Traversability Learning from Aerial Images with Fully Convolutional Neural Networks

Borges, C. D. B
Programa de Ps-Graduao em Engenharia Eltrica e de Computao
Universidade Federal do Cear – Campus Sobral
Sobral, Brazil
davidborges@protonmail.com

***Abstract*—Traversability computation is a key step for robot path planning. It allows the incorporation of knowledge about traversable and non-traversable terrain into the robot's planning algorithms. It is possible to use aerial data to compute traversability maps for large regions of land and use them to assist ground robot navigation. Most published methods to generate these maps from aerial images use terrain classification in combination with predefined traversability values for each class or simple heuristics. However, current methods based on classification or handcrafted heuristics present limitations with regard to output quality and processing time. To overcome these issues, this work describes a new method to compute a traversability map from an aerial image in one forward pass through a fully convolutional neural network. It analyzes the technical features of this approach relative to other methods. The experiments yield evidence that the proposed model can generate traversability maps faster while also providing lower error outputs.**

***Index Terms*—traversability, mapping, convolutional neural networks, aerial images, ground robot, path planning**

I. Introduction

The traversability index was proposed by Seraji [1]. It is a simple way to distinguish traversable from non-traversable terrain and apply this knowledge to navigation strategies for robots. Traversability is a quantification of how much a given region is suitable to be crossed by a moving robot or vehicle. The concept was extended and applied in robot planning tasks. Researchers used a wide range of sensing devices and methods to estimate traversability maps from the ground [2]–[6].

With the rise in popularity of low cost multirotor drones and the increased availability of high resolution satellite images, a new idea to assist in robot navigation surfaced: the use of aerial images to add long range planning capabilities to ground robots. This was explored, for example, by [7]–[11] and others. In these works, some of the proposed methods to generate traversability maps require depth data [7], [10], while others need only RGB images [8], [9], [11]. The core traversability computation approach also varied between heuristics [7], [11] and classification combined with heuristics [8]–[10].

However, classification is computationally expensive if executed over all sub-regions of an image. On the other hand, using only handcrafted heuristics disregards the possible advantage of learning hidden structures in image data. This work proposes a machine learning approach to compute traversabilities from raw image data. It is an extension of the method described in [11] that replaces the heuristic-based sub-region computations with end-to-end traversability estimation based on a fully convolutional neural network. This method is described in detail in section II. The experiments performed to validate the network architecture are explained in section III. Results and a discussion about the merits and drawbacks of the method are explored in sections IV and V. Section VI closes this paper with a brief review.

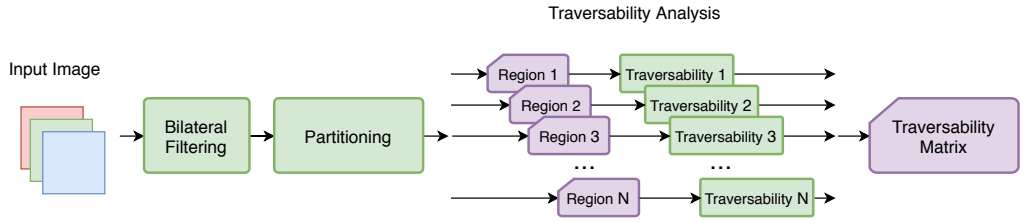
II. Proposed Method

The traversability mapping approach previously described in [11] is shown in Fig. 1a. The input image, which is an aerial view of the ground region, is submitted to bilateral filtering and partitioned into sub-regions that may overlap. A handcrafted heuristic is used to compute a degree of traversability for each sub-region. These values are then stored in a traversability matrix that preserves the relative position of the sub-regions. This matrix is used to build a graph representation of the region and its traversabilities; in this graph, vertices represent sub-regions and edge weights are inversely proportional to the traversability between adjacent regions. A shortest path search algorithm is then used to find a path between any chosen source and target sub-regions.

