"On-the-spot Training" for Terrain Classification in Autonomous Air-Ground Collaborative Teams

Jeffrey Delmerico[1], Alessandro Giusti[2], Elias Mueggler[1], Luca Maria Gambardella[2], and Davide Scaramuzza[1]

[1]Robotics and Perception Group, University of Zurich, Switzerland
{jeffdelmerico, mueggler, sdavide}@ifi.uzh.ch
[2]Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Switzerland
{alessanndrog, luca}@idsia.ch

Abstract. We consider the problem of performing rapid training of a terrain classifier in the context of a collaborative robotic search and rescue system. Our system uses a vision-based flying robot to guide a ground robot through unknown terrain to a goal location by building a map of terrain class and elevation. However, due to the unknown environments present in search and rescue scenarios, our system requires a terrain classifier that can be trained and deployed *quickly*, based on data collected *on the spot*. We investigate the relationship of training set size and complexity on training time and accuracy, for both feature-based and convolutional neural network classifiers in this scenario. Our goal is to minimize the deployment time of the classifier in our terrain mapping system within acceptable classification accuracy tolerances. So we are not concerned with training a classifier that generalizes well, only one that works well for this particular environment. We demonstrate that we can launch our aerial robot, gather data, train a classifier, and begin building a terrain map after only 60 seconds of flight.

Multimedia Material: This paper is accompanied by a video illustrating the approach, available at: `http://rpg.ifi.uzh.ch`

Keywords: terrain classification, air-ground collaboration, search-and-rescue, deep learning, convolutional neural networks

1 Introduction

In search-and-rescue scenarios, time is a critical factor in the success of the first responders [9], who must often put themselves in dangerous situations in order to provide aid. Unmanned systems have the possibility of providing new capabilities for them, as well as increasing their safety and decreasing the response time in delivering that aid. However, one challenge is that disaster scenarios (e.g. earthquakes) may alter the environment so that any prior maps are no longer valid, and even the types of terrain that are present may have changed.

This research was funded by the the Swiss National Science Foundation through the National Center of Competence in Research Robotics (NCCR).

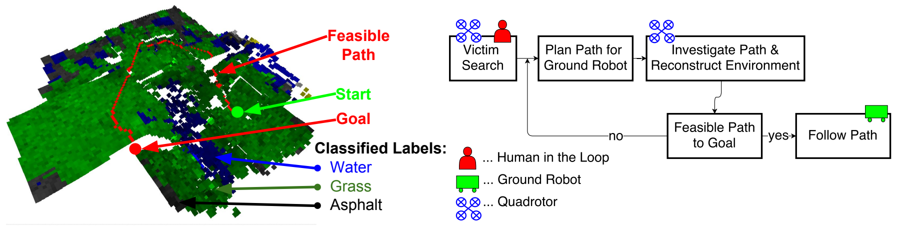


Fig. 1: Overview of the search and rescue scenario: a terrain map generated from one of our field experiments (left) and the workflow of the collaborative team (right). The map includes the elevation information, terrain classification, and the path found for the ground robot. We investigate *on-the-spot* training of the classifier that provides the terrain classification in our map.

Consequently, robotic systems must be capable of gathering and using data on demand, without a reliance on a priori maps or pre-trained classifiers.

In order to provide a fast unmanned response, we have developed a system where a flying robot guides a ground robot by mapping the terrain and finding a traversable path for it to follow. We assume no prior knowledge about the environment, so our system explores an unknown map and operates in the following stages, illustrated in Figure 1. First, a human operator flies the micro aerial vehicle (MAV), searching for a victim. Camera imagery from the MAV's down-looking camera is used to generate an initial classification of the terrain. Once the goal is found, the MAV engages in *autonomous* vision-guided flight while 3D reconstruction and ground robot planning are computed. The MAV incrementally maps the elevation and replans the ground robot trajectory until a feasible path the goal is completely explored.

