Soccer Ball Detection using Fully Convolutional Neural Network

Lab CudaVision – Learning Vision Systems on Graphics Cards

Tridivraj Bhattacharyya

Rheinische Friedrich-Wilhelms-Universität Bonn
tridiv.b@uni-bonn.de, Matriculation Number: 3035538

Abstract. Object detection in Computer Vision has witnessed significant improvements recently through the use of Fully Convolutional Neural Networks. However even though these networks perform impressively while learning spatial data, they can do so only for single frames. To learn features over a sequence of frames, Convolutional LSTM architecture has provided good results. This is tested out in the following work while detecting a Soccer Ball based on a problem formulated along the rules for RoboCup.

1 Introduction

In recent times, deep learning has provided increased efficiency in complex computer vision problems such as semantic segmentation, object localization, detection and tracking to name a few. The systems to detect instances of an object in an image have evolved from brute-force template matching to seeking out blobs in probability heat-maps. Fully Convolutional Neural networks have been exceptional with regards to this. Despite that an additional problem lies in future frame predictions from sequences of frames. Convolutional LSTM or Convolutional GRU models have provided solutions for this by retaining information from frames over multiple time steps to draw an inference.

The problem for this project is formulated as an Object Detection task related to soccer ball. The RoboCup is a robotics soccer competition where the robot must detect objects on the soccer field such as ball, goal posts, opponents etc. and be able to play soccer. Our system is developed based on the criteria of this competition. We test three variations of the SweatyNet models [Schnekenburger et al., 2017] used by the Humanoid Robot Sweaty in the RoboCup Soccer AdultSize League. We also attempt to improve the model by implementating a Conv-LSTM on top of it so as to predict intermediate frames where the network is unable to detect the ball for multiple reasons.

2 Literature Review

2.1 Fully Convolutional Network for Semantic Segmentation

The term Fully Convolutional Network comes from the fact that the network does not contain any dense layer which are typically used for classification problems. This allows pixel-to-pixel mapping from images of any size to create Object Segmentation. This has been outlined by the work [Long et al., 2014], in which information from a deep, coarse layer is combined that of a shallow, refined layer to create a pixel-wise prediction for each Object.

2.2 U-Net

U-Net [Ronneberger et al., 2015] leverages the power of FCNN and an encoder-decoder architecture to implement a fast segmentation model. The architecture consists of the contracting (encoding) part to learn context and the expanding(decoder) part to localize the pixels. Addition of skip connections (concatenation from encoder to decoder) allows a more refined and precise final output. The U-Net [Ronneberger et al., 2015] utilizes data augmentation such as elastic deformations on a relatively small dataset to learn segmentation on bio-medical images.

2.3 Sweatynet

SweatyNet [Schnekenburger et al., 2017] uses the principles of both [Long et al., 2014] and [Ronneberger et al., 2015] to implement a feed-forward convolutional network which contains relatively less number of parameters, can be trained fast on a small dataset and is less prone to over-fitting. There are three variations of the SweatyNet model based on SegNet [Badrinarayanan et al., 2015], which introduced the encoder-decoder architecture for segmentation tasks. The SweatyNet 1 is shown in Figure 1.

The SweatyNet 2 and 3 models reduce the number of parameters to reduce learning time. The SweatyNet 3 differs from Figure 1 in using 1×1 convolutions instead of the dotted convolutions. These reduce the number of channels before being passed through the 3×3 ones. In all cases, a $640\times$ 512 resolution image is passed through the encoder and downsampled via maxpooling. In the decoder, the image is upsampled to a quarter of the original size and each upsampling layer is concatenated with the output of the encoder layer having same size so as to retain details from a higher resolution. The number of output channels represent the number of object classes learned by the network.

2.4 Convolutional LSTM

FCNNs while being useful in pixel-to-pixel segmentation, only works for single frame inputs. It does not learn anything with respect to temporal data. This is where Recurrent Neural Network architectures become advantageous. However,

