Soccer Robot Perception

Lab Report: Lab CudaVision – Learning Vision Systems on Graphics Cards
Autonomous Intelligent Systems, 2019

Pratika Kochar, Baraa Hassan

Object detection and segmentation in computer vision have witnessed significant improvements through the use of unified perception convolutional neural network. NimbRoNet2, the winner of RoboCup 2019 AdultSize is one such example. In this work, we implement and try to reproduce the results of the NimbRoNet2's visual perception system. This network has two output heads one to detect the ball, robots and thin goalposts and the other to segment field, field lines and background. These two heads help the robot to have a semantic understanding of the scene, so it can decide it's next step.

1 Introduction

In recent times, deep learning has offered increased efficiency in complex computer vision problems such as semantic segmentation, localization of objects, feature detection, and object detection. It has been developing through the years, starting from the trivial brute-force technique to various complex techniques. Nowadays, semantic segmentation is one of the key problems in the discipline of computer vision. Looking at the big picture, semantic segmentation is one of the high-level tasks that paves the way towards complete scene understanding. The importance of scene understanding as a core computer vision problem is played up by the fact that an increasing number of applications depends on inferring knowledge from images. Some of those applications include self-driving vehicles, human-computer interaction and virtual reality. With the popularity of deep learning in late years, many semantic segmentation problems are being tackled using deep architectures, most often convolutional neural networks, which surpass other approaches by a large margin in terms of accuracy and efficiency. Unified Perception Convolutional Neural Network, NimbRoNet2 [2] has been exceptional to solve both problems with its two output heads to understand the full scene for the robot and to help the robot decide for the succeeding step. NimbRoNet2 [2] has been trained in such a way that it is capable of recognizing soccer field objects- ball, goalposts, robots. It can also differentiate between field, field lines and background. We implement the visual perception system of NimbRoNet2 and try to achieve the similar performance.

2 NimbRoNet2 Architecture

The model uses an encoder-decoder architecture similar to pixel-wise segmentation models. Similar to NimbRoNet2 [2], the model has two output heads; one for object detection and the other for pixel-wise segmentation. The detection head predicts the location of ball, robots and goalposts while the segmentation head segments the field, lines and the background.

2.1 Encoder: Pre-trained Resnet18

The architecture of the encoder is the is same as that of the Resnet18 architecture, but the global average pooling and fully connected layers are removed to accomplish the segmentation and detection tasks. We use the pre-trained ResNet18 for transfer learning, thereby decreasing training and computational efforts. Results of the intermediate layers of ResNet18 are also fed to the decoder in order to provide high resolution details of the input image.

2.2 Decoder

In the decoder layer, transpose-convolutional layers are used to up-sample the representations. Considering the importance of location-dependent features useful for objects like goalpost, we use location-dependent convolutional layer. There is a shared learnable bias between both the output head. The architecture is illustrated in Figure 1

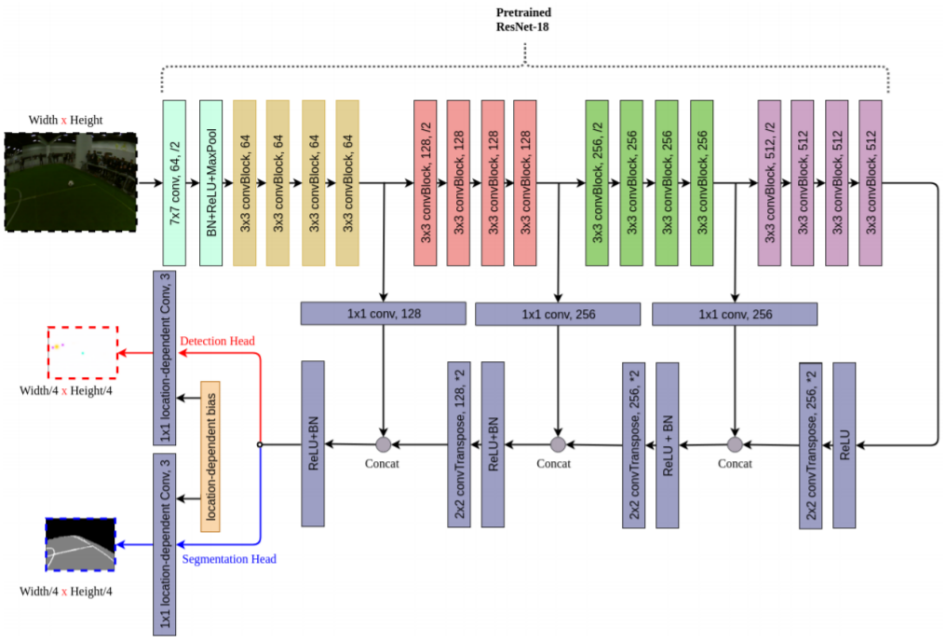

Figure 1: NimbRoNet2 architecture [2].

