Detection and Tracking of the soccer ball
CudaVision - Learning Computer Vision on GPUs

Agawal Pratik Kumar, Kaur Amrit

Universität Bonn
s6pragar@uni-bonn.de, Matrikelnummer: 3056132
s6amkaur@uni-bonn.de, Matrikelnummer: 3055863

Abstract. In this paper, we are implementing the SweatyNet1 model, showed in Figure 2, from Humanoids RoboCup Workshop 2017 for detection and localization of a Soccer ball with feedforward Fully Convolutional Neural Networks (FCNN). The network architecture has a contracting path to capture the context and a symmetric expanding path that enables precise localization. We show that the network can be trained in few hours. After training the base model i.e SweatyNet1, a ConvLSTM layer is added on top of it and trained taking continuous sequence of frames. We show that adding convLSTM layer has a significant improvement in localizing the ball. And finally in the post processing step we are calculating Recall and False Detection Rate to check the accuracy of the models.

1 Introduction

The detection of the ball and other objects have become more and more challenging due to intraclass variations, occlusion and deformation. Therefore, achieving reliable results with traditional algorithms based on color segmentation and edge detection within captured images becomes increasingly difficult. We use convolutional networks mostly for classification tasks, where the output to an image is a single class label. But in many visual recognition tasks, the output should include localization i.e., a class label is supposed to be assigned to each pixel. Our key insight is to build fully convolutional networks that take arbitrary size input and produce correspondingly-sized output with efficient inference and learning.

In this project first we will create an encoder-decoder which will detect a soccer ball in the given image with FCNN. On top of it we will add a convLSTM layer for better localization. The input to the encoder and the output from the decoder is:

Input to the model: A RGB image of size 640 x 512 (height x width).
Output from model: An image of size 160 x 128 (height x width).

The only preprocessing step is to create target labels corresponding to each image. The object i.e the ball, is labeled at the center of its base. A normal distribution with a variance of $\sigma^2 = 6$ is centered around the label coordinates

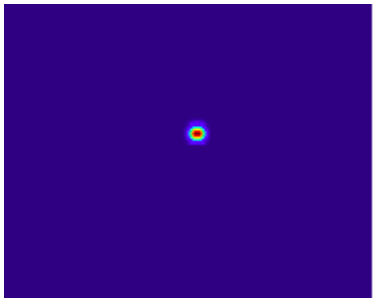


Fig. 1: Input and Target label of the model.

as it would be impossible to train with pixel-accurate prediction. We multiplied the whole matrix by 100 as probability values can get very small. The target label is down-sampled to the size of 128 x 160 which is $1/4^{th}$ of the size of image to reduce computational effort and costly data transfers from the GPU.

2 Related Work

In this section we discuss other approaches related to FCNN that gave us inspiration for our project.

Fully Convolutional Networks for Semantic Segmentation [1]: Semantic segmentation means labelling each pixel with the class of its enclosing object or region. Looking at the big picture, semantic segmentation is one of the high-level task that paves the way towards complete scene understanding. Fully convolutional networks was build that take input of arbitrary size and produce corresponding output with efficient inference and learning. FCNNs produce coarse output maps, for pixel-wise prediction, these coarse outputs needs to be connected back to the pixels hence they introduced deconvolution layers for upsampling. They observed that the 32 pixel stride at the final prediction layer limit the scale of detail in the upsampled output, so to make the learning through upsampling more effective and efficient and to retain the details of the image skip connections are added that combine the final prediction layer with lower layers with finer strides.

This approach serves as a starting point for our model. Skip connections exist between layers of the same resolution from the encoder to the decoder path to have finer grained spatial information from higher layers available in the decoder.

U-Net: Convolutional Networks for Biomedical Image Segmentation [2]: In this paper, fully convolutional network is trained with very few training images. Since deformation is the most common variation in tissues in biomedical, excessive data augmentation is applied to the training images which allowed network to learn invariance. In this work, pooling operators are replaced by up-

sampling operators hence increased the resolution of the output. To localize, high resolution features from the lower layers are combined with the upsampled output.

V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation [3]: It aims to segment prostate MRI volumes. The task is challenging because the prostate can have different appearance in different scans due to deformations and variations of the intensity distribution. This work proposed an approach to 3D image segmentation based on a volumetric, fully convolutional neural network. This model was also build based on the same idea as previous models. Pooling layers have been replaced by convolution layers. It has a compression path, which reduces the image size, and a decompression path which retrieves the original size of the image. Features extracted from early stages are forwarded through skip connections which helps in preserving the fine grained detail that would be otherwise lost in the compression path and due to this the quality of the final contour prediction was also improved.

3 Network Architecture

3.1 SweatyNet-1

Encoder: The dataset consists of images coupled with their target labels is feeded as input to the encoder. 12 convolution layers are applied in the Encoder with Batch-Normalization and ReLU as the activation function applied after each convolution. In the initial layer we are using 8 filters. 2 x 2 max-pooling operation with stride 2 is applied 4 times in between convolution layers for downsampling. After each max-pooling layer, the number of filters in convolution layer gets doubled. The encoder decoder network uses padding 1 throughout the convolution layers. In order to speed up training and to achieve more efficient parameter usage there are connections between the pooling layers as shown in the Figure 2.