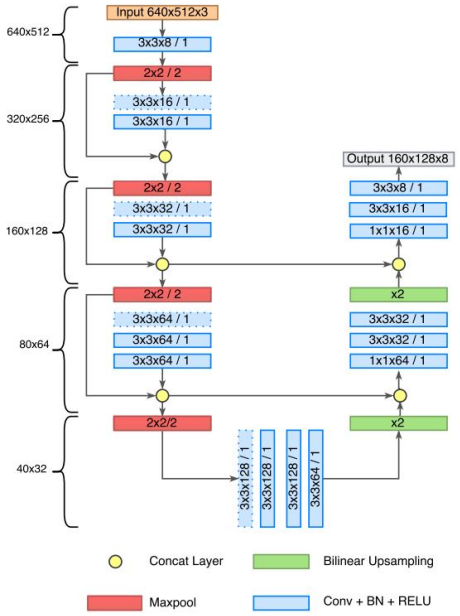


Fig. 1: Sweatynet 1 and 2(no dotted convolutions) [Schnekenburger et al., 2017]

traditional RNN models like Long-Short Term Memory and Gated Recurrent Networks are usually used for vector inputs such as word/sentence formation. LSTMs use three gates namely input, output and forget gate to process and remember temporal information. The spatial data is disregarded in this case and might cause ambiguity in the final output. [Shi et al., 2015] show that the problem of learning spatial data by LSTMs can be solved by using convolution operations on the state-to-state and input-to-state gates. Instead of using feature vectors, a 3-D input can be fed to the newly formed Convolutional LSTM network. The FC-LSTM equations are then recomputed as,

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \circ c_t + b_o) \\
h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$

where, i_t is the input gate, f_t is the forget gate, c_t is the cell state, h_t is the hidden state and o_t is the output. The notations $*$ and $\circ$ are used for convolution and element-wise matrix product respectively.

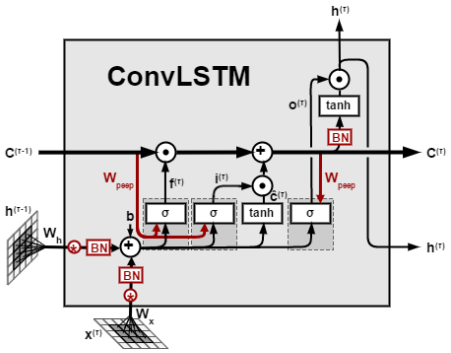


Fig. 2: Convolutional LSTM cell architecture[http://bsautermeister.de/research/frame-prediction]

3 Proposed Method

3.1 Problem Formulation

The idea here is to recognize a soccer ball on a field of play. Considering the rules of Robocup we constraint the problem to a single ball on a green pitch. The camera view is also assumed to be from the top of the robot. Therefore the network should learn to recognize the features of the ball possibly while being placed on a green turf. It learns to discriminate the ball against the background by modelling a probability heat-map where the background is 0. The inference is drawn as $Probability(foreground = ball|background = 0)$.

3.2 Pre Processing

The teacher signal for the network is created by fitting a normal with a variance of 4 at the location of the ball. By default, the values of the normal might be too low to learn a proper distinction from the background. Therefore it is also scaled up by a fixed amount to assist the learning process.

3.3 Post Processing

Once the network learns the features of the soccer ball, it provides a heat-map similar to the ground truth with a Gaussian blob approximately at the location of the ball. The output image is blurred, morphed and diluted to reduce background noise if any. This is further thresholded and OpenCV's contour detection algorithm is applied on top of it. All the detected contours are filtered with respect to a minimum and maximum area threshold. The bounding box coordinates for the contours passing the criteria are obtained and rectangles are drawn on the original image using these values.

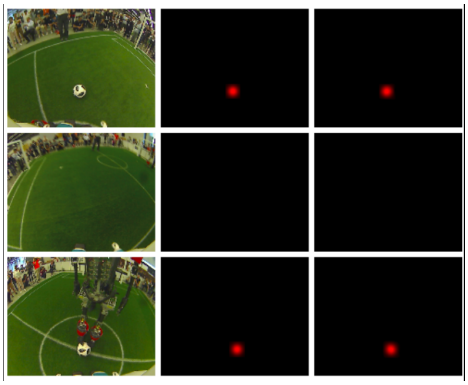


Fig. 3: Original Image(Left), Ground Truth(Middle) and Prediction(Right)

3.4 Metric Calculation

The performance of the network is ascertained via the Confusion Matrix. This in turn is used to calculate Recall and False Detection Rate

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{false detection rate} = \frac{\text{false positives}}{\text{false positives} + \text{true positives}}$$

A contour is detected as a true positive if its centre lies within a certain euclidean distance threshold of the original one. The threshold is set as the half the length or width of the ground truth bounding box (whichever is higher) as a pixel precision accuracy is almost impossible to achieve.