During the *victim search* stage of operation, we gather data to train a classifier based on the terrain classes that are present in the environment. Our goal is to minimize the *overall* response time: the combination of classifier training time, aerial mapping and exploration, and ground robot traversal. Therefore, the faster this classifier can be trained, the shorter our ground robot's response time will be in delivering aid to the goal. This rapid training will of course come at the expense of quantitative performance, but in our search and rescue scenario, state of the art accuracy is not as important as response time: our terrain classifier does not need to generalize to many different conditions, since it's deployed immediately and locally.

1.1 Related Work

Other works have considered terrain classification from aerial imagery, but this is the first paper to address "on-the-spot" classifier training. An unmanned helicopter is used in [13] to gather multimodal aerial data, exhaustively exploring the area, to create an a priori terrain classification map that is then used to compute a ground path. High altitude, high resolution aerial images have also been used to perform terrain classification [1,8]. These approaches utilize pre-trained classifiers that model a fixed set of classes that are known a priori.

Performance of machine learning algorithms is strongly related to the quality and amount of available training data. This is especially true with image classification problems, which operate on high-dimensional inputs and are well-known to be difficult to handle in machine learning. Unfortunately, in our problem scenario we must handle severe limitations in the amount of available training data, which has to be acquired on the spot, and at the same time ensure very fast training time. The former problem has been studied more extensively than the latter, especially in the context of CNNs.

It is well known that CNNs are powerful image classifiers as long as enough data is provided [7]. Limited training set sizes are especially tricky for CNNs because they have many free parameters and can represent complex functions of their inputs. As such, they are prone to overfitting [2] on training data, which is very likely when training data is scarce. Previous works have studied adhoc network layers [15,14] that help to mitigate the effects of reduced training set sizes on accuracy. A different approach, which we adopt in this paper, is to augment training data by synthesizing a large amount of plausible training samples from a small set of actual samples.

In addition to CNNs, we also implement a more traditional approach to image classification that uses Local Binary Pattern texture descriptors [11] as features for a standard statistical classifier. This approach, which was standard until a few years ago, has now been superseded by deep learning for most tasks involving challenging visual pattern recognition problems. However, the limited availability of training data and the strict requirements in term of training time make the feature-based approach competitive in our scenario.

1.2 Contributions

Within the context of our search and rescue scenario, we consider the problem of rapid classifier training, and to the best of our knowledge, this is the first paper to analyze this problem with respect to robotic deployment. We make the following contributions:

- We study the relationship of training data volume on classifier accuracy and training time for both feature-based and CNN-based classification approaches. We demonstrate that it is possible to achieve good results with small amounts of training data and time.
- We propose a procedure for fast deployment of a terrain classification system in an unknown environment, including data collection, training, and classifier integration on a robotic system, all performed in-flight.

2 Technical Approach

Our robot team consists of a lightweight MAV and an all-terrain ground vehicle that can climb moderate grades and traverse small obstacles. Our MAV [5] is

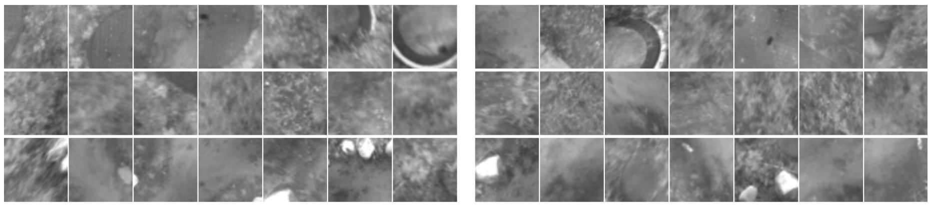


Fig. 2: Example patches from each class in the *canyon* dataset showing the ambiguity of the classes (training set on the left, testing set on the right). The rows represent (top to bottom) *rock*, *grass*, and *water*.

