Classification of land cover using semantic segmentation

by

Dishan Anupama Nahitiya

B.S., Kansas State University, 2017

A REPORT

Submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Approved by:

Major Professor
Daniel Andreason

Copyright

Abstract

In agricultural fields, knowledge about the proportion of the soil surface covered with live vegetation and crop residue cover is key to assess the risk of soil erosion by wind and water. Live vegetation and residue cover act as an effective barrier that reduces the raindrop impact on the soil surface that can potentially break soil aggregates and wash away the soil particles and nutrients in the soil solution. Traditional methods for quantifying live vegetation and soil residue cover include line transects and sets of reference images, two methods that have proven accurate, but highly time-consuming and repetitive. This research aims at training a Deep Convolutional Neural Network (DCNN) to automate the classification of bare soil, crop residue, and live vegetation from downward-facing images of agricultural fields. A SegNet model, which is a deep convolutional encoder-decoder architecture for robust pixel-wise semantic segmentation, was trained using batch sizes of 4 images and a learning rate of 0.01. The training dataset consisted of 3300 images and the test set consisted of 645 images. All images were collected from agricultural fields and experimental plots across Kansas State University Experiment Stations. Images were first auto-labeled and then labels were manually revised by a human using the MATLAB Image Labeler application. The SegNet model resulted in an accuracy of 90% in the training set and 84% in the test set. Despite the intricate patterns, shapes, and colors given by soil, plant, and stubble element, the trained SegNet shows promising results for automating the classification of land cover from images. The trained SegNet was also implemented on a web-based application to help farmers, field agronomists, and scientists to process images for better assessment of the risk of soil erosion and to quantify the impact of soil and water conservation practices.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Dr. Andres Patrignani for providing me guidance throughout many projects. Thank you to Dr. Daniel Andresen, for supporting me every step of the way in my masters. Special thanks to Mohammad Bisheh, Sarthak Khanal, and Narmadha Mohankumar for helping me understand the topic of machine learning.

Dedication

I dedicated this work to my beloved parents Sunil Nahitiya and Kumudunie Pangoda and my brothers Hashan Nahitiya and Prashan Nahitiya.

1. Introduction

By 2050, agricultural production needs to be increased by about 71% (Rosegrant & Cline, 2003) to meet the rising global food demand. Unfortunately, the increasing pressure to increase agricultural production has acted as a catalyst for soil degradation and soil erosion (Lal et al., 2020). As a result of more intensive farming, each year an estimated 24 billion metric tons of fertile soil are lost due to soil erosion (Pretty et al., 2011). Thus, there is an increasing need for the adoption of soil and water conservation practices that improve productivity, resiliency, and sustainability of agricultural systems to reduce the rate of soil degradation (Fageria et al., 2005; Mitchell et al., 2012). Conservation practices that promote soil cover by live vegetation and crop residue can result effective to reduce soil erosion and preserve or enhance the structure and fertility of the soil (Gabriel et al., 2021).

In a crop field, soil cover can usually be present in the form of vegetation canopy and crop residue. Crop residue cover includes stems and stalks left on the soil surface after harvesting the previous crop, while canopy cover refers to the actively growing vegetation present in the field. Parts of the field that are not covered by live vegetation or crop residue usually consist of exposed bare soil. The combination of crop residue and canopy cover acts as an effective barrier to minimize soil erosion by deflecting kinetic energy from the raindrops that can break down soil aggregate and wash away the soil particles and nutrients in the soil (Xin et al., 2016). Dissolved soil nutrients carried by surface runoff usually end up in downstream water bodies and their excessive accumulation can cause eutrophication of water reservoirs that can have hazardous effects on marine life (Ekholm & Lehtoranta, 2012). An actively growing vegetation and crop residue cover also protect the soil surface from wind erosion (Duniway et al., 2019). For

instance, previous studies found that preserving 20% of residue cover on the soil surface can reduce wind erosion by about 50%, and maintaining 50% of the soil surface covered by crop residue can reduce wind erosion by 95% (Donald W. Fryrear, 1985). Residue cover can also play an important role in soil water conservation by reducing the soil evaporation rate (Flerchinger et al., 2003). Apart from acting as a protective barrier to soil, agronomists use canopy cover as a key attribute to track crop growth and estimate light interception (Shepherd et al., 2018).

Over the years, simple and practical methods have been developed to quantify the soil cover in field conditions. The line transect method (Sloneker, 1977) consists of using a measuring stick or tape to create a transect along which a trained operator counts the number of marks on the stick or tape intersecting stubble pieces or live vegetation. The transect is often repeated several times across the field to obtain an accurate field average of residue and vegetation cover. Another practical method that has gained popularity is the photo-comparison method, in which a trained operator uses a predefined set of images representing pre-calculated images of crop residue or canopy cover for a specific crop. The user needs to visually compare a selected area in the field (e.g. using a quadrat) to the set of reference images to estimate the percent of residue or vegetation cover. In general, the line transect method is more reliable than the photo-comparison method, which tends to overestimate residue cover by 6 to 10 percent (Laflen et al., 1981). While these methods provide a first-order approximation of the soil cover, they can result in tedious, time-consuming, and labor-intensive. In recent years, the increasing processing power of mobile devices has propelled the use of downward-facing digital image analysis for quantifying soil cover. Digital image processing has the potential to rapidly and accurately assess the amount of residue and vegetation cover in agricultural fields to better guide the implementation of soil and water conservation practices. For instance, a new application

called Canopeo uses a simple, but effective, thresholding algorithm that can rapidly quantify the percent of green vegetation from a downward-facing digital image (Patrignani & Ochsner, 2015). However, more advanced classification methods are required for learning and recognizing complex patterns and textures in images involving bare soil, live vegetation, and crop residue (Agarap, 2018; Narayanan et al., 2016).

There have been several research studies quantifying soil cover using machine learning methods, however, prior research work has been primarily focused on classifying green canopy cover (Levien et al., 1999). A recent study that aimed at quantifying crop residue and green vegetation using machine learning uses the Random Forest (RF) algorithm, which is an ensemble decision tree approach (Riegler-Nurscher et al., 2018). The main limitation of this approach is that it fails to effectively capture the difference between the stubble and soil due to similarities in color ranges and texture. In light of these limitations, more sophisticated methods capable of identifying the complex patterns in an image are required to achieve higher land cover classification accuracy. The recent advances in image analysis using deep neural networks have proven to be highly effective in analyzing more complex image patterns. Deep learning is a subfield of machine learning, which typically uses an architecture with a significantly larger number of layers stacked together to identify complex patterns (Voulodimos et al., 2018). Deep neural networks (DNN) are designed to continually analyze input data and learn to represent multiple levels of abstraction through generating weights and featured maps by using pattern recognition algorithms (Lecun et al., 2015). Semantic segmentation, also known as pixel-wise segmentation has been remarkable in classifying and objects in images in a class-enclosed region. The main purpose of this method is to make a dense prediction and assign a categorical label to every input pixel of an image. This technology has become very popular in the fields of

medicine (Litjens et al., 2017), natural language processing (Lecun et al., 2015), and computer vision (Voulodimos et al., 2018). This task is closely related to the semantic segmentation problem and it is one of the popular methods which is widely used in robots, robot-assisted surgery, and intelligent military system (Wang et al., 2018).

