Soccer Robot Perception

CudaVision - Learning Computer Vision on GPUs

Aakash Aggarwal and Suhaila Mangat Paramban

Universität Bonn
aakash@uni-bonn.de, Matrikelnummer: 3272727
s6sumang@uni-bonn.de, Matrikelnummer: 3277917

Abstract. In this paper, we present the implementation of the model in [1] for detection and localization of ball, goalposts, and robots along with pixel-wise segmentation of field and lines. The paper [1] discusses the visual perception model "NimbroNet2" (Figure 2) and different criteria they used for training the model. We provide a detailed explanation of pre-processing, training and post-processing steps. The results show that the model can be trained to attain high performance in a few hours of training. The model is build up by transfer learning from the base model "ResNet-18".

1 Introduction

Computer vision is the interpretation and understanding of the real world through the analysis of an image or a video sequence. In most cases, computer vision is focused on objects expected to appear in the scene. This project focuses on computer vision tasks, detection, localization and semantic segmentation in a RoboCup soccer environment. In robot soccer, authentic perception of soccer-related objects, like a soccer ball, robots, goalposts and field boundaries plays a huge role in the robot's performance.

Detection: Identification and localisation of objects in the image.
Localization: Identification of the location of an object in an image.
Semantic segmentation: Labelling each pixel in the image with the class of its enclosing object or region(pixel-wise classification).

Deep Learning-based convolutional neural networks for visual perception have witnessed far-reaching success in this field. In Robocup 2019, the new unified perception convolutional neural network "NimbRoNet2" assisted the team NimbRo AdultSize to remarkably improve the visual perception pipeline since RoboCup 2018. The motivation behind this project is the successful implementation of the fully convolutional network described in [1] that takes arbitrary size input, normalizes it and produces output images of fixed size with adequate inference and learning. We were given the exact labeled data set which gave team NimbRo the result indicated in [1]. Figure 1 shows one example of input images and corresponding target images.

Fig. 1: Example input and targets for detection and segmentation head

2 Network Architecture

The model consists of two output heads, one for object detection and the other for semantic segmentation. Both heads have 3 output channels. The detection head gives the location of the ball, robots, and goalposts while the segmentation head is for pixel-wise classification of lines, field, and background.

A pre-trained ResNet-18 has been used for encoding spatial features, whereas a comparatively smaller decoder has been employed. The shortened decoder was one of the modifications made in [1] to support computational limitations and real-time perception. The Global Average Pooling (GAP) and the fully connected layers in the model are removed because the primary focus of ResNet was recognition tasks. The layers are up-sampled by transposed convolution. Shared location-dependent bias in the last layer helped in utilizing the location-dependent features and reducing the number of parameters.

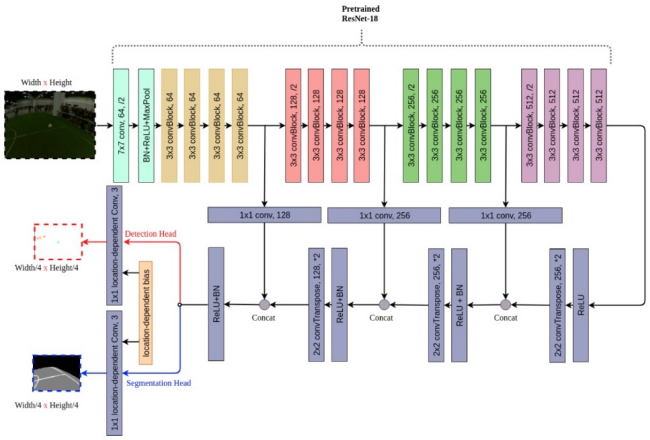

Fig. 2: NimbRoNet2 architecture[1]. There are 4 convolution blocks with 2 convolution layers each. Batch-normalization and ReLU activation function is applied to each convBlock output.

3 Related Models

This section briefly discusses a few network models which are related to NimbroNet2. The NimbRoNet2 encoder-decoder architecture is closely related to the to pixel-wise segmentation models like SegNet[2] and U-Net[3]. In the detection head, mean squared error loss was put to use as inspired by the SweatyNet[4].

3.1 SegNet

SegNet[2] was mainly developed for outdoor and indoor scene understanding by performing semantic segmentation. It's a fully convolutional neural network architecture. The architecture consists of a sequence of encoder layers which resembles 13 convolutional layers in the VGG16 network[6] and respective decoder layers. The encoder is made of convolutional layers followed by ReLU, max-pooling, and sub-sampling. The decoder up-samples the image using the max-pooling indices. This helps in re-training high-frequency details in the segmented images. In the original implementation of SegNet, the model was trained end-to-end using stochastic gradient descent.

