[Team 21] Project 2.2
Segmentation of Crop Damage

Daniel Mendez Lozada
dmendez@ncsu.edu

Alhiet Orbegoso
aoorbego@ncsu.edu

Chanae Ottley
cottley@ncsu.edu

I. Problem Description

Given a data set comprised of *731* RGB images of crop fields with a resolution of *4608x3456* pixels, our task is to be able to create a model that can predict regions that are damaged by highlighting them.

Images were captured using a drone camera and taken in the *Cherry Farm* research station. Out of the set, *585* images are used for training and are provided with their respective annotations, where crop damage segmentation is represented with a white mask. The remainder of the images will be used for testing. The overall goal of this project is to perform semantic segmentation using a deep learning approach.

II. Methodology

Segmentation is a process that involves analyzing, labeling, and grouping pixels in an image[4]. Since we are interested in identifying regions where crop damage has occurred, we require implementing a binary semantic segmentation, i.e., classifying all the image pixels into two different classes of objects: damaged and not damaged.

In this project, we implemented a deep neural network inspired by the *SegNet's* deep encoder-decoder architecture [2]. It is a deep, fully convolutional neural network architecture for semantic pixel-wise segmentation consisting of an encoder and a decoder that extracts features from the image through filters. See Figure 1.

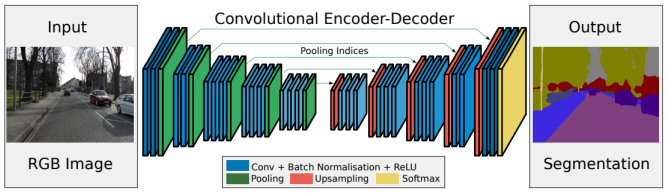

Fig. 1. SegNet architecture [2].

Our final model uses the SegNet's VGG16 architecture, which has 13 convolutional layers and three linear layers with the ReLU activation function for its encoder part. Each encoder has a convolutional and batch normalization layer with a corresponding decoder that upsamples the features using the max pooling indices from their encoding pair. The last decoder uses a sigmoid layer to classify each pixel. Implementation was done using *Pytorch* library and can be reviewed in the project repository [6].

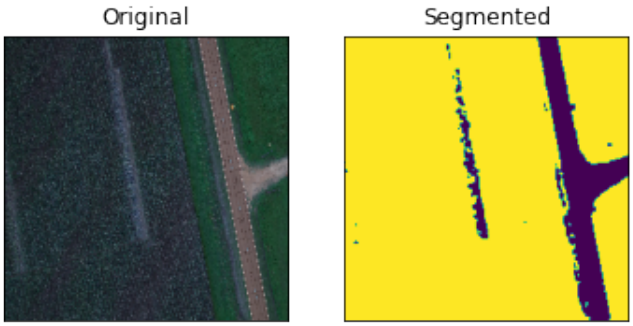

Fig. 2. Segmentation using thresholding.

To validate our model, we originally intended to use image processing techniques since they are simple and classical approaches for segmentation. Particularly, we compared our model results with *thresholding segmentation*. Figure 2 shows the result of applying this technique to one image from the training set. Note that damaged crop regions are confused with similar patterns such as brown land and roads.

As observed in Figure 2, comparing our results with pre-processing techniques does not help us determine if they are reasonable or not. i.e. it is hard to validate our results since these methods perform segmentation using only pixel information in the entire image, which contrasts with our deep-neural-network approach where segmentation is done using information from all images in the entire data set.

Therefore, before using the final Segnet model, a less complex architecture was used as the baseline to ensure the feasibility of the final model. Such model is a basic autoencoder with 6 convolutional layers per encoder and decoder, and between them, ReLU activation functions were added. This basic model performed well on the data set; nevertheless, crop-damaged sections with horizontal pattern were not adequately captured (See Figure 3). Based on those results, we determined the model selection was accurate, but a deeper architecture was necessary to get horizontal and more complex patterns.

III. Model Training and Selection

A. Model Training

Among the 585 images available for training, we chose to follow the *70/30* rule and used 409 images for training, 87 for

