Find Coordinates

Sivakumar Balasubramanian

March 3, 2018

1 Problem Statement

The task is to implement a prototype of a visual object detection system. We are required to find a location of an object dropped on the floor from a single RGB camera image. We need to find the normalized coordinate of the center of the object in the given image.

2 Approach

The problem can be approached as an object recognition task. One approach is to use the classical Computer Vision (CV) methods for feature representation such as SIFT [1], HOG [2], LBP [3] coupled with a Machine Learning(ML) framework for classification or detection. The other way is to use Deep learning for the same without specifying the representation and allowing the model to select its own features [4]. The two approaches are explained below and the latter is used for this project.

2.1 Classical Approach

The steps for implementing the CV approach is as follows.

- Use the coordinates of the center of the object to make a copy of the square block of pixel values of a particular size 's' around it and store them as separate images.
- Do this for each image to create a dataset containing object images in different orientations.
- Collect blocks of similar sizes from other regions of the images without the object image.

- Using CV tools create feature representations for the above datasets.
- Train a ML model using these features as the training data.
- Pass a sliding window of the same size 's' for the test image.
- Using the model predict the block with the object.
- The center of that block can be used as the center of the object.

These algorithms have the following disadvantages:

- Choosing one feature representation might not be versatile enough for the prediction.
- It is difficult to choose the most suitable representation for the task.
- When this prediction needs to be done for different objects with different shapes the feature representation process becomes tedious.

2.2 Deep Learning approach

The Deep Learning approach uses just the given data and labels to train a model and learn necessary features. This model can be used for predicting the coordinates of the center of the object in the test images. The Convolution Neural Network (CNN) model is chosen for the task as it has been effective for vision based tasks compared to other Algorithms [5][6][7]. The approach used and the various analysis are explained in the subsequent sections.

3 Convolution Neural Network regression

The CNN method has been successfully used for image classification [5]. It consist of convolution layers and other layers such as max pooling, ReLU activation layer, fully connected layer, etc. In classification tasks a sigmoid activation layer is used at the end to get normalized outputs from the fully connected layers. When The CNN is used for regression the activation function is made linear. This gives continuous output values.

For the Find coordinates task, the labels of normalized coordinates are transformed to pixel values for training. The final fully connected layer is made to give 2 outputs, i.e, the x and y pixel coordinate values of the center of the object. The predicted pixel values are re-normalized before calculating the accuracy. The CNN method generally requires large datasets for generalization. Thus it is essential for collecting huge amounts of training data. But in the case with very less training data, we use data augmentation techniques to create our own data as explained in the next subsection.

3.1 Data Augmentation

A total of 129 images of a particular phone dropped on the floor with corresponding normalized coordinates were provided. To get more data with the images the following steps were followed. Figure 1, shows the augmented images of one of the given image.

- The 129 images were rotated 180 degrees and the corresponding labels are changed. This gave us a total of 258 images.
- These 258 images were mirrored in the left to right direction giving rise to 516 images in total.
- The 516 images were mirrored in to top to bottom direction to obtain 1032 images.

Figure 3.1: From left to right: the actual image, image rotated by 180°, image flipped left to right, image flipped top to bottom

The scikit-learn library was used to split this data into training and Validation data. We get 825 training samples and 207 validation samples.

3.2 Training and Analysis

The model was built using the Keras library in python with the Theano back-end. Tuning the parameters of the CNN to get the desired results is a difficult task. Factors like the feature length of the convolution layers, the depth of the network, training time, the optimizer used play a major role in getting best performance. We use a mean squared error metric as the loss function and the Adam optimizer [8].

First, a shallow network with just one convolution layer with 10 features was trained for 50 epochs. The training and validation accuracy are obtained. The problem aims to get a coordinate value within 0.05 normalized radius of the actual coordinate value. When this criteria was used as the error margin, we get very low training and validation accuracy. But if the error margin was increased to 0.1 and 0.15 normalized radius, the training and validation accuracy improved. The same with networks with one convolution layer and feature lengths of 20 and 30 were used and trained for 50 epochs. The training accuracy increased for all the criteria as the number of features increased. Figure 3.2 shows a plot of the training and validation accuracy for 10, 20, 30 features for a criteria of 0.05, 0.1, 0.2 normalized radius. This shows

that with more complex network architecture and more training we can achieve better performance. A factor we need to consider is overfitting. The more complex architectures tend to overfit for small datasets. It is essential to add dropout layers during training to avoid this and collect more training data.

