IIT Bombay

Department of Electrical Engineering

Dual Degree Project Report

Image Segmentation in Aerial/Satellite Images

Submitted by:

Mrudul Jambhulkar

Roll No: 21D070044

Dual Degree (B.Tech + M.Tech) in Electrical Engineering

Guide:

Prof. Rajbabu

Department of Electrical Engineering

Abstract

Image segmentation plays a vital role in agricultural applications by enabling precise identification of regions such as crops, weeds, soil, and water bodies. Accurate segmentation supports key tasks including crop monitoring, yield estimation, and resource management. However, agricultural imagery varies significantly across different sensing modalities—RGB, multispectral, and hyperspectral—each providing distinct spectral and spatial information. Understanding segmentation performance across these domains is therefore essential for developing robust and generalizable models.

In this study, experiments were conducted on five diverse datasets—PhenoBench, HiRISE, Agriculture-Vision, UAV-HSI-Crop, and the Malegaon Agricultural Dataset—using multiple segmentation models to explore domain-specific challenges. The study also deals with how large pretrained models, can be effectively leveraged for domain adaptation and achieved reasonable segmentation performance. Additionally, techniques for handling class imbalance are discussed, addressing one of the key limitations in agricultural data. The results further demonstrate effecive segmentation performance across several crop-related classes in hyperspectral imagery with limited data and heavy class imbalance.

Contents

Abstract **2**

1 Introduction **4**
1.1 Introduction 4

2 Literature Review **5**
2.1 Zero-shot Image Segmentation 5
2.1.1 Vision Transformer 5
2.1.2 Segment Anything Model (SAM) 6
2.2 Knowledge Distillation 8
2.2.1 Distillation Loss 8
2.2.2 Types of Knowledge Distillation 8
2.3 DeepLab Models for Semantic Segmentation 10
2.3.1 DeepLabV3 10
2.3.2 DeepLabV3+ 14
2.4 Hyperspectral Image Segmentation 18
2.4.1 3D U-Net Architecture 18
2.4.2 3D U-Net in Hyperspectral Imaging 19
2.5 Multi-View Self-Constructing Graph Convolutional Network(MSCG-Net) 20
2.5.1 MSCG-Net Architecture 20
2.5.2 Adaptive Class Weighting Loss (ACW) 21

3 Datasets **23**
3.1 PhenoBench Dataset 23
3.2 HiRISE Image Dataset 24
3.3 Agriculture-Vision Dataset 24
3.4 UAV-HSI-Crop-Dataset 26
3.5 Malegaon Agricultural Dataset 28

4 Study Methodology **30**
4.1 Agriculture Vision Dataset 30
4.1.1 Experiment with DeepLabV3+ 30

4.2 HiRISE Image Dataset 34
4.2.1 Knowledge Transfer from SAM 34
4.3 UAV-HSI-Crop-Dataset 38
4.3.1 Experiment with 3D U-Net 38
4.4 PhenoBench Dataset 42
4.4.1 Experiment with MSCG-Net 42
4.4.2 Experiment with DeepLabV3 43
4.5 Malegaon Agricultural Dataset 45
4.5.1 Experiment with SAM 45

5 Conclusion and Future Work 47

Chapter 1

Introduction

1.1 Introduction

Image segmentation in aerial and satellite imagery is a crucial component of precision agriculture, enabling accurate identification of crops, weeds, and nutrient-deficient regions. Effective segmentation directly supports decision-making in farm management, optimizing irrigation, fertilization, and targeted interventions. Agricultural imagery, however, presents challenges such as variations in spatial resolution, lighting conditions, seasonal effects, and complex field textures. Recent advancements in computer vision, particularly transformer-based architectures and foundation models like the Segment Anything Model (SAM) and Vision Transformers (ViT), offer zero-shot segmentation capabilities but are computationally demanding and not specifically optimized for agricultural imagery with fine-grained structures and high spectral dimensionality.

This study systematically explores the application of large pretrained models and knowledge distillation techniques for segmentation across multiple imaging domains—RGB, multispectral, and hyperspectral. It also integrates DeepLab (V3 and V3+), MSCG-Net, and 3D U-Net architectures to evaluate performance across diverse sensing modalities.

Experiments were conducted on five datasets, highlighting key challenges such as class imbalance, reduced feature resolution, and multi-scale object segmentation. The primary objective is to develop a unified approach capable of handling segmentation across diverse domains by leveraging domain differences. The current work deals with systematically exploring models and datasets, and evaluating performance to identify strengths and limitations of existing and proposed methods.

Chapter 2

Literature Review

This chapter discusses about all the models and techniques explored for image segmentation and their advantages and limitations. It goes over several SOTA architectures and discusses how they can be be used to tackle challenging segmentation tasks.

2.1 Zero-shot Image Segmentation

2.1.1 Vision Transformer

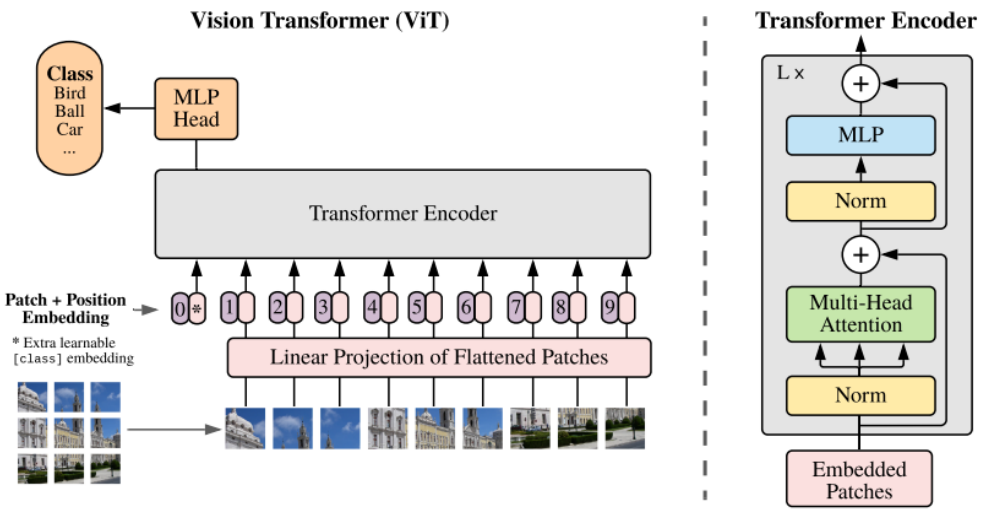


Figure 2.1: Vision Transformer [3]

Vision transformer(ViT) follows similar architecture as that of a transformer used for NLP tasks and employs attention mechanisms to capture global dependancies which regular CNNs struggle with due to limited receptive field. If we directly apply self attention to images it would require each pixel to attend to every other pixel resulting

in very high computational complexity. Several approximations have been explored to make self-attention mechanisms more efficient and enhance the efficiency of self-attention mechanisms.

To handle 2D images, the image $x \in \mathbb{R}^{H \times W \times C}$ is reshaped and divided into a sequence of flattened non-overlapping 2D patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) represents the original image resolution, C is the number of channels, and (P, P) is the size of each image patch. The number of patches is given by $N = \frac{HW}{P^2}$, which also becomes the input sequence length for the Transformer. The transformer operates with dimension D throughout its layers, so each flattened patch is projected to D dimensions using a trainable linear layer to generate the patch embeddings. Transformers do not capture the spatial arrangement of these patches automatically as self-attention treats all the tokens equally. Thus, to incorporate spatial information of the image positional encodings are added to the patch embeddings [3]. A spatial learnable embedding called x_{class} token is prepended to the sequence of patch embeddings, resulting in $z_0^0 = x_{\text{class}}$. The final state of this token after passing through the Transformer encoder, denoted z_L^0, is used as the image representation y. During both pre-training and fine-tuning, a classification head is attached to z_L^0. This head is a multi-layer perceptron (MLP) with one hidden layer during pre-training, and a single linear layer during fine-tuning. The MLP contains two layers with a GELU non-linearity [3].

ViT is pretrained on large datasets like ImageNet-21k, JFT-300M, etc. and then fine-tuned on smaller datasets (e.g., ImageNet-1k).

2.1.2 Segment Anything Model (SAM)

SAM is a zero-shot model developed by Meta AI and is used for image segmentation problems using prompt engineering. It is primarily designed to work on 1024x1024 RGB images although the model has flexibility to take any input resolution to predict the segmentation masks. The success of this model relies on task , model and data [1] .

- **Task:** A prompt can be a set of foreground / background points, a rough box or mask, free-form text, or any information indicating what to segment in an image. The task is to develop a valid segmentation mask given the prompt. If the given prompt is ambiguous, i.e, it refers to multiple objects, it should output a reasonable mask for at least one of those objects.

- **Pre-training:** A pre-training algorithm is needed to simulate a sequence of prompts for each training sample and compare the model's mask predictions against the ground truth. A valid mask has to be predicted even when the prompt is ambiguous.

- **Zero-shot transfer:** The pre trained model should have the ability to respond to

any promot at the time of inference. The model should be to correctly perform a task without having seen any explicit examples of that task during training [1].

Model Architecture

The SAM model has 3 components [1] -

1. **Image encoder :** MAE pre-trained Vision Transformer (ViT) acts as encoder to extract features from the high resolution inputs. It runs once per image and can be applied before providing prompts to the model.

2. **Prompt encoder :** 2 sets of prompts are considered - sparse (points, boxes and text) and dense (masks). Prompt encoder takes these prompts to guide the segmentation process. SAM also allows automatic mask generation without prompts.

3. **Mask decoder :** The mask decoder takes image embeddings and prompt embeddings from image encoder and prompt encoder, respectively and an output token to predict the segmentation mask. A decoder block uses self-attention on prompts and cross-attention in prompt-to-image embedding and also in prompt-to-image embedding to update all embeddings. After two of these decoder blocks the image embeddings are upsampled followed by an MLP to map the output token to a linear classifier. The classifier then computes the mask foreground probability at each image location.

If an ambiguous prompt is given , then the model will average multiple valid masks. 3 mask outputs are generally sufficient to address this. The masks are ranked based on their IoU score . [1] .

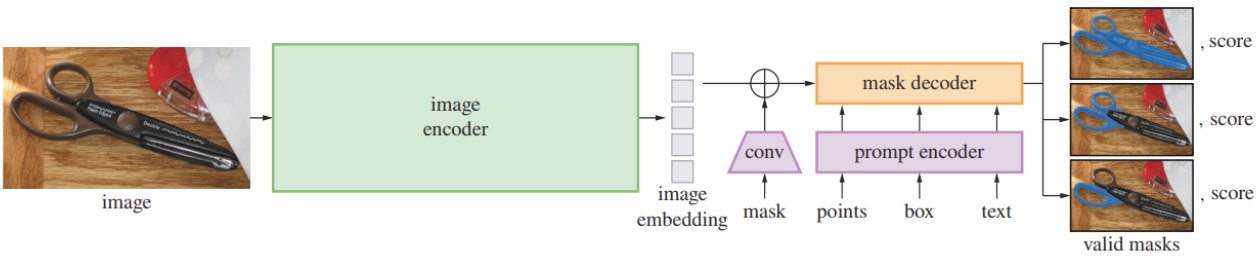


Figure 2.2: SAM overview [1]

Limitations

- SAM is designed for instance segmentation. However it can be adapted for semantic segmentation via clustering algorithms like K-means clustering or DBSCAN .

- SAM may not generalize well when there is domain difference. It is not pretrained largely on agricultural datasets and may not be very efficient in segmentation.
- SAM works for RGB images and can't be effectively used on multispectral/hyperspectral image data which is commonly used in precision agriculture.

2.2 Knowledge Distillation

Knowledge distillation is a machine learning technique that transfers knowledge from a large pretrained model (teacher) to a simple smaller model (student). It is widely used for model compression [11].

2.2.1 Distillation Loss

To make the student network follow the teacher network, two loss functions are used.

1. **Hard Loss :** It is the loss computed between the ground truth and student model's final output logits.
2. **Distillation Loss :** It is also called soft loss. It is the loss computed between the soft targets of student and teacher model [11].

2.2.2 Types of Knowledge Distillation

Response Based Knowledge Distillation

Response based knowledge refers to the response of the final layer of the teacher model. In this method - the student model tries to mimic the final predictions of the teacher model [10].

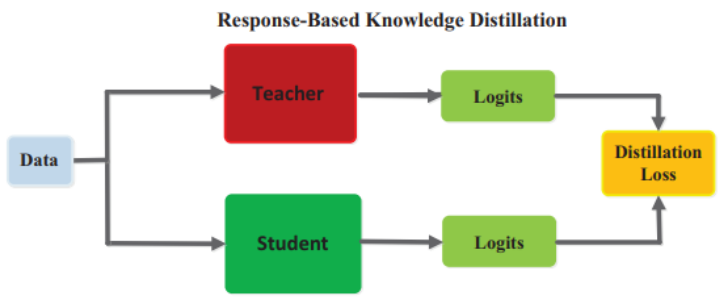


Figure 2.3: Response-based knowledge distillation [10]

The distillation loss is given by :

$$L_{\text{ResD}}\left(p(z_t, T),\, p(z_s, T)\right) = \mathcal{L}_R\left(p(z_t, T),\, p(z_s, T)\right) \tag{2.1}$$

Here, z refers to the logits of the last fully connected layer of the network. $p(z, T)$ refers to the softmax value with temperature factor T and tells the probability of belonging to a class and acts as soft targets.

$\mathcal{L}_R\left(p(z_t, T),\, p(z_s, T)\right)$ is the KL Divergence loss between the soft logits of student and teacher model [10].

Feature-Based Knowledge Distillation

This method aims to match the intermediate feature activations between the student and the teacher model.

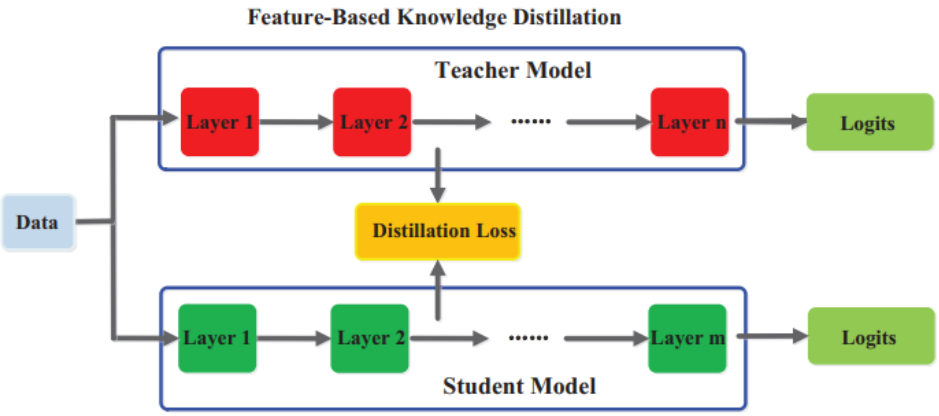


Figure 2.4: Feature-based knowledge distillation [10]

Here, distillation loss is given by :

$$L_{\text{FeaD}}\big(f_t(x), f_s(x)\big) = \mathcal{L}_F\left(\Phi_t(f_t(x)), \Phi_s(f_s(x))\right) \tag{2.2}$$

where $f_t(x)$ and $f_s(x)$ are the feature maps of the intermediate layers of teacher and student models, respectively. $\Phi_t(f_t(x))$ and $\Phi_s(f_s(x))$ are transformation functions, used to project the feature maps of teacher and student models to the same shape. $\mathcal{L}_F(\cdot)$ indicates the similarity function used to match the feature maps of teacher and student models [10].

