, inside convolution 3 steps are getting performed

A. Input
We know that image is an array. Black and white image is a 2D array whereas RGB image is a 3D array

B. Feature
Feature Detector is a filter which we apply on the input and perform elementary multiplication operation pixel by pixel this allows us to obtain feature from the input image like if we apply canny, we get all the edges. It is also an array of any dimension

C. **Feature Map**
Feature Map or Activation Map, contains all the values or all the feature from the image after applying the filter or the filters or the feature detector.

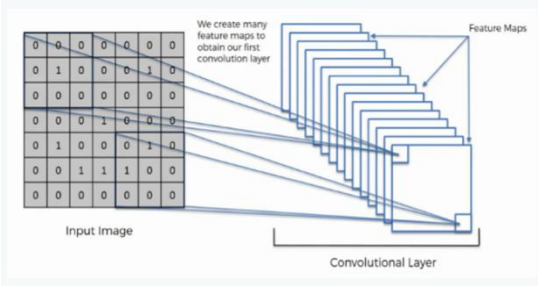

Figure 2 [6] How filter is applied and feature maps are created to form Convolution Layer

2.2 Pooling

Pooling is nothing but the identification of a particular feature with spatial features. This helps the

convolutional neural network to identify that feature irrespective of the location of that feature in the particular position.
There is a different type of Pooling like Max Pooling, Mean Pooling, Sum Pooling.
We have discussed about pooling so many of you will about “How does Pooling work?"
The Conceptual thing is we select n x n pixel from the image and considering the type of pooling, the model selects the value and discards the other values.
Since we have used Max pooling in the project just to select the max value pixel since max value represent the most dominating feature, below image, shows how the max-pooling works.

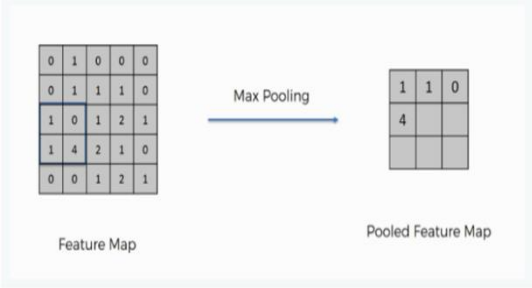

Figure 3 [6] Max pooling: selecting the maximum value from the n x n matrix

In the figure 3 we have selected a 2 x 2 matrix and selecting the maximum value from the box like from highlighted box in feature map since ‘4’ is the maximum value so we have selected ‘4’ and since we have selected the maximum value irrespective of the position of the feature, so will always get that feature resulting in the removal of the distortion both spatial and textural distortion.
We can say that pooling thus reduces the size of the feature map which results in a reduced number of nodes which reduces the complexity and that too without losing any vital feature.
Pooling also removes the problem of overfitting problem: a problem where our model learns too many features and resulting in reducing the accuracy of the model.

2.3 Flattening

Flattening is nothing but putting all the values from pooled feature map into one single vector

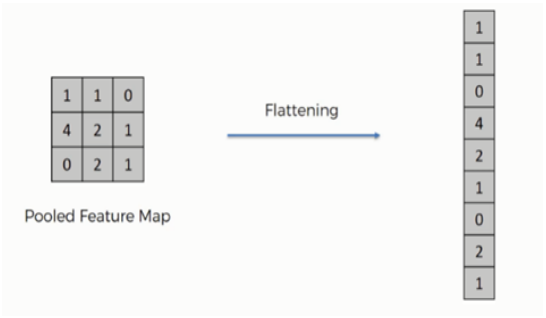

Figure 4 [6] Flattening the Pooled feature map

So, with flattening there is a question that “why there is no loss of spatial structure with flattening” as in this step all the features are mapped into single vector we are not adding or removing any sort of feature values we are just flattening them.
High values in the feature map represent spatial features and since we have applied max pooling, we kept all these features and flattening just puts these features into a single layer or column-like structure as shown in Figure 4, thus we are keeping all the features.
Without Convolution and Pooling, if we apply Flattening, we won't get any information about the other pixels.

2.4 Full Connection

Full Connection is adding artificial neural network to our convolutional neural network. The whole purpose of involving ANN is to combine the feature to get

prediction and classification and also data is combined into a wide variety of attribute which makes CNN more capable of prediction and classification of the object in an image which is the whole purpose of the CNN.
There are basically 3 layers in Full Connection as shown in Figure 5: Input Layer, Full Connection layer (ANN), Output layer.
Vectors which are generated form flattening serves as the input layer, these inputs pass through full connection layer where all the prediction and classification is done and the output/outputs are passed to the output layer.

