Cudavision Lab SS 2019: Final Report

Saikat Roy and Albert Gubaidullin

Universität Bonn
{saikatroy, s6alguba}@uni-bonn.de, Matrikelnummer: 3063935, 3214542

Abstract. The project involves designing an Encoder-Decoder based Deep Convolutional Neural Networks for humanoid robot part detection. Tranfer learning is used on the encoder to enable us to train the model with a relatively small number of training examples. We provide a dataset of 450 images and their corresponding labels for four type of body parts of humanoid robots in multiple poses and viewing angles for training the Deep Neural Network. The final model is trained on a combined dataset of all participating teams in the Lab. The model is evaluated on a hold out dataset of nearly 400 images while the remaining are used for training and validation.

1 Introduction

The major goal of this project is to extend the technique in [1] to the task of detecting humanoid robot body parts. Instead of the 3 class problem of detecting a football, goal posts and a robot in the previous work, this one extends the problem to detecting the problem to a 4 class scenario of detecting *head, trunk, arms* and *legs* of the humanoid robot. The report continues with a description of the dataset and then moves onto a detailed description of the methodology. This is followed by a short elaboration on the technical implementation and discusses the results and then finally concludes with possible improvements that can be plugged into the framework.

2 Dataset

One of the targets of the project was to provide a set of labelled data to use for the problem of detecting humanoid robot body parts. At this stage of the project we provide (and also use ourselves) a set of 450 RGB images as well as a set of JSON files for each corresponding image with a small bounding box around usually a central point of the robot body part in question. The original images have a resolution of 720×1280 pixels which are resized to 480×640 (VGA) resolution before training. Each JSON file has separate bounding boxes defined for 4 classes - *Head, Trunk, Arms* and *Legs*. All classes can have none or multiple entries for bounding boxes based on various conditions in the image such as multiple robots, partially occluded parts and so on.

Due to the difficulties in training our model with just our own gathered data, we also incorporate the data gathered by the other teams in this session of the

CudaVision lab for training and evaluation. We upscale or downscale raw images from each team as necessary while also doing the same for the output maps. We end up with a dataset of 1385, 197 and 397 images respectively for the training, validation and test sets respectively. The use of a fixed random seed enables our splitting function to create exactly similar splits across various runs and allows for fair evaluation across multiple runs of our algorithm.

3 Methodology

The central methodology is based on using the technique described in [1] where an Encoder-Decoder based Deep Convolutional Neural Network (DCNN) architecture is proposed. A similar architecture for semantic segmentation while using pretrained deep models was proposed in [2]. The general idea is to use Transfer Learning [3] to initialize the Encoder based on weights learned on the Imagenet Dataset [4] and extend this partially pretrained network into a DCNN based humanoid robot body part detector. We elaborate on the variations of the architecture, preprocessing, training and post-processing techniques used in the following sub-sections:

3.1 Architecture

An Encoder-Decoder based Deep Convolutional Neural Network architecture similar to [1] as illustrated in Fig. 1 is used in this work as the machine learning model. A pretrained ResNet-18 [5] on the Imagenet dataset is used as base model for the encoder. We attempt recreate the previous work including the use of a similarly 4x downsampled resolution in our output map as the previous work. However, one small distinction we noticed that we needed to perform in our model was balancing the dimensions of the tensors in the encoder to decoder residual connections. We zero-pad the smaller tensor before concatenation of each residual connection. Only the final tensor is cropped for ease of comparison with the target output tensor.

3.2 Preprocessing

There are a number of preprocessing steps as part of the learning pipeline to incorporate the lower-resolution output map as well as to preprocess the raw images for use in the pretrained encoder. The following steps are followed for the low resolution true output map creation:

1. A 3D tensor consisting of the original image resized to VGA resolution is used as the input to the network.
2. The output tensor consists of a tensor which is 0.25 times the dimensions of the image tensor with 1 channel per class is created with the center of the bounding box for a robot body part being set at 1.0 while the rest are set at 0.