In this study, we hypothesize that semantic segmentation using a DNN network will be capable of accurately classifying the crop residue, live vegetation, and bare soil components obtained from images of agricultural fields.

2. Methodology and Experimental Design

2.2 Method description

The proposed method for classifying live vegetation, crop residue, and bare soil consists of using a deep learning convolutional neural network (CNN). Convolutional layers serve as feature extraction tools by applying a filter to the input image that generates a feature map (Figure 1).

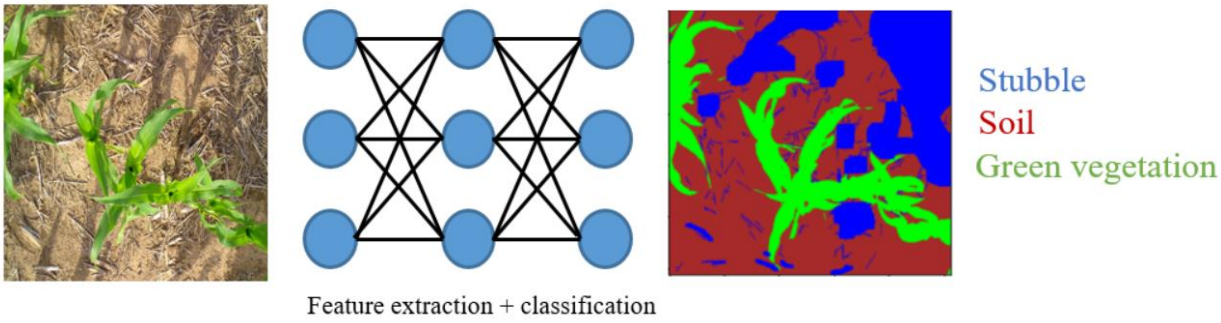

Figure 1. An illustration of the classification process of the DNNs network. The DNN model predicts the probabilities based on patterns learned through the training process of DNN models which includes self-feature extraction and classification. The model includes neurons (blue circles in the middle network) that activate based on certain patterns and assign probabilities that will help in making the decision. The neuron network will activate based on the weights stored during the training process.

Certain steps were followed in the study in required training the model as its mentions as follows.

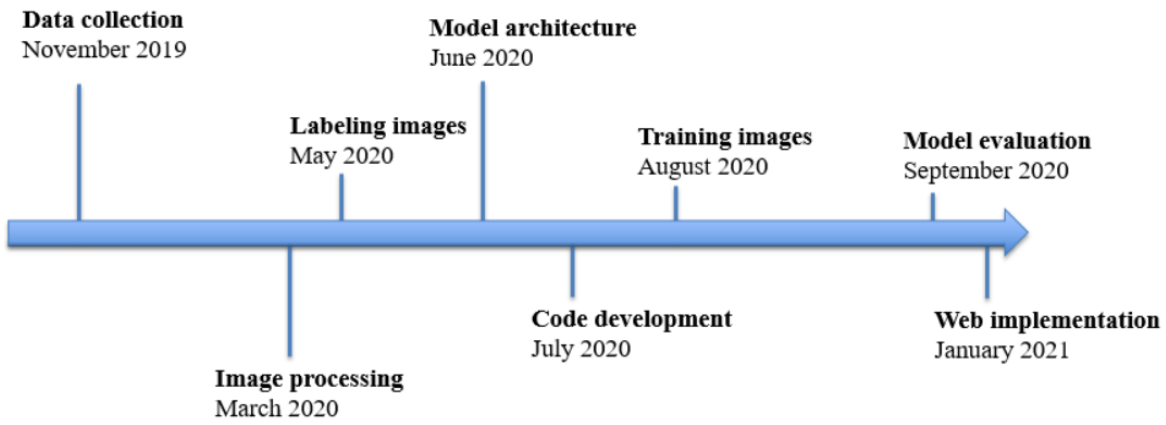

Figure 2. Flow chart of process steps in the experiment. Process steps include data collection, image processing, data labeling, development, training, and evaluation of the model, and web implementation.

As mentioned in Figure 2, the experiment starts with collecting input data for the model. Next, the labels of the images are designed and generated via MATLAB. Then the images were divided into two sets, train and test datasets. Once all the images are prepared the proposed semantic segmentation model was trained with the image data set and also evaluated.

Semantic segmentation classification algorithms were used in identifying intrinsic patterns to classify the soil cover based on a defined category set. Our goal in this method is to use an RGB color image (512 x 512 x 3) to create a pixel-wise probability segmentation map for each class label. Popular DNN architectures for semantic segmentation include U-net (Ronneberger et al., 2015), DeconvNet (Ronneberger et al., 2015), and SegNet (Badrinarayanan et al., 2017), which showed promise in the field of computer vision. In this study, we are exploring SegNet structure, a design that carries indices from max-pooling layers from the encoder to the decoder. A disadvantage of a U-Net is that it carries the full-featured map to the decoder, requiring heaving amounts of memory consumption. Similarly, a DeconvNet uses a

fully connected layer which makes it larger to train. Therefore we favored the use of a SegNet structure which requires fewer parameters than the other structures.

2.1.1 SegNet Architecture

The Seg (semantic segmentation) Net (Network) architecture is an encoder-decoder based architecture proposed by (Badrinarayanan et al., 2017) in the year 2015. The encoder layer includes 5 encoders and consists of 13 layer of convolutional layers included in the VGG16 network (Simonyan & Zisserman, 2015). Each encoder produces a featured map with the help convolutional layer which applies dot product by sliding a filter sized patch across the two-dimensional image. The encoder also includes a convolutional layer followed by a batch normalization layer (Li et al., 2019) and a rectified linear unit layer (RELU) (Zhong et al., 2019). The pooling layers which follow help to create a spatial resolution of the maps. The architecture discards the fully connected layers and only retains high-resolution feature maps in the decoder outputs, which can reduce the number of trainable parameters significantly.

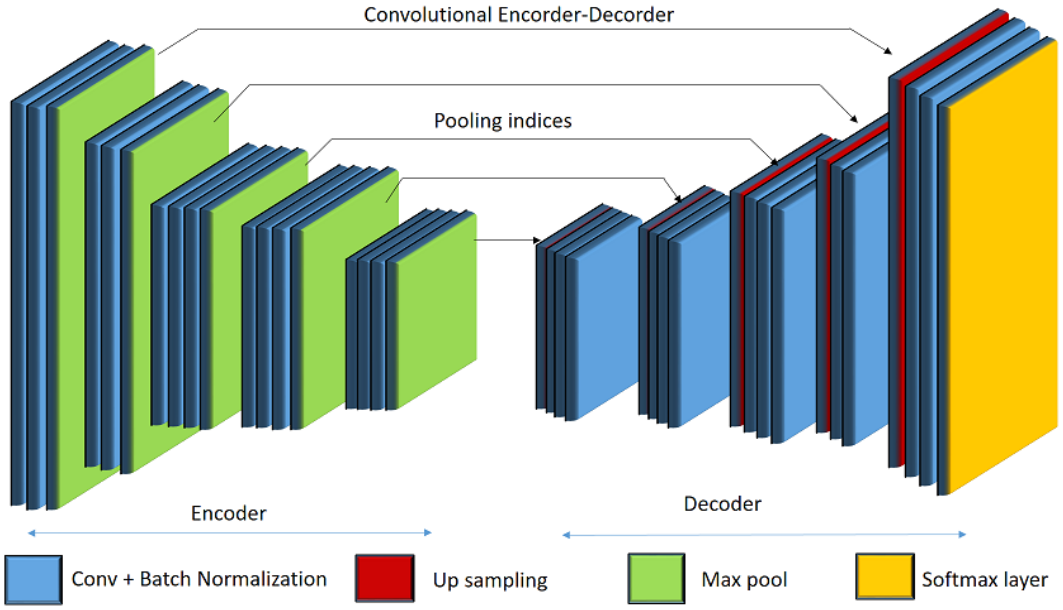

Figure 3. An illustration of proposed SegNet architecture. SegNet, an encoder-decoder architecture using pooling indices to carry the full feature map throughout the model. The structure does not include any fully connected layers due to convolutional behavior. The decoder retrieves pooling indices from the encoder and produces the feature maps. (Badrinarayanan et al., 2017).