4 Implementation and Evaluation

4.1 Network Architecture

The SweatyNet models as in Figure 1 are used for the initial learning. The final output is converted to a single channel one because only a single class is learned. Three Convolutional LSTM cells with output channels of 32, 16 and 1 respectively are then appended to the SweatyNet model through which the inferred probability heat-maps are passed. The idea is inspired from [Siam et al., 2016] where Convolutional GRU is used instead.

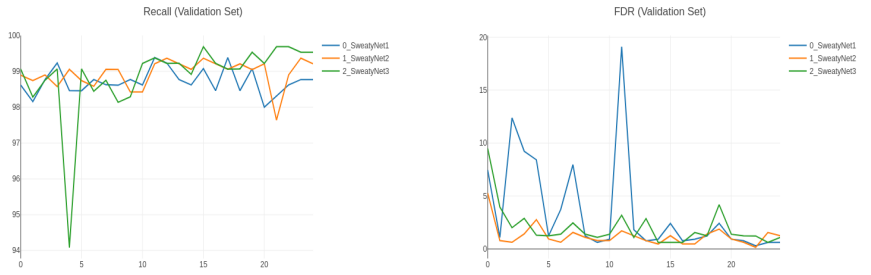
4.2 Training

The training is done in two phases. The overall dataset of 3626 images is randomly split into 2900 training images and 726 validation images. First the SweatyNet model is trained alone on the training set. The training data at this

time is shuffled and augmented with random horizontal flipping and color jitter. This makes the network reasonably robust to position, brightness and color shifts between frames. The network converges in a very few epochs while it learns to separate the background after the second epoch itself. It is still trained for 25 epochs to test against over-fitting. Once the SweatyNet is trained, the Conv-LSTM layers are added to the model and the SweatyNet weights are frozen. This allows creation of sequence of heat-maps as input for the Conv-LSTM section. Shuffling is turned off and care is taken for the training set to contain enough sequences with the ball being present in the image. Without this, the network fails to learn properly. The count of training and validation images remain the same. The learning rate in both cases is kept at 10^{-3} and the Adam optimizer is used. The loss is calculated over entire images using Mean Squared Error term.

4.3 Evaluation

The three variations of the SweatyNet model are first tested for speed and over-fitting. The three models display similar recall and false detection rates over 25 epochs. However models 2 and 3 are relatively faster on account of having less parameters. Testing is carried out on an unseen dataset of 937 images.



(a) Recall vs Epochs (b) False Detection Rate vs Epochs

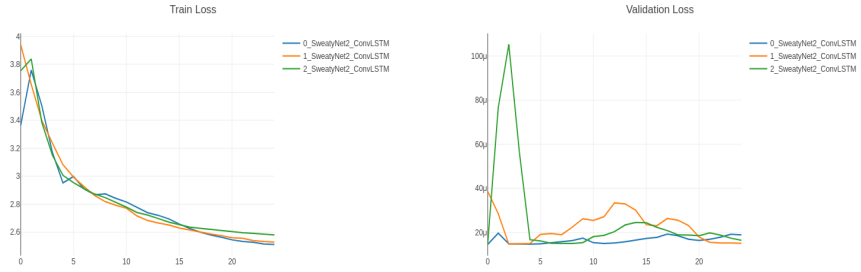
Fig. 4: Metrics on Validation Set

The SweatyNet with Conv-LSTM model is tested with different sequence lengths of 2 and 3. The network works very well in filling up intermediate frames which were missed by the original SweatyNet. The Recall improves along with an increased False Detection Rate. The reason is Convolutional LSTM enables detection of new contours in frames previously marked as False Negatives. Some of these newly detected contours are marked as False Positive on account of not matching the distance threshold criteria. It is seen that while training with Convolutional LSTM alongside transfer learning, the network overfits after a few epochs. Therefore early stopping is necessary. The metrics are provided in Table 1. Training the network with Convolutional LSTM from the ground up

and without freezing the weights was also tried but did not yield satsifactory results. The SweatyNet predictions in this case are not as good as that when trained without Conv-LSTM.

	No Conv-LSTM		Conv-LSTM (seq=2)		Conv-LSTM (seq=3)		Conv-LSTM (seq=4)	
	Recall(%)	FDR(%)	Recall(%)	FDR(%)	Recall(%)	FDR(%)	Recall(%)	FDR(%)
SweatyNet1	97.46	0.88	98.79	1.53	98.7	1.61	98.54	1.9
SweatyNet2	98.46	1.43	98.89	3.48	99.01	3.64	99.01	3.64
SweatyNet3	96.81	0.34	96.92	0.23	97.3	1.72	97.1	1.8

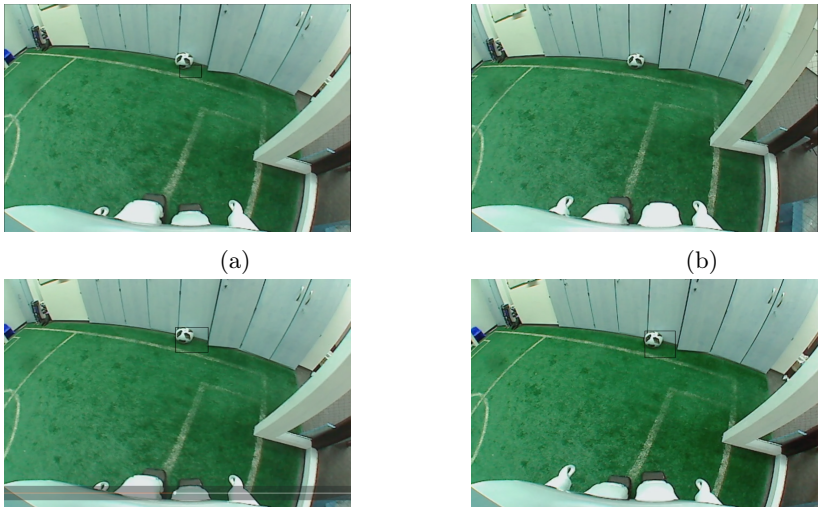
Table 1: Metrics on Test Set



(a) Training Loss vs Epochs (b) Validation Loss vs Epochs

Fig. 5: Train vs Validation Loss using Conv-LSTM shows signs of slight Overfitting (Blue-Sequence Length 2, Orange - Sequence Length 3, Green - Sequence length 4)

As a final generalization test, the models are tested on un-annotated frames obtained from an Youtube video with completely different camera parameters and viewpoint. Even here the network performs reasonably well as it is able to detect the ball at multiple scales and lighting changes. However, it struggles with a high number of False Positives especially in cases where other circular objects with similar features (color) to that of the ball is present. This difference is probably because the original SweatyNet discriminates against 8 classes while the present one only learns a single class. The problem of False Positives is amplified while using Convolutional LSTM as the network tries to compensate in consecutive frames when it detects other objects except the ball. The output for this test can look a bit non-intuitive as the detected Gaussian blobs sometimes have holes/artifcates inside them. This happens because the heat-map is upsampled for better visualization.



(a) (b)

(c) (d)

Fig. 6: Sequence of frames with detected bounding boxes. a) and b) are without Conv-LSTM and c) and d) are including Conv-LSTM

5 Conclusion & Future Work

In conclusion, the experiments prove the robustness of Encoder-Decoder architectures along with the claims of speed for the original SweatyNet [Schnekenburger et al., 2017]. The Convolutional LSTM also builds on the performance by recognizing temporal patterns in the data. There is still however room for improvements. It can be interesting to test with Convolutional GRU as it is known to be less computationally expensive. Moreover, [Siam et al., 2016] use the Convolutional GRU after the final downsampling layer. This has not been tried out for our model. It is also important to note, the problem here is restricted with multiple constraints. To relax them, a dataset with more variations in scene and ball instances is needed.

References

Badrinarayanan, Vijay et al. (2015). "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *CoRR* abs/1511.00561. arXiv: 1511.00561. URL: http://arxiv.org/abs/1511.00561.

Long, Jonathan et al. (2014). "Fully Convolutional Networks for Semantic Segmentation". In: *CoRR* abs/1411.4038. arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038.

Ronneberger, Olaf et al. (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597. arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

Schnekenburger, Fabian et al. (2017). "Detection and Localization of Features on a Soccer Field with Feedforward Fully Convolutional Neural Networks (FCNN) for the Adult-Size Humanoid Robot Sweaty". en. In: Proceedings of the 12th Workshop on Humanoid Soccer Robots, 17th IEEE-RAS International Conference on Humanoid Robots. Birmingham: IEEE, pp. 1 –6. URL: http://nbn-resolving.de/urn:nbn:de:bsz:ofb1-opus4-26557.

Shi, Xingjian et al. (2015). "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *CoRR* abs/1506.04214. arXiv: 1506.04214. URL: http://arxiv.org/abs/1506.04214.

Siam, Mennatullah et al. (2016). "Convolutional Gated Recurrent Networks for Video Segmentation". In: *CoRR* abs/1611.05435. arXiv: 1611.05435. URL: http://arxiv.org/abs/1611.05435.