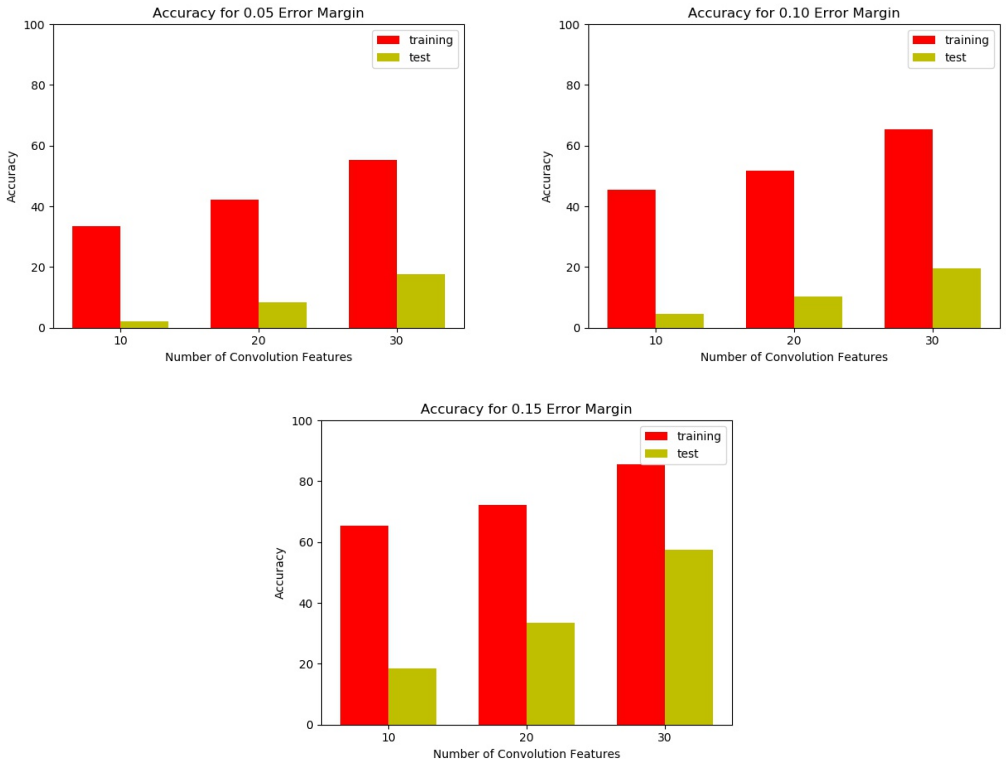

Figure 3.2: Plots showing the increase in training and validation accuracy with increase in number of convolutions for a CNN with single convolution layer. The plots are made for error margins of 0.05, 0.10, 0.15.

A CNN with 2 convolution layers with 50 convolutions in the first and 20 in the second was used. This network was trained for 100 epochs. A training accuracy of 87.5 % and validation accuracy of 65.7 % was obtained for this network. This network is used as the submitted code.

4 Other Methods

Other models such as Linear, Ridge and SVM regression were used. The models were built using scikit-learn library. The images were stacked as a single column vector of data instances. The dimensions of the data points are very large compared to number of data points. Thus Principal Component Analysis(PCA) was used for dimensionality reduction. The results

showed an underfit with just around 7 % on training accuracy. The deep learning models capture deeper features compared to the regression models and give better performance.

5 Conclusion and Discussion

From the experiments, we can conclude that the CNN model is a very good solution for this problem. With optimal number of hidden layers we can get high performance on both the training and validation accuracy. Also, the training samples given had phones on different floor patterns. But the number of images with different phone positions was less for each floor pattern. The training data collection could have included more positions of the phones per floor pattern for better results.

As next steps, an edge image of the training samples can be used for training to check if the performance improves. Other optimizers such as rmsprop, adagrad,etc and loss functions can be used to find the right model for this particular dataset. More data augmentation such as shifting the images to get newer phone positions can be performed to increase the training data.

In a futuristic perspective the major advantage of the CNN model is that with suitable training samples this can be generalized to find the center coordinates of different phones and even different objects without the need for feature representation.

6 REFERENCE

[1] David G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2, ICCV 99, pages 1150, Washington, DC, USA, 1999. IEEE Computer Society.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. Computer Vision and Pattern Recognition, 2005.

[3] T. Ojala, M. Pietikinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. Pattern Recognition, 29:5159, 1996.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521:436444,2015.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classication with deep convolutional neural networks. Neural Information Processing Systems, 2012.

[6] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional models for semantic segmentation. The IEEE Transactions on Pattern Analysis and Machine Intelligence,2016.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. Computer Vision and Pattern Recognition, 2014.

[8] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).