Decoder: The output of the encoder is upsampled by a factor of 2 and passed to the Decoder. In the Decoder there are 6 Convolution layers with Batch-Normalization and ReLU as activation function applied after each convolution. Features from the encoder are combined with the upsampled output which helps in preserving the fine grained spatial information from higher layers available in the decoder as a result it will help in localization. And finally model generates the output as shown in Figure.1 predicting the location of the ball in the frame.

3.2 ConvLSTM model

A convLSTM model is build on top of Sweaty Net-1 Model. The architecture of this model is shown in Figure.3. The output from decoder coupled with its target label serves as the input to the convlstm model. 2 convolution layers are

applied with the kernel size 3 x 3. Filters of size 20 and 1 are used in the first and second convolution layer respectively. Then LSTM layer is applied which consists of input gate, forget gate, output gate and a memory cell. LSTM is designed to incorporate long-term dependencies. This feature of LSTM will help us in predicting where the ball can be in the next frame which when compared with the actual target helps in building a more powerful model.

A sequence of 10 frames of size 128 x 160 is passed as an input to this model at a time. The input is a 5-D tensor i.e batchsize x sequence length x channels x width x height of image.

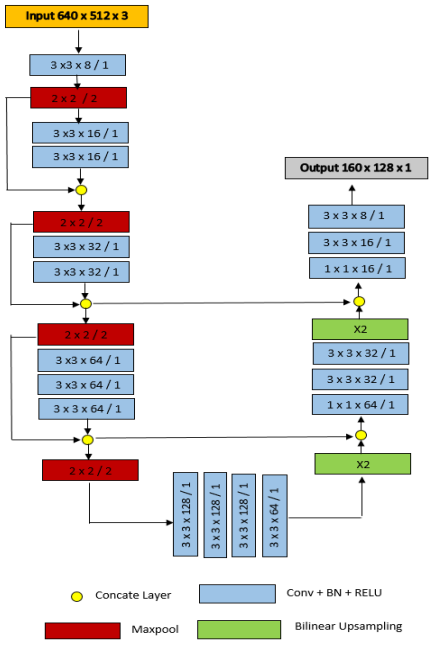


Fig. 2: Architecture of SweatyNet-1

4 Training

Models are trained by minimizing the mean squared error (MSE) between the output and the target label with the Adam-optimizer and with learning rate of 10^{-3}. As Adam-optimizer is used it will automatically adapt the learning rate in later stages of training.

The convolutions performed in each stage of encoder decoder uses kernels of size 3 x 3 with stride and padding 1. The SweatyNet-1 is trained end-to-end on the full dataset with a batch-size of 20.

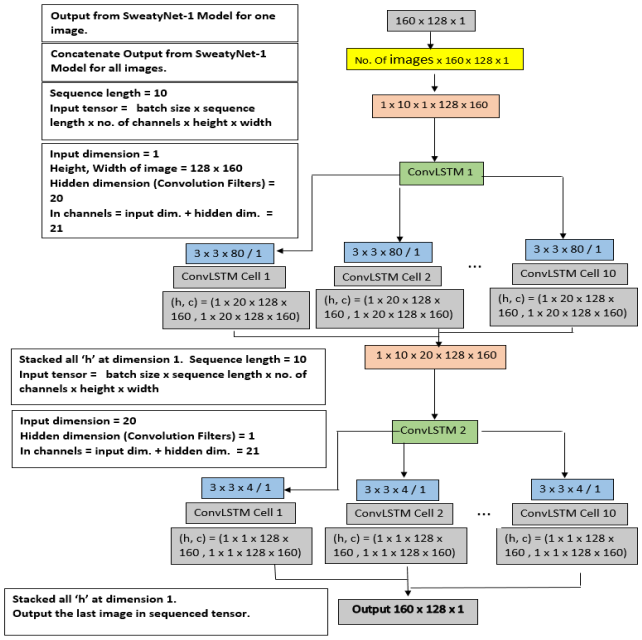


Fig. 3: Architecture of convLSTM

Dataset of 1876 images are divided in 70-30 ratio forming training set and validation set respectively.

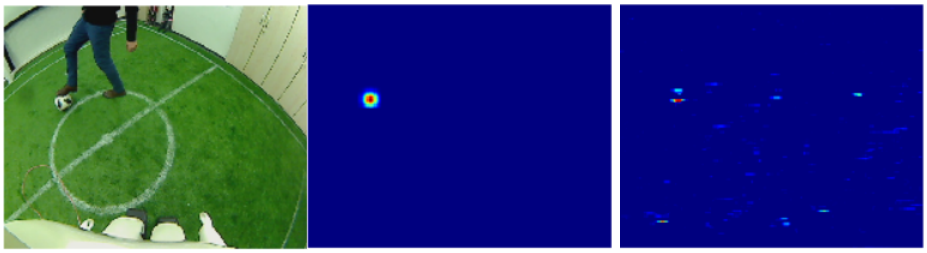
Once the base model is trained, we took another set of 935 images and we provide it to our encoder decoder trained model for 1 epoch in order to concatenate the encoder decoder output and target labels as one tuple. This newly generated dataset(This dataset is continuous sequence of frames) is loaded into our convlstm model and trained for 101 epochs for a sequence length of 10 and batch size 1. The architecture of convlstm model is as described in the Network Architecture 3.2 section.