The key component of the SegNet is that each encoder corresponds to a decoder in the hierarchical order. As mentioned in Figure 4, each upsampling layer in the decoder receives pooling indices from the corresponding encoder concatenate with the non-leaner upsample on the feature maps.

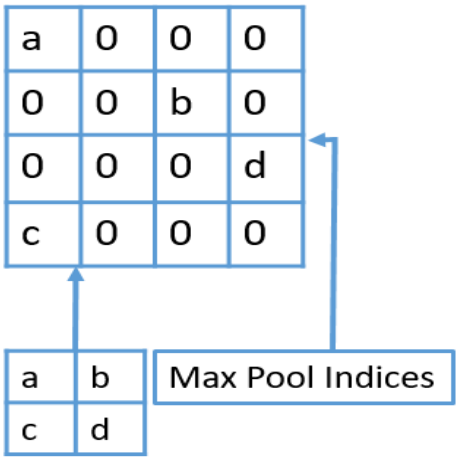

Figure 4. An illustration of the process of the SegNet structure. a, b, c, d are values that are carried through feature maps. The SegNet uses max pooling indices to upsample the feature maps (Badrinarayanan et al., 2017).

2.2 Dataset

The dataset of images for this study consisted of downward-facing digital images obtained from agricultural fields containing soil, stubble, and green vegetation. Images were collected at several Kansas State University Experiment Research Stations. Images were taken with a combination of mobile devices and point-and-shoot cameras about 1.5 meters above the ground. Before labeling, images were downsized to 512 x 512 pixels. The dataset contained a total of 3972 images that were manually aggregated based on the predominant classes in each image Table 1

Table 1. Soil cover is identified as three key attributes of soil, stubble, and Plant. This table indicates the distribution of the images.

Category	Number of images
Soil	212
Stubble	262
Soil and stubble	461
Soil and plant	410
Stubble and plant	112
Soil, stubble, and plant	1846
Total	3300

The labeling process was carried in two steps, an automated step and a manual step. The automated labeling step consisted of first classifying green vegetation using the existing algorithm in Canopeo (Patrignani & Ochsner, 2015). After classifying green pixels, bare soil and crop residue were classified using Otsu's thresholding technique (Jun & Jinglu, 2008). This classification sequence provided a first-order approximation to the three classes present in the images. In the classifying process, Canopeo was successful in identifying the green canopy in images[1]. The rest of the pixels of the image was then classified using the OTSU method, which has the slight ability to distinguish between stubble and soil based on thresholds.

[1] The canopeo algorithm uses extracted RGB layer information and calculate the color ratios of red to green (R/G) and blue to green (B/G) and an excess green index-ExG (2G–R–B)

ExG = 2*G - R – B , RG = R/G , BG = B/G

plant = RG < 0.95 & BG < 0.95 & ExG > 20

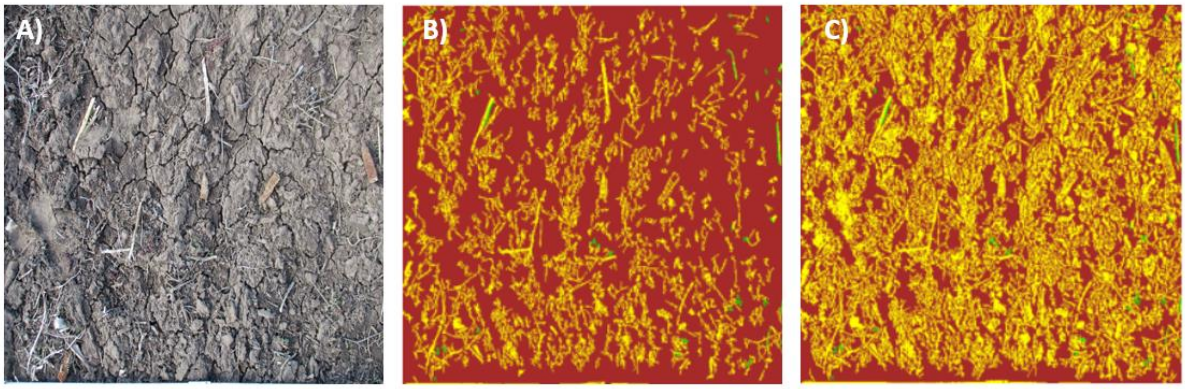

Figure 5. Residue, stubble, and plant classification using a combination of Canopeo and Otsu's method. A) Original image. B) Classified image with lower threshold sensitivity (0.35). C) Canopeo and Otsu classified image with a higher threshold sensitivity (0.65). The images are classified as soil (brown), residue (yellow), and plant (green).

As seen in Figure 5, the Otsu method continuously needed supervision and adjustment of threshold sensitivity in the labeling process. In the scenario, the Otsu method had a higher chance of identifying the stubble and soil in the ground and a higher threshold misclassify most of the soil classes as residue. Due to this behavior, after the images were auto-labeled, the labels in each image were inspected by a trained operator, and changes in pixel labels were manually revised. The entire labeling process was conducted using the MATLAB Image Labeler application (Mathworks, Inc., Natick, MA). Pixels in the image was be categorized into classes of stubble, live green vegetation, and bare soil. The labeled image set contains 3300 images and 645 test images possessing a resolution of 512 x 512 per a labeled image.

Table 2. Pixel count per class label across the entire dataset.

Data set	**Soil**	**Plant**	**Stubble**	**None**
Train set	381258802	239173488	244491360	151550
Test set	75812418	39904756	53095226	8336

After investigating the class category ‘none’ we conclude that best fit pixels represented by this class can be reassigned as soil class. During the processing stage in the training process of the model deep learning model, we re-assigned the ‘none’ pixels to class soil.

2.3 Experimental Setup

SegNet architecture consists of 4 encoders and corresponded decoders with the same architecture as the experimental architecture and the filter sizes for the convolutional layers were varied throughout the model in contrast to the recommended constant filter size of 64 in the original SegNet structure.

Table 3. Information for each layer of the SegNet architecture used in this study. The encoder is a combination of a convolutional layer, batch normalization layer, and Relu layer, and the decoder consists of upsampling layers.