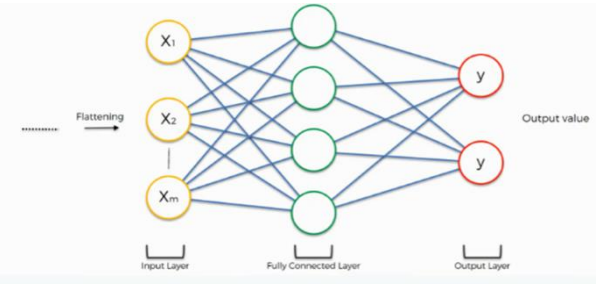

Figure 5 [6] Layers in Full Connection

3. Implementation

We have used the model knowledge to create our own CNN model for the object detection. Keras Framework is used which uses TensorFlow at the backend.
The model will accept the image as an input and we need to create to folders one for validation and other for testing. Generally, we divide the data in such a way that about 80% images are for training and 20% images are for testing for example if we are using 10000 images then 8000 images for training and 2000 for testing. This ratio is not mandatory, but for the better performance and testing, we follow this ratio.

In this we see the steps to build the model using the CNN knowledge that we have discussed we see how the convolutional layer is created, how is pooling, flattening, and full connection using python code.
The first step is to check the format of the image, here we are just checking whether RGB format is coming first or not. Initialization of CNN is done using "Sequential()" [3] prior using this make sure that Keras library is called.
Post initialization convolutional layer is created using Conv2d() [3] function in conv2d() we need to provide the filter, kernel_size and input_shape, an activation function is also required I have used ReLU, [4] to remove the linearity from the model. ReLU [4] removes all the negative values and replaces them with zero

$$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ 0 & \text{if } x_i < 0. \end{cases}$$

Equation 2 ReLU equation

for max pooling add(MaxPooling2d()) [3]is used in this function we need to provide the pool_size.
Below is the code which is used to initialize the CNN and creating a convolutional layer and also calling activation function.

```
"#Initializing CNN
model = Sequential()
#Convolutional Layer
model.add(Conv2D(32,(2, 2), input_shape = input_shape))
model.add(Activation('relu'))
#Pooling
model.add(MaxPooling2D(pool_size=(2,2)))"
```

we can add multiple convolutional layers in order to extend the accuracy.
Flattening can be done using Flatten() [3].
For full connection Dense() [3] is used for adding hidden layer in the full connection, for the hidden layer we will be using ReLU [2] to remove linearity Dropout() [3] is used

to remove the chances of overfitting. Dense() [3] is again used to create output layer and sigmoid activation is used for binary output and softmax activation function is used for the more than two output

```
"#Full Connection
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))"
```

values in the Dense() [3] defines the number of the hidden layer.
For compiling the model compile() [3] is used

```
"#Compiling CNN
model.compile(loss='binary_crossentropy'
,optimizer ='adam',
metrics =['accuracy'])"
```

binary_crossentropy is used when we dealing with binary output and for categorical output categorical_crossentyopy, class_node is used. Adam optimizer is used as it is the replacement of stochastic gradient descent and also it achieves satisfactory result very fast.
Before starting the training of the model we should make sure we need to apply some changes to the data so that every time new data is received by the model and these changes are zooming the image flipping it, rescaling it, applying shear ranges so to do that ImageDataGenerator() [3] is used both on training data and test data.
We can't directly use data into the model so to make data so in order to prepare data flow-from-directory() [3] is used here we specify the directory, target size and batch size also class mode is selected for the type of output above function is used for both train data and test data.

```
"train_generator=train_datagen.flow_from_
directory(train_data_dir,
target_size =(img_width, img_height),
batch_size=batch_size,
class_mode ='binary')"

"validation_generator=test_datagen.flow_fr
om_directory(validation_data_dir,
target_size =(img_width, img_height),
batch_size=batch_size,
 class_mode ='binary')"
```

The last thing to do in our model is start training of the data so for that fit-generator() [3] is used which contain test data, steps-per-epochs, epochs, validation data, validation steps.

```
"model.fit_generator(train_generator,
steps_per_epoch=nb_train_samples//
batch_size,
epochs=epochs,
validation_data = validation_generator,
validation_steps = nb_validation_samples
// batch_size)"
```

