Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Automated Crater Detection and Classification with Machine Learning

by

ZHAO, Sihang

Email: sihang.zhao21@imperial.ac.uk
GitHub username: acse-sz4521
Repository: https://github.com/ese-msc-2021/irp-acse-sz4521

Supervisors:

Prof. Gareth Collins
Dr. Marijan Beg

September 2022

Abstract

With the promising progress of deep neural networks, object detectors (ODs), like YOLO V5, can now rapidly process pictures and recognize the contained entities. In a vast scope of the scientific research field, like planetary science, ODs can be applied to various object counting or classifying tasks. In particular, crater detection algorithms (CDAs) have helped scientists process thousands or even millions of pictures and detect millions of craters. The number and sizes of craters contain rich information. However, like humans, CDAs also make mistakes and produce incorrect results. In real-world problems, we often fail to train a good model or effectively evaluate the quality of a model due to imperfect datasets. Iterative training, such as active learning, will cost high labor when the amount of data is huge, and there will also be problems of error accumulation. To solve these problems, this paper designs and implements the Metamorphic Crater Generator (MCG), an algorithm can generate very realistic images of craters with the required size and number. This work also proposes a CDA test method and a CDA training iteration strategy based on MCG, which realizes the automated testing and iterative training progress without manually checking. These methods improved metrics such as mAP_0.5 and recall of the YOLO V5 by about 4%, and even better for smaller craters (diam $<$ 10km).

1 Introduction

Deep neural network-based object detectors (ODs), like YOLO [1][2], can now rapidly process pictures and identify the entities using convolutional neural networks. ODs can be applied to various object counting or classifying tasks in many scientific research fields. In planetary science, crater detection algorithms (CDAs) have helped scientists detect thousands or even millions of craters through images, and real-time video processing[3][4][5]. Several existing reviews have presented a comprehensive overview of Machine Learning-Based CDAs [6]. The number and sizes of those craters contain rich information like an absolute age, or relative age of the geographic unit [7]. Moreover, craters' shape (morphology) and their ejecta blankets can also provide clues to subsurface properties [8]. Thus it is a significant problem for planetary scientists.

On the one hand, CDAs can automatically detect and classify the craters, with some features that humans cannot even detect. Computers will also never lose attention or feel tired, so their perceptual sensitivity to critical target events (in this case, craters) will not deteriorate [9]. On the other hand, with the development of aerospace technology and observation technology, we can observe more and more planets from more and more angles. So there are more craters in the pictures that need to be processed and identified. The scope of craters that need to be detected is not merely vast; it is increasing and nearly infinite for all practical purposes. The above fact means that manually identifying all craters is an impossible task for human beings; that is another reason why CDAs are not only important but also necessary.

The state-of-the-art (SOTA) model YOLO V5 as a CDA has been proved successful in Themis data from Mars [3][10]. However, like humans, CDAs also make mistakes and produce incorrect results. For instance, CDAs may miscount some craters which exist and also count some similar patterns as craters. These kinds of mistakes can mislead scientists to wrong conclusions, such as the ambiguous age of geological units and whether the great events occurred in a particular position of the planetary bodies. Though the known ODs are good enough to compete the CDA tasks, existing manually generated craters data sets like Robbins, [11] are not perfect, which causes difficulties for scientists in training or testing their models:

- During the training, when a CDA is detecting craters that are not annotated with the labels, the model will count them as “False positive,” so the loss function will have a negative influence back to the model and prevent parameters' improvement. This is the "False False Positive" (FFP) flaw.

- During the testing, from the other side, the unlabelled craters in the test set will lead to an incorrect evaluation of performance.

For training, manually checking whether a crater is mislabelled one by one after every iteration of CDAs training and using them as the training data in the next iteration is a strategy called active learning [12]. However, that will still cause a lot of labor. Especially when we explore the universe further with better technologies, there will be exponentially growing newly discovered craters on different planets. It is conceivable that the inadequacies of annotating craters with the naked eye will also appear in active learning. What's worse, the error may be accumulated from each iteration.

As for testing, several different techniques were designed for testing deep learning systems, such as convolutional neural networks (CNN) models and recurrent neural networks (RNN) models [13], and several techniques were applied to test domain-specific applications, such as Q&A systems [14] and auto-driving systems [15][16], both of them have similarity with CDAs due to the difficulty to construct a reliable test set to simulate real-world situations. However, there are no systematic methods like Metamorphic testing [17]designed specifically for testing CDAs.

This paper aims to fill these critical gaps by designing and implementing a **Metamorphic Crater Generator (MCG)**, based on MCG, I proposed a **CDA training-iteration strategy** and a **CDA testing method** for training and testing CDAs in imperfect datasets. In summary, this paper makes the following contributions:

- Designing and implementing a Metamorphic Crater Generator (MCG) that can crop craters from real images, generates fake craters, and fuse them into the original background under some conditions. MCG will automatically record the location and size of the generated craters and add them to the labels. Then, by comparing the different performances on different sizes of craters before and after processing MCG, we can get a valid estimate of the model's performance even when the test data is imperfect. We can observe our workflow effectively trigger the test models' miscounting errors, especially for small craters.
- This work also proposed a training-iteration strategy that uses craters generating techniques of MCG, to replace the patterns that are detected as craters by CDA but without labels. Then the processed images are used as the training set for the next training iterations. This method successfully avoids every hand-craft checking work and can make the training process of CDAs fully automatic.

2 Code metadata

All code in this work is written in Python. This work trained all models by YOLO V5 on Google Colab.

All of our computational solution and method implementation has been released on Github[18]. You can find the list of hyperparameters and install instructions in the README file and Jupyter notebook in that repository.

3 Methodology

My solution consists of the following three parts listed below with the description of whether they are original and how I make extensions on them :

- YOLO V5-based CDA: YOLO V5 is an existing model. The model works in Python $\geq 3.7.0$ environment, including PyTorch≥ 1.7. This work made fine-tuned hyperparameters and adjusted the label-writing and result-display part.

- The Metamorphic Crater Generator (MCG): All code is designed and implemented individually and originally.
- CDA training-iteration strategy and CDA testing method based on MCG: original standalone code that was designed and implemented individually and originally.