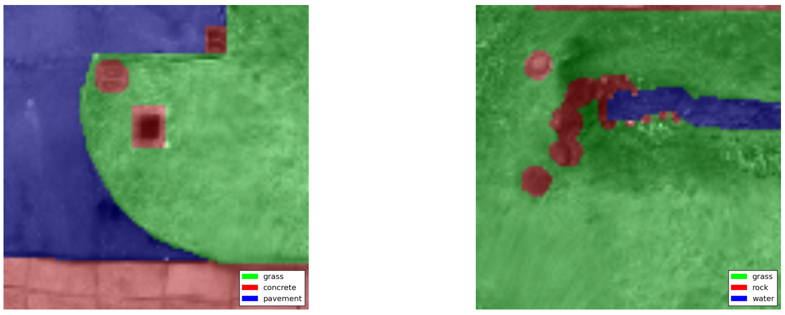
equipped with a downward-looking camera, and flies in autonomous and vision-assisted manual flight modes using the visual odometry pipeline SVO [6]. The images from this camera are additionally used for terrain classification, and for elevation mapping using the keyframe-based monocular dense reconstruction pipeline REMODE [12]. We use both the estimated terrain class and elevation to determine traversable paths in the map, and estimate their costs in terms of response time. An accurate terrain classifier is a critical component of this system, in order to identify non-traversable regions (e.g. water), as well as to distinguish between terrain classes that would cause a slow response time (e.g. mud) and a fast one (e.g. concrete).

2.1 Terrain Mapping

We consider a finite region of the ground surface to map, and discretize it into a 2D grid of uniformly sized cells. Our overall system seeks to populate the cells in this map with terrain class estimates and elevation, such that a feasible path can be found for the ground robot using that information. Within this system, the classification pipeline utilizes the estimated pose of the MAV to project its image stream into the map. Therefore, when we classify patches in an input image, we can associate the output with particular cells in the map. Additionally, since we have a precise estimate of the elevation of our MAV, we know the absolute scale of these patches, and we can train our classifier with data at only the scales we expect to observe during the mission, greatly simplifying the collection of training data. We accumulate classifications over time by averaging the class probabilities at each map cell over the number of observations there, with a neutral prior of $\frac{1}{n}$, where n is the number of terrain classes.

2.2 Classification

We compare two alternative approaches for classifying an image patch into a terrain class: *feature-based*, which computes Local Binary Pattern (LBP) texture



(a) *driveway* dataset (b) *canyon* dataset

Fig. 3: Ground truth labels for our two datasets. Note that we only consider the major classes in our classification problem: {grass, concrete, pavement} for *driveway* and {grass, rock, water} for *canyon*.

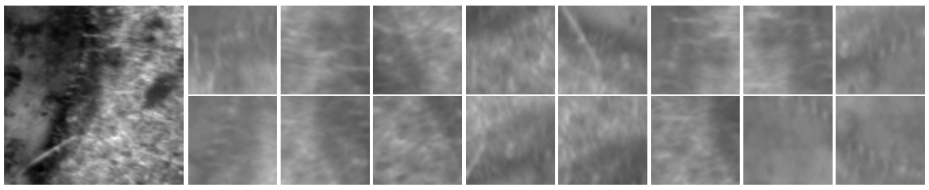


Fig. 4: Example of data augmentation output. On the left is the input patch (150×150 px). All of the smaller (50×50 px) training patches on the right have been produced using the approach described in Sec. 2.4.

descriptors [11] and then applies a logistic classifier; and *CNN-based*, which uses a Convolutional Neural Network for classification.

The *feature-based* approach operates on a 50x50 pixel image patch as input. On each patch, we compute an LBP descriptor, using 8 neighbors, uniform coding [11] and a given radius r, resulting in a 10-element feature vector. For a given input, we compute the descriptor for $r = 1, 2, 4, 8, 12$ pixels and concatenate the resulting feature vectors. We also add the variance of the image, which is not captured by LBP, as an additional feature. This results in a 51-dimensional feature vector. We then apply a standard classification pipeline based on feature scaling to zero mean and unary variance, and a logistic regression classifier with L2-norm penalty for regularization, and regularization strength parameter = 1.0.

The *CNN-based* architecture implements interleaved convolutional (conv), max-pooling (MP), and Rectified Linear Units (ReLU) layers, followed by fully-connected (FC) layers for classification. This architecture has been shown to perform well in a wide range of pattern recognition tasks [4]. The input layer of the network receives the raw intensity values of a 50×50 pixel patch. The network is then structured as follows: conv-3x3 with 20 output maps; ReLU;

MP-2x2; conv-5x5 with 20 output maps; ReLU; MP-5x5; FC with 100 output neurons; FC with with n output neurons, where n is the number of classes. The network is trained using stochastic gradient descent [3], with a base learning rate $\alpha = 0.01$, exponential policy for learning rate decay with $\gamma = 0.998$, and momentum $\mu = 0.9$.

