The approach for solving the given problem using these architectures would involve the following steps:

1. Pre-processing of the input images - converting them to numpy arrays, flattening them to 1D arrays, and applying run-length encoding to save space.
2. Splitting the data into training and testing sets.
3. Training the deep CNN architectures on the training set using the input images and their corresponding crop row labels.
4. Evaluating the performance of the trained models on the testing set using the input images and predicting the crop row labels.
5. Converting the predicted crop row labels back to RLE format and submitting them for evaluation.

Model Architecture and implementation details:
In the above implementations:

1. **U-Net**:

- Encoder: 4 Conv2D layers with increasing filters (64, 128, 256, 512) followed by MaxPooling2D layers.
- Decoder: 4 Conv2DTranspose layers with decreasing filters (512, 256, 128, 64) followed by concatenation and Conv2D layers.
- Output: 1 Conv2D layer with a sigmoid activation function.

2. **LinkNet**:

- Encoder: 4 Conv2D layers with increasing filters (64, 128, 256, 512) followed by MaxPooling2D layers.
- Decoder: 3 UpSampling2D layers followed by concatenation and Conv2D layers with decreasing filters (256, 128, 64).
- Output: 1 Conv2D layer with a sigmoid activation function.

3. **SegNet**:

- Encoder: 4 Conv2D layers with increasing filters (64, 128, 256, 512) followed by MaxPooling2D layers and indices stored.
- Decoder: 4 UpSampling2D layers with indices used for pooling followed by Conv2D layers with decreasing filters (512, 256, 128, 64).
- Output: 1 Conv2D layer with a sigmoid activation function.

All models used a batch size of 16, and the number of epochs was set to 25. The optimizer used for all models was the Adam optimizer with default parameters, and the loss function was binary_crossentropy. The metric used for evaluation was accuracy.

