2015 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AUTONOMOUS GROUND SYSTEMS (AGS) TECHNICAL SESSION
AUGUST 4-6, 2015 - NOVI, MICHIGAN

FAST INCREMENTAL LEARNING FOR AUTONOMOUS GROUND NAVIGATION

Artem Provodin
New York University
New York, NY

Liila Torabi
Net-Scale Technologies
Morganville, NJ

Urs Muller
Beat Flepp
Michael Sergio
NVIDIA Corp
Santa Clara, CA

Jure Žbontar
Yann LeCun
New York University
New York, NY

L. D. Jackel
North C Technologies Inc.
Holmdel, NJ

ABSTRACT

A promising approach to autonomous driving is machine learning. In machine learning systems, training datasets are created that capture the sensory input to a vehicle as well as the desired response. One disadvantage of using a learned navigation system is that the learning process itself may require both a huge number of training examples and a large amount of computing. To avoid the need to collect a large training set of driving examples, we describe a system that takes advantage of the immense number of training examples provided by ImageNet, but at the same time is able to adapt quickly using a small training set for the driving environment.

INTRODUCTION

Off-road autonomous ground navigation is still an unsolved challenge. The difficulty of the task arises from the enormous variability that an Unmanned Ground Vehicle confronts when it leaves the relatively well characterized domain of the road. While, in principle, it should be possible to create a rule-based system that codifies all situations that might be presented to the vehicle, such approaches are brittle, and tend to fail when new environments are encountered.

A more promising approach to autonomous driving is machine learning. In such systems, training datasets are created that capture the sensory input to a vehicle as well as the desired response. These responses may be provided by a human driver who initially teleoperates the vehicle, or may be gleaned from the vehicle's prior driving experiences.

A number of systems have been described that learn drivable vs non-drivable terrain using handcrafted features based on color or texture [1]. In recent years, Convolution Neural Networks (ConvNets) [2], in which the features are

learned rather than handcrafted, have been shown to be the most accurate in numerous image recognition tasks [3].

A disadvantage of using a learned system is that the learning process itself may require a huge number of training examples and a large amount of computing. As an illustration, the most accurate networks used in the ImageNet competitions [4] train on over one million examples and often require days of training on powerful processor clusters. While such expensive training may be fine for some applications, it is unacceptable for systems that have to adapt quickly and only have limited training data. Here we describe systems that take advantage of the huge number of training examples provided by ImageNet, but are able to adapt quickly using a small training set.

THE TERRAIN CLASSIFICATION TASK

Our networks were designed to train on regions of an image where the terrain type was known, say from stereo data, or by labeling by a human operator. The networks would then classify regions that were not known, usually from an entirely new frame.

SYSYEMS OVERVIEW

Our systems start with ConvNets that have been trained on ImageNet. We then preserve some of the initial layers of these trained ConvNets for use as feature extractors for our navigation task. A separate navigation training set is then created using the features extracted from patches taken from images in our driving environment.

These patches are labeled using variety of methods, such as obstacles from stereo, or by observing the regions a human teleoperator chooses to drive over or avoid. The patches, which varied between 119 x 119 pixels and 29 x 29 pixels, were labeled by the class of the center pixel.

Next, a fully-connected neural network with one or more hidden layers was trained using these feature/label pairs, i.e. each training example contains a feature vector obtained from the feature extractor applied to a patch and the label for that patch: “ground plane” (drivable), “footline” (marginal”) or “obstacle” (not-drivable).

Figure 1: The modified CoroBot in a test environment

As a test case for our system, we chose to learn 3-D obstacles from 2-D images using 3-D point cloud data from a stereo system to provide ground truth. To allow easy, rapid testing, we used a CoroBot Jr. robot modified to carry a Point Grey Bumblebee stereo camera on a 50 cm long stalk. We collected data in a park in Holmdel, NJ over all seasons. See Figure 1.

TRAINING AND TEST IMAGES

In our experiments we focused on whether we could use labeled data from a single video “training” frame to classify pixels in a “test” frame that was captured about 0.5 seconds later. The test frame differed from the training frame due to motion of the robot. The training and test frames and associated “ground truth” are shown in Figure 2.