So to save the weights from the model we will use the save-weights()

```
"model.save_weights('model.h5')"
```

4. Result

We have achieved an accuracy of 95.81% and the validation accuracy is 83.92%, since we have used 3 convolutional layers that is the reason, we have achieved the accuracy above 90% and also rescaling the image also helps in increasing the accuracy. With the use of only one convolutional layer, the accuracy was in mid-80%.
Using more convolutional layer is the most effective way to improve the accuracy, adding convolutional layer is not the only solution, we can reduce the size of the kernel, Rescaling the image is one solution and also try to use low size and low-resolution images as these are faster to load.

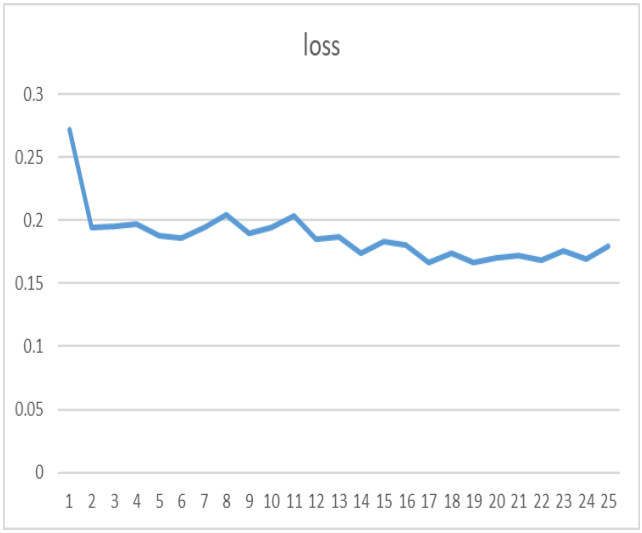

Figure 6 loss per epoch

With each iteration we are getting lower loss which a good result as loss predict how well our model is performing.

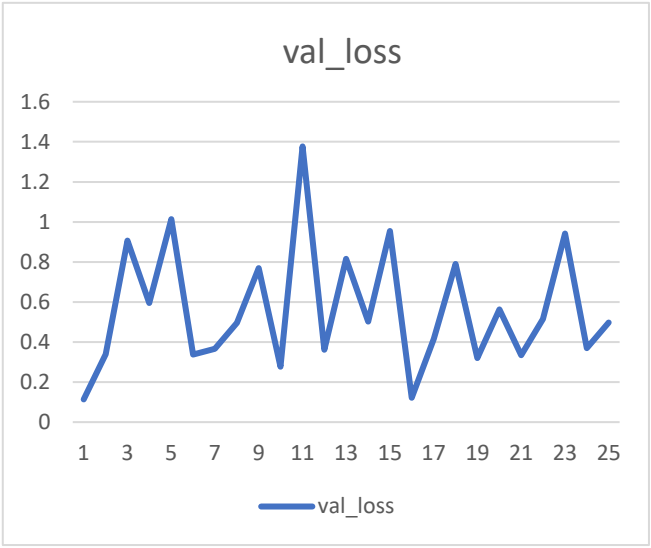

Figure 7 val_loss per epoch

Val_loss is same as loss but this works on the validation data. Basically, this is loss on the test/validation data.

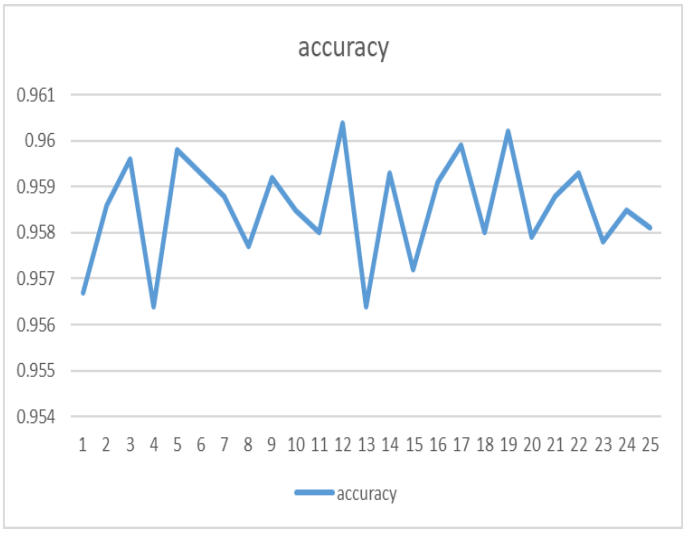

Figure 8 accuracy level per epoch

Accuracy is the level of correctness on the training set.