3.1 Descriptions for YOLO V5-based CDA and Data Formats

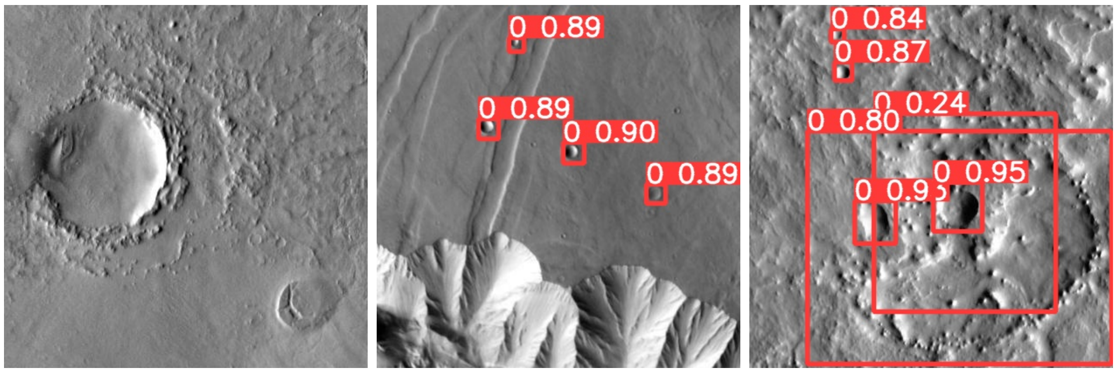

Figure 1: Example input of Thermal Emission Imaging System (THEMIS)[19] images and our output annotated by YOLO V5. The "0" in red labels means this object is detected as a "crater," and the following numbers represent our model's confidence in identifying this object. The red rectangles are aimed to cover the whole objects' bodies that have been detected.

Machine learning-based Automated Crater Detection and Classification tasks take images or real-time videos as input. The expected output is an annotated image with rectangles or circles surrounding every crater that appear in that image, including the large craters that exceed the edge of the image and small ones that can hardly be identified individually with the naked eye.

In this work, I implemented and extended the existing CDA based on YOLO V5. The first step to developing the CDA is training the Neural Network.

3.1.1 Input Imagery

The THEMIS images are images taken by The Thermal Emission Imaging System (THEMIS) [19], an instrument that combines a 5-wavelength visual imaging system with a 9-wave length infrared imaging system, which has been in a 2-hour orbit around Mars since 2001. The training set we use is the same images chosen by [3].

Benedix et al. selected the latitudinal band from $35°90N$ to $35°90S$ of the equator to avoid the image projection. In this work, I also use this area, which can represent about 60% surface on Mars. The images are first processed by global mosaic[20], partitioned into 6380 tiles, and were reduced into 889 tiles, each 416×416 pixels, by manually checking. The final training resource comprised the augmented images (rotated and reflected, vertically and horizontally), from 889 tiles to 3,556 tiles with 7048 labeled craters. The resolution is $100m/pixel$, and the size of labeled craters are around $1000m \sim 20,000m$.

Each of the images was labeled in the labels' txt files. The first column of digits represents the class of target objects. The second and third column of digits represents the relative location of the image. The fourth and fifth column of digits is the height and width of the rectangles that contain the craters. As there was only one class, which was craters, the first columns were all zero. The relative location of the images was the x and y coordinates of the craters' central location. With those labels, CDAs can locate and identify the craters while training.

3.1.2 Loss function

YOLOv5 uses the BECLogits loss function to calculate the loss of the objectness score, the class probability score is the cross-entropy loss function (BCEclsloss), and the loss function for the bounding box takes the GIoU Loss [21].

In crater detection tasks, we only have one class so I dropped the class probability score in this work.

$$GIoU = IoU - \frac{|C/(A+B)|}{|C|}, \tag{1}$$

The GIoU Loss is used to calculate the loss of the bounding box. For two arbitrary boxes, A and B, we find a minimum closed shape C so that C can include A and B, then calculate the ratio of the area of C that does not cover A and B to the total area of C, and then use A subtract this ratio from the IoU of B (in equation 1). Similar to IoU, GIoU can also be used as a distance, and loss can be expressed as:

$$L_{GIoU} = 1 - GIoU, \tag{2}$$

3.2 Metamorphic Crater Generator (MCG)

The Metamorphic Crater Generator (MCG) is made up of the following steps:

1. Crater extraction: Cropping craters from the original image, with their rectangular surroundings. The size of surroundings are hyperparameters. In our task, I set them as two pixels to ensure the craters remain the same shape.
2. Crater generation: Resizing the craters into the size we want and pasting them onto the original background using Poisson fusion [22]. In this process, we need to make circle masks for every crater we cropped. The size of the Mask depends on the size of resized craters and the shape of the mask is a circle contained by a square. While pasting our generated craters, I designed an algorithm (source code can be found in the repository) to ensure they will not overlap with the original real craters.
3. Generated-crater annotation: Recording the size and location of our generated craters, and writing them in the format in Table 1 into the txt files which will be our new labels for testing or training.

Algorithm 1 also gives an overall description of how MCG works. Additional details of the algorithm, including the sub-functions such as how to avoid overlapping are included in the source code in the repository.

3.2.1 Decision in Crater Extraction

In this work, I cropped craters from every single image and only pasted craters generated by the same image onto the background. This is because the crater images in our dataset come from pictures of different regions of Mars. When the latitude and longitude span are large, the difference in the angles which satellites shoot and the angles at which Mars receives sunlight, as well as the surrounding environment and geological features of craters, will change the shape of the crater image or the directions of their shadows. (e.g., the shadows of craters in the third image of Figure 1 are on the right-hand side, and the crater seems sunken. However, the shadows of the craters in Figure 2 are on the left part, and the craters seem to protrude. Cropping the craters from Figure 2 and pasting them into the background of the third image of Figure 1 apparently will cause mistakes.) This decision also helped a lot with the Poisson Fusion in the next section.

Algorithm 1 Metamorphic Crater Generator Algorithm

```
Input: a image $background$ in size 416*416, a List $labels$ containing all craters location and size
    information in $background$, resizing parameter $size$
Output: an image of generated craters added into original images $new\_background$, a txt file
    $new\_labels$ containing the labels for original craters and generated craters
 1: $skip \leftarrow False$
 2: $mine\_field \leftarrow []$                              ▷ List of labels for avoiding overlapping
 3: $new\_labels \leftarrow []$                              ▷ Initialisation
 4: for $i$ in $Labels$ do
 5:     $minefield$ append $i$                               ▷ Recording labels, avoiding overlapping
 6:     crop craters from $Background$ by $i$ as cropped crater $c$
 7:     save $c$ as images in $cropped\_image\_list$
 8: end for
 9: for $j$ in $cropped\_image\_list$ do
10:     resize $j$ as $size$
11:     $mask \leftarrow make\_mask(j)$                      ▷ Make mask for preparing for the Poisson fusion
12:     $loop\_count \leftarrow 0$
13:     while $overlapping == Falseandout\_of\_boundary == False)$ do
14:         $l \leftarrow random\_location, random\_size$
15:         $l \leftarrow (x, y) \leftarrow randomInt(416, 416)$
16:         if $count >= 100$ then
17:             $skip \leftarrow True$
18:             $break$
19:         end if
20:     end while
21:     if $skip == False$ then
22:         $poisson\_fusion(j, mask, background)$           ▷ generate craters on original background
23:         $minefield$ append $l$                           ▷ Recording labels, avoiding overlapping
24:         $new\_labels$ append $l$                         ▷ Write labels into txt files
25:     end if
26: end for
27: $new\_background \leftarrow background$                  ▷ Save Image
```

Some strategies and adjustable parameters are listed below:

- The Number of fake craters to generate: Users can set the number of craters he or she want to add to the background. The default number is the number of cropped craters, which is also the number of not overlapped craters smaller than 200×200 pixels and the length of the $cropped\_image\_list$ in the pseudo-code in Algorithm 1.
- The size of fake craters: Users can set the size of craters he or she wants to add to the background. One option is set a integer between 2×2 pixels (radius around 200 meters) and 200×200 pixels (radius around 20,000 meters). Another option is to set an integer P between 2 to 10. The size of the generated crater will be the original size of the cropped crater image divided by P. The purpose of setting it as an adjustable parameter is to allow the MCG to generate craters of different sizes to perform purposeful data augmentation or evaluation (e.g., testing the performance of our model with craters smaller than 1km).