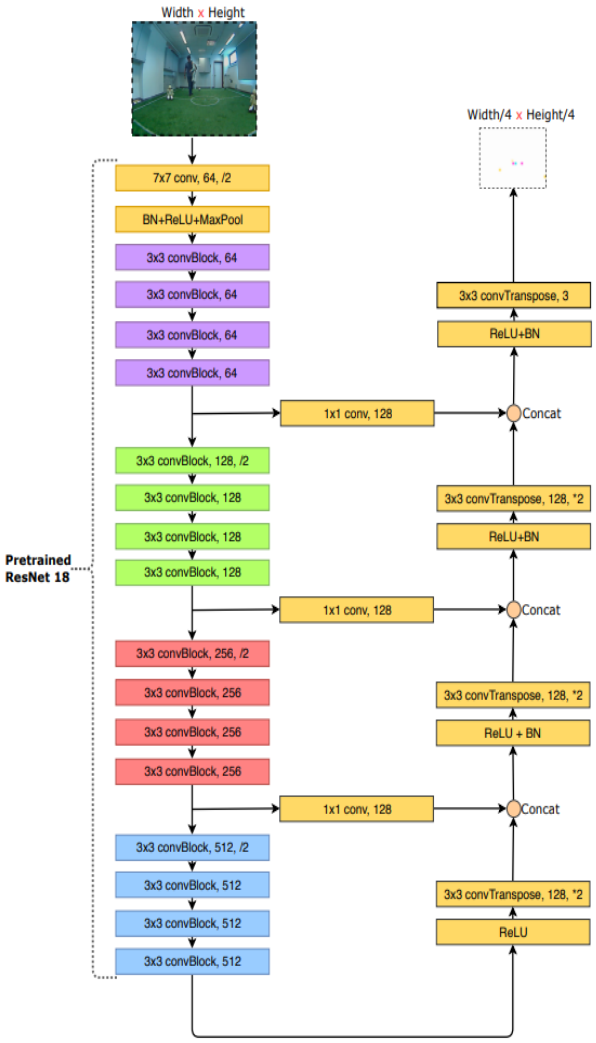

Fig. 1. The architecture in [1] used mostly unchanged other than some minor padding for the residual tensors

Dataset\Metrics	Recall	False Detection
Training Set	0.8151	0.1972
Validation Set	0.8144	0.1988
Test Set	0.4188	0.6010

Table 1. Recall and False Detection metrics for the training, validation and test sets

3. Each center coordinate of bounding box calculated above is multiplied with a factor of $0.25 \times \frac{x^i_{new}}{x^i_{old}}$ where x^i_{new} and x^i_{old} are the new and old lengths of the tensor respectively in the dimension i. This is done to rescale the coordinates to the smaller output tensor from the bounding box coordinates on the original image.
4. After the non-zero pixels are assigned, a gaussian blob centred on the non-zero pixels in each channel is set. The gaussian has a mean of 0 and a standard deviation of 8. The radius of the gaussian blob is fixed at 8 pixels.

3.3 Training, Post-Processing and Evaluation

The training is performed simply using Mean Squared Error loss and the `Adam` optimizer [6]. A generic learning rate of 0.01 is used the entire model is trained for 100 epochs. The encoder is trained without any weights being frozen from the start of the training. A training procedure uses milestones of improving accuracy on the validation set to select the best model. A degradation by more than a specific fraction of the *best* accuracy forces the model to reload the previous *best model*. The accuracy metric used was defined as:

$$ACC = \frac{TP}{TP + FP + FN} \tag{1}$$

The post processing module is used to reduce the network output to the final prediction for the humanoid robot parts. The following operations are applied in sequence to each channel (2D) of the network output.

1. **Thresholding:** The 2D array/image is thresholded at a value of 0.2 with any response below this value being ignored as low responses.
2. **Morphological Opening:** The thresholded image is subjected to a morphological opening which is essentially an erosion and dilation in sequence. Both operations are performed with a 3×3 kernel. Erosion essentially removes noisy pixels/activations from the image while dilation creates a more solid blob out of the remaining pixels.
3. **Contour Detection:** The next step involves contour detection for the image from the previous dilation step. This allows us to detect separate contours representing separate robot body parts from each channel.

