Journal of the Astronautical Sciences manuscript No.
(will be inserted by the editor)

A Deep learning approach to Hazard detection for Autonomous Lunar landing

Rahul Moghe · Renato Zanetti

Received: date / Accepted: date

Abstract A deep learning approach is presented to detect safe landing locations using LIDAR scans of the Lunar surface. Semantic Segmentation is used to classify hazardous and safe locations from a LIDAR scan during the landing phase. Digital Elevation Maps from the Lunar Reconnaissance Orbiter mission are used to generate the training, validation, and testing dataset. The ground truth is generated using geometric techniques by evaluating the surface roughness, slope, and other hazard avoidance specifications. In order to train a robust model, artificially generated training data is augmented to the training dataset. A UNet-like neural network structure learns a lower dimensional representation of LIDAR scan to retain essential information regarding safety of the landing locations. A softmax activation layer at the bottom of the network ensures that the network outputs a probability of a safe landing spot. The network is also trained with a cost function that prioritizes the false safes to achieve a sub 1% false safes value. The results presented show the effectiveness of the technique for hazard detection. Future work on electing one landing spot based on proximity to the intended landing spot and the size of safety region around it is motivated.

Keywords Hazard Detection · Machine Learning · Autonomous Landing · Semantic Segmentation

1 Introduction

Autonomous landing is essential for safety and reliability of future space exploration missions. Hazard detection during landing enables automatic detection and navigation of hazards on the surface. The Hazard Detection System (HDS) is a primary component of the cross-NASA developed Autonomous Landing

R. Moghe · R. Zanetti
The University of Texas at Austin, Austin, TX 78712
E-mail: rahul.moghe@utexas.edu

and Hazard Avoidance Technology (ALHAT) sensor suite [1–4]. It provides guidance, navigation and control capabilities for autonomous landing under robust lighting conditions. It generates a Digital Elevation Map (DEM) using a LIDAR sensor which can then be processed to detect hazards on the landing area. In order to determine safe landing locations, the DEM is analyzed for candidate locations that satisfy mission specifications such as slope, terrain roughness, and proximity to hazards. Convolutional Neural Networks (CNN) are ideally suited to recognize desired patterns on spatially correlated input data such as images. The safety of a landing spot depends on the elevation of the surface in its vicinity. Moreover, complex mission specifications can be incorporated by aggregating them in the CNN training, thereby shifting the complexity to pre-flight operations and making the on-board inference computationally efficient for real time application. In this paper, we present a robust learning approach to detect hazardous and safe landing locations using CNNs. In particular, we use Semantic Segmentation to determine safe landing locations by analyzing the DEM of the Lunar surface.

Previous studies on autonomous landing have passive optical sensors like cameras as well as active sensors like LIDAR. LIDAR based methods have become popular because they are robust to different lighting conditions. The ALHAT sensor suite primarily uses LIDAR sensors. Research on the choice and construction of sensors to be used for autonomous landing has been studied in the literature [5, 6]. Much recent work at NASA focuses on the development of NASA's next generation HDS through the SPLICE project [7]. Obtaining real time and reliable DEMs using these sensors has also been the focus of previous works [8–10]. Various studies have concentrated on the Terrain Relative Navigation (TRN) aspect of autonomous landing, wherein sensor observations are compared to a surface map to obtain position estimates for navigation. This paper focuses on the Hazard Detection and Avoidance (HDA) phase of autonomous landing which typically takes place between $0.5 - 2$ km above the Lunar surface. In particular, given a sensor observation, candidate landing locations that satisfy safety and mission specifications are detected.

Various studies on hazard detection using active sensors have been conducted in the past [11–13]. The geometric footprint of the spacecraft and the slope and roughness of the terrain were evaluated to determine the safety of the landing location. Although our work uses similar landing specifications, we assign a discrete value to the safety of a landing location and use physical instead of synthetic data from a Lunar mission to evaluate our algorithm. It is important to note that more complexity in defining safety of a landing location can be added without any change to flight-software complexity. In all of the previous studies, adding more complexity results in increased computational burden of the algorithm to be used in real time. The approach presented in this paper is oblivious to the added complexity. This is because the added complexity makes the pre-processing more complicated but does not affect the training phase of our deep learning approach. To the best of our knowledge, our approach to move the computationally complex aspect of the HDA algorithm from onboard to offline is a novel contribution of this work.

In the past, machine learning methods have been used for crater identification on the Lunar surface [14–18]. CNNs were used to infer the center and the radius of the crater by analyzing the digital elevation map of the surface. The detected craters were then used for TRN, wherein a comparison with an existing map provides a position measurement used for navigation. In this paper, we instead train CNNs to find safe landing locations. A number of tests for LIDAR based navigation have been performed in the past. The ALHAT sensor suite was fully integrated with the Morpheus vehicle to test its TRN capabilities for precision landing [19]. Past studies have evaluated and improved passive optical sensor based TRN algorithms [20]. Flash LIDAR and laser altimeter based TRN was tested on a fixed-wing aircraft [3].

Semantic Segmentation is a technique used in computer vision wherein a CNN is trained to classify parts of the image into a set of predetermined classes [21]. This technique enables us to determine whether the spot on the surface belongs to the safe or the hazardous class. The CNN outputs a probability of a landing location being safe. A threshold value on the probability enables us to infer the class.

The DEM collected from the Lunar Reconnaissance Orbiter (LRO) mission is used for training, testing and validation. Even though there is an abundance of training data, all possible hazardous situations are not included in the training data due the high dimensionality of the system. Hence, noise is added to the training data to make the trained model robust to sensor noises. Additionally, a popular technique called data augmentation is used to add diversity to the training data [22]. In this technique, artificial training data is generated by applying randomly chosen transformations to existing training data.

This paper presents a deep learning approach to Hazard Detection and Avoidance for precision Lunar landing. The data preparation for creating the training, testing and validation data sets from the LRO mission is described in Section 2. This includes the mission specifications, ground truth data generation as well as the data augmentation techniques. The training framework presented in Section 3, describes the network topology, the loss function, the training methodology, and finally the metrics and techniques used for evaluation of the trained model. The results on the testing data set are given in Section 4. Finally, some concluding remarks and future directions are discussed in Section 5.