Relation-Based Knowledge Distillation

In this method the student model aims to mimic the relationships between the feature maps as observed in the teacher model.

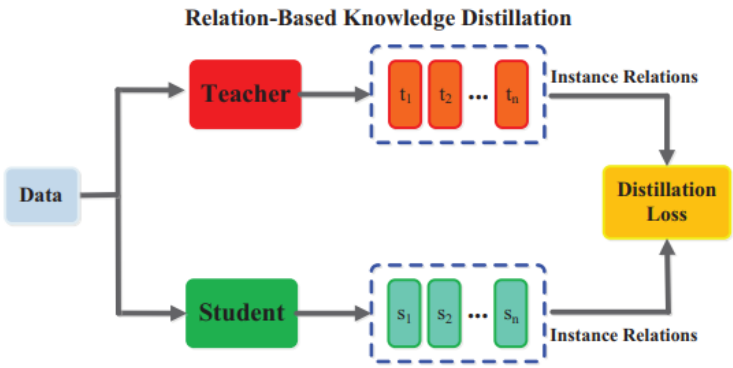


Figure 2.5: Relation-Based Knowledge Distillation [10]

The distillation loss is given by :

$$L_{\text{RelD}}(f_t, f_s) = \mathcal{L}_{\text{R1}}\big(\Psi_t(\hat{f}_t, \check{f}_t), \Psi_s(\hat{f}_s, \check{f}_s)\big) \tag{2.3}$$

Here, f_t and f_s refer to the feature maps extracted from the teacher and student networks, respectively. From each model, pairs of feature maps—$\hat{f}_t$, $\check{f}_t$ for the teacher and $\hat{f}_s$, $\check{f}_s$ for the student—are selected for comparison. The functions $\Psi_t(\cdot)$ and $\Psi_s(\cdot)$ measure the similarity between paired feature maps from the teacher and student models, while $\mathcal{L}_{\text{R1}}(\cdot)$ quantifies correlations between these feature pairs [10].

2.3 DeepLab Models for Semantic Segmentation

2.3.1 DeepLabV3

DeepLabV3 [4] is a deep neural network architecture proposed by Google for semantic segmentation tasks. It utilizes atrous(dilated) convolutions to control the receptive field without introducing additional parameters and overcomes several challenges faced in image segmentation .

Problems faced in Image Segmentation

- **Reduced feature resolution :** In Deep Convolutional Neural Networks (DCNNs) due to consecutive pooling or strided convolutions for downsampling to extract higher level features, the feature resolution decreases as we go deeper into the network. This maybe useful for image classification but it causes a disadvantage in image segmentation tasks where spatial information is important to generate accurate segmentation masks.

  To address this issue, atrous or dilated convolutions are proposed to increase the receptive field without decreasing the feature resolution.

- **Existence of objects at multiple scales :** The same class object can be present at different scales in the image data. Deeper layers capture high-level information as they have larger receptive fields but small objects suffer. If the receptive field is too large, the network may struggle to segment small objects effectively, and if the receptive field is too small, it may struggle to segment large objects. Thus, the network should learn to recogniseize the object at different sizes for efficient segmentation.

  To overcome this challenge, Atrous Spatial Pyramid Pooling module is used for multi-scale feature extraction [4] [6] .

Atrous Convolution

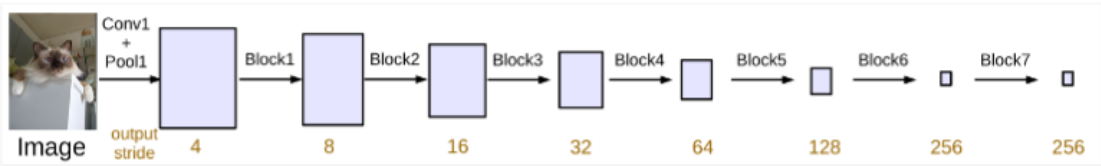


Figure 2.6: Going deeper without atrous convolution [4]

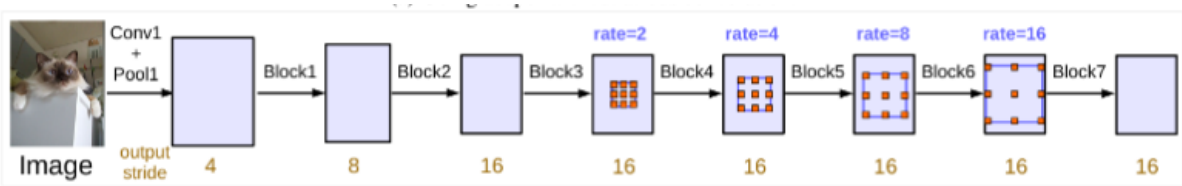


Figure 2.7: Going deeper with atrous convolution. Atrous convolution with rate greater than 1 is applied after block3 when output stride = 16 [4]

Deeper layers have higher receptive field to capture abstract feature representations . Receptive field can be be increased by increasing the stride in a CNN layer but that would decrease the resolution of the output feature map. Fig 2.6 shows the output stride (ratio of input image size dimension to the feature map) increases to as high as 256 at block 7, implying very low feature map resolution.

For each location i on the output y and a filter w, atrous or dilated convolution is defined as :

$$y[i] = \sum_{k} x[i + r \cdot k]w[k] \tag{2.4}$$

Here , atrous rate r corresponds to the stride with which the input map is sampled. The atrous convolution operation is equivalent to convolving x with an upsampled filter w obtained by inserting $r - 1$ zeros between the filter values along each spatial dimension [4]. The parameter r controls the receptive field without introducing any additional

parameters and keeping the stride equal to 1 without downsampling the feature map. Fig 2.7 shows that the feature resolution remains constant after block 3 (at output stride = 16) by introducing dilated convolutions with different r in subsequent layers.

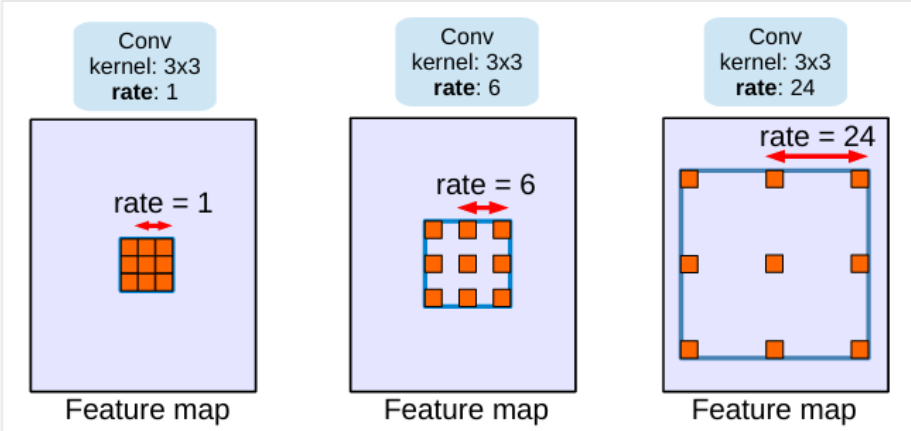


Figure 2.8: Atrous convolution with kernel size 3x3 and different rates. Standard convolution corresponds to atrous convolution with rate = 1. Employing large value of atrous rate enlarges the model's field-of-view, enabling object encoding at multiple scales [6]

Atrous Spatial Pyramid Pooling (ASPP)

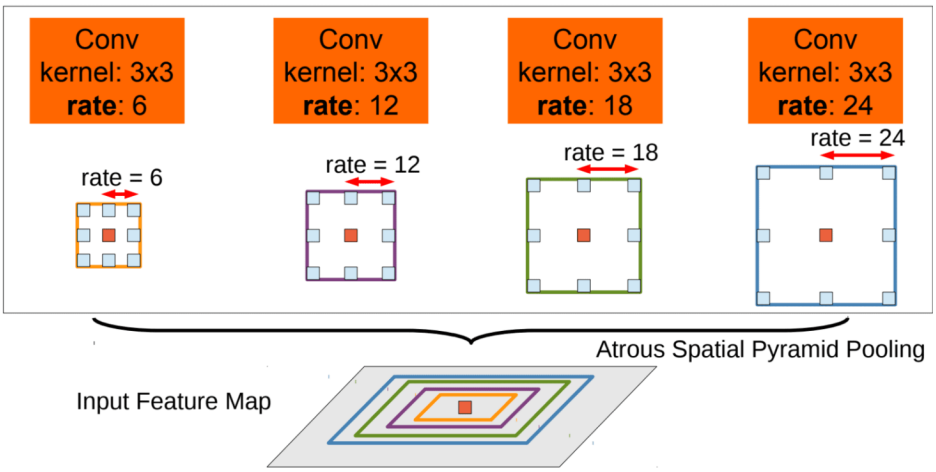


Figure 2.9: To classify the center pixel (orange), ASPP exploits multi-scale features by employing multiple parallel filters with different rates. The effective Field-Of-Views (FOV) are shown in different colors [6]

Various architectures like - image pyramid , encoder-decoder and spatial pyramid pooling have been employed to capture multi-scale context. Spatial pyramid pooling (SPP) [7] divides the feature map into spatial bins at different pyramid levels and pooling is applied to each of these bins and finally the outputs from all the levels are concatenated to produce a fixed length representation. This pooling at different levels helps the model to capture multi-scale context.

In DeepLabV3 , atrous convolution is used as the tool to capture multi-scale context in SPP. Multiple atrous convolutional layers at different atrous rates are applied in parallel to capture information at different scales and the extracted features are fused together to generate the final output [4] [6].

But, as the the atrous rate increases , the number of filter weights which act on valid regions of feature map (i.e not the padded zero region) decreases and in extreme case , the filter effectively becomes a 1x1 filter as only the center pixel is effective.

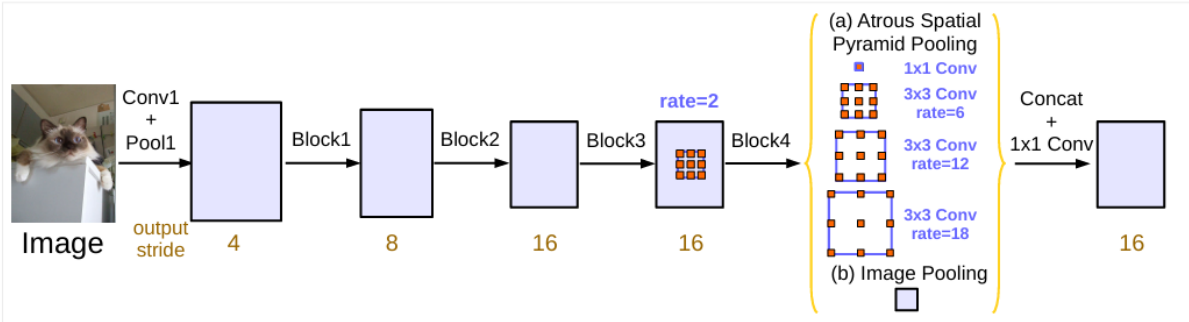


Figure 2.10: Parallel modules with atrous convolution (ASPP), augmented with image-level features [4]

DeepLabV3 modifications to ASPP :

- To overcome the problem associated with high atrous rates , a 1x1 convolution operation (i.e global average pooling) is first applied to capture global context .
- Batch normalization is used within ASPP
- One 1x1 and three 3x3 convolutions are applied. For 3x3 atrous convolution - the atrous rates are set to (6,12,18) for output stride = 16.
- Each convolutional layer has 256 output channels.
- The resulting feature maps are then concatenated and passed through (1x1,256) convolutional layer [4] [6] .

DeepLabV3 Architecture

Fig. 2.11 shows the architecture of DeepLabV3. It uses ResNet-101 as its backbone. The output feature map from block4 is passed through the ASPP module. The output features from the different submodules of the ASPP are concatenated and passed through 1x1 convolution for mixing these features. It is followed by another 1x1 convolution block to generate the final prediction [4] [6].

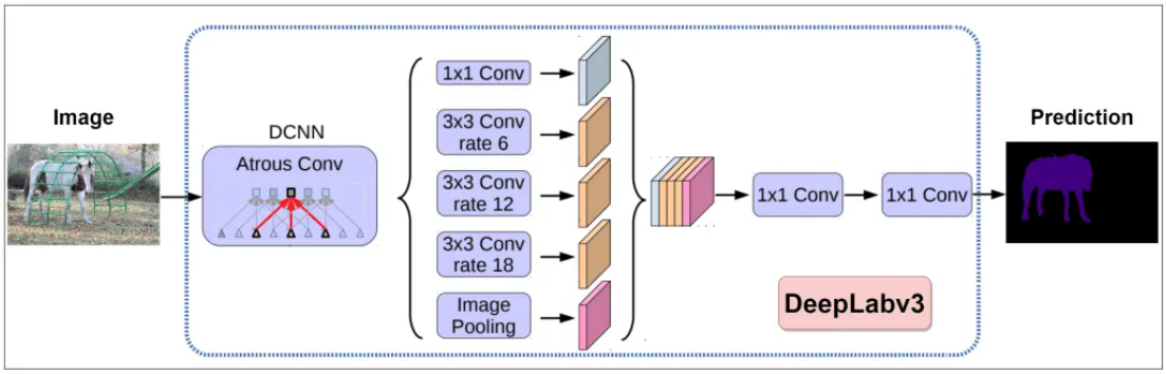


Figure 2.11: DeepLabV3 Architecture [6]

2.3.2 DeepLabV3+

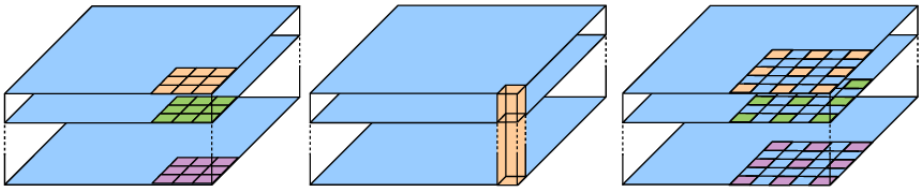
Limitations of DeepLabV3

- ASPP modules can encode multi-scale contextual information but the fine boundary details of the objects is still missing .
- Atrous convolution can capture boundary details by extracting denser feature maps but maintaining higher dimensional feature maps for so many layers in a DCNN causes a lot of overhead on the GPU memory and is computationally expensive .
- Encoder-decoder architectures like in U-Net work well in these cases. The encoder successively decreases the feature map resolution to capture long-range information and then the decoder is employed to upsample the feature map and restore spatial information such as object boundaries which get lost in the encoder path.