2.3 Data Collection

Our target scenario would involve first responders arriving at a disaster site, deploying an MAV, selecting a few regions in the image stream to use as training data for each class, training the classifier in-flight, and then beginning to classify the terrain in the environment. To facilitate experiments that investigate the effect of training data volume on classifier performance, we captured several datasets of outdoor environments with multiple terrain classes. For each environment, a discretized map of size $10\,\mathrm{m} \times 10\,\mathrm{m}$ and cell size $0.1\,\mathrm{m}$ was constructed as described in Sec. 2.1, and the image stream from the MAV was projected into the map. We rectify the images and then crop patches from them that are centered on cells in the map. Using the estimated pose of the camera, each patch is selected as a $1.5\,\mathrm{m} \times 1.5\,\mathrm{m}$ region of the terrain when projected to the ground surface, and then resized to be 150×150 pixels. Note that in this patch, the actual cell represents the center 10×10 pixel region, but we also save the surrounding area for training. For each map cell, we crop a patch if the 150×150 pixel patch is fully contained in the image.

We flew our MAV over two environments, which we name *driveway* and *canyon*, and recorded approximately 3000 images each. We then generated a set of patches as described above, and stored all of the patches for each cell, yielding maps that contained up to 37 patches per cell, with an average across the two datasets of about 9 per cell. A few cells at the periphery of each map received 0 patches, and so we ignore these cells for training and testing. Example patches from the *canyon* dataset can be seen in Fig. 2, showing the challenge of distinguishing the three classes from each other.

In order to generate a map labeled with ground truth, we overlayed and averaged patches from each cell to create a mosaic image, and then annotated this by hand for terrain class. Fig. 3 shows the ground truth maps for the two datasets. Since all of our training data is manually labeled, we assume that all of the labels are correct (i.e., dataset quality is not an issue). This is a common assumption in nearly all machine learning applications to robotics, and handling noisy training datasets is still an active research topic in machine learning [10]. However, uncertainty in our MAV's pose estimate, and strong distortion from our wide-angle lens, results in some patches being projected to incorrect cells in the map, causing some polluted training data.

2.4 Data augmentation for classifier training

We adopt a data augmentation algorithm that allows us to generate an arbitrary number of unique samples from a single image patch. We utilize this procedure

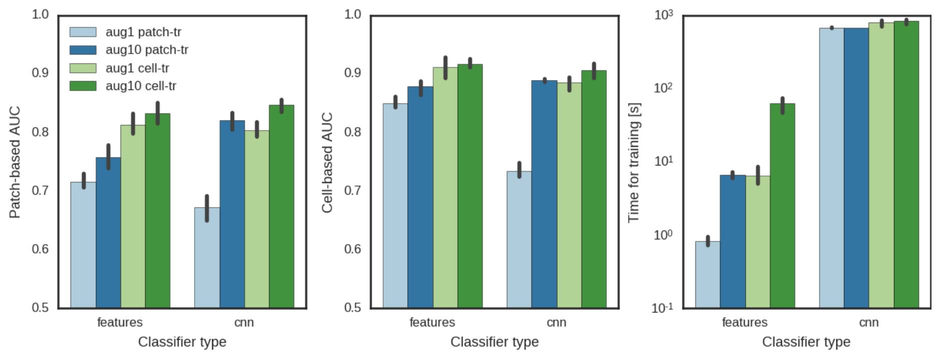


Fig. 5: Effect of training data augmentation strategies on classification performance (left, center) and training time (right). We compare 4 approaches: without data augmentation (aug1, light) and with 10 random variations per patch (aug10, dark); considering a single training patch per training cell (patch-tr, blue), or considering all patches that backproject to a given training cell (cell-tr, green). We report the AUC of the classifiers computed on all testing patches (left) and after averaging all testing patches belonging to the same testing cell (center).

to effectively boost the size of our training set without additional data collection and labeling. We sample a patch from a random cell in one of our datasets, then perform the procedure described below to generate a training patch, and repeat until we have reached the desired number of training samples for each class.