2 Data Preparation

2.1 Specifications

Table 1 contains the mission specifications considered in this paper and their respective acceptable limits used for training the CNN. The main parameters considered when deciding safety precautions are the slope, geometry of the spacecraft and its landing pads, and the size of the features on the surface. Computational complexity is also constrained since the DEMs have to be

Parameters	Requirements
Heading of the spacecraft	$< \pm 10^\circ$ from the vertical
Slope	$< 5^\circ$
Proximity to hazards	> footprint size and system uncertainties
Landing position error	$< 100m$
Surface feature size	$< 20cm$
DEM Processing time	$<$ 5 seconds

Table 1 Mission specifications for a safe landing spot on the Lunar surface.

processed and used for navigation. The intended or preplanned landing site must be within certain distance of the calculated landing site. In this paper however, we aim to find all the safe landing locations rather that the best one. Post-processing to obtain the best possible landing spot is a topic of future research.

2.2 Ground Truth

For this study, we use the DEM data collected during the LRO mission which started in 2009 [23]. A large DEM of the Lunar surface spanning between the equator and 15° north latitude and longitude between 30° and 60° is used for creating the training and validation set. The testing dataset is sample from a different region spanning between 15° north and 30° north latitude and 0° and 30° longitude. Smaller areas of size $(400, 400)$ pixels are then randomly sampled from this dense DEM. Each pixel represents the elevation of the surface at that position. The sampled DEM simulates a LIDAR scan generated during the HDA phase of the spacecraft descent phase. Although the LRO data was collected using an Altimeter and further processed, The ground truth LIDAR DEM can be simulated by adding noise. This is explained in the sequel. Existing formulas from the literature are then applied to calculate the slope of the terrain at a particular spatial position. In this paper, a third order difference method is used to calculate the partial derivatives in a global x and y direction [24]. This is given by

$$f_x = \frac{(z_{SE} - z_{SW} + \sqrt{(}2)(z_E - z_W) + z_{NE} - z_{NW})}{(4 + 2\sqrt{2})g} \tag{1}$$

$$f_y = \frac{(z_{NW} - z_{SW} + \sqrt{(}2)(z_N - z_S) + z_{NE} - z_{SE})}{(4 + 2\sqrt{2})g} \tag{2}$$

wherein, z_X denotes the elevation of the neighboring position on the map in the X direction where X can be E, W, SE, SW, NE, or NW. Here, g is the spatial resolution of the DEM which can be approximately calculated using the radius of the moon and dimensions of the scan. The slope of the surface at a spatial position is calculated according to

$$\theta = \tan^{-1}(f_x^2 + f_y^2) \tag{3}$$

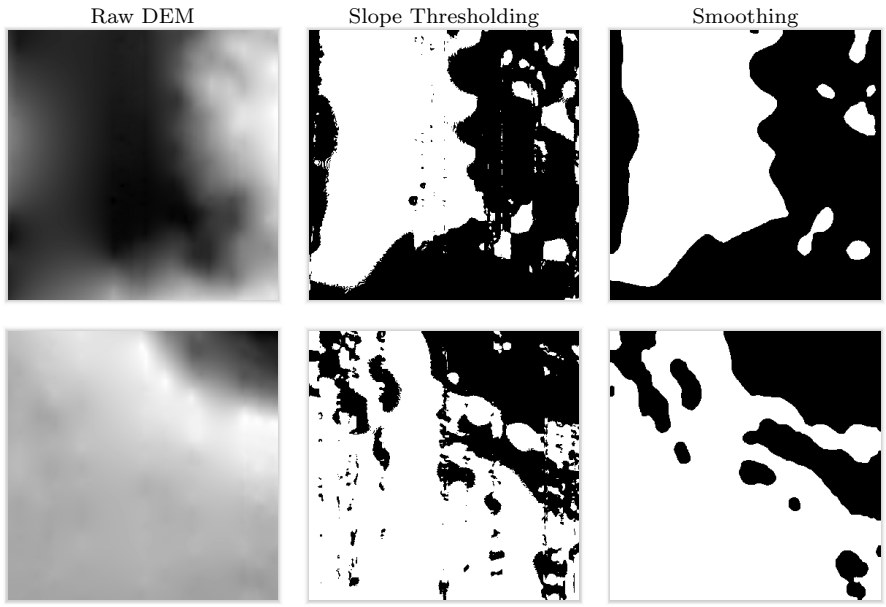

Fig. 1 Two examples of training data containing the raw DEM in the first column, the slope thresholded DEM in the second column and the final smoothed DEM based on proximity and minimum size of the hazards in the right column. The safe areas are in white and hazardous ones are in black. The first and third columns are used as input and output for training respectively.

The slope is thresholded according to the specifications and a binary map is generated with the flat and rough patches labeled as 1's and 0's respectively. An imbalanced or sparse binary map populated primarily with 0's or 1's can prove detrimental to training. This is because such maps make the denominator of the cost function zero. Hence, only the training data with $25 - 75\%$ flat areas is added to the training data set. The obtained map represents a high resolution slope map of the terrain. A median filter of appropriate size is then applied to remove small obstacles from the map. A minimum filter of appropriate size inflates the hazardous areas and ensures that the landing locations are a certain distance away from them. The input DEM is centered and normalized with the mean and standard deviation of its pixel values. Figure 1 shows 2 examples of the sampled DEM on the left and the binary map showing safe and hazardous areas on the right.

2.3 Sensor Noise and Data Augmentation

A data augmentation technique is applied to improve robustness properties of the trained network model. This method artificially increases the size and variety of the training dataset by transforming existing training data. Two

types of transformations are applied to the input data. First, robustness to noisy sensor data is improved by adding a zero mean white Gaussian noise and applying a Gaussian blur to the training data. The variance of the white Gaussian is a value below the limit of 0.5 and is randomly chosen at train time. Since, the input DEM is normalized to have a variance of 1, the value of 0.5 is half of the input variance. The second type applies geometric transformations to the inputs and outputs from the training dataset to form additional training data. Rotation by a multiple of 90°, vertical and horizontal flipping, and transposing operation are examples of this kind. Each of the transformations have a probability of 0.5 to be applied to the normalized training DEM. We use the fast image augmentation technique called Albumentations. Using data augmentation allows the CNN to be robust to errors in the DEM and allows the network to learn from a larger and richer dataset.

3 Learning Framework

3.1 Network Topology

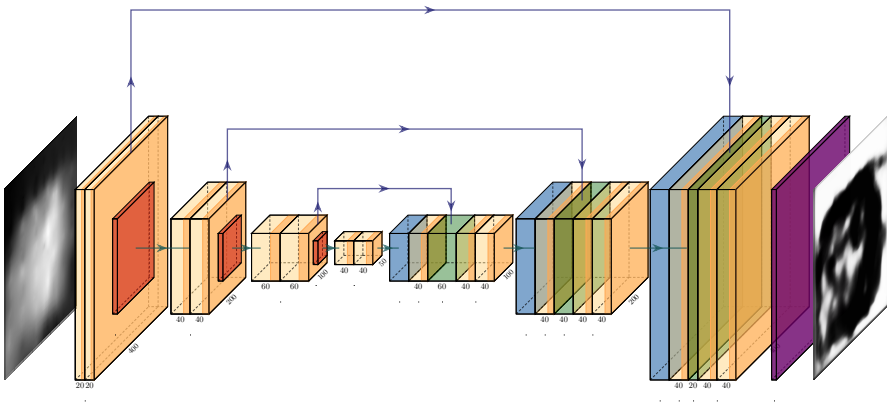

Fig. 2 The figure above describes the topology of the network. Yellow layers: fully connected convolutional layers with ReLU activation, Red layers: Max Pooling layers, Blue layers: Upsampling layers, Green layers with connections: Residual or skip connections, and Purple Layer: Sigmoid Layer.

