Deep Learning Traversability Estimator for Mobile Robots in Unstructured Environments*

Marco Visca[1], Sampo Kuutti[1], Roger Powell[2], Yang Gao[3], and Saber Fallah[1]

[1] Connected and Autonomous Vehicles Lab (CAV Lab), University of Surrey, Guildford, GU2 7XH, UK. m.visca@surrey.ac.uk
[2] Cybernetics Group, Remote Applications in Challenging Environments, UK Atomic Energy Authority, Culham Science Centre, OX14 3DB
[3] Space Technology for Autonomous and Robotic Laboratory (STAR LAB), Surrey Space Centre, University of Surrey, Guildford, GU2 7XH, UK

Abstract. Terrain traversability analysis plays a major role in ensuring safe robotic navigation in unstructured environments. However, real-time constraints frequently limit the accuracy of online tests especially in scenarios where realistic robot-terrain interactions are complex to model. In this context, we propose a deep learning framework trained in an end-to-end fashion from elevation maps and trajectories to estimate the occurrence of failure events. The network is first trained and tested in simulation over synthetic maps generated by the OpenSimplex algorithm. The prediction performance of the Deep Learning framework is illustrated by being able to retain over 94% recall of the original simulator at 30% of the computational time. Finally, the network is transferred and tested on real elevation maps collected by the SEEKER consortium during the Martian rover test trial in the Atacama desert in Chile. We show that transferring and fine-tuning of an application-independent pre-trained model retains better performance than training uniquely on scarcely available real data.

Keywords: Deep Learning · Transfer Learning · Mobile Robotics.

1 Introduction

Autonomous traversability analysis of unstructured terrains is a crucial task in many sectors, such as rescue robots for disaster areas, agriculture, nuclear plants, and space exploration. The primary goal of traversability analysis is to ensure the safety of the robotic system and reduce its dependency on human control by autonomously assessing the surrounding terrain. Moreover, in contrast to navigation in structured environments, where a clear distinction between obstacle

* This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. The authors are grateful to the Autonomous Systems Group of RAL SPACE for providing the SEEKER dataset.

and non-obstacle is possible, unstructured natural terrains present continuous difficulty values which mostly depend on the specific robot mobility capabilities. This makes the definition of safe trajectories considerably more challenging as the algorithm has to take more numerous and complex metrics into account. On the other hand, real-time navigation requirements often impose stringent constraints on the overall software complexity.

In this context, several terrain analysis algorithms, which differently trade-off between accuracy and computational speed, have been proposed [13]. Among them, square-grid cost maps based on geometric analysis are often considered the most successfully deployed on real systems [3] [16]. The main reason for their success is their implementation simplicity and relatively low computational workload. However, they often make use of overly conservative assumptions which could lead to sub-optimal navigation performance [3] [16]. Other works have proposed to use accurate physics-based simulators to assess the traversability of trajectories [6]. However, in spite of their accuracy which allows to maximise the optimality of trajectory planning, their computational workload is often unbearable for on-board resources and real-time navigation.

In recent years, deep learning methods have gained an increasing popularity for their ability of extracting features from high-dimensional inputs and their efficient parallel computing [8]. In this context, deep learning has demonstrated remarkable capabilities to improve the autonomy of mobile robots [9]. Other works have proposed to exploit deep learning models to estimate mobile robot traversability metrics [1] [2]. However, these methods often assess traversability over arbitrary-shaped patch of terrains (e.g. circular, or squared). Moreover, state-of-the-art deep learning methods often require substantial amounts of data to provide sensible predictions, while their availability is often limited for many robotic applications of interest.