The upper row of Figure 2 shows the images used for training (left) and testing (right). The images are 512 x 384 pixels. For training we picked 512 patches with randomly chosen centers of size 119 x 119 pixels. Training and test data used patches instead of pixels because we hypothesized that having context around the pixel in question would improve classification.

The lower row of Figure 2 shows the “ground truth” for the above images derived from stereo using the methods described in [5].

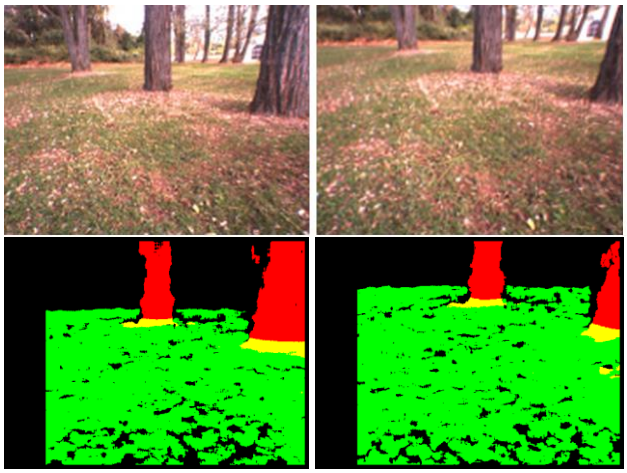

Figure 2: The training frame (left) and test frame (right) for our experiments are shown in the top row. The corresponding ground truth for each frame is shown in the bottom row. “Ground truth” labeling is obtained from stereo: “obstacles” (colored red) are defined as objects that rise more than about 4 cm above the nearby ground, “footlines” (colored yellow) are at the borders of obstacles with the “ground plane” which is colored green. The black regions on

the left and on the top of the ground truth images are artifacts of the stereo coverage.

BASELINE SYSTEMS

To see how well simple systems perform, we created two networks that processed raw RGB image data and then output terrain classifications. The simplest network was a single layer neural network using 119 x119 pixel patches of RGB data as input (SLNN-RGB).

We also trained a fully-connected network with one hidden layer containing 20 units (MLNN-RGB). Results for both these networks are shown in Figure 3. The SLNN-RGB network had a test error of 16% while the MLNN-RGB network has a test error 12%. The classification is sufficiently poor that it would be difficult to use the results of either of these networks for effective path planning.

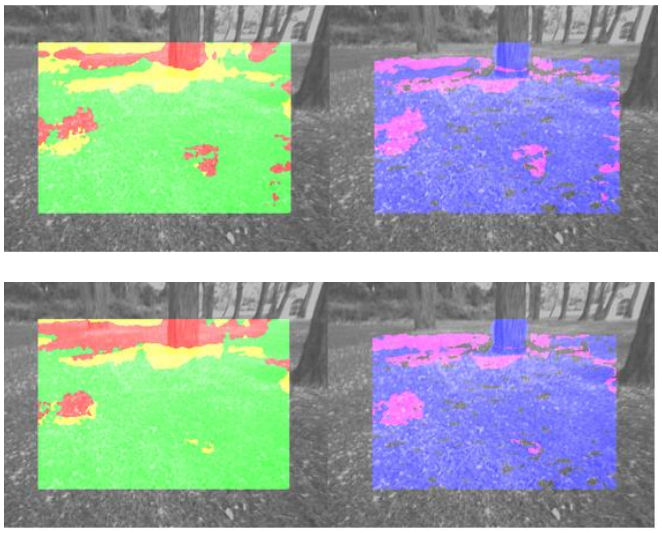

Figure 3: Test results for the SLNN-RGB network (top row) and MLNN-RGB network (bottom row). The images in the left column show the labels created by the network, and the images in the right column show where the labels agree (blue) and disagree (magenta) with the ground truth. The unclassified borders around the images result from the need to have the entire 119 x 119 patch fully contained within the test image with only the location of center pixel of patch assigned the label for the whole patch. Note that overlapping patches can be assigned different labels depending on the label of the center pixel.

