ERDC/CRREL TR-18-7

Cold Regions Research and Engineering Laboratory

US Army Corps of Engineers®
Engineer Research and Development Center

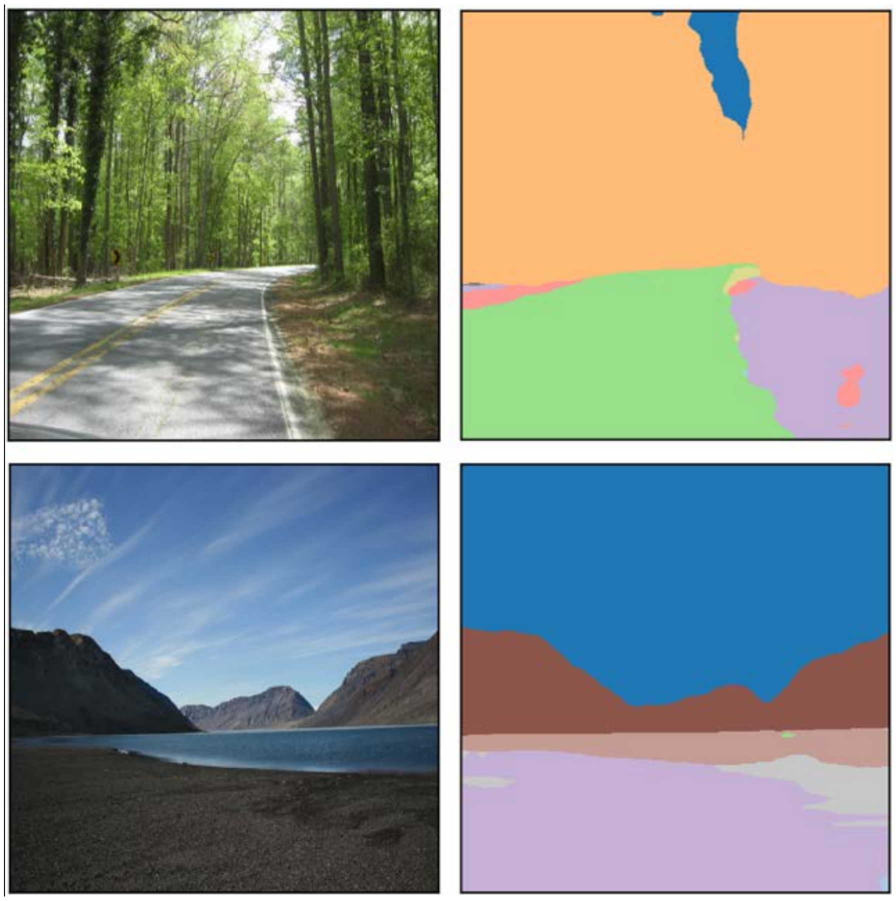
ERDC
INNOVATIVE SOLUTIONS for a safer, better world

Terrain Characterization for Rendering and Field Evaluation

Utility of Machine Learning Algorithms for Natural Background Photo Classification

Lauren E. Waldrop, Carl R. Hart, Nancy E. Parker, Chris L. Pettit, and Scotlund McIntosh

June 2018



Approved for public release; distribution is unlimited.

The U.S. Army Engineer Research and Development Center (ERDC) solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at http://acwc.sdp.sirsi.net/client/default.

ERDC/CRREL TR-18-7
June 2018

Utility of Machine Learning Algorithms for Natural Background Photo Classification

Lauren E. Waldrop, Carl R. Hart, and Nancy E. Parker

U.S. Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
72 Lyme Road
Hanover, NH 03755-1290

Chris L. Pettit

U.S. Naval Academy
Aerospace Engineering Department
590 Holloway Road
Mail Stop 11-B
Annapolis, MD 21402

Scotlund McIntosh

U.S. Army Natick Soldier Research, Development, and Engineering Center (NSRDEC)
15 Kansas Street
Natick, MA 01760

Final Report

Approved for public release; distribution is unlimited.

Prepared for Natick Soldier Research, Development, and Engineering Center
Warfighter Directorate, Fiber and Textile Science Team (RDNS-SEW-TMF)
10 General Greene Avenue
Natick, MA 01760

Under Funding Account U4355556, "Machine Learning for Natural Background Classification"

Abstract

In support of the Terrain Characterization for Rendering and Field Evaluation effort, the U.S. Army Corps of Engineers, Engineer Research and Development Center (ERDC), Cold Regions Research and Engineering Laboratory (CRREL), assisted the Natick Soldier Research, Development, and Engineering Center (NSRDEC) in evaluating machine learning algorithms to automatically classify three vegetation types (tree, shrub, and herbaceous), and a non-vegetated type in terrestrial images. In a previous partnership between CRREL and NSRDEC, researchers developed the Global Natural Background Image Database (GNBID), a collection of natural background images classified by vegetation attributes to include vegetation type and height, leaf shape, leaf color, and many others. Following deployment, the GNBID successfully improved on-the-ground understanding of natural background environments and quickly revealed the need for a larger database. Manual classification methods proved time intensive and variable, thus CRREL explored the feasibility of automatically identifying features using machine learning algorithms. In this scope-of-work study, we explore a multitude of computer vision techniques, settling on a supervised deep-learning technique. Here we present the advantages and disadvantages of various techniques, classification results from a subset of images, and recommendations for future research in this area.

Contents

Figures and Tables

Figures

Tables

Preface

The research described in this report was funded by the U.S. Army Natick Soldier Research, Development, and Engineering Center (NSRDEC) for the Terrain Characterization for Rendering and Field Evaluation program under Funding Account U4355556, "Machine Learning for Natural Background Classification."

The work was performed by the Biogeochemical Sciences Branch (CEERD-RRN) and the Signature Physics Branch (CEERD-RRD) of the Research and Engineering Division (CEERD-RR), U.S. Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory (ERDC-CRREL). At the time of publication, Dr. Justin Berman was Chief, CEERD-RRN; Dr. Andrew Niccolai was Chief, CEERD-RRD; and Mr. J. D. Horne was Chief, CEERD-RR. The Deputy Director of ERDC-CRREL was Mr. David B. Ringelberg, and the Director was Dr. Joseph L. Corriveau.

COL Bryan S. Green was the Commander of ERDC, and Dr. David W. Pittman was the Director.

Acronyms and Abbreviations

AHE	Adaptive Histogram Equalization
CLAHE	Contrast-Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
CONUS	Continental United States
CRREL	Cold Regions Research and Engineering Laboratory
ERDC	U.S. Army Engineer Research and Development Center
GLC2000	Global Land Cover 2000
GNBID	Global Natural Background Image Database
GUI	Graphical User Interface
HSV	Hue, Saturation, Value
IoU	Intersection over Union
LCCS	Land Cover Classification System
MIT	Massachusetts Institute of Technology
NSRDEC	U.S. Army Natick Soldier Research, Development, and Engineering Center
OCONUS	Outside the Continental United States
RGB	Red, Green, Blue
SLIC	Simple Linear Iterative Clustering
SME	Subject Matter Expert

1 Introduction

1.1 Background

Terrain characterization has long been a critical research area for military applications. To improve clothing and vehicle camouflage performance, the U.S. Army Natick Soldier Research, Development, and Engineering Center (NSRDEC) has categorized terrain based on visual natural backgrounds for field testing and evaluation. Although terrain characterization historically has been derived from overhead-collected data, this approach lacked the on-the-ground vantage point that is crucial for the evaluation of camouflage performance. In a partnership between the U.S. Army Cold Regions Research and Engineering Laboratory (CRREL) and NSRDEC, researchers developed a database and subsequent Graphical User Interface (GUI), called the Global Natural Background Image Database (GNBID). This database is composed of crowd-sourced terrestrial photos manually classified by natural background attributes (Parker et al. 2017; Parker and Jarvis 2017). The GNBID enables the user to select continental United States (CONUS) natural background analogue locations for areas of interest outside the continental United States (OCONUS). Following deployment, the GNBID successfully provided an improved on-the-ground understanding of natural background environments globally and quickly revealed the need for a larger database to both expand global coverage and increase photos within specific classes. Manual classification methods are time intensive and variable because of their subjectivity, even with a trained set of subject matter experts (SMEs). Therefore, to expand the database of images, new methods are required to decrease budget and labor requirements.

To address this need, we turn to computer vision techniques, which have been developed for automated identification of features within scenes. New approaches are rapidly evolving with respect to deep-learning neural networks like the convolutional neural network, greatly improving the visual classification of scenes (Zhou et al. 2017). This report outlines a brief exploration of both unsupervised and supervised image segmentation techniques to determine the utility of machine learning algorithms for vegetation feature identification within terrestrial ground-based images.

1.2 Objectives

To improve understanding of terrain types, beyond what's available in the GNBID, requires a larger database that maintains photos with improved global coverage and more natural background types. With shifting priorities and emerging areas of interest, greater global coverage within the GNBID would offer improved selection of natural background analogues and ultimately more accurate and realistic testing locations. Other critical gaps in the database include few photos from boreal environments and urban settings. Other program areas within NSRDEC might benefit from database expansion, particularly as soldier and vehicle camouflage development focuses on cold regions. However, expansion of the database by manual classification (as in the GNBID) would be both tedious and time-consuming, requiring intensive training with a high likelihood of variability. Instead, automated classification through machine learning techniques potentially reduces classification time, providing a substantial cost savings and enabling more accurate identification of natural background analogues.

To meet this need we identified the following objective: *Explore the feasibility of automatically identifying photos with similar visual natural backgrounds using unsupervised and supervised image segmentation techniques*. To do so, we assessed the limitations and capabilities of several machine learning models using the approach outlined below.

1.3 Approach

This project serves as a proof-of-concept study to identify the feasibility of unsupervised and supervised machine learning models to group photos with similar natural backgrounds. To meet this objective, we first researched and assessed several models and used a selection of photos from the GNBID to evaluate their limitations and capabilities. Using those results, we selected the model with the best performance to conduct a more in-depth evaluation using the entire dataset. This report documents and reviews the exploration of machine learning algorithms for identifying vegetation features within images, including comparisons between multiple models and techniques for improving performance. The report concludes by outlining recommended future research directions.

2 Global Natural Background Database

2.1 Classification attributes

During the development of the GNBID, the goal was to categorize photos by visual features so that photos with similar natural backgrounds would group together. Initially, this involved placing photos into broad land cover categories based on the Global Land Cover 2000 (GLC2000) dataset. However, this classification scheme placed photos in classes that were either too broad (photos within the same class looked dissimilar) or too narrow (photos that looked similar were placed in different classes). To remedy this, CRREL SMEs developed a new classification scheme, one that identified quantitative and qualitative attributes within a photo; assigned metrics to each photo based on these attributes; and then sorted photos, via the GUI, with similar metrics. NSRDEC fielded this technology and found more consistent alignment between terrestrial photos of areas of interest and their corresponding analogues as compared to previous classification schemes. However, the methodology proved to be time consuming and labor intensive. Each SME required training to "think alike" and identify attributes similarly for the photo sorting to succeed. Because the end product was a useful tool to NSRDEC, they requested that CRREL explore the use of computer vision techniques to reduce the time involved and error derived from manual classification.

The GNBID contains many detailed classification attributes; but, due to the short execution window for this project and the innovative nature of the research, the vegetation-type attribute served as a starting point for our investigation of machine learning. More specifically, photos in the GNBID are classified into one of four vegetation types: tree, shrub, herbaceous, or bare. During the development of the GNBDID, the SMEs used definitions from the Land Cover Classification System (LCCS) (Di Gregorio and Jansen 2005) to classify vegetation within the photos (Parker et al. 2017). Parker et al. (2017) reviews the criteria for each vegetation type. Figures 1–3 show images of the three vegetation types, and Figure 4 illustrates representative non-vegetated photos.

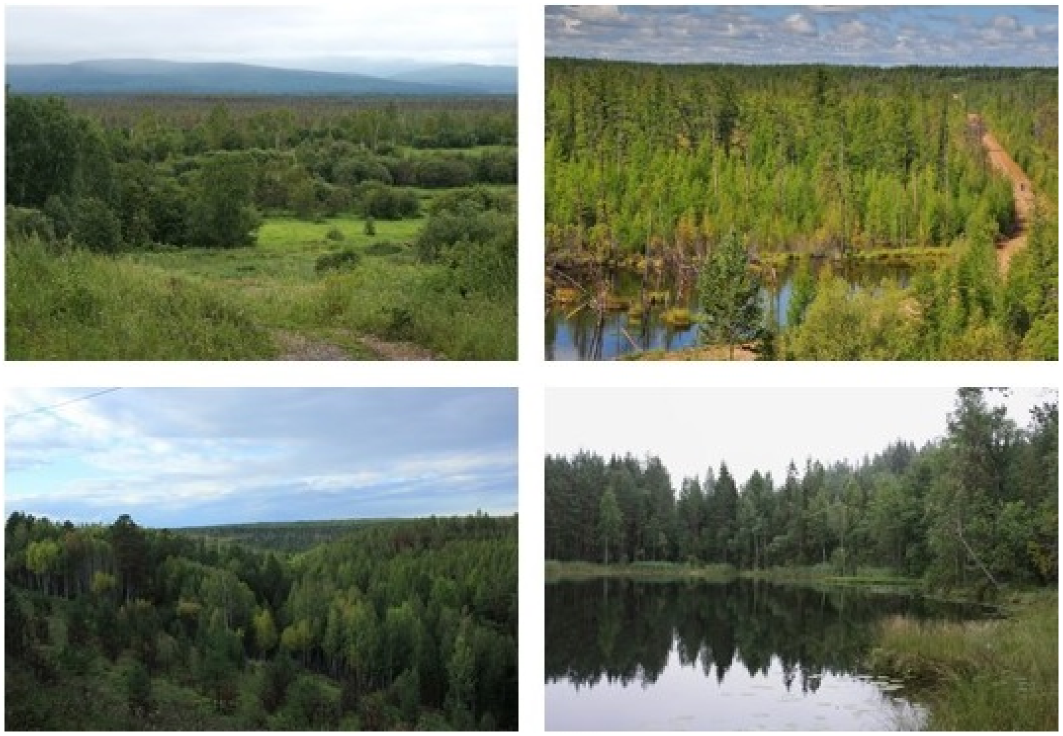
The SMEs initiated manual classification by placing photos into classes according to the dominant vegetation. A vegetation type was considered dominant if it served as the uppermost canopy level and represented a canopy cover of greater than 10%. For example, if an image contained 5%

tree cover, 30% shrub cover, and 60% herbaceous cover, it was considered a shrub-dominant photo because the shrub layer is the uppermost canopy exceeding the 10% threshold (Parker et al. 2017). However, for the purpose of this project, CRREL researchers defined dominant vegetation according to pixel ratio (this is defined more clearly in section 4).

2.2 Vegetation types from the GNBID

To begin exploring computer vision techniques, we selected 3–4 representative photos of each vegetation type (Figures 1–3) and for the non-vegetated category (Figure 4). Each selected photo represents a scene that contains one clearly dominant vegetation type (tree/shrub/herbaceous) or no vegetation at all. These were also selected based on ease of manual classification (human-based classification) (i.e., we did not select any sample photos that resulted in high variability amongst the SMEs).

