Semi-Supervised Scene Traversability Classification for Autonomous Mobile Robots

Rongfei Lu
Stanford University
rongfeil@stanford.edu

Nick Goodson
Stanford University
ngoodson@stanford.edu

Abstract

In this work, a generative neural network architecture is applied to traversability estimation for autonomous mobile robots. Two models are trained, each on different datasets from unqiue mobile robot applications. The first application is an indoor differential-drive robot and the second is a self-driving car. The model employs an Autoencoder and a DCGAN to extract informative features form input images which can be used to classify the observed scene as traversable or not. The semi-supervised learning approach employs an automted data labelling scheme, resulting in large experimental time savings.
The indoor dataset is somewhat homogenous such that the DCGAN is able to learn a reasonable approximation to the underlying distribution and generate realistic images over the support. The resulting model proves reasonably capable at classification of traversability. The same cannot be said for the self-driving car dataset whose distribution proves too diverse to be accurately captured by the generative model. The resulting generated images consistently show modal collapse even when applying well documented DCGAN stabilizing solutions.

1. Introduction

Traversability estimation capabilities are imperative for autonomous mobile robots in real world environments. The complexity of these environments inhibits the development of models considering all possible contingencies apriori. Instead, robots need facilities for decision making during unexpected events and in unfamiliar localities. In real applications, it is not possible to observe the environment's state directly and instead it must be estimated from some combination of system models and sensors. For instance, camera's are commonly employed to capture rich information about the environment.

Optimal sequential decision making when the state cannot be directly observed is the field of Partially Observable Markov Decision Processes (POMDPs). Directly solving POMDPs is usually intractable so algorithms based on heuristics and simplifying assumptions are employed instead. This work focuses on the state estimation component of the broader decision making problem as applied to traversability estimation for mobile robots.

Many previous approaches to traversability estimation have relied on 3D depth information from lidar point clouds [8][13][14]. Suger *et al*. [14] employ a Bayesian learning algorithm on point clouds to build a grid based traversability map. The algorithm was tested on a ground robot operating in an outdoor environment and a semi-supervised learning approach was developed to reduce data-labelling time. Positive data was automatically labelled from trajectories created by a human manually controlling the robot. A disadvantage of the use of lidar alone is it doesn't provide the rich information needed to understand hazards in complex situations. Naturally cameras can be employed to collect more informative visual information about the environment. Early approaches to visual traversability estimation have considered building terrain maps [1], and detecting "collision danger regions" based on divergence metrics [2]. A recent approach to visual traversability estimation is GONet [7]. In this work, Hirose, Sadeghian *et al*. developed a neural network architecture, trained on images from a fish-eye camera, which can accurately predict the traversablity of the environment. The model uses an Autoencoder and a Deep Convolutional Generative Adverserial Network (DCGAN) [5] to generate a fake image which is a traversable imitation of the observed scene. If the true image and generated image are sufficiently similar, the scene is classified as traversable. The image data was collected similarly to [14], by manually operating the robot through environments and applying an automated labelling scheme.The GONet model architecture inspired and is heavily utilised in this project. Training DCGANs is notoriously unstable and several works have been published with architectures and heuristics for improving convergence. Arguably the most prominent contribution

came from Radford *et al*. [11]. The architecture presented is highly effective and has been successfully applied to a range of applications. The same architecture is adopted for this project along with several of the training heuristics from other sources [12][10]. These heuristics are based on a combination of theoretical and empirical evidence.

In this work, generative traversability estimation models are applied to two mobile robot applications, with the objective of learning to identify hazards rather than specifying them manually. The first application considered is an indoor differential-drive robot. The corresponding model is trained using images from the GO-Stanford dataset [6] which was collected inside several buildings on the Stanford campus. The original GONet model was trained on this dataset and the objective is to replicate the performance obtained by the authors. The second application investigated is self-driving cars, specifically the Kitti dataset [3]. Here, the aim is to determine whether the generative model can cope in this more complex domain. For both applications, sequential decisions in the underlying POMDP are simplified to a series of conditionally independent one-shot decisions, such that there is no temporal dependence. The model takes as input an RGB image from a forward mounted monocular camera on the mobile robot. This image is fed into a feature extraction module based on a DCGAN whose output is then classified by a fully-connected network. The binary classification states whether the observed scene is safely traversable by the robot or not. The model architecture and training procedure are described in Section 2. Section 3 delineates the datasets and automatic labelling procedure. Lastly, the experimental results are presented in section 4.