In this paper, we propose a deep learning model to estimate traversability metrics from a simulator (Section 2). Our formulation has the advantage to explicitly address learning over feasible trajectories, thereby considering the robot mobility constraints and providing direct information in terms of trajectory planning. Furthermore, we propose to address the problem of data scarcity by developing a synthetic dataset based on the OpenSimplex noise algorithm (Section 3). We show that, despite some degradation in performance, a model trained on the synthetic dataset can retain characteristics of generic unstructured terrains and, thus, be used as the baseline model of a real use-case scenario. We show evidence of this by transferring on real data from the SEEKER Martian rover test trial in the Atacama desert in Chile and comparing the synthetic model performance with training based solely on the limited amount of available real data (Section 4).

2 Traversability Prediction Model

We propose to formulate the traversability prediction as an image classification problem by using a standard Convolutional Neural Network (CNN) architecture.

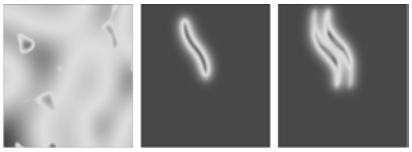


Fig. 1: The three input layers. From the left: the terrain elevation map, the trajectory centre, and the wheel trace.

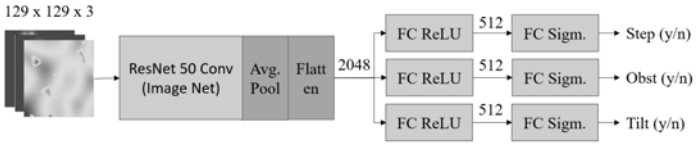


Fig. 2: The proposed CNN architecture.

This choice is motivated by the well-demonstrated CNNs' capabilities to process and find patterns in high-dimensional spatial inputs [4]. Moreover, we propose to assess terrain traversability directly over feasible trajectories. In this way, failure prediction can be achieved by considering the actual robot mobility constraints, thereby increasing the accuracy of prediction. The remainder of this section illustrates the proposed methodology.

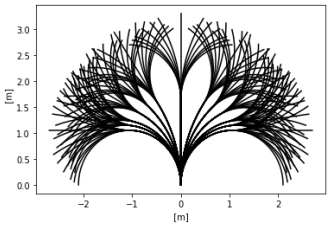
2.1 Input Features

To enable the use of CNN architectures, an approach is devised which gives the terrain and trajectory input features a three-channel image-like representation. Each channel is a 129×129 grid, where each pixel position corresponds to an (x,y) coordinate with respect to the rover centre (between -4 and +4 meters) and the robot is assumed positioned in the centre of the map and oriented in the positive direction of the vertical axis. A visual description of the three channels is illustrated in Fig. 1. The three channels from left to right are: (1) the terrain elevations, where the value of each cell is the normalized z elevation value for that (x,y) coordinate, (2) the trajectory left from the robot on the map, where the channel has its peak (value of 1) at the robot centre and exponentially decreases to 0 at the wheel track, and (3) the trace left from the wheels on the map, where each trace has its peak at the wheel centre and exponentially decreases to 0 outside the wheel; furthermore, a higher value is given to the cells where both the front and rear wheels pass.

In this way, the feature processing can be addressed on regions of the terrain of particular relevance to the failure prediction (i.e. the regions under the robot ride and the wheels) and directly over feasible trajectories. Hence, each 129×129×3 image represents one terrain-trajectory input feature that is fed to the neural network for traversability prediction.

Wheelbase [cm]	119.8
Track Width [cm]	83.5
Swing Radius [cm]	52.0
Ride Height [cm]	10.8
Max Step [cm]	12.0
Max Inclination [°]	36

(a)



(b)

Fig. 3: (a) Seekur Jr. Robot (Courtesy of Generation Robots), and (b) portion of the robot action space for no initial point turn rotation.