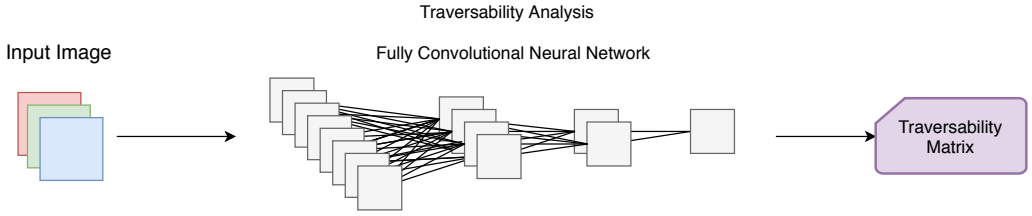
The present work builds upon this approach, but replaces the conventional preprocessing, partitioning and traversability computation pipeline with a fully convolutional neural network (FCN), as displayed in Fig. 1b. The proposed traversability estimation method uses an FCN to derive a traversability matrix directly from raw RGB images. The Traversability FCN (TFCN) architecture is shown in Fig. 2. It is a small network, composed of three phases of convolutions and pooling. Batch normalization steps are performed after convolutional layers as a way to mitigate internal covariate shift and reduce training time [12]. In order to counter overfitting, the TFCN also performs dropout [13] and L2 regularization [14].

III. Experiments

To validate the new traversability estimation method, the proposed TFCN was trained and tested on the Aerial Traversability and Planning Dataset [11]. This section describes ATPD, the training configuration and the cross-validation approach employed in the experiments.



(a) Traversability mapping approach from [11].



(b) TFCN proposal

Fig. 1. Difference between traversability computation models.

A. Dataset

The Aerial Traversability and Planning Dataset (ATPD) [11] contains 8 orthorectified aerial images of size 1000×1000 pixels, with specified spatial resolutions of about $0.3 \times 0.3\ m^2/p$. Each image has a corresponding traversability label, in which every pixel from the original image is assigned a binary value: 1 if the original pixel is in a traversable sub-region and 0 otherwise. These images and labels were used to assemble the training inputs and outputs. To fit the size of the TFCN output, the traversability labels were resized to 125×125 pixels.

B. Data Augmentation

ATPD only includes 8 aerial images with labeled traversabilities. It is a very small dataset. For this reason, the training sets were augmented with random rotations, shifts, flips, shear and zoom. The range of rotations was 0 to 360; maximum of 30% horizontal or vertical shift; horizontal and vertical flips; maximum 30 shear; and 20% zoom. In the cases the outcome of these operations required padding (e.g. rotation, shift and shear), the resulting image was padded by reflection, so that added pixels were similar to those in their neighboring region.

C. Training

The TFCN was trained with Adam optimization [15], mean squared error (MSE) loss and early stopping [16]. Training was carried on for a maximum of 100 epochs, but interrupted if there was no validation loss improvement of at least 0.001 after 20 consecutive epochs. A checkpoint routine was used to save the weight configuration with the best validation loss during each training execution. This configuration was the output of the training phase in the experiments.

D. Cross-validation

An image-based leave-one-out cross-validation was employed to study the generalization capability of the TFCN. In this approach, each of the eight aerial images from ATPD was used once for testing, while the other seven were selected to train the TFCN. These seven images were further split into a training set composed of six images and their augmentations; and a validation set of one image and its augmentations. Training sets had $6 \times 4 \times 32 = 768$ samples (images $\times$ steps $\times$ augmentations), while validation sets had $1 \times 4 \times 32 = 128$.

The chosen cross-validation approach is fair because it guarantees that samples from the test set were not used in any way during training. Moreover, the best weight configuration is chosen based on the loss computed on the validation set, which also has no intersection with the training and test sets.

IV. Results

The behavior of training and validation losses, along with visualizations of the test images, test labels and TFCN outputs from each leave-one-out step can be seen in Figs. 3–10. Because training was done with early stopping, the number of epochs in each leave-one-out step was variable. This is shown in the graphs as an abrupt interruption of the loss curves after 20 epochs without validation loss improvement. The average time to complete one epoch was 28 seconds. The full leave-one-out took 4 hours to complete.