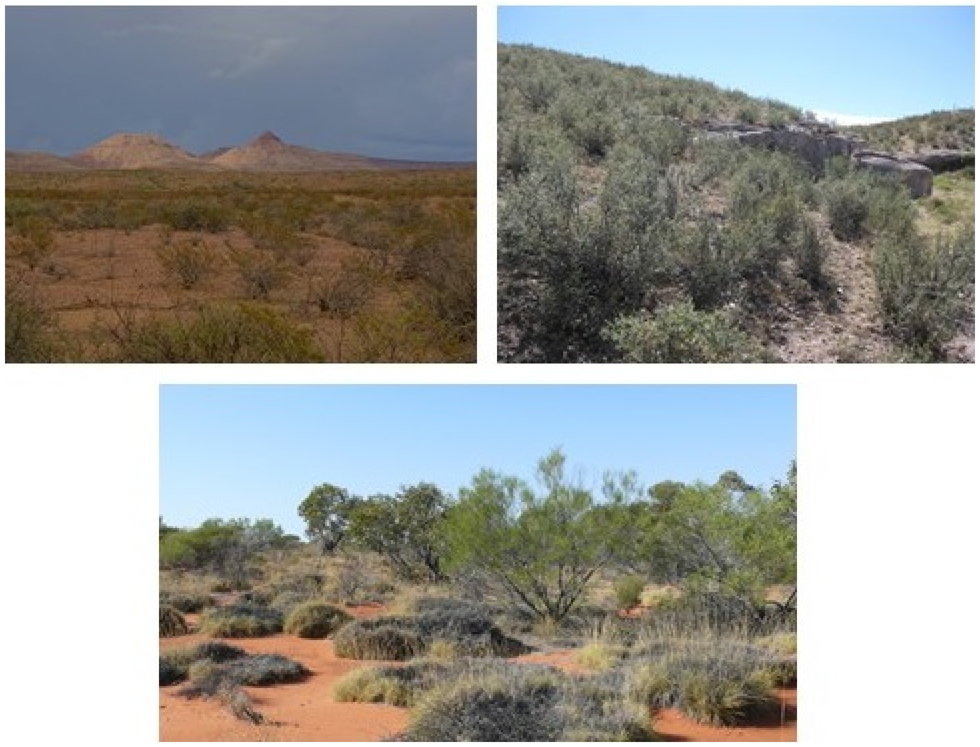
Figure 1. Representative tree-dominant photos.



There is quite a bit of variability in all three of the vegetation types. The most basic difference, specifically for trees and shrubs, is the deciduous versus coniferous seasonal cycle. While this characteristic is often easy to identify visually, it can be difficult given the lighting and scale of the photos. Trees can also look very different based on their habitat. For example, palm trees (common in warm/wet environments and low latitudes) are structurally quite different from coniferous trees (common in cold environments and at high latitudes/elevations). Additionally, seasonality affects the color and presence or absence of foliage. For the purpose of this project, we focused solely on identifying the vegetation type and not these more detailed identifying features.

Shrubs have much of the same variability described above (Figure 2). Like trees, shrubs have woody stems. The one key difference between a tree and a shrub is height. According to the LCCS, anything with a woody stem taller than 5 m is considered a tree while everything less than 5 m is a shrub (Di Gregorio and Jansen 2005). Many of the other characteristics are similar to those of trees. However, the height restriction can make distinguishing between trees and shrubs quite challenging, particularly when a photo lacks scale. This will also undoubtedly raise a challenge for the machine learning techniques.

Figure 2. Representative shrub-dominant photos.



Herbaceous plants lack the woody stem that defines trees and shrubs. However, there is still great variation across herbaceous plants, including color, leaf shape, and height, all making the identification process challenging (Figure 3). Because they are structurally quite different from trees and shrubs, these are probably the easiest vegetation type to classify manually. It can be challenging though when interspersed with woody vegetation, mainly because the shorter herbaceous plants are often shadowed by the other, taller vegetation.

Figure 3. Representative herbaceous-dominant photos.

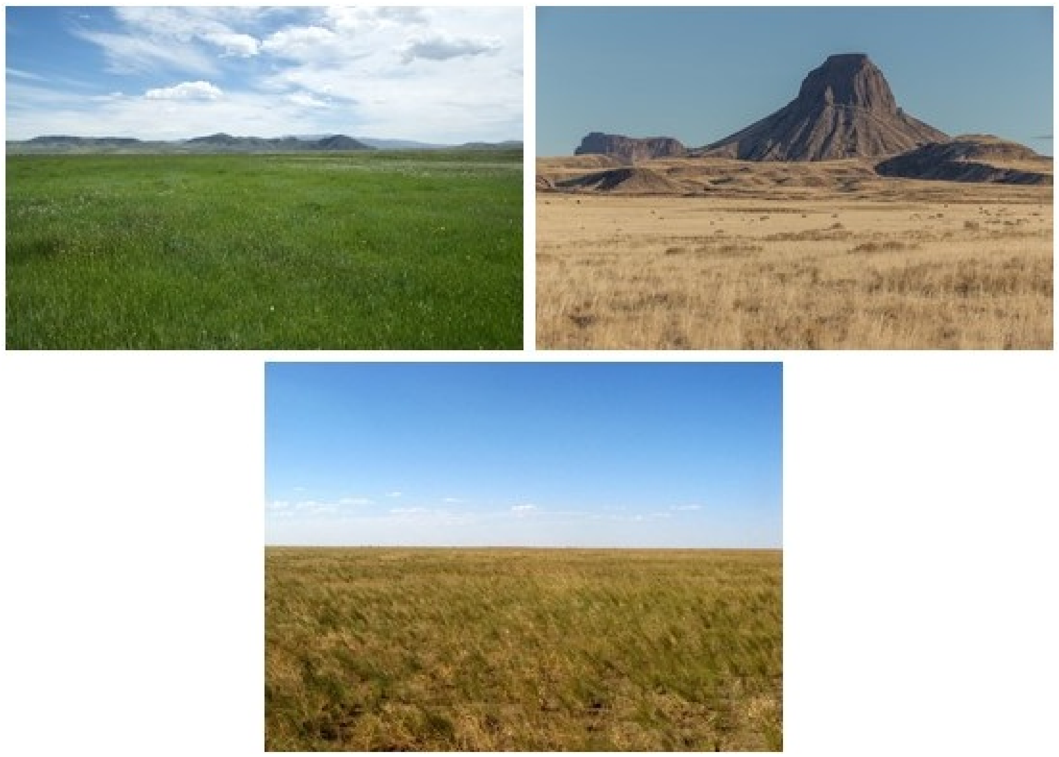
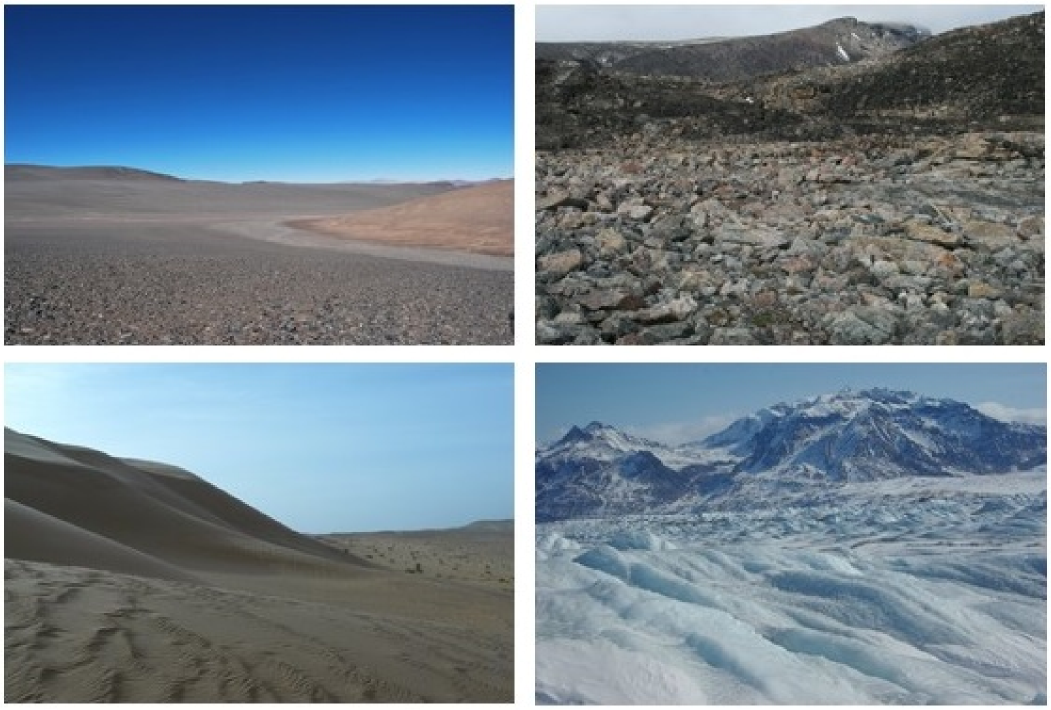


Figure 4 shows a selection of representative, primarily non-vegetated photos. This category may contain sparse vegetation (<4% of the ground cover in the photo) but is dominated by non-vegetative features. The photos may include natural features, such as rock, soil, water, and ice, or man-made features, such as buildings, roads, and parking lots. SMEs assumed no vegetation beneath photos dominated by snow or ice (Figure 4, bottom right) and classified these photos as “bare” within the GNBID.

Figure 4. Representative bare-dominant photos.



3 Image Segmentation Techniques

In both the image processing and computer vision domains, a long-standing challenge is the problem of image segmentation. A useful image segmentation algorithm groups perceptually similar regions (i.e., superpixels), which reflect global characteristics of an image. Furthermore, an image segmentation algorithm should be efficient, running in a time that is linearly proportional to the number of image pixels (Felzenszwalb and Huttenlocher 2004). Efforts to satisfy these requirements resulted in two broadly different approaches to unsupervised image segmentation.

Fundamentally, unsupervised image segmentation algorithms operate on either a graph-based or gradient-ascent approach (Achanta et al. 2012). A graph-based approach treats each pixel as a node in a graph. Edge weights between two nodes are proportional to the similarity between neighboring pixels. Minimizing a cost function defined over the graph generates superpixels. A gradient-ascent approach initially begins with a rough clustering of pixels relative to an estimate for the probability density function of the image feature space. Gradient ascent refines the initial clustering iteratively until a convergence criterion is satisfied. Upon convergence, clustered data points define a superpixel. Sections 3.1–3.3 describe three approaches to unsupervised image segmentation.

In contrast to unsupervised image segmentation, a highly sophisticated model built with a high-quality training dataset characterizes a supervised image segmentation algorithm. Commonly, models are neural networks built with a training dataset generated by expert annotation of images on a pixel-by-pixel basis. Section 3.4 describes one particular model (the dilated neural network).

3.1 Felzenszwalb (Felzenszwalb and Huttenlocher 2004)

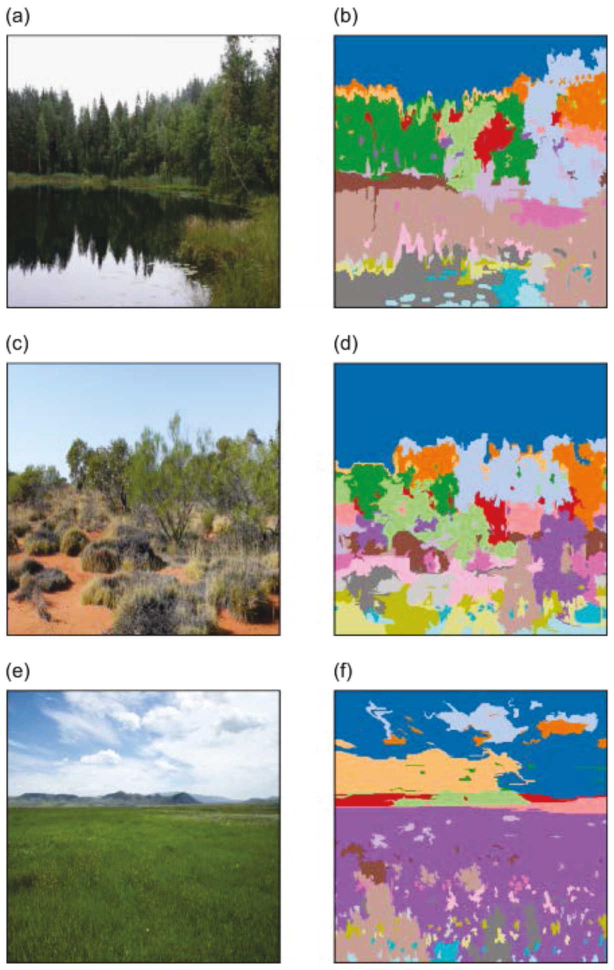
Felzenszwalb's algorithm is a graph-based method that clusters pixels into superpixels. The image grid defines a graph of pixel nodes connected to other nodes within a neighborhood. Color images, comprising three separate intensity images (i.e., red, blue, and green channels), are segmented separately and then intersected for an overall image segmentation. The algorithm builds upon classical clustering methods whereby an adaptive segmentation criterion quantifies the evidence for a boundary between two regions (potentially superpixels). Evidence for a boundary between two regions follows from the intensity differences across the regions and the intensity differences within each region.

A comparison function evaluates whether there is evidence of a boundary by weighting intensity differences within two regions and comparing this weighted result to the intensity differences across each region. The weighting of internal differences is a function of both region size and a tunable parameter. The tunable parameter, k, in Felzenszwalb and Huttenlocher's (2004) paper effectively sets a scale of observation. For large magnitudes of the tuning parameter, the algorithm tends to generate larger superpixels.

This method adheres well to image boundaries although it produces highly irregular superpixel sizes and shapes. It is $O(N\log N)$ complex, where N is the number of pixels, making this algorithm fast in practice. There is neither control over the number of superpixels nor their compactness, which is the ratio of area to perimeter (Achanta et al. 2012).

As an illustrative example, Figure 5 shows image segmentations of representative tree-, shrub-, and herbaceous-dominant images. An implementation of Felzenszwalb's algorithm in the scikit-image software library (van der Walt et al. 2014) generated the segmentations. A coarse parameter search was conducted for both the scaling parameter, $k \in \{100, 1000, 10000\}$, and the level of Gaussian image smoothing, $\sigma \in \{0.1, 1, 10\}$. In general, perceptually similar regions are most clearly generated for $k = 100$ and $\sigma = 1.0$. Segmentation of the tree-dominant image, Figure 5(a), effectively demarcates the sky, wooded regions beyond the body of water, reflected trees from the body of water, and even lily pads on the water in Figure 5(b). Segmentation of the shrub-dominant image, Figure 5(c), outlines the sky, upper and lower portions of shrubs, and several rocks on the sandy soil in Figure 5(d). Segmentation of the herbaceous-dominant image, Figure 5(e), largely captures the broad grassy field, the distant mountains, and distinct clouds in Figure 5(f). When the scaling parameter is larger, superpixels encompass multiple regions that are perceptually distinct, resulting in a segmentation that is too coarse. The scaling parameter used here is consistent in magnitude with the scaling parameter used by Felzenszwalb and Huttenlocher (2004), which was 300.

Figure 5. Image segmentations using Felzenszwalb's algorithm, with the scale parameter, *k*, set to 100 and Gaussian smoothing, *σ*, set to 1.0: (*a*) tree-dominant image and (*b*) segmentation, (*c*) shrub-dominant image and (*d*) segmentation, and (*e*) herbaceous-dominant image and (*f*) segmentation.



3.2 Quick shift (Vedaldi and Soatto 2008)

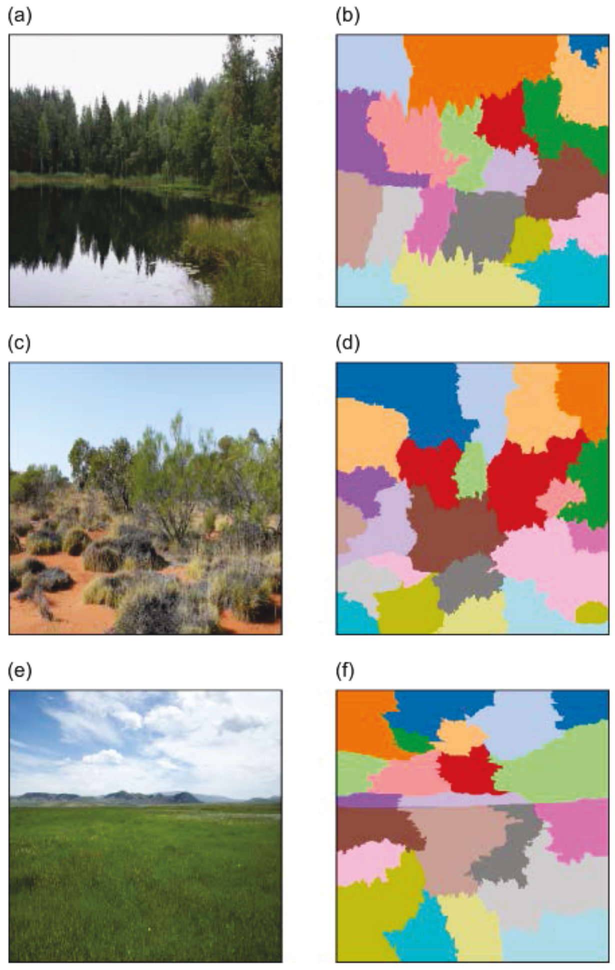
The quick-shift algorithm is a gradient-ascent method that uses a mode-seeking procedure. Local maxima of a probability density function, for the intensity feature space of an image, are located through an iterative search. Specifically, a mode-seeking clustering algorithm begins by generating a Parzen density estimate (i.e., a nonparametric density estimate via a kernel function, be it Gaussian or otherwise) of the intensity feature space, given N data points, and then nudges each data point iteratively along a trajectory to the nearest mode. The trajectory of each data point approximately follows along a gradient of the estimated probability density function. Once a group of data points converges to a mode, those data points are assigned to a single cluster.

Two common mode-seeking algorithms are mean shift and medoid shift. Mean shift evolves the trajectory of each data point by approximately following the gradient of the probability density function by maximizing a quadratic lower bound on the window used for the Parzen density estimate. The primary drawback of mean shift is its computational complexity, being $O(dN^2T)$, where d is the dimensionality of the data space and T is the number of iterations. Medoid shift constrains the trajectories of data points to go through other data points. The chief advantages resulting from this change are that only a single iteration is required, no stopping/merging heuristic is required, and the data space may be non-Euclidean. The complexity of a basic implementation of medoid shift is $O(N^3 + dN^2)$. A major disadvantage of medoid shift is a possibility of misidentifying modes of the underlying probability density function, which is termed an overfragmentation of modes.

An alternative to mean shift and medoid shift is an algorithm called quick shift. In this case, each data point moves to the nearest neighbor, which has a higher probability density. All data points connect to form a single tree, and modes are distinguished by eliminating branches longer than a particular threshold, τ. The algorithm adheres to boundaries relatively well; however, it is very slow with a complexity of $O(dN^2)$ (d being a small constant). It does not allow for control of the number or size of superpixels (Achanta et al. 2012).

As an illustrative example, Figure 6 shows image segmentations of the same representative tree-, shrub-, and herbaceous-dominant images as in using the quick-shift algorithm (Figure 5). An implementation of the algorithm in the scikit-image software library (van der Walt et al. 2014) generated the segmentations. Image conversion from RGB (red, green, blue) to CIELab color space occurs prior to segmentation. The algorithm has four tunable parameters: a ratio (r) between color-space proximity and image-space proximity, the Gaussian window width (w) for generating the Parzen density estimate, the maximum data distance (τ from above), and the level of Gaussian smoothing (σ). A coarse parameter search was conducted for the ratio, $r \in \{0, 0.1, 1.0\}$; Gaussian window width, $w \in \{10, 30\}$; and data distance, $\tau \in \{10, 30\}$. Gaussian smoothing was set to 1.0, which represents a moderate level of smoothing. In general, perceptually similar regions are most clearly generated for $r = 1.0$, $w = 10$, and $\tau = 30$. Segmentation of the tree-dominant image, Figure 6(a), separates the sky into three regions, broadly segments the wooded regions beyond the body of water and reflected in the water, and separates the sky-reflecting water into two regions in Figure 6(b). Segmentation of the shrub-dominant image, Figure 6(c), outlines the sky in a disjointed manner and is able to distinguish shrubs in the center to left portions of the image in Figure 6(d). Segmentation of the herbaceous-dominant image, Figure 6(e), largely captures the broad grassy field, the distant mountains, and sky in a disjointed manner in Figure 6(f). When the ratio parameter is small, perceptually distinct regions are merged together, resulting in too coarse of a segmentation. Larger maximum data distance (τ) results in fewer superpixels without much loss in boundary adherence. Larger Gaussian windows also result in fewer superpixels; however, some loss in boundary adherence can occur.

Figure 6. Image segmentations using the quick-shift algorithm with the ratio parameter, *r*, set to 1.0; Gaussian window width, *w*, set to 10; data distance, *τ*, to 30; and Gaussian smoothing, *σ*, set to 1.0: (*a*) tree-dominant image and (*b*) segmentation, (*c*) shrub-dominant image and (*d*) segmentation, and (*e*) herbaceous-dominant image and (*f*) segmentation.



3.3 Simple linear iterative clustering (Achanta et al. 2012)

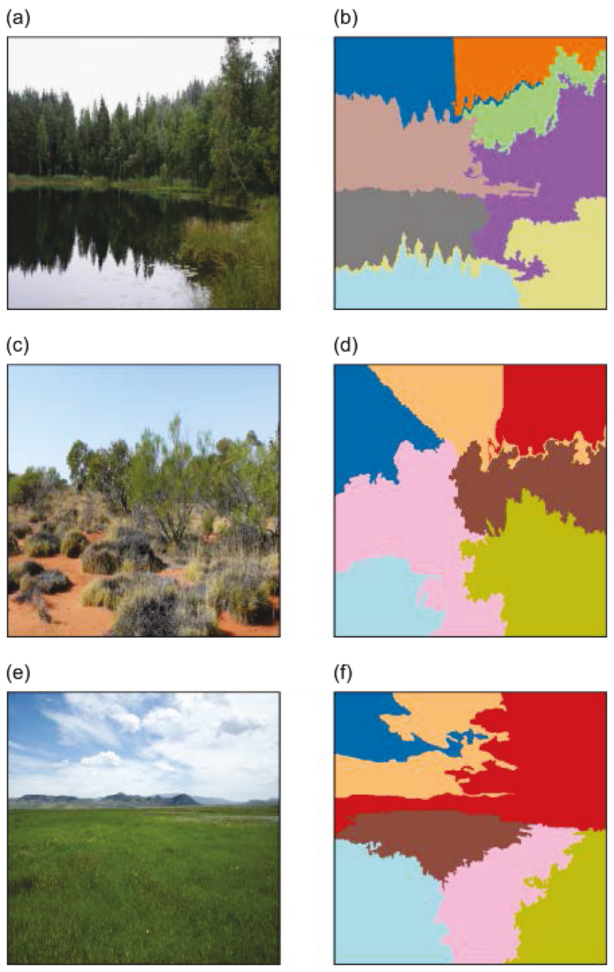
Simple linear iterative clustering (SLIC) adapts a k-means clustering approach for efficiently generating superpixels. The k-means clustering algorithm is adapted with two distinct modifications:

1. The search space is limited to a region proportional to the superpixel size.
2. A weighted distance measure, combining color (CIELAB color space) and spatial proximity (x, y pixel locations), is used to control size and compactness of superpixels.

The weighted distance measure normalizes Euclidean distances for space by the maximum distance within a cluster and color by a constant value. The constant value weights the relative importance of color similarity to spatial proximity. When the constant (m, called compactness) is large, spatial proximity is more important, resulting in more compact superpixels. When the compactness constant is small, the superpixels adhere more closely to image boundaries; however, they are more irregular or less compact. The range of m is $[1, 40]$ when using the CIELAB color space. The computational complexity is $O(N)$, which is independent of the number of superpixels, making this algorithm very fast. Only Felzenszwalb's algorithm is comparable in complexity.

As an illustrative example, Figure 7 shows image segmentations of the same representative tree-, shrub-, and herbaceous-dominant images, as in Figure 5, using the SLIC algorithm. An implementation of the algorithm in the scikit-image software library (van der Walt et al. 2014) generated the segmentations. Image conversion from RGB to CIELAB color space occurs prior to segmentation. The algorithm has three tunable parameters: the maximum number of superpixels (k), the compactness parameter (m), and the level of Gaussian smoothing (σ). A coarse parameter search was conducted for the number of superpixels, $k \in \{10, 100, 1000\}$; compactness parameter, $m \in \{1, 10, 100\}$ (where 100 is effectively 40); and Gaussian smoothing, $\sigma \in \{0.1, 1, 10\}$. In general, perceptually similar regions are most clearly generated for $k = 10$, $m = 10$, and $\sigma = 1$. Segmentation of the tree-dominant image, Figure 7(a), separates the sky into two regions, broadly segments the wooded regions beyond the body of water and reflected in the water, and generates a single superpixel for water reflecting the sky in Figure 7(b). An artifact of the post-processing imposed by the maximum number of superpixels is joining of superpixels for trees in the foreground and trees reflected in the water. Segmentation of the shrub-dominant image, Figure 7(c), outlines the sky with three regions, broadly identifies the shrubs in the foreground, and joins shrubs in the background with foreground stones and soil in Figure 7(d). Segmentation of the herbaceous-dominant image, Figure 7(e), segments the broad grassy field into four superpixels, merges the mountain range with a large portion of sky, and captures some variation in the sky (upper left) in Figure 7(f). An ideal maximum number of superpixels lies somewhere between 10 and 100. When the compactness parameter is very large, superpixels become square; and when it is too low, it generates too fine of a segmentation.

Figure 7. Image segmentations using the SLIC algorithm, with the maximum number of superpixels, *k*, set to 10; compactness parameter, *m*, set to 10; and Gaussian smoothing, *σ*, set to 1.0: (*a*) tree-dominant image and (*b*) segmentation, (*c*) shrub-dominant image and (*d*) segmentation, and (*e*) herbaceous-dominant image and (*f*) segmentation.

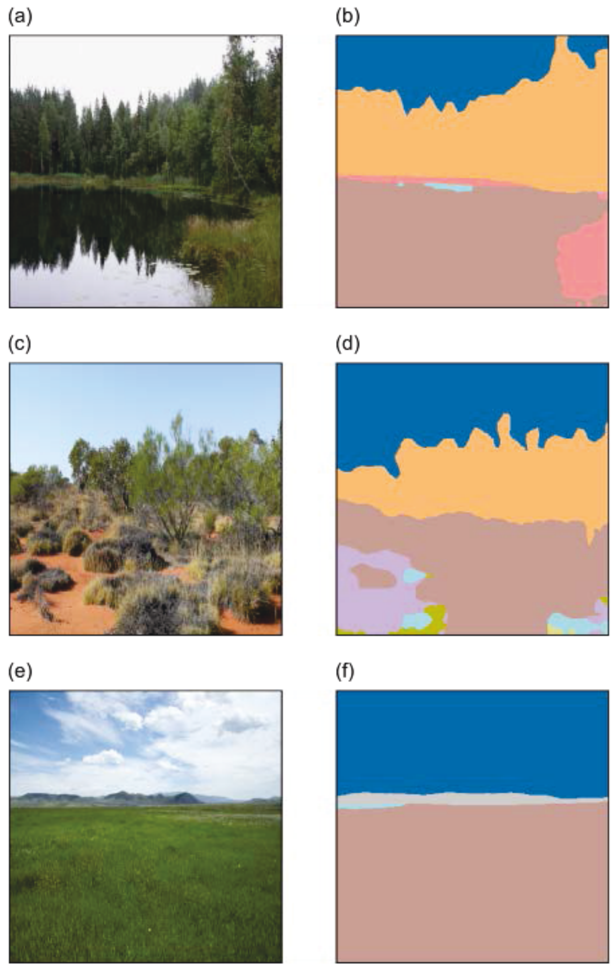


3.4 Dilated neural network (Zhou et al. 2017)

A dilated neural network model is a supervised image segmentation technique that also generates a semantic labeling (Zhou et al. 2017). The neural network model is a modification of the 16-layer neural network model, known as VGG-16, by dropping a couple of pooling operations and replacing the following convolutions with dilated convolutions. A bilinear upsampling layer completes the last layer of the model. A comprehensive dataset comprising over 20,000 expertly annotated images, forms the training, validation, and test sets for the dilated neural network. An expert labels each image with object and part annotations on a pixel-level basis. From this dataset, a scene-parsing benchmark was created with the top 150 objects. The dilated neural network is trained on the benchmark dataset.

As an illustrative example, Figure 8 shows image segmentations of the same representative tree-, shrub-, and herbaceous-dominant images, as in Figure 5, using the dilated neural network model (http://sceneparsing.csail.mit.edu/model). Appendix A describes in detail how to use the PyCaffe interface for testing and evaluating this model. Prior to segmentation, several image preprocessing steps are required: resize to 384 by 384 pixels; subtract the RGB means of the training dataset; transpose the dimensions from height, width, and channel to channel, height, and width; set scale from [0, 1], to [0, 255]; and swap channel ordering from RGB to BGR. The model requires no tuning parameters besides the image itself. Here we conducted a test to evaluate the effect of Gaussian smoothing, $\sigma \in \{0.1, 1, 10\}$. In general, perceptually similar regions are most clearly generated for $\sigma = 0.1$, possibly as a consequence of maintaining a distinction between the pixel-level granularity of the original training dataset. This is in contrast to unsupervised image segmentation techniques where $\sigma = 1.0$ generates perceptually similar regions. This segmentation of the tree-dominant image, Figure 8(a), does very well in separating the sky, trees, grass, and water in Figure 8(b). Surprisingly, the water with tree reflections does not present a challenge to the model. Segmentation of the shrub-dominant image, Figure 8(c), outlines the sky and tops of shrubs and categorizes the lower portions of stems, grass, and rocks as plants in Figure 8(d). In general, shrub-dominant images present the greatest challenge for the model. Segmentation of the herbaceous-dominant image, Figure 8(e), segments the broad grassy field cleanly and identifies the mountain range and sky in Figure 8(f). Interestingly, the variation in the sky cover does not present an issue for the dilated neural network as a consequence of the labeling scheme.

Figure 8. Image segmentations using the dilated neural network algorithm with Gaussian smoothing, *σ*, set to 0.1.: (*a*) tree-dominant image and (*b*) segmentation, (*c*) shrub-dominant image and (*d*) segmentation, and (*e*) herbaceous-dominant image and (*f*) segmentation.



3.5 A comparison between unsupervised and supervised image segmentation techniques

Both unsupervised and supervised image segmentation algorithms are increasingly fulfilling the goals set forth by Felzenszwalb and Huttenlocher (2004): group perceptually similar regions, and run in a time linearly proportional to the number of pixels. Among existing unsupervised algorithms, selection criteria for study included an open implementation and minimal manual interaction. Satisfying these criteria are Felzenszwalb and Huttenlocher's (2004) graph-based algorithm, the quick-shift gradient-ascent algorithm (Vedaldi and Soatto 2008), and the graph-based SLIC algorithm (Achanta et al. 2012). These image segmentation techniques, also known as superpixel algorithms, have varying levels of success, depending on the selection of one or more tuning parameters. In general, superpixel generation may serve as a preprocessing step in further segmentation algorithms (Achanta et al. 2012). For example, one method of multi-class image segmentation uses superpixels to compute color, texture, geometry, and location features to train classifiers for a finite number of object classes (Gould et al. 2008). Along these lines, state-of-the-art supervised image segmentation algorithms solve the image segmentation problem very well. Here we reviewed and tested the dilated neural network model trained on a scene-parsing benchmark dataset (Zhou et al. 2017). Compared to the unsupervised approaches, the dilated neural network segments images with greater success. Table 1 summarizes the characteristics of the techniques described above. Appendix B contains code listings for the generation of image segmentations by each algorithm.

Table 1. Summary of characteristics for unsupervised and supervised image segmentation techniques.

Algorithm	Unsupervised (U), Supervised (S)	Speed[1] (slow, medium, fast)	Number of parameters[2]
Felzenszwalb	U	fast	1
Quick shift	U	slow	3
SLIC	U	fast	2
Dilated neural network	S	fast	0

1. The speed designation is on the order of hours, minutes, and seconds for slow, medium, and fast, respectively.

2. Not including Gaussian smoothing.

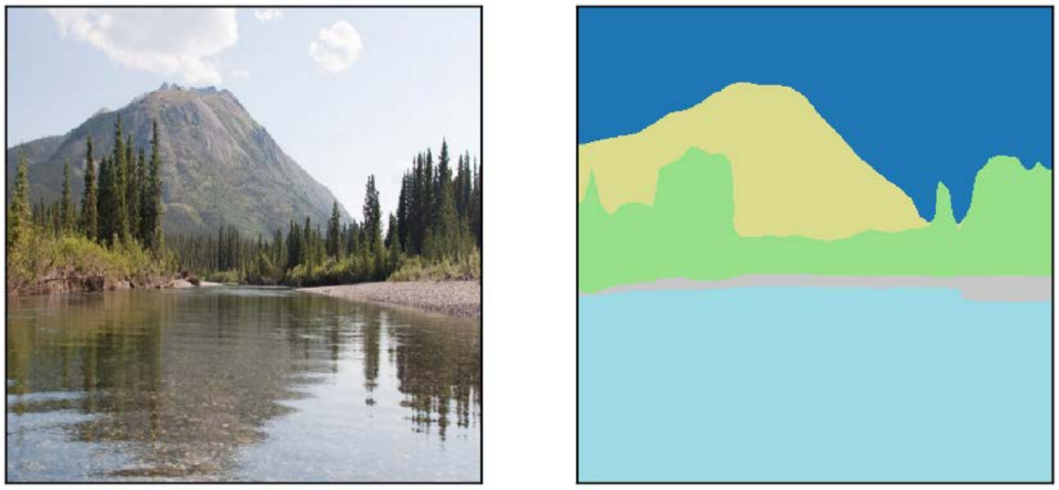
A visual comparison of Figures 5 through 8 demonstrates the advantages of a supervised image segmentation when compared to unsupervised image segmentation. Additional benefits include fast speed and no parameter specification. For these reasons, we explore implementations of the dilated neural network and datasets used for semantic segmentation. The next section introduces semantic segmentation and relevant datasets broadly and specific implementation details further on.

4 Supervised Image Segmentation Datasets and Techniques

4.1 Semantic segmentation and relevant networks

In the field of computer vision, semantic segmentation takes as input an image and attempts to assign a label to each pixel. The resulting output is a mask, which is often easier to analyze because objects of the same class maintain the same pixel color and clear boundaries between objects are established. Figure 9 demonstrates the semantic segmentation of a tree-dominant-vegetation ground-based scene.

Figure 9. Semantic segmentation demonstration. The box on the left illustrates the initial image while the box on the right demonstrates a mask generated after semantic segmentation.



Semantic comprehension of visual scenes and objects in an image is relevant for many applications, including autonomous vehicle systems and land cover classification and boundary delineation on remotely sensed images. For our purposes, we were interested in generating a mask for each ground-based scenic image in the GNBID for which we had a dominant vegetation type ground truth label (bare-, herb-, shrub-, or tree-dominant). Ultimately, we would look to the mask as a way of determining the dominant vegetation type for a given scene. In previous efforts related to this project (Parker et al. 2017; Parker and Jarvis 2017), these image-level ground truth labels were annotated by multiple experts, thereby exposing the labeling process to inconsistencies across annotators. Through the automation of dominant vegetation labeling by way of semantic segmentation, we hope to alleviate these inconsistencies.

Dense pixelwise semantic labeling is possible by the implementation of convolutional neural networks (CNNs). Examples of these include SegNet (Badrinarayanan et al. 2017), FCN-8s (Shelhamer et al. 2017), and DilatedNet (Yu and Koltun 2016), each distinct in their respective architectures.

4.2 Semantic segmentation datasets

We set out initially to identify semantic segmentation datasets that would suit our primary purpose of providing masks whose dominant class corresponded with that of the ground truth vegetation type. In our study of available semantic segmentation datasets, we identified four that align most closely with our ground truth dataset: CityScapes (Cordts et al. 2016), Pascal-Context (Mottaghi et al. 2014), the SUN database (Xiao et al. 2010), and the Massachusetts Institute of Technology (MIT) ADE20K (Zhou et al. 2017).

Each of the four datasets maintain nuanced advantages and disadvantages when looking through the lens of our specific objective. More specifically, CityScapes, one of many semantic segmentation datasets with pixel-level labels, recognizes vegetation and terrain as classes but neglects to provide any specificity beyond those two categories. To the best of our assessment, the Pascal-Context dataset failed to provide the specificity needed for discrete outdoor vegetation categories or “stuff” classes (i.e., classes derived from large regions of an image that are not defined by an explicit shape). Within Pascal-Context, “stuff” classes such as sand and snow and discrete object classes such as plant and tree are represented, but this dataset overlooks any categories that might potentially align with the herbaceous ground truth category. One notable exception to these limitations is the SUN database, which does provide 131,067 images and 4479 object categories at the time of this writing. From those object categories, we could sufficiently map to our ground truth labels; however, the SUN database reportedly maintains noisy labels at the object level (Zhou et al. 2017).

In our hunt to find a pixel-level labeled image dataset, we landed on the MIT ADE20K and its subsequent benchmark, SceneParse150 (Zhou et al. 2017).

4.3 SceneParse150 and the DilatedNet

The SceneParse150 dataset was appealing for a couple of reasons. First, SceneParse150 provides 150 objects and is derived from the ADE20K dataset, which contains 22,210 densely labeled scene-centric images, all labeled by a single annotator with an unrestricted vocabulary. Second, SceneParse150 also presents the specificity of interest for "stuff" classes. "Stuff" classes are of high interest for this particular study as herbaceous- and bare-dominant photos maintain a large percentage of pixels that align with "stuff" classes. SceneParse150 maintains 35 "stuff" classes and 115 discrete object classes.

In their paper, Zhou et al. (2017) demonstrate that the DilatedNet outperforms both the FCN-8s and the SegNet in both pixel accuracy and mean intersection over union (IoU) on the SceneParse150 validation set. Therefore, we wanted to initially test whether the DilatedNet trained on MIT's SceneParse150 benchmark dataset could sufficiently provide masks whose dominant class corresponded with that of the ground truth vegetation type.

Zhou et al. (2017) implement and integrate what they call a cascade segmentation module to counter the long-tail distribution associated with the pixel ratios of objects. The justification for this distribution is that "stuff" classes tend to largely dominate the percentage of annotated pixels while smaller, more discrete objects occupy a smaller percentage of pixels. When integrated with other baseline networks such as the SegNet and the DilatedNet, the cascaded versions outperform their non-cascaded counterparts in both pixel accuracy and mean IoU on the MIT SceneParse150 validation set. We were not able to obtain Zhou et al.'s DilatedNet with integrated cascade segmentation module at the time of this writing. Instead, we relied on Zhou et al.'s trained DilatedNet model (http://sceneparsing.csail.mit.edu/model/) and used PyCaffe to run the model (installation instructions are in Appendix A of this report).

To assess whether the DilatedNet trained on MIT's SceneParse150 could identify the dominant ground truth vegetation type in an image, we ran all ground-based images with vegetation-dominant labels through the network and implemented post-processing (discussed in the next section) on the output to identify dominant classes of particular interest.

4.4 Mapping object classes

The SceneParse150 dataset maintains 150 object classes; however, only the following were of particular interest for this study: dirt track, earth, field, grass, palm, plant, rock, sand, and tree. We identified this set of classes as having the potential to map to one of our four ground truth classes. The ground truth classes for this study include bare, herb, shrub, and tree. To reconcile the differences between the sheer volume of classes available within the SceneParse dataset and those of particular interest, we recognize that transfer learning is a highly applicable methodology; however, for this study, we attempted to leverage the trained network and post-process the output. For each image, we ranked the classes in the output according to the recorded pixel ratios and subsequently took the intersection of the output classes and the classes of interest to generate our final ranked list. We then recorded the class with the largest percentage of pixels as the dominant label for that image.

After this preliminary manipulation, the resulting output, in some cases, revealed images whose dominant classification maps to a ground truth classification using the mapping scheme presented in Table 2.

Table 2. Mapping from classes of interest to ground truth classes.

SceneParse150 Classes of Interest	Ground Truth Classes
Field, Grass	Herb
Dirt track, Earth, Rock, Sand	Bare
Tree, Palm	Tree
Plant	Shrub

4.5 Contrast enhancement and sharpening techniques

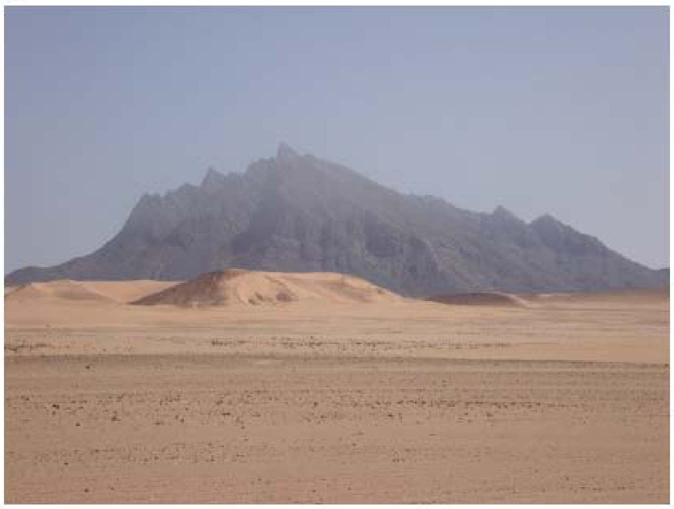
4.5.1 Contrast-Limited Adaptive Histogram Equalization

Global contrast enhancement via histogram equalization is beneficial when an entire image is bright or dark, or when the background is not easily distinguishable from the foreground. Generally, images within the GNBID are not low contrast images; but we wanted to employ a similar technique, Contrast-Limited Adaptive Histogram Equalization (CLAHE) (Zuiderveld 1994), for our herbaceous imagery in an attempt to enhance the edges of grass blades and crop stalks and the boundary between earth and herb.

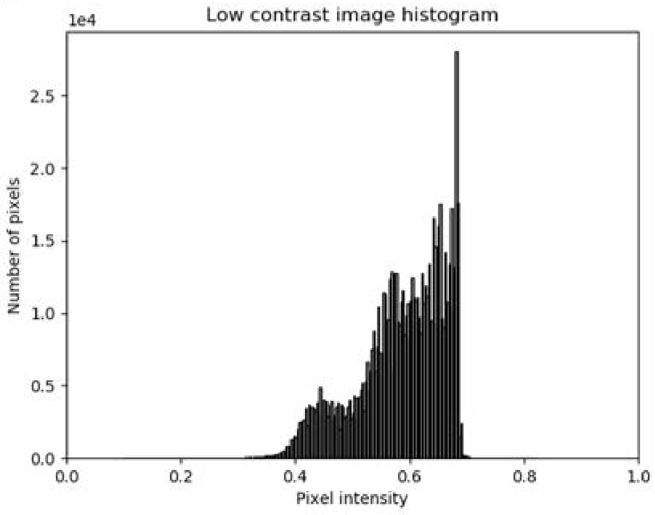
Histogram equalization (Russ 1994) is an image enhancement technique that modifies image intensities to enhance global contrast. This is especially useful when an image maintains an intensity histogram with few distinct peaks, thereby signifying a high density of pixels with intensities in narrow ranges. Figure 10 illustrates an intensity histogram for a low contrast bare-dominant image.

Figure 10. Demonstration of an image's pixel intensity histogram: low contrast bare-dominant image (*a*) and corresponding image histogram (*b*).

(a)



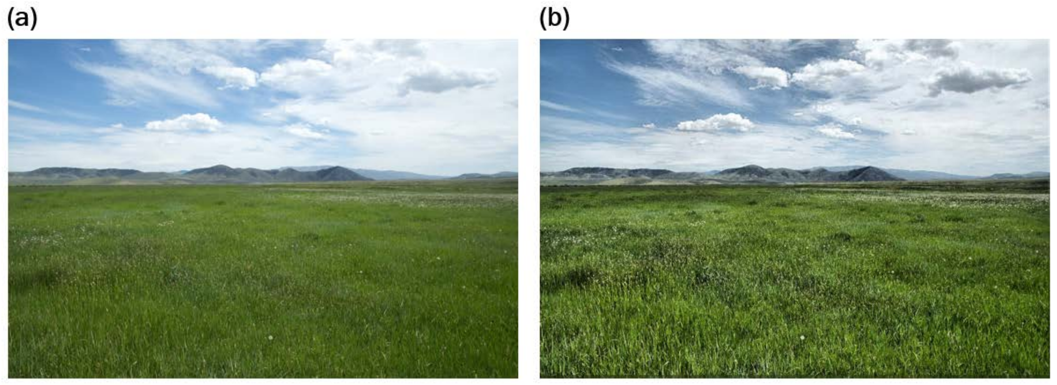
(b)



In Figure 10(b), we can easily see that some intensity values are underutilized, namely those below 0.4 and above 0.7. Histogram equalization works to equalize pixel frequency across all possible intensity levels by spreading out intensity values within peaks and condensing them in the valleys. The intensity distribution achieves greater uniformity when each of the original pixel intensities is transformed to new values via the image's cumulative histogram.

Adaptive Histogram Equalization (AHE) applies the same principles as histogram equalization to several segments of an image, thereby enhancing local contrast much like CLAHE. However, CLAHE is the preferred method as AHE can often produce artificial contrast variation in homogeneous regions of an image. For this particular project, we leveraged scikit-image (van der Walt et al. 2014) to apply CLAHE to our herb-dominant images. We subsequently ran the resulting contrast-enhanced images through the pre-trained DilatedNet to assess whether we could increase the accuracy of predictions. Figure 11 demonstrates the effect of applying CLAHE on a ground truth herbaceous image. In Figure 11(b), the CLAHE technique effectively enhances local contrast of the grass blades in the foreground and local detail of the patch of white flowers off to the far right without introducing pronounced artifacts in uniform regions of the image.

Figure 11. Effect of applying CLAHE to a ground truth herbaceous image: (*a*) original herb-dominant image and (*b*) resulting image after applying CLAHE.



It is important to note that all of the techniques mentioned above typically operate on grayscale images. However, scikit-image's CLAHE implementation provides default functionality whereby if the function is presented with an RGB image, it will convert the image to the HSV (Hue, Saturation, Value) color space and perform CLAHE on the value channel. Once the algorithm completes, the image is converted back to the RGB color space.

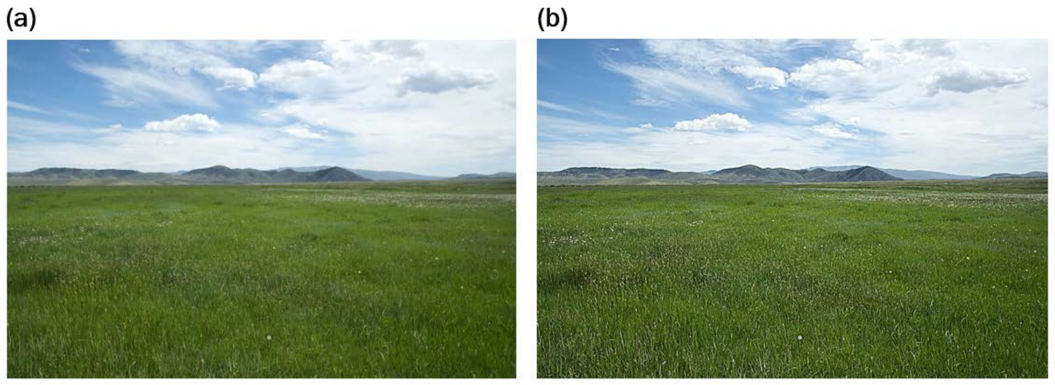
4.5.2 Unsharp masking

In addition to enhancing local contrast of herb-dominant images, we wanted to explore whether image sharpening techniques, like unsharp masking, could emphasize texture while enhancing local detail. The key to unsharp masking is the creation of a high-pass filter, or unsharp mask. In terms of implementation, first a Gaussian filter is used to blur the original image. The blurred image is then subtracted from the original image to create the unsharp mask. The value of blur-size, or σ, applied to the Gaussian filter dictates the amount of blur; and for landscape images, Weinmann and Lourekas (2014) recommend a σ between 1.0 and 1.5.

Once an unsharp mask is generated, it is multiplied by a value called *the amount*, which is typically represented by a percentage. The amount dictates the level of contrast added at edges. Recommended values for the amount for landscape images range from 100% to 150% (Weinmann and Lourekas 2014).

To determine whether this technique had any effect on increasing the percentage of images with herb-dominant ground truth labels as having a dominant class of either grass or field, parameters on either side of the suggested ranges were tested for both the blur-size, $\sigma \in \{1.0, 1.5\}$, and the amount, $a \in \{100, 150\}$. Figure 12 illustrates an herb-dominant image after applying the unsharp masking technique with a blur-size of 1.5 and an amount of 150%. The effects of unsharp masking with the suggested parameters both at the upper extremes are quite subtle. Enhanced local contrast within the field at the foreground is present; however, the effects are more subdued compared to the same image post-CLAHE (Figure 11[b]). Section 5.2.3 provides an assessment of unsharp masking on model performance.

Figure 12. Effect of applying unsharp masking to a ground truth herbaceous image with blur-size, σ, set to 1.5 and amount set to 150%: (*a*) original herb-dominant image and (*b*) resulting image after applying unsharp masking.



5 Identification of Dominant Vegetation Type via a Pre-Trained DilatedNet

5.1 DilatedNet

Initially, we wanted to assess whether the trained DilatedNet presented in Zhou et al. (2017) could sufficiently output a dominant (i.e., largest pixel ratio) SceneParse150 class of interest with our ground truth images. Recall that the following set contains the classes of interest from the SceneParse150 dataset: dirt track, earth, field, grass, palm, plant, rock, sand, and tree.

The GNBID comprises 178 bare-, 411 herb-, 214 shrub-, and 353 tree-dominant labeled images. Table 3 illustrates baseline results after we passed our unaltered images through the pre-trained network. It is important to note that the columns correspond to only those SceneParse150 classes of interest. The implications for this are as follows: some images may report the largest ratio of pixels allocated to the SceneParse150 class, "sky," for example. Nevertheless, because "sky" is not in our set of classes of interest, we do not report it as "dominant" in any of the tables or figures presented here.

Table 3. Ground truth and DilatedNet classification comparison. Classes listed under "Actual Class" correspond to ground truth classifications while classes listed under "Predicted Class" correspond to DilatedNet classifications.

	Predicted Class								
Actual Class	Dirt Track	Earth	Field	Grass	Palm	Plant	Rock	Sand	Tree
Herb	2	123	155	96	1	13	1	4	16
Tree	0	27	7	8	2	3	0	1	305
Shrub	3	89	30	6	1	8	4	4	69
Bare	2	91	6	0	0	0	10	66	3

Images with bare-dominant ground truth labels maintained the largest ratio of pixels allocated to the earth (91 images, 51.1% of all bare-dominant images) or sand (66 images, 37.1% of all bare-dominant images) categories. If both earth and sand SceneParse150 classes are mapped to the bare-dominant ground truth category (as demonstrated in Table 2), then the DilatedNet correctly classifies 88.2% of all bare-dominant images.

As illustrated in Table 3, images with an herb-dominant ground truth label maintain the largest ratio of pixels allocated to either the field (155 images, 37.7% of all herb-dominant images), earth (123 images, 29.9% of all herb-dominant images), or grass (96 images, 23.4% of all herb-dominant images) SceneParse150 categories. In an attempt to increase the ratio of field or grass pixels for a single image and thereby increase the total percentage of herb-dominant classifications by the pre-trained network, we investigate in section 5.2 whether contrast enhancement or image sharpening techniques briefed in sections 4.5.1 and 4.5.2 aide in the delineation of grass blade or earth–vegetation boundaries.

For shrub-dominant ground truth images, the DilatedNet classified 89 images, or 41.6% of images, as earth; 69 images, or 32.2% of images, as tree; and 30 images, or 14.0% of images, as field. In our original mapping of SceneParse150 classes to ground truth classes, we had initially speculated that the SceneParse150 class, plant, would map in a straightforward manner to the ground truth class, shrub. However, as evidenced here, a disparate assortment of SceneParse150 classes including earth, field, and tree, to name a few, compose the shrub output over all shrub-dominant images, possibly as a consequence of many factors. More specifically, the pre-trained model has difficultly discerning a shrub from the earth in cases where sparse shrub vegetation is present in the foreground of the image. Furthermore, in many shrub-dominant images, shrub vegetation in the background is often mistaken by the model as field, likely due to the appearance of a high density of the shrub vegetation. Finally, scale is likely a contributing factor to the large percentage of shrub-dominant images labeled as tree.

Lastly, for images with predominantly tree-type vegetation (tree-dominant ground truth label), the DilatedNet correctly classified 87.0% of images as tree- or palm-dominant.

5.2 Contrast enhancement and sharpening

5.2.1 Motivation

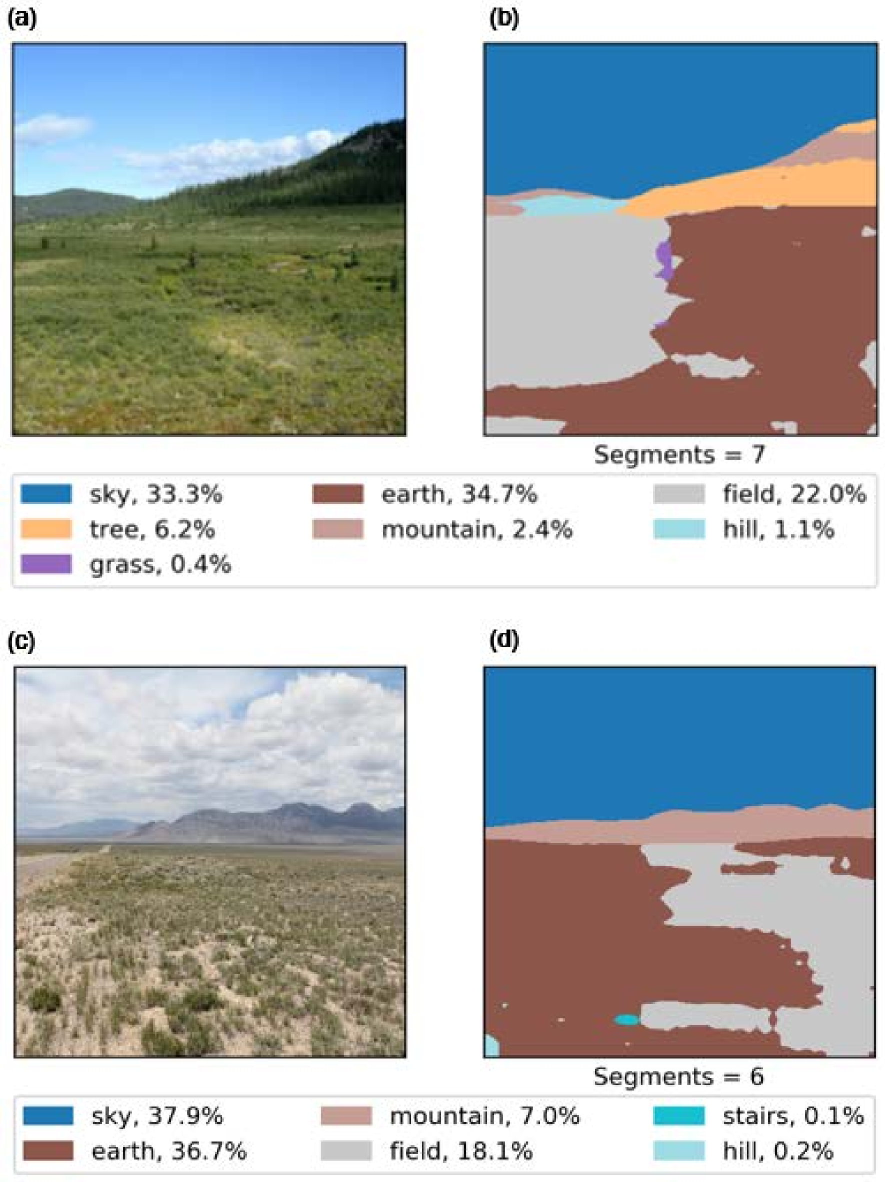
As evidenced in the previous section, the pre-trained DilatedNet performed poorly on both herb- and shrub-dominant images. More specifically, our ground truth labels did not align with our perceived mapping to classes in the SceneParse150 dataset. For example, we originally mapped

SceneParse150 classes field and grass to our ground truth class herb (Table 2). However, we find that the earth category maintains the second largest percentage of images (123 images, 29.9% of all herb-dominant images). We wanted to initially test whether contrast enhancement or sharpening techniques could improve the earth/herb margin or provide more local detail to grass blades and crop stalks to ultimately decrease the number of pixels allocated to the earth class and increase those allocated to the field or grass class within each image. While shrub-dominant images suffer from a similar ambiguous mapping, we wanted to investigate the efficacy of these techniques on herb-dominant images first since the resulting mapping for shrub-dominant images is even more unclear.

Figure 13 further motivates our case for exploring the techniques mentioned in 4.5.1 and 4.5.2. In Figure 13, the segmentation results (b) and (d) for the original images (a) and (c) indicate that for both images, the earth class maintains the largest number of pixels (not including sky). Taking a closer look at the segmentation results for Figure 13(a), we can see that while pixels in the left-hand portion of the foreground are classified as field, pixels in the right-hand portion of the foreground are classified as earth. Less one small oval-shaped section of the foreground in the center of the image, the texture and color appear to remain uniform across the foreground, and so one might expect the assignment of the majority of pixels in the foreground to the field class. The segmentation results for Figure 13(c) illustrate a similar effect, but we also observe that the network does not easily parse individual tufts of herbaceous vegetation surrounded at the base by earth or sand. The DilatedNet assigns those pixels associated with tufts to the surrounding earth category as illustrated in Figure 13(d).

After observing these initial results, we postulate that enhancement or sharpening techniques may more clearly allow the network to “see” the distinction between herb and earth.

Figure 13. Ground truth herb-dominant sample images (*a*) and (*c*) and their corresponding segmentation via the DilatedNet (*b*) and (*d*).



5.2.2 Sample segmentation results after CLAHE and unsharp masking

Prior to running the entire batch of herb-dominant CLAHE and unsharp masked images through the network, we first examined a few sample segmentations to determine if the techniques were a worthwhile pursuit.

Figure 14 illustrates the segmentation results of two CLAHE-modified, herb-dominant images. Figure 14(a) and (c) illustrate the effects of CLAHE on Figure 13(a) and (c). After running these enhanced sample images through the network, we found that CLAHE increased the percentage of pixels allocated to field from 22.0% presented in Figure 13(b) to 57.5% in Figure 14(b). Additionally, applying CLAHE increased the percentage of pixels allocated to field from 18.1% in Figure 13(d) to 20.3% in Figure 14(d). However, the dominant class for Figure 14(d) is nevertheless earth. The results presented here along with several other sample images demonstrate that applying CLAHE to our images could aid in increasing the proportion of pixels assigned to field or grass and decreasing the proportion assigned to earth.

Figure 14. CLAHE-enhanced ground truth herb-dominant sample images (*a*) and (*c*) and their corresponding segmentations via the DilatedNet (*b*) and (*d*).

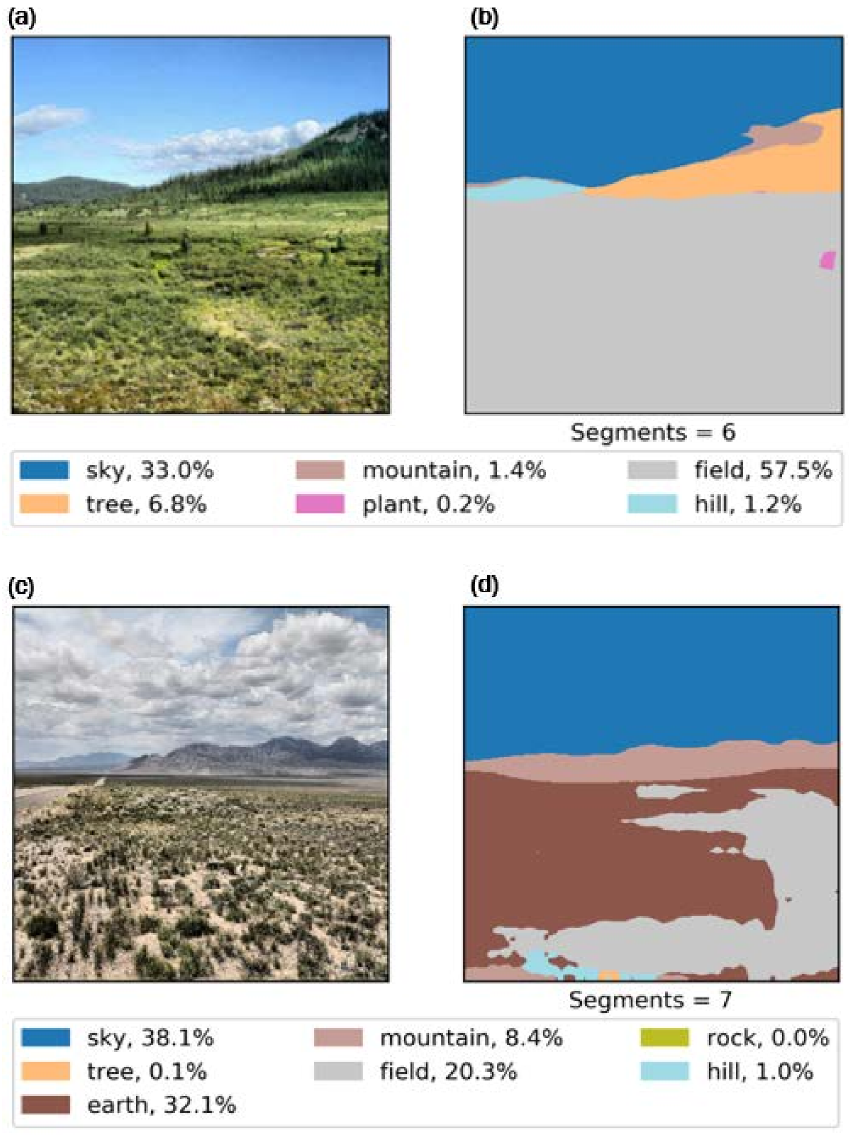
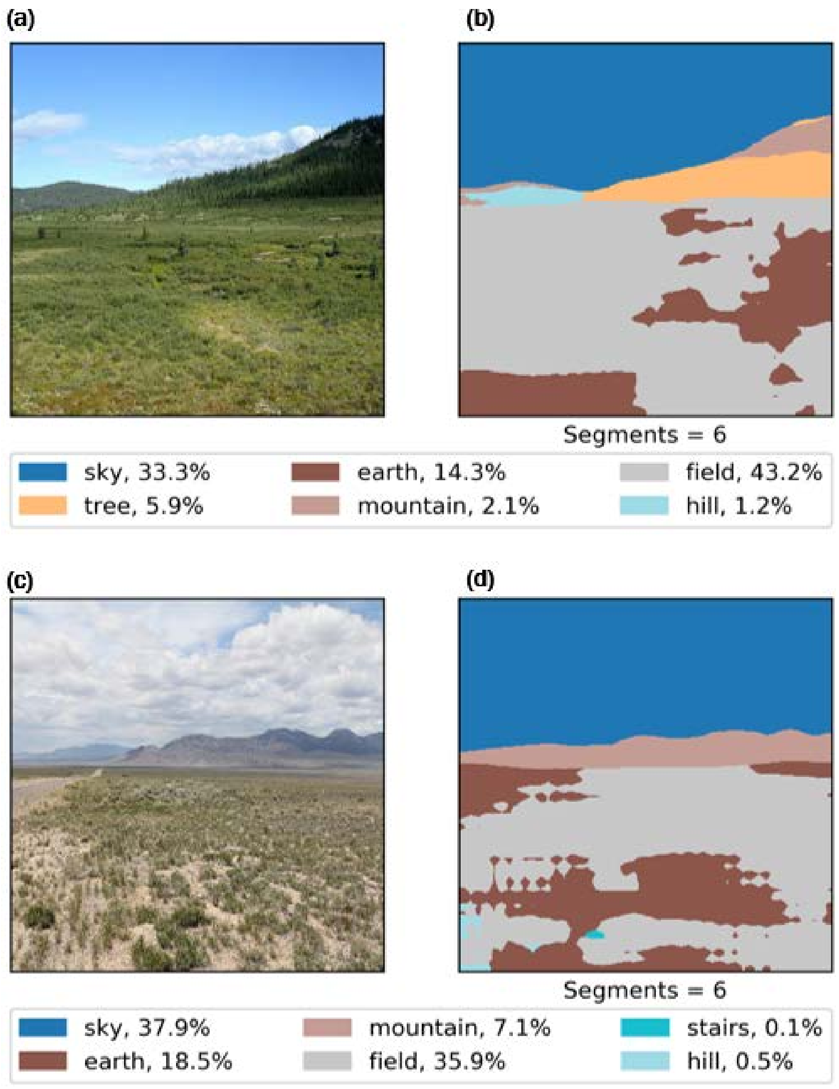


Figure 15 illustrates the segmentation results (b) and (d) of two herb-dominant images (a) and (c) after unsharp masking. Compared to the segmentation of the original image in Figure 13(a), the segmentation of Figure 15(a) increases the percentage of field pixels from 22.0% in Figure 13(b) to 43.2% in Figure 15(b). By applying the unsharp masking technique as seen in Figure 15(c), the percentage of pixels allocated to the field category increased from 18.1% in Figure 13(d) to 35.9% in Figure 15(d). Additionally, the unsharp masking technique appears to have attempted to parse out the boundary between earth and herb where individual tufts of vegetation are present (Figure 15[d], left portion of the image). In both cases, unsharp masking allows the field category to supersede the earth category as the dominant SceneParse150 classification.

Figure 15. Sharpened ground truth herb-dominant sample images (*a*) and (*c*) and their corresponding segmentation via the DilatedNet (*b*) and (*d*).



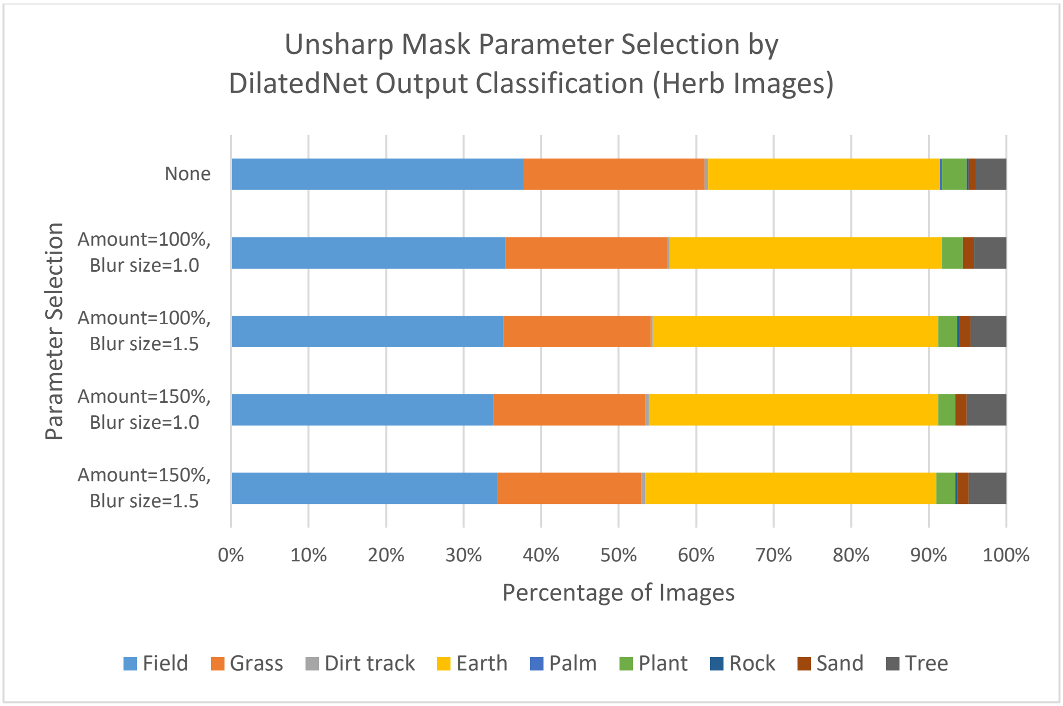
These two herb-dominant images demonstrate that it is feasible to increase the proportion of pixels from earth to field or grass by applying either image enhancement or sharpening prior to passing the image through the network.

5.2.3 Unsharp masking parameter selection

We tested a combination of parameters (amount and blur-size) for unsharp masking. For each combination, we passed the resulting herb-dominant sharpened images through the DilatedNet.

Figure 16 shows the resulting distribution of SceneParse150 dominant classes for each combination. As illustrated by Figure 16, unmask sharpening, irrespective of the parameter selection, increases the proportion of ground truth herb-dominant images labeled as earth dominant as compared to no parameter selection (i.e., no unsharp masking applied) despite the promising results seen in our sample images (Figure 15).

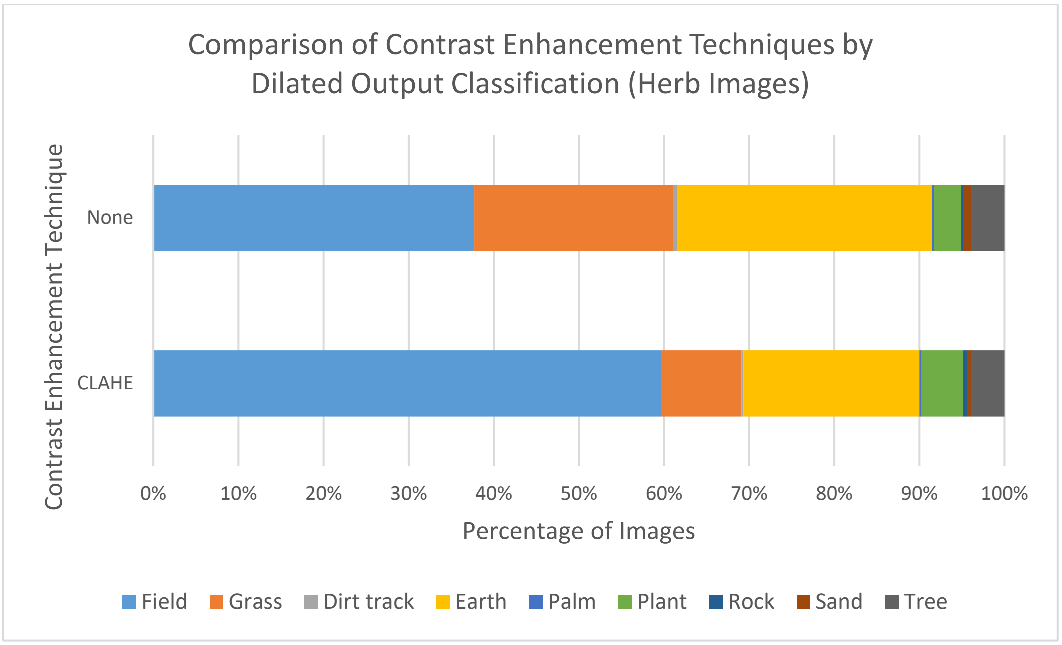
Figure 16. Distribution of DilatedNet output classifications for various parameter selections associated with the unsharp masking technique.



5.2.4 CLAHE results

We applied CLAHE to the full batch of herb-dominant images. As illustrated in Figure 17, applying CLAHE to herbaceous imagery increased the number of images with a dominant SceneParse150 category of field from 155 images (no enhancement) to 245 images (CLAHE enhanced). We did observe a decrease in the number of images with a dominant SceneParse150 category of grass from 96 images (no enhancement) to 39 images (CLAHE enhanced); however, the vast majority images with a modified label saw a change to field as the dominant class. CLAHE enhancement also decreased the number of images with a dominant SceneParse150 category of earth from 123 images (no enhancement) to 85 images (CLAHE enhanced). Therefore, we can conclude that, at least for herb-dominant images, applying CLAHE to images prior to passing through Zhou et al.'s (2017) DilatedNet helps to clarify the mapping between GNBID ground truth labels and SceneParse150 categories; however, this mapping is not absolute since herb-dominant ground truth images with an earth-dominant classification are still present.

Figure 17. Distribution of DilatedNet output classifications for original and CLAHE-modified images.



6 Concluding Remarks and Recommendations for Future Work

This report outlines the proof-of-concept study performed to explore the feasibility of using machine learning to expand global coverage in the GNBID. We presented an investigation of unsupervised versus supervised image segmentation techniques to achieve this goal. After an initial review of several unsupervised techniques, we chose a supervised technique: the DilatedNet, a deep convolutional neural network, trained on the SceneParse150 dataset.

As evidenced in the results section, we encountered limitations using the pre-trained DialatedNet. The mapping from our GNBID ground truth labels to one of the SceneParse150 classes was not always straightforward, especially for herb- and shrub-dominant imagery. To resolve these issues, we recommend applying transfer learning.

High similarity between the SceneParse150 training images and the ground truth vegetation images available within the GNBID suggests that transfer learning is a potentially promising approach for automated labeling of vegetation within a terrestrial photo. To implement this approach, the DilatedNet is copied, either in its entirety or partially, to a new network. The new network is then trained to semantically segment images according to vegetation type. (This is in contrast to the pre-trained DilatedNet explored in this study, which was trained on 150 classes.) Two approaches exist for the training phase when transfer learning is pursued (Yosinski et al., 2014): either the weights of the copied network are frozen (i.e., not changed) or fine-tuned. Past work has shown that transfer learning is useful for both semantically segmenting off-road scenes (Holder et al. 2016) and computer-aided detection of medical imagery (Shin et al. 2016).

Our study found that the DilatedNet performed well for bare- and tree-dominant images but struggled with herbaceous- and shrub-dominant images. The application of CLAHE on herb-dominant photos enhanced local contrast and edge delineation. After running the enhanced photos through the network, we observed a 58% increase in the number of images with a dominant SceneParse150 category of field and a 31% decrease in the number of images with a dominant SceneParse150 category of earth, which suggests that performing CLAHE to on-the-ground photos prior to classification may yield more accurate results. Additional investigation is necessary to determine whether CLAHE-derived images could improve the classification of shrub-dominant photos according to the mapping scheme provided in this study. We also recommend broadening the scope of classification to include more detail. For example, the GNBID contains more specific attributes beyond vegetation type, including vegetation height, leaf shape, leaf color, phenology (i.e., green, transition, or dormant), etc. The database would benefit from automated classification of these characteristics as well.

In summary, we found that the supervised classification approach is promising for this application. This study establishes the limitations and strengths of applying existing machine learning algorithms for classifying natural backgrounds in terrestrial imagery. Our results suggest that there is potential to improve image classification capabilities, which translates into enhanced research and development tools (e.g., the GNBID) and mission-planning decision-aid tools for the Army. We recommend further work to establish these methods for the purpose of reducing time, labor, and costs for research and development tools.

References

Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. 2012. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (11): 2274–2281.

Badrinarayanan, V., A. Kendall, and R. Cipolla. 2017. SegNet: A Deep Convolutionan Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (12): 2481–2495.

Cordts, M., M. Omran, S. Ramos, T. Rehfeld, M. Enweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 27–30 June, Las Vegas, Nevada, 3213–3223.

Di Gregorio, A., and L. J. M. Jansen. 2005. *Land Cover Classification System: Classification Concepts and User Manual*. Rome, Italy: United Nations, Food and Agriculture Organization.

Felzenszwalb, P. F., and D. P. Huttenlocher. 2004. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision* 59 (2): 167–181.

Gould, S., J. Rodgers, D. Cohen, G. Elidan, and D. Koller. 2008. Multi-Class Segmentation with Relative Location Prior. *International Journal of Computer Vision* 80 (3): 300–316.

Holder, C. J., T. P. Breckon, and X. Wei. 2016. From On-Road to Off: Transfer Learning Within a Deep Convolutional Neural Network for Segmentation and Classification of Off-Road Scenes. In *Proceedings, Computer Vision—ECCV 2016 Workshops, Part 1*, 8–10 and 15–16 October, Amsterdam, the Netherlands, ed. G. Hua and H. Jégou, 149–162.

Mottaghi, R., X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. 2014. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 23–28 June, Columbus, Ohio, 891–898.

Parker, N. E., and S. L. Jarvis. 2017. *Global Natural Background Image Database User Manual*. ERDC/CRREL SR-17-4. Hanover, NH: U.S. Army Engineer Research and Development Center.

Parker, N. E., S. L. Jarvis, M. L. Maxson, W. Breitkreutz, C. L. Scott, M. Ekegren, R. Detsch, R. A. Melloh, and S. McIntosh. 2017. *Development of the Global Natural Background Image Database for Camouflage Field Evaluation*. ERDC/CRREL TR-17-5. Hanover, NH: U.S. Army Engineer Research and Development Center.

Russ, J. C. 1994. *The Image Processing Handbook, Second Edition*. Boca Raton, Florida: CRC Press.

Shelhamer, E., J. Long, and T. Darrel. 2017. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (4): 640–651.

Shin, H. C., H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. 2016. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging* 35 (5): 1285–1298.

van der Walt, S., J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors. 2014. Scikit-Image: Image Processing in Python. *PeerJ* 2:e453. http://dx.doi.org/10.7717/peerj.453.

Vedaldi, A., and S. Soatto. 2008. Quick Shift and Kernel Methods for Mode Seeking. In *European Conference on Computer Vision*, Part IV, 12–18 October, Marseille, France, ed. D. Forsyth, P. Torr, and A. Zisserman, 705–718.

Weinmann, E., and P. Lourekas. 2014. *Photoshop CC: Visual QuickStart Guide*. San Francisco, California: Peachpit Press.

Xiao, J., J. Hays, K. Ehinger, A. Oliva, and A. Torralba. 2010. SUN Database: Large-scale Scene Recognition from Abbey to Zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 13–18 June, San Francisco, California, 3485–3492*.

Yosinski, J., J. Clune, Y. Bengio, and H. Lipson. 2014. How Transferable are Features in Deep Neural Networks? In *Advances in Neural Information Processing Systems*, ed. Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, *27*: 3320–3328.

Yu, F., and V. Koltun. 2016. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proceedings of the International Conference for Learning Representations (ICLR), 2–4 May, San Juan, Puerto Rico*.

Zhou, B., H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. 2017. Scene Parsing through ADE20K Dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 21–26 July, Honolulu, Hawaii, *5122–5130*.

Zuiderveld, K. 1994. Contrast Limited Adaptive Histograph Equalization. In *Graphic Gems IV*, 474–485. San Diego: Academic Press Professional.

Appendix A: Installation of PyCaffe

Installation of PyCaffe, a python interface to Caffe (http://caffe.berkeleyvision.org/) models, is required to use the dilated neural network developed by Zhou et al. (2017). The python interface enables the use of Caffe models for testing and evaluation.

A specific source code distribution of PyCaffe for Windows is provided at https://github.com/BVLC/caffe/tree/windows. Several notes on how to build the PyCaffe interface from source code are documented by the author of the Windows distribution; however, we encountered several errors during the build process. The following procedure successfully builds Caffe and PyCaffe for a Windows 10 machine with Python 3.5:

1. Install CMake (https://cmake.org/).
2. Install git for Windows (http://gitforwindows.org/).
3. Install Visual C++ 2015 build tools (http://landinghub.visualstudio.com/visual-cpp-build-tools) by running a custom install to include the Windows 10 SDK.
4. Install CUDA 8.0 (https://developer.nvidia.com/cuda-80-ga2-download-archive) and cuDNN 5.0 (https://developer.nvidia.com/rdp/assets/cudnn-8.0-windows10-x64-v5.0-ga-zip).
5. Install the latest Anaconda distribution (https://www.anaconda.com/download/).
6. Add cmake.exe, python.exe, conda.exe, and git.exe to the system PATH variable.
7. Clone the git repository to a project directory:

```
git clone https://github.com/BVLC/caffe.git
cd caffe
git checkout windows
```

8. Modify "build_win.cmd" in caffe/scripts. Under the first outer else statement, ensure the following are set:

```
MSVC_VERSION=14
WITH_NINJA = 0
```

```
CPU_ONLY=1
CUDA_ARCH_NAME=Kepler
CMAKE_CONFIG=Release
USE_NCCL=0
PYTHON_VERSION=3
BUILD_PYTHON=0
BUILD_PYTHON_LAYER=1
RUN_TESTS=1
RUN_LINT=0
RUN_INSTALL=1
```

Add to the cmake command the following:

```
-DCMAKE_C_COMPILER="C:/Program Files (x86)/Microsoft Visual
Studio 14.0/VC/bin/cl.exe" ^
-DCMAKE_CXX_COMPILER="C:/Program Files (x86)/Microsoft Visual
Studio 14.0/VC/bin/cl.exe" ^
-DCUDNN_ROOT="C:/Program Files/NVIDIA GPU Computing
Toolkit/CUDA/v8.0/cuda" ^
```

9. Modify "WindowsDownloadPrebuiltDependencies.cmake" in caffe/cmake. Under `if(USE_PREBUILT_DEPENDENCIES)`, change `_pyver 27 to _pyver 35`.
10. Modify "CMakeLists.txt" in caffe/ by changing `set(python_version "2" ... to set(python_version "3" ...`
11. Modify "Dependencies.cmake" in caffe/cmake. Add an `else` condition prior to the final `endif()` statement in the Python block.

```
else()
  if(NOT "${python_version}" VERSION_LESS "3.0.0")
    # use python3
    find_package(PythonInterp)
    find_package(PythonLibs)
    find_package(NumPy)
    # Find the matching boost python implementation
    set(version ${PYTHONLIBS_VERSION_STRING})

    STRING( REGEX REPLACE "[^0-9]" "" boost_py_version ${ver-
sion} )
    find_package(Boost 1.46 COMPONENTS "python-
py${boost_py_version}")
```

```
    set(Boost_PYTHON_FOUND ${Boost_PYTHON-PY${boost_py_ver-
sion}_FOUND})

    while(NOT "${version}" STREQUAL "" AND NOT Boost_PY-
THON_FOUND)
      STRING( REGEX REPLACE "([0-9.]+).[0-9]+" "\\1" version
${version} )

      STRING( REGEX REPLACE "[^0-9]" "" boost_py_version
${version} )
      find_package(Boost 1.46 COMPONENTS "python-
py${boost_py_version}")
      set(Boost_PYTHON_FOUND ${Boost_PYTHON-PY${boost_py_ver-
sion}_FOUND})

      STRING( REGEX MATCHALL "([0-9.]+).[0-9]+" has_more_ver-
sion ${version} )
      if("${has_more_version}" STREQUAL "")
        break()
      endif()
    endwhile()
    if(NOT Boost_PYTHON_FOUND)
      find_package(Boost 1.46 COMPONENTS python)
    endif()
  endif()
  if(NOT Boost_PYTHON_FOUND)
    find_package(Boost 1.46 COMPONENTS python)
  endif()
  if(PYTHONLIBS_FOUND AND NUMPY_FOUND AND Boost_PYTHON_FOUND)
    set(HAVE_PYTHON TRUE)
    message(STATUS "Set HAVE_PYTHON is TRUE...")
    if(Boost_USE_STATIC_LIBS AND MSVC)
      list(APPEND Caffe_DEFINITIONS PUBLIC -DBOOST_PY-
THON_STATIC_LIB)
    endif()
    if(BUILD_python_layer)
      list(APPEND Caffe_DEFINITIONS PRIVATE -DWITH_PY-
THON_LAYER)
      list(APPEND Caffe_INCLUDE_DIRS PRIVATE ${PYTHON_IN-
CLUDE_DIRS} ${NUMPY_INCLUDE_DIR} PUBLIC ${Boost_INCLUDE_DIRS})
```

```
      list(APPEND Caffe_LINKER_LIBS PRIVATE ${PYTHON_LIBRAR-
IES} PUBLIC ${Boost_LIBRARIES})
    endif()
  else()
    message(STATUS "Set HAVE_PYTHON is FALSE, since PY-
THONLIBS_FOUND is ${PYTHONLIBS_FOUND}, NUMPY_FOUND is
${NUMPY_FOUND}, and Boost_PYTHON_FOUND is ${Boost_PY-
THON_FOUND}...")
    find_package(PythonLibs)
      message(STATUS "python_version:STR=${python_version}")
      message(STATUS "PYTHONLIBS_FOUND:BOOL=${PY-
THONLIBS_FOUND}")
      message(STATUS "PYTHON_LIBRARIES:PATH=${PYTHON_LIBRAR-
IES}")
      message(STATUS "PYTHON_INCLUDE_DIRS:PATH=${PYTHON_IN-
CLUDE_DIRS}")
      message(STATUS "PYTHONLIBS_VERSION_STRING:STR=${PY-
THONLIBS_VERSION_STRING}")
    find_package(PythonInterp)
      message(STATUS "PYTHONINTERP_FOUND:BOOL=${PYTHONIN-
TERP_FOUND}")
      message(STATUS "PYTHON_EXECUTABLE:PATH=${PYTHON_EXECUTA-
BLE}")
  endif()
```

12. Create conda environment with Python 3.5, and activate the new environment:

```
conda create -n run-pycaffe python=3.5 anaconda
activate run-pycaffe
```

13. Add the following conda channels and install the following packages (note the anaconda install already includes numpy, scipy, scikit-image, and pyyaml):

```
conda config --add channels conda-forge
conda config --add channels willyd
conda install --yes cmake ninja protobuf==3.1.0 pydotplus
graphviz
```

14. Run “build_win.cmd” in the caffe/ directory as “scripts/build_win.cmd”. This will build Caffe and PyCaffe.

15. Add PyCaffe (i.e., \path\to\caffe\python\caffe) to the PYTHONPATH system variable within a python script.

Appendix B: Code listings

Code developed in Python 3.5 (for the dilated neural network) and 3.6 (for all other image segmentation algorithms) generated the image segmentation results in section 3. Additionally, code written for contrast enhancement and image sharpening was developed in Python 3.6. The code referenced here generated image results seen in sections 4 and 5.

The following sections present the image segmentation, contrast enhancement, and image sharpening code listings.

B.1 Felzenszwalb and Huttenlocher's (2004) algorithm

```
"""
Explore image segmentation by Felzenszwalb and Huttenlocher's
(2004)
algorithm.

Reference:
    Felzenszwalb, P. F. and D. P. Huttenlocher. 2004. Efficient
Graph-Based
        Image Segmentation. International Journal of Computer Vi-
sion 59 (2):
        167 - 181.

@author: Carl R. Hart
"""

import os

import numpy
from matplotlib import pyplot
np = numpy
plt = pyplot

import skimage.io
import skimage.color
import skimage.filters
import skimage.filters.rank
import skimage.morphology
import skimage.segmentation
import skimage.transform

from skimage import img_as_float

# Get directories and listing of photos
workDir = os.getcwd()
dataDir = '..\\data\\photos-exemplars'
dirList = os.listdir(dataDir)

dirSep = '\\'

# Keep only jpgs in dataDir list
nFile = len(dirList)

for iFile in range(nFile):
    if not( dirList[iFile].endswith('jpg') ):
        dirList.pop(iFile)

#%% Save plot module

def savePlot(img, segMask, imgName, segTitle, nSeg, savePath):

    (f, ax) = plt.subplots(1, 2)

    ax[0].imshow(img)
    ax[1].imshow(segMask, cmap = 'tab20')

    ax[0].set_xticks([])
    ax[0].set_yticks([])
    ax[1].set_xticks([])
    ax[1].set_yticks([])

    ax[0].set_title(imgName)

    ax[1].set_title(segTitle)
    ax[1].set_xlabel('Segments = ' + str(nSeg))

    f.savefig(savePath, dpi = 300, format = 'png')
    plt.close()

#%% Segment image by Felsenszwalb’s graph based image segmenta-
tion method

nImg = len(dirList)

felsScale = [100, 1000, 10000]
felsSigma = [0.1, 1, 10]

nScale = len(felsScale)
nSigma = len(felsSigma)

for iImg in range(nImg):
    # Load image, and resize to 384 x 384 pixels
    img = skimage.io.imread( dirSep.join([workDir, dataDir,
dirList[iImg]]) )
    img = img_as_float(img)
    img = skimage.transform.resize(img, (384, 384))

    for iScale in range(nScale):
        for iSigma in range(nSigma):
            # Segment image by Felsenszwalb’s graph-based image
segmentation
            segMaskFels = skimage.segmentation.felzenszwalb(img,
                        scale = felsScale[iScale], sigma =
felsSigma[iSigma])

            imgName = dirList[iImg][0:dirList[iImg].find('_')]
            segTitle = ('Felsenszwalb,\n' + 'scale = ' +
str(felsScale[iScale])
                + ', $\sigma$ = ' + str(felsSigma[iSigma]))
            nSeg = segMaskFels.max() + 1
            segFileName = (imgName + '_seg-Fels_scale' +
str(felsScale[iScale])
                + '_sigma' + str(felsSigma[iSigma]) + '.png')
            savePath = dirSep.join([workDir, 'plot', 'segmenta-
tion', 'fels',
                                        segFileName])
            savePlot(img, segMaskFels, imgName, segTitle, nSeg,
savePath)
```

B.2 The quick-shift algorithm (Vedaldi and Soatto 2008)

```
"""
Explore image segmentation by the quick shift algorithm (Vedaldi
and Soatto,
2008) algorithm.

Reference:
    Vedaldi, A. and S. Soatto. 2008. Quick Shift and Kernel Meth-
ods for Mode
        Seeking in D. Forsyth, P. Torr, and A. Zisserman (Eds.):
ECCV 2008,
        Part IV, LNCS 5305: 705 - 718.

@author: Carl R. Hart
"""

import os

import numpy
from matplotlib import pyplot
np = numpy
plt = pyplot

import skimage.io
import skimage.color
import skimage.filters
import skimage.filters.rank
import skimage.morphology
import skimage.segmentation
import skimage.transform

from skimage import img_as_float

# Get directories and listing of photos
workDir = os.getcwd()
dataDir = '..\\data\\photos-exemplars'
```

```
dirList = os.listdir(dataDir)

dirSep = '\\'

# Keep only jpgs in dataDir list
nFile = len(dirList)

for iFile in range(nFile):
    if not( dirList[iFile].endswith('jpg') ):
        dirList.pop(iFile)

#%% Save plot module

def savePlot(img, segMask, imgName, segTitle, nSeg, savePath):

    (f, ax) = plt.subplots(1, 2)

    ax[0].imshow(img)
    ax[1].imshow(segMask, cmap = 'tab20')

    ax[0].set_xticks([])
    ax[0].set_yticks([])
    ax[1].set_xticks([])
    ax[1].set_yticks([])

    ax[0].set_title(imgName)

    ax[1].set_title(segTitle)
    ax[1].set_xlabel('Segments = ' + str(nSeg))

    f.savefig(savePath, dpi = 300, format = 'png')

    plt.close()

#%% Segment image by quickshift clustering in Color-(x,y) space

nImg = len(dirList)

qsRatio = [0, 0.1, 1.0]
qsKernelSize = [10, 30]
```

```
qsMaxDist = [10, 30]
qsSigma = [1]

nRatio = len(qsRatio)
nKernel = len(qsKernelSize)
nDist = len(qsMaxDist)
nSigma = len(qsSigma)

for iImg in range(nImg):
    # Load image, and resize to 384 x 384 pixels
    img = skimage.io.imread( dirSep.join([workDir, dataDir,
dirList[iImg]]) )
    img = img_as_float(img)
    img = skimage.transform.resize(img, (384, 384))

    for iRatio in range(nRatio):
        for iKernel in range(nKernel):
            for iDist in range(nDist):
                for iSigma in range(nSigma):
                    # Segment image by quickshift method
                    segMaskQS = skimage.segmentation.quick-
shift(img,
                        ratio = qsRatio[iRatio],
                        kernel_size = qsKernelSize[iKernel],
                        max_dist = qsMaxDist[iDist],
                        sigma = qsSigma[iSigma], convert2lab =
True)

                    imgName =
dirList[iImg][0:dirList[iImg].find('_')]
                    segTitle = ('Quickshift,\n' + 'ratio = ' +
                                str(qsRatio[iRatio]) + ', kernel
= ' +
                                str(qsKernelSize[iKernel]) + ',
dist = ' +
                                str(qsMaxDist[iDist]) + ',
$\sigma$ = ' +
                                str(qsSigma[iSigma]))
                    nSeg = segMaskQS.max() + 1
                    segFileName = (imgName + '_seg-quickshift_ra-
tio' +
                                   str(qsRatio[iRatio]) + '_ker-
nel' +
                                   str(qsKernelSize[iKernel]) +
'_dist' +
                                   str(qsMaxDist[iDist]) + '_sig-
ma' +
                                   str(qsSigma[iSigma]) + '.png')
                    savePath = dirSep.join([workDir, 'plot',
'segmentation',
                                'qs-lab', segFileName])

                    savePlot(img, segMaskQS, imgName, segTitle,
nSeg,
                             savePath)
```

B.3 The SLIC algorithm (Achanta et al. 2012)

```
"""
Explore image segmentation by the SLIC algorithm (Achanta, et
al., 2012)

Reference:
    Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua, S.
Süsstrunk. 2012.
        Efficient Graph-Based Image Segmentation. IEEE Transac-
tions on Pattern
        Analysis and Machine Intelligence 34 (11): 2274 - 2281.

@author: Carl R. Hart
"""

import os

import numpy
from matplotlib import pyplot
np = numpy
plt = pyplot
```

```
import skimage.io
import skimage.color
import skimage.filters
import skimage.filters.rank
import skimage.morphology
import skimage.segmentation
import skimage.transform

from skimage import img_as_float

# Get directories and listing of photos
workDir = os.getcwd()
dataDir = '..\\data\\photos-exemplars'
dirList = os.listdir(dataDir)

dirSep = '\\'

# Keep only jpgs in dataDir list
nFile = len(dirList)

for iFile in range(nFile):
    if not( dirList[iFile].endswith('jpg') ):
        dirList.pop(iFile)

#%% Save plot module

def savePlot(img, segMask, imgName, segTitle, nSeg, savePath):

    (f, ax) = plt.subplots(1, 2)

    ax[0].imshow(img)
    ax[1].imshow(segMask, cmap = 'tab20')

    ax[0].set_xticks([])
    ax[0].set_yticks([])
    ax[1].set_xticks([])
    ax[1].set_yticks([])

    ax[0].set_title(imgName)
    ax[1].set_title(segTitle)
    ax[1].set_xlabel('Segments = ' + str(nSeg))

    f.savefig(savePath, dpi = 300, format = 'png')

    plt.close()

#%% Segment image by slic algorithm

nImg = len(dirList)

slicCompact = [1, 10, 100]
slicSeg = [10, 100, 1000]
slicSigma = [0.1, 1, 10]

nCompact = len(slicCompact)
nSlicSeg = len(slicSeg)
nSigma = len(slicSigma)

for iImg in range(nImg):
    # Load image, and resize to 384 x 384 pixels
    img = skimage.io.imread( dirSep.join([workDir, dataDir,
dirList[iImg]]) )
    img = img_as_float(img)
    img = skimage.transform.resize(img, (384, 384))

    for iCompact in range(nCompact):
        for iSeg in range(nSlicSeg):
            for iSigma in range(nSigma):
                # Segment image by slick method
                segMaskSLIC = skimage.segmentation.slic(img,
                    n_segments = slicSeg[iSeg],
                    compactness = slicCompact[iCompact],
                    sigma = slicSigma[iSigma],
                    multichannel = True, convert2lab = True)

                imgName =
dirList[iImg][0:dirList[iImg].find('_')]
                segTitle = ('SLIC,\n' + 'n-seg = ' +
                            str(slicSeg[iSeg]) + ', compact = ' +
                            str(slicCompact[iCompact]) + ',
$\sigma$ = ' +
                            str(slicSigma[iSigma]))
                nSeg = segMaskSLIC.max() + 1
                segFileName = (imgName + '_seg-slic_n-seg' +
                               str(slicSeg[iSeg]) + '_compact' +
                               str(slicCompact[iCompact]) +
'_sigma' +
                               str(slicSigma[iSigma]) + '.png')
                savePath = dirSep.join([workDir, 'plot', 'segmen-
tation',
                            'slic', segFileName])

                savePlot(img, segMaskSLIC, imgName, segTitle,
nSeg,
                         savePath)
```

B.4 The dilated neural network (Zhou et al. 2017)

```
"""
Explore image segmentation by a dilated neural network (Zhou, et
al., 2017).

Reference:
    Zhou, B., H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A.
Torralba. 2017.
        Scene Parsing through ADE20K Dataset. Proceedings of the
IEEE
        Conference on Computer Vision and Pattern Recognition,
5122-5130.

@author: Carl R. Hart
"""

#dataDir = [home, '\\documents\\research\\natick\\data\\photos-
exemplars'];

import os
import sys
import numpy
import matplotlib.patches as mpatches
from matplotlib import pyplot, colors
np = numpy
plt = pyplot

import skimage.io
import skimage.color
import skimage.filters
import skimage.filters.rank
import skimage.morphology
import skimage.segmentation
import skimage.transform

from skimage import img_as_float, img_as_ubyte

# Get directories and listing of photos
workDir = os.getcwd()
dataDir = '..\\data\\photos-exemplars'
dirList = os.listdir(dataDir)

dirSep = '\\'

# pyCaffe interface directory
pyCaffeDir = dirSep.join([workDir, 'caffe', 'python'])

# Add pyCaffe to system path
sys.path.insert(0, pyCaffeDir)

import caffe

# MIT segmentation model directory
mitSegDir = dirSep.join([workDir, 'segment-mit'])
modelDef = dirSep.join([mitSegDir, 'deploy_DilatedNet.prototxt'])
modelWeights = dirSep.join([mitSegDir, 'Dilated-
Net_iter_120000.caffemodel'])

# Load segmentation model based on a dilated network architecture
net = caffe.Net(modelDef, modelWeights, caffe.TEST)
```

```
# Keep only jpgs in dataDir list
nFile = len(dirList)

for iFile in range(nFile):
    if not( dirList[iFile].endswith('jpg') ):
        dirList.pop(iFile)

#%% Get segmentation labels for dilated neural network

def getLabels(pathToLabels):

    dirSep = '\\'

    labelsFile = dirSep.join([pathToLabels, 'objectInfo150.txt'])

    labels = []

    with open(labelsFile, 'r') as f:

        # Skip over header
        s = f.readline()
        s = f.readline()

        while len(s) > 0:
            iTab = s.rfind('\t')
            iComma = s[iTab + 1:].find(',')
            iNewLine = s[iTab + 1:].find('\n')

            if not( iComma == -1):
                labels.append(s[(iTab + 1):(iTab + iComma + 1)])
            else:
                labels.append(s[(iTab + 1):(iTab + iNewLine +
1)])

            s = f.readline()

    return labels

#%% Save dilated net segmentation plot
```

```
def saveDNetPlot(img, segMask, imgName, segTitle, nSeg, segLa-
bels, savePath):

    # Generate segmentation legend
    segId = np.unique(segMask)
    segId = np.asarray(segId, dtype = int)

    nPixel = len(segMask.flatten())

    segLegend = []

    for iSegId in range(len(segId)):
        nPixelInSeg = np.count_nonzero(segMask == segId[iSegId])

        segLegend.append((segLabels[segId[iSegId]] + ', ' +
                          '{:.1%}'.format(nPixelInSeg/nPixel)))

    # Generate plot
    (f, ax) = plt.subplots(1, 2)

    ax[0].imshow(img)
    segImg = ax[1].imshow(segMask, cmap = 'tab20', norm = col-
ors.LogNorm())

    ax[0].set_xticks([])
    ax[0].set_yticks([])
    ax[1].set_xticks([])
    ax[1].set_yticks([])

    ax[0].set_title(imgName)

    ax[1].set_title(segTitle)
    ax[1].set_xlabel('Segments = ' + str(nSeg))

    # Add legend to an image:
    # https://stackoverflow.com/questions/25482876/how-to-add-
legend-to-imshow-
    #   in-matplotlib

    # Get the colors of the segmentation ids, according to the
```

```
cmap used by
    # imshow
    segColors = [segImg.cmap(segImg.norm(value)) for value in
segId]
    # Create a patch (proxy artist) for every color
    patches = [mpatches.Patch(color = segColors[i], label = seg-
Legend[i])
                for i in range(len(segId))]
    # Put those patched as legend-handles into the legend
    ax[0].legend(handles = patches, bbox_to_anchor = (0, -0.6,
2.2, 0.5),
      ncol = 3, loc = 2, mode = 'expand', borderaxespad = 0)

    f.savefig(savePath, dpi = 300, format = 'png')

    plt.close()

#%% Segment image by a dilated neural network (Zhou et al., 2016,
2017)
# Image size requirements and mean of RGB channels are taken from
the m-file,
# https://github.com/CSAILVision/sceneparsing/blob/master/de-
moSegmentation.m

segLabels = getLabels(mitSegDir)

nImg = len(dirList)

sigma = [0.1, 1, 10]

nSigma = len(sigma)

for iImg in range(nImg):
    # Load image
    img = skimage.io.imread( dirSep.join([workDir, dataDir,
dirList[iImg]]) )
    img = img_as_float(img)

    # Resize image, caffe model requires an image in 384 x 384
pixels
```

```
    img = skimage.transform.resize(img, (384, 384))

    # Mean of red, green, and blue channels of the trained seg-
mentation dataset
    mu = np.array([109.5388, 118.6897, 124.6901])

    # Set up transformations to apply to image; subtract mean
from each color
    # channel, transpose dimensions of image from H x W x C
(height, width,
    # channel) to C x H x W, set scale of image from [0, 1] to
[0, 255], and
    # swap channel ordering from RGB to BGR
    transformer = caffe.io.Transformer({'data': (1, img.shape[2],
img.shape[0],
                                                 img.shape[1])})
    transformer.set_mean('data', mu)
    transformer.set_transpose('data', (2, 0, 1))
    transformer.set_raw_scale('data', 255)
    transformer.set_channel_swap('data', (2, 1, 0))

    for iSigma in range(nSigma):
        # Apply Gaussian filter to image
        img = skimage.filters.gaussian(img, sigma =
sigma[iSigma],
                                       multichannel = True)

        # Get image segmentation probabilities from dilated neu-
ral network
        # (better than conv-net, see Zhou et al., 2016)
        net.blobs['data'].data[...] = transformer.prepro-
cess('data', img)
        segProb = net.forward()

        # Network output is a dictionary, with a single key; get
values
        dictKey = list(segProb.keys())
        segProbVal = segProb[dictKey[0]]

        # Get indices of most likely class values; each index
```

```
corresponds to a
        # member of the "stuff" class
        segProbVal = np.squeeze(segProbVal)
        segMaskDN = np.argmax(segProbVal, axis = 0)
        segMaskDN = img_as_ubyte(segMaskDN) - 1

        imgName = dirList[iImg][0:dirList[iImg].find('_')]
        segTitle = ('DilatedNet, $\sigma$ = ' +
str(sigma[iSigma]))
        nSeg = len(np.unique(segMaskDN))
        segFileName = (imgName + '_seg-dilatednet' + '_sigma' +
                       str(sigma[iSigma]) + '.png')
        savePath = dirSep.join([workDir, 'plot', 'segmentation',
'dn',
                                segFileName])

        saveDNetPlot(img, segMaskDN, imgName, segTitle, nSeg,
segLabels,
                     savePath)
```

B.5 CLAHE

```
import os

import skimage.io as io
import skimage.exposure as ex

# Initialize data and results directories
workDir = os.getcwd()
dataDir = '.\\data\\photos-exemplars'
dirList = os.listdir(dataDir)
resultsDir = '.\\results\\Herb\\CLAHE'

nFile = len(dirList)
dirSep = '\\'

for iFile in range(nFile):
    fullPath = dirSep.join([dataDir, dirList[iFile]])
    img = io.imread(fullPath)
    equalizeLocal = ex.equalize_adapthist(img) # CLAHE
    path = dirSep.join([resultsDir, dirList[iFile]])
    io.imsave(path, equalizeLocal)
```

B.6 Unsharp Masking

```
import os

import skimage.io as io
import skimage.filters as filter
from skimage import img_as_float

import numpy as np

# Initialize data and results directories
workDir = os.getcwd()
dataDir = '.\\data\\photos-exemplars'
dirList = os.listdir(dataDir)
resultsDir = '.\\results\\Herb\\USM'

nFile = len(dirList)
dirSep = '\\'

for iFile in range(nFile):
    fullPath = dirSep.join([dataDir, dirList[iFile]])
    img = img_as_float(io.imread(fullPath))

    # Gaussian blur (blur-size = 1.5)
    blurred = filter.gaussian(img, 1.5, multichannel=True)
    sharper = np.clip(img + (img - blurred) * 1.5, 0, 1.0)

    path = dirSep.join([resultsDir, dirList[iFile]])
    io.imsave(path, sharper)
```

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD-MM-YYYY)* June 2018	**2. REPORT TYPE** Technical Report/Final	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Utility of Machine Learning Algorithms for Natural Background Photo Classification		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER U4355556
6. AUTHOR(S) Lauren E. Waldrop, Carl R. Hart, Nancy E. Parker, Chris L. Pettit, and Scotlund McIntosh		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Engineer Research and Development Center (ERDC) Cold Regions Research and Engineering Laboratory (CRREL) 72 Lyme Road Hanover, NH 03755-1290		**8. PERFORMING ORGANIZATION REPORT NUMBER** ERDC/CRREL TR-18-7
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Natick Soldier Research, Development, and Engineering Center Warfighter Directorate, Fiber and Textile Science Team (RDNS-SEW-TMF) 10 General Greene Avenue Natick, MA 01760		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		
13. SUPPLEMENTARY NOTES Terrain Characterization for Rendering and Field Evaluation		

14. ABSTRACT

In support of the Terrain Characterization for Rendering and Field Evaluation effort, the U.S. Army Corps of Engineers, Engineer Research and Development Center (ERDC), Cold Regions Research and Engineering Laboratory (CRREL), assisted the Natick Soldier Research, Development, and Engineering Center (NSRDEC) in evaluating machine learning algorithms to automatically classify three vegetation types (tree, shrub, and herbaceous), and a non-vegetated type in terrestrial images. In a previous partnership between CRREL and NSRDEC, researchers developed the Global Natural Background Image Database (GNBID), a collection of natural background images classified by vegetation attributes to include vegetation type and height, leaf shape, leaf color, and many others. Following deployment, the GNBID successfully improved on-the-ground understanding of natural background environments and quickly revealed the need for a larger database. Manual classification methods proved time intensive and variable, thus CRREL explored the feasibility of automatically identifying features using machine learning algorithms. In this scope-of-work study, we explore a multitude of computer vision techniques, settling on a supervised deep-learning technique. Here we present the advantages and disadvantages of various techniques, classification results from a subset of images, and recommendations for future research in this area.

15. SUBJECT TERMS
Camouflage (Military science), Computer algorithms, Computer vision, Machine learning, Remote sensing, Terrain characterization, Vegetation classification

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	69	**19b. TELEPHONE NUMBER** *(include area code)*