Atrous Depth-Wise Separable Convolution

Depthwise separable convolution factorizes a standard convolution into two parts:

- Depthwise convolution – Performs a spatial convolution independently for each input channel.
- Pointwise convolution – Depthwise is followed by a pointwise (i.e., (1×1) convolution). Pointwise convolution is employed to combine the output from depthwise convolution across channels.



(a) Depthwise conv. (b) Pointwise conv. (c) Atrous depthwise conv.

Figure 2.12: 3x3 Depthwise separable convolution decomposes a standard convolution into (a) a depthwise convolution (b) a pointwise convolution. In this work, (c) atrous separable convolution is employed. [5]

Depthwise separable convolution are much more computationally efficient than standard convolutions .

Xception Architecture

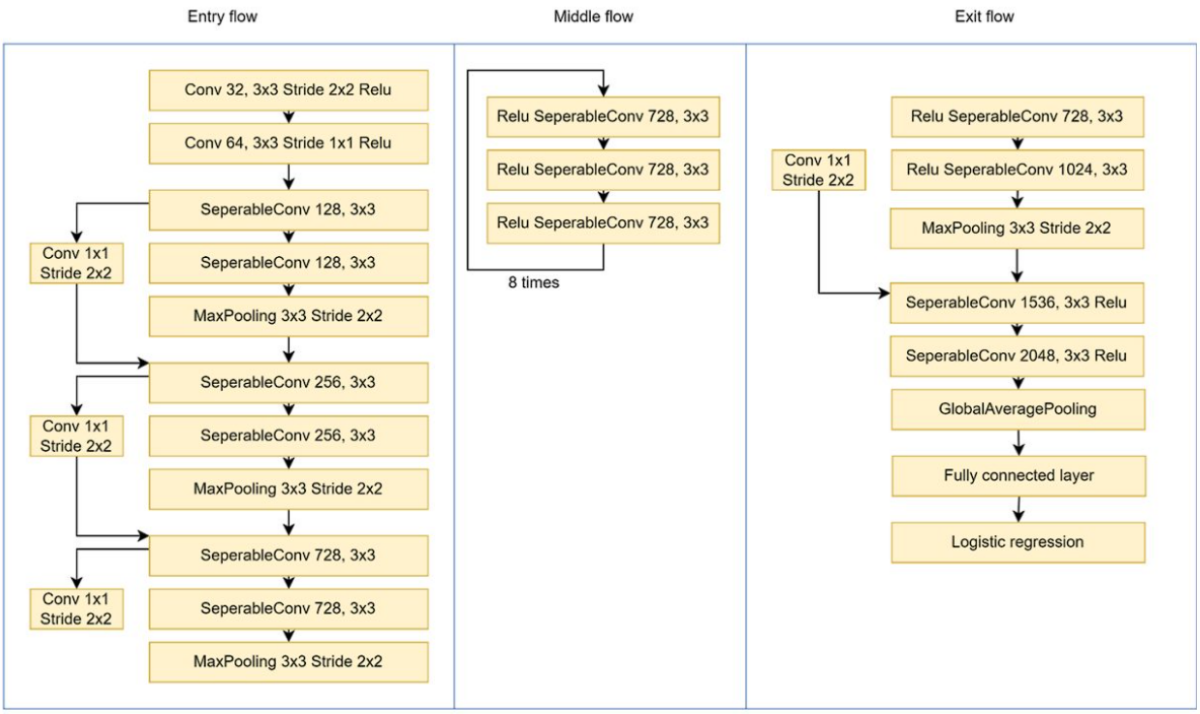


Figure 2.13: Xception architecture - consists of entry flow to capture low level features followed by middle flow to capture higher level features and then exit flow for learning complex image features and producing final segmentation results. [9]

The Xception model completely decouples the task of learning spatial and cross-channel relations by using modified depthwise separable convolution. Here , the order of 2 operations in depthwise separable convolution is reversed , i.e. first pointwise convolution is performed then followed by depthwise convolution [8].

The architecture consists of 3 parts as shown in fig. 2.13:

1. **Entry flow :** Extracts low-level features using 3×3 convolutions (32 and 64 filters) followed by modified depthwise separable convolutions and max pooling for dimensionality reduction.

2. **Middle Flow :** Repeated eight times with 3×3 depthwise separable convolutions (728 filters) and ReLU activations to learn high-level representations

3. **Exit Flow :** Applies deeper separable convolutions (728–2048 filters), followed by Global Average Pooling and a fully connected layer for final image classification [8]

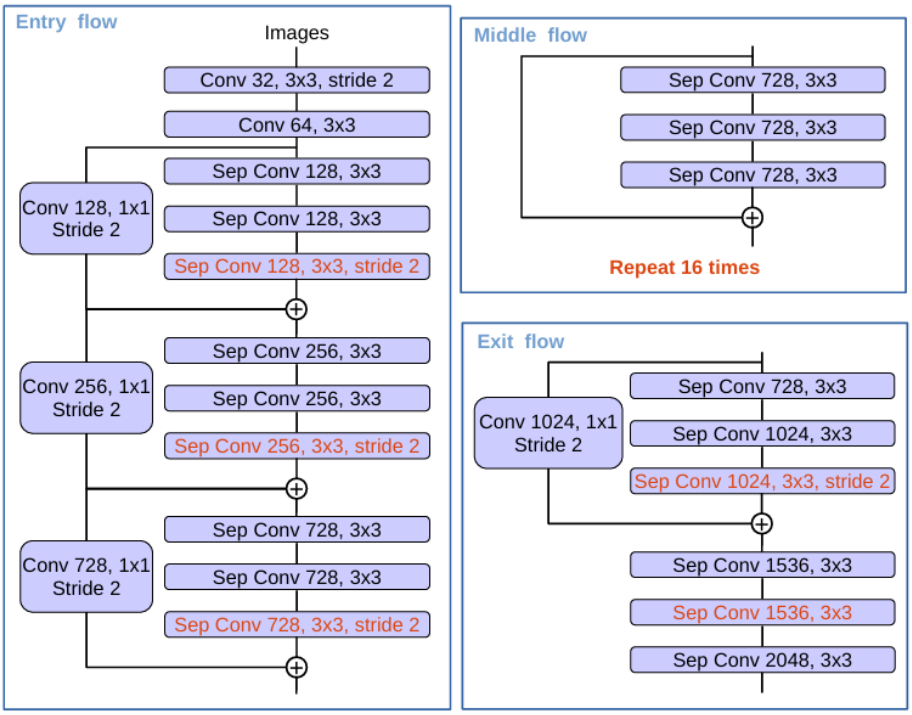


Figure 2.14: Aligned Xception Architecture [5]

DeepLabV3+ utlizes a modified version of the Xception architecture as shown in fig. 2.14.

- It introduces more layers in the architecture.
- All max pooling operation are replaced by atrous depthwise separable convolutions with striding to maintain feature resolutions.
- It adds extra batch normalization and ReLU after the 3x3 depthwise separable convolution blocks for training stability [5].

DeepLabV3+ Architecture

DeepLabV3+ extends the DeepLabV3 model to an encoder-decoder architecture as shown in fig. 2.15.

Encoder : It comprises of the standard DeepLabV3 model as the encoder. The atrous convolutions employed in the encoder extract the features at different scales. Aligned Xception model is used as the backbone and the primary feature extractor in this study.

Decoder :A simple decoder module is used to refine the segmentation results and effectively capture the object boundaries. High-level encoder features (output stride=16) are upsampled by 4× and concatenated with low-level backbone features. A 1×1 convolution block reduces channels of low-level features to balance importance with encoder features. A few 3×3 convolution blocks and final 4× bilinear upsampling gives sharper object boundaries and detailed predicted segmentation results [5] [6].

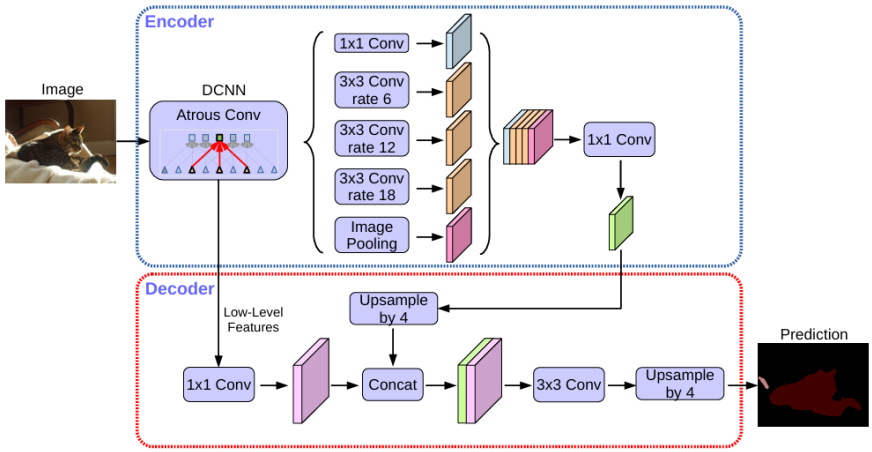


Figure 2.15: Architecture of DeepLabv3+ [5]

Limitations

- DeepLabV3+ is primarily designed to handle RGB images and has pretrained weights available only for 3 channels. Thus, it may struggle with multispectral or hyperspectral images commonly found in agriculture.

- Although it can capture object details at multiple scales, agricultural data from drones is captured in diverse weather conditions and at different camera angles but the architecture doesn't have any specific mechanism to capture this.

- Even after ASPP modules and encoder-decoder structure agricultural data consists of objects like weeds which are present at very small scale and the model may struggle with segmenting small objects due to downsampling in encoder path.

2.4 Hyperspectral Image Segmentation

2.4.1 3D U-Net Architecture

3D U-Net is an extension of the U-Net architecture commonly used in medical image segmentation to work on sparsely annotated volumetric images [12].

Volumetric image is a 3-dimensional image obtained by stacking multiple slices of 2D images commonly obtained from MRI/CT scan. Volumetric images comprise of voxels(3D equivalent of pixels) and each voxel carries feature information for that location in that volume.

In this architecture 3D convolutions are employed , i.e the kernel slides across all the 3 dimensions instead of only 2 dimensions across a 2D image in standard convolution. The 3D U-Net comprises of :

- **Analysis (encoder) path :** To progressively downsample the input and extract features.
- **Synthesis (decoder) path :** To progressively upsample feature map and restore spatial resolution for accurate image segmentation.

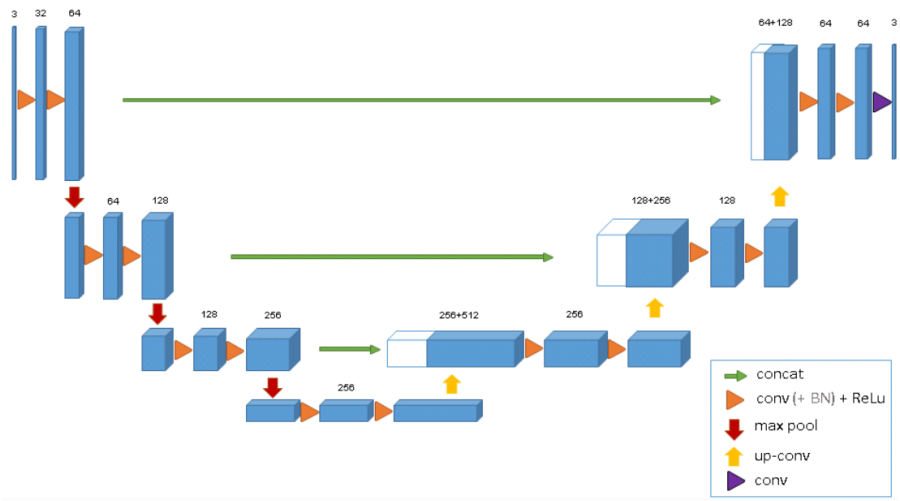


Figure 2.16: The 3D U-net architecture. Blue boxes represent feature maps. The number of channels is denoted above each feature map [12]

.

Encoder Path

Each encoder block consists of :

- Two 3×3×3 convolutions, each followed by a ReLU activation layer. Batch Normalization is applied before each ReLU.

- It is then followed by a 2×2×2 max pooling (stride = 2) to reduce spatial dimension.
- Number of channels is doubled before each pooling layer to prevent bottlenecks (as too much information may get compressed in small resolution feature maps with less channels if number of channels are not increased) [12].

Decoder Path

Each decoder block consists of :

- An up-convolution (transposed convolution) of 2×2×2 (stride = 2) to upsample the feature map.
- Two 3×3×3 convolutions with ReLU activations.
- Shortcut connections from the encoder layers are present to provide high-resolution features which are then concatenated in decoder path [12].

Output

The final 1×1×1 convolution maps the features to the number of semantic labels (3 in this case) [12].

2.4.2 3D U-Net in Hyperspectral Imaging

3D U-Net can be employed hyperspectral images obtained from satellite images for segmenting agricultural data and identify types of crops, weeds, soil etc.

Hyperspectral images consist of over 200 bands and most of the bands are often redundant. Dimensionality reduction techniques like PCA can reduce it to smaller number of bands (around 30 in this study) while preserving most of the variance and these 30 image slices can then be considered a volumetric image for the 3D U-Net model. By processing the data in 3 dimensions the model can effectively capture the relations and learns the information offered by multiple bands of the hyperspectral image.

Limitations

- 3D U-Net can effectively capture information from multiple bands but can be computationally expensive when working with large number of bands.
- Although dimensionality reduction techniques like PCA can reduce the number of bands but to preserve sufficient variance, generally over 30 bands would still be required and so many parameters can cause the model to overfit.
- Small scale objects like weeds will still be difficult to segment by the model as the analysis path causes loss of spatial resolution and boundary details.

2.5 Multi-View Self-Constructing Graph Convolutional Network(MSCG-Net)

Graph Neural Networks (GNNs) can model long range dependencies in the data quite well and can be useful for image segmentation.