3.2 U-Net

U-Net[3] is also developed for image segmentation. The network architecture is based on the fully convolutional network. It has been named U-Net because the contracting or the encoder part, and the decoder or the expansive part of the network are symmetric. U-Net was particularly developed as Convolutional Network for Biomedical Image Segmentation. It deals with the unavailability of a huge amount of training images in biomedical tasks. It predicts the class label of each pixel by providing a local region (patch) around that pixel. The advantages of using patches as input are, the network can localize and the training data becomes much larger than the number of training images. The encoder consists of the typical convolutional network made of multiple convolutions followed by ReLU and max pooling operations. It has the addition of skip connections that allow a more refined and precise output.

3.3 SweatyNet

SweatyNet[4] uses a single fully convolutional neural network to detect and localize the ball, opponent team robot, and other related objects and features on the soccer field. There are many similarities between NimbroNet and SweatyNet architectures. It also has an encoder-decoder design similar to SegNet[2]. The decoder was shorter than the encoder and in the encoder, after each convolution, batch normalization and ReLU activation are applied similar to NimbRoNet2. In the target, SweatyNet created heat-maps around the objects. There are three similar architectures of SweatyNet called SweatyNet-1, SweatyNet-2, and SweatyNet-3. They differ mainly in the number of layers and the number of parameters, to decrease the inference time.

4 Data pre-processing

We used the data set employed by team NimbRo to train the robots for RoboCup 2019. The data was collected from the Robocup matches in the previous years as well as manually in their small arena. Input colored images were normalized according to ImageNet Distribution with mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]. Finally, these images were also resized to a fixed size of (480,640) for creating batches of data.

For segmentation head, the target images were resized to (120,160) i.e. $\frac{1}{4}$ of fixed input image size. While resizing, nearest-neighbor interpolation was applied which takes the value of the closest lattice point. Interpolation is required to estimate the value of each pixel in the down-sampled image. Finally, the pixels were assigned to different classes(Background, Line, and Fields) as per the values of the target image.

For detection head, Gaussian blobs were created from the annotation files, given in the XML format. For Robot and Ball, center point was computed using all four points of the tagged rectangle, while for Goal Post bottom-middle point was measured by computing the center of the bottom edge of the rectangle. The center of the annotation file was given as mean, and covariance for each class was taken as per Table 1. Since predicting a canonical center point is more challenging for the robot, the variance for the robots has been kept as double of the variances for the ball and the goalpost. This assisted in less penalization of NimbroNet2 for not producing the exact target labels.

5 Training

Adam was chosen as the optimizer, which is known to perform well for deep networks. The learning rate was initialized to 0.001 and the betas(coefficients for computing running averages of gradient and its square) were given values 0.8, 0.999, which will control the intrinsic variation of the learning rate. These hyper-parameters were tuned by running several trials. Overall 8868 samples were collected for the blob detection and 1192 samples for segmentation head. Out of this, 80 percent of images from the blob data set and segmentation data set were used for training, while the remaining data was split equally for testing and validating our model. The network was trained for 300 epochs with each epoch running over 50 batches of blob and segmentation data sets. The batch size of 32 helped the network converge well.

Two different losses were applied for detection and segmentation heads each. For detection head, the mean squared error was applied with a weight of 0.03 whereas, for segmentation head, cross-entropy was utilized. Total variation loss was added to the output of all result channels except the line segmentation channel with weights 0.000002 for detection and 0.00003 for segmentation. Total variation loss is a measure of the noise in the images. It is calculated as the sum of the absolute differences for neighboring pixel-values in a given image. Adding total variation loss to the training loss is a regularization which ensures

spatial continuity and smoothness in the generated image to avoid noisy results. It helped in reducing false positives, notably in field detection by removing the rough texture of the image and thereby encouraging blob responses.

The training time was higher as compared to [1], as it took 8 hours on Nvidia GeForce RTX^{TM} 2080 GPU. To speed up the training, an effort was made to decrease the time required to load the data set. Reading and parsing an XML file for annotations and as well as input image path was slowing down the training process. Hence, the down-sampled target images and input image path were cached in the ram itself to quicken the loading. The network was trained by freezing the weights of the encoder network. The learning curve (train and validation) and output image examples from the test phase are shown in the Results.

6 Data post-processing

Estimation of centroids was required in the output image to get a single coordinate for each object. The output image was thresholded with appropriate values that were arbitrated by trying different figures. Then the centroids were calculated by making a contour around the predicted object. The thresholding numbers are reported in Table 1. Furthermore, when the distance between 2 centroids was less than or equal to 10 pixels, merging of the points was performed to get a new centroid. This helped in reducing false positives and improving the precision of our model. The fusion of points significantly enhanced the performance, especially for Robot detection. For segmentation head, only hard max was performed to classify each pixel and no other post-processing was required.