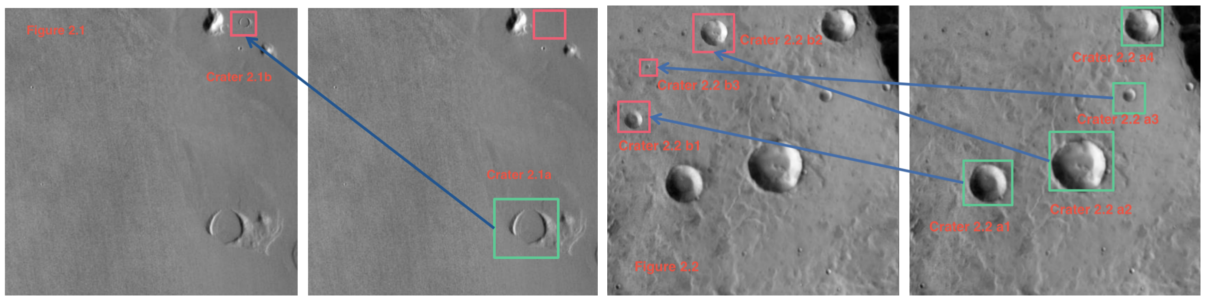

Figure 2: Example outputs of MCG. The craters on the left-hand side in red rectangles are craters generated by MCG by the craters on the right-hand side in the green rectangles. As you can see, the craters are cropped, resized, processed edges, and finally generated into the original background. MCG can also avoid the newly generated craters overlapping with the original craters or generating craters out of the boundaries. That is the reason why Crater 2.2a4 is not generated in the images on the right-hand side.

3.2.2 Tricks in Implementation of Poisson Fusion: Bigger is Better

Thank to the previous work of OpenCV [23], where the function of Seamless paste was perfectly implemented. In this work, we can cropped craters and make masks for each of them by our algorithms and then directly import the package to use the Seamless paste function to do the fusion. However, the result will not be satisfying if the mask is made by the size of cropped craters.

The trick of making generated craters looks like real craters is not making a "perfect" mask for every crater but making a little bigger one.

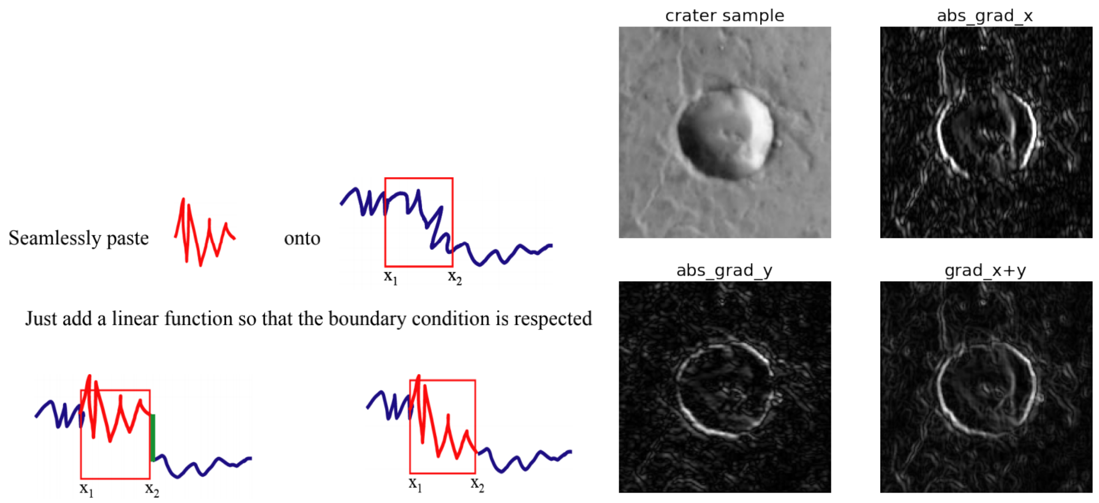

Figure 3: Examples of Seamless paste in 1-d cases by Chuanjie Zhu[24], and the gradient image of the sample crater. When we want to paste the red wave into the blue curve, we need the shape inside the red rectangle to be as same as the original red wave in overall, by the meantime, we also need the splice to be smooth. In our cases, we need to do the same thing for a 2-D image, from 2 directions x and y. The core idea of Poisson Fusion[22] is not to directly superimpose the two images but to let the target image (dst) "grow" a new one in the fusion part according to the guide field (actually the gradient field) of the source image (src). The overall principle is to keep the gradient field of the generated pixels in the fusion region as consistent as possible with the gradient fields of the pixels in the fusion part of the source image, and it is reflected in the equation solution that the gradient difference is as small as possible.

It is only necessary to provide the gradient field of the source image and let the target image generate the fusion part according to its own characteristics and the gradient field corresponding to the source image. Since the target image generates the fusion region according to its own characteristics, the fusion result will appear more natural.

I noticed that our craters are cropped from the same background as the target images, which means the surrounding areas (white areas in the masks shown in Figure 4) are supposed to have similar features and gradients with the targets images naturally. So I tried to make a looser mask for craters, then I got larger buffer areas for making a gradient interpolation. And the result shown in Figure 4 proved my assumption.

