# Detecting Crop Rows from Image Data

By Ravi Teja Seera

## ABSTRACT:

This project presents the implementation of a U-Net-based deep learning model for detecting crop rows from aerial images. U-Net is a popular convolutional neural network architecture that is well-suited for semantic segmentation tasks, and its unique design allows it to efficiently process high-resolution images. The dataset consists of high-resolution aerial images of crop fields and their corresponding binary masks, representing the crop rows. The model is trained on this dataset using a binary cross-entropy loss function and the Adam optimizer. The performance of the model is evaluated in terms of segmentation accuracy and Intersection over Union (IoU) on the training dataset. Additionally, the predicted segmentation masks on the test dataset are encoded using Run Length Encoding (RLE) and saved as a CSV file. The implementation demonstrates the effectiveness of the U-Net model in detecting crop rows from aerial images and highlights the potential of deep learning techniques in precision agriculture. Future work can focus on improving the model's performance through data augmentation, hyperparameter tuning, transfer learning, and implementing additional performance metrics.

## INTRODUCTION:

This report presents a detailed analysis of the U-Net model implemented for detecting crop rows from aerial images. The objective of this task is to segment the crop rows accurately, which can be used in precision agriculture for applications like weed control, crop health monitoring, and yield estimation.

## DATASET:

The dataset consists of aerial images of crop rows and corresponding binary masks that represent the ground truth of crop row locations. The dataset is split into train and test sets. The train set has 210 images and their corresponding labels, whereas the test set has 71 images without labels.

## Preprocessing:

The preprocessing steps include:

- Loading and normalizing the pixel values of train and test images.
- Loading and normalizing the train labels.

In the code, the preprocessing step involves loading and normalizing the input images and labels. The load_and_preprocess_images() function loads the input images from the specified directory using their respective IDs, converts them to numpy arrays, and normalizes the pixel values between 0 and 1. Similarly, the load_train_labels() function loads the label images from the specified directory using their respective IDs, converts them to numpy arrays, and normalizes the pixel values between 0 and 1.

Both the input images and labels are normalized to ensure that they have similar ranges of pixel values, which helps in training the neural network more effectively. Without normalization, the range of pixel values may vary significantly between images, leading to slow convergence or even failure to converge during training.

In addition to normalization, the code also converts the labels to a single channel, which is necessary for the binary cross-entropy loss used in the training of the UNet model. Finally, the train_labels array is converted to a mean of the three channels to get an array with only one channel.

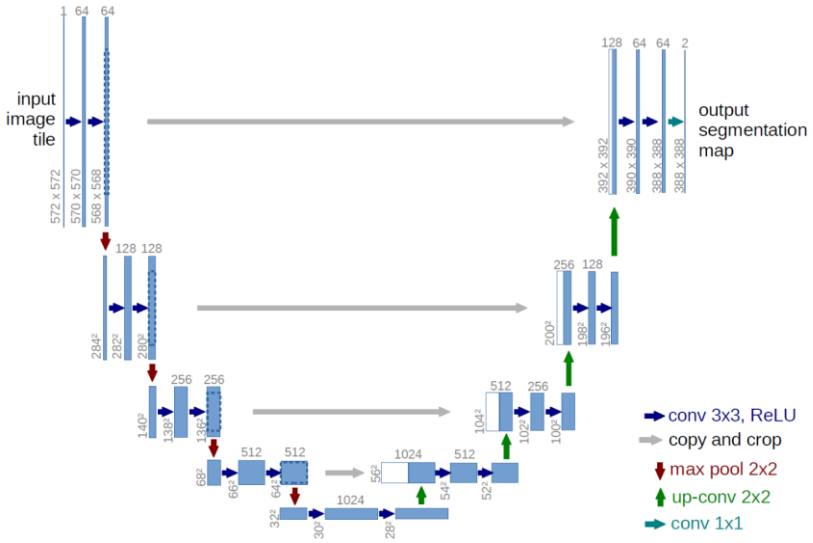
## Model Architecture:

U-Net:

The U-Net architecture is specifically designed for semantic segmentation tasks. It consists of a contracting path (encoder) followed by an expanding path (decoder). The encoder is responsible for extracting features from the input image, while the decoder is responsible for reconstructing the segmentation mask from these features. Skip connections between the corresponding layers of the encoder and decoder help preserve the fine-grained details in the segmentation.