Object	Channel	Color	Variance	Error Tolerance (Pixel Difference)	Threshold (Contours)
Ball	0	Red	5	6	0.8
Robot	1	Green	10	12	0.9
Goalpost	2	Blue	5	8	0.88

Table 1: Important parameters used in pre-processing and post-processing.

7 Metric Calculation

For detection, if the difference between the true label and the predicted label was less than a certain tolerance, then it was marked as a true positive, else false positive. These error tolerances are given in Table 1. If an object was absent in prediction and as well as ground truth, then it was noted as true negative, while a missed detection was termed as a false negative.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall(RC) = \frac{TP}{TP + FN} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$FalseDetectionRate(FDR) = 1 - Precision \tag{4}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

- TP : True Positive
- FP : False Positive
- FN : False Negative

8 Results

The final performance of the network on unseen test dataset in summarized in Table 2 and Table 3.

Object	F1	Accuracy	Recall	Precision	FDR
Ball(Our Implementation)	0.996	0.994	0.996	0.996	0.004
Ball([1] Implementation)	0.998	0.996	0.996	1.0	0.0
Robot(Our Implementation)	0.997	0.996	0.994	1.0	0.0
Robot([1] Implementation)	0.979	0.973	0.963	0.995	0.004
Goal post(Our Implementation)	0.992	0.986	0.995	0.989	0.011
Goal post([1] Implementation)	0.981	0.971	0.973	0.988	0.011

Table 2: Results of the detection branch

Type	Accuracy	IOU
Field(Our Implementation)	0.987	0.968
Field([1] Implementation)	0.986	0.975
Lines(Our Implementation)	0.868	0.782
Lines([1] Implementation)	0.881	0.784
Background(Our Implementation)	0.974	0.959
Background([1] Implementation)	0.993	0.981

Table 3: Results of the segmentation branch

8.1 Input and Output Samples

Figure 3 shows few examples of outputs from both the heads.