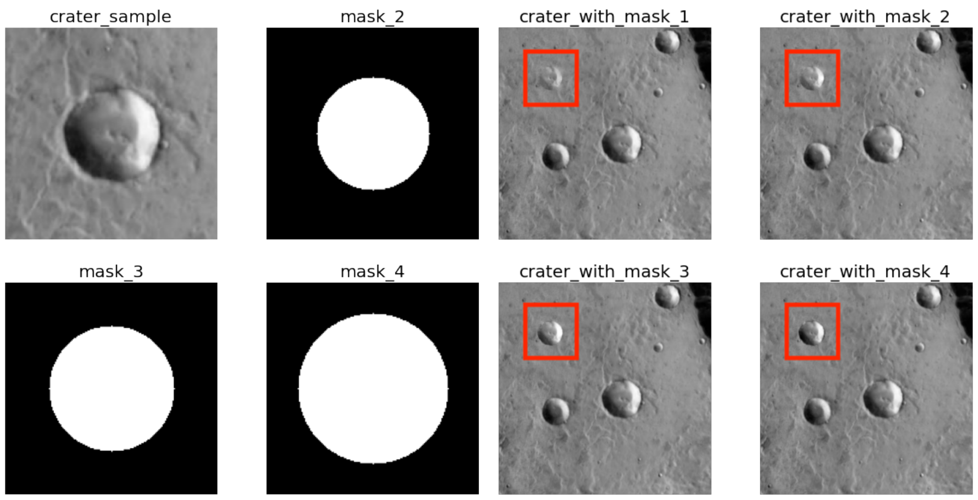

Figure 4: Examples in making masks in different sizes and their result.Obviously, the color of crater_with_mask_1 is too light, and its shape is also blurry, as the mask grows larger, the crater_with_mask_2 and crater_with_mask_3 are having better colors and clearer shapes, however, the shadows are not dark enough. After setting mask_4, the generated crater looks just perfect as a real one.

3.3 CDA testing method based on MCG

CDA testing method can evaluate a model even if the test set is imperfect. The flow chart explain how the CDA testing method works.

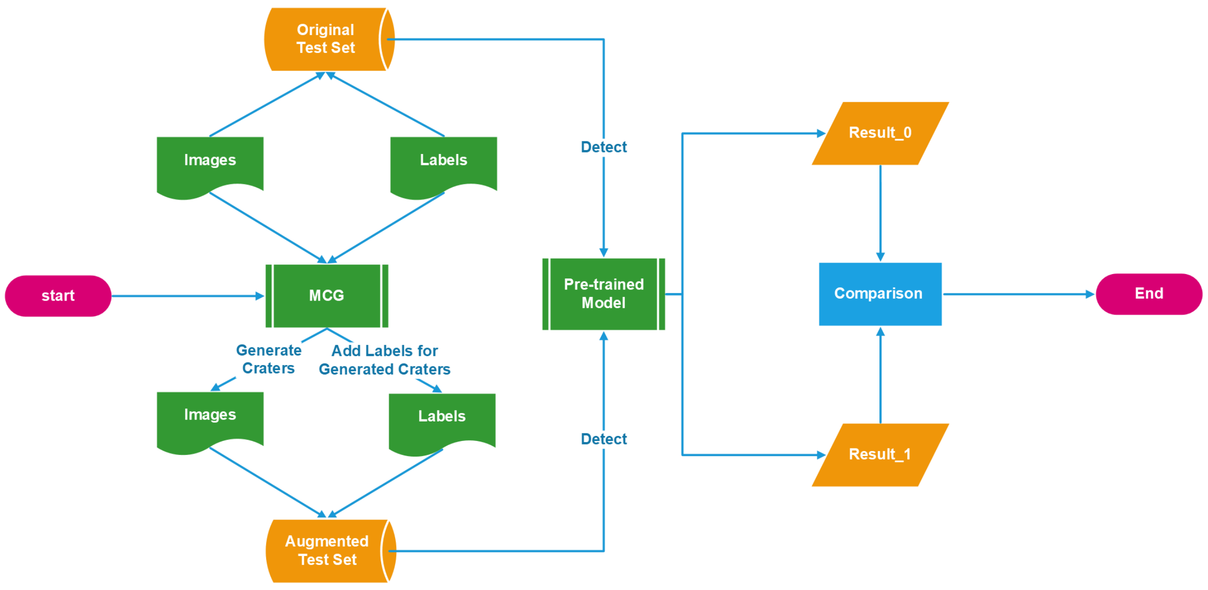

Figure 5: Work flow chart of CDA testing method

In this workflow, an augmented test set will be created by MCG. We can further use the generated craters and their labels as the augmented ground truth. This method brought two benefits:

- Even if the original test set is not perfect, maybe with many craters mislabeled, and there is no other test set for us to test the model's generalization ability, we can still evaluate the models by observing the difference in results (e.g., recall rate, mAP) between before and after processing MCG on the test set.

- Because MCG can generate craters of different sizes and locations, we can now adjust these parameters and try to trigger more mistakes so that we can know in what case our model will have the worse performance.

An example is given here: assuming we have a model m, and we get three different test sets by generating three different sizes of craters: $test_1$: craters are $5 \sim 10$ pixels in the image, $test_2$: craters are $10 \sim 20$ pixels in the image, and $test_3$: craters are $20 \sim 30$ pixels in the image. And we get three different results ($recall_1$ from $test_1$, mAP_2 from $test_2$ and etc.)

If $recall_1$ $<recall_2 = recall_3 =$ original recall, We can know that the model lacks the ability to detect craters that are smaller than 20 pixels.

3.4 CDA training-iteration strategy based on MCG

CDA training-iteration strategy aims to train a model effectively even when the training data is imperfect. And this strategy saved a lot of labor from active learning.

Because the result of MCG is more than satisfying, the newly generated craters on images can be treated as new ground truth together with their labels. We used these craters to overlap the "crater suspects". There are only two situations:

- The "crater suspect" is a real crater: using a newly generated crater to replace the suspect will avoid FFP. Which benefits our training.
- The "crater suspect" is not a crater: using a newly generated crater to replace this suspect can prevent the iterations from accumulating errors.

The flow chart in Figure 6 explains how the CDA training-iteration strategy works. And Algorithm 2 describes the algorithm of MCG_Replace_Edition.

