Scene-parsing for disaster environments by means of a convolutional neural network

Solvi Arnold, Shinshu University, s_arnold@shinshu-u.ac.jp
Kimitoshi Yamazaki, Shinshu University, kyamazaki@shinshu-u.ac.jp

Disaster robotics poses particular challenges for computer vision, both in terms of image material characteristics (due to motion blur, difficult light conditions, lack of up/down orientation, etc.), and in terms of learning data (limited availability, difficulty of annotation due to image quality, etc.). We developed a system for real-time scene-parsing, intended for use in a support system for operators of remote-controlled mobile robots employed in disaster areas. Our testbed is video footage gathered by a snake-like mobile robot exploring an (artificial) collapsed building environment. The core of the system is a relatively small-scale convolutional neural network. Our approach combines pixel-level learning with superpixel-level classification, in an effort to learn efficiently from a relatively small number of partially annotated frames. Our classification system is capable of real-time operation, and demonstrates that convolutional neural networks can be employed effectively even under the harsh conditions imposed by disaster robotics.

Keywords: disaster robotics, rescue robots, computer vision, convolutional neural networks, scene parsing.

1. Introduction

Recent years have seen increasing interest in the potential of robotic systems to assist in disaster response operations. The 2011 Tohoku earthquake and tsunami, and the ensuing nuclear disaster at the Fukushima power plant, in particular, have spurred the establishment of various initiatives for the development of disaster robotics. We are developing a scene parsing system to assist operators of remote controlled robotic systems in making sense of the environment by labelling, in real-time, objects and surfaces in an incoming video stream. In particular, our system targets a snake-like robot designed for exploration of collapsed structures. This platform highlights many of the characteristic difficulties one encounters when trying to apply computer vision techniques in disaster robotics. For example, while typical computer vision research employs datasets of sharp images from well-lit environments, computer vision for disaster robotics must be able to cope with unforgiving light conditions, motion blur, dirty lenses, etc. Furthermore, disaster robotics poses particular challenges for any system that relies on learning data. Recent years have seen a trend towards bigger and bigger datasets in machine learning research, where sheer scale has become a point of appeal [1]. However, producing suitable learning data tends to be a time-consuming effort. A disaster robotics system should be employable on short notice in unexpected environments, stressing the importance of learning efficiently from limited data.

2. Image classification using convolution nets

Recent years have seen convolutional neural networks (CNNs) take a central position in the field of visual recognition, achieving state-of-the-art results on many problems. A common application is image classification, where networks are trained to identify the object depicted in an image (e.g. [2]). In this case network output is a vector giving for each class under consideration the estimated likelihood of the image belonging to that class. A limitation of this approach is that it assigns a class to the whole of an image. When an image contains items from multiple classes, such systems cannot tell us which classes occur where in the image (although it may be set up so as to give high probabilities to multiple classes). For actual scene-parsing, classification must be performed on a finer level. This has led to the pursuit of pixel-wise classification [3] [4] [5]. Here nets are trained to guess the class-membership of every individual pixel in an image. Pixel-wise classification handles scenes with multiple objects naturally, and captures spatial structure. An obvious issue is computational cost: typically the output has even higher dimensionality than the input (namely image resolution multiplied by the number of classes). Also, as results for individual pixels tend to be jittery, post-processing to integrate pixel classifications over larger areas is necessary for smooth results, adding further complexity and computational cost to the system. Use on raw video streams from badly lit environments imposes additional challenges, such as the presence of blurred and dark areas whose pixels cannot meaningfully be assigned a definite class. With these factors in mind, we developed a CNN system that operates at an intermediate scale: the net trains on annotated pixel data, but performs classification on superpixels. Below we explain our approach, starting with our dataset.

3. Dataset

Here we describe our dataset. We obtained 67 minutes of raw footage from a snake-like robot moving around in an environment that models a collapsed building. From this footage we extracted a set of 259 frames as our dataset for learning (90% of the set) and accuracy evaluation (). Fig. 1 shows example frames.

Fig. 1. Example frames from the dataset.
Note the substantial blur and limited visibility.

3.1 Pre-processing

The dataset is normalized by subtracting the mean and dividing by the standard deviation over all pixels. We also remove the fish-eye deformation of the camera. The dataset is augmented by including rotated and flipped versions of the images.