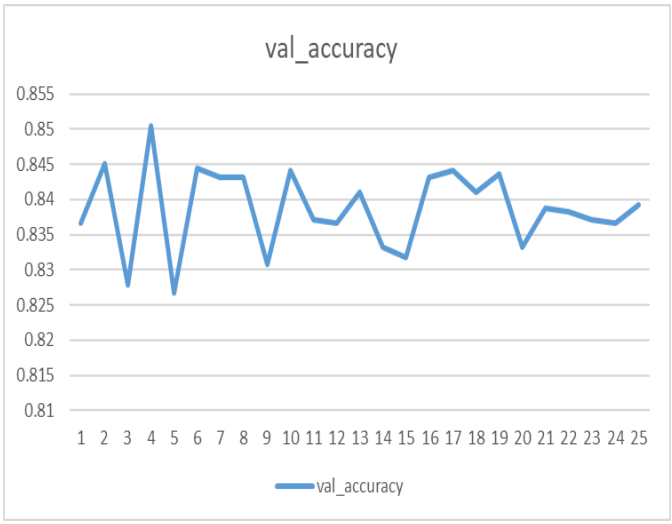

Figure 9 val_accuracy per epoch

Val_accuracy is the level of accuracy outside the data or on the validation set.

Table 1 value of loss, accuracy, val_loss, val_accuracy per epoch

Epoch	loss	accuracy	val_loss	val_accuracy
1	0.2723	0.9567	0.1135	0.8366
2	0.1941	0.9586	0.3391	0.8451
3	0.1949	0.9596	0.907	0.8278
4	0.197	0.9564	0.5947	0.8505
5	0.1875	0.9598	1.014	0.8268
6	0.1859	0.9593	0.3377	0.8444
7	0.1947	0.9588	0.3661	0.8431
8	0.2042	0.9577	0.4965	0.8431
9	0.1901	0.9592	0.769	0.8308
10	0.1939	0.9585	0.277	0.8441
11	0.2035	0.958	1.3769	0.8372
12	0.1851	0.9604	0.3611	0.8366
13	0.1873	0.9564	0.8148	0.8411
14	0.174	0.9593	0.5018	0.8333
15	0.1831	0.9572	0.9548	0.8318
16	0.1801	0.9591	0.1223	0.8431
17	0.1668	0.9599	0.4164	0.8441
18	0.1739	0.958	0.7906	0.841
19	0.1664	0.9602	0.3199	0.8436
20	0.1701	0.9579	0.5627	0.8333
21	0.1723	0.9588	0.3346	0.8387
22	0.1687	0.9593	0.5155	0.8382
23	0.176	0.9578	0.9423	0.8372
24	0.1697	0.9585	0.369	0.8366
25	0.1792	0.9581	0.4975	0.8392

5. Conclusion

In this paper, we have discussed about the basic knowledge about Convolutional Neural Network and we have also discussed how to build our own model for the object detection and classification. Our result shows that our model is giving some good accuracy and we have also discussed the ways to increase the accuracy of the model.

Validation accuracy, we are need to increase in the near future and need to reduce the validation loss. We can add more range in the data in future so we can remove the problem of the overfitting and underfitting we can also increase the number of epochs and batches to improve these variables but with increasing number of epochs, but we should not increase epochs with too, this will cause a problem of overfitting and thus reducing the accuracy.

References

[1] J. Wu, Convolutional neural networks, Nanjing University, China: National Key Lab for Novel Software Technology, 2020.

[2] MACHINECURVE, "MACHINECURVE," [Online]. Available: https://www.machinecurve.com/index.php/2018/12/07/convolutional-neural-networks-and-their-components-for-computer-vision/#.

[3] Keras, " Keras Documentation," [Online]. Available: https://keras.io/.

[4] N. W. T. C. M. L. Bing Xu, "Empirical Evaluation of Rectified Activations in Convolution," 2015.

[5] O. D. i. 2. Y. A. Survey, "Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye," 2019.

[6] superdatascience, "superdatascience," [Online]. Available: https://www.superdatascience.com/.