TABLE I
Cross-Validation Results

Leave-one-out Step	Total Epochs	Best Epoch	Training Loss	Validation Loss	Test Loss
1	100	93	0.0974	0.0949	0.1265
2	83	63	0.1106	0.0951	0.1199
3	65	45	0.1103	0.0701	0.1161
4	57	37	0.1232	0.0589	0.1011
5	41	23	0.1643	0.0784	0.0712
6	52	32	0.1336	0.0869	0.0820
7	45	25	0.1627	0.5474	0.2302
8	79	59	0.0950	0.1093	0.5137

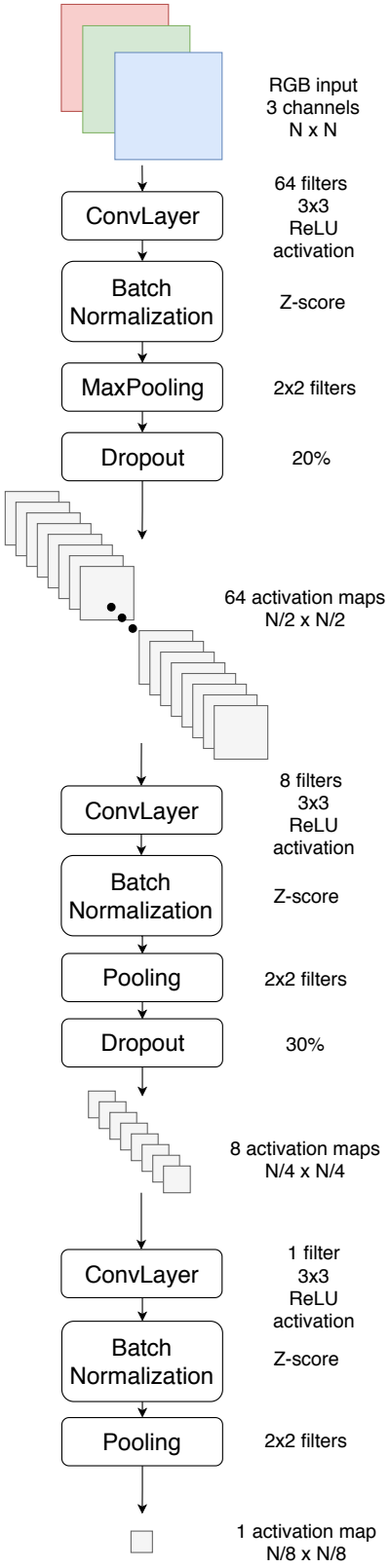


Fig. 2. TFCN architecture.

The training, validation and test losses obtained using the best weight configuration in each leave-one-out step, together with the number of epochs to finish the training phases, are summarized in Table I. These losses are the sum of the MSE and the L2 regularization penalties.

V. Discussion

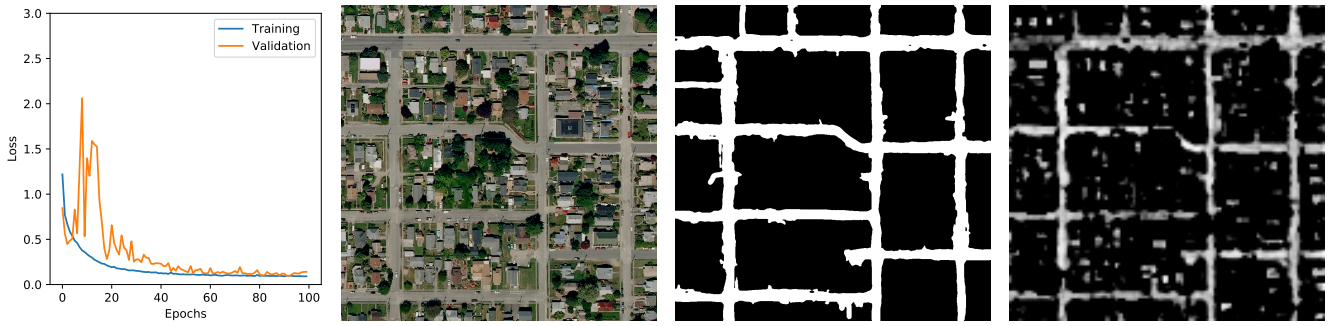
It can be seen in Figs. 3–9 that the TFCN outputs closely resemble their corresponding ground-truth labels. This is supported by their relatively low test loss values in Table I. Besides, a number of issues can be observed in the traversability maps. For example, in the output of Fig. 3, some road patches were marked as not traversable, making it difficult to find shortest paths that might go through these sub-regions. In Fig. 4, roofs were set as traversable, possibly because their colors and textures look almost the same as the roads'. The power cables and trails in Fig. 5 were given low traversabilities, even though they do not represent real obstacles for a ground robot. Despite the notable global performance of the TFCN, these small failures can be significant when applying the model in a real scenario. Further research is needed to deal with these issues without compromising the overall achievement.