Given an input (150×150 px) patch from a training image, we produce any number of (50×50 px) patches using the following steps: select a point P within a distance of 10 px from the center of the input patch; sample a random rotation $\alpha \in [0, 2\pi)$; sample a random scaling factor $k \in [0.9, 1.1]$; define a square patch centered on P, rotated by α, with an edge size of $k \cdot 50$ px; warp this square to a (50×50 px) image patch; and finally subtract the mean of the patch to normalize it. Fourteen examples of random augmentations from a single (150×150 px) input are shown in Fig. 4.

3 Experimental Results

The focus of our classification method is rapid adaptation to previously unknown terrains—fast and specific training, instead of the ability to generalize. Therefore, our experiments concern the trade-off between the amount of training data, training time, and accuracy on unseen parts of the map. For the training time, however, we not only take the computation time into account, but also the time required for data acquisition *on the spot*. Since the critical factor in search-and-rescue missions is the overall response time [9], we perform a series of experiments to address the training time vs. accuracy trade-off in a variety of scenarios.

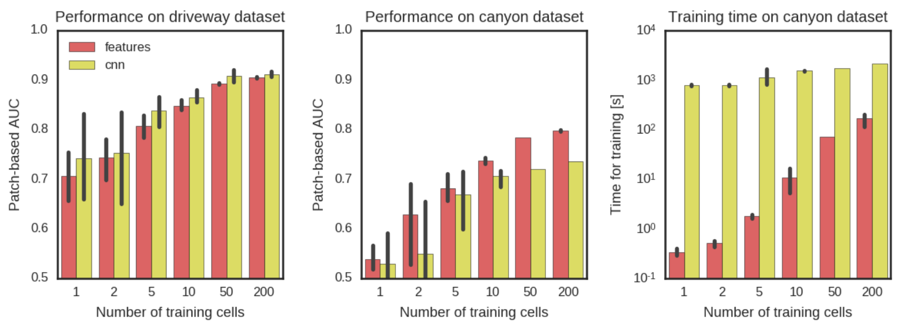


Fig. 6: Patch-based AUC as a function of the number of training cells per class (x axis) for the driveway (left) and canyon (center) datasets; data are reported separately for the feature-based (red) and cnn-based (yellow) classifiers.

3.1 Classifier Accuracy and Training Time Experiments

For each trial on each dataset, we randomly sample 250 cells per class to be used as potential training data; let S_{tr} be the resulting set of cells. Among the remaining cells, we randomly sample 100 per class to evaluate our classifiers on (S_{te}). The sampled cells have, on average, 11 patches each. For a given experiment, we consider the following parameters:

– number of training cells per class (that will be randomly sampled from S_{tr}).
– whether to train classifiers on a single patch per training cell (tr-patch) or all patches that map to a given training cell (tr-cell).
– how many random variations of each training patch to generate with data augmentation when building the training set for our classifiers.
– whether to use the feature-based classifier or the CNN.

Once we train the chosen classifier on the resulting training set, we evaluate it on all patches in S_{te}. We compute three metrics:

– The patch-based performance of the classifier; it is expressed as the Area under the ROC Curve (AUC) computed over all patches in S_{te}.
– The cell-based performance of the classifier; it is expressed as the Area under the ROC Curve (AUC) computed over all cells in S_{te}. A given cell is classified by averaging the classification vectors obtained for all cells that belong to it.
– The overall training time of the classifier (on a single Intel i7 core), including the time required to compute features for the training data.

The AUC is the preferred metric for evaluating classifier performance because it is not affected by the prevalence of classes in the testing set, does not depend on a threshold, and also captures the quality of the probabilistic information that the classifier produces. Note that any dummy classifier (e.g. a random classifier, or a classifier that only returns the majority class) yields an AUC of 0.5. This is

Algorithm 1: Mission Deployment of Terrain Classifier Pipeline

```
1  Define map size ($M \times N$ meters) and resolution $r$;
2  Launch quadrotor and fly to survey height under vision-based control;
3  foreach terrain class do
4      Select a region in the image stream that represents the class;
5      Label the map cells that the region projects to with the class label;
6      foreach subsequent image do
7          if image projects over map cell with label $l$ then
8              Crop a training patch centered on the cell and label it with $l$;
9  When enough patches for each class have been collected, begin training;
10 Once the classifier has been trained, begin constructing terrain class map.
```

the baseline for all the AUC plots. We repeat each experiment five times, with different random selections for the subset of S_{tr} that is used for training in each trial. Figures 5 and 6 report these results, with standard deviation over the 5 trials as error bars.