The structure of the CNN trained in this paper is given in Figure 2. The UNet architecture used in this framework is adopted from the structure used for biomedical representation [25]. The idea behind using the UNet CNN architecture is to reduce the dimensionality of the input DEM and retain the essential data from the DEM using the top half of the network, and then projecting this essential information back onto the DEM in the bottom half of the UNet. The sampled DEM of size $(400, 400)$ and the binary map generated

Table 2 The table of layers sizes, their inputs, outputs and filter sizes.

	Name	Input Size	Output Size	Layer Width	Kernel Size
1	Conv-2D	400	400	20	6
2	Conv-2D	400	400	20	12
3	Max Pooling	400	200	20	2
4	Conv-2D	200	200	40	12
5	Conv-2D	200	200	40	12
6	Max Pooling	200	100	40	2
7	Conv-2D	100	100	60	12
8	Conv-2D	100	100	60	12
9	Max Pooling	100	50	60	2
10	Conv-2D	50	50	40	12
11	Conv-2D	50	50	40	12
12	Up Sampling	50	100	40	2
13	Conv-2D	100	100	40	6
14	Concatenate (13 + 8)	100	100	100	-
15	Conv-2D	100	100	40	12
16	Conv-2D	100	100	40	12
17	Up Sampling	100	200	40	2
18	Conv-2D	200	200	40	6
19	Concatenate (18 + 5)	200	200	80	-
20	Conv-2D	200	200	40	12
21	Conv-2D	200	200	40	12
22	Up Sampling	200	400	40	2
23	Conv-2D	400	400	40	6
24	Concatenate (23 + 2)	400	100	60	-
25	Conv-2D	400	400	40	12
26	Conv-2D	400	400	40	12
27	Sigmoid	400	400	1	1

above, are the input and the output to the CNN respectively. As mentioned before, the input DEM is first centered and normalized before training since only the relative values of the elevations affect the measure of safety of a landing spot. All the fully connected convolutional layers use the Rectified Linear Unit (ReLU) as the activation function [8]. Since, the output is to be constrained between zero and one, a Sigmoid activation function is applied to the last layer. A threshold is then applied to the output for binary classification. The network also includes residual or skip connections between lower and higher layers wherein a lower layer is concatenated as is to a higher layer of the network [26]. This facilitates free flow of gradients without passing through the nonlinear layers and helps us avoid the degradation problem in deep neural networks.

3.2 Loss Function

The loss function used for training is the Jaccard loss which is known to work well on unbalanced data where most of the image may belong to one class [27]. The Jaccard loss is evaluated as the Intersection over Union with a smoothing factor as given below.

Case	A	B	C	D	E	F	G
K	1	10	20	35	50	65	75

Table 3 The weights used in the weighted Jaccard loss function while training are shown above. The $K = 0$ case is the one where the original Jaccard loss function is used for training. The successive cases show training with increasing preference for lowering FS.

$$\mathcal{L}(Y_{pred}, Y_{true}) = s\Big(1 - \frac{sum(Y_{pred} \odot Y_{true}) + s}{sum(Y_{pred} + Y_{true}) - sum(Y_{pred} \odot Y_{true}) + s}\Big) \quad (4)$$

where the choice $s = 100$ is made to allow for data with unbalanced classes to train quickly. Here, $\odot$ represents the element wise product and the $sum(\cdot)$ represents the sum of all the elements of the matrix. In this paper, the prediction Y_{pred} is thresholded with a constant threshold. More advanced methods based on recent work on multi-threshold techniques may be pursued in future work [28]. This is a popular choice of loss function for evaluating semantic segmentation models.

The Jaccard loss function weighs the False Positives and False Negatives equally. For Lunar Landing applications however, having a low number of False Positives (i.e. hazardous sites classified as safe) is much more critical than having few False Negatives (i.e. safe sites classified as hazardous). In this paper, the network is also trained introducing a weight to the Jaccard loss function to more heavily penalize False Positives. For a binary output Y_{pred} (pixel values are either zero or one), the term in the numerator of equation (4) can be interpreted as the total number of True Safes (TS). The denominator, which is the intersection, can be broken down into the sum of True Safes, False Safes (FS) and False Hazardous (FH) as follows.

$$\begin{aligned} &sum(Y_{pred} + Y_{true}) - sum(Y_{pred} \odot Y_{true}) = \\ &= sum(Y_{pred} \odot Y_{true}) + sum(Y_{pred} \odot (1 - Y_{true})) + sum((1 - Y_{pred}) \odot Y_{true}) \\ &= TS + FS + FH \end{aligned}$$

Therefore, the Jaccard loss, given in terms of TS, FS, and FH is as follows.

$$\mathcal{L}(Y_{pred}, Y_{true}) = s\Big(1 - \frac{TS + s}{TS + FS + FH + s}\Big) \quad (5)$$

We define the weighted Jaccard loss function by prioritizing the FS as follows.

$$\mathcal{L}_{FS}(Y_{pred}, Y_{true}) = s\Big(1 - \frac{TS + s}{TS + K * FS + FH + s}\Big) \quad (6)$$

where K is the weight on the FS while training. The values used for training are given in the Table 3.

3.3 Training and Evaluation

The Adam optimizer [29] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ is used for training the network. The validation (200 DEMs) and testing (100 DEMs) datasets are created in a similar way to the training set and the network is trained for 4000 epochs. The weights that output a minimum validation loss are stored and used for testing. The NVIDIA GeForce GTX 2080 TI GPU that was used for training and testing took an entire day for the network to train and fine tune on the GPU. The CNN outputs a $(400, 400)$ image with pixel values between zero and one. ~~A plot of histogram of these pixel values for a test data shows the distribution of pixel values in the test DEM output.~~

3.4 Metrics

Four metrics commonly used in semantic segmentation applications are evaluated. These are i. Pixel Accuracy, ii. Mean Accuracy, iii. Mean Intersection over Union (IoU), and iv. Frequency weighted IoU and are explained in detail in the following four sub-sections.