A larger problem, however, can be identified in Fig. 10, in which there is an attempt to compute the traversability of image 8 from ATPD. In this case, the TFCN provided a bad traversability estimate. The test loss for this image was significantly higher. An hypothesis to explain this behavior relies on the fact that image 8 is remarkably different from the others with respect to terrain type. Image 8 portrays a sandy landscape with scattered vegetation, however, sandy terrain is not present in other ATPD images. Since the TFCN had no training with this kind of terrain, the performance of the model on image 8 was degraded. This problem also highlights that the TFCN is relying heavily on color information to compute its output, since a strong change in color between the training and test sets produced such a negative impact on performance. A possible advance would be to find a method to weight the influence of color and texture in the inner-workings of the TFCN and improve its generalization capabilities.

TABLE II
MSE Comparison

Input Image ATPD ID	Borges et al. [11] MSE	TFCN MSE
1	0.1566	0.1245
2	0.2066	0.1177
3	0.1714	0.1137
4	0.1265	0.0986
5	0.1394	0.0688
6	0.1420	0.0796
7	0.1756	0.2279
8	0.1488	0.5114

A comparison was made to evaluate the TFCN results relative to previous work about traversability based on aerial images. This comparison required a quantitative measurement of the quality of traversability outputs, either by an accuracy or error estimate over a test set. At the time of writing, only one

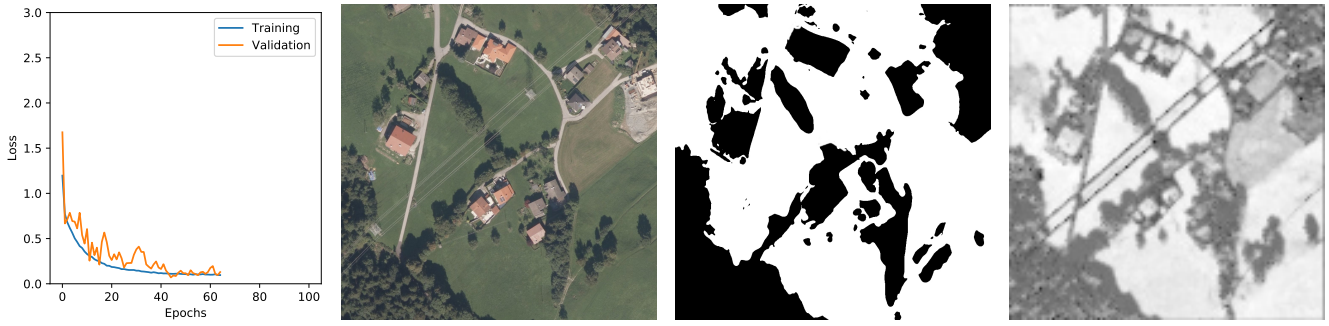


(a) Training and validation losses (b) Test image (c) Test label (d) TFCN output

Fig. 3. Leave-one-out step 1.

(a) Training and validation losses (b) Test image (c) Test label (d) TFCN output

Fig. 4. Leave-one-out step 2.



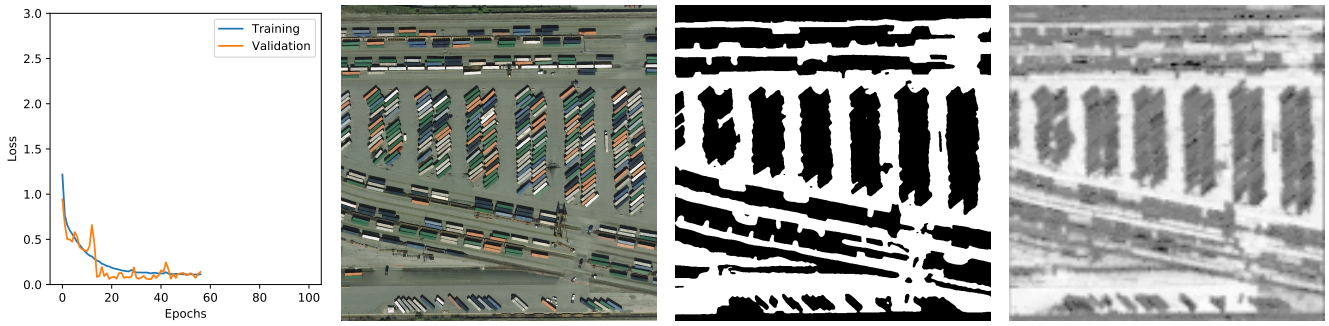
(a) Training and validation losses (b) Test image (c) Test label (d) TFCN output