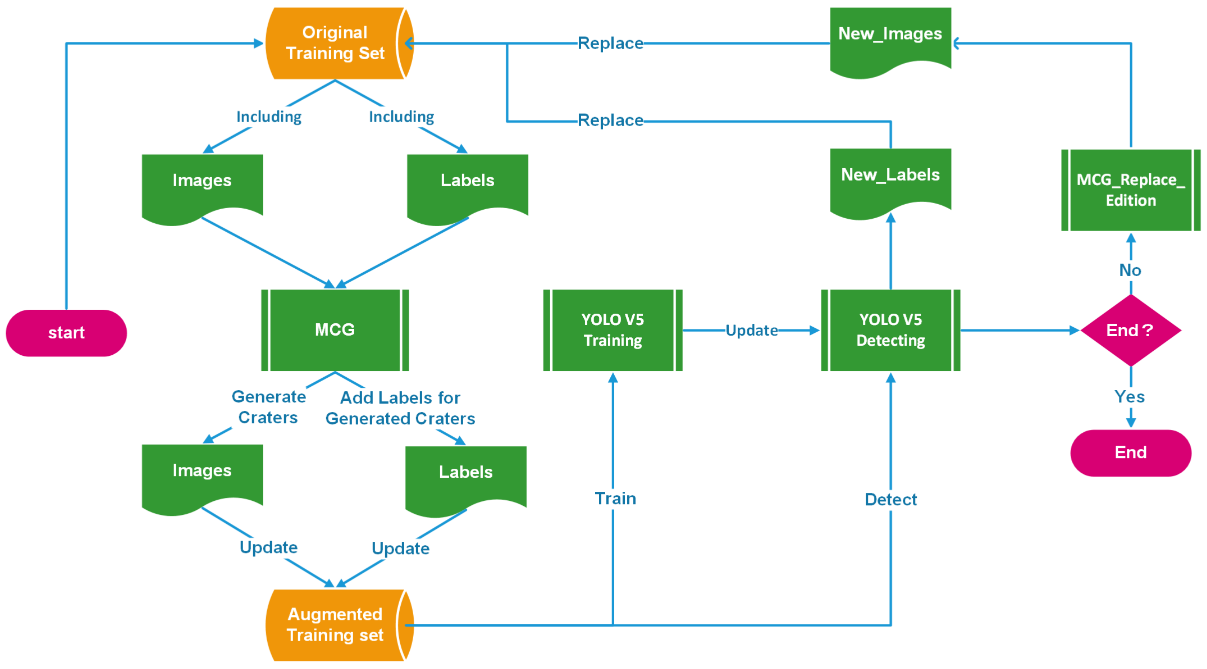

Figure 6: Work flow chart of CDA training-iteration strategy based on MCG. The key to this workflow is the MCG_Replace_Edition, which ensures we can have iterated training automatically.

There are 2 different applications of MCG in this strategy. The first one is the same algorithm as the MCG proposed in Algorithm 1. In this case, it works as a part of data augmentation. The second one in this workflow is **MCG_Replace_Edition**. Which is the key step in CDA Training-iteration strategy.

MCG_Replace_Edition first compares the original labels with the label reported by YOLO detection and replaces all craters that only exist in the detection labels with the craters generated by the MCG as the same size. This algorithm can skip the step of checking if a "crater" reported in models detection is indeed a crater or not by hand-craft or complicated algorithms. MCG_Replace_Edition will generate a metamorphic crater from other real craters (i.e. craters in original labels) and locate them in the same place with the same size as the reported crater. Because the generated craters are natural and treated as new ground truth, this method can successfully prevent error accumulation during iterations. When the result of the model on the validation set stops improving, the user can manually end this workflow.

Algorithm 2 Metamorphic Crater Generator Algorithm (Replace Edition)

```
Input: a image $background$ in size 416*416, a List $labels$ containing all craters location and
    size information in $background$, a List $Detect\_Labels$ containing all craters location and size
    information in $background$ that detect by CDAs,
Output: an image of generated craters added into original images $new\_background$
 1: $mine\_field \leftarrow []$                                  ▷ List of labels for judging overlapping
 2: $new\_labels \leftarrow []$                                                    ▷ Initialisation
 3: for $i$ in $Labels$ do
 4:     $minefield$ append $i$                               ▷ Recording labels, judging overlapping
 5:     crop craters from $Background$ by $i$ as cropped crater $c$
 6:     save $c$ as images in $cropped\_image\_list$
 7: end for
 8: for $j$ in $Detect\_labels$ do
 9:     $skip \leftarrow False$
10:     for $K$ in $Detect\_labels$ do
11:         if $J$ overlapped with K then
12:             $skip \leftarrow True$
13:             break
14:         end if
15:     end for
16:     if $skip == False$ then
17:         $child\_crater \leftarrow$ randomly from $cropped\_image\_list$
18:         resize $child\_crater$ as $j.size$
19:         $mask \leftarrow make\_mask(child\_crater)$   ▷ Make mask for preparing for the Poisson fusion
20:         $poisson\_fusion(j, mask, background)$ ▷ generate craters to replace the crater suspects
21:     end if
22: end for
23: $new\_background \leftarrow background$                                          ▷ Save Image
```

4 Experiments and Results

4.1 Evaluation of MCG

MCG in this work generated over 10000 craters in our given data set. The examples given by Figure 7 are selected randomly from the generated images.

I designed a questionnaire to evaluate how MCG works. In this questionnaire, I post seven images randomly selected in the images generated by MCG. I manually annotated the fake craters and real craters with red rectangles. In each question, there are only one real crater and one or two fake craters. Participants should identify the correct combination of real craters and fake craters. I use Descriptive statistics, and Analysis of variance (ANOVA) [25] to analyze and summarize our result.

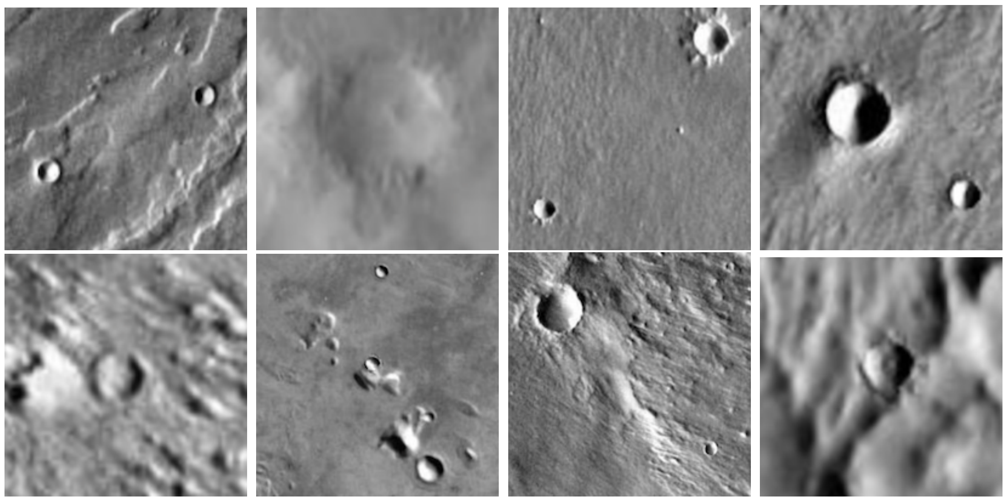

Figure 7: Craters generated by MCG

There are about 35 participants completed this questionnaire. I divided them into three groups by their self-positioning: "Experts", "Ordinary", and "Know nothing"

Answering correct answers with a high confidence level (i.e., Selecting the correct answer with "I am sure") will be scored 1 point. Answering a correct answer with a lower confidence level (i.e., Select the correct answer with "I guess") will be scored 0.5 points. All the other answers, including the wrong answer with a different confidence level or "I cannot tell," will be scored 0. The results are listed below:

- Unsurprisingly, **Nobody in all the participants** correctly answered all the questions (with confidence level "I am pretty sure" nor "It's a guess").
- There are 3 participants in Experts group, and they scored 0.33 on average. Only one of the three "experts" scored 1 point in the whole test. This means their performance is worse than people who totally randomly select options with their eyes closed.
- One-third of participants (12) claim themselves as "Not experts but saw relative images before" (Cyclops group). They scored 0.54 on average. The best score is given by a participant who correctly answered three questions with a high confidence level and missed the other three questions, finally scoring 3.
- 20 Participants who claim themselves as "I believe Mars orbits the Earth" (MOE group), their average score is 0.2. The best participants scored 2.