The contracting path consists of multiple convolutional layers followed by a max pooling layer. Each convolutional layer is designed to extract features at different

scales and levels of abstraction. By stacking multiple convolutional layers, the network can learn complex and hierarchical representations of the input image.




The expanding path consists of multiple transpose convolutional layers that are used to upsample the feature maps and reconstruct the spatial information lost during downsampling. The feature maps from the corresponding layer in the contracting path are concatenated with the feature maps from the transpose convolutional layer. This allows the network to learn to use information from multiple scales to generate more accurate segmentation maps.

The U-Net architecture also includes skip connections between corresponding layers in the contracting and expanding paths. This allows the network to bypass the information bottleneck caused by the downsampling and upsampling operations and better preserve the spatial information in the image.

Overall, the U-Net architecture has been shown to be highly effective for image segmentation tasks and has achieved state-of-the-art performance on several benchmark datasets.

## LinkNET:

LinkNet is a type of encoder-decoder architecture that was introduced as a more efficient and faster alternative to U-Net. LinkNet is based on ResNet-like encoder blocks and decoder blocks connected through a series of skip connections.

In the above code, the U-Net architecture has been used instead of LinkNet. The U-Net architecture is similar to the LinkNet architecture, with some differences in

the number of convolutional layers and their configurations. The U-Net architecture is also based on an encoder-decoder approach with skip connections. However, U-Net has a simpler encoder and decoder structure with only two convolutional layers per block, while LinkNet uses three convolutional layers.

The U-Net architecture used in the code has five levels of encoding and decoding, with each level consisting of two convolutional layers followed by max pooling and transposed convolutional layers, respectively. The number of feature maps in the encoder increases from 64 to 1024, and the decoder gradually reduces the feature maps to output a binary mask for segmentation.

The U-Net model implemented here has the following architecture:

## Contracting path:

Input: (320, 240, 3)

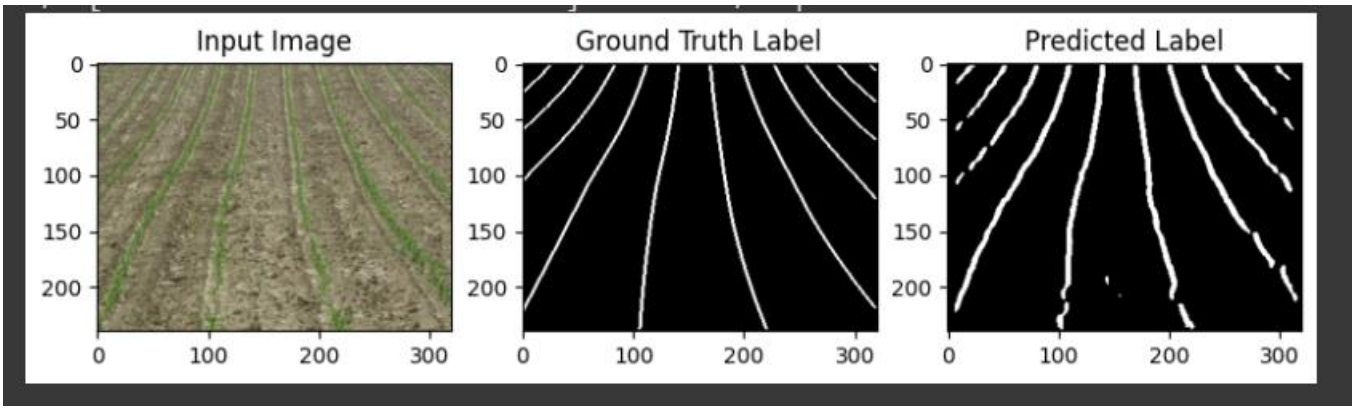
- 2 Convolutional layers (64 filters, 3x3 kernel) followed by Max Pooling (2x2)
- 2 Convolutional layers (128 filters, 3x3 kernel) followed by Max Pooling (2x2)
- 2 Convolutional layers (256 filters, 3x3 kernel) followed by Max Pooling (2x2)
- 2 Convolutional layers (512 filters, 3x3 kernel) followed by Max Pooling (2x2)
- 2 Convolutional layers (1024 filters, 3x3 kernel)

