DEEP LEARNING BASED CROP ROW DETECTION REPORT

HARI KRISHNA KAMANA

ABSTRACT

Precision agriculture requires detecting crop rows from video frames in real time, but current deep learning methods, such as U-net, struggle due to a lack of large-scale labeled datasets, the diversity of crops, and the variation in crop appearance at different growth stages. To address this challenge, we use crop row detection system that utilizes U-net's output. Our method refines U-net's results and interpolates the input video stream without requiring retraining. As more labeled data became available, enabling the network to use both local and global features to classify pixels as crop rows, demonstrating consistent performance across the entire dataset with a mean IoU score.

INTRODUCTION

In agriculture, site-specific treatments such as applying herbicides and fertilizers are often done manually, which is costly, inefficient, and can result in poor yield due to delayed treatment. Precision and autonomous agriculture technologies are being developed to replace manual labor, with agricultural sprayers being a vital component. However, to ensure safe and optimal navigation, crop row detection is necessary. This involves identifying crop rows from image/video data, which is a machine learning and computer vision task. However, crop row identification faces several challenges, including poor image quality due to outdoor lighting conditions and camera motion, high weed density, varying crop row widths and growth stages, inconsistent pixel densities, and imperfections in the field. Overcoming these challenges is essential for successful crop row detection and for enabling the autonomous navigation of agricultural vehicles in crop fields.

DATA SET AND MODELING

Both train and test images are contained in the Images folder. Each image is originally a (320, 240) RGB JPEG image. They are converted to numpy array and then flattened to 1D array which contains pixel information. The labels of the training data are contained in train_labels folder. The test labels are also (320, 240) size image with 3 channels but only contains two types of pixel, 0 and 255.

```python
import numpy as np
import cv2
import os

# os.mkdir('/content/drive/MyDrive/Kaggle/train_label_images')

dir_path = '/content/drive/MyDrive/Kaggle/train_labels'
image_dir_path = '/content/drive/MyDrive/Kaggle/train_label_images'
for file in os.listdir(dir_path):
  # print(os.path.join(dir_path, file))
  filename = os.path.splitext(file)[0]
  img_array = np.load(os.path.join(dir_path, file))
  cv2.imwrite(os.path.join(image_dir_path, filename + '.JPG'), img_array)
```

MODELING

At the base of this approach our method for crop row detection uses a supervised learning method, which given an image classifies a region of image pixels into either of two classes, crop-row or background; such a task is commonly known as semantic segmentation. We use a simple U-net based neural network architecture for solving this task. The architecture of the U-Net is kept light in terms of number of layers and trainable parameters so as not to overfit to the small training set available.

The deep neural network we use for crop row identification follows a U-Net architecture with skip connections between encoder and decoder layers. We limited the size of the network to 4 encoder and decoder blocks to avoid overfitting, as our labeled dataset is small. The encoder blocks extract features from the input through convolution operations, batch normalization, and ReLU activation. The output of each encoder block has more feature maps but retains the spatial dimensions. Max pooling is used to increase the receptive field, and a pooling layer halves the spatial dimensions.