2.2 Network Architecture

Figure 2 illustrates the proposed neural network architecture. The ResNet50 network [5] pre-trained on ImageNet [12] is chosen as the baseline of the prediction model. This choice is motivated by the remarkable performance demonstrated by Imagenet pre-trained residual networks as baseline architectures for transfer learning problem. Indeed, although our inputs are quite dissimilar from ImageNet images, exploiting pre-learned low level features (e.g. vertical or horizontal edges, which are common to all image classification problems) has proved to give faster convergence than training from scratch [15]. Conversely, the original top Fully Connected (FC) layer is removed and replaced with three randomly initialized FC layers to learn the application-dependent features (one for each failure event as described in Section 2.3). Each FC layer has 512 neurons and randomly initialized weights. Finally, three FC layers with sigmoid activation functions provide the failure predictions.

2.3 Robot Model and Failure Events

A simplified kinematic robot model is developed in Python to emulate the robot navigation over unstructured terrains. The dimensions and the mobility capabilities of the robot model are selected according to the 4-wheel skid-steering Seekur Jr robot [10]. Its main features are summarised in Fig. 3a. Hence, each trajectory is defined according to the mobility capability of our robot as a combination of an initial point turn rotation (18 rotations for multiple of 20˚including no rotation) followed by two arcs of length 1.65 meters and different radius (13 different possibilities). This leads to a total of 3042 possible trajectories 3.3 meters long each. Fig. 3b illustrates a portion of the action space for no initial rotation.

Three failure events are defined: *step*, *obstacle*, and *tilt*. Failure for *step* occurs when the differential elevation of the terrain underneath the robot wheels for two consecutive time steps is above the maximum traversable step of the robot. Failure for *obstacle* occurs if one or more of the terrain elevation points underneath the robot base is higher than the robot ride height. Finally, failure for *tilt* occurs if the inclination of the robot with respect to the vertical

Algorithm 1 Generating natural terrains with OpenSimplex

1: **for** all x,y **do**
2: $m = noise2d(x * \alpha_m, y * \alpha_m) * \beta_m + \gamma_m$
3: $p = (noise2d(x * \alpha_p, y * \alpha_p) * \beta_p + \gamma_p)^{\delta}$
4: $w = intrp(noise2d(x * \alpha_w, y * \alpha_w) * \beta_w + \gamma_w, u, d)$
5: $Z(x, y) = p * w + m * (1 - w)$
6: **end for**
7: **return** Z

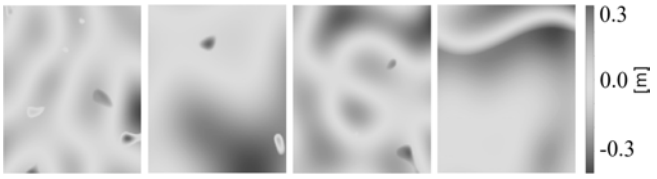


Fig. 4: Examples of 8 m × 8 m maps generated using the OpenSimplex algorithm.

direction is above the maximum traversable inclination. To reduce the computational workload, the dynamics of the system are not taken into account. This is a reasonable assumption for robotics navigation at low speed (which could be the case in some realistic scenarios, such as planetary exploration and nuclear reactor maintenance) [16] [7]. Traverse is simulated by placing the robot on sequential trajectory points (equally spaced at 6 cm intervals along each arc) and computing for each one of them the robot static pose and orientation and the elevation of the points under the rover base. Hence, the occurrence of the three failure events is recorded for each combination of terrain and trajectory.

3 Dataset Generation

3.1 OpenSimplex Synthetic Maps Generation

To reduce the data scarcity problem of mobile robot applications, synthetic maps are generated using the OpenSimplex noise algorithm, a popular approach to generate realistic unstructured environments [14]. In this work, the OpenSimplex Python API is used along with three filtering techniques to render realistic terrains. A description of the approach is illustrated in Algorithm 1. The *noise2d* function is the Python API which takes as input an (x,y) coordinate and outputs a number in [−1,1] according to the OpenSimplex algorithm. Hence, additional heuristic parameters are used to filter the result of Opensimplex. Specifically, α_m, α_p, and α_w act on the noise frequency, β_m, β_p, and β_w scale the output, while γ_m, γ_p, γ_w offset the output. In this way, α_m, β_m, and γ_m are set in Line 2 to control the generation of obstacles. Line 3 controls the generation of plain regions by using a smoothing coefficient $\delta \in [0,1]$ in addition to the α_p, β_p, and γ_p parameters. Line 4 controls the interpolation between obstacles and plains, where *intrp* is a function returning 1 if the first argument is larger than u, 0 if it