2. Model

The key mechanism through which the model classifies an input image as traversable is by generating an image I' which looks similar to the input image I and appears to come from the positive manifold of examples. Here the term positive is used to indicate a traversable scene. Logically, if the original image showed a traversable scene, then the generated image should look very similar. In contrast, if the generated image is very different, the original image likely showed a non-traversable scene.

Architecture

The full model architecture is illustrated in Figure 1. Two distinct modules are used to first transform the input image into informative features, then perfrom classification using said features. The *feature extraction* module uses an Autoencoder (*InvGen*) to encode the input image I into a latent vector z which is then passed through the *generator* network (*Gen*) of a DCGAN. An additional discriminator network (*Dis*) is used during training of the DCGAN, and

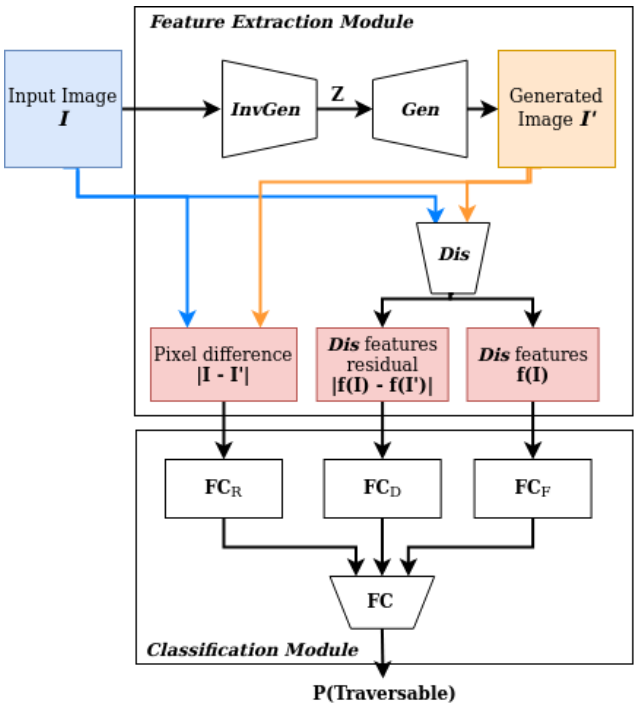

Figure 1. The generative traversability estimation network architecture. The features of the last conv layer of *Dis* are used by the classification module.

also for extracting features in the final model. *Gen* outputs a fake image I' used to estimate traversability. Features extracted from the real and fake images are compared by the *classification module* which returns the probability that the scene is traversable. Three features are passed from the *feature extraction* module to the classifier.

- The residual between the images $\Phi_R = |I - I'|$
- The difference in the features from the last convolutional layer of *Dis* $\Phi_D = |f(I) - f(I')|$
- The *Dis* features for the real image $\Phi_F = f(I)$

The importance of each of these features to classification accuracy was investigated through an ablation study in [7]. The classification module consists of three fully-connected networks which each output 10 features. These 30 features are concatenated then fed into a final fully-connected layer which outputs the binary classification probability.
Gen's architecture consists of a fully-connected layer followed by four transposed convolution layers, the last of which outputs an image with the same shape as the training data. The fully-connected layer accepts a latent vector z of size 100. This size was chosen to maintain consistency with the original GONet paper for comparative purposes. The transposed convolution layers each use a filter size of 4

and a stride of 2 resulting in the shapes $16\times16\times256$, $32\times32\times128$, $64\times64\times64$ and $128\times128\times3$. ReLU non-linearities are used after all but the last layer.
Dis uses four convolution layers followed by a single fully connected layer which outputs a logit for binary classification. The convolution layer shapes are identical to *Gen's* transposed convolution layers, including the filters and strides, but are applied in the opposite order. For *Dis* ELU non-linearities are used, a choice which has been shown to improve convergence [4]. *InvGen* shares its architecture with *Dis* except for the final fully-connected layer which outputs a latent vector z. ReLU non-linearities are again used. All three convolutional networks use Batch-normalization with $\epsilon = 2e^{-5}$ and $\mu = 0.1$, parameters specified in [11]. Additionally, experiments showed that the quality of generated images were improved through the inclusion of spectral-normalization [9] in the discriminator.