MSCG-Net [18] utilizes a Self-Constructing Graph(SCG) module which constructs an undirected graph without relying on prior knowledge graphs and adjacency matrices [19].

2.5.1 MSCG-Net Architecture

Backbone Feature Extraction

- It uses SEResNeXt-50/101 pretrained backbone
- The first layer was adapted for 4-channel input (RGB + NIR). Weights for NIR channel were obtained by replicating the red channel weights.

SCG Module

- Input to this module has 1024-channel feature maps
- Adaptive pooling was employed to compress to 32×32 node features (1024 total nodes).
- The SCG module aim to learn a mean matrix μ and a standard deviation matrix σ of a Gaussian using two single-layer CNNs.

  By reparameterization trick similar to the Variational Autoencoder(VAE), the latent embedding Z is obtained by:

  $$Z \leftarrow \mu + \sigma \cdot \varepsilon \tag{2.5}$$

  where ε is noise from a standard normal distribution ($\varepsilon \sim \mathcal{N}(0, I)$).
- Adjacency matrix is then computed based on the node similarity. The adjacency matrix is built via node similarity:

  $$A = \mathrm{ReLU}(Z \times Z^{\top}) \quad \text{where} \quad Z \in \mathbb{R}^{N \times d} \tag{2.6}$$

  Each element A_{ij} represents the similarity between node i and node j.

  The adjacency matrix is then regularized to produce a normalized matrix $\hat{A}$
- The SCG module provides the following outputs: node features X', adjacency matrix $\hat{A}$, and a residual prediction $\hat{y}$ [19] [18].

Graph Convolutional Network (GCN)

GCN implements message-passing alogirthm to transmit information to nodes in the graph by a combination of linear transformations over one hop neighbourhoods followed by sigmoid non-linearity [19].

The extracted feature maps from the backbone are augmented by rotating the maps by 0 , 90 and 180 degrees so that the model to exploit rotation invariance. These maps are then passed to the SCG-GCN blocks. The SCG-GCN blocks then produces 3 outputs corresponding to these rotations and these are merged by element-wise addition to produce the final segmentation result:

$$\text{Fused} = (\hat{y} + Z^{(2)}) \oplus (\hat{y}_{90} + Z^{(2)}_{90})_{r90} \oplus (\hat{y}_{180} + Z^{(2)}_{180})_{r180} \tag{2.7}$$

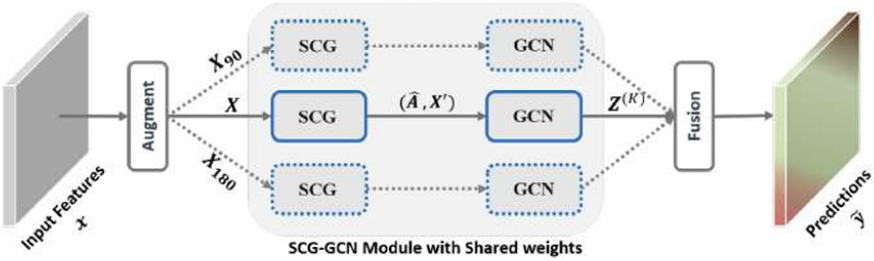


Figure 2.17: MSCG-Net Architecture [18]

2.5.2 Adaptive Class Weighting Loss (ACW)

In agricultural image segmentation, the distribution of classes is highly uneven. An adaptive weighting method from [18] is used that updates class weights during training rather than using pre-computed fixed weights.

Iterative Frequency Calculation

The pixel frequency for each class j is updated at each training step t:

$$f_j^t = \frac{\hat{f}_j^t + (t-1) \cdot f_j^{t-1}}{t} \tag{2.8}$$

where $\hat{f}_j^t$ is the current batch frequency and $f_j^0 = 0$. This maintains a running average of class frequencies.

Adaptive Weight Computation

Class weights are computed using median frequency balancing:

$$w_j^t = \frac{\text{median}(f_j^t | j \in C)}{f_j^t + \epsilon} \tag{2.9}$$

where C is the number of classes and $\epsilon = 10^{-5}$ prevents division by zero. Rare classes get higher weights.

Pixel-wise Weight Normalization

Weights are normalized and broadcast to pixel level:

$$\tilde{w}_{ij} = \frac{w_j^t}{\sum_{j \in C}(w_j^t)} \cdot (1 + y_{ij} + \tilde{y}_{ij}) \tag{2.10}$$

where $y_{ij} \in (0, 1)$ is the ground truth and $\tilde{y}_{ij} \in (0, 1)$ is the prediction for pixel i and class j.

Positive-Negative Balanced Loss

Instead of traditional cross-entropy, we use a balanced function that considers both positive and negative samples:

$$\mathbf{p} = \mathbf{e} - \log\left(\frac{1 - \mathbf{e}}{1 + \mathbf{e}}\right) \tag{2.11}$$

where $\mathbf{e} = (\mathbf{y} - \tilde{\mathbf{y}})^2$ represents the squared error between predictions and ground truth.

Adaptive Multi-Class Weighting Loss

Combining adaptive weighting with the balanced loss function:

$$\mathcal{L}_{acw} = \frac{1}{|Y|} \sum_{i \in Y} \sum_{j \in C} \tilde{w}_{ij} \cdot p_{ij} - \log\left(\text{MEAN}(d_j | j \in C)\right) \tag{2.12}$$

where d_j is the Dice coefficient for class j:

$$d_j = \frac{2 \sum_{i \in Y} y_{ij} \bar{y}_{ij}}{\sum_{i \in Y} y_{ij} + \sum_{i \in Y} \bar{y}_{ij}} \tag{2.13}$$

Chapter 3

Datasets

This chapter provides brief description of all the datasets used for the experiments.

3.1 PhenoBench Dataset

Dataset size: 1,407 training images and 772 validation images.
Semantic annotations: Pixel-wise labels for background, crops, weeds, and partial crops (partial crops/weeds $< 50\%$ visible).

The dataset consists of 1024×1024 pixel RGB images that were taken with a UAV fitted with a high-resolution camera in actual field conditions. The native resolution of the original aerial imagery was 11664×8750 pixels. The UAV flew at a height of about 21 meters, which is equivalent to a ground sampling distance (GSD) of roughly 1 millimeter per pixel.

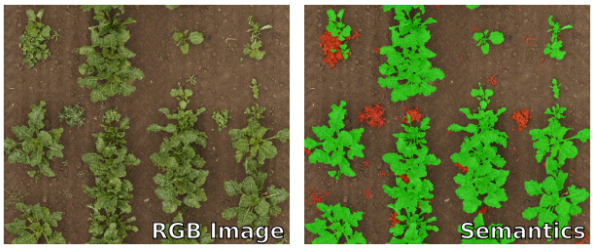


Figure 3.1: Image from PhenoBench dataset

The PhenoBench dataset provides semantic annotations for vegetation segmentation tasks with the following label classes: background (label 0), crop (label 1), weed (label 2), partial crop (label 3), and partial weed (label 4). To reduce ambiguity at vegetation boundaries and simplify the classification task, we merge the partial classes into their respective primary categories. Specifically, label 3 (partial crop) is reassigned to label 1 (crop), and label 4 (partial weed) is reassigned to label 2 (weed). After this remapping, the dataset effectively contains three semantic classes: background, crop, and weed. This

merging strategy ensures more robust model training and cleaner separation between meaningful vegetation classes in the field imagery [13].

3.2 HiRISE Image Dataset

This dataset consists of images obtained from HiRISE instrument and downloaded from from public space archives such as PDS Geosciences Node Orbital Data Explorer [15]. The images are obtained from image sensors operating in the visible (VIS) and Near InfraRed (NIR) spectrums on board probes orbiting Mars [14].

It consists of over 60,000+ unlabeled grayscale images with 227x227 resolution. This dataset is used for detection and segmentation of martian landforms.

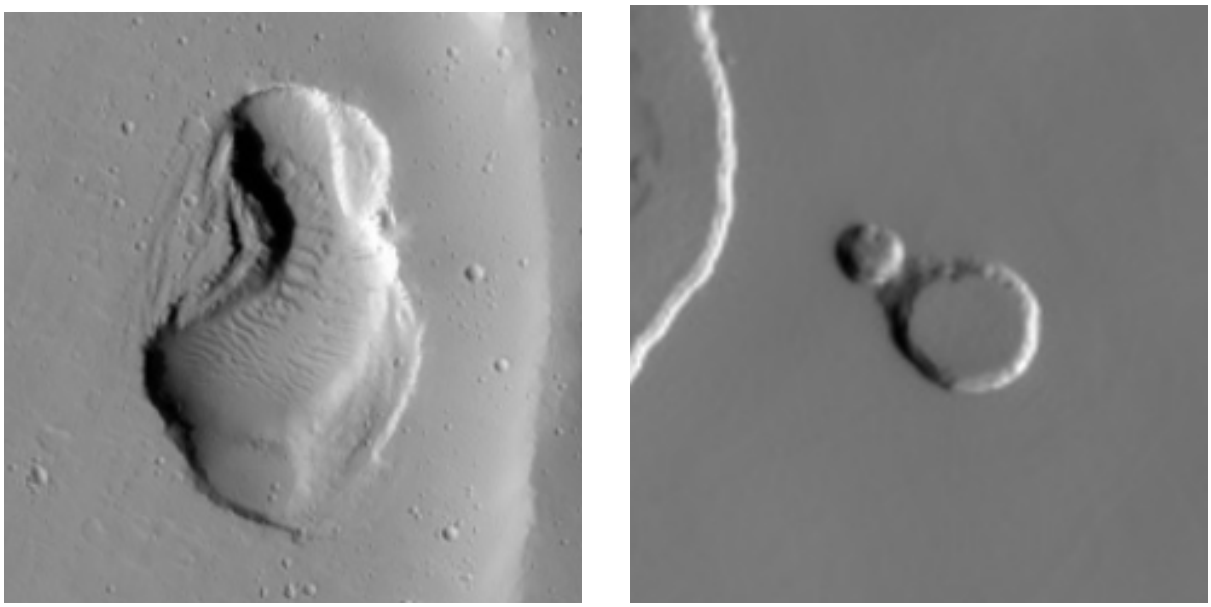


Figure 3.2: Mars terrain images [15]

3.3 Agriculture-Vision Dataset

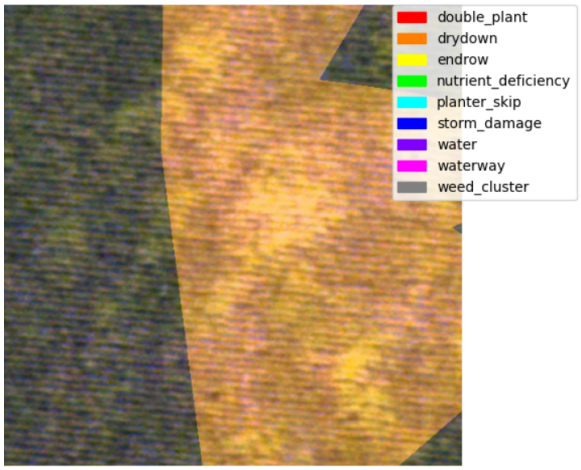
Agrivision challenge dataset - 2021 is used in this study. It is an aerial agricultural multispectral image dataset consisting of 4 bands (NIR and RGB) with a resolution of 512x512. It consists of farmland images captured between 2017-2019 in the United States [16].

It consists of 56,944 train images and 18,334 validation images. The dataset also provides valid masks for valid pixels captured by the camera and boundary masks for indicating which pixels belong to the farmland. Valid farmland pixels were considered by the model for learning and evaluation.

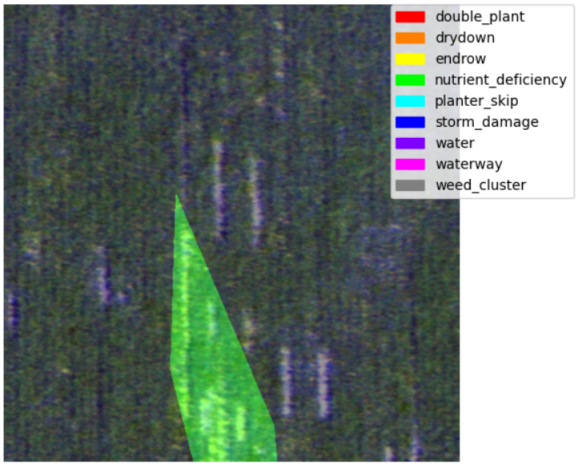
The dataset consists of 9 classes - (1) double plant, (2) drydown, (3) Endrow, (4) Nutrient deficiency, (5) Planter skip, (6) Storm damage, (7) Water, (8) Waterway, (9)

Weed cluster. Out of these 9 classes, storm damage is not considered for evaluation.

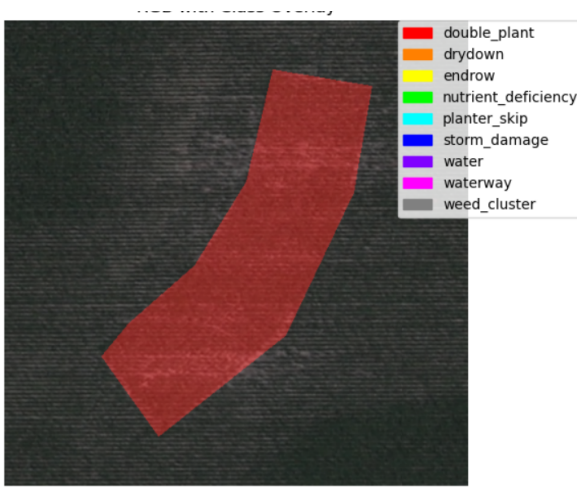
The semantic labels are quite difficult to interpret visually and do not seem very accurate for some classes like drydown and nutrient deficiency as shown in fig. 3.3a and 3.3b. The dataset also exhibits some class imbalance as shown in - fig. 3.4 and 3.5.



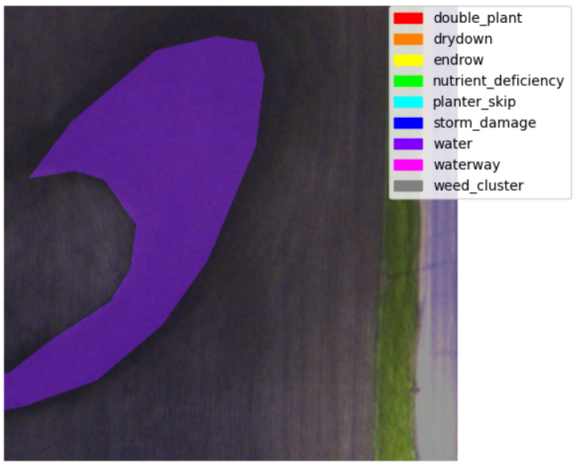
(a) Drydown



(b) Nutrient Deficiency



(c) Endrow



(d) Water

Figure 3.3: Sample images from AgriVision dataset along with their ground truth class overlay : (a) Drydown, (b) Nutrient Deficiency, (c) Endrow, and (d) Water presence.