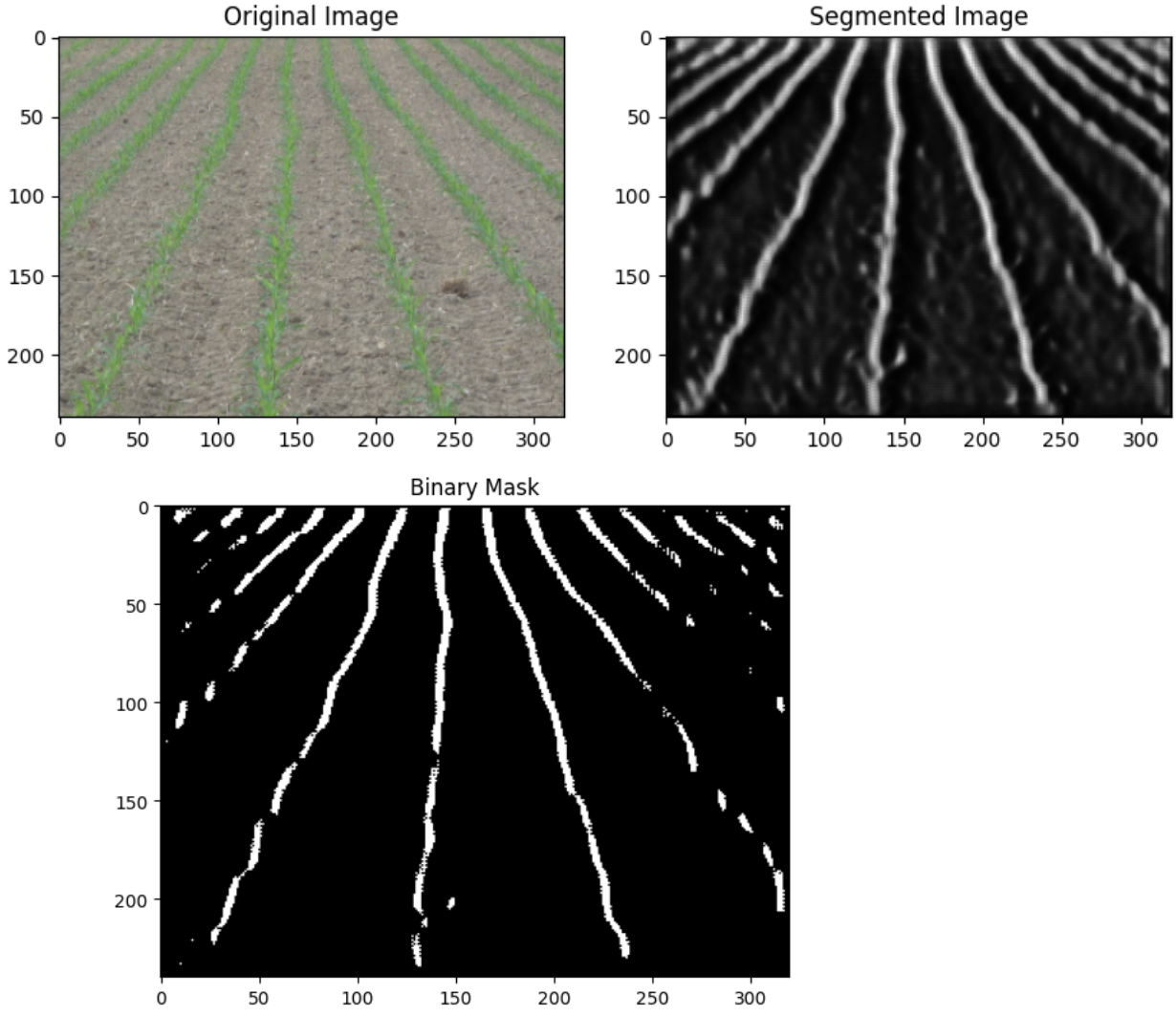
3. Results

We used two strategies for training the model:

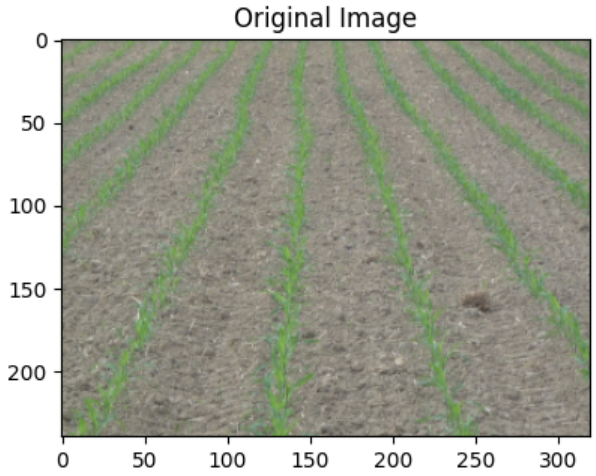
- Using the input data directly
- Using data augmentation for input images

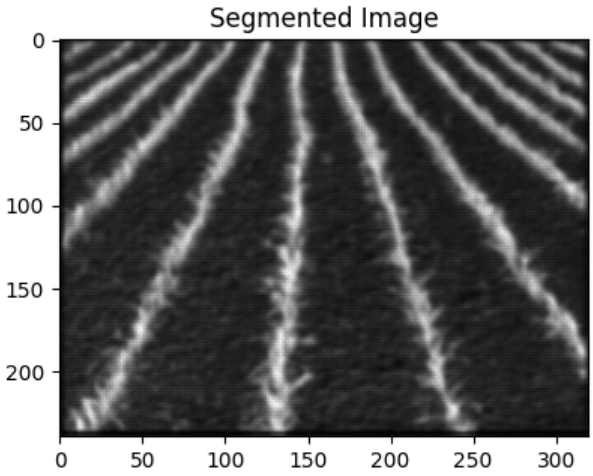
We can observe the following results for our different models:

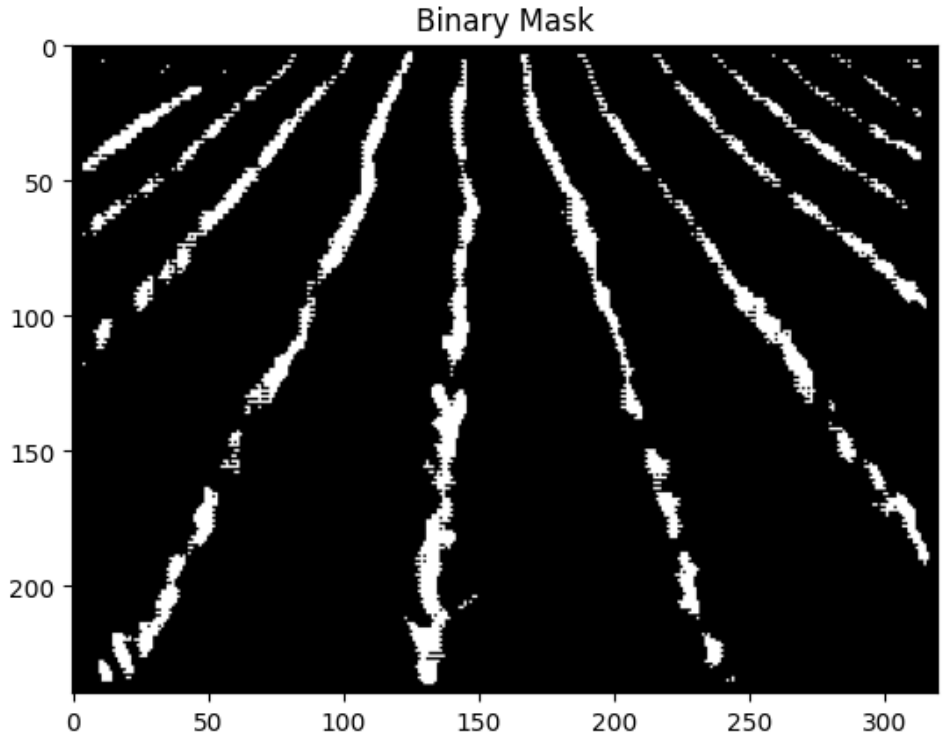
1. Unet Model results:



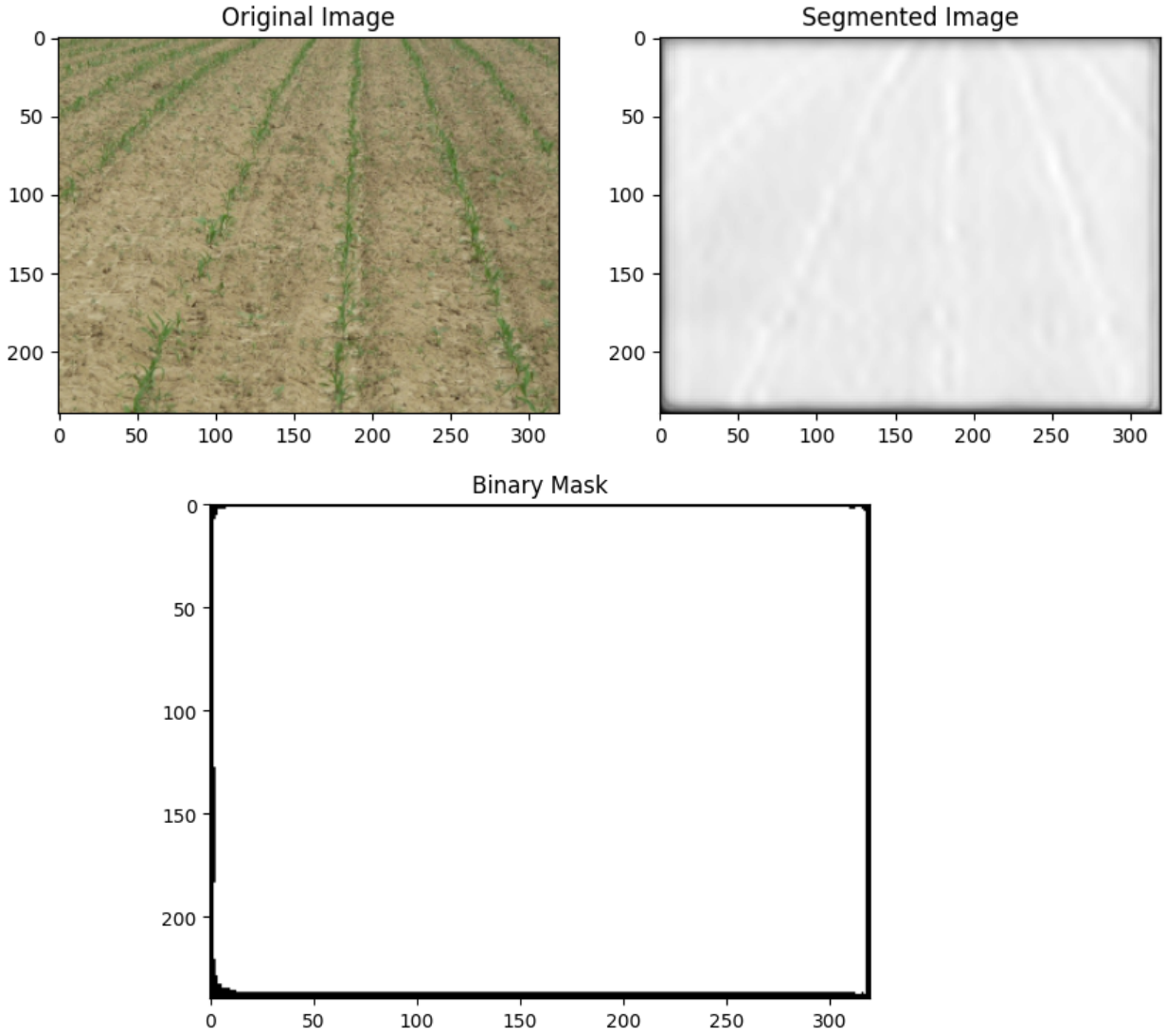
2. Unet Model results with data Augmentation:



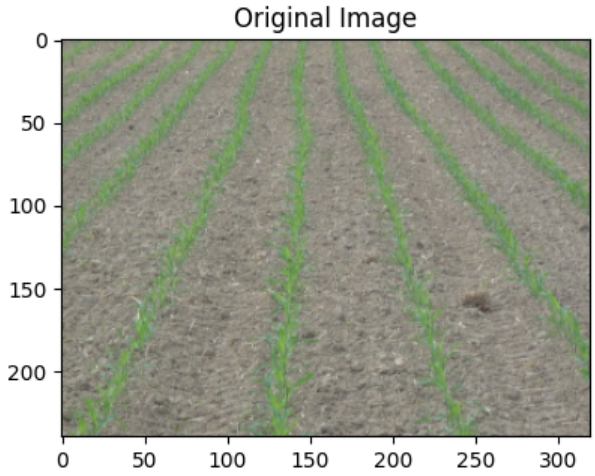


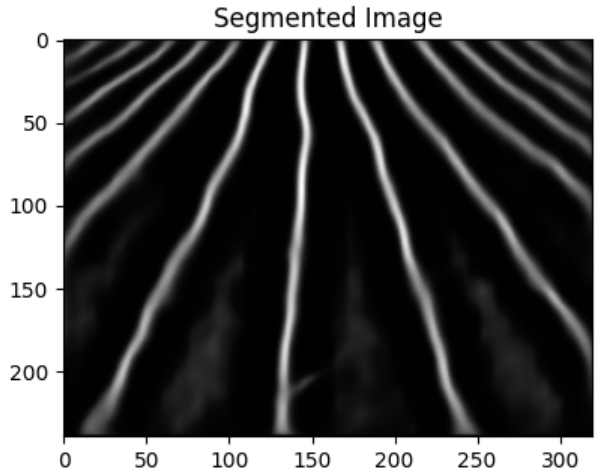


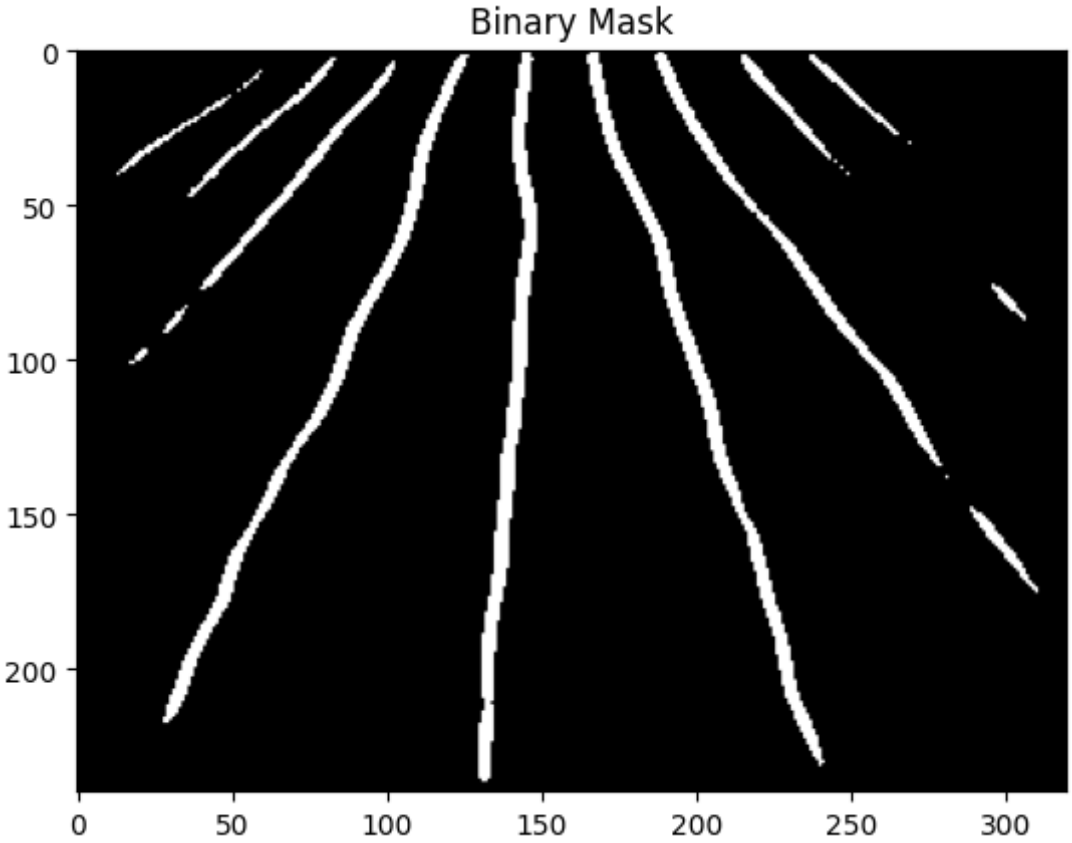