Fig. 5. Leave-one-out step 3.

work [11] could be reproduced and tested for this comparison. Table II contains the MSE between the test labels and the model outputs for all images in ATPD. The TFCN surpassed [11] in almost all cases (1 through 6) by a significant margin. In cases 7 and 8, however, it was surmounted. For image 8, the cause is clear: it has been seen before that this image was a challenge to the TFCN, given the lack of training data in sandy terrain. For image 7, a good hypothesis to account for the unusually high MSE is that in this specific leave-one-out step, image 8 was used for validation and, for the same reason mentioned before, it disrupted the training process. The overall MSE comparison shows that the TFCN overcomes the mentioned competitor with respect to quality of traversability outputs. However, further research is needed to fix the observed generalization issue.

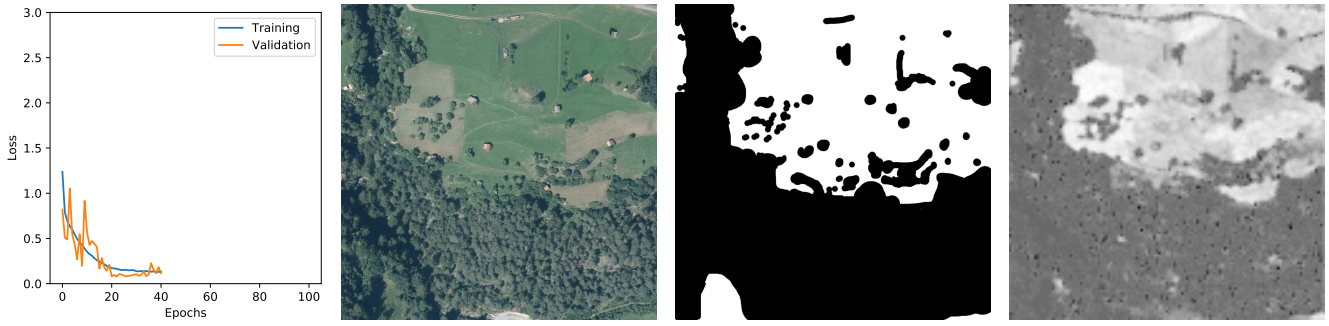
To analyze the TFCN with regard to time, two papers were selected. The works by Hudjakov and Tamre [9] and Borges et al. [11] were chosen because they clearly state the processing times achieved by their models. Moreover, [9] also uses CNNs to compute traversabilities, making it a good candidate for comparison and validation of a new CNN architecture.

The processing times in Table III regarding the work by H. & T. were extracted directly from the paper, in which the authors compute traversability maps over images of 2000×2000 pixels. The values for images of 1000×1000 pixels were extrapolated from information about total patterns in the image



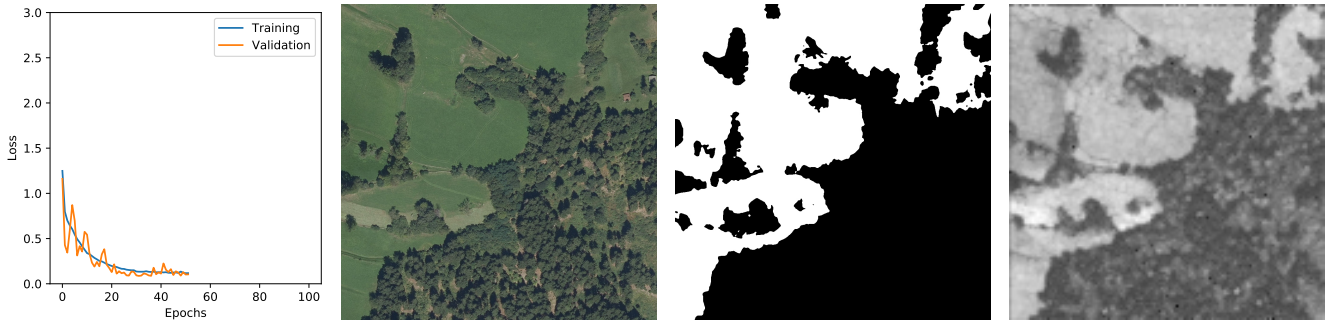
(a) Training and validation losses (b) Test image (c) Test label (d) TFCN output