Training

Training of the models was undertaken in three stages. First the DCGAN was trained on positive examples only. The training procedure involves a minimax game whereby *Gen* attempts to produce examples from the positive manifold of the training data given a latent variable z. *Dis* tries to distinguish between real examples from the training data and fake images generated by *Gen*. Ultimately *Gen* is trying to fool *Dis* into classifying the images it produces as real. The optimal solution is a Nash equilibrium whereby neither network can improve their performance by unilaterally changing their policy. The *dsicrimintor* loss function is based on the standard binary cross-entropy loss and is given by

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p(x)} log D(x) - \mathbb{E}_z log(1 - D(G(z)) \quad (1)$$

The label inversion trick described in [4] is used for the *generator* loss function giving

$$\mathcal{L}_G = -\mathbb{E}_z log(D(G(z)) \quad (2)$$

One-sided label smoothing [15] was also employed to help prevent domination by *Dis* which has an easier task than *Gen* and can quickly end up with 100% accuracy. This problem arises due to the tendency of neural networks to act overly confident when extrapolating. Label smoothing entailed assigning classification targets not exactly equal to 1.0 for real images. In this case

$$y^i = 0.9 + 0.1 \times \mathcal{N}(0, I) \quad (3)$$

was selected. The use of only positive data is important for training the DCGAN because *Gen* needs to learn to create traversable images.
The second training step is for the Autoencoder *InvGen*. The same positive training data is used as for the DCGAN but now each input image is passed through *InvGen* to generate a latent vector z. *Gen* takes this latent vector as input and uses it to generate a fake image. The loss should be minimized when the generated image has identical pixel values to the input image. A suitable loss function is given by

$$\mathcal{L}(z) = (1-\lambda)\mathcal{L}_R(z) + \lambda\mathcal{L}_D(z) \quad (4)$$
$$\text{with } \mathcal{L}_R(z) = ||I - Gen(z)|| \quad (5)$$
$$\text{and } \mathcal{L}_D(z) = ||f(I) - f(Gen(z))|| \quad (6)$$

Once again $f(\,\cdot\,)$ is the features of the last convolutional layer of *Dis*. The hyper-parameter $\lambda \in [0, 1)$ is a relative weighting between the two loss contributions. During training of *InvGen*, the weights of *Dis* and *Gen* are frozen.
Finally, the *classification module* is trained on a small set of hand-labelled positive and negative images from the dataset. The training procedure involves passing the images though the entire architecture to output traversability probabilities which are compared to the ground truth labels through a binary cross-entropy loss. All training was done on a single NVIDIA Tesla K80 GPU.

3. Dataset and Features

Two datasets from different robotics applications were used for training separate generative traversability estimation models. The first dataset, Go-Stanford [6], contains 78711 128×128 RGB images collected by two fisheye cameras mounted to a turtlebot 2. All images were taken by manually operating the robot inside buildings on the Stanford campus. A small 2400 image subset is hand-labelled and contains equal amounts of positive and negative examples. This subset was used to train the classifier with a $50 : 50$ train-test split. The remainder of the data was automatically labelled using a velocity threshold and contains only positive images. Both the DCGAN and *Invgen* were trained on this entire positive subset. The train-test split for the positive data was $80 : 20$.

The second dataset used for this project is Kitti [3]. Several raw data sequences were compiled to build the training data for the model and an automatic labelling scheme was developed using the auxiliary sensor data provided. The full parsing procedure involved the following steps. First the raw sequences were concatenated into a single large pool of 43202 images. Next the vehicle velocity was used as a threshold to split the data into likely positive and negative groupings. A subset of these images were loaded along with their velocities and checked manually to tune the threshold velocity. Relatively few examples are required to train the classifier and hence, a small subset of the positive and negative data was collected for hand labelling. An application was created that shows

the user an image with corresponding velocity and asks for a positive or negative classification. This was used to efficiently hand-label a small classifier training set of 2400 images. All images were then resized to 128×128 through a centred crop followed by down-sampling. The same train-test splits were employed as for the GO-Stanford dataset.

For both robot applications, input images were normalized to the range $[-1, 1]$. The data was augmented at training time using random horizontal flips with 50% probability.

4. Experiments and Results

Hyper-parameters

The vast majority of experimental time for both datasets was spent training the DCGAN to accurately generate positive data examples. Several stabilizing techniques were employed which included hyper-parameter tuning. For the GO-Stanford dataset, the DCGAN was trained for a total of 41 epochs. This number seems relatively low, but training for longer actually resulted in a reduction in quality of the generated images. A constant learning rate of 0.0004 was used for *Gen*. For *Dis* the learning rate was initialized as 0.0007 and maintained for 10 epochs before being reduced to 0.0003. This higher initial learning rate proved necessary to prevent the *discriminator* accuracy immediately going to zero and staying there indefinitely. The one sided label smoothing was also delayed for the first 20 iterations for this same reason. A batch size of 64 was selected based on a parameter sweep. An Adam optimizer was employed with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The latter is the default setting while the former is widely used for training GANs. When training *InvGen*, a fixed learning rate of $1e^{-6}$ was used.

