PyCDA: An Open-Source Library for Automated Crater Detection. M. R. Klear[1], [1]Launchpad.AI, 149 Natoma St., San Francisco, CA 94105; (michael.klear@colorado.edu).

Introduction: Automated crater detection has been the subject of research for decades, since before [1]. With the maturation of deep neural networks, notable advances in computer vision have been achieved [2]. More recent research in automated crater detection algorithms has shown promising results by applying neural networks to this problem [3].

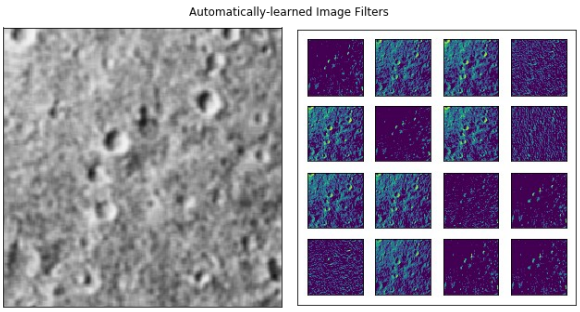

Figure 1: Convolutional Neural Networks learn to filter pixel values to produce useful features.

Much of this research claims to achieve near-human performance on crater detection tasks [4]. Such claims are difficult to support because these projects use human-generated annotations for testing. Variance in humans performing task is high [5], so reports of performance at this level should only claim to reproduce the work of a human annotator with some level of accuracy.

This research has the potential to greatly improve the scope of crater surveys. Deploying these models at scale, however, involves a number of practical challenges.

The Python Crater Detection Algorithm (PyCDA) package is an open-source crater detection library for conducting crater surveys with applied neural network models. Python presents an ideal open-source community for building and supporting this package.

Multiple Models: Originally released as a detection-only model, the alpha version of PyCDA (0.1.x) failed to perform on many datasets. This version features a single model that is trained on annotated images from the Mars Express mission, and differences between Mars and other solar system bodies are great enough to render this model useless outside of Mars.

Indeed, much of the research in this domain has pointed out that models tend to fail when applied to different terrains or bodies [3]. Researchers propose training or fine-tuning models for novel regions and bodies.

The utility of PyCDA in future releases, then, will be to enable users to retrain models using annotated images.

Public Model Repository: Training models is computationally expensive. The model featured in PyCDA 0.1.X, as an example, took several hours using an NVIDIA Tesla K80 GPU; this same task on a CPU computer would take days to weeks to complete.

PyCDA's intended users may not have access to such hardware, so training models represents a significant time investment for teams. This highlights the utility of creating a public repository for trained models; a team can try a set of pre-trained models to find one with an acceptable performance for the task at hand.

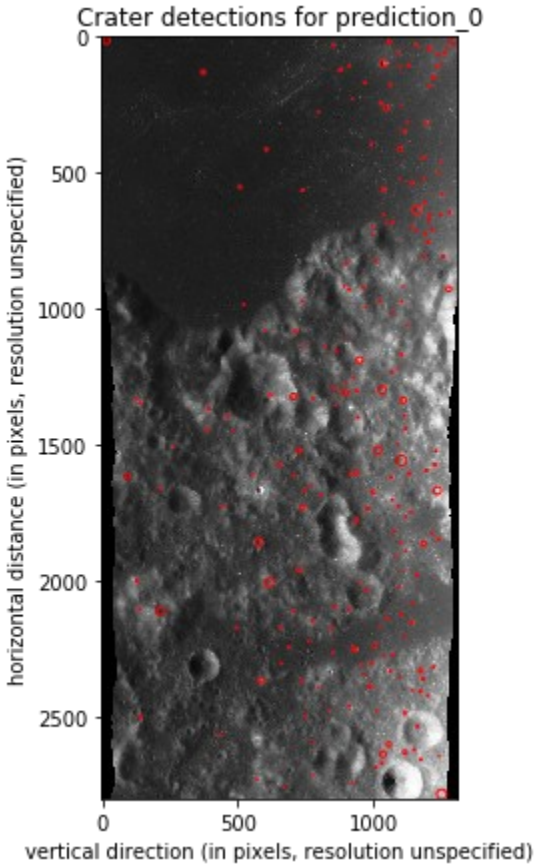

Figure 2: PyCDA 0.1.X fails to yield useful results on this lunar image

PyCDA 0.2.X API: PyCDA 0.2.X will provide three primary methods.

The first method is detect. When an appropriate model is selected, the detect method will return a list of detected craters given an input image.

An input image can optionally be assigned user-provided annotations. These are required to execute the remaining two primary methods.

The test method is useful in model selection. This calls the model to make detections and compares these with the user-provided annotations to give a measurement of model error. A precursor to this method is included in PyCDA 0.1.X (pictured).

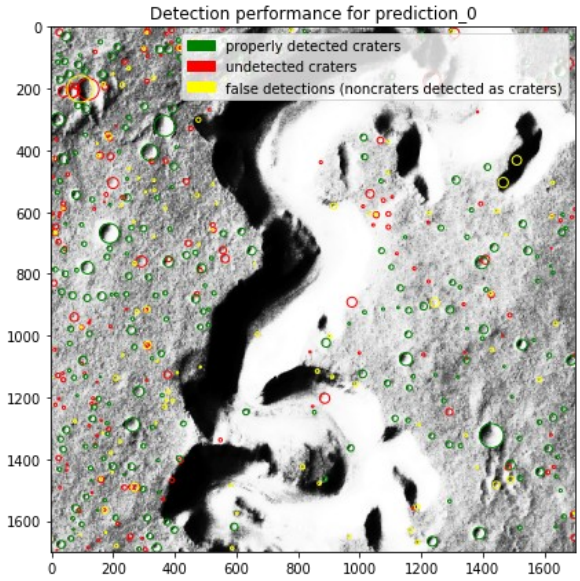

Figure 3: The pilot version of PyCDA features a test method.

The train method allows a user to retrain a model given the annotations provided with an input image.

By reducing these three tasks to simple calls to the model object, conducting partially-automated crater surveys should be easy to do in practice.

Crater Surveys with PyCDA: The process of conducting a survey of sub-km impact craters with PyCDA follows from these methods.

The surveyor must provide a number of hand-labeled annotations. It is advisable to identify the set of unique terrain types in within the survey area and provide hand-labeled examples from each of these terrain types.

The surveyor should then look for appropriate models on the public repository. A model trained on the same body is ideal, but a similar body may work. The surveyor should test a set of models on the provided hand-labeled data to identity the best performer. This also provides an estimate of uncertainty due to measurement error when interpreting the results of the survey.

If no model is acceptable, the best-performing model should be re-trained to improve performance for the survey at hand. After retraining the model, a surveyor is encouraged to share the model on the public repository for future use.

References:

[1] Vinogradova et al. (2002) *Proceeding of the IEEE Aerospace Conference, 3201, 3211*. [2] Krizhevsky A., Sutskever, I. and Hinton, G. E. (2012) *NIPS proceeding, 4824-1, 4824-9*. [3] Cohen, et al. (2016) *47th Lunar and Planetary Science Conference*, arXiv:1601.00978 [4] Emami E., et. al (2018) *49th Lunar and Planetary Science Conference*, researchgate [5] Robbins et al. (2014) *Icarus 234, 109, 131*.

Additional Information: For more information, please call Michael Klear at 650-218-1844 or send an e-mail to michael.klear@colorado.edu .