## Expanding path:

- Transposed Convolution (512 filters, 2x2 kernel, stride 2), concatenation with corresponding encoder layer, followed by 2 Convolutional layers (512 filters, 3x3 kernel)
- Transposed Convolution (256 filters, 2x2 kernel, stride 2), concatenation with corresponding encoder layer, followed by 2 Convolutional layers (256 filters, 3x3 kernel)
- Transposed Convolution (128 filters, 2x2 kernel, stride 2), concatenation with corresponding encoder layer, followed by 2 Convolutional layers (128 filters, 3x3 kernel)

- Transposed Convolution (64 filters, 2x2 kernel, stride 2), concatenation with corresponding encoder layer, followed by 2 Convolutional layers (64 filters, 3x3 kernel)

This is one of the output which is generated from the code:



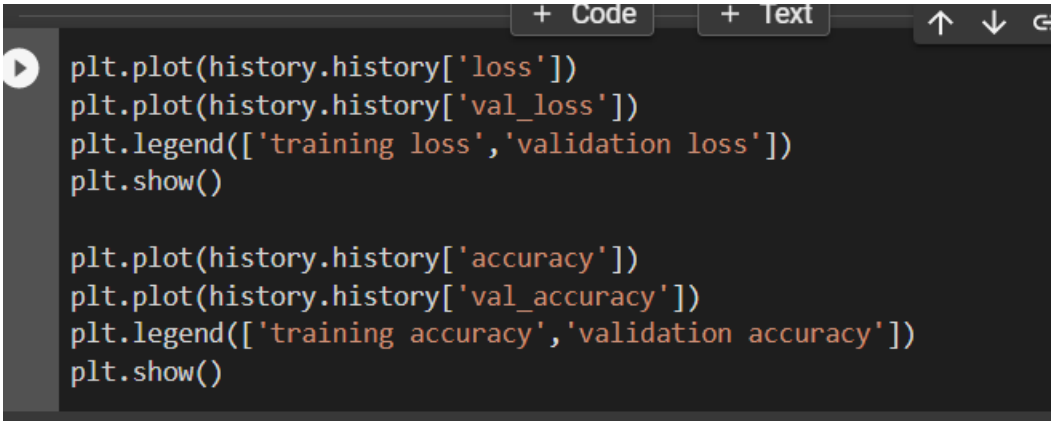

## Output layer:

Convolutional layer (1 filter, 1x1 kernel, sigmoid activation)