Set	**Type**	**# Filter**	**Layer/Padding**	**Input size**	**Output size**
Encoder 1	Conv+BN Relu	64	M,2	512 x512 x3	512 x 512 x 64
Encoder 2	Conv+BN+Relu	128	M,2	512 x 512 x 64	256 x 256 x128
Encoder 3	Conv+BN+Relu	256	M,2	256 x 256 x128	128 x 128 x 256
Encoder 4	Conv+BN+Relu	256	M,2	128 x 128 x 256	64 x 64 x 256
Encoder 5	Conv+BN+Relu	512	M,2	64 x 64 x 256	32 x 32 x 512
Decoder 5	Deconv	256	U,2	32 x 32 x 512	64 x 64 x 256
Decoder 4	Deconv	256	U,2	64 x 64 x 256	128 x 128 x 256
Decoder 3	Deconv	64	U,2	128 x 128 x 256	256 x 256 x 64
Decoder 2	Deconv	32	U,2	256 x 256 x 64	512 x 512 x 3
Decoder 1		softmax			512 x 512 x 3

M stands for max-pooling U stands for upsampling (Mittal et al., 2018)

The training of the DNN model was conducted with the *Python 3.7.9* version on an *Anaconda 4.9.2* virtual environment. *Keras*, a high-level deep learning module offered by *TensorFlow* version 2.3 was utilized in the training process of the model. Moreover, *Sklearn, Numpy, Pandas, Matplotlib* libraries were utilized in supporting other functions. Pre-trained SegNet model architecture was tuned using hyperparameters in the project. The experiment was conducted in a desktop computer equipped with a 3.8 GHz Ryzen 9 3900x 12-core processor and a single GeForce RTX 2080 (NVIDIA, Santa Clara, CA) graphics processing unit (GPU). To speed up the training of the SegNet, the image data processing was conducted using 32-bit single-precision floating-point format.

2.3.3. Tuning hyperparameters

Table 4. The hyperparameter training. For the process of training, hyperparameters were tested under the key main feature of Learning rates, optimizers, batch size, and dropout. The table demonstrates the relative experiment.

Parameter	**Variations**
Learning rates	0.1, 0.01
Optimizers	RMS prop, Adam
Batch size	1, 4
Dropouts	20%, 40%.
Epochs	1000, 3000

In model training, gradient descent optimization (Zhong et al., 2019) was used to minimize the cost function as far as possible. Categorical cross-entropy (CE), a softmax activation algorithm was utilized to identify the divergence between class probability in the distributions (Janocha & Czarnecki, 2017). Then the softmax function squashes a multi-classification vector generated by CE into a value between 0 to 1 creating a probability spread for each class which ultimately adds up to 1 for each pixel. The probabilities of classes can be identified as one-of-many classification which depicts that each sample pixel in the sample set should belong to one category at all times.

Fine-tuning of the hyperparameters was conducted before the training model. This process improves the accuracy and efficiency of the model. As mentioned in Table 4, well-known optimizers for semantic segmentation, Root Means Square Propagation (RMSProp) and Adaptive Moment Optimization (Adam) were used (setiawan et al., 2018) for this experiment. Then, we devised a short experiment including a small range of settings and optimizers. The baseline experiment was designed as follows.

- Size of the images used: 512 x 512 pixels
- Number of the channel in the image: 3
- Networks optimizers: RMS prop
- Learning rate for the optimizer: 0.01
- The dropout rate in the classifier network: 0.4
- Number of epochs for classifier training: 200

Once the hyperparameters were analyzed, we started training the best-performing model with the input image data for multiple iterations (epochs).

2.4. Evaluation

Evaluation for the best performing model was conducted by analyzing the confusion matrix and overall accuracy over the test data set. The confusion matrix describes how well the classifier predicts each class for each pixel. With help of the confusion matrix, we analyzed the factors of precision, recall, and F1-score as mentioned in the following equations.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Where F-1 score, Precision, and recall are formulas calculated from the confusion matrix. TP stands for true positive, FP stands for false positive, TN stands for true negative, and FN stands for false negative. According to eq. 2, Precision specifies the fact that of being accurate

and eq.3 Recall stands for whether the actual positive was identified through the classification process.

Accuracy of confusion matrix proves to be a better indicator of the model strength in identifying the distributions of the classes. (Mittal et al., 2018) .We used the F1 score as a weighted average on precision and recall to evaluate the model accuracy. F1 score illustrates the balance between Precision and Recall.

3. Results and Discussion

First, the hyperparameters were tuned based on the baseline experiment to fine-tune the model. The experiment was conducted under a constant seed value in Tensorflow to make stable output for every training.

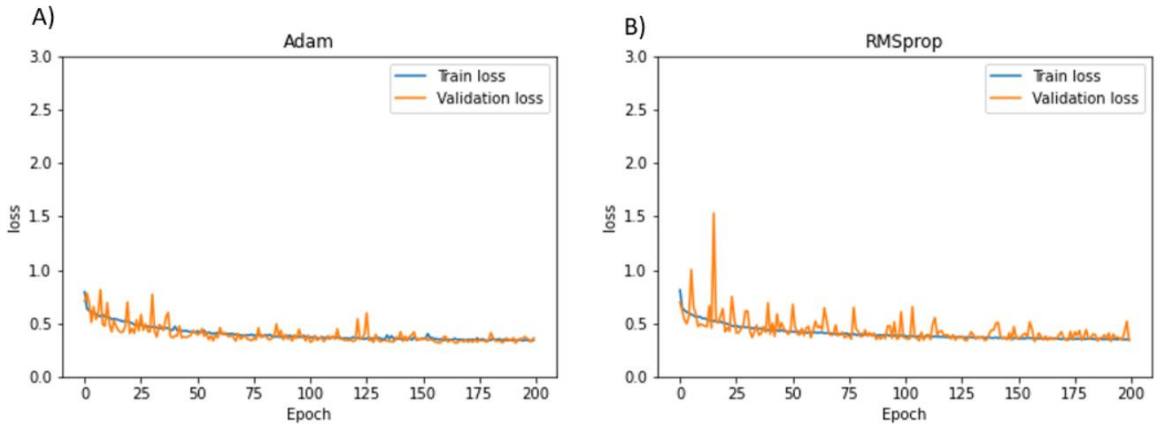

Figure 6. Optimizer variation results in loss vs Epochs. Adam optimizer (A) and RMS prop (B) were used and both optimizers managed to converge as expected, but the RMS prop converged slightly earlier and produce less loss than Adam.

Gradient descent optimization algorithms are proven to be significant in multiclass classifications for the baseline experiment and the core algorithm which helps the neural network minimize the loss. Figure 6 depicts that both optimizers demonstrated promising results during the training process. RMS prop optimizer converged slightly earlier than the Adam optimizer to achieving similar losses during the training process. Then the batch size was adjusted to experiment with the loss fluctuation that occurred in training.