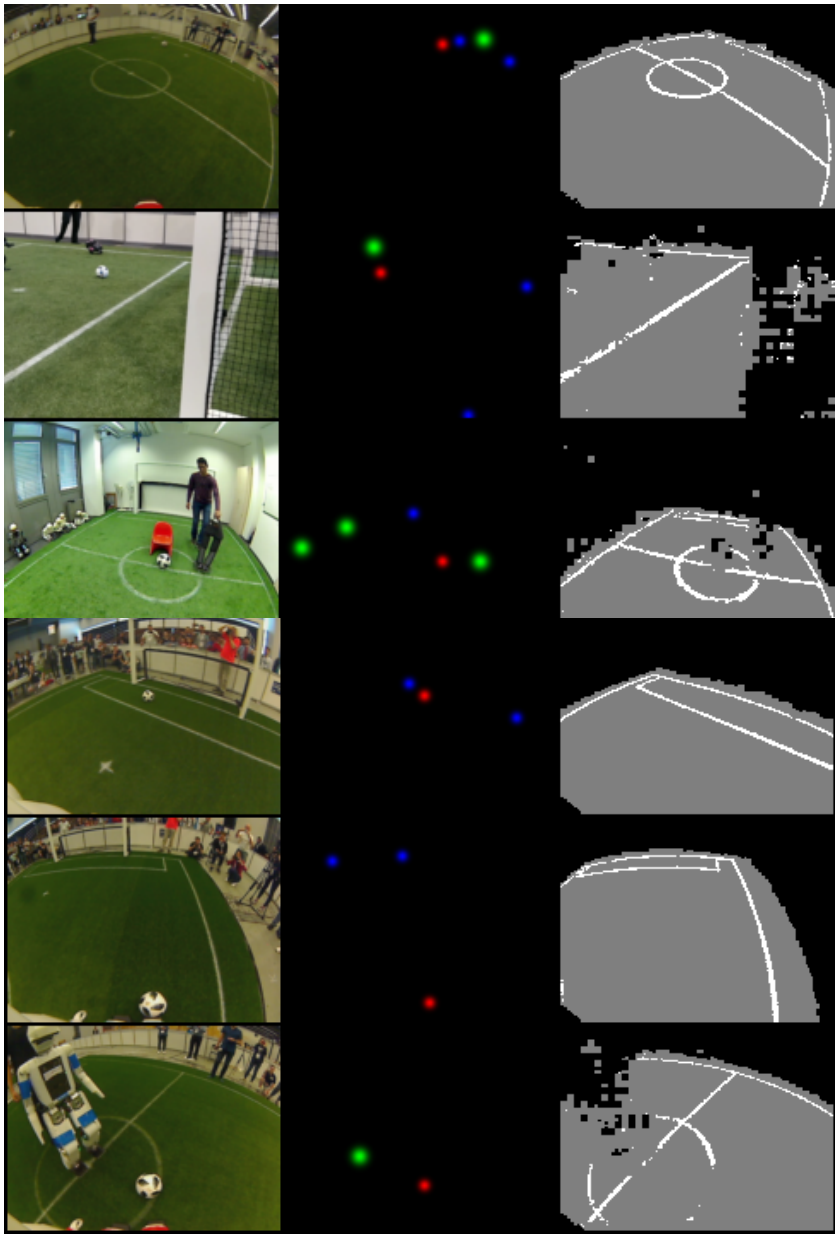

Fig. 3: Sample input images and corresponding outputs

9 Loss Curves

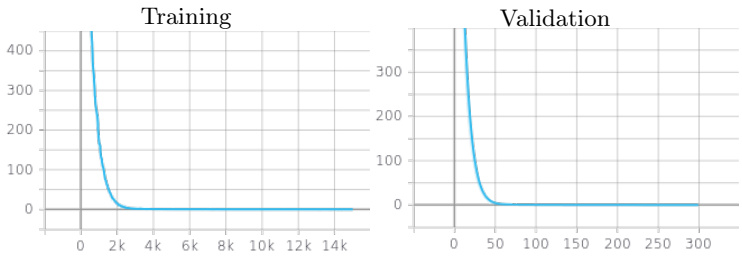

Fig. 4: Mean Squared Error Loss

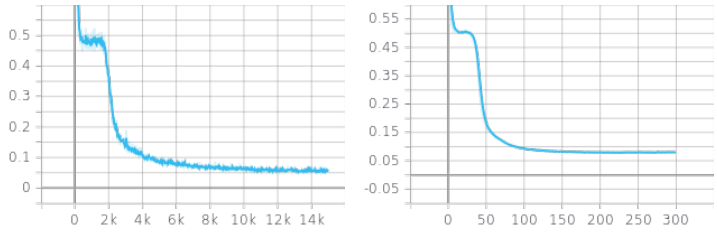

Fig. 5: Cross Entropy Loss

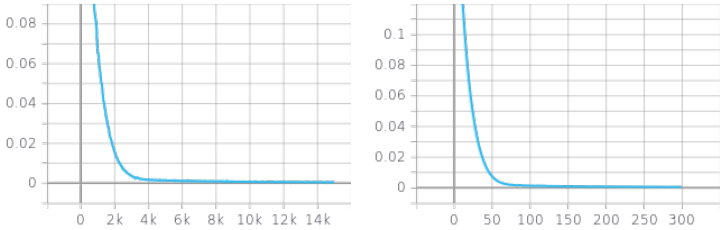

Fig. 6: Total Variation Loss (Blobs)

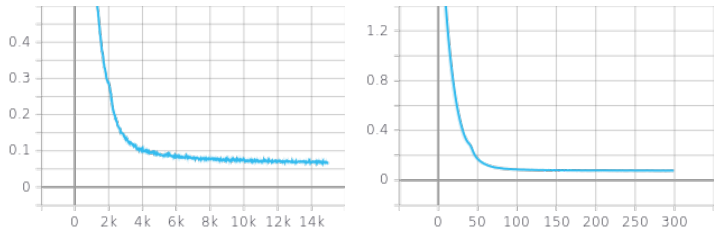

Fig. 7: Total Variation Loss (Segmentation)

10 Conclusion

In conclusion, it is shown from the implementation that the model can learn to detect soccer-related objects and robots. In Ball detection, accuracy equal to the [1] implementation was achieved and outperformed in the case of Robot and Goalpost. In segmentation, as compared to [1] implementation, similar accuracy was achieved for field classification. However, for lines and background, our results are few fractions behind [1]. With more training data and better tuning, the model can be improved further. Our training time was more in comparison to [1]. In winner implementation, the progressive image resizing helped in reducing the training time but the current implementation did not employ this technique which resulted in more time consumption.

11 Acknowledgement

We thank Hafez Farazi for his continuous guidance and helpful feedback for the successful completion of this project.

References

1. Rodriguez, Diego, Hafez Farazi, Grzegorz Ficht, Dmytro Pavlichenko, André 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." In Robot World Cup, pp. 631-645. Springer, Cham, 2019.
2. Badrinarayanan, Vijay et al. (2015). “SegNet: A Deep Convolutional Encoder- Decoder Architecture for Image Segmentation”. In: CoRR abs/1511.00561. “arXiv: 1511.00561.
3. Ronneberger, Olaf et al. (2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: CoRR abs/1505.04597. arXiv: 1505.04597.
4. Detection and Localization of Features on a Soccer Field with Feedforward Fully Convolutional Neural Networks (FCNN) for the Adult-Size Humanoid Robot Sweaty
5. Goodfellow, Bengio and Courville, 2016. Deep Learning.
6. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.