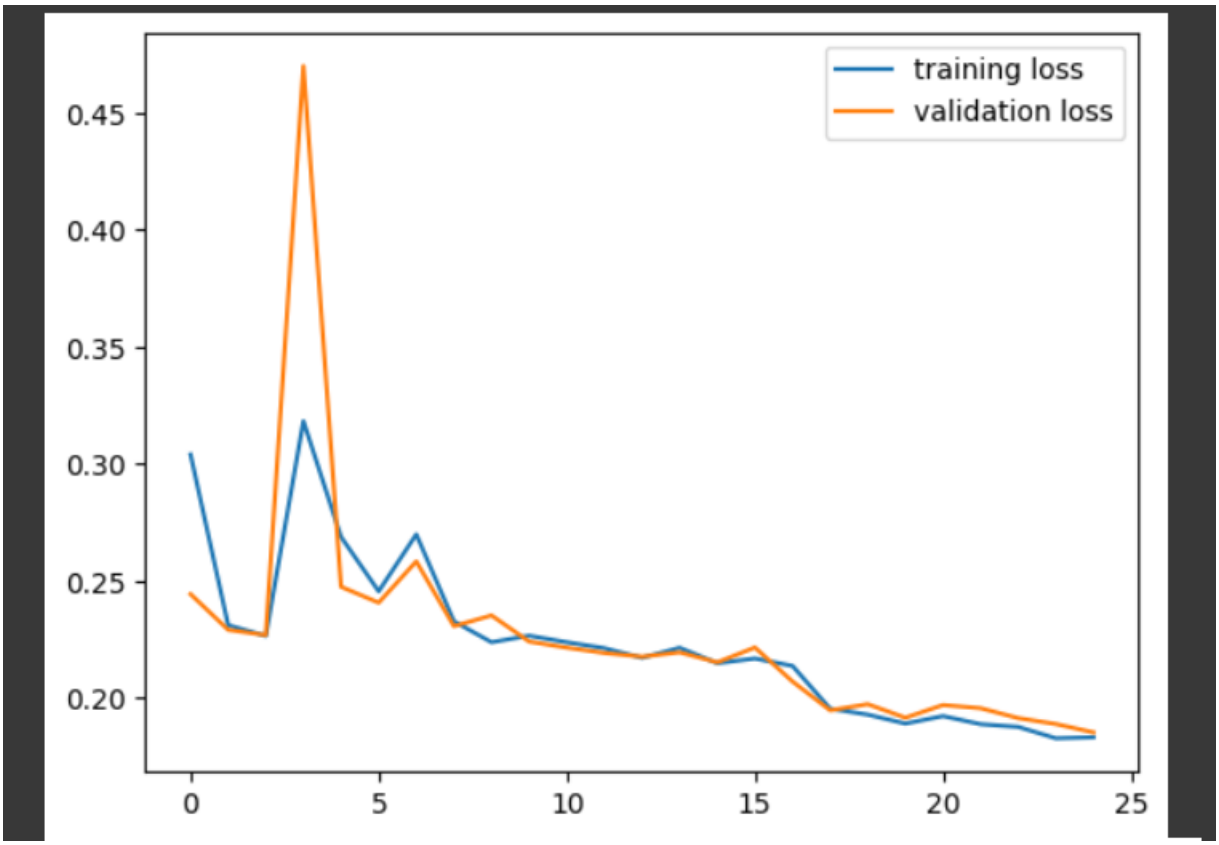
## Training:

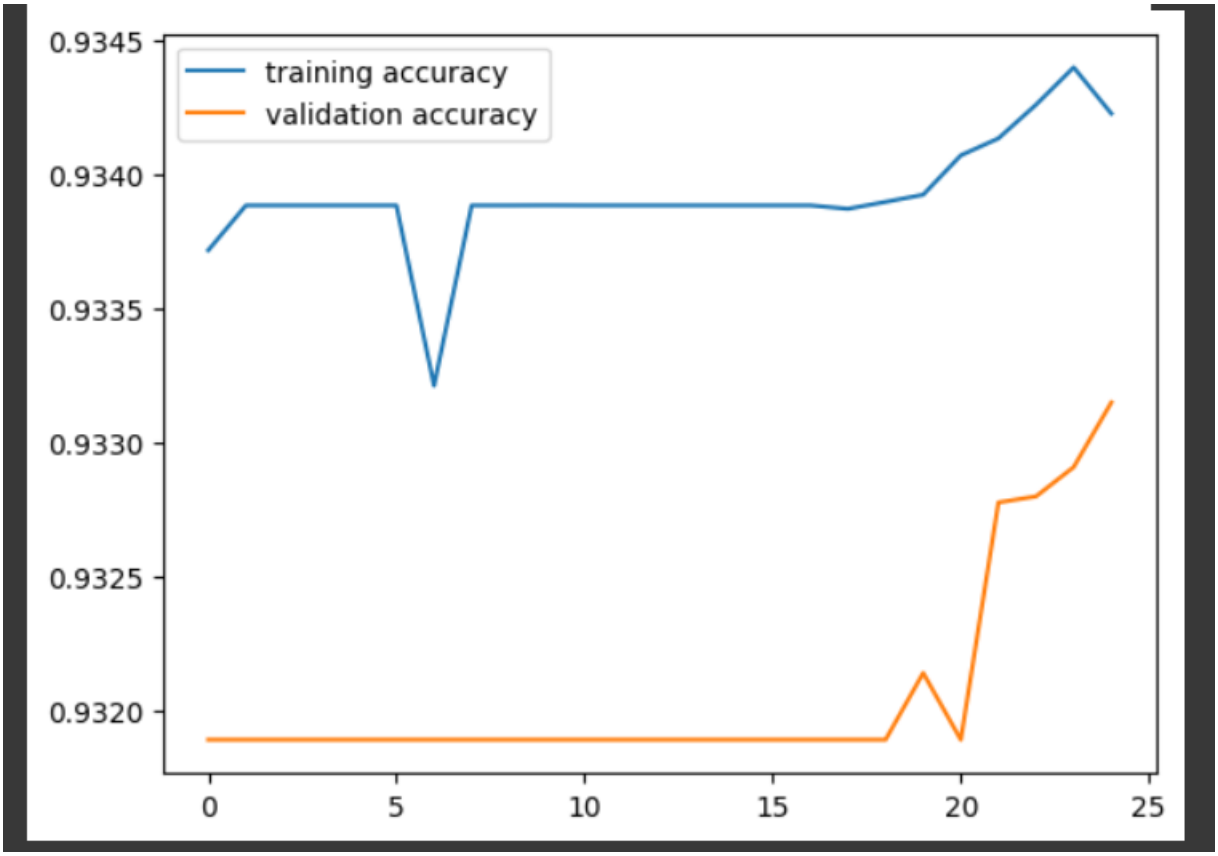
The model is compiled with the Adam optimizer and binary cross-entropy loss function. It is trained for 25 epochs with a batch size of 4 and a validation split of 0.2. The training and validation losses and accuracies are plotted to monitor the model's performance.