- Pixel Accuracy: $\frac{\sum_{safe,haz} n_{aa}}{\sum_{safe,haz} t_a}$
- Mean Accuracy: $\frac{1}{n_{cl}} \sum_{safe,haz} \frac{n_{aa}}{t_a}$
- Mean Intersection over Union (IoU): $\frac{1}{n_{cl}} \sum_{safe,haz} \frac{n_{aa}}{(t_a + \sum_j n_{ba} - n_{aa})}$
- Frequency weighted IoU: $\frac{1}{(\sum_k t_k)} \sum_{safe,haz} \frac{t_a n_{aa}}{(t_a + \sum_j n_{ba} - n_{aa})}$

wherein, n_{ab} is the number of pixels of class a classified as class b by the network, $n_{cl} = 2$ is the number of classes, and $t_a = \sum_b n_{ab}$ is the total pixels belonging to class a. Here, a and b are either the safe or hazardous class. While calculating these quantities safe and hazardous as considered as two different classes. For example, for the safe (hazardous) class, a True Positive (TP) is the number of pixels belonging to the safe (hazardous) class classified as safe (hazardous) pixel by the algorithm. These are denoted by $n_{safe,safe}$ and $n_{haz,haz}$ respectively. Also, since these are the only two classes, the TP for the safe class is identical to the True Negative (TN) for the hazardous class. Let False Positives and False Negatives be denoted by FP and FN respectively. Given below are the intuition behind using each of the metrics.

3.4.1 Pixel Accuracy

Pixel accuracy is the total number of correct pixel classifications in the test dataset divided by the total number of pixels in the test dataset. Here, $N_{pixels} = 400 \times 400 \times 100 = 16,000,000$ is the total number of pixels in the 100 test samples each of size $(400, 400)$. The correctly classified pixels correspond to the safe landing locations classified as safe, as well as the hazardous locations classified as hazardous.

$$\text{Pixel Accuracy} : \frac{\sum_{safe,haz} TP}{N_{pixels}}$$

3.4.2 Mean Accuracy

Mean accuracy is the percentage of true positives averaged over the two classes, i.e. the average of i. Number of correctly identified safe landing locations over the total number of landing locations and ii. Number of correctly identified hazardous landing locations over the total number of hazardous locations. Note that $TP + FN$ is the total number of pixels in the ground truth belonging to each class.

$$\text{Mean Accuracy} : \frac{1}{2} \sum_{safe,haz} \frac{TP}{TP + FN}$$

Pixel accuracy tells us the total accuracy of the safe and hazardous locations combined. However, if there is a large difference between the total number of safe or hazardous pixels, the pixel accuracy maybe be misleading as it hides the inaccuracies of the less represented class (Ex. mostly safe of mostly hazardous DEM). To address this, the Mean accuracy metric was introduced which averages the accuracies for each class. Hence, if the hazardous pixels in a mostly safe DEM is incorrectly classified, the Mean accuracy will expose that.

3.4.3 Mean Intersection over Union

Mean intersection over union is the average, over the two classes, of the intersection of safe (hazardous) pixels in both the truth and prediction divided by the number of pixels that are either safe (hazardous) or are classified as safe (hazardous) by the algorithm. The denominator is also known as the union.

$$\text{Mean Intersection over Union} : \frac{1}{2} \sum_{safe,haz} \frac{TP}{TP + FN + FP}$$

The False Positives are not adequately represented in the Pixel and Mean accuracies. For example, the case in which all the safe pixels are classified as safe sets both the accuracies to be high indicating a good performance by the UNet. However, the fact that there may be a lot of hazardous pixels that are also classified as safe is ignored. This inaccuracy is exposed by Intersection over Union.

3.4.4 Frequency Weighted Intersection over Union

Frequency weighted intersection over union is the weighted average over all pixels of the intersection of safe (hazardous) pixels in both the truth and prediction divided by the number of pixels that are either safe (hazardous) or are classified as safe (hazardous) by the algorithm.

$$\text{Frequency Weighted Intersection / Union} : \frac{1}{N_{pixels}} \sum_{safe,haz} \frac{(TP + FN) * TP}{TP + FN + FP}$$

The Frequency weighted IoU, similar to its accuracy counterpart, is introduced to the handle the less represented classes. In summary, the accuracies handle the positive predictions and the IoU handle the missed predictions of the UNet.

3.4.5 Confusion Matrix

The confusion matrix is a common way to evaluate classification problems. Since semantic segmentation problems have a two dimensional output, the confusion matrix for pixel level prediction is calculated. The four entries of the confusion matrix are listed below. These values are then normalized with the total number true safe and hazardous pixels respectively.

- True Safe (TS): Number of safe pixels classified as safe
- False Safe (FS): Number of hazardous pixels classified as safe
- True Hazardous (TH): Number of hazardous pixels classified as hazardous
- False Hazardous (FH): Number of safe pixels classified as hazardous

3.5 Receiver Operating Characteristic Curve

The value used to threshold the CNN output is chosen using the Receiver Operating Characteristic (ROC) curve. A trained CNN classifies each pixel in the LIDAR image into two classes, namely, Safe and Hazardous. To evaluate the performance of the CNN for different thresholds, we plot the ROC curve [30]. The ROC curve is a two dimensional representation of the classifier performance. It plots the True Positive Rate (TPR) vs. the False Positive Rate (FPR). Since there are only two classes, the ROC curve can be plotted for either class. We choose to plot the curve for safe landing locations. That is to say, True Safe (TS) is a safe landing location classified as safe, True Hazardous (TH) an hazardous landing location classified as hazardous, False Safe (FS) is an hazardous landing location classified as safe, and finally False Hazardous (FH) a safe landing location classified as hazardous. The goal is to have few FS (type I errors), as those are the most dangerous for the mission.

The TPR, also known as Recall or Sensitivity in the literature, is defined using the TS and the FH for a given value of the threshold.

$$TPR = \frac{TS}{TS + FH} \tag{7}$$

Similarly, for a given threshold, the FPR, which is also known as inverse Recall, is related to the FS and TH as given by the following equation.

$$FPR = \frac{FS}{FS + TH} \tag{8}$$

While varying the threshold value between zero and one, the TPR and FPR values are calculated and plotted against each other. The area under the

ROC curve (AUROC) is calculated using the trapezoidal approximation. If each pixel in the image was classified randomly as safe or hazardous with a probability of 0.5, the ROC would have been a straight line between $(0, 0)$ and $(1, 1)$. In an ideal situation, when the prediction is perfect, the TPR should be equal to 1 for every nonzero value of FPR. In this case, the AUROC is equal to 1. The closer the AUROC is to being 1, the better the CNN is in terms of distinguishing between Safe and Hazardous pixels.