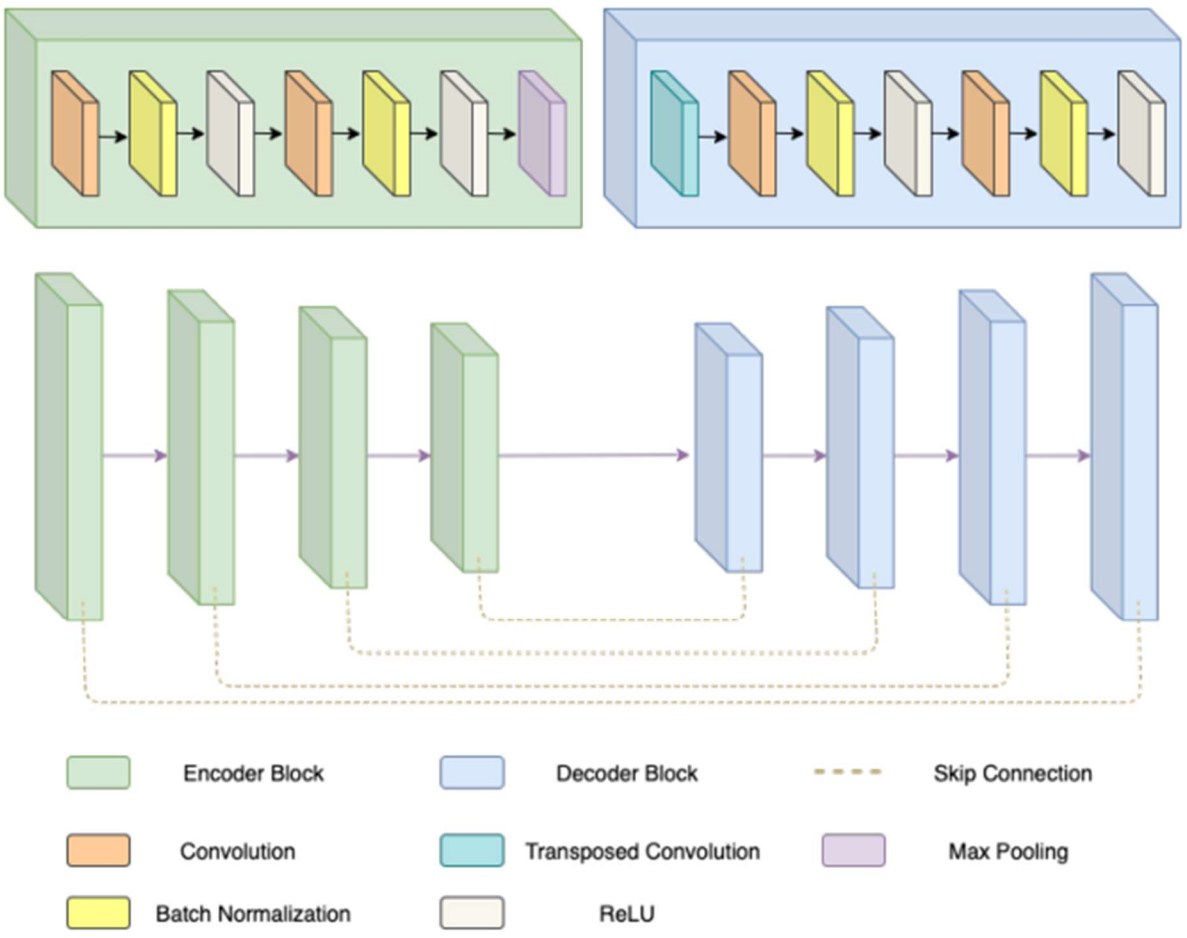
In the U-net architecture, the ith encoder block takes an input tensor with dimensions f x h x w, where f represents the number of feature maps in the input and h and w represent the image dimensions. The encoder block uses convolution operations, batch normalization to reduce covariate shift, and ReLU activation for non-linearity to extract features from the input. The output of the through a max pooling layer, which reduces the spatial dimensions to h/2 and w/2. This resulting output is referred to as encoding. encoder block expands the number of feature maps but maintains the spatial dimensions h and w. To increase the receptive field of the Convolutional Neural Network, the output of the ReLU activation is passed

The input-output relation of an encoder block can be shown by the following equation:

$$output_{\mathrm{i}} = ReLU[BatchNorm2d\{Conv2d(input_{\mathrm{i}})\}]$$

After applying convolution operations and batch normalization to reduce covariate shift, the encoder block of the U-Net expands the number of feature maps while maintaining the spatial dimensions of the input image data. To increase the receptive field of the network, a max pooling operation is performed, followed by passing the output through a pooling layer that reduces the spatial dimensions by half.

$$encoding_{\mathrm{i}} = MaxPool(output_{\mathrm{i}})$$



Decoder consists of a down sampling path made up of four blocks (green) and the encoder consists of an up sampling path made up of four blocks (blue). Skip connections are added between corresponding encoder-decoder blocks to pass on features learned by the encoder to the decoder.

The jth decoding blocks perform a similar operation as shown in but is preceded by the transposed convolution. This reverts the spatial dimensions of the input to its original size thus completing one stage of the decoding operation. decodingj = T ransposedConv(inputj) outputj = ReLU[BatchNorm2d{Conv2d(decodingi)}] .For passing information about learned features between encoder and decoder blocks, skip connections are used as can be seen in Figure . The skip connections denote a concatenation operation where the output of encoder block and the input to the decoder block are concatenated along the first axis before being passed through the decoder block. If xi is the output of the ith encoder block, yj is the input to the jth decoder block and j = N − i where N is the total number of decoder blocks, then a skip connection between these two blocks exist to perform the following operation

LOSS FUNCTION

We trained the network for 20 epochs. The loss function chosen to be minimized was the dice loss between the neural network's predictions and the ground truth.

The cross entropy loss has limitations in detecting clear boundaries, so we opted for the dice loss instead. The statistical distribution of labels in the input training data greatly affects the training accuracy when using cross entropy loss. Weighted cross entropy can help, but it does not solve the problem entirely. We used a mini batch gradient descent algorithm with the Adam optimizer for training, and found that a batch size of 16 was sufficient for convergence while also fitting within our GPU memory constraints.