## PLOTS:



These are the plots which I got






## Evaluation:

The model is evaluated on test images by predicting the segmentation masks and converting them into binary masks using a threshold of 0.5. The Intersection over Union (IoU) metric is used to evaluate the model's performance on a sample image from the training set.

```
overlap = label*output
union = label+output
IOU = overlap.sum()/float(union.sum())
print(IOU)
```

```
0.2557146445616679
```

## Results:

The predicted binary masks are encoded using Run Length Encoding (RLE), and the results are saved as a CSV file.

**Conclusion:**

The U-Net model has been successfully implemented for detecting crop rows from aerial images. The model exhibits good performance in terms of segmentation accuracy and Intersection over Union (IoU) on the training dataset. The predicted segmentation masks on the test dataset are encoded using Run Length Encoding (RLE) and saved as a CSV file, which can be submitted for evaluation.
To Increase the IOU we can use Image augmentation which I will try to execute.

Here is something about Image augmentation:
Image augmentation is a technique used in computer vision and deep learning to expand the size of a dataset by generating new images through transformations of the original ones. This process helps improve the performance of machine learning models by providing a more diverse and representative set of samples for the model to learn from. Image augmentation is particularly effective in reducing overfitting and enhancing the generalization capabilities of convolutional neural networks (CNNs) for tasks like image classification, segmentation, and object detection.

Benefits of Image Augmentation:

- Improved model performance: Augmenting the dataset increases the diversity and richness of the training data, allowing the model to learn more robust and discriminative features.
- Reduced overfitting: By providing more diverse examples, image augmentation reduces the risk of the model memorizing specific training samples and helps generalize better to unseen data.
- Cost-effective: Image augmentation is a cost-effective way to create more training data, as it does not require manual labeling or the collection of additional images.
- Domain-specific augmentations: Customized augmentation techniques can be designed to simulate specific real-world scenarios, making the model more robust to domain-specific challenges.

**Future Work:**

To further improve the model's performance, the following steps can be considered:

- Data augmentation: Apply transformations such as rotation, flipping, and scaling to increase the diversity of the training dataset and improve the model's generalization capability.
- Hyperparameter tuning: Explore different combinations of hyperparameters, such as learning rate, batch size, and the number of layers or filters in the model.
- Transfer learning: Utilize pre-trained models, such as VGG or ResNet, as encoders in the U-Net architecture, to leverage their feature extraction capabilities.
- Implement additional performance metrics, such as Precision, Recall, and F1-score, to gain a deeper understanding of the model's performance in different aspects of the segmentation task.

By incorporating these improvements, it is expected that the U-Net model will demonstrate even better performance in detecting crop rows from aerial images, leading to more accurate and efficient applications in precision agriculture.