3.6 Post-processing

The metrics mentioned above are indifferent to class boundaries. The hazards are already inflated to account for the proximity to the hazards. Hence, accuracy of segmentation away from the class boundaries is critical for autonomous landing operations. If most of the misclassification occurs at the class boundaries, the performance of the CNN can be better evaluated by post processing the CNN output. The erosion of the CNN output with a square matrix of ones is calculated before generating the confusion matrix for each of the cases from Table 3. Hence, a pixel value is considered safe only if all of its neighboring pixels including itself are classified as safe. Metrics evaluated after such a post processing measure the performance of the CNN inside the class regions.

4 Results

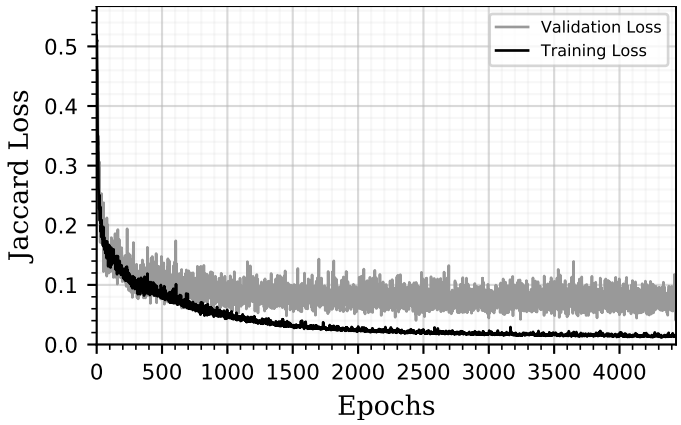

Fig. 3 The training and validation loss vs. epoch number for $K = 0$ case is shown above.

The CNN was trained while monitoring the training and validation loss simultaneously. The loss history on the training and validation set for the $K = 0$ case is given in the Figure 3. The weights that result in a minimum validation loss are stored and used for evaluating the performance of the CNN.

	Pixel Acc.	Mean Acc.	Mean IoU	Freq. weighted IoU
train	98.16%(±0.004)%	98%(±0.004)%	96.08%(±0.016)%	96.4%(±0.013%)
val	92.24%(±0.19)%	92.81%(±0.13)%	84.98%(±0.52)%	86.1%(±0.46%)
test	91.66%(±0.22)%	92.44%(±0.16)%	84.17%(±0.61)%	85.16%(±0.55%)

Table 4 Accuracy results on training, validation, and testing dataset for the $K = 0$ case evaluated using the metrics given in Section 3. A threshold of 0.6 was used to calculate the above metrics.

$K = 1$		Predicted	
		H	S
True	H	91.31	8.69
	S	8.01	91.99

Table 5 Normalized confusion matrices for pixel level prediction for safe and hazardous classes on the test dataset. The S and H columns or rows denote the Safe and Hazardous classes respectively. This table denotes the results for $K = 0$ case which is the network trained with unweighted Jaccard loss function.

As is seen from the figure, the minimum validation loss is slightly higher than its training loss counterpart. Stopping the training at this epoch is ideal to prevent overfitting of the data on the training dataset. Networks trained with different weights K, although have similar training times, are initialized with the weights trained for the $K = 0$ case so as to reduce their overall training time.

Once the CNN is trained, a few example outputs from the testing dataset are calculated and shown in Figure 4. The first column shows the raw DEM images from the sensor and the second column shows the ground truth safe and hazardous locations as calculated using techniques from Section 2. The CNN outputs a value between 0 and 1 for each pixel as shown in the third column. The output was then thresholded with a value of 0.6 to produce the final prediction in the forth column.

Since the CNN outputs a pixel value between 0 and 1, a histogram of the pixel values from the CNN output of a test input is shown in Figure 5. A majority of the pixels values are concentrated near zero and one. This means that the changing the threshold value barely changes the results unless the threshold value is close to 0 or 1.

The metrics given in Section 3.4 are evaluated on the training, validation and testing data and compiled in Table 4. The training dataset has the highest accuracy since the network is trained on it. The validation and testing dataset have similar accuracies which ensures that the CNN is not over or under fitted.

The confusion matrices for pixel level predictions for each of the seven K cases are calculated in Tables 5 and 6. The rows in the matrix are normalized since the accuracy is evaluated separately for each class. The average pixel values in the CNN output for TS, FS, TH, and FH are shown in Table 7 for each entry in the confusion matrix for $K = 0$ case. The histogram of the pixel values are consistent with the high FS and low FH values in this matrix respectively. One would imagine the average pixel value of the FS and the FH