is lower than d, or linearly interpolates between 0 and 1 otherwise. Finally, Line 5 combines the results of the previous three operations by interpolating between obstacles and plain regions and assigning the elevation value to the elevation matrix Z. The implementation of Algorithm 1, with the parameters used in this paper, is made available at UNSTR-NAV. We remark that the process is fully automated and, by different tuning of the algorithm parameters, different terrain conditions can be achieved, such as rough, wavy, and smooth terrains, as well as mountains and depressions. Fig. 4 illustrates some examples of generated maps.

3.2 Dataset Collection and Training

A total of 56840 synthetic elevation maps is generated with the method described in Section 3.1. Then, the robot traverses each map with 3042 trajectories and collects failure events with the method described in Section 2.3. The resulting dataset is composed of approximately 1.7e8 samples. The dataset is randomly divided among training (90%), validation (8%), and test (2%) datasets. Moreover, since safe trajectories are considerably more numerous than failures for each terrain (90.4% against 9.6%), a reduced and better balanced subset is extracted for training and validation to avoid excessive bias in prediction (5.7e5 and 4.9e4 samples respectively). Conversely, all maps and trajectories of the test set are retained to assess final performance (3.4e6 samples).

The network is trained by means of supervised learning and binary cross-entropy loss function [11]. The parameters used for training are: RMSprop optimizer, learning rate 1e-4, dropout 20%, and L2 regularization 0.001. During the first epoch, only the 3 FC layers are trained, while the ResNet weights are kept frozen. Then, the whole network is unfrozen and trained for 10 epochs.

4 Results

4.1 Prediction Performance - Synthetic Dataset

The results of the trained model on the synthetic test dataset are illustrated in Table 1. An overall accuracy of 98% can be observed, with the accuracy of each failure event above 96%. However, since safe and unsafe trajectories are extremely unbalanced in the test set (roughly 1e7 vs 4e5 samples respectively), accuracy by itself can not be considered as a representative metric. For this reason, recall, precision, and F1 score are used to provide a more informed representation of the actual model performance. Specifically, an overall high recall (94.4%), and low precision (68.4%) are observed, with a consequent F1 score of 79%. This means that the network tends to be conservative, being able to correctly predict the majority of dangerous trajectories, but at a price of relatively high false-positive rate. A possible explanation for the low network precision could depend on the high sensitiveness of a correct prediction to small variations in the image. Indeed, even just one different pixel in the elevation map could lead the same trajectory to be safe or unsafe, making it a relatively challenging

Table 1: Synthetic Dataset Model Performance

(a) Confusion Matrices

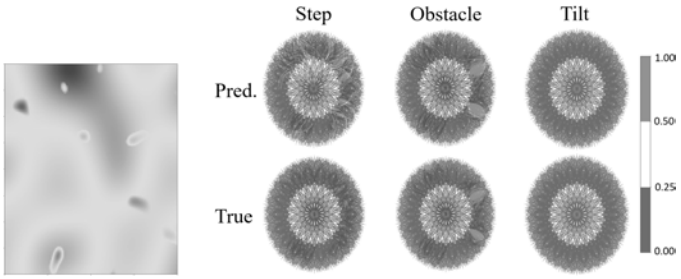
Step	**Pred. Safe**	**Pred. Fail**
True Safe	3126951	113457
True Fail	13615	201689

Obstacle	**Pred. Safe**	**Pred. Fail**
True Safe	3211837	53842
True Fail	9644	180389

