CS230

Image Segmentation and Object Detection of Lunar Landscape

Chen Wu, Ziyang Yuan
{chw177, yuanzy}@stanford.edu

Abstract

This project aims to find a solution to semantically segment lunar landscape images and detect some of their objects. There is already some realization of the method posted online. We used YOLOv2 to realize the object detection and use FCN-8s and DeepLabv3+ to do the semantic segmentation. From the result, FCN-8s and DeepLabv3+ can achieve a better result than posted models online.

1 Introduction and Motivation

Object detection and semantic segmentation play an important role in deep learning. They offer a basic foundation for some new technologies such as auto-driving. A lot of the data sets are used to realize the auto-driving inside the city so that the computers need to recognize pedestrians, buildings, traffic lights, etc.

Throughout this project, we apply several CNN algorithms to recognize a lunar landscape photo as its segmentation equivalent of the sky, ground, large rocks, and small debris. Different from the cityscape samples, people do not have a large number of real moon landscape images that are sufficient to train a robust deep neural network. Obviously, it is more costly to get more pictures of the lunar landscape than to get cityscape images. Therefore, a synthetic image of the lunar landscape is chosen for our CNN training rather than the real pictures. We tried several different kinds of ways to realize this task, including YOLO, FCN-8s, and DeepLabv3+. We believe it will be the preparation for the more complicated model of automatic driving on the lunar surface which can be deployed by various lunar exploration projects. We also regard this project, semantic segmentation, as a supplement of our study of CNN. The paper [1] is used as a guideline and starting point. All of our project code is in [Github Link].

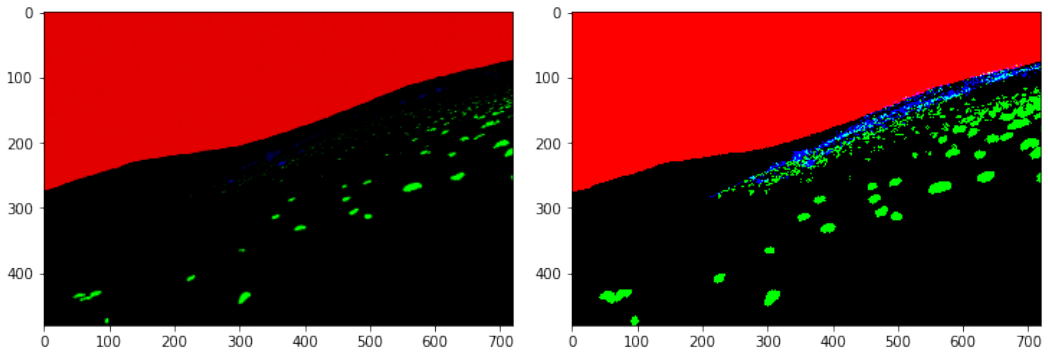
2 Dataset and features

Data processing is the main step that has been worked on during the whole project, apart from architecture selection. Our data set contains the synthetic image of the lunar landscape together with their corresponding class segmentation (ground truth) and the bounding box coordinate of the big rocks provided by the Ishigami Laboratory group of Keio University and posted on Kaggle as a game.

For the YOLOv2 algorithm, the bounding box according to the YOLO vectors is converted to do transfer learning. However, it turns out that the bounding boxes of some of our images are defined as inaccurately or completely wrong. Therefore, we shrank but improve the overall quality of the input size by filtering out the sample which does not fit in the ground truth. By detecting whether the center point of a bounding box lays inside a blue pixel (marked as a big rock in the ground truth), we filtered out some bounding boxes that were inaccurately labeled. We used (13,13) grid cells.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

For the segmentation algorithm, we used pairs of a synthetic lunar landscape image and its segmentation label. Two sets of segmentation labels, namely "ground truth" and "cleaned", containing class segmentation of background, ground, large rocks, and small rocks. As alarmed on Kaggle, the ground truth is not perfect. The pixels' RGB value in ground truth ranged from 0 to 226 with majorities on 0 and 225 sides. But the distribution of values between 0 and 225 is not sparse enough to be ignored. It is suggested to use the cleaned version provided which contains only 0 and 255 RGB values or to use self-cleaned ground truth. We tried to clean up the ground truth our-selves because, clearly, the raw ground truth contains much information than the cleaned one. The RGB of each pixel in the ground truth is set to be 255 if it is not 0 originally and all original 0s are kept. However, it turned out it overly exaggerates the class feature, shown in Fig.[1].