The U-Net model uses binary cross-entropy loss as its loss function, which is also known as negative log-likelihood loss or log loss. This loss function measures how different the predicted segmentation mask is from the ground truth segmentation mask. The loss function is calculated by taking the average of the cross-entropy loss for all pixels in the segmentation mask.

TRAINING AND VALIDATION DATA SET

To effectively train the U-Net model, it is necessary to have a large dataset with annotations for ground truth. This dataset should be divided into training and validation sets to prevent overfitting. During each epoch of training, the model is trained on the training set and evaluated on the validation set. The model's performance is assessed using mean Intersection over Union (mIoU) and the mIoU on the validation set is calculated at the end of each epoch to evaluate the model's improvement on unseen data. To prevent overfitting, early stopping is implemented based on the performance of the model on the validation set. Specifically, if there is no improvement in mIoU for a certain number of epochs,

training is stopped early to prevent further overfitting. To increase the size of the dataset and improve generalization performance, data augmentation techniques such as random flipping and rotation are applied during training.

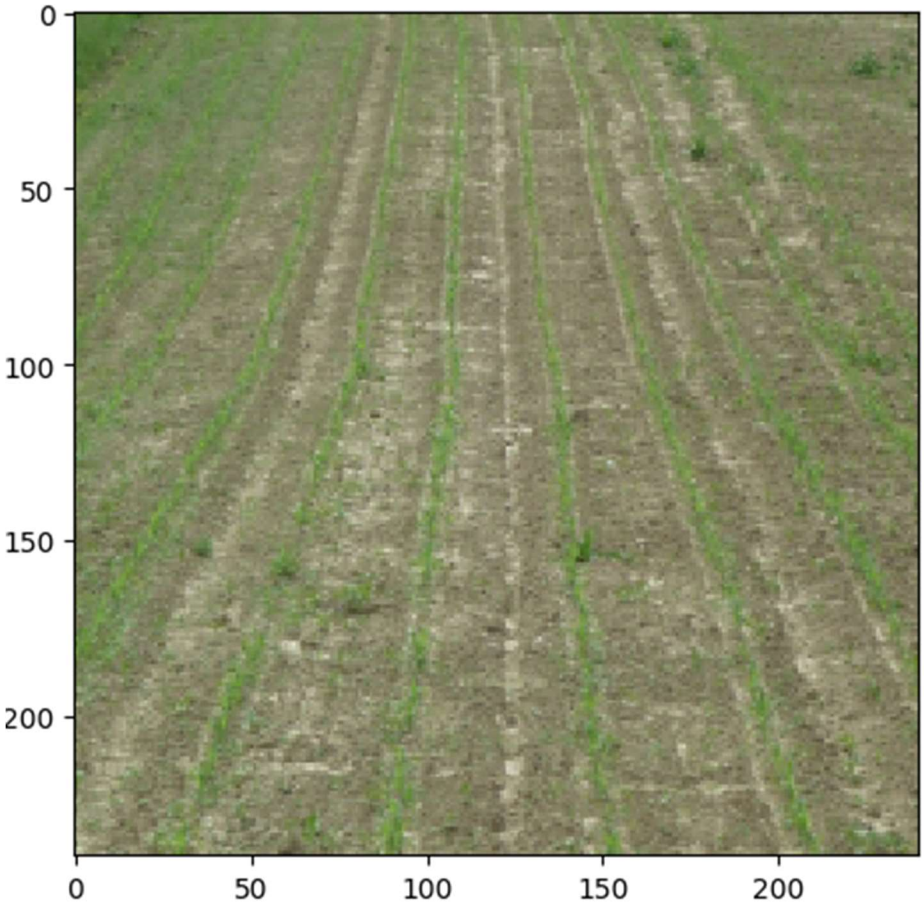
```
X_test = np.zeros((len(img_list_X_test), img_height, img_width, img_channels), dtype=np.uint8)
for i, img_path in enumerate(img_list_X_test):
    # read image
    img_test = imread(img_path)
    img_test = resize(img_test, (img_height, img_width, img_channels), mode='constant', preserve_range='True')
    X_test[i] = img_test
```