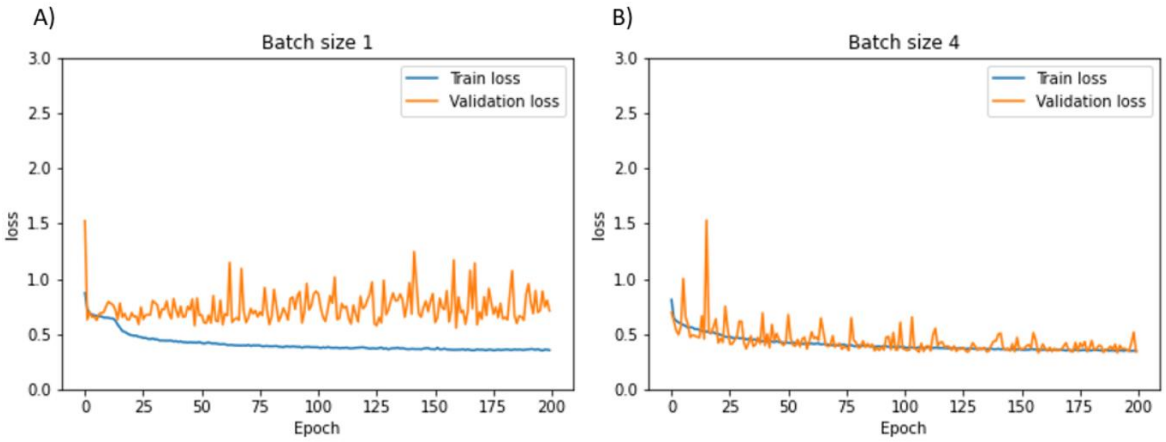

Figure 7. Batch size variations result in loss vs epochs. The training results of the batch sizes for batch sizes 1 and 4. The batch size refers to the number of samples that the model work through before it updates the model parameters in each training session.

Unexpectedly, according to Figure 7, a lower batch size of size as 1 image which training performance was not significant enough to be considered for an output. We assume that this may result in higher fluctuations in losses encountered in the training process. Due to hardware constraints, we had to reduce to 4 images per batch. Furthermore, we apply dropout regularization methods to avoid overfitting in the training process. More training was needed to increase the accuracy of the model and the process of which the model starting mimicking the output rather than predicting the output

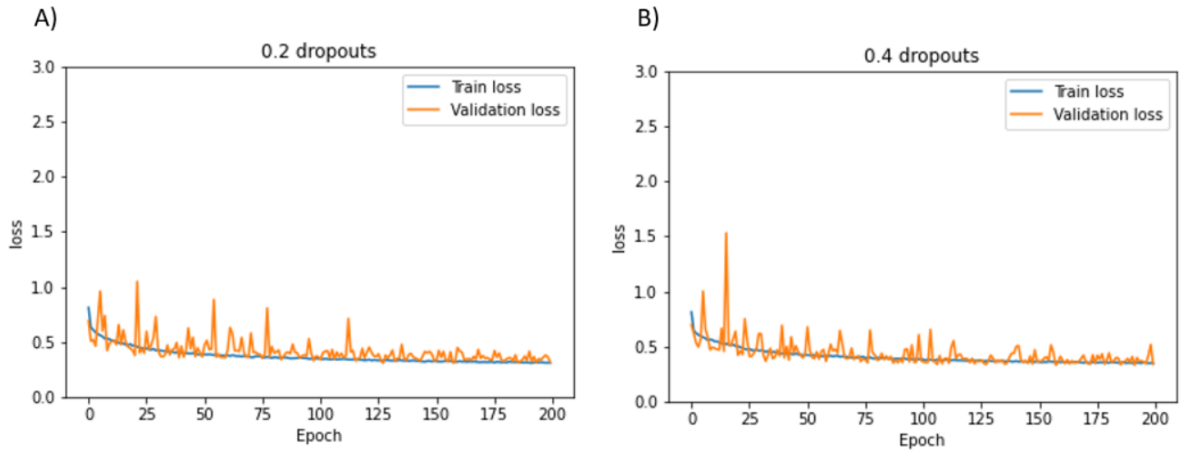

Figure 8. Dropout variations on loss vs epochs**.** Results from changing dropout relative value of 0.2 vs 0.4. Both curves behave similarly and 0.4 dropout shows less fluctuation.

Figure 8 states that the loss function validation loss curve and the training loss curve initially converged on 200 epochs. 0.4% dropouts were giving slight variation in the loss. After implementing the dropout we continuously increased the number of epochs to achieve higher accuracy without overfitting. Figure 9 shows the model behavior training under higher epochs. The accuracy slightly improved with time.

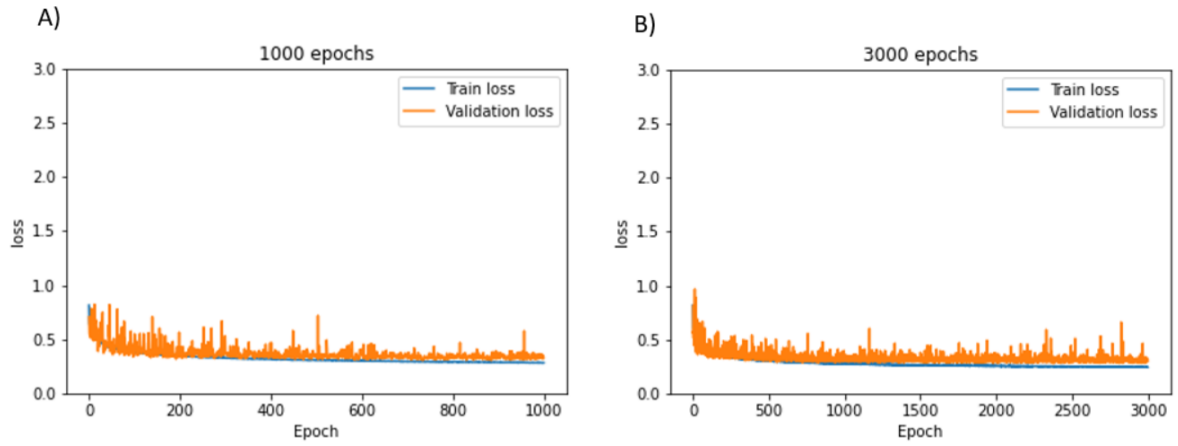

Figure 9. Variants on epochs. Training results for variation epochs values. The training slight improvement of the test accuracy but overall can be neglected.

After fine-tuning the hyperparameters, the tuned model was trained and evaluated model performance with the 644 of the test images. Results were gathered based on the variations of epochs.

Table 5. Training results after training the model based on the SegNet model.

Training epoch	**Validation loss**	**Training loss**	**Test accuracy**
200	0.3396	0.3501	0.8424
1000	0.3127	0.2816	0.839
3000	0.3003	0.2447	0.8461
5000	0.3079	0.2407	0.8483

Observations from Table 5 illustrate that there was no significant improvement for the stubbles' Training loss over time. But the validation loss and overall test accuracy increase improved slightly. The largest evaluation matrices values were evaluated at model trained to 3000 epochs.

Table 6. Evaluation results on trained best performing SegNet model which was trained for 3000 epochs under 0.01 learning rate.

Label	**F1-score**	**Precision**	**Recall**
Stubble	0.77	0.86	0.69
Soil	0.85	0.79	0.93
Canopy	0.92	0.95	0.91

After training the best performing model, F1 scores were analyzed to identify the classification performance of each class. As evaluated in Table 6, the model was able to identify stubble with higher precision (0.86) than soil (0.79) but failed to have a lower recall of 0.69. In contrast, soil class acquired a higher recall (0.93) than stubble (0.69). This indicates that stubble carried in most of the scenarios, classify more as soil and less as stubble. Overall the green canopy has the highest success in classifying having the highest precision (0.95) and recall (0.91).