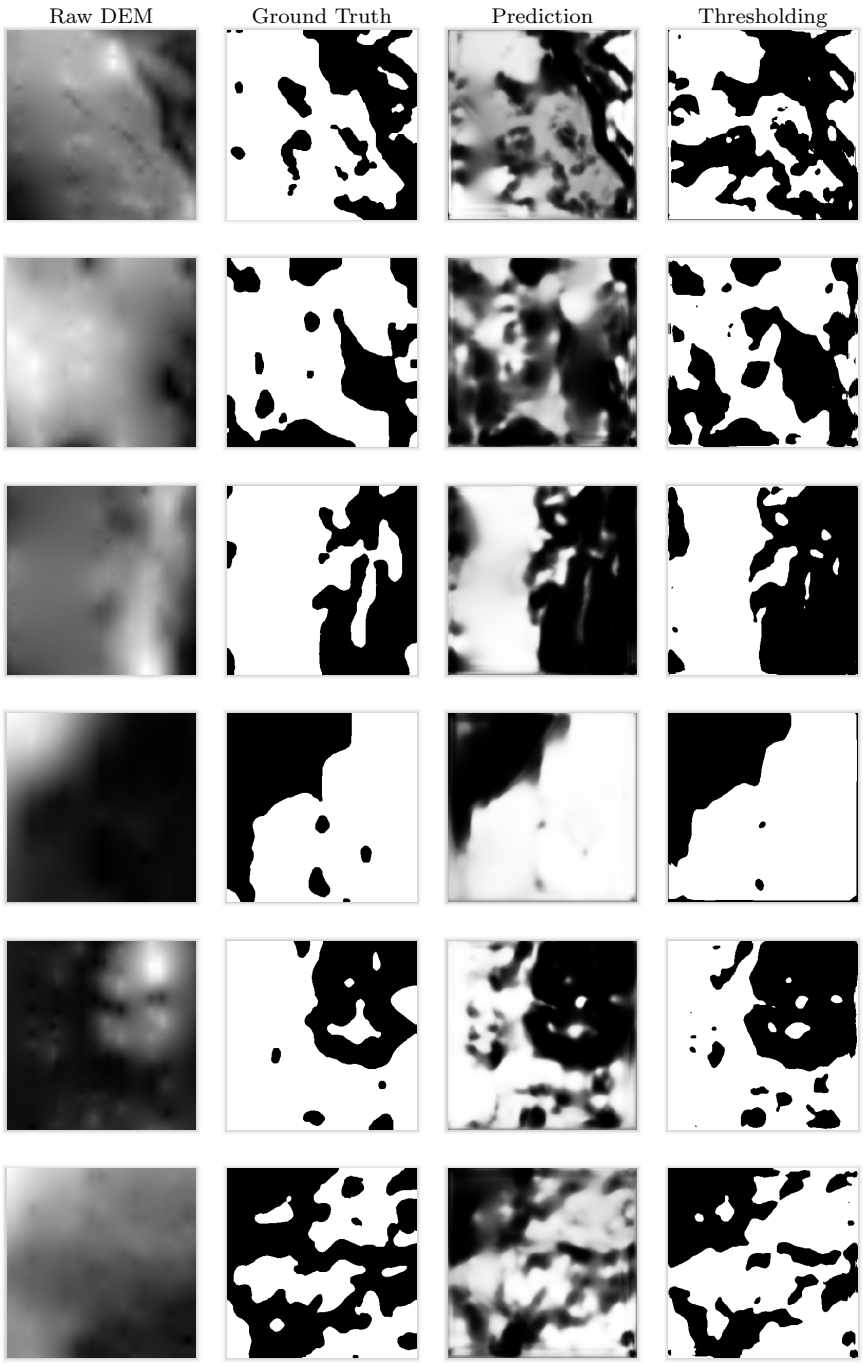

Fig. 4 The four columns denote the raw DEM, the processed ground truth, the CNN output and the thresholded CNN output. The six rows are example show six example input DEMs being processed. The prediction of the trained CNN is thresholded with a value of 0.6 to generate the final output.

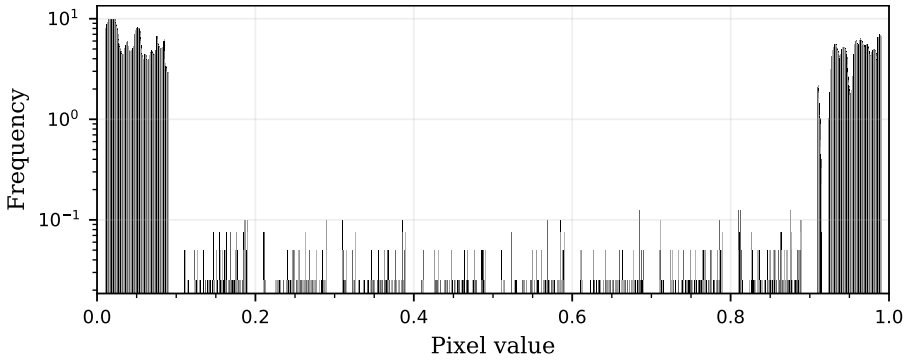

Fig. 5 The histogram of the CNN prediction normalized to form a density function is given above. A log scale is used for better visualization and the empty bins are removed.

$K = 10$		Predicted	
		H	S
True	H	94.42	5.58
	S	11.09	88.91

$K = 20$		Predicted	
		H	S
True	H	95.94	4.06
	S	13.72	86.28

$K = 35$		Predicted	
		H	S
True	H	97.13	2.86
	S	16.79	83.21

$K = 50$		Predicted	
		H	S
True	H	98.35	1.65
	S	22.25	77.75

$K = 65$		Predicted	
		H	S
True	H	99.01	0.99
	S	27.97	72.03

$K = 75$		Predicted	
		H	S
True	H	98.69	1.31
	S	25.30	74.70

Table 6 Normalized confusion matrices for pixel level prediction for safe and hazardous classes on the test dataset. Cases B through G corresponding to the entries in Table 3 are showcased here.

		Predicted	
		Haz.	Safe
True	Haz.	0.0014	0.9852
	Safe	0.0354	0.9991

Table 7 Average probability for each entry in the confusion matrix on the test dataset for the $K = 0$ case.

would be close to the threshold. However, this is not the case since the CNN was trained on pixel values of 0 and 1.

The variation of the TPR and FPR as function of the threshold value is depicted in Figure 6. This curve shows the decreasing TPR and increasing FPR with increasing weight K respectively. The ROC curve for the trained CNN calculated, for all the weights, on the testing dataset is shown in Figure 7. The AUROC calculated from the ROC curve is found to be 0.926 for $K = 0$ case. As the weight K is increased, the AUROC approximately demonstrates a decreasing trend. This shows that the performance of the CNN as a whole is sacrificed in order to improve the False Safe value which is desirable and in accordance with the objectives of the algorithm.

In terms of the computational complexity, the prediction from a DEM on an average takes 67 ms per DEM on the GPU which is well within the computational specifications. For a test input DEM, the output of the CNN

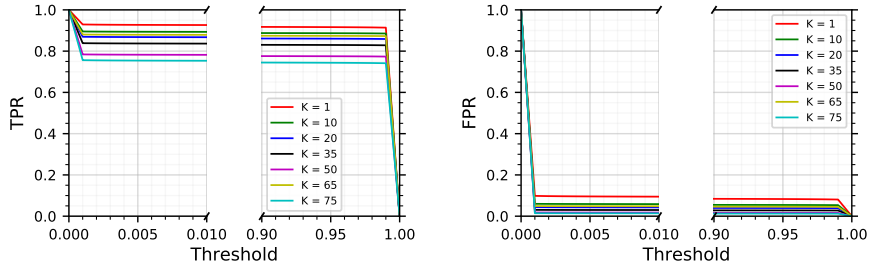

Fig. 6 Plot of the True Positive Rate vs. threshold on the left and the False Positive Rate vs. threshold on the right for the trained CNN for each of the cases from Table 3. The FPR plot on the right shows how the False Positive Rate is minimized by increasing the weight K on the False Safes. Consequently, the TPR is sacrificed as the weight is increased.