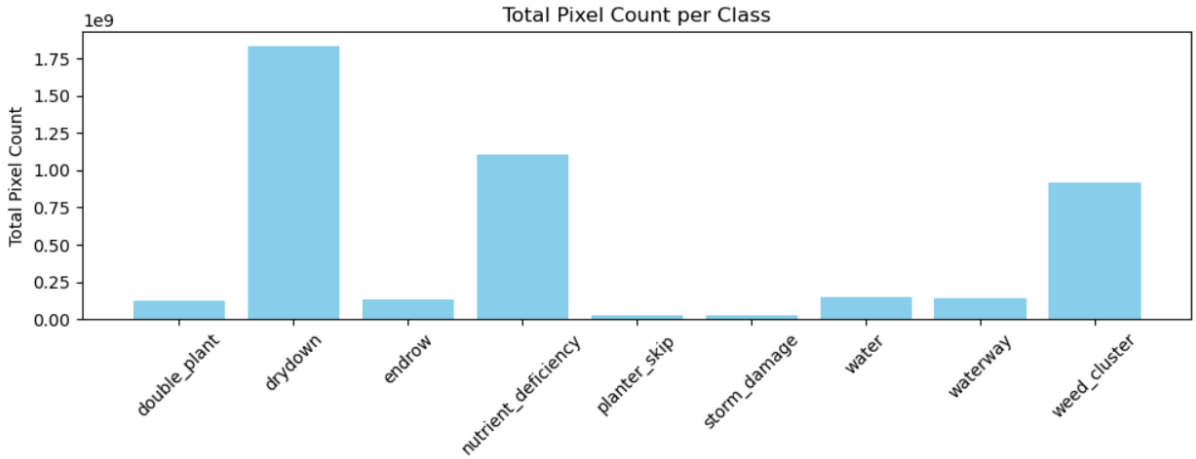


Figure 3.4: Total pixel count per class

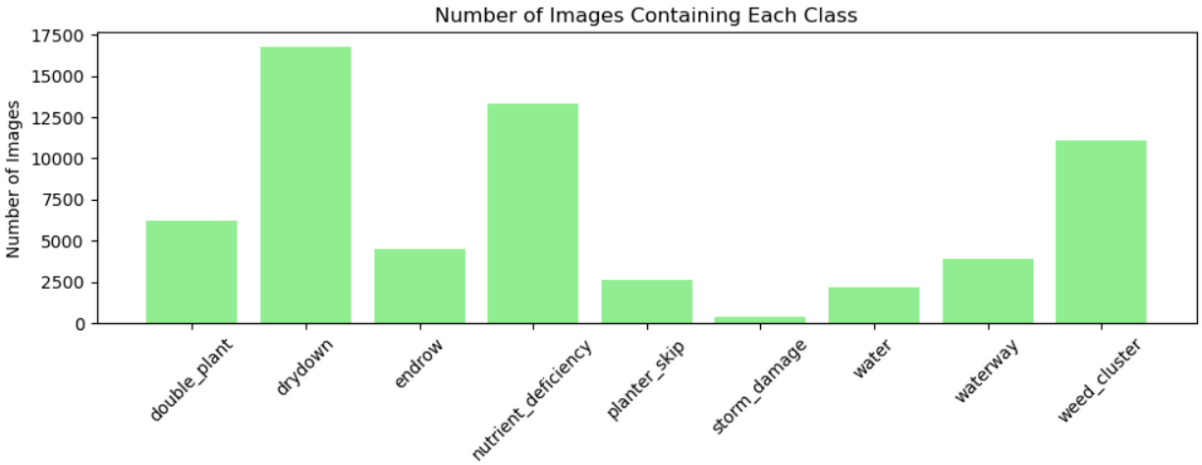
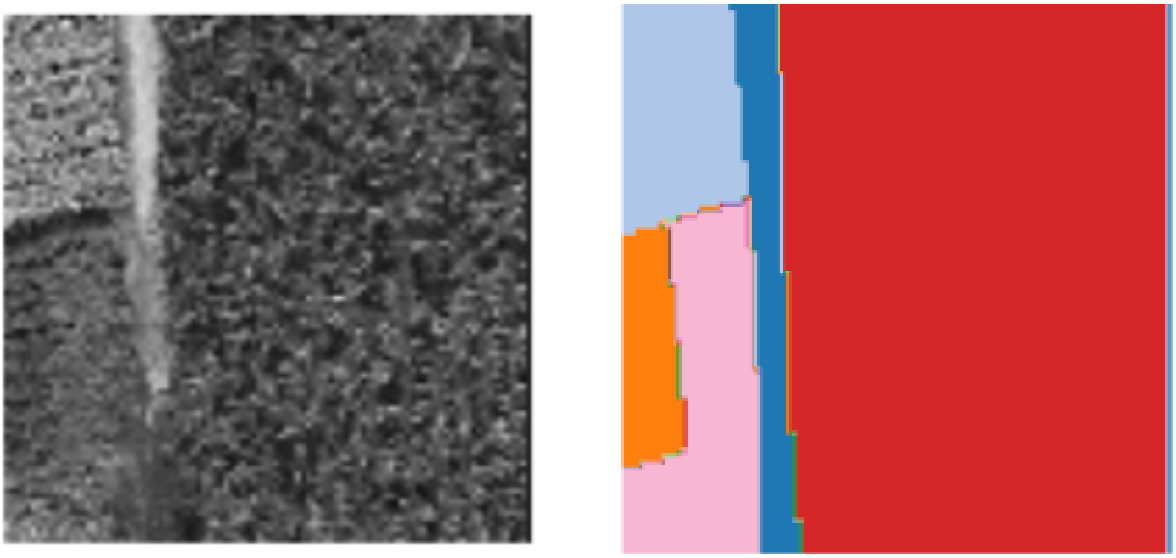


Figure 3.5: Number of images containing each class

3.4 UAV-HSI-Crop-Dataset

The data was collected using an electric hexacopter from a study area located in Shenzhou City, Hebei Province, China. The area consists of two sub-regions: the plots of Majiakou Village (MJK_N, MJK_S) and the plots of Xijingmeng Village (XJM). The flight height of UAV was set to be 100 m, resulting in a spatial resolution of 0.1 m per pixel [17].



(a) Image from UAV-HSI crop dataset (b) Ground truth

Figure 3.6: Sample image from the UAV-HSI crop dataset and its corresponding ground truth.

This dataset consists of hyperspectral images with a resolution of 96 x 96 and a total of 200 bands. It consists of total of 30 classes which includes 27 different types of crops. The number of images in train, validation and test set are 312, 34 and 87 respectively.

The classes are: Chinese cabbage, Millet, Leaf mustard, Green bean, Spinach, Bok Choy, Turnip, Cotton, Corn, Carrot, Sorghum, Pumpkin, Kohlrabi, Scallion, Sweet potato, Peanut, Sesame, Beans, Road, Tobacco, Holly, Cauliflower, Eggplant, Daikon, Sichuan peppercorn, Mulched field, Tree, Okra, Bare soil & Weed, NULL. The dataset has very high class imbalance as shown in the fig. 3.7 .

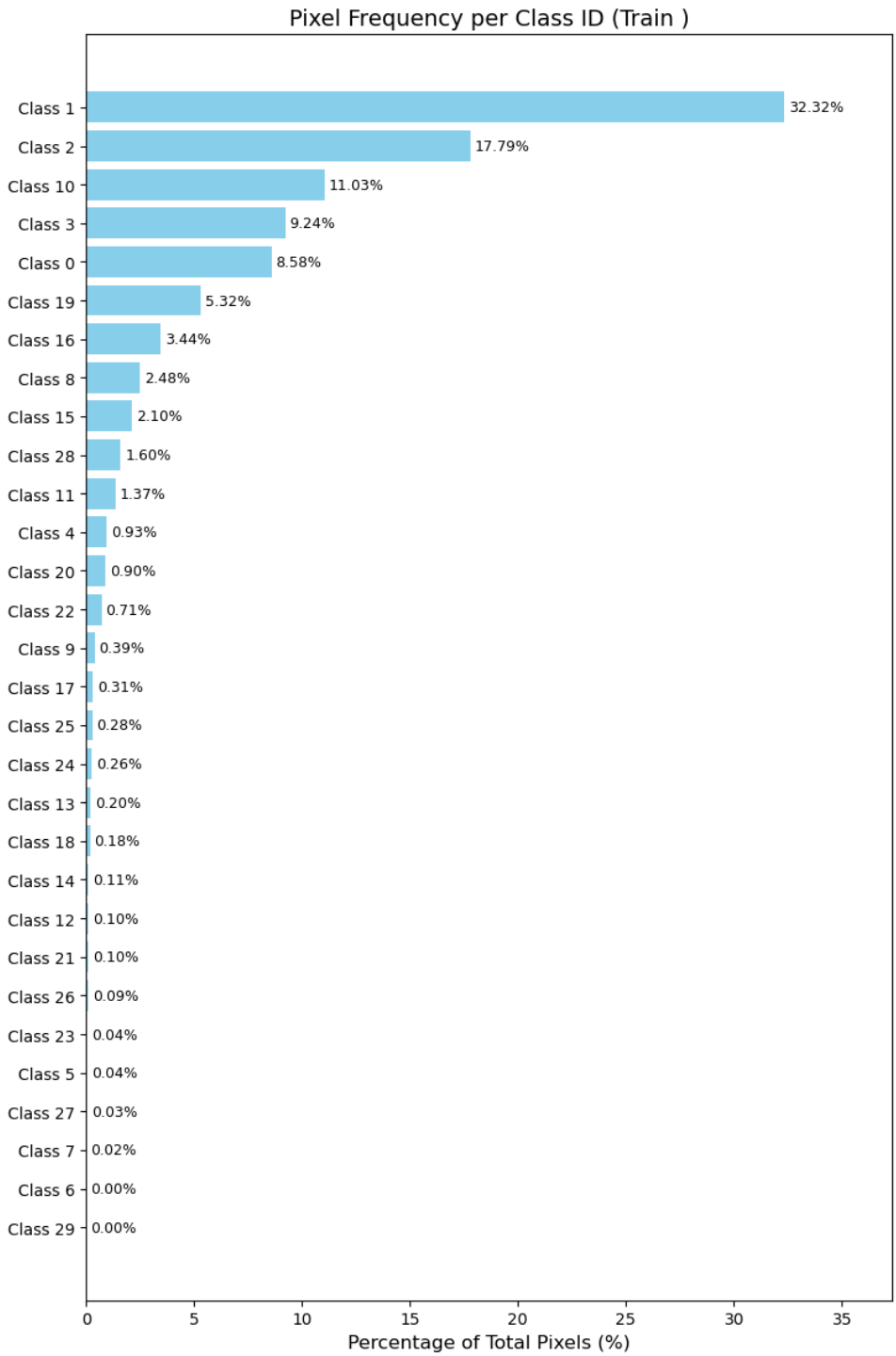


Figure 3.7: Pixel frequency per class

3.5 Malegaon Agricultural Dataset

This dataset was obtained from the Department of Civil Engineering, Indian Institute of Technology Bombay.The images were collected from a farmland region in Malegaon, Nashik.

The data consists of unlabeled multi-spectral images obtained at a resolution of

5cm/pixel by flying a UAV at a height of 53 meters. The multi-spectral image comprises of four bands stacked in the order Green, Red, RedEdge and NIR. For better segmentation, the orthomosaic image obtained from Parrot Sequoia camera is converted into tiles of size 512 × 512 pixels.

Figure 3.8: Sample image from multispectral dataset

Chapter 4

Study Methodology

This chapter provides the description of all the experiments conducted involving the training details and results followed by their limitations.

4.1 Agriculture Vision Dataset

4.1.1 Experiment with DeepLabV3+

Key Details

- DeepLabV3+ model with aligned Xception backbone was used to conduct this experiment. The output stride set to 16.
- Adaptive Class Weighting (ACW) loss function from the MSCG-Net paper [18] was used as the loss function as it handles class imbalance well. It was modified for multi-label segmentation by replacing the initial softmax activation with sigmoid for efficiently computing loss of all classes against their respective ground truth. For loss computation, a pixel can have non zero probability to belong to multiple classes. Non-valid farmland pixels were ignored for loss computation and evaluation
- The backbone was first initialized with pretrained weights from ImageNet — a large-scale dataset for image classification. DeepLabV3+ expects 3 channels RGB whereas AgriVision dataset has 4 bands. Thus, the first convolution layer of the backbone was modified to take 4 channels. The weights for the NIR channel were obtained by just replicating the weights of red channel.
- The model was trained with Adam optimizer and using differential learning rates - 10^{-4} for the backbone and 10^{-3} for ASPP + Decoder.

Results

Class	Color (RGB)	Description
double_plant	(255, 0, 0)	Red
drydown	(255, 128, 0)	Orange
endrow	(255, 255, 0)	Yellow
nutrient_deficiency	(0, 255, 0)	Green
planter_skip	(0, 255, 255)	Cyan
storm_damage	(0, 0, 255)	Blue
water	(128, 0, 255)	Purple
waterway	(255, 0, 255)	Magenta
weed_cluster	(128, 128, 128)	Gray

Table 4.1: Class-to-color mapping used for visualizing segmentation masks.

The trained model was evaluated on validation dataset. Fig. 4.1 - 4.8 shows some of the predicted segmentation results and their respective Ground Truth (GT) images for several classes of AgriVision dataset.