CONVNET NETWORK ARCHITECTURES

To improve upon the simple SLNN-RGB and MLNN-RGB networks, we investigated a number of network architectures using ConvNets. This section describes some of the architectures we evaluated.

ConvNet Features Trained From Scratch (NYU-ConvNet)

The first network we tried used a ConvNet with 2 feature extraction layers trained on ImageNet examples. An additional final layer had units that represented the possible ImageNet classes and was fully connected to the last pooling layer. This initial network was trained on 1,000,000 ImageNet examples representing 1000 classes. The network scanned the image in 119 x 119 pixel RGB patches. One forward pass at each location in the image required ~ 25,000,000 multiply-add operations. Fully training our network took a few days on an NVIDIA Tesla compute server. The trained networks scored about 35% first-choice correct on a 1000 class problem.

Our objective here was not to create the very best ImageNet classifier, but rather to learn feature vectors (the output of the penultimate layer in our network) which could then be used for fast training of a linear network that uses navigation sensor input coupled with labeled terrain classes as training examples.

With the feature extraction layer weights frozen and the linear network appemded, we created the "NYU-ConvNet" classifier that had ~ 19,000 independent parameters or weights that were learned using the single training image.

Results for this network are shown in Figure 4. The left image shows the color-coded labels produced by the network. As in Figure 2, green is ground plane, yellow is footline, and red is obstacle. The right image shows the regions (blue tint) where the network output was in agreement with the "ground truth". Test error for this network was 2.8%. Comparison of

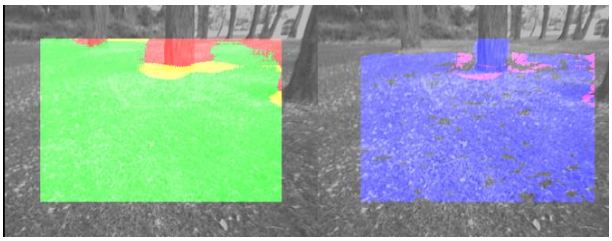

Figure 4: Test classification using NYU-ConvNet features that were trained from scratch. The color coding is the same as Figure 3.

Figures 3 and 4 clearly shows that the NYU-ConvNet classifier provided much better terrain classification than the simple SLNN-RGB and MLNN-RGB networks.

This network generally does a good job of identifying the large trees as obstacles and the grassy area as ground plane. We note that there is some "blooming" of the tree trunk which may be, in part, due to the pooling in the network which

causes it to sacrifice some spatial resolution. Nevertheless, the NYU-ConvNet classification appears adequate for effective path planning.

ConvNet based on VGG network

The VGG network [6] holds the first and the second place in the ImageNet localization and classification task. This neural network has 16 learned weight layers. We used all but the last 3 layers of this network with accompanying published weights for feature extraction. The advantage of using the "front end" of the VGG network is that we avoided the lengthy training that would have been required if we had to train the feature extraction from scratch.

Extracted features from the VGG network were fed into a fully connected single-layer which was rapidly trained using the usual training image (Figure 2).

We performed a number of experiments that compared the results using the VGG features with those obtained using the NYU-ConvNet as a function of the number of training examples. We trained both networks with 32, 64, 128, and 512 patches and measured the performance on the test image. Table 1 compares the test error rates for the NYU-ConvNet and the VGG network. The rates shown are the averages of several runs with different random choice of training patches.

Training samples	NYU-ConvNet	VGG network
32	5.6%	8.8%
64	5.0%	4.6%
128	4.9%	3.7%
512	2.8%	1.8%

Table 1: Error rate comparison table of deep ConvNet (VGG) versus a smaller network (NYU-ConvNet)

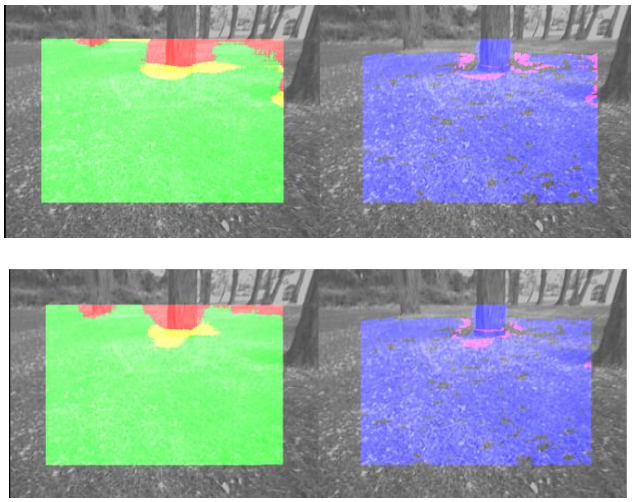

Figure 5: Classification results for the NYU-ConvNet (top row), and the VGG Network (lower row). The color coding is the same as Figure 3.

