Rajeev Ram – Thursday, December 20th, 2018
ECOL 499: Independent Research Project
Final Code Results

IDENTIFYING WINTER ANNUAL PLANTS OF THE SONORAN DESERT BY PERFORMING SEMANTIC SEGMENTATION WITH A U-NET ARCHITECTURE

A. Overview

The purpose of this project is to use machine learning to aid in locating and identifying of winter annual plant species in the Sonoran Desert. By automating this classification process, researchers can dedicate more effort and energy to tracing and studying the ecological dynamics of these species over time.

For this project, I have trained a neural network to perform semantic segmentation on images of winter annual plants. The goal of this project is to, given a set of images that contain common species of winter annual plants, accurately pinpoint areas (i.e., pixels) of the image that are and are not parts of plants.

I. Winter Annual Plants

Annual plants are species that complete their life cycle – including germination and the production of seeds – within one year. Winter annual plants germinate during autumn and winter and achieve maturity during the spring or summer of the following calendar year. Because annual plants tend to be small in size and irregular in shape, traditional image classification techniques used in other areas of plant biology are not particularly adaptable to this type of problem. Furthermore, because winter annual plants tend to be a more specialized area of study in desert ecology, there is not a significant amount of literature that describes techniques to utilize semantic segmentation to the identification and classification of winter annual plants. Therefore, it is difficult to apply already existing machine learning paradigms to this type of problem.

II. U-Net Architecture

The U-Net architecture describes a neural network configuration that is applied to the problem of semantic segmentation in biomedical images. The original U-Net architecture, and its corresponding problem area and application, is described in the following paper:

https://arxiv.org/abs/1505.04597

Traditional neural networks generally require thousands of annotated data samples to be trained successfully. This architecture makes heavy use of data augmentation – that is, warping already existing data into different shapes and sizes – to reduce the amount of training data needed, while still maintaining the predictive power of the machine model.

This advantage is primarily why I chose to apply the U-Net architecture to my problem of segmenting winter annual plants: the available data for my problem is highly limited. The hallmark of the U-Net architecture is a symmetric contracting and expanding path that enables the pixel-by-pixel labeling of images. I will not describe the details of what this path involves, but essentially, the consequence is that images can be precisely labeled in a way that allows users to clearly visualize and utilize the results.

B. Procedure

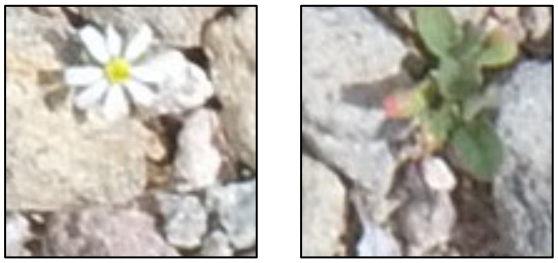
I decided to accomplish my machine learning task using the Keras library in the Python programming language. Keras is a commonly used and well-documented library used for a variety of deep learning tasks. More importantly, many other image segmentation problems using neural networks have been implemented in Python with Keras.

In particular, I have adapted the code from the original application of U-Net to biomedical imagery, located in the following repository:

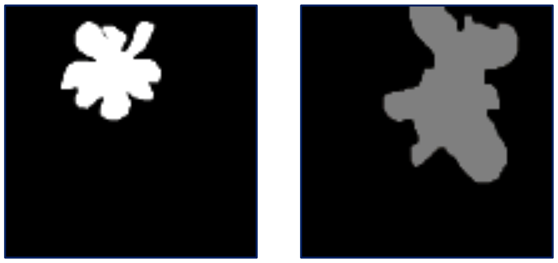
https://github.com/zhixuhao/unet

I. Labeled Images

For the time being, I have chosen to apply a machine model to two of the most common winter annual species in the Sonoran Desert: *Eriophyllum lanosum* (ERLA) and *Erodium texanum* (ERTE). The following figures show examples of the original and training data.



Original Training Images



Corresponding Labels

ERLA is labeled in white; ERTE is labeled in gray; the rest of the background – rocks, dirt, etc. – are labeled in black.

II. Adapting Code

The original code (in the repository referenced above) applies a binary model to the biomedical images for cell identification. However, in my case, the eventual goal is to train a three-class model, as described above. For now, though, I have adapted the code to also be a binary model, which means both plant species are combined into one class. That is to say, both the white color and gray color in the labels are treated as white color in the mask.