3 Proposed Method

3.1 Problem Formulation

The idea here is to semantically understand the scene, using the two heads of our model. Hence, one head will robustly detect the soccer ball, goalposts, and robots in the field while the other head will segment the field, lines and background, using the properties: texture, shape, color, brightness of the input image.
Following sections describes the pre-processing and the post-processing techniques incorporated.

3.2 Dataset

3.2.1 Object Detection Dataset

Dataset consists of input images and the corresponding annotation file containing location of objects. We resize all the input images to 480x640 using nearest neighbor interpolation. To facilitate the task of detection, we generate a single heatmap for each input image. Each heatmap consists of gaussian blobs at the corresponding location of each object - ball, robots, goalposts. Each object is represented by different colours. We use a smaller radius for the goalposts and balls and a large radius for the robots. Gaussian blobs are considered so that the object shape in the heatmap is smooth, thereby enabling the model to learn easily. Size of each heatmaps is 1/4th of the input image size i.e. 120x160. Prior training, we store the heatmaps for each input image. Data augmentation is achieved using color jitter transformation.

3.2.2 Segmentation Dataset

Dataset consists of input images and groun-truth images.We resize all the input images to 480x640 using nearest neighbor interpolation and all the ground-truth images to 1/4 of the input images size i.e. 120x160.

3.3 Pre-processing

3.3.1 Object Detection

Each heatmap for the corresponding input image has gaussian blobs at the location of each object with colours [normalised tensors] defined as: cyan for balls [0 1 1], magenta for goalposts [1 0 1] and yellow for robots [1 1 0]. We change all the zero values to 0.2 to force the model to not have any zero values in order to decrease the false postives.

3.3.2 Segmentation

We map the ground truth images to the class indices using the grayscale values of the corresponding image. We have 3 classes: background, field and field lines where the class labels are obtained by mapping the grayscale value of black background to 0 (index for

first class), the grayscale value of thegray field to 1 (the index of second class) and the grayscale value of white lines to 2 (index of the third class).

3.4 Post-processing

3.4.1 Object Detection

For the output produced by the model, we remap back any value in the range [0,0.4] to zero. To remove the noise, values out of the range [0,1] are mapped back to the range of normalized pixel values [0,1].

3.4.2 Segmentation

For the output produced by the model, we remap the class indices to grayscale values. Each pixel is assigned a class according to the highest predicted probability. Post this, we remap the class index 0 to black representing the background, the index of 1 to gray representing the field and the index of 2 to white representing the field lines.

4 Implementation

We implement a unified deep convolutional neural network to achieve the task of object detection and pixel-wise segmentation. The trained model, therefore, has two output heads: one for detecting the balls, robots, goalposts and the other for segmenting field, field lines and background. Since the pretrained Resnet-18 is used as the encoder, the parameters are frozen for the first 50 epochs and in the next 50 epochs both the encoder and the decoder are trained together with a smaller learning rate for the encoder. In order to speed up the training we not only store the heatmaps prior to training but also downsample the input images and their corresponding ground-truth followed by a gradual upsampling during the first 50 epochs. In the next 50 epochs, the model is trained with the images having their original size.

4.1 Feedforward

The input is fed to the encoder i.e. pre-trained Resent18 to extract object features. As a result, the input image is down-sampled. Therefore in the decoder, the image is up-sampled by the convolution transpose layer to reserve the image size preceeded by non-linear Relu and batch normalization. Upsampled image generated is then concatenated with the intermediate output of the encoder as shown in Figure 1. This is repeated for the other intermediate outputs of the encoder. In the final step instead of the convolutional layer, location dependant convolutional layer [1] is used. The resulting image is subject a common location bias. This is done not to increase the number of model parameters. Model has two output heads - detection head and segmentation head. Object detection head outputs a heatmap with different coloured gaussian blobs differentiating ball, robots and goalposts. The segmentation head outputs an image with different colours for background, field lines and field.