(a) (b)

Figure 1: (a) provided clean version ground truth and (b) shows the self-modified version of ground truth.

Originally, the raw ground truth and the cleaned version should be almost identical to human eyes, but here we can actually easily see a lot more features and it turned out later that it makes the learning more difficult. Therefore, finally, the cleaned version is used for easier training.

We divided our data set into a training set with 6400 images and a validation set with 2000 images. Start from YOLO, which will be discussed in detail later, some of the badly-paired synthesized images and segmentation which will definitely hurt the training are discovered and ruled out.

3 Methods

3.1 Object Detection

YOLOv2 was used as the object detection model to find the bounding boxes around big rocks in the photo [2]. We clone part of the work from Allan Zelener/YAD2K (https://github.com/allanzelener/YAD2K.git) to generate a pre-trained convolutional neural network model. The input of this model was (m,416, 416, 3) pixel data and the output is (m, 13, 13, 425). There are 50,962,889 trainable parameters in the model. We used the full YOLO pre-train weights from a Github project (https://github.com/allanzelener/YAD2K.git). The loss function for YOLOv2 was defined as the following:

$$
\begin{aligned}
L(x_i, y_i) = \lambda_{\textbf{coord}} & \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] + \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 + \sum_{i=0}^{S^2} L_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
\tag{1}
$$

Based on the labeled boxes, we generated 5 anchors for the YOLOv2 algorithm. We performed the training with Tensorflow 2.0, Keras 2.2.4, and Python 3.

3.2 Image Segmentation

FCN-8s and DeepLabv3+ are used to find the segmented classes of the synthetic images. For both of the models, we use the same loss function, cross entropy[2], and the same optimization model, Adams. The runnable architectures are selected from GitHub and modified to be compatible with Tensorflow2 and Python3.

$$L(x_i, y_i) = -\sum_i y_i \log x_i \tag{2}$$

FCN-8s[3] was originally expected to achieve a fundamental goal of segmentation training quality, but it turns out that it reached the highest quality among all methods we sought.

The code we borrowed follows the construction in the cited paper[4] in which a VGG16 is used to encode the input image to feature maps. Feature maps from the last 3 consecutive VGG-16 blocks are fused together and then transpose back to the original shape (Fig.[2])[3]. Originally, any layer that is not pool layer is set to be trainable because the VGG-16 we used is a pre-trained model and we were not sure whether the feature it learned can be compatible with the lunar landscape. However, it turned out that the amount of the weights is too large for p2.xlarge GPU. Even with the minibatch size of 8, the learning will be stopped by the system after several epochs. Therefore, several combinations might need to be tried to find a balance of more thorough training, setting more layers to be trainable, and a more affordable request of computing resources. Finally, we found that by freezing the first 15 layers and using a mini-batch size of 16 will carry out reasonable training.