Tilt	**Pred. Safe**	**Pred. Fail**
True Safe	3425479	15806
True Fail	163	14264

(b) Classification Performance

	Acc.	**Recall**	**Prec.**	**F1 Score**
Step	0.963	0.937	0.640	0.760
Obstacle	0.982	0.950	0.770	0.850
Tilt	0.995	0.989	0.474	0.641
Overall	0.980	0.944	0.684	0.793



(a) (b)

Fig. 5: Qualitative example of events prediction. (a) Synthetic elevation map, (b) prediction (top) and ground truth (bottom) of step, obstacle, and tilt probability.

image classification problem. Therefore, while the model could have successfully learned macro-associations of elevation points and trajectories to safe or unsafe areas, it might struggle to seize much more subtle local differences. However, the tendency to conservativeness is not excessively detrimental for the specific application of robotic navigation as long as safety is ensured. For instance, an increased rate of false alarms can be tolerated if it results in reliable identification of dangerous trajectories.

A qualitative example of the failure prediction is illustrated in Fig. 5. The image on the left represents the elevation map under analysis, while each subsequent image represents the robot action space (i.e. 3042 different trajectories from the map centre) with probability of failure occurrence encoded with 3 colours (**green**: less than 0.25, **yellow**: 0.25-0.5, **red**: more than 0.5). The network accurately predicts most of the dangerous trajectories due to obstacle, as wells as the absence of tilt failures. Conversely, some conservativeness can be observed for the step failure predictions. Nevertheless, trajectories laying in completely failure-free areas of the map are correctly predicted as safe which is

(a) (b)

Fig. 6: (a) Image from the Atacama desert and (b) the extracted elevation map.

a sign that the network has successfully learned to differentiate among macro distributions of elevation points.

The experiments are tested on a Intel Core i5 6500 Skylake and NVIDIA GeForce GTX 1050 graphic card. The resulting running time for a complete simulation of the 3042 trajectories is assessed at around 58s and 17s for the Python simulator and deep learning model respectively. Therefore, the deep learning model is able to reduce the computational time by approximately 70%.

4.2 Prediction Performance - Planetary Mission Use Case

In this section, the traversability estimator is analysed on real unstructured terrains from the SEEKER Martian rover test trial in the Atacama desert in Chile [17]. Fig. 6a shows an example of stereo camera image from which the test site elevation maps have been generated. From the SEEKER dataset, elevation maps from 10 traverses are selected for a total of 1289 m. The real data are partitioned in 8×8 metres elevation maps to be consistent with the dimensionality of the input data accepted by our model. An example of extracted map from the SEEKER dataset can be observed in Fig. 6b. The final dataset is composed of 645 maps, which is approximately 1.1 % of the synthetic dataset size. Moreover, data augmentation is performed to help reducing the data scarcity problem (by rotating each image by 90, 180, and 270 degrees). Hence, each sample is labelled according to the three failure events by running the traverse simulator. Finally, one of the rover traverse is randomly selected and all its samples are removed from the training set to be used as the test set.

First, the transferring performance on the SEEKER test set of our baseline model (i.e. pre-trained on the synthetic data but without further training on the real data) can be observed in Table 2. We observe that no failure for tilt is present in the real data. Therefore, we are limited in the evaluation of this event. However, the network is able to predict 99.9% of the samples correctly as tilt safe. Furthermore, also the step and obstacle classes are extremely unbalanced towards safe trajectories. Similarly to the synthetic data, the accuracy is above 99% (i.e. the model is able to classify nearly all the samples correctly as safe). Conversely, the performance for the failure prediction considerably drops both for the step and obstacle events. A possible explanation for this is that the synthetic data may not represent with sufficient realism many of the geometric distributions responsible for failure events in the real data. Specifically, the step