The use of the deep learning method was then compared with the previously utilized image labeling techniques to evaluate the performance of the trained semantic segmentation model. The Canopeo and Otsu threshold techniques were highly supportive effective in labeling the images. The SegNet model was evaluated on the factors of precision, recall, F1-score as mention in Table 7.

Table 7. Evaluation of the labeling algorithms Canopeo and Otsu algorithms. Initially, the Canopeo method classifies the green vegetation in the image. Otsu methods mainly focus on classifying remainder pixels into residue and soil.

Label†	**F1-score**	**Precision**	**Recall**
Stubble	0.48	0.64	0.38
Soil	0.74	0.65	0.86
Canopy	0.96	0.98	0.95

† Otsu's methods mainly focus on classifying remainder pixels into residue and soil. Images for this test were labeled using the Otsu algorithm with a parameter value of 0.35, which was a reasonable thresholding sensitivity based on several sample images.

In both techniques thresholding and deep neural network learning, the Canopeo algorithm was nearly successful in classify the green vegetation in the images. The F1-score values in stubble (0.48) and soil (0.74) were comparatively lower than the stubbles (0.77) and soil (0.85)

classification in the Deep learning model. This confirms that the SegNet model performs better in identifying the slight pattern difference between soil and residue and the necessity of manual cleaning after utilizing the auto labeling algorithm.

The trained SegNet model was then visually evaluated for model performance. Diverse images were considered for the evaluation process. Input images were analyzed through the model and output from the softmax layer was analyzed through an Argmax operation to (returns the index of largest predicted category) to the predicted label for the cover.

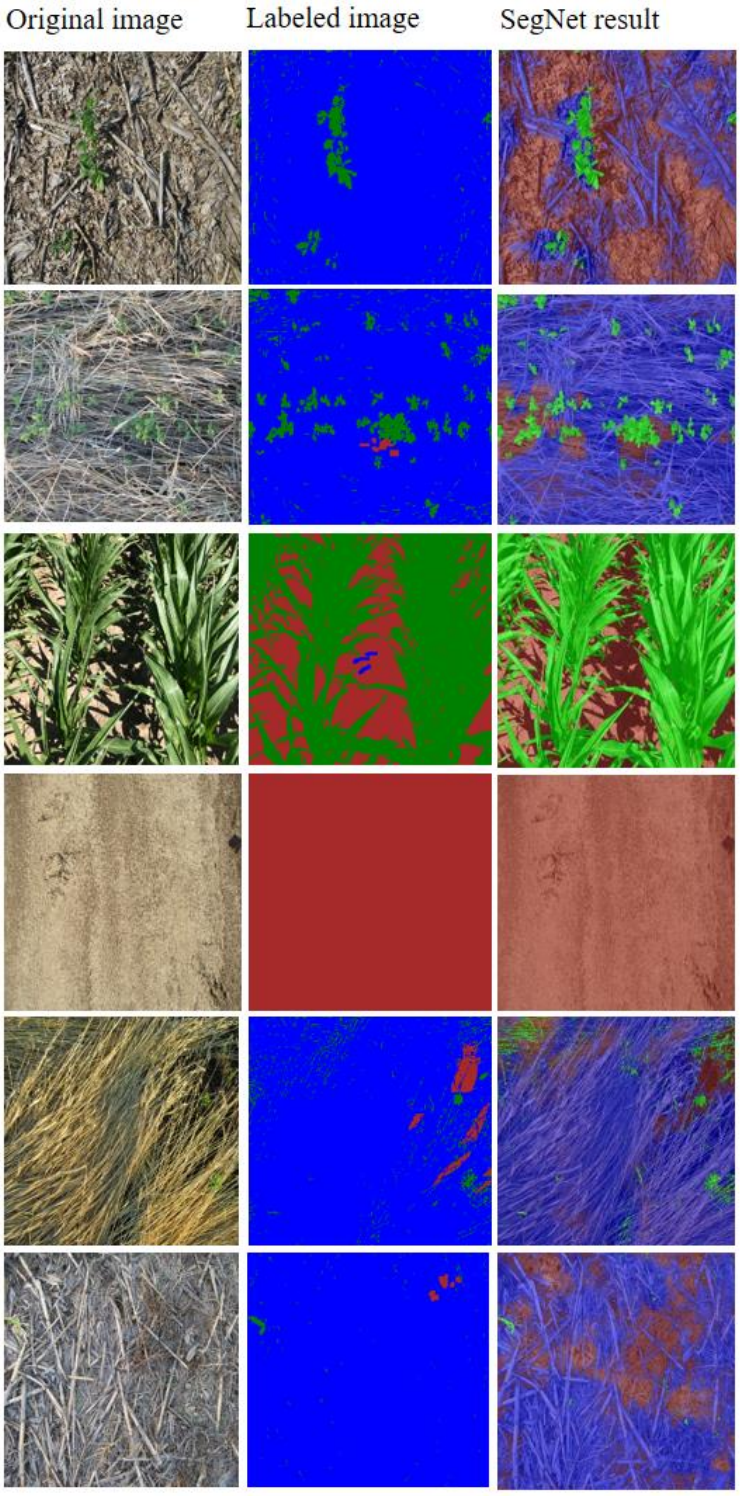

Figure 10. Examples of predictions from the trained SegNet, left column has the original image, the middle column consists of the corresponding labeled images labeled from MATLAB, and the rightmost column has the images consist of the prediction from the trained SegNet model, and classes can identify as soil (brown), stubble (blue) and canopy (green).

According to Figure 10, the trained SegNet model was able to identify the canopy cover much more accurately. In most cases, the model was able to properly classify the strong, significant features in stubble, but fail to perform well in classifying weaker, plain features.

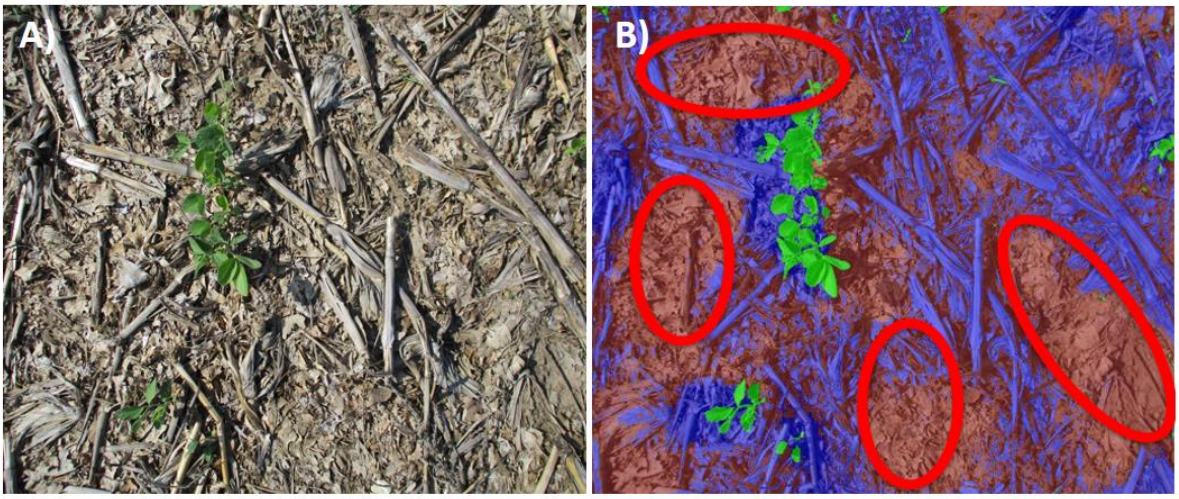

Figure 11. The image pair shows a scenario of misclassifications from the trained SegNet model. A) Original image of soil cover and the B) Predicted image from the SegNet model. The red circles indicate the misclassified regions.