3. Segnet Model Results:



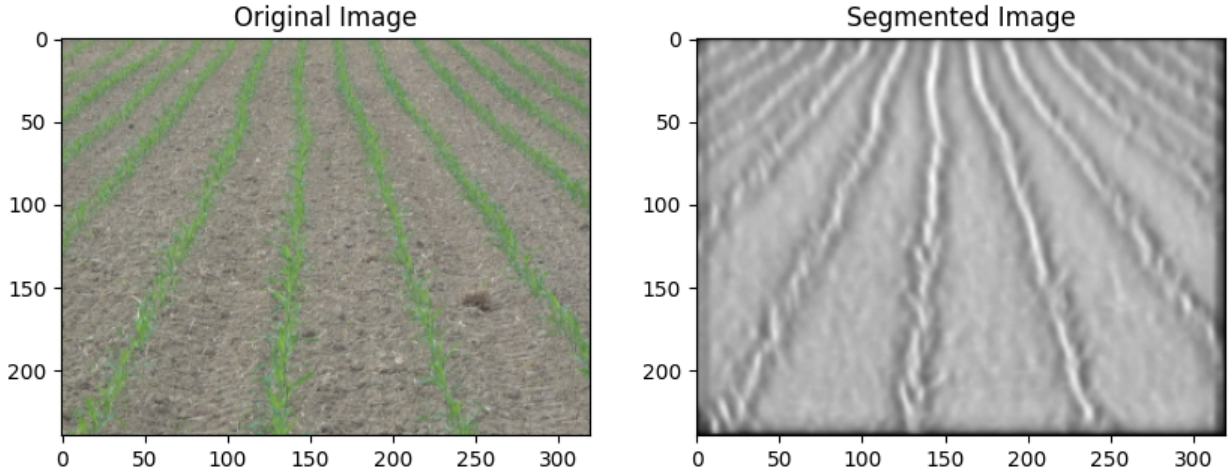
4. Segnet Results with Data Augmentation:

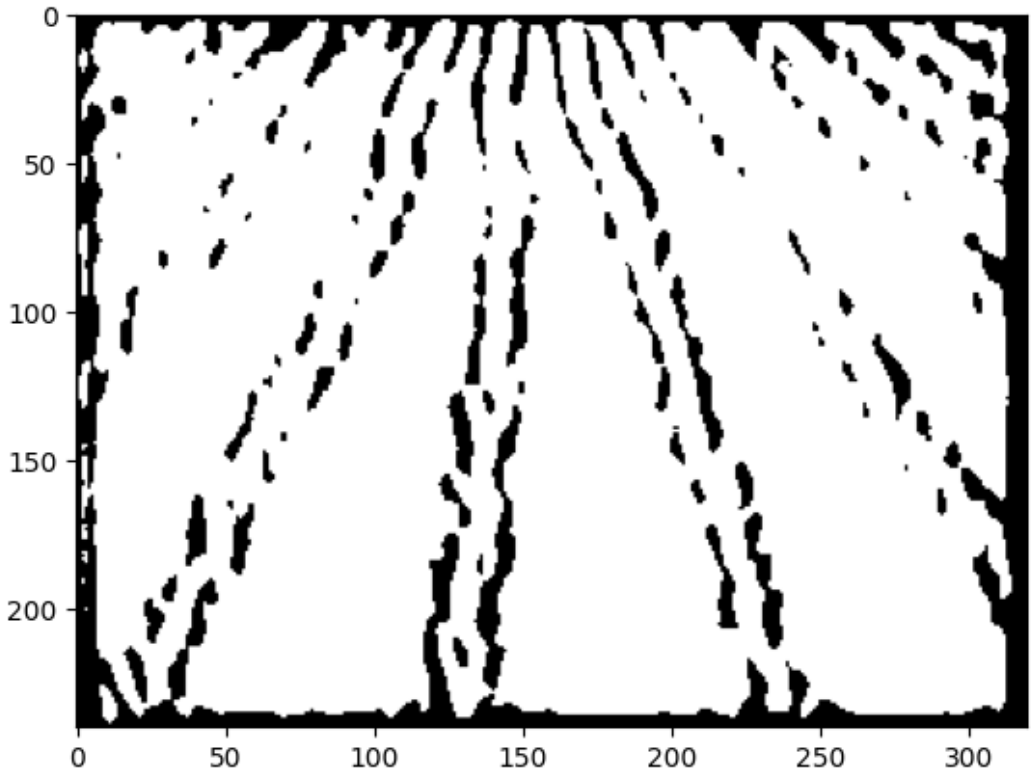




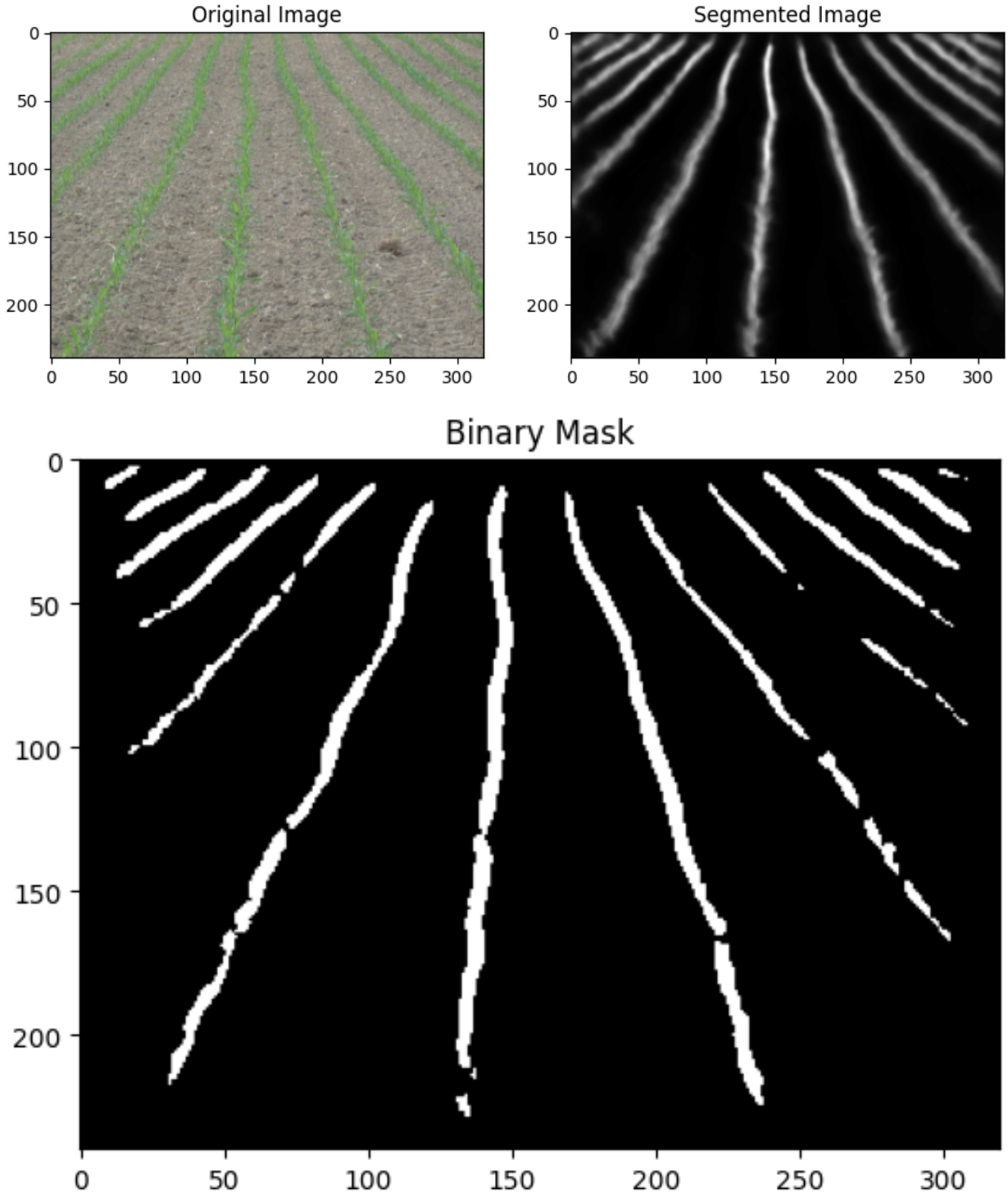


5. LinkNet Results:





6. LinkNet Results with Data Augmentation:



Benefits of Data Augmentation

- Incorporating data augmentation into the training process has significantly improved the accuracy of our models. By applying various transformations, such as rotation, scaling, and flipping, we have effectively increased the diversity and size of our dataset. This has allowed our models to better generalize to unseen data and become more robust to variations in the input images.
- The increased accuracy can be attributed to the model's ability to learn more nuanced features and patterns from the augmented dataset, which in turn leads to better performance on the validation set.
- The data augmentation techniques have also helped mitigate overfitting, enabling the model to maintain a more stable performance across different epochs and varying conditions.

4. Conclusion

In conclusion, we implemented three different deep learning architectures, U-Net, LinkNet, and SegNet, for the semantic segmentation task to identify plant rows in given images. By adding data augmentation techniques such as rotation, flipping, and brightness adjustments, we were able to enhance the performance of the models by providing more diverse training data. This led to improved accuracy and generalization capabilities for the models, enabling them to handle unseen data more effectively.

All three models consisted of an encoder-decoder structure with varying numbers and types of layers, and they were trained using a batch size of 16, 25 epochs, and the Adam optimizer. Upon evaluation, it was observed that the incorporation of data augmentation indeed contributed to better accuracy across all the models.

As a result, the study demonstrates the importance of selecting an appropriate deep learning architecture for the task at hand and highlights the benefits of using data augmentation techniques to improve model performance. Moving forward, further optimization and experimentation with different architectures or hyperparameters can lead to even better results in semantic segmentation tasks.