Table 2: Real Dataset Transferring Model Performance

(a) Confusion Matrices

Step	**Pred. Safe**	**Pred. Fail**
True Safe	132880	811
True Fail	109	48

Obstacle	**Pred. Safe**	**Pred. Fail**
True Safe	132622	521
True Fail	269	436

Tilt	**Pred. Safe**	**Pred. Fail**
True Safe	133735	113
True Fail	0	0

(b) Classification Performance

	Acc.	**Recall**	**Prec.**	**F1 Score**
Step	0.993	0.306	0.056	0.094
Obst.	0.994	0.618	0.456	0.525
Tilt	0.999	-	-	-
Overall	0.995	0.561	0.251	0.347

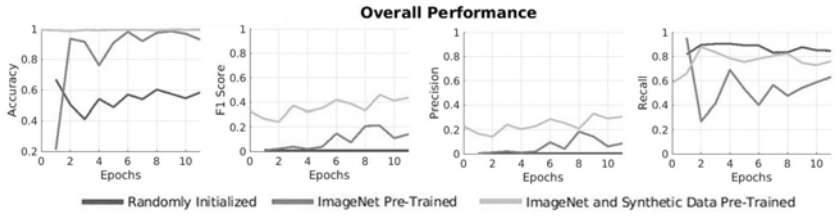
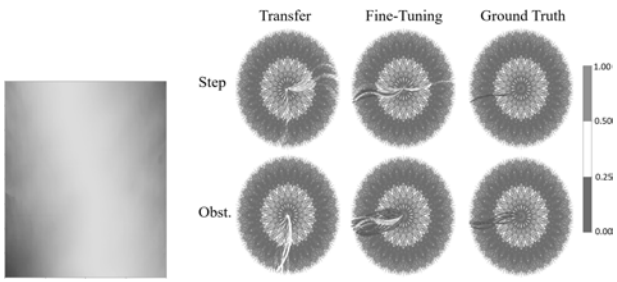


Fig. 7: Overall test performance of differently pre-trained models while training on the real data. Epoch 0 on the yellow line is the transfer learning performance (i.e. our synthetic pre-trained baseline model before training on the real data).

event seems the most largely influenced both in terms of recall and precision (respectively 46% and 6%), while the obstacle event has been able to retain considerably better performance (recall of 62% and precision of 46%), which means that the network has learned to generalize more effectively to this type of failure in the real scenario.

Then, the performance of training on the SEEKER dataset is analysed for three differently pre-trained models: (1) a model with randomly initialized network parameters, (2) a model pre-trained on ImageNet only, and (3) our baseline model initialized with ImageNet weights and pre-trained on our synthetic dataset as described in Sections 2 and 3. Hence, the three models are trained on the SEEKER training set for 11 epochs and the maximum F1 score on the test set is used as the convergence point of their performance. Fig. 7 summarises our findings. The randomly initialized network fails to learn useful features in the dataset, resulting in poor performance. Meanwhile, the network pre-trained on ImageNet shows some initial improvement, learning useful features for its task, but overfits after 8 epochs at 21% F1 score, resulting in 54% recall and 14% precision. Conversely, the model pre-trained on our synthetic dataset improves its performance more effectively when fine-tuning on the real dataset, resulting in a final F1 score, recall and precision of 46%, 76% and 31%, respectively. This provides evidence of the improved capability of our baseline model to transfer



(a) (b)

Fig. 8: (a) Elevation map from SEEKER dataset, and (b) step and obstacle predictions before and after model fine-tuning compared with the ground truth.

features of relevance to the traversability analysis problem. Most importantly, this model is able to outperform the ImageNet model in terms of recall, demonstrating it has learned how to correctly classify failure events more reliably. In Fig. 8 a qualitative example is illustrated of the prediction capabilities of our baseline model before and after fine-tuning on the real data. As with the previous results, the transferred model is initially unable to correctly identify the dangerous trajectories both for the step and obstacle events while evidently improves after the model fine-tuning.