3.2 Training Pipeline Experiments

In these experiments, we demonstrate our full training and classification system in the context of a search and rescue mission, and we deploy our quadrotor in the same two environments as our previous experiments. Our experimental design is detailed in Algorithm 1, and represents our proposed procedure for mission deployment of our system.

We designed a software interface that allows the user to select a rectangular region of an image and label it as a terrain class. The selected region is projected to the surface map, and the cells that it overlaps are assigned the corresponding label. We deploy our flying robot, fly over a region in the map with a consistent terrain class, and then select that region in the image stream using our software. Meanwhile, patches from the image stream that are centered on these cells are projected to the surface map, cropped to a consistent size (150 × 150 px, as before), and associated with the labeled cells. After labeling a region for each terrain class, all of the patches that subsequently project to these labeled cells are collected together as the training set. In this way, we gather training patches for n terrain classes by simply selecting n regions of interest on the image stream.

We follow this procedure for both the *driveway* and *canyon* datasets, and are able to perform our full training procedure (see Alg. 1), from MAV launch to classification, in 60.44 seconds and 60.12 seconds, respectively (averaged over 5 trials each, computed on a single core of an Intel i7). These times each include approximately 8 seconds of training for the *feature-based* classifier on 1000 patches of each terrain class, randomly sampled from the training patches accumulated up to the start of training. We then proceed to survey the environment as we would in a mission scenario, and classify patches sampled from the image

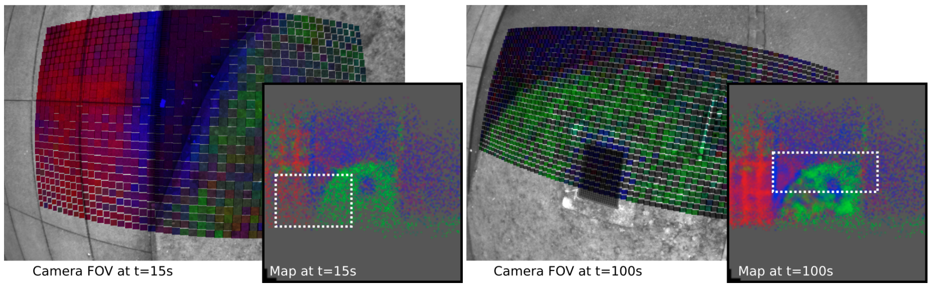


Fig. 7: Snapshots of the terrain classification map using a classifier that was trained in-flight. At each time t since the classifier finished training, there is a camera image overlaid with the current classification in the map, and the terrain class map (where RGB colors represent classification probability) overlaid with the area in view of the camera (white dotted rectangle) (right).

stream, while accumulating terrain class probability estimates in the map cells to which they project. Examples of the terrain class maps, generated using an "on-the-spot" trained classifier, are shown in Fig. 7, with ground truth labels in Fig. 3 for reference.

4 Discussion

The experiments in Sec. 3.1 demonstrate that when training data is limited, data augmentation is necessary to achieve acceptable performance from the CNN. Conversely, the feature-based classifier performs reasonably well even without data augmentation (Figure 5). This is expected, since the CNN operates on raw image data and must be taught rotational invariance through large training sets and/or data augmentation, while the feature-based classifier uses features that are rotationally-invariant by design.

We additionally found that classifier performance computed at the level of testing cells is higher than performance computed at the level of individual patches (Fig. 5 center vs left). In the former case, one averages multiple classifications obtained for different patches that map to the same cell, thus increasing the accuracy of the classification.