Fig. 2. (Row-wise Left to Right) Examples of output channels of 2 robots before and after morphological opening. 4 pairs of images are given, 1 from each output channel. The first one is the output map after thresholding while the second is after morphological opening

4. **Central Point Detection:** A minimal bounding box is created for each contour and the central point for each bounding box is used as a detected body part.

For evaluation of the detected body parts, the center point of each contour is used on the true labels for each channel/body part and a set of true points is obtained. We proceed to search for a valid true point for each predicted point. A margin of 4 pixels is used to evaluate the correctness of the prediction. Each predicted point which does not contain a corresponding true point contributes towards the False Positive percentage. Each true point which was not predicted contributes towards the False Negative percentage.

4 Implementation

We use the PyTorch Deep Learning framework to define, train and test our models. Our entire preprocessing pipeline is based in Python 3.7 with heavy dependencies on NumPy, SciPy and PyTorch. The pretrained ResNet-18 model used was downloaded from the ones provided as part of the torchvision model zoo. The image processing modules used in postprocessing and evaluation sections used the Python interface for OpenCV2. The experiments were run on Google Colab using a 16GB Tesla T4 GPU, 12GB of RAM and Intel Xeon 2.30GHz CPU.

5 Results and Discussion

There were multiple difficulties faced during the convergence of the learning process. While training with a single robot type during the previous submission, a completely frozen encoder seemed to be have sufficient model capacity to learn the target function. However, after incorporating the high degree of variability

Datasets	Training Set		Validation Set		Test Set	
Body Parts	**Recall**	**False Detection**	**Recall**	**False Detection**	**Recall**	**False Detection**
Head	0.9304	0.0648	0.9383	0.0491	0.4956	0.5011
Hands	0.8758	0.1668	0.8701	0.1743	0.4624	0.5457
Legs	0.7934	0.2097	0.7509	0.2669	0.3943	0.6024
Trunk	0.6704	0.3463	0.6502	0.3750	0.3326	0.6753

Table 2. Recall and False Detection Rates for each body part on the training, validation and test sets

of input from the images captioned by each team, a frozen encoder is incapable of providing a useful model. The various metrics for the training and validation are summarized in Table 1. While the training and validation sets maintain a decent level of Recall and False Detection rates, both deteriorate for the Test Set which highlights a degree of overfitting. Table 2 illustrates the error metrics on each body part of the humanoid robots.

5.1 Necessity of Morphological Opening

The usefulness of morphological opening was explored as part of our evaluation methods. Theoretically, the major function that could be served by morphological opening would be to provide the consequent contour detector with a relatively noise free output map with distinct clusters of pixels - representing a detected body part. However, it is observed empirically that this is not necessary as the absence of the morphological opening module does not significantly impact output maps. This is illustrated by Figure 2 which provides an example of feature maps from each of the output classes.

6 Conclusion and Possible Improvements

The project in its current state is an extension of [1] to the task of humanoid robot part detection. There are however multiple avenues to explore to improve results. A major difficulty we faced was, seemingly, the low amount of training data to capture the variability of the humanoid robot scales and poses. A larger dataset would probably be useful in training a more robust detector in future implementations. The issue with overfitting can also be addressed with network regularization techniques such as Dropout on the input image before feeding it into the encoder. Also, a comparison between other state-of-the-art models trained on the Imagenet as the Encoder with the Resnet-18 model used in this work can be a feasible direction of exploration. Having said that, the current version of the humanoid robot detection system highlights the various challenges of designing such a system and illustrates ways forward towards a workable model for the task.

References

[1] G. Ficht, H. Farazi, A. Brandenburger, *et al.*, "Nimbro-op2x: Adult-sized open-source 3d printed humanoid robot", in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2018, pp. 1–9.

[2] G. Lin, A. Milan, C. Shen, *et al.*, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1925–1934.

[3] J. Yosinski, J. Clune, Y. Bengio, *et al.*, "How transferable are features in deep neural networks?", in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[4] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge", *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[5] K. He, X. Zhang, S. Ren, *et al.*, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

Fig. 3. An example of a set of predicted (yellow) and true (blue) output maps. The images show a varying degree of correctness of predictions