The VGG network had worse performance when the training dataset was small, but improved with more training examples. We believe that this behavior was due to the ~14,000 weights in the SLNN used with the VGG Network compared to the ~7000 weights in the SLNN used with the of the NYU-ConvNet. Figure 5 shows the classification results for the NYU-ConvNet (top row), and the VGG Network (lower row).

ConvNet based on Clipped VGG network

To test our hypothesis that the "blooming" visible in the tree trunks of the test image was in part due to the pooling layers in the VGG network, we discarded all but the first two layers of the VGG network and used the output of these layers as a feature extractor followed by a single layer perceptron in what we call the "Clipped VGG network".

Results for this network are shown in the lower row of Figure 6. While blooming is still present, it is reduced from that of the full VGG network. The error rate for the clipped network is 2.4%. An advantage of the clipped network is that much less processing is required than with the full VGG network.

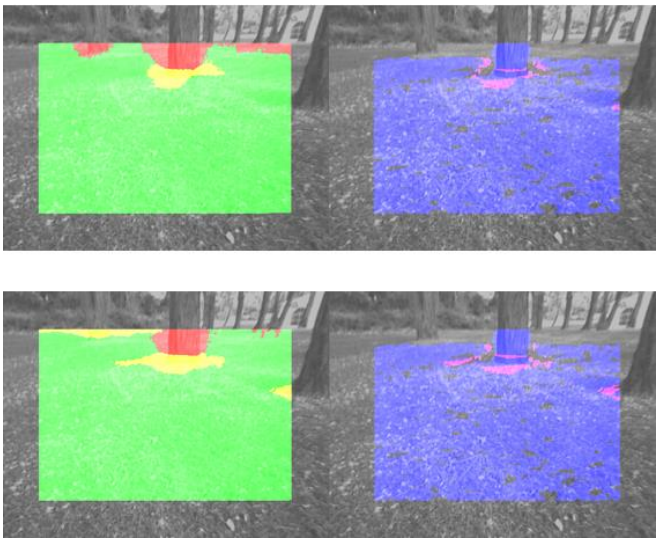

Figure 6: Comparison of classification by the VGG network (top row) with that of the Clipped VGG network (bottom row). The color coding is the same as Figure 3. Note the reduced blooming near the tree trunk with the Clipped VGG network.

LEARNING FROM A HUMAN TEACHER

In previous experiments we used stereo-based obstacle detection for path planning which was, in turn, used to navigate the CoroBot robot through a dirt trail in the woods. Navigation was successful until the robot encountered clumps of small plants growing in the trail. These small plants were interpreted as an obstacle and forward progress of the robot ceased, even though the CoroBot could have easily driven over the plants. If a human could teach the robot that driving over small plants or similar apparent obstacles is safe, off road navigation would be much improved.

Figure 7: A trail through the woods where stereo-based obstacle detection classified the small plants on the trail as "obstacles."

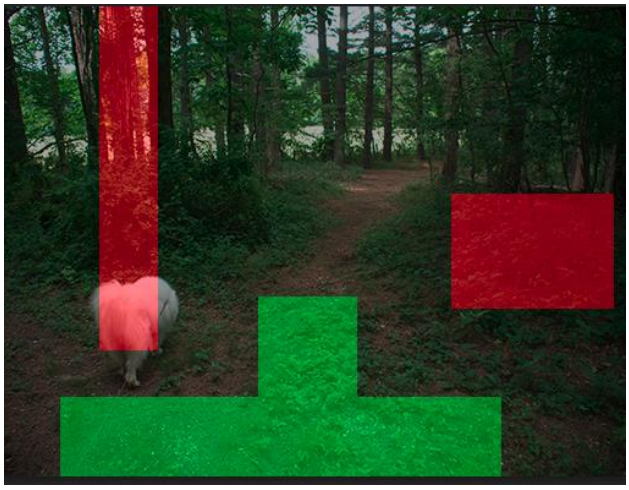

Figure 8: Hand-labeled regions (green and red tint) of drivable and obstacle terrain of Figure 7.