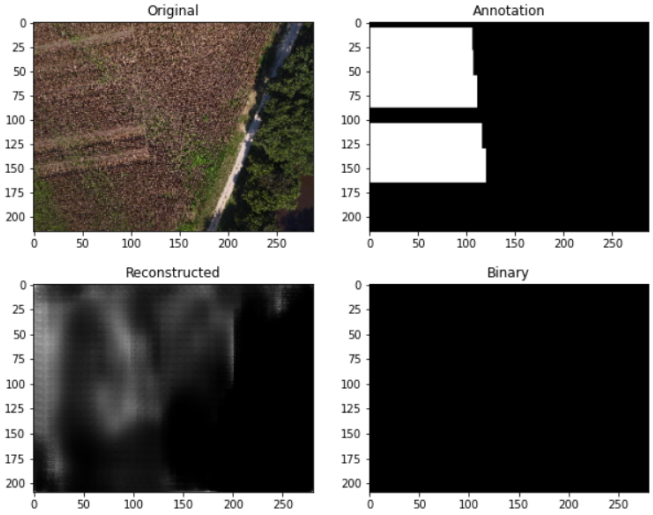

Fig. 3. Prediction of baseline model on horizontal pattern

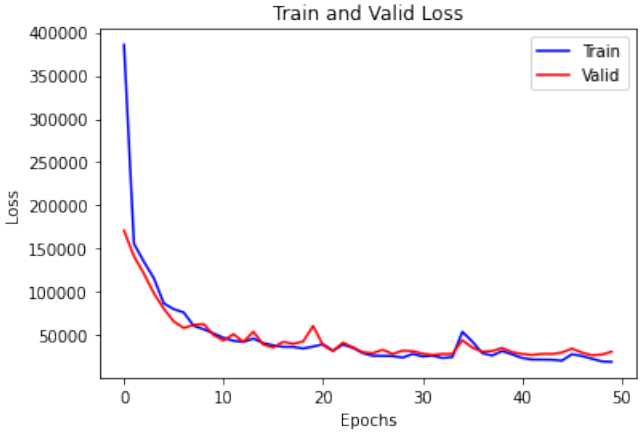

Fig. 4. Final training and validation loss plot.

validation, and 89 for testing. The test set is not used during training and is reserved for evaluation of the model.

For training, we use *Binary Cross Entropy* (BCE) as our loss function, given that we are predicting the probability of an image section being classified as crop-damage. The BCE loss is performed pixel-wise and summed over all pixels in the image. Figure 4 shows the training and validation loss of our final model. Note that loss values converge quickly and overfitting is negligible.

During training, we used a *model checkpoint* at the end of every epoch to save the model parameters only if the validation loss was less than the loss from the previous epoch. This method guarantees to preserve the best observed model in terms of validation loss. Particularly, the best model in Figure 4 was observed during epoch 48 and had a final loss of 23091 and 27170 in the training and validation sets, respectively.

For training, all images were resized using a ratio of 1:16 in order to fit them into memory and make training faster.

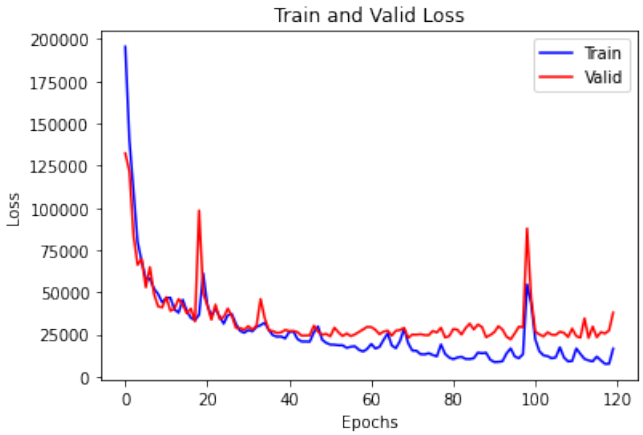

Fig. 5. Model trained for 120 epochs.

B. Model Selection

For our final model, we trained using batch sizes of 8 images during a total of 50 epochs and using an *Adam* optimizer with a learning rate of 0.001. These hyperparameters were determined experimentally by trial and error. Figure 4 shows the training and validation loss of our final model using these hyperparameters.

For reference, Figure 5 shows the model trained for 120 epochs with a greater learning rate. Note here that the model presents over-fitting after the 50th epoch, and spikes are noticeable during training. We can then observe a great improvement when comparing these plots with the plots in Figure 4, where the optimal hyperparameters where used.

IV. Evaluation

Our final model has a *Sigmoid* activation in the final layer, making the output to have a range of values between 0 to 1, which represent the probability of each pixel being part of a crop-damaged section. Therefore, to have a binary image, we kept all pixels with probability greater or equal to 0.5. Figure 6 shows an example of this reconstruction process.

Once we have binary images or mask images, we can measure the model performance using the metrics described in the following paragraph.

We used overlap based metrics to compare our results to the ground truth provided. We considered the *Jaccard Index*, also known as IoU, and the *Dice similarity coefficient* as ranking metrics for our approach.