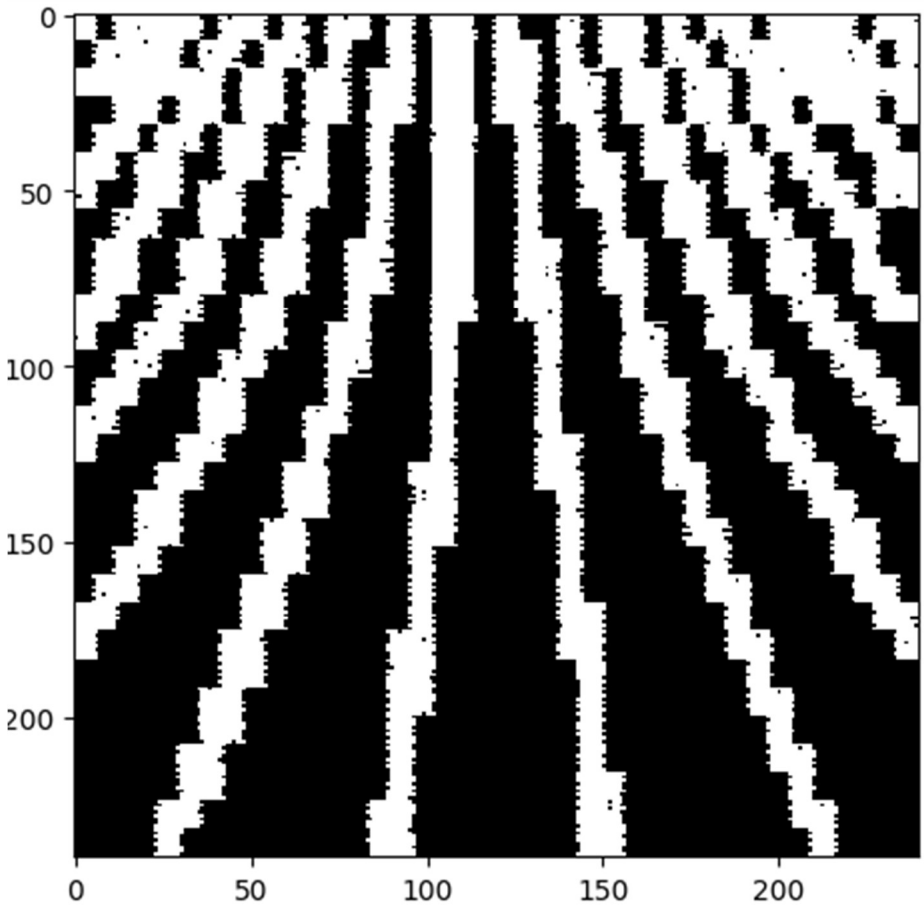
```
#Model checkpoint
checkpointer = tf.keras.callbacks.ModelCheckpoint('model_for_nuclei.h5', verbose=1, save_best_only=True)

name = "SqueezeUnet-{}".format(int(time.time()))
callbacks = [
        tf.keras.callbacks.EarlyStopping(patience=3),
        tf.keras.callbacks.TensorBoard(log_dir='tensorboard/{}'.format(name))]

results = model.fit(X_train, y_train, validation_split=0.1, batch_size=4, epochs=20, callbacks=callbacks)
```

To assess the effectiveness of the crop row detection models, mean Intersection over Union (mIoU) was used as a performance metric. The mIoU measures the degree of overlap between the predicted crop rows and the actual (ground truth) crop rows. Conversely, the Dice loss measures the dissimilarity between the predicted and actual crop rows. To compute the mIoU, the intersection and union of predicted and actual crop rows were computed for each image, and the average of their intersection-over-union values over all images in the test set was calculated. The mIoU ranges between 0 and 1, with higher values indicating better model performance.

RESULT

To evaluate the performance of the crop row detection model, one can use standard metrics such as precision, recall, F1-score, and the Dice coefficient. Additionally, inspecting the predicted segmentation masks visually can provide insights into the model's accuracy and consistency. The model can be an effective tool for crop row detection in precision agriculture, but its performance depends on various factors, including the quality and size of the training dataset, the choice of hyperparameters, and the optimization algorithm used. Therefore, it is crucial to carefully choose and optimize these factors to achieve the best performance possible for crop row detection with the model.


0
50
100
150
200
0
50
100
150
200



CONCLUSION

Image segmentation is a useful method for analyzing images in fields such as medical imaging, remote sensing, robotics, and self-driving cars. Crop row detection using supervised learning and U-Net architectures demonstrates the potential of machine learning and computer vision techniques to enhance agriculture and optimize crop yield. The results showed that this method achieved better performance than existing approaches in terms of accuracy and speed of inference. Moreover, the proposed technique demonstrated the ability to adapt to different image domains in real-time when labeled training data is limited, indicating its online domain adaptation capabilities.