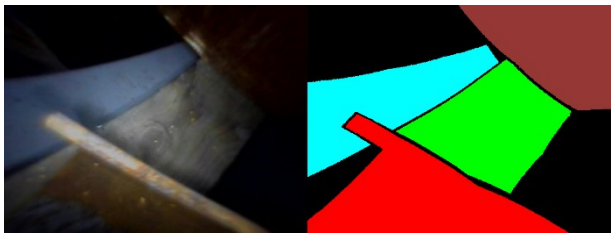


Fig. 2. Example of annotation.

3.2 Annotation

Frames were annotated by placing coloured polygons over individual objects (with different colours mapping to different classes), using Microsoft PowerPoint's polygon drawing functionality. Fig. 2 shows an example annotation and Table 1 gives the set of classes.

Table 1. The set of classes.

1	Pipe (inside)	7	Nylon curtain
2	Pipe (outside)	8	Ground
3	Mesh	9	Concrete
4	Wooden beam	10	Fitting
5	Iron beam	11	Blackout (auto-annotated)
6	Rusty steel		

4. System

We do not feed whole images into the network's input, but instead we dynamically source 'multi-scale patches' (abbreviated below as MSP) from the dataset.

4.1 Multi-scale patches

A MSP is a (W x H x C x S) array of image data, where:

W = width of patch (48 here)
H = height of patch (48 here)
C = number of colour channels in image (3 here (RGB))
S = number of scales per MSP (3 here)

Fig. 3 illustrates the concept. In practice, many vision-oriented neural network libraries assume a three dimensional input space (W x H x C), and the library we used (Pylearn2 [6]) is no exception. Fortunately colour channels and scales can safely share a dimension, so MSPs were implemented as W x H x (C * S) arrays. To determine the scales, we have an array $[s_0 \cdots s_{S-1}]$ of scale parameters, here set to $[1, 3, 5]$.

MSPs are composed as follows. We start with the coordinates for a single pixel in an image (the way this pixel is selected differs between learning and classification, as explained below), which we refer to as the 'focal pixel'. For every scale s_i in the scale array, we copy from the image a (($s_i \cdot$W) x ($s_i \cdot$H) x C) region centred on the focal pixel. This image patch is then scaled down in its width and height dimensions by a factor s_i, resulting in a patch of size (W x H x C). After doing this for all scales, we combine the resulting patches into a single array, which is used as input data for the net. The correct classification for a given MSP is the category of its focal pixel. Hence the output of the net is a 1D vector of length equal to the number of categories (just as it usually is when classifying in one-class-per-image fashion).

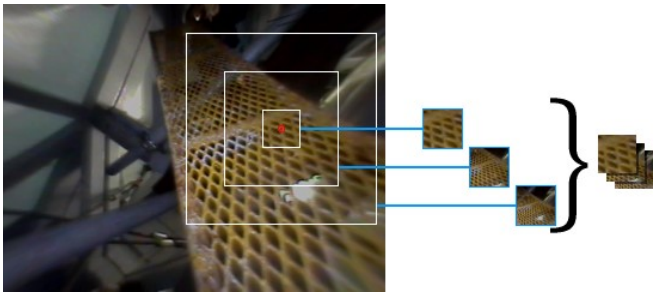


Fig. 3. Multiscale patch concept. The focal pixel is indicated with a red marker. White outlines indicate the source regions for each scale. Source regions are copied and scaled down to the same size, then combined into a 9-channel (3 scales times 3 colour channels) compound image.

4.2 Advantages of a patch-based approach

One important advantage of the patch-based approach is that it allows for easy balancing of the category distribution presented to the net during learning. When creating a benchmark dataset it is easy to arrange similar occurrence frequencies for all categories, but in practical use scenarios one is likely to be confronted with highly skewed occurrence frequencies across categories. When presenting a net with whole images (as done in e.g. [3] [4] [5]), the more common categories will have a stronger effect on the learning process. This may be desirable when categories' importance is proportional to their occurrence frequency (as it is when aiming to improve benchmark scores), but in practical applications importance and frequency are not necessarily linked (in e.g. the case of a robot exploring a collapsed building, we can expect to see far more rubble than survivors, but correctly detecting survivors is of greater importance than correctly detecting rubble). Given than each patch functions as an example of one category, it is trivially easy to balance (or even counter-bias) the category distribution seen during training.