5 Testing

The models are tested on a dataset of 257 images. It took less than a minute for the encoder-decoder as well as convLSTM model to generate results . The results generated from convLSTM model was better than the encoder-decoder results.

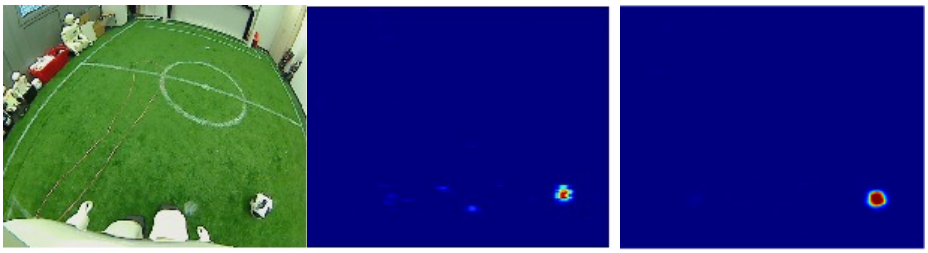
6 Post Processing

After running the model on test data, in the post processing step keeping the threshold value as 8, contours of the model outputs are determined using opencv.

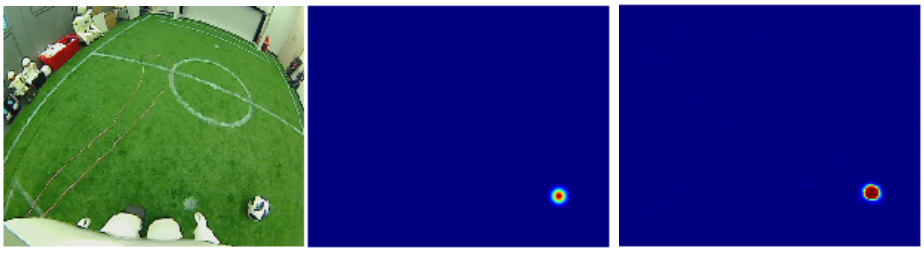


(a) Test image (b) Target Label (c) Bad SweatyNet-1 output

Fig. 4: Bad case scenario on Test Dataset



(a) Test image (b) sweaty output (c) convLSTM output



(d) Test image (e) sweaty output (f) convLSTM output

Fig. 5: Comparing output of convLSTM and SweatyNet-1

The threshold value is something which we need to determined in the testing phase by trying different values for a valid detection.

Taking the center value of the contour Recall and False Detection Rate has been calculated to measure the accuracy of the models. Recall is defined as true positives (TP) divided by the sum of TP and false negatives (FN). The false detection rate (FDR) is the number of false positives (FP) divided by the number of all detections.

$$Recall(RC) = \frac{TP}{TP + FN} \tag{1}$$

$$FalseDetectionRate(FDR) = \frac{FP}{FP + TP} \tag{2}$$

A detection is classied as true positive if a local maximum with sufcient magnitude is detected within a radius of six pixels around the coordinates of the label.

7 Results

All the datasets are taken from [5]. We trained and tested our models on Tesla K80 GPU provided by Google Colaboratory with 12.72 GB RAM. Training the SweatyNet-1 model for 101 epochs from scratch took around one hour to converge. And training the convLSTM model for 101 epochs took around one and half hours to converge. While testing it took less than a minute for both the models to generate results. From the results shown in Table 1 we can see that Recall of convLSTM model is higher than the SweatyNet-1 model which indicates that convLSTM surpasses the SweatyNet-1 model.

Model	Recall	FDR
Encoder Decoder	0.92	0.1
convLSTM	0.94	0.102

Table 1: Recall and FDR of models

8 Conclusion

We saw that implementing convLSTM on top of SweatyNet-1 improved the performance. The ball is detected very reliably. By looking at the results we can say that though we could able to locate the ball for maximum 94% of the cases but position of the ball can be investigated more accurately by tuning the parameters or by adding more convolution layers in the convLSTM model in order to estimate its influence on localization on the field of play.

9 Aknowledgement

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

References

1. E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 640–651, Apr. 2017.

2. O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," arXiv:1505.04597 [cs], May 2015, arXiv: 1505.04597. [Online]. Available: http: //arxiv.org/abs/1505.04597

3. F. Milletari, N. Navab, and S. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," CoRR, vol. abs/1606.04797, 2016. [Online]. Available: http://arxiv.org/abs/ 1606.04797

4. V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," CoRR, vol. abs/1511.00561, 2015. [Online]. Available: http://arxiv.org/ abs/1511.00561

5. Dataset Available: https://imagetagger.bit-bots.de/images/

6. https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef