Soccer Robot Perception

Lab Vision Systems: Learning Computer Vision on GPUs

Taouil, Ilyass
Shoeb, Youssef

Universität Bonn
s6iltaou@uni-bonn.de, Matrikelnummer: 3301516
s6yoshoe@uni-bonn.de, Matrikelnummer: 3316747

Abstract. In this report we present an implementation of NimbRoNet2, a unified perception convolutional neural network that was used in the 2019 RoboCup edition. The network architecture follows an encoder-decoder framework, with a pre-trained ResNet18 as the encoder, and a decoder with two output heads for pixel-wise segmentation and object detection. The network was able to achieve comparable results to the original implementation. We also propose an improved network that uses a more efficient pre-trained backbone that reduces by half the parameters, allowing a faster inference time, but without losing too much accuracy.

1 Introduction

The RoboCup is an annual competition where teams from all around the globe meet to participate in various challenges that vary in terms of tasks and robot sizes. In particular, this project is related to the RoboCup Humanoid League, whose final goal is to develop a team of humanoid robots able to win against the best football players on the planet by 2050.

The task's difficulty comprises two main challenges. The ability to play the football game at a high level (i.e. beating football world champions), and the technical challenges. Such technical challenges comprehend dynamic walking, running and jumping, but also state-estimation and visual perception.

This report focuses on the perception side of the challenge, which used to be limited in terms of difficulty in previous RoboCup editions by using specific color schemes to easily differentiate between the various components involved. These, however, have been gradually removed from the competition and additional layers of difficulty were introduced, including a larger field in the 2019 edition of the RoboCup.

A robust perception system has to deal and overcome challenges like scaling due to the larger size of the pitch (i.e. ball, goalposts, and opponents are further away), lighting conditions, viewing angles and distortions.

In the following report we present the unified perception convolutional neural network NimbroNet2 [8] that performs object detection for balls, goalposts, and robots, as well as pixel-wise classification of lines and fields. The presented network won all adult size competitions in the 2019 edition of the RoboCup.

2 Related Work

NimbroNet2 [8] is a deep learning based perception pipeline that uses an encoder-decoder architecture similar to SegNet [2] and U-Net [9] for both pixel-wise classification and detection. NimbroNet2 significantly improves upon the 2018 RoboCup version and SweatyNet [10].

2.1 SegNet

SegNet [2] is a deep convolutional neural network for semantic pixel-wise segmentation primarily targeted to road scene understanding. The network uses an encoder-decoder type of architecture followed by a pixel-wise classification layer. The architecture of the encoder used is topologically the same as the VGG16 [11] network, except for the removed fully connected layers. The decoder network used by SegNet maps the low resolution encoder feature maps to full input resolution features. The non-linear up-sampling process of the decoder uses the computed pooling indices of the encoder during the respective max-pooling steps which removes the need to learn the up-sampling process. The network is trained end-to-end using SGD [3].

2.2 U-Net

U-Net [9] is an end-to-end deep convolutional network for biomedical image segmentation. The network consists of a so called contracting path and expansive path. The contracting path follows the typical structure of a convolutional network, where there is a hierarchy of convolutions followed by ReLU and max-pooling in order to downsample the input. Every downsampling step doubles the number of features. The expansive path upsamples the features maps performing an up-convolution in order to half the feature channels. The expansive feature maps are concatenated with the respective cropped feature map from the contracting path. Finally a 1D convolution is used to map the high-dimensional feature vector to the desired number of classes.

2.3 SweatyNet

SweatyNet [10] is a deep convolutional neural network that detects the position of the ball, goalposts, opponents, and line elements reliably. The network uses an encoder-decoder architecture where the decoder is shorter than the encoder due to computational limitations. In total the network consists of twelve convolutional layers, four max-pooling layers, six transpose-convolution layers, and two bilinear upscaling layers. ReLU and batch normalization are also used in the network. SweatyNet was able to achieve a recall of 99%, and a false detection rate of 1.2 for the detection of the ball, and goalposts. The network was used as the perception network for Sweaty, which was the finalist of the 2017 RoboCup in Nagoya.

3 Approach

3.1 Dataset Preparation

In the following section we discuss the steps taken to prepare the respective datasets for training the detection and segmentation task. For both the segmentation and detection task, the dataset was split with a ratio of 70:15:15 for training, validation and testing respectively. All images were resized to the same size, and normalized according to the ImageNet[5] distribution before being passed through the network.

3.2 Detection

The detection dataset consisted of 8857 samples, with each sample containing an input image and its labeled target. The input is a jpg image taken from the onboard camera of the humanoid, while the target is a xml file with each detected object labeled with its class label and bounding-box coordinates.

For training the network for the detection task, the xml files were parsed for each input image, and a probability map was created as the target output of the network. The probability map consisted of three channels, one for each object (ball, robot, goalpost). Each channel stores a Multivariate Gaussian distribution for each instance of the object, with the mean equal to the center of the bounding box, and a fixed covariance. For the goalposts and the robots, the center was taken as the bottom-middle point of the bounding box. For the robots the covariance was further increased by a factor of 0.5 as the annotation of the canonical center point for the robot is usually more difficult.

3.3 Segmentation

The segmentation dataset consisted of 1192 samples. The inputs are jpg images from the camera viewpoint of the humanoid, and the targets are color segmented targets png images.

For training the network for the segmentation task, the targets were encoded using a three dimensional vector representing the probability of each pixel belonging to a respective class (i.e. field, line or background). Any object on the field other than the lines, was discarded and labeled as part of the field.

3.4 Network Architecture

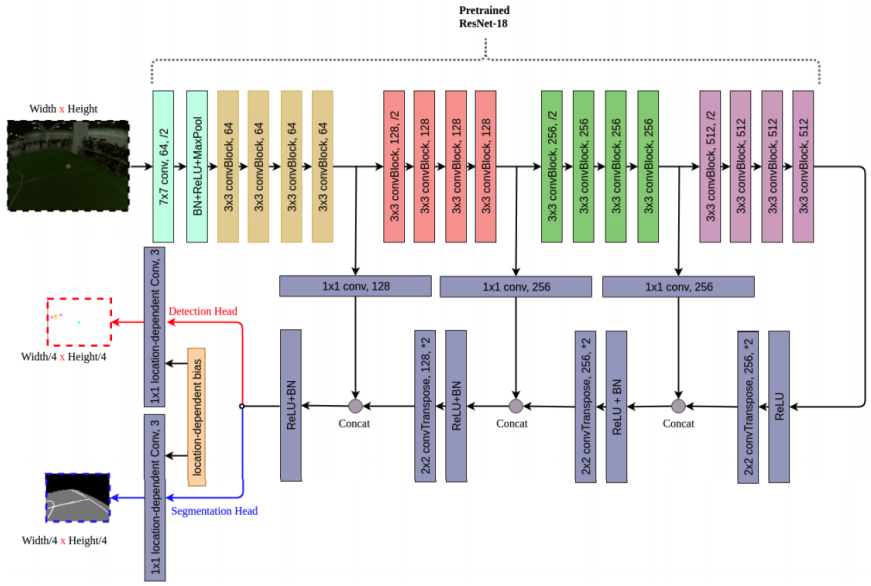

Fig. 1. NimbroNet2 Architecture [8]

Fig. 1 shows the overall architecture of NimbroNet2. The network follows the pixel-wise segmentation encoder-decoder architecture used in SegNet [2] and U-Net [9]. A pre-trained ResNet-18 [6] with the Global Average Pooling (GAP) and the fully connected layers removed is used as the encoder. The decoder consists of a ReLU activation function followed by three transpose-convolution layers, each transpose-convolution is followed by a ReLU activation function and batch normalization. Before each ReLU activation function an output from the encoder is concatenated with the output of the transpose convolution. The decoder has two output heads, one for detection, and one for segmentation. Each head uses a location-dependent convolution layer [1] with a shared learnable bias. There are also skip connections present between the layers of the encoder and the decoder to provide high-resolution details of the input to the decoder.

The detection head produces a probability map for the locations of the balls, robots, and goalposts, each in a separate channel. The segmentation head outputs the probability of each pixel belonging to one of the following: field, line, or background.

In total the encoder has five downsampling blocks and the decoder has three upsampling blocks, making the outputs of the model one fourth of the original inputs dimension. To allow for a consistent output size, it is important to select the input images such that both the height and the width are divisible by 32.

3.5 Training

To train the model, progressive image resizing inspired from Brock et al. [4], and Yosinski et al. [14] was used. For the first fifty epochs, the encoder part was frozen, and 128x160 images were used so that the inaccurate random initialization could be quickly corrected by learning from large batches of small images. For the following fifty epochs both encoder and decoder were trained jointly on images of size 256x320. In the last 50 epochs, the full sized 480x640 images were used to train all parts of the model to learn the small details.

An Adam optimizer [7] was used to train the model, and a cyclic learning rate [12] was used to determine the learning rate. For the cyclic learning rate, a lower base learning rate and maximum bound was used for the encoder in comparison to the decoder, with the intuition that a pre-trained model will need less time to converge.

For the segmentation head, cross-entropy loss was used to measure the difference between the output of the model and the target. To account for class imbalances in the segmentation dataset, the classes were rescaled with the following weights [0.4, 0.7, 0.9] for field, background, and lines respectively. Total variation loss was added to the field and background segmentation channels as a regularization term to the image. This pushes neighboring pixels to have similar values. For the detection head, the mean squared error was utilized to measure the difference between the output and the target probability map.

A unified training approach was used where each batch consisted of roughly 10% segmentation samples and 90% detection samples. The batch percentages for each task reflect the amount of data contributed by each dataset once both datasets were combined. For each batch, segmentation and detection images were passed through the network, and the individual losses were combined to give one unique loss which is then optimized by PyTorch's automatic differentiation engine.

4 Results

In the following section the quantitative and qualitative results obtained for the training process of the detection and segmentation are presented, along with the explanations of the post-processing step took in order to obtain such results.

4.1 Detection: Post-Processing

The output received from the detection head of the network is a three channel probability map, where each channel contains probability distributions overlaid on the detected object with a certain mean and covariance. Therefore, in order to quantitatively define the accuracy of the detection head, the target and predicted outputs for the respective image need to be compared. To do so, computer vision techniques and sub-pixel processing methods are used. More precisely, both target and prediction are processed using *findContours* methods from the OpenCV

library in order to obtain the $2D$ coordinates of the local-maxima for each object in each channel. Furthermore, the predicted output received by performing the detection forward pass with the raw image was noisy, thereby resulting in a lot of fall positives. Hence, to deal with the noisiness of the prediction a value truncation was performed where all detection with a probability below 0.5 where set to 0.

4.2 Detection: Results

Fig. 2 shows some randomly selected results from the detection head on the test set. We compare the input image and target probability map with the output from the model.

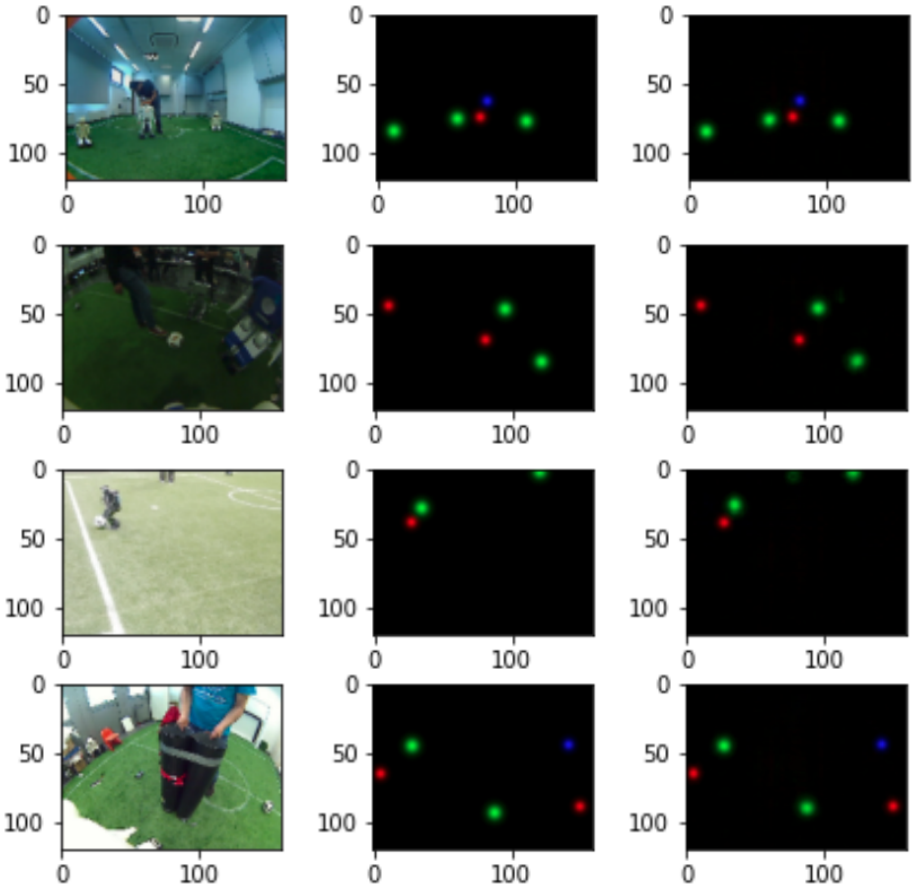

Fig. 2. Object Detection Results. From left to right: input image, target, prediction.

Table 1 shows the *F1* score, *accuracy*, *recall*, *precision* and *FDR* for the detection task compared to the original implementation. The achieved results are not as accurate as the original implementation, but still acceptable.

Detection Results					
Type	F1	Accuracy	Recall	Precision	FDR
Ball (Ours)	0.984	0.973	0.976	0.993	0.007
Ball (NimbroNet2)	**0.998**	**0.996**	**0.996**	**1.0**	**0.0**
Goal (Ours)	0.962	0.938	0.944	0.982	0.018
Goal (NimbroNet2)	**0.981**	**0.971**	**0.973**	**0.988**	**0.011**
Robot (Ours)	0.943	0.918	0.911	0.978	0.022
Robot (NimbroNet2)	**0.979**	**0.973**	**0.963**	**0.995**	**0.004**
Total (Ours)	0.963	0.943	0.944	0.984	0.016
Total (NimbroNet2)	**0.986**	**0.986**	**0.977**	**0.994**	**0.005**

Table 1. Object Detection Performance.

Fig. 3 shows the training loss progression for the detection training throughout the 150 epochs. The spikes in the plot below represents the image upscaling, when the network starts training on higher resolution images.

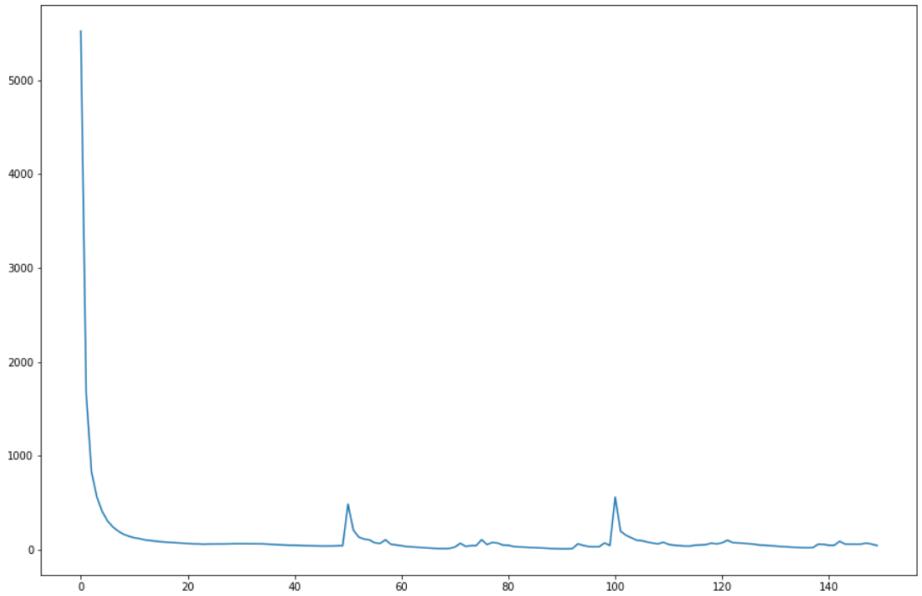

Fig. 3. Detection Training Loss.

4.3 Segmentation: Results

Fig. 4 shows some randomly selected results from the detection head on the test set. We compare the input image and target labels with the output from the model.

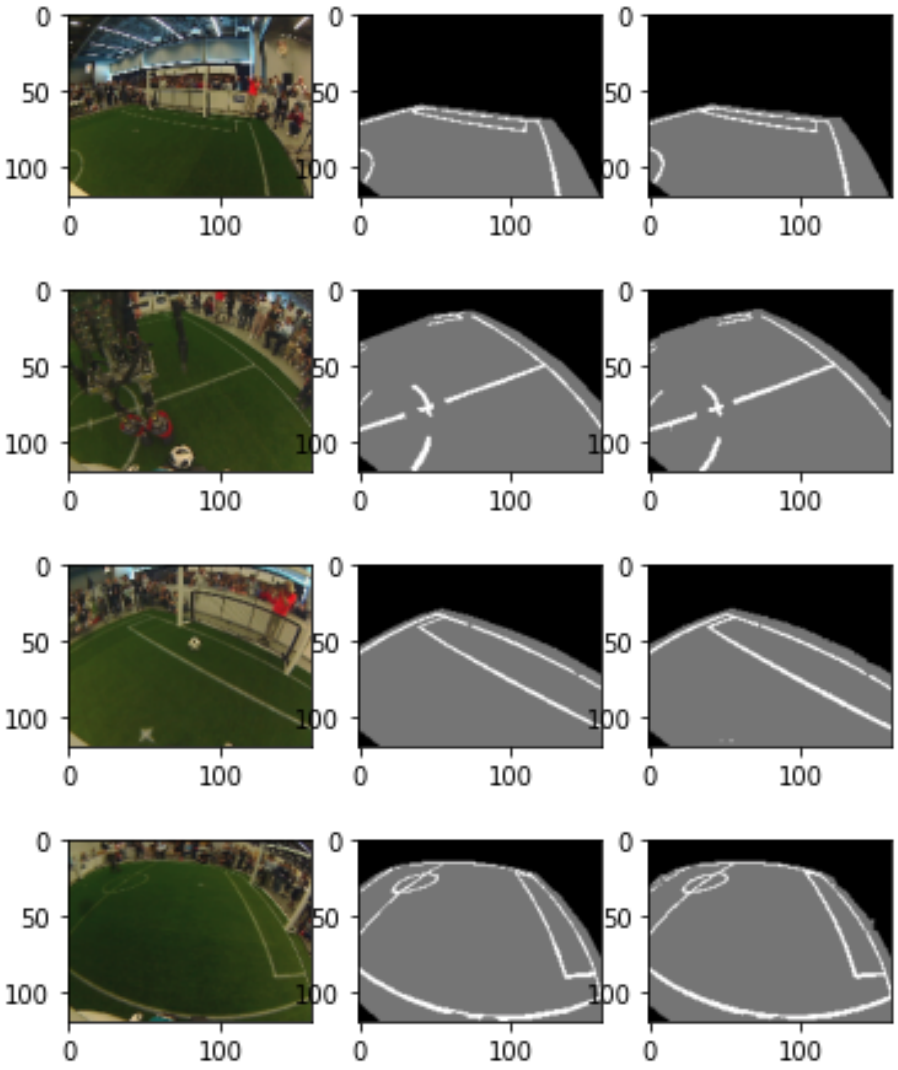

Fig. 4. Image Segmentation Results. From left to right: input image, target, prediction.

Table 2 shows the *accuracy*, and *IOU* for the segmentation task compared to the original implementation. Our model was able to outperform the original implementation in the overall *accuracy* and *IOU*.

Segmentation Results		
Type	Accuracy	IOU
Field (Ours)	**0.990**	**0.978**
Field (NimbroNet2)	0.986	0.975
Lines (Ours)	**0.921**	**0.815**
Lines (NimbroNet2)	0.881	0.784
Background (Ours)	0.988	**0.983**
Background (NimbroNet2)	**0.993**	0.981
Total (Ours)	**0.986**	**0.926**
Total (NimbroNet2)	0.953	0.913

Table 2. Image Segmentation Performance.

Fig. 3 shows the training loss for the segmentation training throughout the 150 epochs. The spikes in the plot below represents the image upscaling, when the network starts training on higher resolution images.

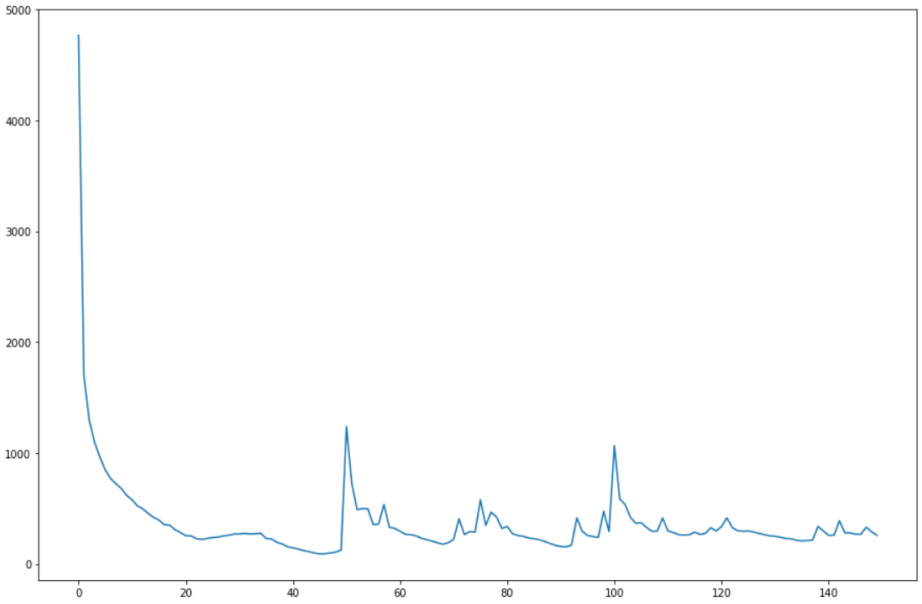

Fig. 5. Segmentation training loss.

5 Improvements

Perception systems like NimbroNet2 are needed to be used on resource constrained systems which require a real-time usage of such algorithms to complete the required tasks. The ResNet-18 backbone used in the unified-model contains 11,689,512 parameters which adds additional overhead to the system when it comes to real-time usage on resource constrained systems. We propose to use a pre-trained EfficientNet-B0 [13] backbone instead in order to improve the efficiency of the network, without losing much on accuracy.

The EfficientNet-B0 backbone decreases the number of parameters down to 5,490,596. In order to use the same decoder, skip connections were added between the EfficientNet-B0 backbone and the decoder at the same spatial resolution as NimbroNet2.

Table 3, and Table 4 respectively show the detection and segmentation results compared to the original model trained by us using a ResNet-18 Backbone.

Detection Results					
Type	F1	Accuracy	Recall	Precision	FDR
Ball (EfficientNet-B0)	0.980	0.964	0.971	0.989	0.011
Ball (ResNet-18)	**0.984**	**0.973**	**0.976**	**0.993**	**0.007**
Goal (EfficientNet-B0)	0.915	0.866	0.885	0.948	0.052
Goal (ResNet-18)	**0.962**	**0.938**	**0.944**	**0.982**	**0.018**
Robot (EfficientNet-B0)	0.929	0.897	0.914	0.945	0.055
Robot (ResNet-18)	**0.943**	**0.918**	**0.911**	**0.978**	**0.022**
Total (EfficientNet-B0)	0.941	0.909	0.923	0.961	0.039
Total (ResNet-18)	**0.963**	**0.943**	**0.944**	**0.984**	**0.016**

Table 3. Object Detection Performance.

Segmentation Results		
Type	Accuracy	IOU
Field (EfficientNet-B0)	0.988	0.968
Field (ResNet-18)	**0.990**	**0.979**
Lines (EfficientNet-B0)	0.880	0.794
Lines (ResNet-18)	**0.921**	**0.816**
Background (EfficientNet-B0)	0.975	0.965
Background (ResNet-18)	**0.988**	**0.983**
Total (EfficientNet-B0)	0.980	0.909
Total (ResNet-18)	**0.986**	**0.926**

Table 4. Image Segmentation Performance.

6 Conclusion

In this report an implementation of the original NimbroNet2 network was presented, describing the dataset preparation steps, the training approach, and the respective results for both detection and segmentation compared to the original results stated in [8]. The presented network achieves similar results for both the detection and segmentation tasks, while improving on the average accuracy and *IOU* of the segmentation head compared to the original paper results. However, a more efficient version of the network is presented where a pre-trained EfficientNet-B0 [13] backbone is used instead of ResNet-18 [6]. This reduces the encoder parameters by more than half, ultimately speeding up the inference process.

References

[1] Niloofar Azizi et al. "Location dependency in video prediction". In: *International Conference on Artificial Neural Networks*. Springer. 2018, pp. 630–638.

[2] Vijay Badrinarayanan et al. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[3] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[4] Andrew Brock et al. "Freezeout: Accelerate training by progressively freezing layers". In: *arXiv preprint arXiv:1706.04983* (2017).

[5] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[6] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

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

[8] Diego Rodriguez et al. "RoboCup 2019 AdultSize winner NimbRo: Deep learning perception, in-walk kick, push recovery, and team play capabilities". In: *Robot World Cup*. Springer. 2019, pp. 631–645.

[9] Olaf Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[10] Fabian Schnekenburger et al. "Detection and localization of features on a soccer field with feedforward fully convolutional neural networks (FCNN) for the Adult-size humanoid robot Sweaty". In: *Proceedings of the 12th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots, Birmingham*. sn. 2017.

[11] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[12] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2017, pp. 464–472.

[13] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[14] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *arXiv preprint arXiv:1411.1792* (2014).