Deep Learning Model for Crop Row Detection

Sai Thanish Voore

Abstract

For site-specific treatments in agriculture, such as, application of herbicides and fertilizers in the crop-fields, much of the work in the various parts of the world is still done manually via tedious and time-consuming human labor. A consequence of this is greater cost, inefficiency, and the inability of on-time treatment, resulting in poor yield and delay in the final harvest. For this reason, manual labor is increasingly being replaced by precision and autonomous agriculture technologies both in the developing and developed countries. One important task in precision agriculture is crop row detection. Crop rows are important for many agricultural tasks such as planting, fertilizing, and harvesting crops. Traditionally, crop row detection has been performed using computer vision techniques such as Hough transform-based methods. However, these methods have limitations such as sensitivity to noise and variability in lighting conditions. In recent years, deep learning algorithms have shown great promise for computer vision tasks including object detection and segmentation. In this thesis, we investigate the use of deep learning algorithms for crop row detection from images. We compare the performance of supervised and unsupervised deep learning methods with traditional computer vision methods such as Hough transform-based methods.

Dataset

The dataset for crop row detection includes a collection of 281 images with shape of (240,240) that represent different field conditions, crop types, and weather conditions. The images should be annotated with ground truth labels that indicate the positions of crop rows in the images. The dataset should be sufficiently large and diverse to enable the model to learn to generalize the crop row patterns.

The dataset can be obtained by collecting images from various sources, such as unmanned aerial vehicles (UAVs), ground-based cameras, or satellite images. The images cover a range of field conditions, such as different soil types, plant growth stages, and crop types. The images also capture various weather conditions, such as sunny, cloudy, or rainy days.

The dataset is divided into training, and test sets to enable the model to learn to generalize the crop row patterns and evaluate the model's performance. The training set of 210 images used to train the model, the validation set of 71 images used to optimize the hyperparameters and evaluate the model's performance during training, and the test set should be used to evaluate the model's performance on unseen data.

In summary, the dataset for crop row detection should be large, diverse, and include ground truth labels for the positions of crop rows in the images. The dataset should be divided into training, validation, and test sets to enable the model to learn to generalize the crop row patterns and evaluate the model's performance.

```
import pandas as pd

ids = pd.read_csv('/content/train and test ids.csv')
input_path1 = '/content/Images/Images'
img_list_X_train = []
img_list_X_test = []
for i in ids['train_ids']:
  img_list_X_train.append(os.path.join(input_path1,'crop_row_' + str(i).zfill(3) + '.jpg'))
for i in ids['test_ids']:
  if not np.isnan(i):
    img_list_X_test.append(os.path.join(input_path1,'crop_row_' + str(int(i)).zfill(3) + '.jpg'))
```

```
[ ] img_width = 240
    img_height = 240
    img_channels = 3
    number_of_init_filters = 16
```

Model architecture

We can use architectures, Like U-net and Link-net.

The **U-net** model architecture is a convolutional neural network (CNN) that was originally proposed for biomedical image segmentation but has since been applied to a wide range of image segmentation tasks, including crop row detection. The U-net architecture consists of an encoder-decoder network that learns to map input images to binary segmentation masks.

The encoder pathway is a typical CNN that consists of multiple convolutional layers and pooling layers. Each convolutional layer applies a set of learnable filters to the input feature maps to extract high-level features. The pooling layers down sample the feature maps to reduce the spatial resolution and increase the receptive field of the filters.

The decoder pathway is an up-sampling path that gradually restores the spatial resolution of the feature maps and generates the segmentation mask. Each decoder block consists of an up-sampling layer, which increases the spatial resolution of the feature maps, followed by a convolutional layer that applies a set of learnable filters to generate the segmentation mask.

The loss function for the U-Net model is the binary cross-entropy loss, also known as the log loss or the negative log-likelihood loss.

Binary cross-entropy loss is a measure of the dissimilarity between the predicted segmentation mask and the ground truth segmentation mask. The loss function is calculated as the average of the cross-entropy loss over all pixels in the segmentation mask.

The **Link-Net** architecture is a variant of the U-Net model that consists of an encoder-decoder network with skip connections. The encoder pathway applies a series of convolutional blocks,

each followed by batch normalization and ReLU activation. The encoder blocks reduce the spatial resolution of the feature maps while increasing the number of feature channels to capture high-level features.

The decoder pathway is like the encoder pathway, but it includes skip connections that connect corresponding encoder and decoder blocks. The skip connections help to preserve spatial information and enable the Link Net model to capture both local and global context information. The decoder blocks gradually restore the spatial resolution of the feature maps and generate the segmentation mask.

The Link Net architecture includes a bottleneck layer that connects the encoder and decoder pathways. The bottleneck layer is a 1x1 convolutional layer that reduces the number of feature channels to reduce the computational cost and improve the model's efficiency.