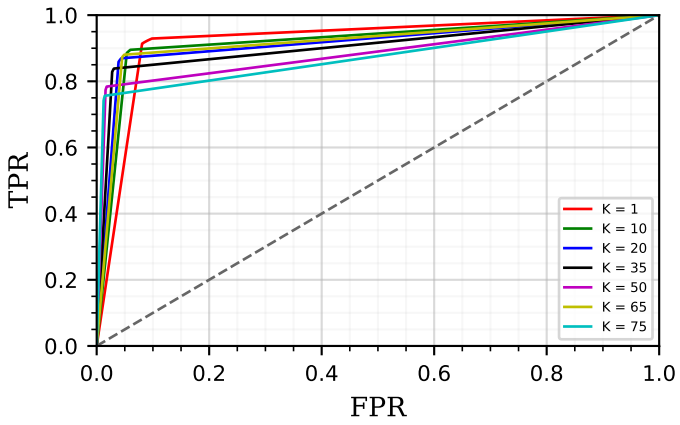

Fig. 7 Receiving Operator Characteristics (ROC) curve of TPR vs. FPR for varying thresholds is plotted for different weights in the weighted Jaccard loss function. The dashed line is the ROC curve for the arbitrary prediction. The Area under the ROC curve (AUROC) which is the area between the ROC, TPR = 0 line, and FPR = 1 line was found to be 0.926 for unweighted $K = 1$ case. The AUROC decreases with an increasing weight K on the False Safe.

$K = 1$		Predicted	
		H	S
True	H	92.98	7.02
	S	10.65	89.35

Table 8 The CNN predictions were post processed and a normalized confusion matrices for pixel level prediction for both the classes on the test dataset. This confusion matrix measures the performance of the CNN within the class regions and away from the class boundaries for the $K = 1$ unweighted case.

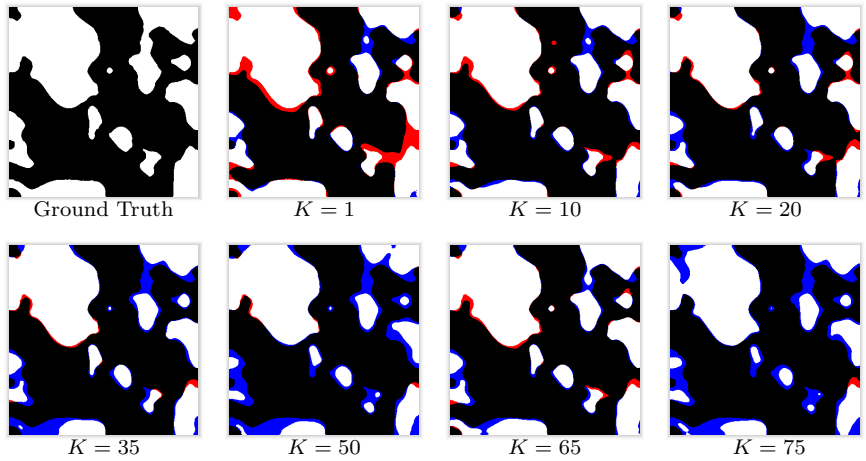

Fig. 8 The above figure shows the ground truth of a test DEM along with the output its output when passed through the seven CNNs trained using different weights in the loss function. The black and the white colored pixels are the True Hazardous and True Safe pixels. The red and the blue colored pixels are False Safe and False Hazardous pixels. As the weight K is increased, the False Safe pixels decrease at the cost of increasing number of False Hazardous pixels as expected. Note that misclassification occurs primarily at the class boundaries.

$K = 10$		Predicted	
		H	S
True	H	95.59	4.41
	S	14.20	85.80

$K = 20$		Predicted	
		H	S
True	H	96.83	3.17
	S	16.98	83.02

$K = 35$		Predicted	
		H	S
True	H	97.78	2.22
	S	20.22	79.78

$K = 50$		Predicted	
		H	S
True	H	98.76	1.24
	S	25.79	74.21

$K = 65$		Predicted	
		H	S
True	H	99.27	0.73
	S	31.48	68.52

$K = 75$		Predicted	
		H	S
True	H	99.03	0.97
	S	28.76	71.24

Table 9 The CNN predictions were post processed and a normalized confusion matrices for pixel level prediction for both the classes on the test dataset. This confusion matrix measures the performance of the CNN within the class regions and away from the class boundaries for the $K = 10$ to $K = 75$ case. The False Safes and False Hazardous rates are re weighted to improve the False Safe rate.

prediction is displayed for the networks trained with different loss functions in Figure 8. The red and blue colored pixels denote the False Safe and False Hazardous pixels respectively in the CNN prediction for the seven K cases. The variation of False Safes with increasing weight is shown in Figure 9. As mentioned previously, misclassification occurs at the class boundaries. In order to differentiate between the errors within and at the class boundary, the CNN output is post-processed by calculating the erosion of the output with a 3×3 square matrix of ones. The confusion matrix hence calculated for the unweighted and the weighted cases are given in Tables 8 and 9. The False Safe measures are lower than their unprocessed counterparts and hence shows a better regional prediction performance measure of the trained CNN.

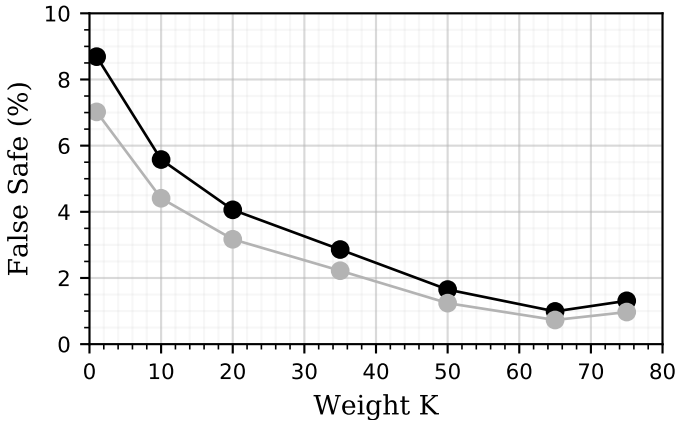

Fig. 9 The black and the grey lines correspond to the False Safe without and with post processing respectively. The False Safes decreases monotonically for the weights using which the CNN was trained. At $K = 75$, the False Safe begins to saturate at a steady state.

5 Summary

A deep learning algorithm called semantic segmentation popularized in computer vision applications is used for identifying safe landing locations on the Lunar surface. The digital elevation map (DEM) from the Lunar Reconnaissance Orbiter (LRO) is used to train a convolutional neural network (CNN) to find safe landing locations on the Lunar surface. Existing algorithms for calculating the slope and roughness are used on the DEM to find the ground truth locations which satisfy certain landing specifications. The training data is corrupted with simulated sensor noise and additional transformed data is augmented to the training data to make the model robust. A UNet-like network architecture is used to train on the augmented training data set. A weighted Jaccard loss function is used while training with different weights on the False Safes. After testing the trained CNN on the randomly sample DEMs from the Lunar surface, it was found that the CNN outputs a mean pixel accuracy accuracy of around 92% on the testing dataset. Incrementing the weight on the False Safes can get the False Safe percentage down to less that 1%. It is noted that misclassifications occur at class boundaries. Other common metrics for evaluating semantic segmentation models are also compared and reported. Future work includes choosing the best safe landing spot; it is strongly recommended that the choice choice of landing location is performed sufficiently far from a safe/hazardous class boundary. Such a choice will significantly decrease the chances of selecting a hazardous landing location.