Similar hyper-parameters were trialled during attempted training of the DCGAN on the Kitti dataset. Learning rate and batch size sweeps were used in the search for some reasonable level of convergence. Unfortunately the combination of architecture and dataset size proved insufficient to achieve convergence, as elaborated on in the Kitti section of the results.

Go-Stanford

A comparison of generated and real positive images from *GO-Stanford* data is shown in Figure 2. Qualitatively, the generated images show traversable scenes similar to the real images. There is some noise and less colour in the fake images. The final *discriminator* accuracy on real images is 0.488 which is close to the Nash equilibrium value 0.5.

The final classification accuracy of the model on the *GO-Stanford* test data was 0.88. This is lower than the best results achieved in [7]. This result could likely be improved by employing a data annotation trick used in the original paper. Specifically, unlabelled data from the likely negative pool of data is annotated using the model. These annotated examples are then fed back in as training data. Additionally, it is possible that two many training epochs were used for the classifier,resulting in over-fitting.

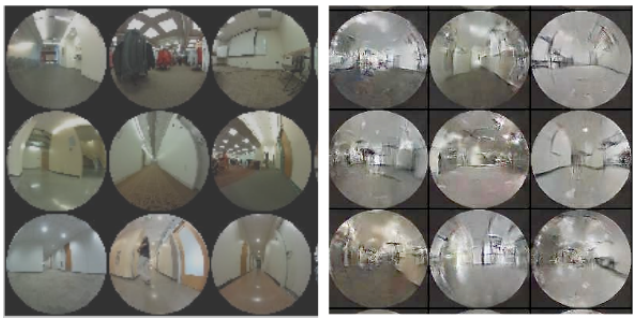

Figure 2. Real (left) and fake (right) example images from GO-Stanford.

When comparing real images with the images generated through the DCGAN and InvGen model, we see that qualitatively, the images bare reasonable resemblence and the most prominent features are preserved. This feature preservation is more conspicuous for the negative samples; however, the generated images are relatively dark compared to the originals. It is not entirely clear where this difference in contrast arises.

Testing Data	200	400	600	800	2400
Accuracy %	81.9	87.2	88.4	89.7	90.2

Table 1. Test Accuracy vs. Training Data size

Quantitatively, our main metric for measuring the model performance is the testing accuracy. As the training data increases, our testing accuracy increases. Overall, it reaches 90.2% accuracy, but still trails behind the performance in the original paper. Also referencing the performance of ResNet 50 and ResNet 152 on the same dataset, our model performance is comparable but did not achieve better performance as described in the paper. Due to the time and computing resource constraints, we have not trained our model with the same number of epochs and utilize both the stereo data and temporal structure. The use of an LTSM block within the classifier to process sequences of images had a notable performance enhancement effect in the original paper. We have assumed all measurements are conditionally independent which is a compromise that prioritizes simplicity at the cost of accuracy.

In addition, we also looked at the numerical difference between the generated and original images for positive and negative data manifolds. We compared both the Manhattan Distance and the Frobenius norm between generated and

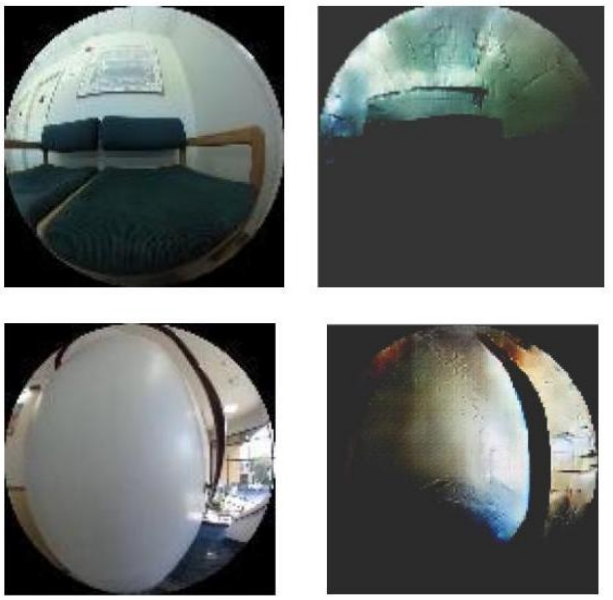

Figure 3. Real (left) and fake (right) example images form the negative data manifold of GO-Stanford.