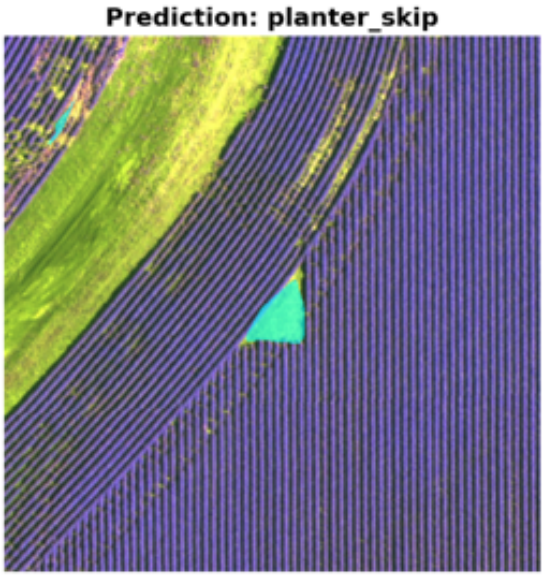


Figure 4.1: Prediction (planter skip)

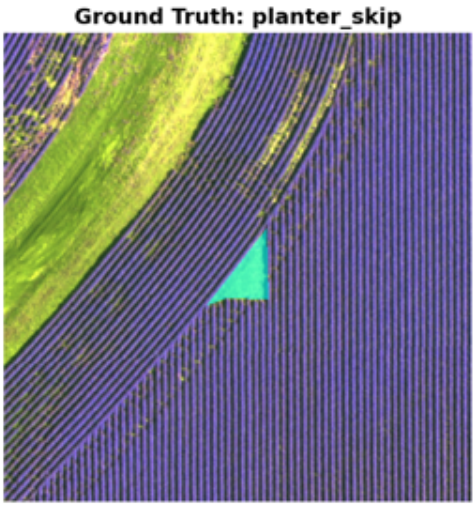


Figure 4.2: Ground Truth (planter skip)

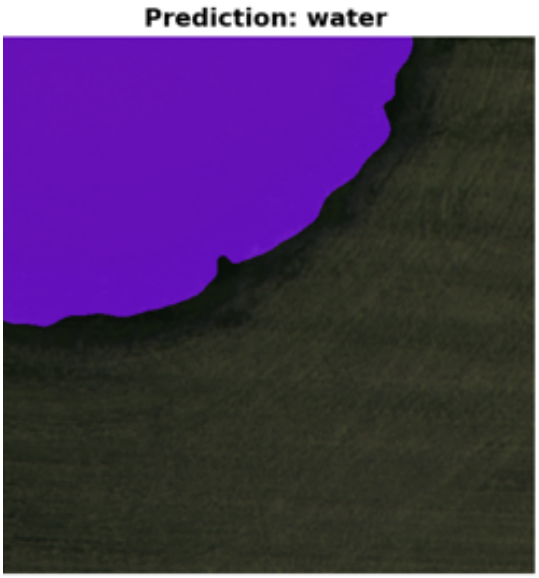


Figure 4.3: Prediction (water)

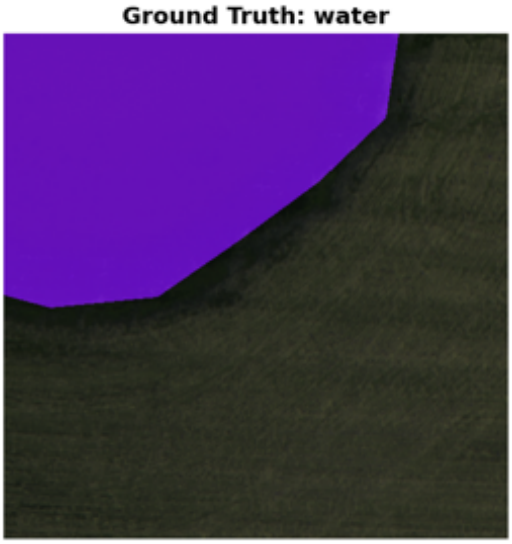


Figure 4.4: Ground Truth (water)

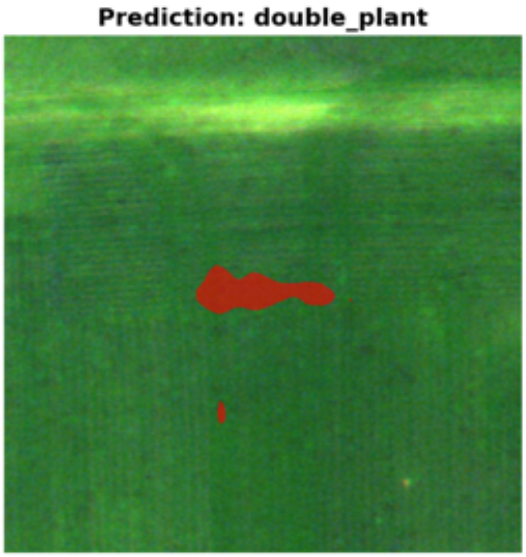


Figure 4.5: Prediction (double plant)

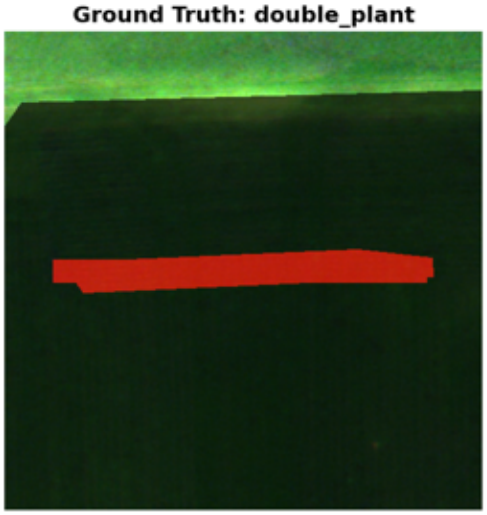


Figure 4.6: Ground Truth (double plant)

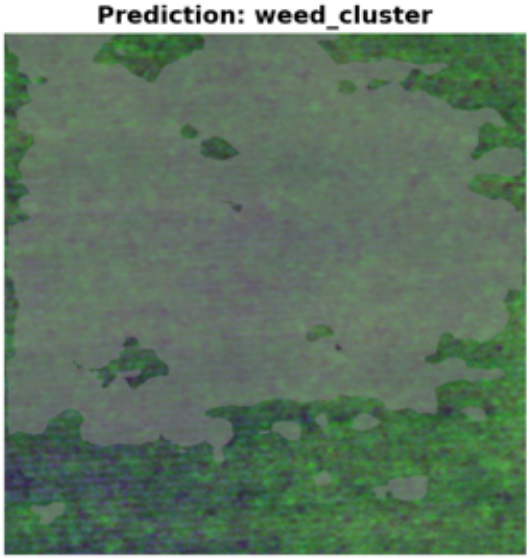


Figure 4.7: Prediction (weed cluster)

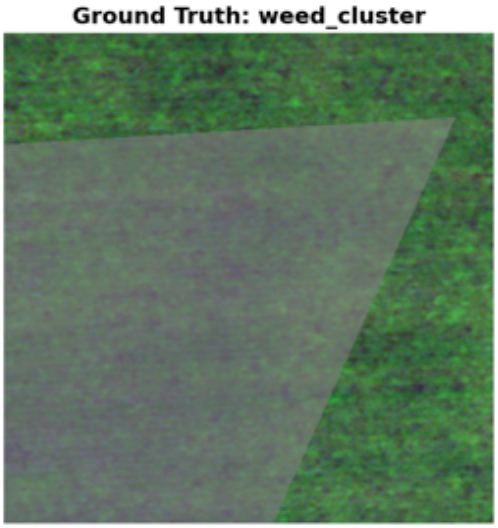


Figure 4.8: Ground Truth (weed cluster)

Intersection over Union (IoU), also known as the Jaccard index, was used as the evaluation metric that measures the overlap between predicted and ground truth segmentation masks.

$$\text{IoU} = \frac{|Prediction \cap GT|}{|P \cup GT|} \tag{4.1}$$

here GT = ground truth

Metrics

Table 4.2 shows the IoUs obtained for the classes of AgriVision dataset using DeepLabV3+.

Table 4.2: Per-class IoU Results

Class	**IoU**
double_plant	0.1969
drydown	0.5633
endrow	0.1919
nutrient_deficiency	0.3723
planter_skip	0.3502
water	0.6645
waterway	0.3746
weed_cluster	0.2564

Limitations

- The dataset is quite challenging with confusing segmentation labels and thus model doesn't perform very well even after taking into account class imbalance.

4.2 HiRISE Image Dataset

4.2.1 Knowledge Transfer from SAM

This method aims to leverage the knowledge of SAM to train a lightweight decoder for segmentation on a task that SAM is not specialized in.

Approach

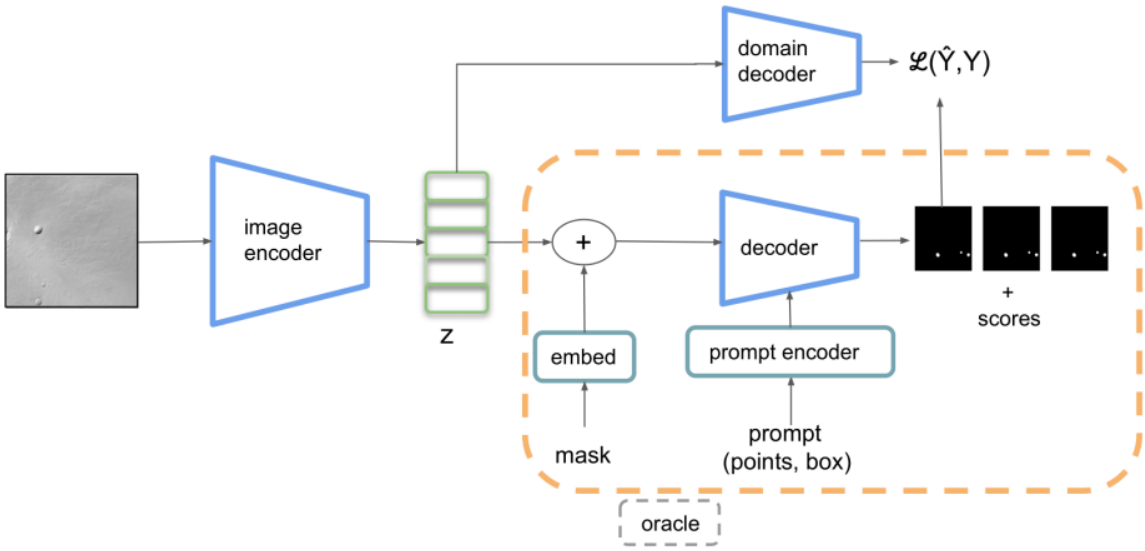


Figure 4.9: Overview of approach [14]

The feature embedding from SAM encoder is feeded as input to a separate domain decoder which is trained over a very small subset of images (17 images). The SAM image encoder is kept frozen during training the domain decoder. The domain decoder consists of 4 deconvolution layers to upsample the features followed by a sigmoid activation function to predict the binary segmentation masks. The data is unlabeled so final prediction of SAM can be used as ground truth and act as pseudo labels for training the domain decoder to indirectly distill its knowledge to the decoder. However, SAM automatic mask generation mode doesn't work well in all cases (example fig. 4.10) and providing point prompts for annotating may need some domain knowledge. So annotations were generated with the help of AI tools (Perplexity).

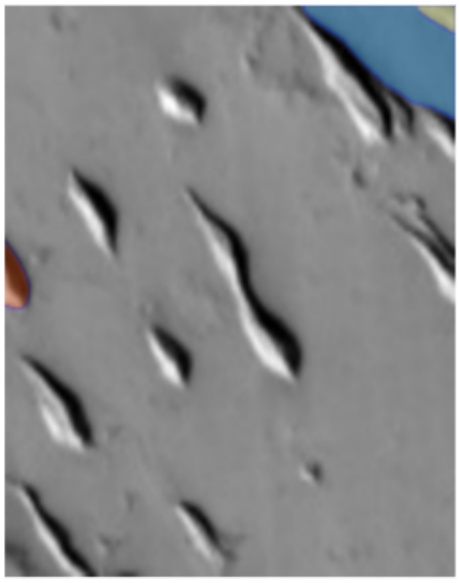


Figure 4.10: SAM automatic mask generation

SAM expects 1024x1024 images and the dataset resolution is 227x227. So to adapt it for downstream task- 4 images were padded to provide a 1024x1024 input to the SAM model as shown in fig. 4.11 .

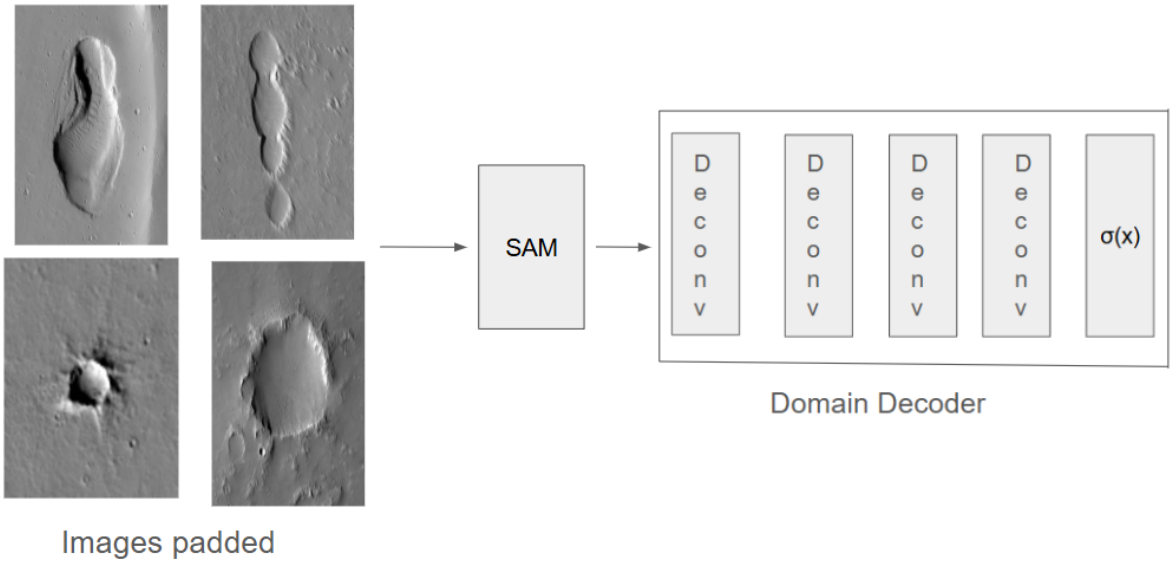


Figure 4.11: Domain decoder

Why not fine-tune decoder directly ?

Fine tuning SAM decoder can turn out to be effective when point prompts are available and when need human intervention to extract information from the ground truth. Also, SAM is trained on a large dataset which does not belong to the domain of this dataset and has thus learnt many patterns which are not relevant for this task. Thus, it may segment regions outside of martian landforms which are not relevant [14].