Training time for the CNN depends mainly on the fixed number of training iterations, which operate on fixed-size batches of data randomly sampled from the training set, and is therefore mostly independent of the number of training samples (right plots in Figs. 5 and 6). The training time of the feature-based classifier, is instead dominated by the time required to compute LBP features on all of the training samples. Therefore, it is heavily affected by the size of the training set and the amount of data augmentation, which as noted above does not have a significant effect on performance.

For both the feature-based classifier and the CNN, performance improves with the number of training cells but the *canyon* dataset is more challenging than *driveway* for both classifier types, due to ambiguous textures (see Fig. 2) However, despite the difference in their structure, the performance of the feature-based and the CNN-based classifiers are similar, in particular when relatively large amounts of training data are available. Note that with a small number of training cells, the performance of the resulting classifiers heavily depends on the choice of such training cells. If "bad" or non-representative training cells are chosen for training, the classifier will underperform. This is visible in Fig. 6 as the height of the error bars.

Training with approximately 10 cells (on average, about 110 patches) per class gives a good compromise between training time and AUC for the feature-based classifier. Increasing the training set size to 100 cells per class slightly improves performance while keeping the training time manageable, with diminishing returns above that dataset size.

With the current settings, the CNN-based classifier requires at least several minutes for full training regardless of the amount of training data and augmentation. This performance could be optimized by one or two orders of magnitude by using multiple CPU cores, adding GPU support, and optimizing the solver parameters for speed. However, at the moment we do not observe enough performance advantage to justify investment in these optimizations.

Our full pipeline experiments show that we can begin classifying approximately 60 seconds after launching the MAV, so surveying the environment for terrain classification with an on-the-spot trained classifier is certainly feasible within the battery life of even a small MAV. The speed and ease with which we can label a large amount of training data is only possible because we collect it from a mobile robot that maintains an estimate of its position and orientation in space. By labeling a region of the environment, rather than an image, we are able to multiply the training set size by automatically labeling subsequent frames using that spatial reference. This allows us to collect thousands of training samples in a matter of seconds without manually labeling all of them.

5 Conclusions

We have proposed and validated a system for "on-the-spot" classifier training for terrain mapping, in the context of a search-and-rescue air-ground robot team. When deployed in a disaster area, our MAV can gather training data, train a classifier, and begin terrain mapping in flight, within one minute of launch. During the development of this system, we thoroughly evaluated several classifier architectures and training approaches in terms of both quantitative performance and training time. This is the first work to demonstrate a system that can be trained and utilized immediately, in situations where response time is critical.

Although our experiments utilized small, fully trained CNNs, we intend to explore whether large pre-trained networks may be a feasible alternative, since we may be able to reduce the CNN training time if we must only train the final layer using the "on-the-spot" data. Our scenario also offers the opportunity for online learning if the user labels new regions of the map once the classifier has been trained. We intend to explore whether higher accuracy can be achieved by utilizing user input to correct or disambiguate difficult classification areas.

References

1. M.R. Azimi-Sadjadi, S. Ghaloum, and R. Zoughi. Terrain classification in SAR images using principal components analysis and neural networks. *IEEE Trans. Geoscience and Remote Sensing*, 31(2):511–515, 1993.
2. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.
3. L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.
4. D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.
5. M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. Autonomous, vision-based flight and live dense 3D mapping with a quadrotor MAV. *J. of Field Robotics*, pages 1556–4967, 2015.
6. C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.
7. Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–97. IEEE, 2004.
8. J. A. Montoya-Zegarra, J. D. Wegner, L. Ladicky, and K. Schindler. Semantic segmentation of aerial images in urban areas with class-specific higher-order cliques. *ISPRS Annals*, 1:127–133, 2015.
9. R.R. Murphy. *Disaster Robotics*. 2014.
10. N. Natarajan, I.S. Dhillon, P.K. Ravikumar, and A. Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.
11. T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.
12. M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014.
13. B. Sofman, J.A. Bagnell, A. Stentz, and N. Vandapel. Terrain classification from aerial data to support ground vehicle navigation. Technical Report CMURI-TR-05-39, Carnegie Mellon University, Jan 2006.
14. N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
15. M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *Int. Conf. on Learning Representations*, 2013.