Figure 4. Real (left) and fake (right) example images from the positive data manifold of GO-Stanford.

original images once normalized, and use the sample statistics as a quantitative metric to judge the model performance. From the Go Stanford dataset, we achieved reasonable performance, with means for both Manhattan Distance and the Frobenius norm relatively low compared to very different images generated by poorly tuned GAN models. However, both metrics do not reveal exactly how similar both images are, and could be distorted. Low values of both norms do not necessarily represent good model performance, but they could still be quantitative reference points.

Metrics	$\mu_{Positive}$	$\mu_{negative}$	$\sigma_{Positive}$	$\sigma_{Negative}$
L1 Norm	189.9	173.2	74.4	56.7
L2 Norm	81.4	79.2	85.4	69.7

Table 2. Sample Statistics of L1 and L2 Norms for the difference between original and generated images

Kitti

Unfortunately, when training the DCGAN model on the Kitti dataset, we consistently encountered modal collapse. This can be clearly observed in Figure 5 which shows a common theme for images returned by the generator. Modal collapse is where a diverse set of classes, or in this case outdoor scenes, are reduced to one or a few repeated images. As well as showing repeated patterns, the generated images are also not particularly similar to the inputs which indicates the distribution is being underfit. After training for more than 40 epochs and with signifcant amounts of hyper-

Figure 5. Real (left) and fake (right) example images from the positive data manifold of Kitti dataset.

parameter tuning, we were still not able to achieve satisfactory results. A few factors could have contributed to the results. First, the partition of the Kitti dataset used is significantly smaller than the GO-Stanford dataset. The section of the Kitti dataset used consists of approximately half the number of images as the Go Stanford dataset. This, lack of data is compounded by the small model which does not have the representational power needed to capture the complex distribution of outdoor environments compared with indoor environments.

5. Conclusion

In this project a semi-supervised generative approach was applied to the problem of traversabiltiy estimation for mobile autonomous robots. The approach performed well in the application from which it was originally developed, specifically for indoor robots. In contrast, the DCGAN used for feature extraction was unable to capture the distribution

of the data for the more diverse self-driving car dataset. This is likely because the dataset itself was not sufficiently large and the model also lacked the necessary representational power. Given more time, it would be desireable to train the model on the full Kitti dataset rather than a subset. Additionally, larger model architectures could be explored. The accuracy achieved on the indoor dataset is reasonable but does not match the results reported in the original GONet paper. One major difference is that we neglected the temporal dependence of measurements for the sake of simplicity. In future work a Recurrent network could be employed to improve the classification results.

Contributions & Acknowledgements

Nick

- Wrote training code for DCGAN
- Wrote training code for InvGen
- Wrote training code for Classifier
- Parsed the Kitti data
- Attempted training of DCGAN on Kitti
- Wrote report

Rongfei

- Built the model in PyTorch
- Trained DCGAN on GO-Stanford
- Trained InvGen on Go-Stanford
- Trained Classifier on GO-Stanford
- Collected and collated experimental results
- Wrote report

GONet Github Repo

`https://github.com/NHirose/GONET`

References

[1] T. Braun, H. Bitsch, and K. Berns. Visual terrain traversability estimation using a combined slope/elevation model. pages 177–184, 09 2008.

[2] J. Byrne and C. J. Taylor. Expansion segmentation for visual collision detection and estimation. In *2009 IEEE International Conference on Robotics and Automation*, pages 875–882, 2009.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Rob. Res.*, 32(11):1231–1237, Sept. 2013.

[4] I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[6] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, and S. Savarese. Go-stanford data-set. `https://cvgl.stanford.edu/gonet/dataset/`, 2018.

[7] N. Hirose, A. Sadeghian, M. Vázquez, P. Goebel, and S. Savarese. Gonet: A semi-supervised deep learning approach for traversability estimation. 03 2018.

[8] X. Meng, Z. Cao, S. Liang, L. Pang, S. Wang, and C. Zhou. A terrain description method for traversability analysis based on elevation grid map. *International Journal of Advanced Robotic Systems*, 15:172988141775153, 01 2018.

[9] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks, 2018.

[10] M. Pasini. 10 lessons i learned training gans for one year, 2019.

[11] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 11 2015.

[12] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[13] J. Sock, J. Kim, J. Min, and K. Kwak. Probabilistic traversability map generation using 3d-lidar and camera. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5631–5637, 2016.

[14] B. Suger, B. Steder, and W. Burgard. Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3d-lidar data. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015:3941–3946, 06 2015.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.