Also, the fact that our annotations are partial means that learning on whole images would require some way to deal with very large numbers of missing target values during training, which is known to be a difficult problem.

Note that, like fully convolutional nets, the patch-wise approach allows input images of any resolution to be processed by one and the same net. No rescaling of input images is necessary when changing to a different input image resolution. This is a plus when working with different robots with different cameras in the same environment.

Table 2. Network architecture. The ● and × marks indicate whether a given layer is included or not, respectively.

	Architecture		
	#1	#2	#3
7x7 convolution	●	●	●
2x2 pooling	●	●	●
Cross-channel normalisation [2]	●	●	●
5x5 convolution	×	●	●
2x2 pooling	×	●	●
Cross-channel normalization	×	●	●
5x5 convolution	×	×	●
2x2 pooling	×	×	●
Cross-channel normalization	×	×	●
Fully connected layer	●	●	●

4.3 Network architecture

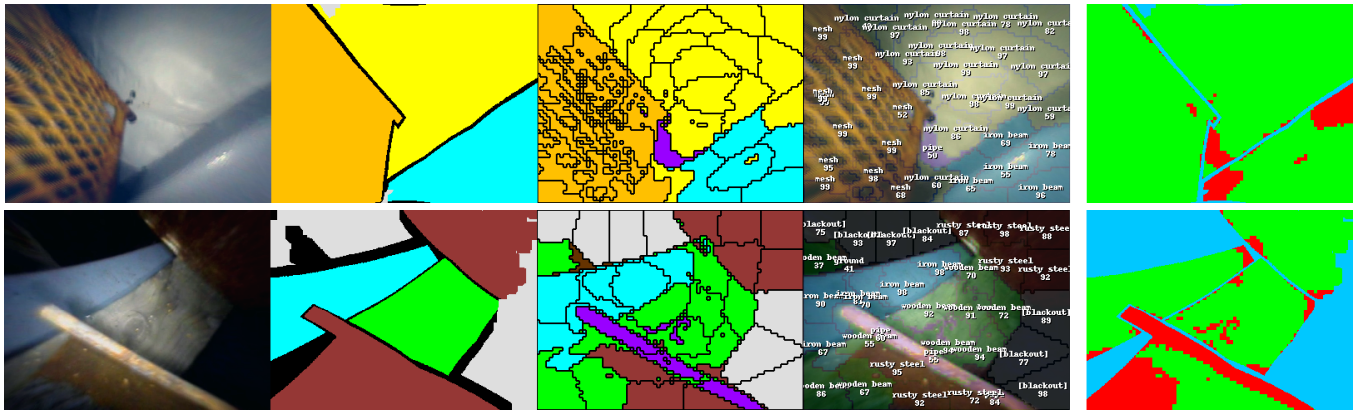
We report results for three CNN architectures, shown in Table 2. The networks were implemented using the Pylearn2 library [6]. Weights are updated using Pylearn2's stochastic gradient descent. The learning rate is fixed to 10^{-6} and weight decay was set to 5×10^{-5}. Training is ran for 50,000 epochs of 1000 weight updates each.

4.5 Learning and classification

During learning, the net is trained on batches of MSPs. The MSPs used for training are generated dynamically during learning. Focal pixels are picked randomly from the set of all annotated pixels (hence the number of learning examples is equal to the number of annotated pixels in the augmented dataset). The pixel's assigned category is set as the target output for the MSP. By weighing pixels' probability of being picked by the scarcity of their category, we equalise category occurrence, as noted above.

Classification is performed as follows. We first split the input image into superpixels using the (OpenCV2 implementation of) the SLIC algorithm [7]. We found that (at least on our data) superpixel segmentation can safely be performed at reduced resolution without notably deteriorating the overall quality of the segmentation. Whereas segmentation takes well over 300ms at the original image resolution of 640x480, at 1/8 of this resolution it takes about 3ms per frame while still providing visually satisfactory results.

This algorithm takes the desired number of superpixels to split the image into as a parameter. We found that for our input images, a setting of 35 provides well-sized superpixels that do not cross class borders much. For every superpixel, we computer the centre pixel, which becomes the focal pixel for one MSP. After extracting the relevant MSPs, we run (35 instances of) the trained network, in parallel (on the GPU). This results in classifications for the superpixels. The final classification is produced by assigning the class found (i.e. the class



a. b. c. d. e.