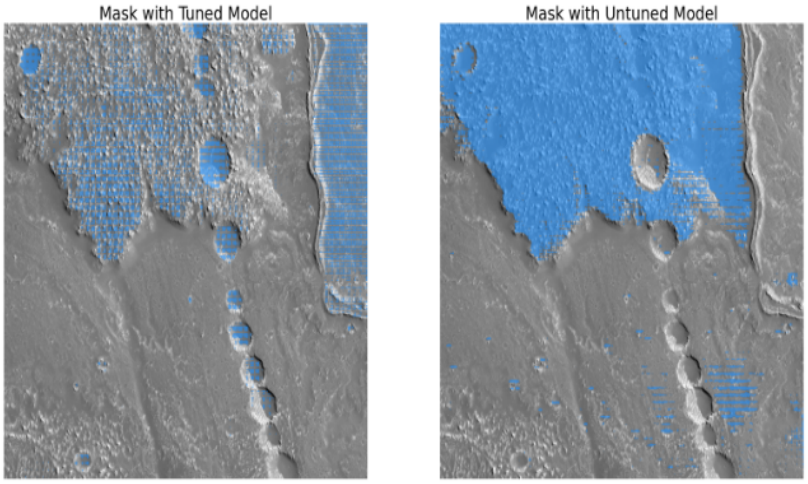


Figure 4.12: Example case where fine tuning the decoder fails [14]

Results

Fig. 4.13 shows the comparison between the deployed approach and the automatic mask generation by SAM.

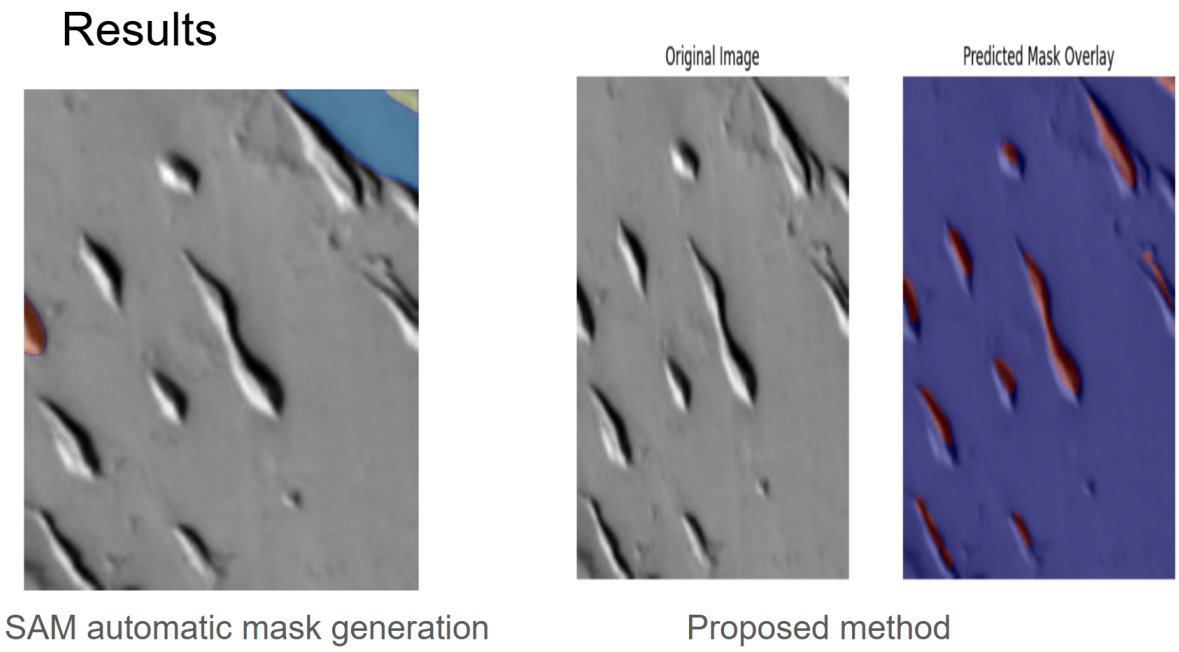


Figure 4.13: Comparison between the deployed approach and SAM automatic mask prediction

Fig. 4.14 shows a sample prediction on test image. Fig. 4.15 shows prediction on image containing the class- bright dune which was not seen by the domain decoder while training.

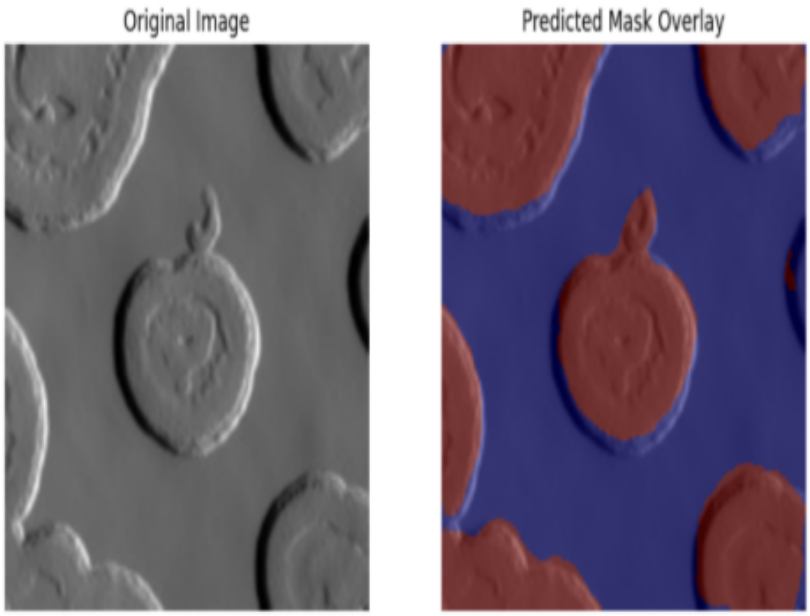


Figure 4.14: Prediction on test image

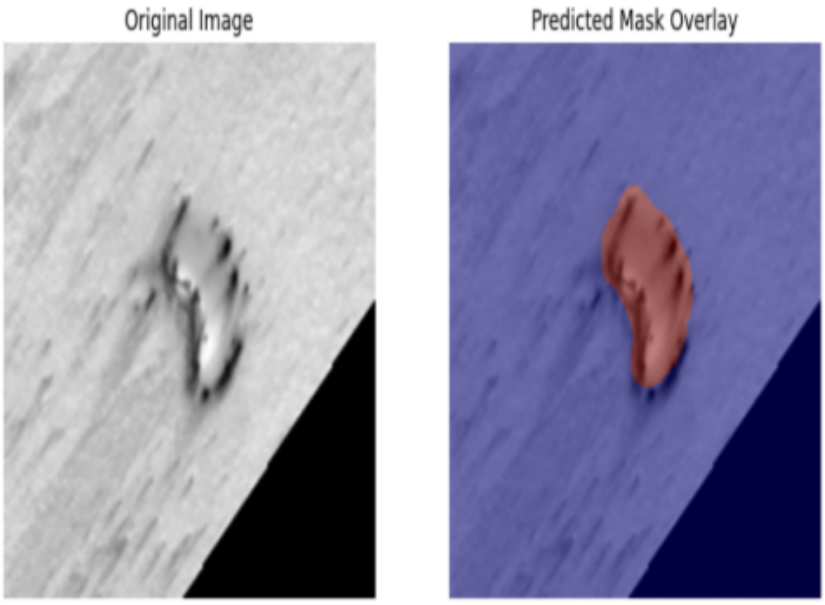


Figure 4.15: Prediction on bright dune (a class not seen by model during training)

Metrics

The dataset was evaluated on a very small test subset($<$ 10 images) with pseudo labels results obtained are shown in table 4.3. Since it is evaluated on a very small set it may not be very accurate to evaluate performance provided that true ground truths are also not available.

Table 4.3: Overall Segmentation Performance Metrics

Metric	Score
Mean IoU	0.5766
Mean Dice (F1 Score)	0.7182
Mean Recall	0.7837

Limitations

- Good quality ground truths are not available for training and evaluating the domain decoder.
- This approach works only for binary prediction and not for segmenting different kind of classes however it can be extended to handle multiclass segmentation if good quality semantic labels are available.

4.3 UAV-HSI-Crop-Dataset

4.3.1 Experiment with 3D U-Net

Spectral Attention + 3D U-Net

Principal Component Analysis(PCA) was applied to reduce hyperspectral dimensions from 200 to 30 bands preserving 99.66% of the variance. Each spectral channel was normalized independently using min-max scaling. An adaptive spectral weighting module that learns the importance of different spectral bands is added to the baseline 3D U-Net architecture.

The model takes input hyperspectral data of dimensions ($1 \times 30 \times 96 \times 96$), where 1 represents the channel dimension, 30 corresponds to the PCA-reduced spectral bands, and 96×96 denotes the spatial resolution. The base 3D UNet processes this input to generate feature maps of shape ($30 \times 30 \times 96 \times 96$), producing 30 feature channels for each of the 30 spectral bands.

The spectral attention mechanism then applies global average pooling along the spatial dimensions, followed by squeeze-and-excitation block consisting of two fully-connected layers with ReLU and sigmoid activation functions. The layers then learn the attention weights between 0 and 1 produced by sigmoid activation. These weights are element-wise multiplied with the original features and summed across the spectral dimension, effectively fusing the most relevant spectral information while suppressing less important bands.

The model then finally produces output of dimension (30 x 96 x 96) which maps to the ground truth label dimensions. The approach is shown in fig. 4.16 .

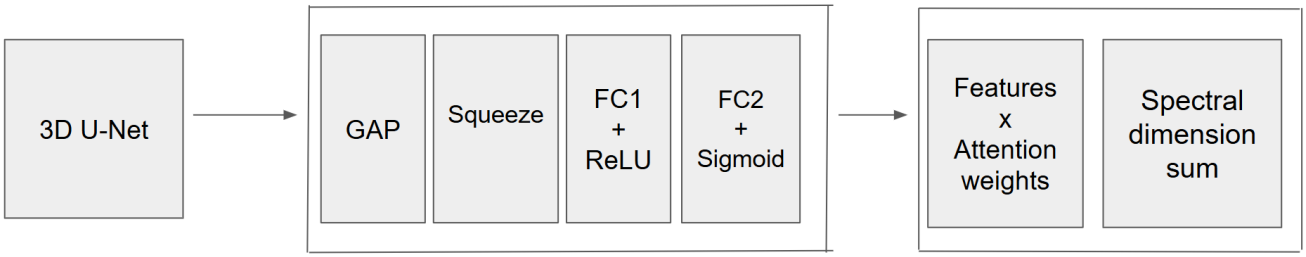


Figure 4.16: 3D U-Net with squeeze and excitation block

Training details

The model is trained end-to-end using a hybrid loss function that balances Cross-Entropy Loss (L_{CE}) and Dice Loss (L_{Dice}) equally:

$$L = 0.5 \times (L_{CE} + L_{Dice}) \tag{4.2}$$

Training is performed for 50 epochs using the Adam optimizer with a learning rate of 5×10^{-4}.

Results

Table 4.4: Class-to-RGB Color Mapping

Class	RGB	Class	RGB	Class	RGB
0	(31, 119, 180)	10	(214, 39, 40)	20	(247, 182, 210)
1	(31, 119, 180)	11	(255, 152, 150)	21	(127, 127, 127)
2	(174, 199, 232)	12	(148, 103, 189)	22	(199, 199, 199)
3	(255, 127, 14)	13	(148, 103, 189)	23	(199, 199, 199)
4	(255, 127, 14)	14	(197, 176, 213)	24	(188, 189, 34)
5	(255, 187, 120)	15	(140, 86, 75)	25	(219, 219, 141)
6	(44, 160, 44)	16	(196, 156, 148)	26	(219, 219, 141)
7	(44, 160, 44)	17	(196, 156, 148)	27	(23, 190, 207)
8	(152, 223, 138)	18	(227, 119, 194)	28	(158, 218, 229)
9	(214, 39, 40)	19	(247, 182, 210)	29	(158, 218, 229)

Figures 4.17 - 4.19 show some sample predictions against their ground truths.

Figure 4.17: Input image-1 (left), Ground Truth-1 (middle), Prediction-1 (right)

Figure 4.18: Input image-2 (left), Ground Truth-2 (middle), Prediction-2 (right)

Figure 4.19: Input image-3 (left), Ground Truth-3 (middle), Prediction-3 (right)

Metrics

Table 4.5 shows the IoU scores of some classes which the model observed more frequently in comparison to rest of the classes.

Table 4.5: Per-Class Intersection over Union (IoU) Scores on test set

Class	**IoU**
0	0.9818
1	0.7101
2	0.8144
3	0.7304
10	0.7040
15	0.3593
19	0.4086
20	0.1877
22	0.5164
28	0.2178

Overall Accuracy (OA) measures the ratio of correctly classified pixels to the total number of pixels. It is defined as:

$$OA = \frac{\sum_{i=1}^{N} n_{ii}}{\sum_{i=1}^{N} \sum_{j=1}^{N} n_{ij}} \tag{4.3}$$

where n_{ii} represents the diagonal elements of the confusion matrix (correct predictions), and n_{ij} denotes the total number of pixels belonging to class i but predicted as class j.

Cohen's Kappa Coefficient evaluates the agreement between predicted and true labels while correcting for chance agreement. It is given by:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{4.4}$$

where $p_o = OA$ is the observed agreement and p_e is the expected agreement by chance, computed as:

$$p_e = \frac{\sum_{i=1}^{N} (n_{i+} \cdot n_{+i})}{\left(\sum_{i=1}^{N} \sum_{j=1}^{N} n_{ij}\right)^2}$$

Here, n_{i+} and n_{+i} denote the sum of the ith row and i^{th} column of the confusion matrix, respectively.

The kappa coefficient obtained was 0.728

Limitations

- For some classes the IoUs were almost 0 and had very poor IoU scores while some of the classes had very good IoU scores. This is because of extreme class imbalance

present in the dataset. The classes which were present more frequently had good scores while some classes were almost absent in the training data and the model failed to segment them .

- The training data is very limited as compared to the number of classes and the spectral channels and thus the model is prone to overfit.

4.4 PhenoBench Dataset

4.4.1 Experiment with MSCG-Net

The experiment was done using the MSCG-Net (MSCG-Rx101) architecture, consisting of ResNet101 backbone. The model was trained using the Adaptive Class Weight (ACW) loss, which combines class-weighted pixel-wise loss with a differentiable Dice loss, helping handle class imbalance in the dataset as classes like weeds are present in scarce amounts. Optimization was performed using stochastic gradient descent (SGD) with momentum 0.9 and weight decay 1e-4, and a polynomial learning rate scheduler was applied over 50 epochs. Some sample results are shown in fig. 4.20 .