The P-value of F: $P = 0.746273 \gg 0.05$, which means the performance given by the Experts group has no significant difference from the Cyclops group or MOE group. None of them can distinguish real craters from fake craters. The results stand for craters generated by MCG are very natural and almost indistinguishable even for experts or scientists of planetary science.

The questionnaire and raw data of the answers can be found in the GitHub repository.

4.2 Evaluation of CDA testing method based on MCG

In this work, I divided a training set, validation set, and test set as the ratio of 6.3:2.7:1. There is no overlapping between these three data sets.

Naming the original training set as "Standard Training" (std_train), the initial validation set as "Standard Test" (std_test).

Because the result of MCG is more than satisfying, the newly generated craters can be treated as new ground truths together with their labels.

The original size-frequency distribution in our test set is shown in Figure 8. The red triangles represent the False Negative (FN) cases. Obviously, the FNs are more likely to be small craters.

We used MCG to process the std_train and std_test and got "Augmented Training" (aug_train) and "Augmented Test" (aug_test). The sizes of generated craters are around 10 pixels $\sim$ 20 pixels.

I trained two models on the std_train and aug_train, then tested them on std_test and aug_test, respectively. These are the observations:

- An improvement can be observed from aug_train on both std_test and aug_test, which is intuitive (This model is better at identifying more craters, in this case, on craters smaller than 1.5 km.)
- A decrease can be observed when we test our model trained by std_train on aug_test, the results of recall rate and precision are worse than the model test on std_test, which means our test method successfully generates the craters that cannot be detected by models. The performance can be found in table 1.
- The model trained on aug_train performed even better on the aug_test, which stands for the model also learned some features of new-generated craters.

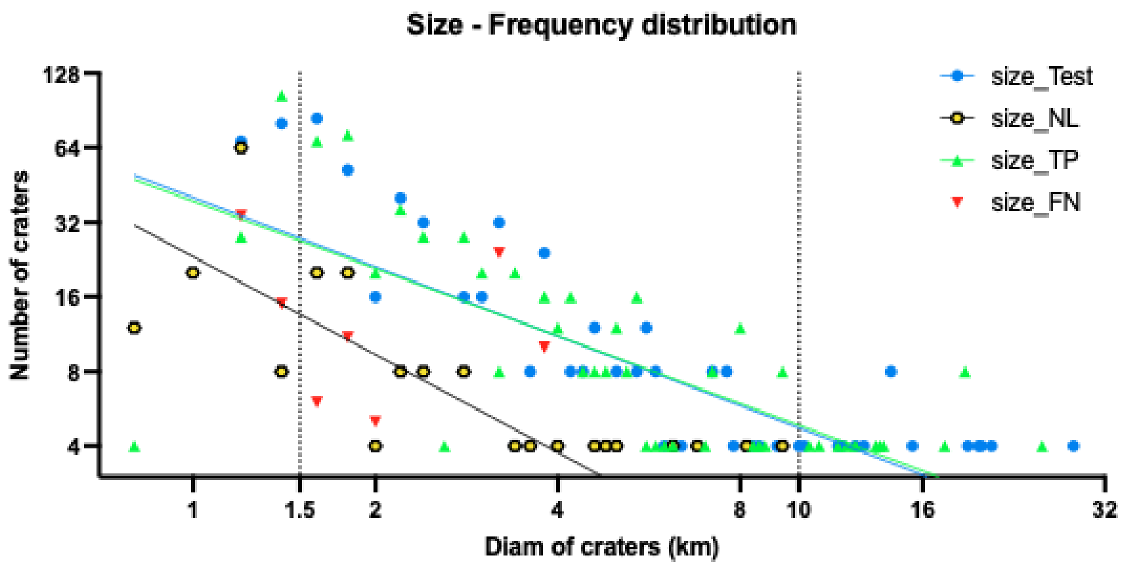

Figure 8: The Size-Frequency distribution. The points and triangles represent a group of craters of similar sizes. Data on the X axis were the mean diam of each crater group. The yellow points represent the craters that were detected but not labeled in our test sets. Because of the FFP cases, we cannot identify them as False Positive directly, so I name them Unlabelled (NL) here. The green triangles represent the true positive cases and the red triangles represent the false negative cases. This distribution is evaluated in standard train and standard test sets.

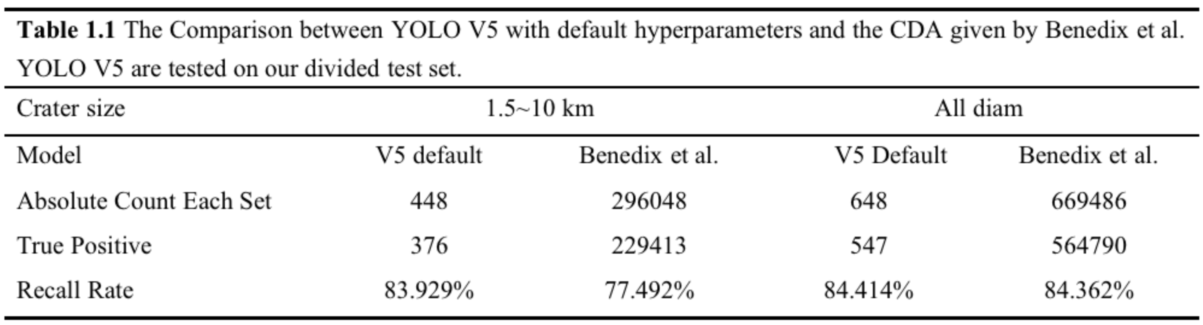

Table 1.1 The Comparison between YOLO V5 with default hyperparameters and the CDA given by Benedix et al. YOLO V5 are tested on our divided test set.