After developing the methods described in previous sections of this paper, we revisited the site, shown in Figure 7, where the CoroBot was stymied by the small plants. We hand-labeled parts of the image in Figure 7 as "drivable" and as "obstacle" as shown in Figure 8.

We trained a single layer perceptron network with a Clipped VGG Network feature extractor using the image of Figure 6 for training and the image of Figure 8 for assigning labels. 512 119 x 119 pixel patches were randomly chosen from the images as training examples with the requirement that the center of the patch was labeled either "drivable" or "obstacle".

Figure 9: A section of the trail that was used for testing that was near the section shown in Figure 7.

The trained network was tested on an image taken from a nearby section of the trail shown in Figure 9.

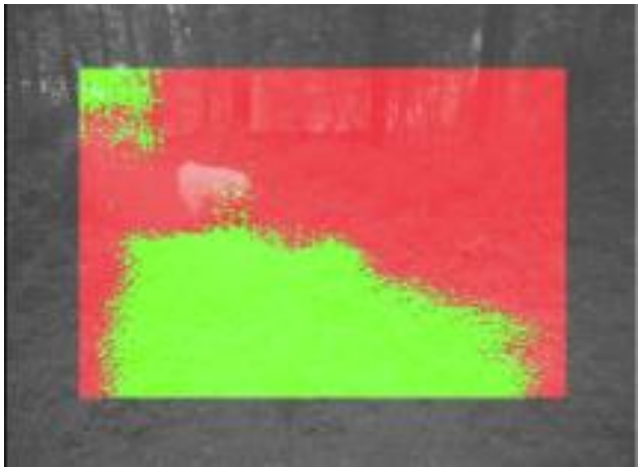

Figure 10: Classification results from the test image of Figure 9. Regions of with small plants are correctly classified

as “drivable” with regions with bushes and trees are classified as “obstacle.

Figure 10 shows the classification provided by our system on the test image. While the results are not perfect, they clearly show as “drivable” the terrain that is interspersed with small plants, regions that using stereo would have been classified as “obstacle”.

CONCLUSIONS

We have shown that using the feature extractors obtained from previously trained ConvNets provides good terrain classification, even though the original ConvNets were trained on a completely different corpus. Training the single layer perceptrons that use the feature vectors extracted by the initial layers of the ConvNets is sufficiently fast that the process provides a promising method for rapid adaptation while navigating in diverse environments.

ACKNOWLEDGEMENTS

We gratefully acknowledge support for this project by Army STTR contract W56HZV-13-C-0014 and Contracting Officer's Representative Rob Karlsen. We also acknowledge Luke Jackel who assisted in terrain data collection.

REFERENCES

[1] See, for example, “Online Learning Techniques for Improving Robot Navigation in Unfamiliar Domains”, Boris Sofman , doctoral dissertation, tech. report CMU-RI-TR-10-43, Robotics Institute, Carnegie Mellon University, December, 2010

[2] “Backpropagation Applied to Handwritten Zip Code Recognition,” Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L. D. Jackel, Neural Computation **1** 541-551, 1989.

[3] See, for example, “ImageNet Classification with Deep Convolutional Neural Networks”, A Krizhevsky, I. Sutskever, and G. Hinton, in proceedings of "Neural Information Processing Systems 2012.

[4] See, for example, http://image-net.org/challenges/LSVRC/2014/index

[5] “Learning long-range vision for autonomous off-road driving”, R Hadsell, P. Sermanet, J. Ben, A Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, Journal of Field Robotics, **26,** 120-144, 2009.

[6] “Very deep convolutional networks for large-scale image recognition”, A. Zisserman and K. Simonyan. http://arxiv.org/pdf/1409.1556.pdf