Fig. 6. Leave-one-out step 4.



(a) Training and validation losses (b) Test image (c) Test label (d) TFCN output

Fig. 7. Leave-one-out step 5.



(a) Training and validation losses (b) Test image (c) Test label (d) TFCN output

Fig. 8. Leave-one-out step 6.

TABLE III
TIME COMPARISON

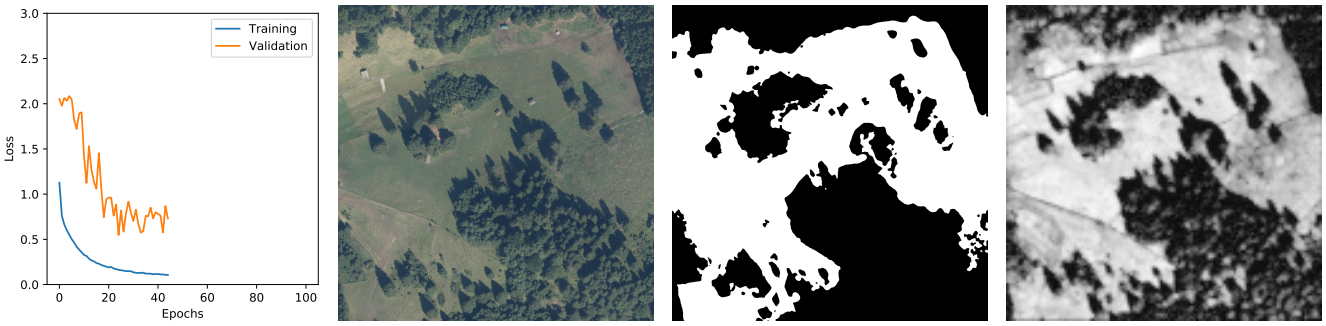
Input Image Size	H. & T. [9] Time (s)	Borges et al. [11] Time (s)	TFCN Time (s)
1000 × 1000	1250 ^{b*} 11 ^{c*}	0.834 ^{a}	0.775 ^{a}
2000 × 2000	5000 ^{b} 44 ^{c}	3.336 ^{a*}	2.583 ^{a}

a. Intel Core i5 650 CPU
b. Intel Core i7 920 CPU
c. 2x Nvidia GTX 570 GPUs
* Estimate based on number of patterns per image and processing speed

and number of patterns classified per second, as stated in the paper. The remaining processing times were obtained from implementations of the models. It is clear from Table III that the TFCN greatly reduces processing times when compared with [9], even when running on less powerful hardware. The reason for this large advantage is that the TFCN generates the entire traversability map in only one forward pass through the network, while the CNN model proposed by H. & T. needs to perform classification on millions of overlapping patterns to produce a single map. The TFCN also outperforms [11] by a narrower margin. This difference, however, could greatly increase if the input images were larger.