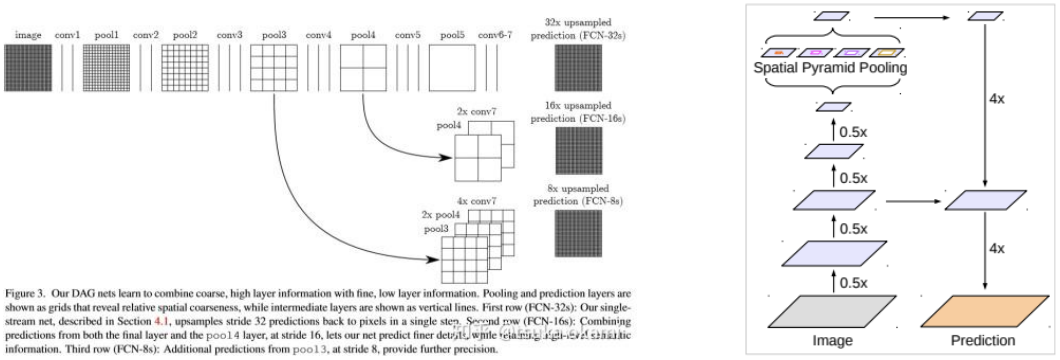


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

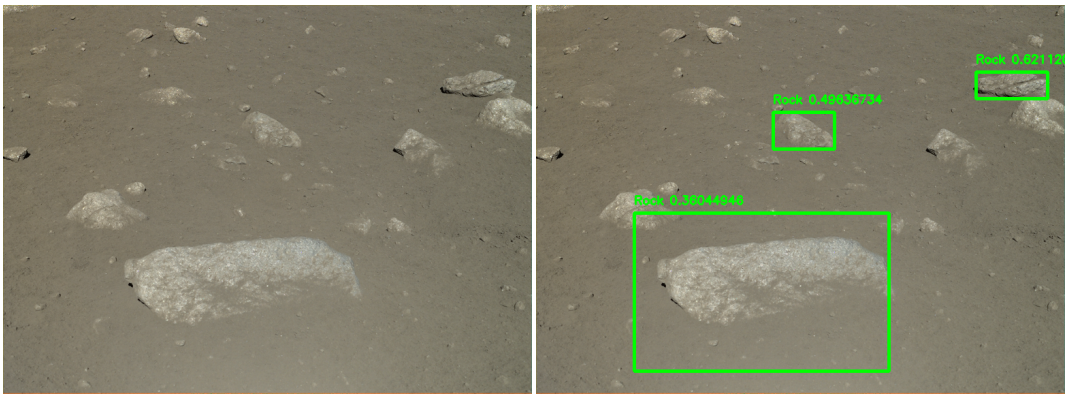
(a) (b)

Figure 2: (a) Structure of FCN-8s and (b) Structure of DeepLabv3+

In the end, we deploy a model of DeepLabv3+[4] which shows a very fast convergence in the learning process. The DeepLabv3+ is based on an atrous convolutional neural network. Throughout the training, we used a learning rate of 0.0005 at the beginning. Then there appeared a very obvious local minimum which gave quite unsatisfied segmentation in which some of the "sky" will be represented by the "ground". At the learning rate of 0.0005, when the loss does not decrease any more, the whole process will be shut down manually. After that, the learning rate will be modified to 0.0007 and the learning will be restarted based on the previously learned weights and the loss function can jump out of the local minimum and start to decrease. No matter we normalized the input or not, this strange behavior would be quite persistent.

4 Result

In object detection, we trained the convolutional neural network with a learning rate of 0.0001 and Adam optimizer. YOLOv2 algorithm generated the bounding boxes predictions with an accuracy of 63.4% mAP. The visualized results are shown by Fig.[3]



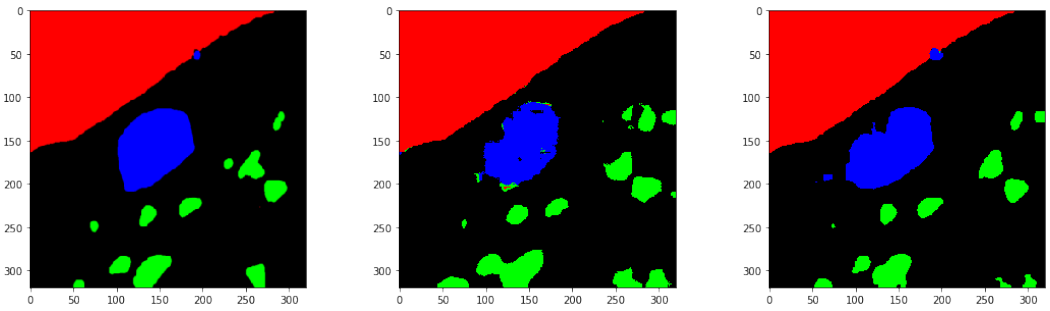
(a) (b)

Figure 3: YOLOv2 rock detection results: (a) Real Lunar Surface; (b) YOLOv2 prediction.

Table 1: Results of FCN-8s and DeepLabv3+

Model	final loss	Pixel-wise Accuracy	MeanIOU of training	MeanIOU of validation
FCN-8s	0.7666	0.9699	0.8829	0.8275
DeepLabv3+	0.08302	0.9728	0.8358	0.8232

In semantic segmentation, it is proved that the pixel-wise accuracy is not a very good metric for result comparison. Instead, a MeanIOU will be a much more reliable metric, which will see the percentage of the overlapping areas of the total area. During the training, the loss function and accuracy is Our result shows that FCN-8s, having MeanIOU of 0.8829, shows slightly better training result than DeepLabv3+ with MeanIOU of 0.8358. However, in the validation stage, both of the methods show results with similar qualities, 0.8275 for FCN-8s vs. 0.8232 for DeepLabv3+. In addition, the loss function of DeepLabv3+ will converge to a stable value in only 15 epoch which is way faster than the 100 epoch needed by the FCN-8s. We summarize the result in the following table.