Fig. 4. Result examples for two frames. (a) Original frame. (b) Annotation (ground truth). (c) Classification obtained from CNN. (d) Classification overlaid on input frame. 5th column: visualisation of the agreement between annotation and classification by CNN. Green and red indicate agreement and disagreement, respectively, and blue indicates absence of annotation. Light grey areas in annotation and classification result indicate pitch-dark areas. These are automatically annotated but disregarded during evaluation (they are almost trivial to classify and hence would artificially inflate classification rates).

receiving the highest probability score) for each superpixel through to all pixels contained in that superpixel. Thus the resulting classification has the same resolution as is used for segmentation. For accuracy evaluation and display, we scale the classification back up to the original image resolution (without smoothing).

5. Results

Table 3 shows accuracy scores against the test set for architectures #1, #2, and #3. We see that the largest of our three architectures attains the best performance, at 83.64% correct classification on the test set. One performance-limiting factor is that the annotations used for training the net are far from perfect. The system is designed to help alleviate the difficulties in interpreting the visual feed from the snake robot, but this same difficulty is faced by the annotators (albeit not under the strain of real-time operation). In practical use, performance could likely be further improved by involving individuals who are familiar with the mission environment in the annotation process.

Table 3. Performance scores on test set

Architecture	#1	#2	#3
Accuracy	78.46%	81.23%	83.64%

6. Processing Video

Above results quantify the system's performance over a set of individual frames (the test set), but when running classification on a video stream, some further enhancements can be introduced to improve user experience.

First off, we introduce probability buffering. Local misclassifications lasting just one or a few frames are common. These can be suppressed by means of temporal smoothing. The idea here is that the class of a pixel location in the directly preceding frames provides a good predictor of that location's class in the current frame. We keep a buffer (at segmentation resolution) that stores probabilities for every class for every pixel, and updates the probabilities, leaky integrator style, with every new result frame. Classifications are then read from the buffer instead of the result frame. This temporal smoothing suppresses incidental misclassifications. Also, whereas a given pixel is assigned (at most) one class at any given time, the buffer also allows the probabilities of the other classes to affect the ease with which the classification will switch to those classes. The temporal smoothing rate (i.e. the decay rate of the buffer) can be adjusted during operation from a simple GUI.

Secondly, we suppress low confidence results. When the highest class probability for a given pixel falls below a set threshold, we suppress the classification and do not assign the superpixel a class. This threshold, too, can be adjusted during operation from the GUI.

By means of multithreading we parallelise the pre-processing (segmentation and finding the focal pixels), CNN processing, and post-processing (updating the probability buffer and drawing an image overlay on basis of the CNN's output) steps. While our implementation leaves room for improvement, we obtain frame rates of approximately 13fps, fast enough to convincingly track live input.

6. Discussion & Conclusions

We have described a Convolutional Neural Network system for scene parsing in a visually challenging environment, taking camera footage from a snake robot exploring a collapsed building as our testbed. Our results show that even in such a difficult setting and with limited learning data, decent recognition rates can be achieved.

Acknowledgements

This work was partly funded by ImPACT Program of the Council for Science, Technology and Innovation (Cabinet Office, Government of Japan). We appreciate Prof. Tadokoro and Prof. Konyo for providing image data. We also thank Yosuke Koishihara for assistance in the annotation efforts.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision,* vol. 115, no. 3, pp. 211-252, 12 2015.

[2] A. Krizhevsky, I. Sutskever , G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25*, 2012.

[3] C. Couprie, C. Farabet, L. Najman , Y. LeCun, "Indoor Semantic Segmentation using depth information," *International Conference on Learning Representations*, 2013.

[4] C. Farabet, C. Couprie, L. Najman , Y. Lecun, "Learning Hierarchical Features for Scene," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, no. 8, pp, 1915-1929, 2013.

[5] N. Höft, H. Schulz , S. Behnke, "Fast Semantic Segmentation of RGB-D Scenes with GPU-Accelerated Deep Neural Networks," *Proceedings of 37th German Conference on Artificial Intelligence (KI),* pp. 80-85, 2014.

[6] I. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien , Y. Bengio, "Pylearn2: a machine learning research library," *arXiv:1308.4214,* 2013.

[7] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua , S. Süsstrunk, "SLIC Superpixels," *EPFL Technical Report, no. 149300,* 2010.