Figure 11 explains the stronger and weaker features predicted by the model. Due to this result, the stubble and soil carry low F1 scores. In most of the scenarios, it depicts more soil and less stubble. Our effort is to increase the accuracy of the quantifying stubble. The model was accurate in identifying the green vegetation in the test image set. Most errors occurred on the soil and stubble classifications.

First, the DNN model was trained and extracted was available for the general public through a web application. SRPNet, the web application takes an image input through a camera or any other device. Then 2D images are processed according to the models' input size. The model then calculates the soil cover with the help of the trained model. After the training, the

model was saved to *Tensorflow* protobuf type and then converted the model to JSON type with the assistance of *TensorFlow.js* convertors. The model was installed in an easy-to-use standalone web application in a combined environment of vanilla JavaScript and *Tensorflow.js 3.3.0*. The web app was then hosted as a standalone, free-to-use open-source web app in GitHub. The app has analyses images with the help of *tensorflow.js*, and *WebGL*, a JavaScript API that helps to access the GPU in the electronic equipment.

The application utilized tidy method which frees up the memory from intermediate tensors which occur inside the function return a new tensor with the converted image. This method will help to save the browser memory without getting overloaded.

```
    const tensorImg = await tf.tidy(() => {
        let image_from_element = document.getElementById("uploaded_image");
        let tensorImg          =
tf.browser.fromPixels(image_from_element).toFloat().expandDims();
        return tensorImg;

    });
```

The program will expand the dimension of the image to 1x512x512x3 which fits the input of the model for the predictions.

```
    prediction = await model.predict(tensorImg);
    tf.dispose(tensorImg);
    var results = await prediction.argMax(3).dataSync();
    tf.dispose(prediction);
```

The prediction and image generation have time complexity O(1) or constant time per pixel. The prediction results are computed using an Argmax function with constant time complexity on each pixel. Moreover, when the image is W x H pixels, then the complexity is O(W*H) to

process the image. Then, the output index is converted to an RGB image in the following code snippet.

```
// create an offscreen canvas
var canvas = document.createElement("canvas");
var ctx    = canvas.getContext("2d");

// size the canvas to your desired image
canvas.width  = 512;
canvas.height = 512;

// get the imageData and pixel array from the canvas
var imgData = ctx.getImageData(0, 0, 512, 512);
var data    = imgData.data;

index   = 0;
counter = 0;
for(let y=0; y < canvas.height; y++){
    for(let x=0; x < canvas.width; x++){
        let index = (x + y * canvas.width)*4;

        pixel_value = results[counter];
        // stubble
        if (pixel_value == 0){
            data[index+0] = 255;
            data[index+1] = 255;
            data[index+2] = 0;
            data[index+3] = 255;

        }else if(pixel_value == 1){
            // soil brown
            data[index+0] = 165;
            data[index+1] = 42;
            data[index+2] = 42;
            data[index+3] = 255;

        }else {
            //live vegetation green
            data[index+0] = 0;
            data[index+1] = 255;
            data[index+2] = 0;
            data[index+3] = 255;

        }
        counter += 1;
    }
}
```

During the prediction process, the image size did not affect the runtime, the app always downsamples the RGB image to 512 x 512 x 3 and then predicts probability matrix of 512 x 512

x 3 and the image is also always generated for this same dimension and the algorithm does not depend on the input image.

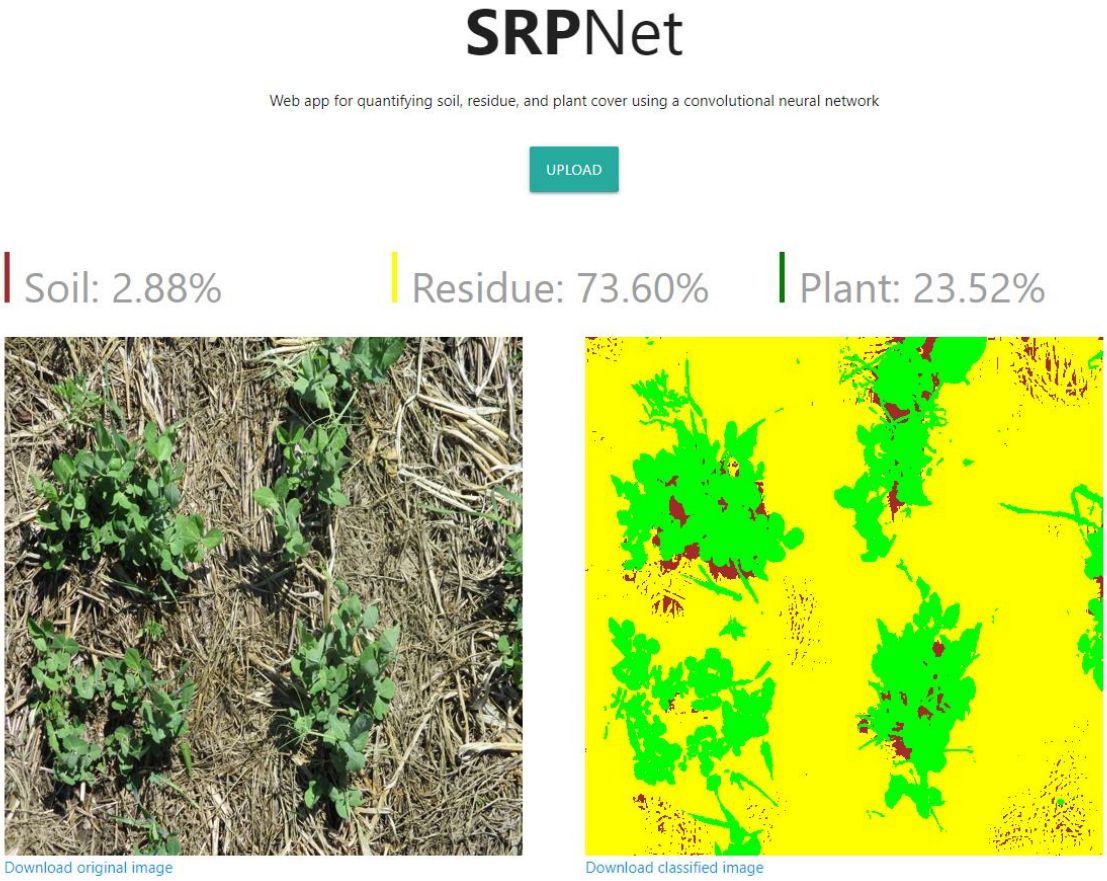

Figure 12. Image of the front-end user interface of the SRPNet web application. The app utilizes the JSON version of the trained SegNet model which takes the input as an image (left) and predicts the labels (right).