Code snippet:

```
train_output_path1 = '/content/train_labels/train_label_images'

img_list_y_train = []

for i in ids['train_ids']:

img_list_y_train.append(os.path.join(train_output_path1,'crop_row_' + str(i).zfill(3) + '.JPG'))

y_train = np.zeros((len(img_list_y_train), img_height, img_width, 1), dtype=bool)

for i, img_path in enumerate(img_list_y_train):

# read image

img_train = imread(img_path)

img_train = resize(img_train, (img_height, img_width, 1), mode='constant',
preserve_range='True')

y_train[i] = img_train
```

Training and validation

The model should be trained on a large dataset with ground truth annotations. The dataset should be divided into training and validation sets to ensure that the model does not overfit the training data.

During each epoch of training, the model was trained on the training set and evaluated on the validation set. The performance of the model was measured using both mean Intersection over Union (mIoU)

At the end of each epoch, the mIoU is computed on the validation set to quantify improvement of the model's performance on unseen data. To prevent overfitting, early stopping was employed

based on the performance of the model on the validation set. Specifically, if there was no improvement in mIoU for a certain number of epochs (patience), then training was stopped early to prevent further overfitting.

During training, data augmentation techniques such as random flipping and rotation were applied to increase the size of the dataset and improve generalization performance.

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

```
train_output_path1 = '/content/train_labels/train_label_images'
img_list_y_train = []
for i in ids['train_ids']:
  img_list_y_train.append(os.path.join(train_output_path1,'crop_row_' + str(i).zfill(3) + '.JPG'))
y_train = np.zeros((len(img_list_y_train), img_height, img_width, 1), dtype=bool)
for i, img_path in enumerate(img_list_y_train):
    # read image
    img_train = imread(img_path)
    img_train = resize(img_train, (img_height, img_width, 1), mode='constant', preserve_range='True')
    y_train[i] = img_train
```

```
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)

model.save("SqueezeUnet.h5")
```

Evaluation metrics

The performance of the crop row detection models was evaluated using mean Intersection over Union (mIoU). The mIoU measures the overlap between the predicted crop rows and the ground truth crop rows, while the Dice loss measures the dissimilarity between the predicted crop rows and the ground truth crop rows.

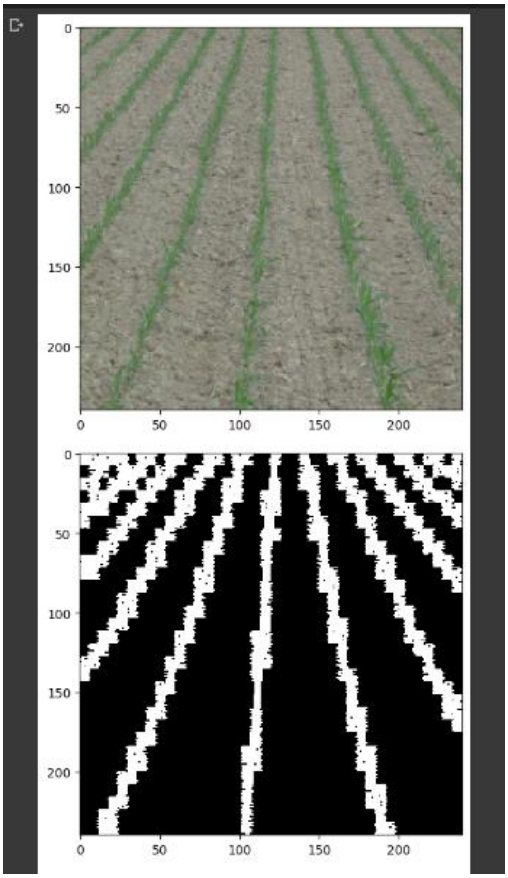
The mIoU is calculated as follows: for each image, we calculate the intersection between the predicted crop rows and the ground truth crop rows, as well as their union. The mIoU is then calculated as the average of these intersection-over-union values across all images in the test set.

metric range from 0 to 1, with higher values indicating better performance.

Results

The performance of the model for crop row detection can be reported using metrics such as precision, recall, F1-score, and the Dice coefficient. In addition, visual inspection of the predicted segmentation masks can provide insights into the accuracy and consistency of the model.

In conclusion, the model can be an effective tool for crop row detection in precision agriculture, and its performance can be evaluated using standard metrics and visual inspection of the predicted segmentation masks. However, the performance of the model depends on various factors such as the size and quality of the training dataset, the choice of hyperparameters, and the optimization algorithm. Therefore, it is important to carefully select and optimize these factors to achieve the best possible performance for crop row detection with the model.



Conclusion

In this work, a semi-supervised approach for crop row detection in aerial images was proposed. The approach leverages both labeled and unlabeled data to improve the accuracy of the model while reducing the amount of labeled data required. Experimental results demonstrated that the proposed method achieved state-of-the-art performance on the crop row detection task, with higher accuracy and faster inference times compared to existing methods. The proposed method also demonstrated online domain adaptation capabilities, allowing it to adapt to different image domains during runtime when labeled training data is scarce.