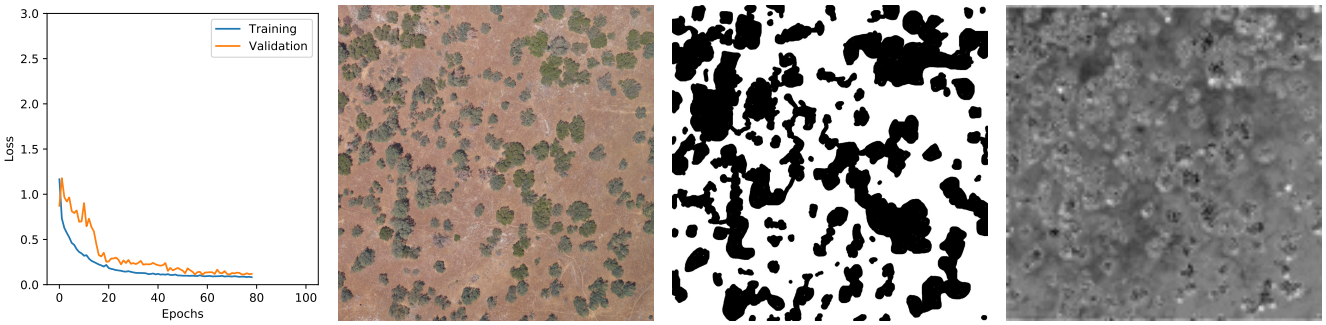
VI. Conclusion

This work proposed TFCN, a new fully convolutional neural network model to compute traversabilities from aerial images with the purpose of assisting ground robot navigation. The



(a) Training and validation losses (b) Test image (c) Test label (d) TFCN output

Fig. 9. Leave-one-out step 7.



(a) Training and validation losses (b) Test image (c) Test label (d) TFCN output

Fig. 10. Leave-one-out step 8.

TFCN was trained on the Aerial Traversability and Planning Dataset and cross-validated using an image-based leave-one-out approach. The results showed that the TFCN outperformed its competitors with regard to quality of traversability outputs, as measured with MSE against ground-truths, and processing time. However, it was shown that the current implementation of the TFCN has a generalization issue when it comes to terrain type. Further research is needed to address this issue.

REFERENCES

[1] H. Seraji, "Traversability index: a new concept for planetary rovers," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 3, pp. 2006–2013 vol.3, 1999.

[2] A. Howard, E. Tunstel, D. Edwards, and A. Carlson, "Enhancing fuzzy robot navigation systems by mimicking human visual perception of natural terrain traversability," *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, vol. 1, pp. 7–12 vol.1, July 2001.

[3] A. Howard, H. Seraji, and B. Werger, "A terrain-based path planning method for mobile robots," *Seventh International Conference on Automation Technology, NASA Jet Propulsion Laboratory (2003)*, 2003.

[4] C. Castejón, B. L. Boada, D. Blanco, and L. Moreno, "Traversable region modeling for outdoor navigation," *Journal of Intelligent and Robotic Systems*, vol. 43, pp. 175–216, Aug 2005.

[5] Y. Guo, A. Song, Y. Cao, and H. Tang, "Research on navigation for search and rescue robot based on traversability," in *Intelligent Robotics and Applications* (C. Xiong, Y. Huang, Y. Xiong, and H. Liu, eds.), (Berlin, Heidelberg), pp. 853–862, Springer Berlin Heidelberg, 2008.

[6] M. Shneier, T. Chang, T. Hong, W. Shackleford, R. Bostelman, and J. S. Albus, "Learning traversability models for autonomous mobile vehicles," *Autonomous Robots*, vol. 24, pp. 69–86, Jan 2008.

[7] N. Vandapel, R. R. Donamukkala, and M. Hebert, "Unmanned ground vehicle navigation using aerial ladar data," *The International Journal of Robotics Research*, vol. 25, pp. 31–51, January 2006.

[8] R. Hudjakov and M. Tamre, "Ortophoto analysis for ugv long-range autonomous navigation," *Estonian Journal of Engineering*, vol. 17, pp. 17–27, 01 2011.

[9] R. Hudjakov and M. Tamre, "Orthophoto classification for UGV path planning using heterogeneous computing," *International Journal of Advanced Robotic Systems*, vol. 10, p. 268, jan 2013.

[10] J. Delmerico, E. Mueggler, J. Nitsh, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning," *IEEE Robotics and Automation Letters*, vol. 2, pp. 664–671, Abril 2017.

[11] C. D. B. Borges, A. M. A. Almeida, I. C. de Paula Junior, and J. J. de Mesquisa Sa Junior, "A strategy and evaluation method for ground global path planning based on aerial images," *Expert Systems With Applications*, vol. XX, pp. XXX–XXX, 2019.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 448–456, PMLR, 07–09 Jul 2015.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[14] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, (NYC, USA), pp. 78–, ACM, 2004.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

[16] L. Prechelt, *Early Stopping — But When? Neural Networks: Tricks of the Trade: Second Edition*, pp. 53–67. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.