The following characteristics from the original source code are important to understand:

- All of the training and testing images must be 256-by-256 in size. This is because the U-Net architecture applies convolution and pooling layers based on powers of two.
- The labeled data is converted to a binary mask represented by a two-dimensional array.

The following modifications were made to the original source code for my purposes:

- The unlabeled training images and test images are saved as RGBA values, not grayscale. However, the labelled images are still converted to a binary mask.
- I used significantly more training data compared to the original use case.
- I split the original script, `main.py`, into distinct `train.py` and `test.py` scripts. However, I still maintain the same data augmentation dictionary.

III. Executing Code

(Please note that Python 3.6 or earlier must be used to successfully run the code! Consider using PyEnv to manage the appropriate dependencies when running from the command line.)

Run the script called `train.py` located in the `code` directory to train a new model from scratch. The resulting weights will be saved to a file called `unet_plant.hdf5`. Both the training (labeled and unlabeled) and testing images are included in the `data` subdirectory.

The following are suggestions for obtaining good results:

- Since there are sixty training images, the batch size times the number of steps per epoch should equal to sixty. For example, a batch size of two means thirty steps per epoch.
- (The training defaults right now are set to `batch size = 2`, `steps per epoch = 30`, `epochs = 20`)
- Less than tent epochs are not suggested as the result accuracy will be too low. More than thirty epochs do not result in a significant decrease in loss or increase in accuracy.

With the current settings, and without a GPU, the current model will be trained in somewhere between one to two hours. **I encourage the use of a GPU for faster processing, if available**.

C. Evaluation

Originally, I selected 250 training images. Ideally, I would have been able to label all of them and subsequently perform 10-fold cross validation to select weights with the lowest loss value. However, because of time constraints, I only managed to label 60 images in total – labeling 60 images took about five hours, so the remaining images could be labelled in a matter of 12 -15 more hours. Since I picked out these sixty images to label *at random*, though, I did not inappropriately engage in feature selection before training.

Furthermore, I randomly chose twenty images out of the remaining 190 images to serve as test data. Thus, even though, I was not able to perform cross-validation to select the best weights, I still maintained a sense of independence between the test data and training data. For future, if cross-validation were implemented, a much larger test set could be used, e.g., using 200 images for training and 50 for testing in partitioned iterations.

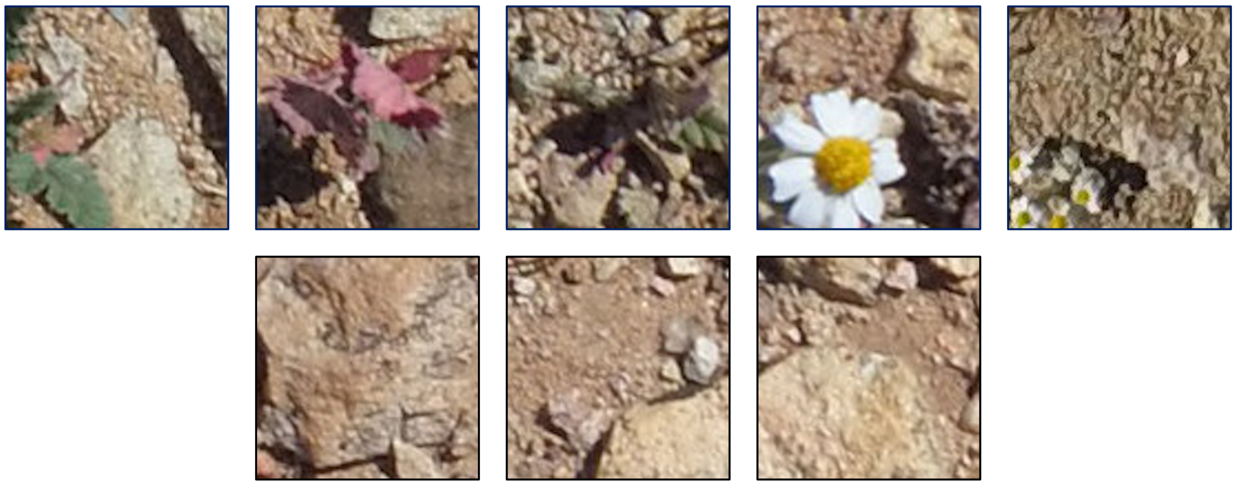
Run the script called `test.py` located in the `code` directory to test an already trained model. The file called **`unet_plant.hdf5`** must exist for the testing script to work. The `weights` subdirectory contains a set of weights used from an already completed training session.