Crater size	1.5~10 km		All diam	
Model	V5 default	Benedix et al.	V5 Default	Benedix et al.
Absolute Count Each Set	448	296048	648	669486
True Positive	376	229413	547	564790
Recall Rate	83.929%	77.492%	84.414%	84.362%

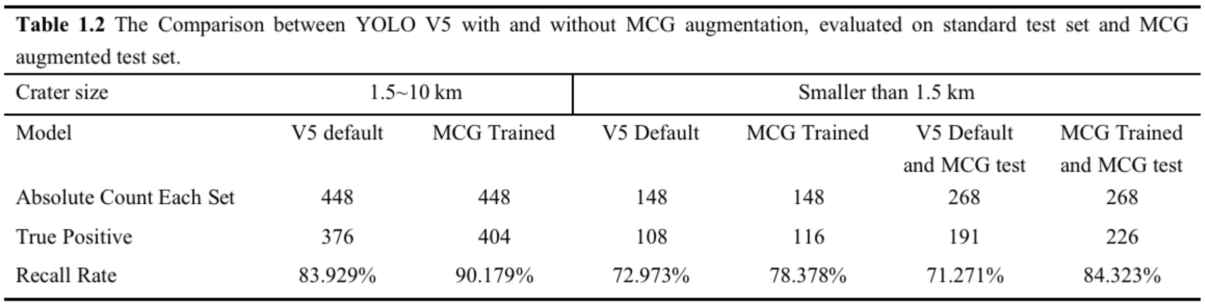

Table 1.2 The Comparison between YOLO V5 with and without MCG augmentation, evaluated on standard test set and MCG augmented test set.

Crater size	1.5~10 km		Smaller than 1.5 km			
Model	V5 default	MCG Trained	V5 Default	MCG Trained	V5 Default and MCG test	MCG Trained and MCG test
Absolute Count Each Set	448	448	148	148	268	268
True Positive	376	404	108	116	191	226
Recall Rate	83.929%	90.179%	72.973%	78.378%	71.271%	84.323%

Table 1: Table 1.1 compares the SOTA CDAs given by Benedix et al. and YOLO V5 with default hyperparameters. And Table 1.2 compares the YOLO V5 with and without MCG Augmentation in the same default hyperparameters. I used a test set that was completely separated from training sources for evaluating YOLO V5 and MCG augmentation so the number of absolute counts in MCD is much less than Benedix et al.

The original test set contains 52 craters that are larger than diam 10 km, 448 craters that in the range of diam 1.5 $\sim$ 10 km, and 148 craters that smaller than diam 1.5 km. The test set generated by MCG also contains 52 that are larger than the diam 10 km, but 636 craters in the range of diam 1.5 $\sim$ 10 km, and 1072 craters that are smaller than diam 1.5 km.

The result shows that the CDA test method based on CDA can effectively generate craters that cannot be detected by original models. In parallel to the test method, this result also shows that MCG is also an effective way to do data augmentations for CDAs. The comparison in Table 1 also shows that both YOLO V5 and MCG augmented YOLO V5 perform better than the model given by Benedix et al.

4.3 Evaluation of CDA training-iteration strategy based on MCG

After three iterations, model ci_train stops improving. And I test it together with std_train on the same default test set std_test. The result shows that the recall rate, precision, and mAp_0.5 has significantly improved.

The result shows that the recall rate, precision, and mAp_0.5 has significantly improved. There is no manual check during the workflows. When the task becomes more complicated and the number of craters grows, this model's superiority will manifest itself compared to active learning and other strategies.

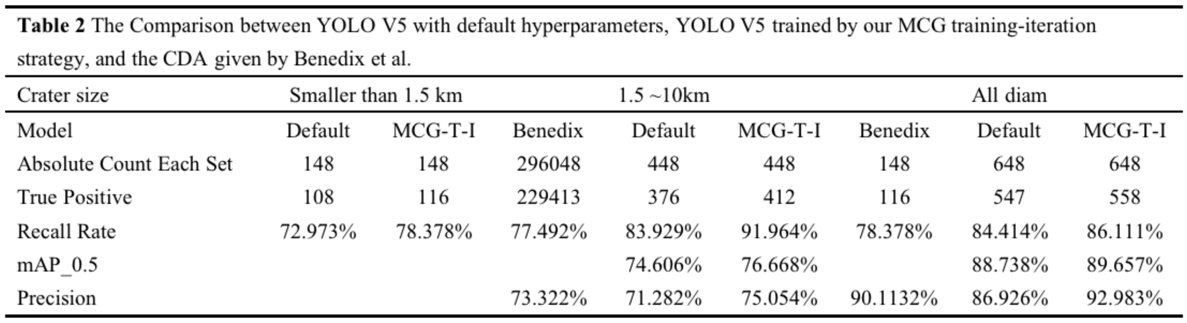
Table 2 The Comparison between YOLO V5 with default hyperparameters, YOLO V5 trained by our MCG training-iteration strategy, and the CDA given by Benedix et al.

Crater size	Smaller than 1.5 km			1.5 ~10km			All diam	
Model	Default	MCG-T-I	Benedix	Default	MCG-T-I	Benedix	Default	MCG-T-I
Absolute Count Each Set	148	148	296048	448	448	148	648	648
True Positive	108	116	229413	376	412	116	547	558
Recall Rate	72.973%	78.378%	77.492%	83.929%	91.964%	78.378%	84.414%	86.111%
mAP_0.5				74.606%	76.668%		88.738%	89.657%
Precision			73.322%	71.282%	75.054%	90.1132%	86.926%	92.983%

Table 2: Table 2 compares the SOTA CDAs given by Benedix et al. and YOLO V5 with default hyperparameters and YOLO V5 trained by the MCG Training-Iteration strategy. I used a test set that was completely separated from training sources to evaluate YOLO V5 and MCG T-I, so the number of absolute counts in MCD is less than Benedix et al.

5 Progress to Date and Future Plans

5.1 Conclusions Discussion

In summary, this work makes the following contributions:

- I designed and implemented a Metamorphic Crater Generator(MCG), which can generate natural-look craters images with customized sizes and locations. I designed a questionnaire and made survey statistics, which proved that the generated craters are almost indistinguishable from the naked eye.
- I designed a CDA test method based on MCG, which can test CDAs even with imperfect data sets. In the data set used in this work. The MCG can effectively generate craters that cannot be detected by original models and cause a significant gap between models for humans to observe.
- In parallel to the test method, I have proved that MCG can also be used as a data augmentation method. The recall rate increased by 3% and the mAP_0.5 increased by 2%
- I proposed and implemented a training-iteration strategy based on MCG, and designed and implemented an MCG_Repalce_Edition for this strategy. The training-iteration strategy can save labor from active learning and also prevent the accumulation of errors. The model trained under this strategy achieved 7% more recall rate on craters of 1.5 $\sim$ 10 km diam and 6% more recall rate on craters smaller than 1.5 km.
- The last but the most significant contribution is that MCG can not only help YOLO V5, but all CDAs based on machine learning. MCG and the strategies based on it are designed to be implemented into any data sets of craters. This work's initial purpose is to eliminate the problem caused by imperfect data.

During this work, the most challenging task was how to make the generated crater natural. There were many realistic problems, like how to smooth the edge and avoid overlapping. These problems always came into a mathematical problem. Luckily, most of them are solved.

The first strength of this work is mentioned before: it eliminated the problem caused by imperfect training data and test data – which is more suitable for real-life situations. We do not always have tabular data, the training set is terrible, and we are having problems evaluating our model in these situations.