4.2 Backward and Training

Since the model has two output heads- Object Detection and Segmentation, there should be a way to differentiate to select the mode of training. To achieve this, we set a flag for that mode and backpropagate the corresponding loss. Parameters are updated using stochastic gradient descent. Adam optimizer with cyclical learning rate [3] is used. In case of object detection, we backpropagate Mean Squared Error loss whereas for segmentation, negative log likelihood loss along with total variational loss is backpropagated. Total variational loss applied to field and background channels decreases the number of false positives especially for field detection.

4.3 Results

4.3.1 Detection Head

The dataset is split into 80%,10%,10% resulting in 7081 image and heatmap pair for training, 886 for validation and 885 for testing. For evaluating the model various metrics are calculated using object-wise confusion matrix as summarised in Table 1. These are calculated for each - robot, ball and goal post.
In figure 2 are the object detection results by the model. Model also is able to detect objects in case of low brightness as seen in Figure 3.

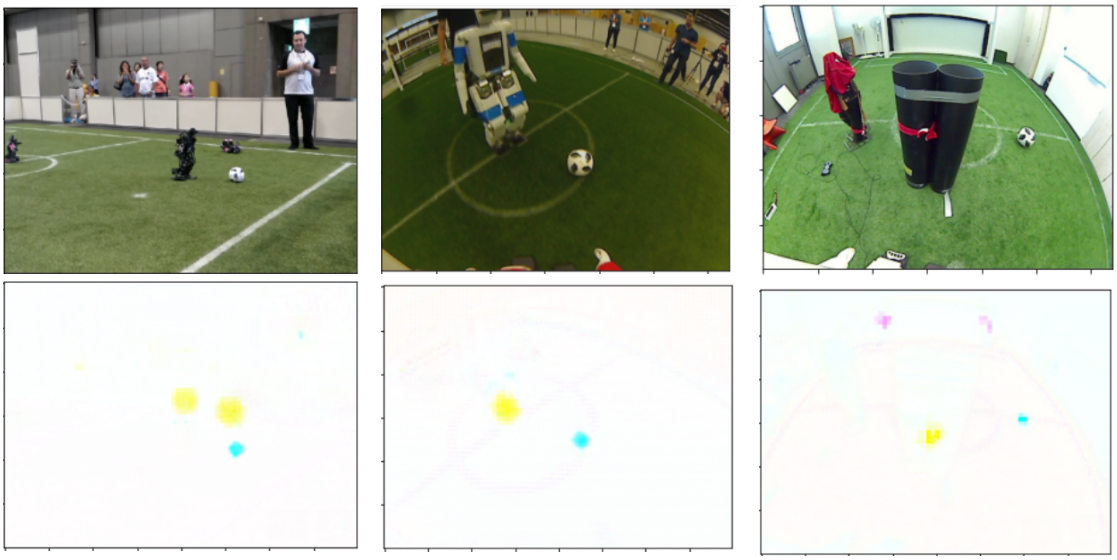

Figure 2: Object Detection Results, first row holds test images, while second displays model's results for ball, goalpost and robot

4.3.2 Segmentation Head

The dataset is split into 80%,10%,10% resulting in 953 image - ground truth with class label indices pairs for training, 120 pairs for validation and 119 pairs for testing. For

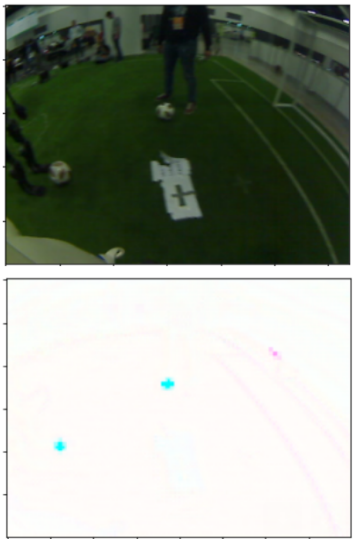

Figure 3: Model output in case of low brightness

Type	*F*1	*Accuracy*	*Recall*	*Precision*	*FDR*
Ball	0.8878	0.7983	0.8783	0.8976	0.1024
Goal Post	0.1083	0.0572	0.0577	0.8704	0.1296
Robot	0.5993	0.4279	0.4879	0.7766	0.2234
Total	0.5318	0.4278	0.4746	0.8482	0.1518

Table 1: Performance for object detection

evaluating the model, accuracy, and intersection over union (IOU) are calculated. These metrics are calculated for each - background, field and lines and also for the complete dataset. Table 2 summarises these results. It can be observed that the model is able to segment field and background with more accuracy than the lines, given that the line segmentation is a tough task.
Figure 4 are the segmented results by the model. The lines are not completely smooth but are segmented properly. Model also produces good results in case of low brightness as seen in Figure 5.