Conflict of interest

This work is partially supported by NASA Johnson Space Center Grant NNX17AI35A.

References

1. Epp, C., Robertson, E., Carson, J.M.: Developing autonomous precision landing and hazard avoidance technology from concepts through terrestrially flight-tested prototypes. In: AIAA Guidance, Navigation, and Control Conference, p. 0324 (2015)
2. Brady, T., Schwartz, J.: Alhat system architecture and operational concept. In: 2007 IEEE Aerospace Conference, pp. 1–13. IEEE (2007)
3. Johnson, A.E., Montgomery, J.F.: Overview of terrain relative navigation approaches for precise lunar landing. In: 2008 IEEE Aerospace Conference, pp. 1–10. IEEE (2008)
4. Amzajerdian, F., Pierrottet, D., Petway, L., Vanek, M.: Development of lidar sensor systems for autonomous safe landing on planetary bodies. In: International Conference on Space OpticsICSO 2010, vol. 10565, p. 105650M. International Society for Optics and Photonics (2017)
5. Amzajerdian, F., Vanek, M., Petway, L., Pierrottet, D., Busch, G., Bulyshev, A.: Utilization of 3d imaging flash lidar technology for autonomous safe landing on planetary bodies. In: Quantum Sensing and Nanophotonic Devices VII, vol. 7608, p. 760828. International Society for Optics and Photonics (2010)
6. Amzajerdian, F., Pierrottet, D., Petway, L.B., Hines, G.D., Roback, V.E., Reisse, R.A.: Lidar sensors for autonomous landing and hazard avoidance. In: AIAA SPACE 2013 Conference and Exposition, p. 5312 (2013)
7. Restrepo, C.I., Sostaric, R.R.: Next-generation nasa hazard detection system development. In: AIAA Scitech 2020 Forum, p. 0368 (2020)
8. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814 (2010)
9. Bulyshev, A., Pierrottet, D., Amzajerdian, F., Busch, G., Vanek, M., Reisse, R.: Processing of three-dimensional flash lidar terrain images generating from an airborne platform. In: Three-Dimensional Imaging, Visualization, and Display 2009, vol. 7329, p. 73290I. International Society for Optics and Photonics (2009)
10. Yan, B., Wang, Y., Feng, L., Zhou, H., Jiang, Z.: Terrain matching based on adaptive digital elevation map. In: 2018 International Conference on Advanced Control, Automation and Artificial Intelligence (ACAAI 2018). Atlantis Press (2018)
11. Ivanov, T., Huertas, A., Carson, J.M.: Probabilistic hazard detection for autonomous safe landing. In: AIAA Guidance, Navigation, and Control (GNC) Conference, p. 5019 (2013)
12. Johnson, A.E., Huertas, A., Werner, R.A., Montgomery, J.F.: Analysis of on-board hazard detection and avoidance for safe lunar landing. In: 2008 IEEE Aerospace Conference, pp. 1–9 (2008). DOI 10.1109/AERO.2008.4526301
13. Jiang, X., Li, S., Tao, T.: Innovative hazard detection and avoidance guidance for safe lunar landing. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **230**(11), 2086–2103 (2016). DOI 10.1177/0954410015625671
14. Emami, E., Bebis, G., Nefian, A., Fong, T.: Automatic crater detection using convex grouping and convolutional neural networks. In: International Symposium on Visual Computing, pp. 213–224. Springer (2015)
15. Cohen, J.P., Lo, H.Z., Lu, T., Ding, W.: Crater detection via convolutional neural networks. arXiv preprint arXiv:1601.00978 (2016)
16. Di, K., Li, W., Yue, Z., Sun, Y., Liu, Y.: A machine learning approach to crater detection from topographic data. Advances in Space Research **54**(11), 2419–2429 (2014)
17. Wang, Y., Wu, B.: Active machine learning approach for crater detection from planetary imagery and digital elevation models. IEEE Transactions on Geoscience and Remote Sensing pp. 1–13 (2019). DOI 10.1109/TGRS.2019.2902198
18. Silburt, A., Ali-Dib, M., Zhu, C., Jackson, A., Valencia, D., Kissin, Y., Tamayo, D., Menou, K.: Lunar crater identification via deep learning. Icarus **317**, 27–38 (2019)
19. Trawny, N., Huertas, A., Luna, M.E., Villalpando, C.Y., Martin, K., Carson, J.M., Johnson, A.E., Restrepo, C., Roback, V.E.: Flight testing a real-time hazard detection system for safe lunar landing on the rocket-powered morpheus vehicle. In: AIAA Guidance, Navigation, and Control Conference, p. 0326 (2015)

20. Cheng, Y., Clouse, D., Johnson, A., Owen, W., Vaughan, A.: Evaluation and improvement of passive optical terrain relative navigation algorithms for pinpoint landing. Spaceflight Mechanics **140** (2011)
21. Thoma, M.: A survey of semantic segmentation. arXiv preprint arXiv:1602.06541 (2016)
22. Buslaev, A., Parinov, A., Khvedchenya, E., Iglovikov, V.I., Kalinin, A.A.: Albumentations: fast and flexible image augmentations. ArXiv e-prints (2018)
23. Riris, H., Sun, X., Cavanaugh, J.F., Ramos-Izquierdo, L., Liiva, P., Jackson, G.B., Schmidt, S., McGarry, J., Smith, D.E.: The lunar orbiter laser altimeter (lola) on nasa's lunar reconnaissance orbiter (lro) mission. In: Conference on lasers and electro-optics, p. CMQ1. Optical Society of America (2008)
24. Zhou, Q., Liu, X.: Error analysis on grid-based slope and aspect algorithms. Photogrammetric Engineering & Remote Sensing **70**(8), 957–962 (2004)
25. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, pp. 234–241. Springer (2015)
26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. eprint. arXiv preprint arXiv:0706.1234 (2015)
27. Jaccard, P.: The distribution of the flora in the alpine zone. 1. New phytologist **11**(2), 37–50 (1912)
28. Gurung, A., Tamang, S.L.: Image segmentation using multi-threshold technique by histogram sampling. arXiv preprint arXiv:1909.05084 (2019)
29. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
30. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition **30**(7), 1145–1159 (1997)