Results

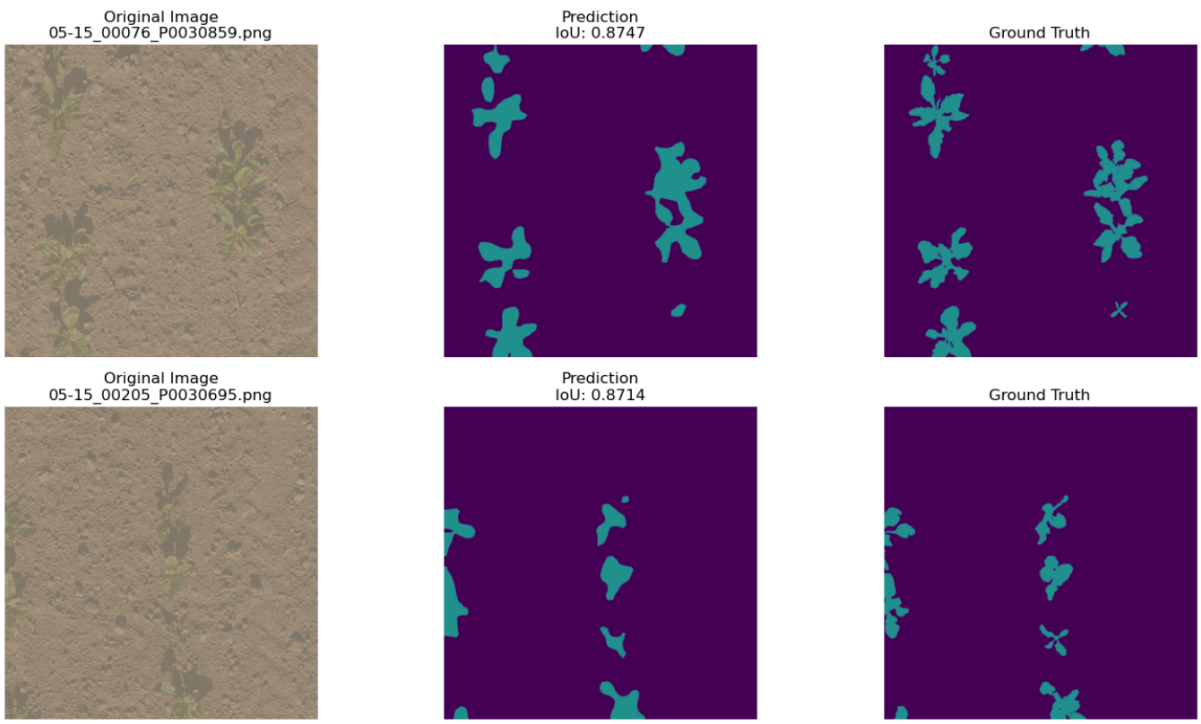


Figure 4.20: Sample results by MSCG-Net

Metrics

Table 4.6 shows the per-class IoU scores obtained.

Table 4.6: Per-class IoU for MSCG-Net on PhenoBench

Class ID	Class Name	IoU
0	Background	0.9749
1	Crop	0.7658
2	Weed	0.4029

Limitations

- MSCG-Net is primarily designed for segmenting objects which occupy more contiguous regions in images rather than small scale object segmentation and thus it had poor IoU for weeds.
- Although Graph Neural Networks(GNNs) can tackle effects like occlusion by understanding inter-relationships via the graph structure but for segmenting small scale objects an encoder-decoder architecture is generally more suited for upsampling and restoring feature resolution. MSCG-Net compresses the feature resolution and leads to loss of details which affects prediction on weeds.
- The experiment was mostly conducted to understand the model and its performance on a dataset other than AgriVision to see how the model can work at detecting and objects at small scale like weeds.

4.4.2 Experiment with DeepLabV3

The DeepLabV3 model was trained on the PhenoBench dataset using a ResNet101 backbone. The backbone outputs two feature maps: the intermediate layer3 features and the final layer4 features. The layer4 features are fed into the main classifier, implemented as a DeepLabHead, which produces the primary semantic segmentation output. During training, an auxiliary classifier FCNHead(consisting of 3x3 convolution followed by ReLU activation) is attached to the layer3 features to compute an auxiliary loss. This auxiliary loss provides additional supervision to the intermediate features, encouraging the network to learn more discriminative representations in the earlier layers and improving gradient flow as gradient becomes smaller as we go towards deeper layers.

The total training loss is computed as the sum of the main loss (from layer4 output) and 0.4 times the auxiliary loss (from layer3 output), both calculated using cross-entropy loss.

$$total_loss = main_loss + 0.4 * aux_loss \tag{4.5}$$

The model was optimized using stochastic gradient descent with momentum and weight decay. A step-wise learning rate scheduler was used to gradually reduce the learning rate.

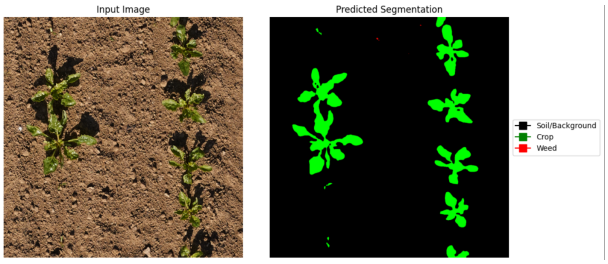


Figure 4.21: Sample prediction-1 on phenobench dataset

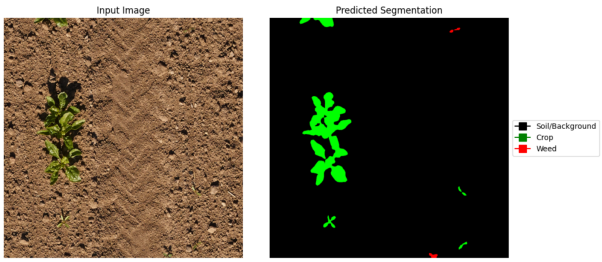


Figure 4.22: Sample prediction-2 on phenobench dataset

Metrics

The table 4.7 shows the per class IoU. The mIoU obtained was 0.79 on validation set.

Class	**DeepLabV3 IoU**
Soil	0.99
Crop	0.91
Weed	0.47
Mean IoU	**0.79**

Table 4.7: IoU performance of DeepLabV3 model on soil, crop, and weed classes.

Limitations

- Even though receptive field can be controlled by dilated convolutions , fine boundaries and feature details for small scale objects like weeds get lost and the model doesn't show very good IoU for weeds.
- DeepLabV3+ can be used to improve its performance which uses encoder-decoder architecture and is known to perform better in such segmentation tasks.

4.5 Malegaon Agricultural Dataset

4.5.1 Experiment with SAM

SAM is primarily an instance segmentation model but can be adapted for semantic segmentation via clustering algorithms.

In this approach, SAM was used to generate instance masks for a given 4-band UAV image (Red, Green, Blue, NIR). The input image is loaded its RGB channels, while the NIR band is used to compute the **Normalized Difference Vegetation Index (NDVI)** as:

$$\text{NDVI} = \frac{\text{NIR} - \text{Red}}{\text{NIR} + \text{Red}}, \tag{4.6}$$

SAM generates multiple binary masks corresponding to potential objects or regions in the image. For each generated mask, we compute the mean NDVI value of the pixels inside the mask. These mean NDVI values serve as feature vectors for clustering.

Since, number of classes were unknown for this dataset - to generate semantic classes, DBSCAN clustering was applied on the standardized mean NDVI features. DBSCAN groups masks with similar vegetation characteristics into clusters, automatically ignoring noise points. Each cluster is assigned a unique integer label, starting from 1, while masks classified as noise are assigned 0 (background).

Fig. 4.23 - 4.24 shows some sample predictions by this approach.

Results

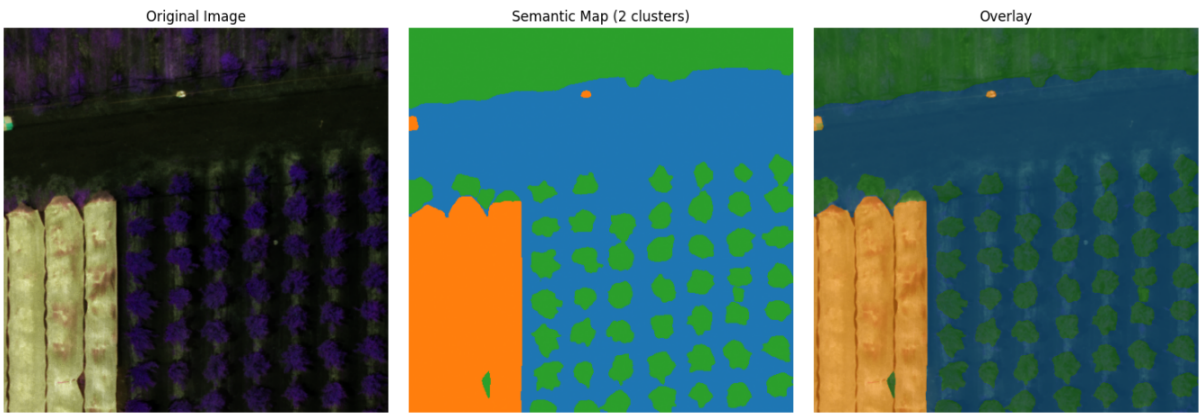


Figure 4.23: Segmentation result-1 by SAM

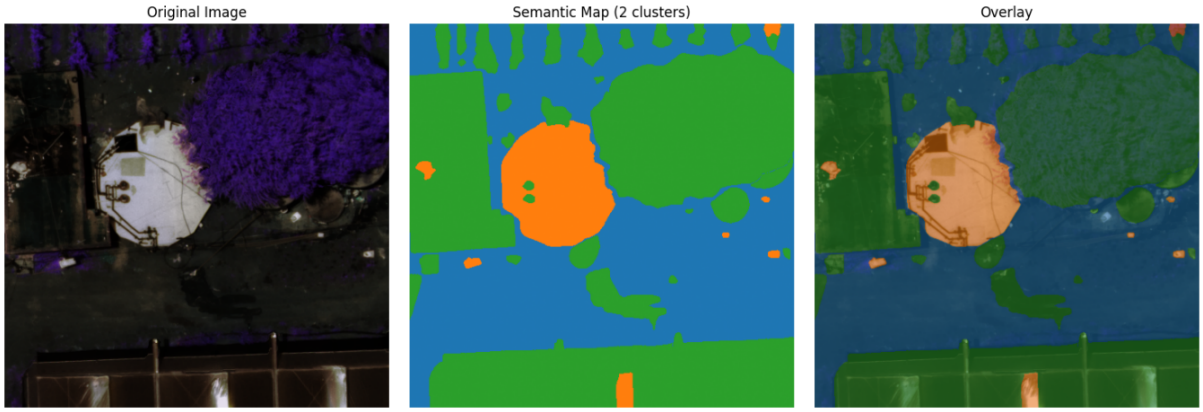


Figure 4.24: Segmentation result-2 by SAM

Limitations

- DBSCAN parameters - *min_points* and ϵ will require consistent tuning to get accurate segmentation.
- Lack of true labels make it difficult to predict segmentation result.
- Clustering solely based on NDVI value may not always be effective for segmentation.

Chapter 5

Conclusion and Future Work

This study on image segmentation in aerial and satellite imagery has demonstrated the effectiveness of various advanced techniques, including zero-shot segmentation with Vision Transformers and SAM, as well as DeepLab models, across diverse datasets such as PhenoBench, HiRISE, Agriculture-Vision, UAV-HSI-Crop, and Malegaon Agricultural Dataset. The experiments highlight the strengths and limitations of these approaches in handling challenges such as class imbalance, reduced feature resolution, and multi-scale object detection. The results indicate that while pretrained models can achieve promising segmentation performance, particularly on RGB imagery, there remains substantial room for improvement in multispectral and hyperspectral domains. In particular, datasets like Agriculture-Vision continue to pose challenges where no single model achieves consistently strong performance.

Future work will focus on exploring models specifically designed for multispectral and hyperspectral image segmentation with a particular focus on AgriVision dataset by leveraging architectures like SpectralGPT, which can better exploit spectral information. Additionally, the application of semi-supervised and self-supervised learning techniques will be important, as most agricultural datasets are either partially labeled or lack high-quality ground truth annotations. Investigating lightweight domain-adaptive models and strategies to handle class imbalance more effectively will also be critical for developing scalable and generalizable segmentation solutions. Overall, the study lays the groundwork for further research into robust, cross-domain image segmentation techniques that can support precision agriculture in a variety of real-world scenarios.

Bibliography

[1] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al., "Segment anything," Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4015–4026, 2023

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[4] Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." arXiv preprint arXiv:1706.05587 (2017).

[5] Chen, Liang-Chieh, et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation." Proceedings of the European conference on computer vision (ECCV). 2018.

[6] https://learnopencv.com/deeplabv3-ultimate-guide/

[7] He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." IEEE transactions on pattern analysis and machine intelligence 37.9 (2015): 1904-1916.

[8] https://viso.ai/deep-learning/xception-model/

[9] Gülmez B. A novel deep neural network model based Xception and genetic algorithm for detection of COVID-19 from X-ray images. Ann Oper Res. 2022 Dec 25:1-25. doi: 10.1007/s10479-022-05151-y. Epub ahead of print. PMID: 36591406; PMCID: PMC9790088.

[10] Gou, Jianping, et al. "Knowledge distillation: A survey." International journal of computer vision 129.6 (2021): 1789-1819.

[11] https://www.ibm.com/think/topics/knowledge-distillation

[12] Çiçek, Özgün, et al. “3D U-Net: learning dense volumetric segmentation from sparse annotation.” International conference on medical image computing and computer-assisted intervention. Cham: Springer International Publishing, 2016.

[13] Weyler, Jan, et al. “Phenobench: A large dataset and benchmarks for semantic image interpretation in the agricultural domain.” IEEE transactions on pattern analysis and machine intelligence 46.12 (2024): 9583-9594.

[14] Julka, Sahib, and Michael Granitzer. “Knowledge distillation with segment anything (SAM) model for planetary geological mapping.” International Conference on Machine Learning, Optimization, and Data Science. Cham: Springer Nature Switzerland, 2023.

[15] Gary Doran, Emily Dunkel, Steven Lu, & Kiri Wagstaff. (2020). Mars orbital image (HiRISE) labeled data set version 3.2 (3.2.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.4002935

[16] Y. W. Z. H. A. S. R. B. H. K. H. K. I. D. Mang Tik Chiu, Xingqian Xu, “Agriculture Vision: A Large Aerial Image Database for Agricultural Pattern Analysis,” CoRR, vol. abs/2001.01306, 2020.

[17] Niu Bowen, Feng Quanlong, Chen Boan, et al. UAV-HSI-Crop-Dataset[DS/OL]. V6. Science Data Bank, 2022[2025-10-22]. https://cstr.cn/31253.11.sciencedb.01898. CSTR:31253.11.sciencedb.01898.

[18] Liu, Qinghui, et al. “Multi-view self-constructing graph convolutional networks with adaptive class weighting loss for semantic segmentation.” Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020.

[19] Qinghui Liu, Michael Kampffmeyer, Robert Jenssen, and Arnt-Børre Salberg. Self-constructing graph convolutional networks for semantic labeling. arXiv preprint arXiv:2003.06932, 2020