D. Results

Below, I've reported some of the prediction results from my trained model. The complete results are included in the `results` subdirectory.

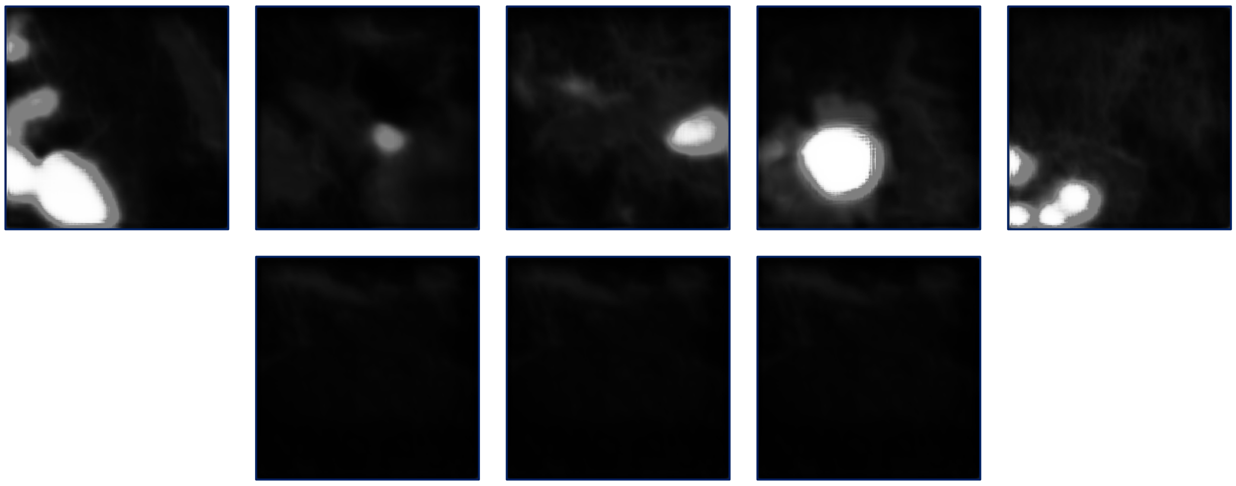
Observing the results, it is clear that the *color of the plant* plays a significant role in allowing the neural network to apply proper labels. Indeed, the second and third pictures are not labelled properly because the ERTE in those is at a different stage of maturity and looks significantly different that the other instances of ERTE within the training set.

On the other hand, the neural network has very little difficultly identifying ERLA. Both the smaller and larger instances, located in the fourth and fifth pictures, are labelled quite accurately.



Testing Images

ERLA is primarily white and yellow; ERTE is primarily green and purple.



Corresponding Prediction Results

The black color corresponds to the background; the white color corresponds to plants.

Finally, the neural network is able to correctly label tiles that do not have any plants, as indicated by the bottom set of three predictions. Such results are particularly promising because the model was only trained with images that do contain plants, indicating that the model is rather easily able to extrapolate its results. This means that the model can be applied to large swaths of land that contain areas both with and without plants and still engender a high degree of both precision and accuracy.

The following table summarizes the results for all twenty test images:

	Total Number Present	Correctly Identified	Number Misidentified	False Positive Rate	False Negative Rate
ERLA	11	11	0	0%	0%
ERTE	13	10	0	0%	23%

The fact that there a no false positives is highly encouraging; the false negative rate for ERTE can likely be reduced by using more testing data, and also by improving the model in various capacities, some of which are discussed below.

I. Suggested Improvements

- The eventual goal is to move to a three-class classification scheme. Thus, some error may be reduced when ERTE and ERLA are actually treated as separate classes. For instance, color may not play such an important role in prediction.
- The original architecture was applied to grayscale training images. Thus, converting the unlabeled data to grayscale from RGBA for training and testing might improve accuracy.
- Marking clear plant boundaries would mostly likely improve performance by a lot; the original literature emphasizes that the architecture contains a special function to emphasize cell-to-cell boundaries.
- Simply introducing more training data would likely help, too.
- Lastly, it would be interesting to modify the code to handle larger images (which would require more layers to be added to the model) and examine the results. This would be especially useful because the model could be applied to broad swaths of terrain.

II. Conclusion

Overall, I have successfully trained a neural network, configured into a U-Net architecture, to perform semantic segmentation on images to locate and label winter annual plants. While there are still ways to improve the model and expand its functionality, I’ve clearly demonstrated the utility and effectiveness in employing machine learning to solve a relatively unexplored classification problem in desert ecology.