The IoU is defined by the area of overlap between two sets divided by the area of union of those sets [1]. Dice similarity coefficient is calculated by dividing the intersection of two sets by the sum of both sets [1]. Equations 1 and 2 show how to compute such scores, where true positive (tp) is the number of positive pixels correctly predicted; false positive (fp) is the number of negative pixels incorrectly predicted and false negative (fn) is the positive pixels incorrectly predicted.

$$IoU = \frac{tp}{tp + fp + fn} \tag{1}$$

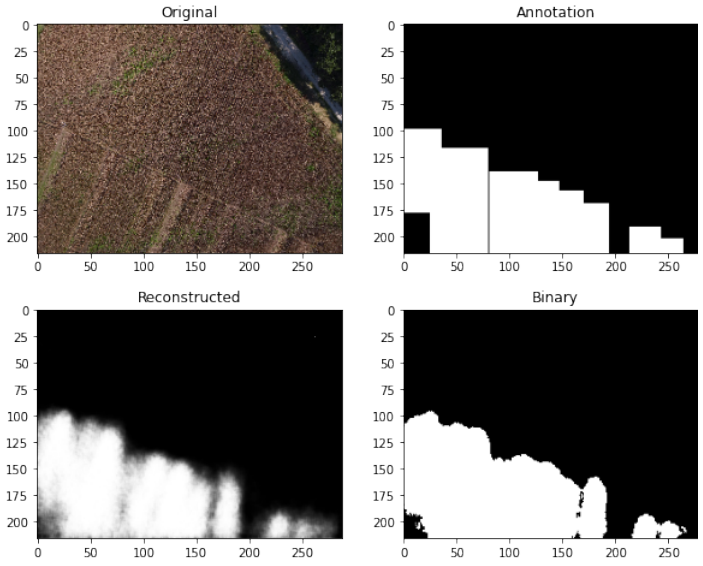

Fig. 6. Image reconstruction.

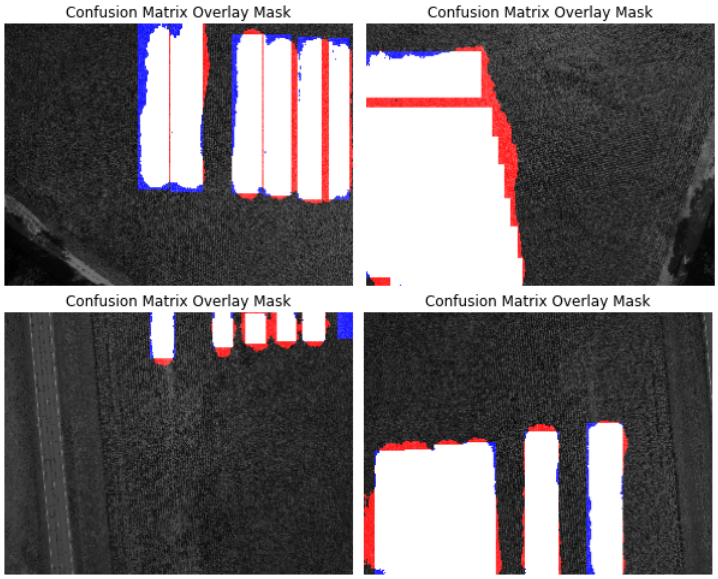

Fig. 7. Predictions on test data set. White = tp, Red = fp, Blue = fn and Black = tn.

$$Dice = \frac{2 * tp}{2 * tp + fp + fn} \tag{2}$$

Tables I and II show the IoU and Dice scores for our baseline and final model, respectively. Our final model scores displayed a better accuracy than the baseline model, which demonstrates that our final model was able to detect crop damage better. Particularly, we improved the IoU score by 15.8% and the Dice score by 8.7% on the test set.

TABLE I
BASELINE MODEL SCORES.

	Training	**Validation**	**Test**
IoU	0.757527	0.716306	0.692813
Dice	0.862038	0.834707	0.818534

TABLE II
FINAL MODEL SCORES.

	Training	**Validation**	**Test**
IoU	0.844780	0.810144	0.802247
Dice	0.915860	0.895115	0.890274

Figure 7 displays some of the test results from our final model. Each image in the figure represents an overlay mask using the confusion matrix that provides tp, fp, tn, and fn. The annotations and predicted results are shown over the gray-scale image they were derived from. The white area represents the accurately detected crop damage (tp), the blue area represents the crop damage that was not detected (fn), and the red area represents falsely detected crop damage (fp).

Note that predictions in Figure 7 are done in our own test set, which consisted of 89 images and their respective annotations. Original project specifications required us to make predictions on an unlabeled data set of 146 images. Although no longer required for this project , such predictions can be reviewed in the project repository [6].

REFERENCES

[1] Taha, A. A., and Hanbury, A. *Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool.* BMC medical imaging vol. 15 29. 12 Aug. 2015.
[2] Vijay Badrinarayanan et. al 2017 *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*
[3] Canny J. *A computational approach to edge detection*. IEEE Transactions on pattern analysis and machine intelligence, 679-698.
[4] *Image Segmentation in 2020: Architectures, Losses, Datasets, and Frameworks*, https://neptune.ai/blog/image-segmentation-in-2020;
[5] Canuma P., *Image Pre-processing*, https://towardsdatascience.com/image-pre-processing-c1aec0be3edf, Oct, 2018.
[6] Project Repository, *CropDamage*, https://github.com/danmenloz/CropDamage