The second strength of this work is that it is designed for every object detection model based on machine learning. So even if there are some newly published models or someone published a set of great hyper-parameters. We can still implement our method on their model.

However, this work and I also have some limitations:

- First of all, the time is relatively tight, and I cannot implement all my ideas. E.g., after I classify the craters by their size and evaluate the models' performances on those different groups. I noticed though the model trained by our Training-Iteration strategy performed great on craters smaller than 10 km, the recall of larger craters that bigger than 10 km diam is even lower than the default model. That may be because the model has learned too many generated small craters. Using MCG to generate some bigger craters randomly may solve this problem but I don't have time to finish proving this conjecture. Also, there may be some flaw in the algorithm looking for the FN and TP cases, because the total number of FN and TP always has a slight gap from the number of labeled craters in the test set.
- Secondly, in my training-iteration strategy, my method has a potential risk that prevents the model from learning new features compared with active learning. And if the model continually learns from the same craters. It may cause the over-fitting issue much sooner than active learning. An easy way to test if the model is cannot learning from the new feature is to test our models in a new test set, and analyze the F1 score and recall rate. We can also embed the result of training sources re-annotated by other models so that we can import more features into it.
- Thirdly, though the test set I used is totally separated from training sources, I cannot find the test sources used by Benedix et al., so the comparison between this work and their CDA can still have possibility to be unfair. However, the model they used is based on YOLO V3, I will not be surprised if our model performs better than it in the same test set. So I put all the data and source code of this work in the repository, for further comparison and evaluation.
- Actually the performance of aug_train_aug_test is too good: In the epoch that is given the best performance, the recall rate is reaching 90% for craters that are smaller than 1.5 km. I think it is because MCG augmented training has learned some features of MCG fusion. So it can accurately distinguish the craters generated by MCG.
- Lastly, MCG uses algorithms to avoid overlapping and out of boundaries. However, in real-world problems, one crater can be overlapped with others or partly out of the image. Optimization for this situation is also an expectable topic in future work.

5.2 Acknowledgments

I thank Mr. Guo, Ao, who provided the initial divided training and test data. My further split and manipulation was based on it.

I want to say thank you to my supervisors, Dr. Beg and Prof. Collins, who gave me enough help and suggestion during the project. Also, thanks to this short but cool project. It fulfilled my childhood dream and romantic imagination between stars. (not crater-liked, though)

I want to express my special appreciation to my friend Caifeng (Miss Zou, Caifeng), Bo (Mr. Bo, Pang), and my love Weiyu, who supported me and helped me overcome my hard moments, helping me face my ADHD bravely and inspiring me to taste the sweet parts of learning. They let me know what kind of person I want to be.

I also want to say thank you to myself. Whatever the outcome of this project, I deserve it.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," pp. 779–788, 2016.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] G. Benedix, A. Lagain, K. Chai, S. Meka, S. Anderson, C. Norman, P. Bland, J. Paxman, M. Towner, and T. Tan, "Deriving surface ages on mars using automated crater counting," *Earth and Space Science*, vol. 7, no. 3, p. e2019EA001005, 2020.

[4] C. Norman, J. Paxman, G. Benedix, T. Tan, P. Bland, and M. Towner, "Automated detection of craters in martian satellite imagery using convolutional neural networks," in *Planetary Science Informatics and Data Analytics Conference*, vol. 2082, no. 2081, 2018, p. 6004.

[5] C. Lee, "Automated crater detection on mars using deep learning," *Planetary and Space Science*, vol. 170, pp. 16–28, 2019.

[6] D. DeLatte, S. T. Crites, N. Guttenberg, and T. Yairi, "Automated crater detection algorithms from a machine learning perspective in the convolutional neural network era," *Advances in Space Research*, vol. 64, no. 8, pp. 1615–1628, 2019.

[7] C. A. T. W. Group *et al.*, "Standard techniques for presentation and analysis of crater size-frequency data," *Icarus*, vol. 37, no. 2, pp. 467–474, 1979.

[8] N. Warner, M. Golombek, J. Sweeney, R. Fergason, R. Kirk, and C. Schwartz, "Near surface stratigraphy and regolith production in southwestern elysium planitia, mars: implications for hesperian-amazonian terrains and the insight lander mission," *Space Science Reviews*, vol. 211, no. 1, pp. 147–190, 2017.

[9] J. E. See, S. R. Howe, J. S. Warm, and W. N. Dember, "Meta-analysis of the sensitivity decrement in vigilance." *Psychological bulletin*, vol. 117, no. 2, p. 230, 1995.

[10] A. Lagain, K. Servis, G. Benedix, C. Norman, S. Anderson, and P. Bland, "Model age derivation of large martian impact craters, using automatic crater counting methods," *Earth and Space Science*, vol. 8, no. 2, p. e2020EA001598, 2021.

[11] S. J. Robbins, I. Antonenko, M. R. Kirchoff, C. R. Chapman, C. I. Fassett, R. R. Herrick, K. Singer, M. Zanetti, C. Lehan, D. Huang *et al.*, "The variability of crater identification among expert and community crater analysts," *Icarus*, vol. 234, pp. 109–131, 2014.

[12] B. Settles, "Active learning literature survey," 2009.

[13] X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, and J. Zhao, "Deepstellar: Model-based quantitative analysis of stateful deep learning systems," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 477–487.

[14] S. Chen, S. Jin, and X. Xie, "Testing your question answering software via asking recursively," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 104–116.

[15] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 143–154.

[16] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[17] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–27, 2018.

[18] S. Zhao, "Automated crater detection and classification with machine learning," 2022. [Online]. Available: https://github.com/ese-msc-2021/irp-sz4521

[19] P. R. Christensen, B. M. Jakosky, H. H. Kieffer, M. C. Malin, H. Y. McSween, K. Nealson, G. L. Mehall, S. H. Silverman, S. Ferry, M. Caplinger *et al.*, "The thermal emission imaging system (themis) for the mars 2001 odyssey mission," *Space Science Reviews*, vol. 110, no. 1, pp. 85–130, 2004.

[20] C. Edwards, K. Nowicki, P. Christensen, J. Hill, N. Gorelick, and K. Murray, "Mosaicking of global planetary image datasets: 1. techniques and data processing for thermal emission imaging system (themis) multi-spectral data," *Journal of Geophysical Research: Planets*, vol. 116, no. E10, 2011.

[21] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[22] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *ACM SIGGRAPH 2003 Papers*, 2003, pp. 313–318.

[23] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[24] C. Zhu, "Image fusion - poisson blending," 2019. [Online]. Available: https://blog.csdn.net/u014485485/article/details/89481501

[25] G. E. Box, "Some theorems on quadratic forms applied in the study of analysis of variance problems, i. effect of inequality of variance in the one-way classification," *The annals of mathematical statistics*, pp. 290–302, 1954.