Type	*Accuracy*	*IOU*
Field	0.9872	0.9693
Lines	0.8972	0.8101
Background	0.9738	0.9614
Total	0.9527	0.9136

Table 2: Summary of model performance for segmentation

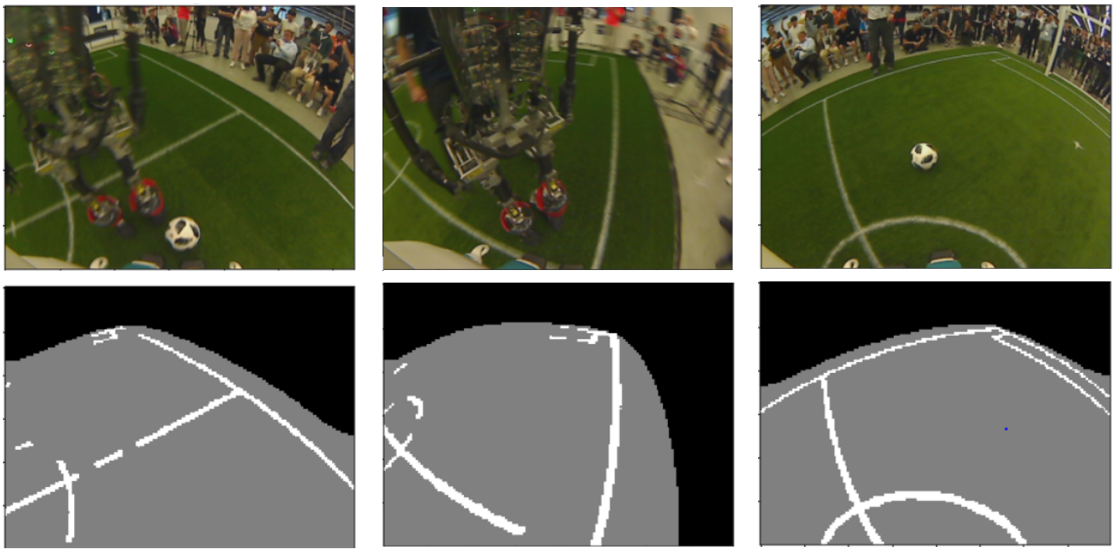

Figure 4: Segmentation Results, first row holds test images, while second displays model's results

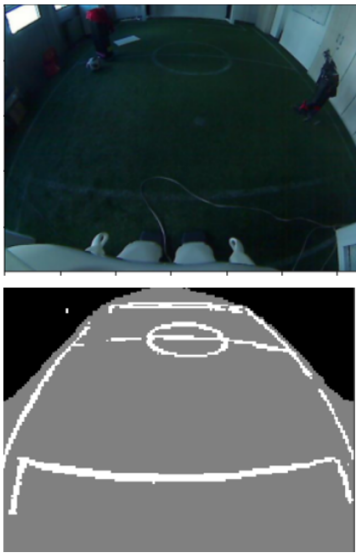

Figure 5: Model output in case of low brightness

5 Conclusion and Future Work

In this work, we implemented NimbroNet2 architecture and achieved comparable results for the segmentation task. In case of object detection, we observe that the thin goalposts are hard to be detected with our model implementation. In order to improve the model performance, we believe that retraining the model with a larger dataset containing more

goalposts examples will lead to better results. It can be observed that the model can perform both the tasks - object detection and segmentation in low brightness as well.

References

[1] Niloofar Azizi, Hafez Farazi, and Sven Behnke. Location dependency in video prediction, 2018.

[2] Diego Rodriguez, Hafez Farazi, Grzegorz Ficht, Dmytro Pavlichenko, Andre Brandenburger, Mojtaba Hosseini, Oleg Kosenko, Michael Schreiber, Marcel Missura, and Sven Behnke. Robocup 2019 adultsize winner nimbro: Deep learning perception, in-walk kick, push recovery, and team play capabilities, 2019.

[3] Leslie N. Smith. Cyclical learning rates for training neural networks, 2015.