5 Conclusion and Future work

This paper has investigated the use of deep learning as a traversability estimator for mobile robots in unstructured terrains. We provided insights on the benefits of the proposed method to predict the occurrence of failure events over feasible trajectories and at a fraction of the time of a sequential traverse simulator. We also showed that by generating a domain-independent synthetic dataset we can learn general features for traversability analysis. Then, by fine-tuning the learned model on the domain-specific real-world data, we can transfer the knowledge to enable the deep neural network to learn traversability analysis on datasets that would be excessively small to train on. Therefore, this technique enables more efficient learning for domains where large real world datasets are not available and where deep learning might not otherwise be a feasible solution due to the scarcity of data.

We are extending this work in several directions. While binary failure metrics could be sufficient to enforce safety in navigation, they cannot provide adequate information to perform optimal path planning. Future works may consider an extension of learning to include more complex continuous metrics. Finally, we discussed how the representativeness of the synthetic data could have a crucial impact during transfer learning. In future works, this could be addressed by generating more realistic synthetic data and by using algorithms specifically devised for efficient transfer learning (e.g. meta-learning).

References

1. Blacker, P., Bridges, C.P., Hadfield, S.: Rapid prototyping of deep learning models on radiation hardened cpus. In: 2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS). pp. 25–32 (July 2019)
2. Chavez-Garcia, R.O., Guzzi, J., Gambardella, L.M., Giusti, A.: Learning ground traversability from simulations. IEEE Robotics and Automation Letters **3**(3), 1695–1702 (2018). https://doi.org/10.1109/LRA.2018.2801794
3. Goldberg, S.B., Maimone, M.W., Matthies, L.: Stereo vision and rover navigation software for planetary exploration. In: Proceedings, IEEE Aerospace Conference. vol. 5, pp. 5–5 (2002)
4. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al.: Recent advances in convolutional neural networks. Pattern Recognition **77**, 354–377 (2018)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015)
6. Helmick, D., Angelova, A., Matthies, L.: Terrain adaptive navigation for planetary rovers. Journal of Field Robotics **26** (04 2009)
7. Iqbal, J., Tahir, A.M., ul Islam, R., Riaz-un-Nabi: Robotics for nuclear power plants — challenges and future perspectives. In: 2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI). pp. 151–156 (Sep 2012)
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
9. Ono, M., Fuchs, T.J., Steffy, A., Maimone, M., Yen, J.: Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In: 2015 IEEE Aerospace Conference. pp. 1–10 (March 2015)
10. ROBOTS, G.: Seekur Jr mobile robot, `https://www.generationrobots.com/en/402399-robot-mobile-seekur-jr.html`
11. Ruby, U., Yendapalli, V.: Binary cross entropy with deep learning technique for image classification. International Journal of Advanced Trends in Computer Science and Engineering **9** (10 2020)
12. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge. CoRR **abs/1409.0575** (2014)
13. Sancho-Pradel, D., Gao, Y.: A survey on terrain assessment techniques for autonomous operation of planetary robots. JBIS - Journal of the British Interplanetary Society **63**(5-6), 206 – 217 (May 2010)
14. Spencer, K.: Introducing OpenSimplex Noise, `https://www.reddit.com/r/proceduralgeneration/comments/2gu3e7/like_perlins_simplex_noise_but_dont_like_the/ckmqz2y/`
15. Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: Artificial Neural Networks and Machine Learning – ICANN 2018. pp. 270–279. Springer International Publishing, Cham (2018)
16. Winter, M., Rubio, S., Lancaster, R., Barclay, C., Silva, N., Nye, B., Bora, L.: Detailed description of the high-level autonomy functionalities developed for the exomars rover. In: 14th Symposium on Advanced Space Technologies in Robotics and Automation (06 2017)
17. Woods, M., Shaw, A., Tidey, E., Van Pham, B., Simon, L., Mukherji, R., Maddison, B., Cross, G., Kisdi, A., Tubby, W., Visentin, G., Chong, G.: Seeker—autonomous long-range rover navigation for remote exploration. Journal of Field Robotics **31**(6), 940–968 (2014)