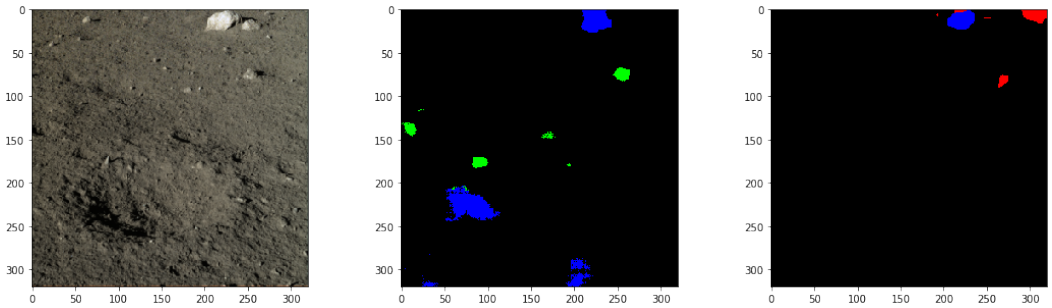


(a) (b) (c)

Figure 4: FCN and Deeplabv3 comparison on validation image set of synthetic lanscapes. (a) Cleaned ground truth; (b) FCN-8s prediction; (c) Deeplabv3+ prediction.

A straightforward comparison of one of our test case is shown in Fig.[5]. Overall, from the validation/test set, both of the two models are doing brilliant jobs. However, by inspecting carefully, we can see that DeepLabv3+ solves one of our deep worries which is the large stone seeming small at a far distance. At coordinate roughly at (200,50), there is, in fact, a large stone shown in the ground truth but becomes very small because of the distance and the confusion with the ground. DeepLabv3+ picked it out. More comparison can be tried in the jupyter notebook provided on GitHub.

We suspect that there is some overfitting in the training of DeepLabv3+ because FCN-8s will almost definitely give a better result in the generalization to the real moon landscape. In fact, we believe that a better trained DeepLabv3+ will have a higher loss value than 0.08.



(a) (b) (c)

Figure 5: FCN and Deeplab comparison on real moon surface . (a) Cleaned ground truth; (b) FCN-8s prediction; (c) Deeplabv3+ prediction.

This result is also compared with a privately posted result shown on [Dimartinot Web Link]. Four models and a 7000 images training set are used and the model with the highest IOU score is FPN, which is about 0.74. We show that the FCN-8s and DeepLabv3+ can actually give better results on this problem.

5 Conclusion and Future Work

In object detection, our training results show that YOLOv2 will be capable to detect most of the larger rocks in an image. The prediction of YOLOv2 was fairly accurate for both synthetic and real lunar images, although the algorithm could occasionally miss some rocks.

Overall, we believe FCN-8s did a better job on this problem because our ultimate goal is to train the model by the abundant amount of synthetic images and to see whether it works for the real moon landscape. FCN-8s has a more consistent performance on both of the synthetic testing and the real moon landscape testing.

However, it is possible that in the future a more realistic method will be realized to synthesize the real moon landscape. Just by eye-balling the image, it is quite obvious to know which one is the real moon picture and which one is the synthetic image. Apart from the synthetic quality, we can also try to augment the data and make it look much identical to the real lunar landscape before the training.

6 Contributions

In this project, Chen Wu focused more on setting up semantic segmentation and Ziyang Yuan focused more on setting up YOLO algorithms. Both of us contribute equally to debugging and modifying each other's focusing areas. Segmentation and YOLO both provide a lot of insights into the overall data preparation stage such as good images' selection, normalization problem, and contrast ratio. Since this is the first time for both of us entering the machine learning field, we also requested some advice about basic debugging methods from Tian Qiu who is a master student in the UCSD CS graduate program.

References

[1] Swarnendu Ghosh, Nibaran Das, Ishita Das, and Ujjwal Maulik. Understanding deep learning techniques for image segmentation. *ACM Computing Surveys (CSUR)*, 52(4):1–35, 2019.

[2] J Redmon and A Farhadi. Yolo9000: Better, faster, stronger. arxiv 2016. *arXiv preprint arXiv:1612.08242*.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.