4. Future work

The deep learning model can improve the accuracy by increasing the number of data in the dataset. We are planning to increase the quantity of the images which represent more scenarios of stubble and soil. This will be helpful for us to help the model to learn slight differences between stubble and soil and improve the accuracy of the DNN model. Furthermore, there will be more land covers categories introduced in classifying images. Moreover, we are planning to use this trained SegNet model for transfer learning to classify classes in digital images in other fields beyond agriculture. We are also planning to host the web application utilizing the model open source for users to use in their respective browsers of their choice.

5. References

Agarap, A. F. M. (2018). Deep Learning using Rectified Linear Units (ReLU). In *arXiv*. https://github.com/AFAgarap/relu-classifier.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

Donald W. Fryrear. (1985). Soil Cover and Wind Erosion. *Transactions of the ASAE*, *28*(3), 781–784. https://doi.org/10.13031/2013.32337

Duniway, M. C., Pfennigwerth, A. A., Fick, S. E., Nauman, T. W., Belnap, J., & Barger, N. N. (2019). *Wind erosion and dust from US drylands: a review of causes, consequences, and solutions in a changing world*. Ecosphere. https://doi.org/10.1002/ecs2.2650

Ekholm, P., & Lehtoranta, J. (2012). Does control of soil erosion inhibit aquatic eutrophication? In *Journal of Environmental Management* (Vol. 93, Issue 1, pp. 140–146). Academic Press. https://doi.org/10.1016/j.jenvman.2011.09.010

Fageria, N. K., Baligar, V. C., & Bailey, B. A. (2005). Role of cover crops in improving soil and row crop productivity. In *Communications in Soil Science and Plant Analysis* (Vol. 36, Issues 19–20, pp. 2733–2757). Taylor & Francis Group . https://doi.org/10.1080/00103620500303939

Flerchinger, G. N., Sauer, T. J., & Aiken, R. A. (2003). Effects of crop residue cover and architecture on heat and water transfer at the soil surface. *Geoderma*, *116*(1–2), 217–233. https://doi.org/10.1016/S0016-7061(03)00102-2

Gabriel, J. L., García-González, I., Quemada, M., Martin-Lammerding, D., Alonso-Ayuso, M., & Hontoria, C. (2021). Cover crops reduce soil resistance to penetration by preserving soil surface water content. *Geoderma*, *386*, 114911. https://doi.org/10.1016/j.geoderma.2020.114911

Janocha, K., & Czarnecki, W. M. (2017). *On Loss Functions for Deep Neural Networks in Classification*.

Jun, Z., & Jinglu, H. (2008). Image segmentation based on 2D Otsu method with histogram analysis. *Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008*, *6*, 105–108. https://doi.org/10.1109/CSSE.2008.206

Laflen, J. M., Amemiya, M., & Hintz, E. A. (1981). Measuring crop residue cover. *Journal of Soil and Water Conservation*, *36*(6).

Lal, R., Blum, W. E. H., Valentin, C., & Stewart, B. A. (2020). *Methods for Assessment of Soil Degradation*. CRC Press. https://books.google.com/books?id=lSAJEAAAQBAJ

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. In *Nature* (Vol. 521, Issue 7553, pp. 436–444). Nature Publishing Group. https://doi.org/10.1038/nature14539

Levien, L. M., Roffers, P., Maurizi, B., Suero, J., Fischer, C., & Huang, X. (1999). *A MACHINE-LEARNING APPROACH TO CHANGE DETECTION USING MULTI-SCALE IMAGERY 1*.

Li, Y., Wang, N., Shi, J., Liu, J., & Hou, X. (2019, March 15). Revisiting batch normalization for practical domain adaptation. *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*. https://arxiv.org/abs/1603.04779v4

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. In *Medical Image Analysis* (Vol. 42, pp. 60–88). Elsevier B.V. https://doi.org/10.1016/j.media.2017.07.005

Mitchell, J. P., Singh, P. N., Wallender, W. W., Munk, D. S., Wroble, J. F., Horwath, W. R., Hogan, P., Roy, R., & Hanson, B. R. (2012). No-Tillage and high-residue practices reduce soil water evaporation. *California Agriculture*, *66*(2), 55–61. https://doi.org/10.3733/ca.v066n02p55

Mittal, A., Hooda, R., & Sofat, S. (2018). LF-SegNet: A Fully Convolutional Encoder–Decoder Network for Segmenting Lung Fields from Chest Radiographs. *Wireless Personal Communications*, *101*(1), 511–529. https://doi.org/10.1007/s11277-018-5702-9

Narayanan, B. N., Djaneye-Boundjou, O., & Kebede, T. M. (2016). Performance analysis of machine learning and pattern recognition algorithms for Malware classification. *Proceedings of the IEEE National Aerospace Electronics Conference, NAECON*, *0*, 338–342. https://doi.org/10.1109/NAECON.2016.7856826

Patrignani, A., & Ochsner, T. E. (2015). Canopeo: A Powerful New Tool for Measuring Fractional Green Canopy Cover. *Agronomy Journal*, *107*(6), 2312–2320. https://doi.org/10.2134/agronj15.0150

Pretty, J., Toulmin, C., & Williams, S. (2011). Sustainable intensification in African agriculture. *International Journal of Agricultural Sustainability*, *9*(1), 5–24. https://doi.org/10.3763/ijas.2010.0583

Riegler-Nurscher, P., Prankl, J., Bauer, T., Strauss, P., & Prankl, H. (2018). A machine learning approach for pixel wise classification of residue and vegetation cover under field conditions. *Biosystems Engineering*, *169*, 188–198. https://doi.org/10.1016/j.biosystemseng.2018.02.011

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. http://arxiv.org/abs/1505.04597

Rosegrant, M. W., & Cline, S. A. (2003). Global Food Security: Challenges and Policies. In *Science* (Vol. 302, Issue 5652, pp. 1917–1919). https://doi.org/10.1126/science.1092958

setiawan, wahyudi, setiawan, wahyudi, Utoyo, M. I., & Rulaningtyas, R. (2018). Vessels semantic segmentation with gradient descent optimization. *International Journal of Engineering & Technology*, *7*(4), 4062–4067. https://doi.org/10.14419/ijet.v7i4.18104

Shepherd, M. J., Lindsey, L. E., & Lindsey, A. J. (2018). Soybean Canopy Cover Measured with Canopeo Compared with Light Interception. *Agricultural & Environmental Letters*, *3*(1), 180031. https://doi.org/10.2134/ael2018.06.0031

Simonyan, K., & Zisserman, A. (2015, September 4). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. http://www.robots.ox.ac.uk/

Sloneker, L. L, and W. C. M. (1977). *Measuring the amounts of crop residue remaining after tillage*.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. In *Computational Intelligence and Neuroscience* (Vol. 2018). Hindawi Limited. https://doi.org/10.1155/2018/7068349

Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., & Cottrell, G. (2018). Understanding Convolution for Semantic Segmentation. *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, *2018-January*, 1451–1460. https://doi.org/10.1109/WACV.2018.00163

Xin, Y., Xie, Y., Liu, Y., Liu, H., & Ren, X. (2016). Residue cover effects on soil erosion and the infiltration in black soil under simulated rainfall experiments. *Journal of Hydrology*, *543*, 651–658. https://doi.org/10.1016/j.jhydrol.2016.10.036

Zhong, L., Hu, L., & Zhou, H. (2019). Deep learning based multi-temporal crop classification. *Remote Sensing of Environment*, *221*, 430–443. https://doi.org/10.1016/j.rse.2018.11.032