ECE549 Final Report
Agricultural Image Segmentation

Ufuk Soylu, Ankit Raj, Yuqi Li, Berk Iskender

***Abstract*—Although there has been a lot of progress in computer vision on the different aspects of understanding and analyzing the natural images obtained using commercial imaging devices, aerial images that are captured using airborne devices are still mostly unexplored. In this project, we try to address a very challenging and practical application of segmentation of aerial agricultural images. We study and analyze different aspects related to deep learning based segmentation – *(i)* comparison between several successful deep neural network architectures, *(ii)* the effect of the training loss, *(iii)* using weighted loss to tackle imbalanced classes, *(iv)* comparison between shallow and wider network, and *(v)* improvement due to pre-training.** [1]

I. Introduction

In this project, we study and analyze different algorithms for aerial image semantic segmentation in agriculture which is of a massive economic potential. We think that solving this problem will eventually pave the way to sustainable agriculture by helping the government and farming industry with more relevant information about the farmland. This will be helpful for meeting society's present food and textile needs without compromising the ability of future generations to meet theirs. For this segmentation task, we use deep neural network based algorithms as they have been proven to be very successful for large-scale computer vision problems in diverse areas such as medicine [1] and astronomy [2].

We use the Agriculture-Vision dataset [3] that consists of multi-modal images (RGB image along with near-infrared measurement channel). We examine the dataset, whose description can be found in Section II. This analysis reveals how few categories have scant images compared to others, which might lead to over-fitting of the network on categories having abundant samples or very poor performance on categories having very few samples. To avoid this poor-performance, we explore the usage of weighted-loss which penalizes the network (depends on the weight value) if it mis-classifies for categories with scant samples, while training. Further, we compare two loss functions popularly used for multi-class segmentation problems and find that *BinaryCrossEntropy*$(\cdot)$ gives the best performance. We implement 4 state-of-the-art architectures for segmentation – *(i)* U-Net, *(ii)* Feature Pyramid Network (FPN), *(iii)* DeeplabV3, and its variants, and *(iv)* Dilated Residual Network (DRN), and do extensive comparison between them. Additionally, we experiment on how deeper networks (as backbone or as feature extractor) help in improving the segmentation performance. Details of the proposed approach and experiments along with extensive results are in Sections III and IV.

[1] Video link: https://youtu.be/Wn4td1cU-6E

II. Dataset

We have used a subset of the large-scale aerial agricultural Agriculture-Vision image dataset [3] which was published for a challenge in CVPR2020 (https://www.agriculture-vision.com/dataset). The dataset contains high-resolution, multi-band images and has corresponding multiple types of patterns annotated by experts. The subset contains 21061 aerial farmland images captured across the US. It contains 12901 training and 4431 validation images. Each image consists of four 512x512 color channels, which are RGB and Near Infra-red (NIR). Each image has a boundary map and a mask. The boundary map indicates the region of the farmland, and the mask indicates valid pixel labeling of the image. The dataset contains six types of annotations: Cloud shadow (C.S.), Double plant (D.P.), Planter skip (P.S.), Standing Water (S.W.), Waterway (W.) and Weed cluster (W.C.). These field anomalies have great impacts on the potential yield of farmlands, thus accurately locating them is extremely important. These six patterns are stored separately as binary masks due to potential overlaps. The organizers have kindly agreed to share the dataset with us.

A. Dataset Examples

Images have the resolution up to 10 cm per pixel (cm/px). Custom filters are used to obtain NIR instead of blue channel. Seperate blue channel images were captured at 20 cm/px resolution - upsampled later to align with corresponding NIR-R-G images. Images are normalized using the 5^{th} and 95^{th} percentile pixel values as thresholds. Then, images are stored as NIR and RGB separately in JPG format. Since unprocessed images have extremely large sizes and sparse annotations, subsampling methods such as flipping and cropping is used (preserving the anomaly patterns), to increase the proportion of annotated pixels and to decrease the computation time and memory consumption. Final image size is 512×512. In Fig. 1, these techniques can be observed.

B. Labeling Issues

Labels are not necessarily exclusive, meaning a pixel labelled as weed cluster can also be labelled as double plant. This is quite different from the common semantic segmentation problem, where the labelled pixel belongs to only one of the classes. We will explain more on designing the loss for such multiple labels in III-B. Furthermore, after examining the

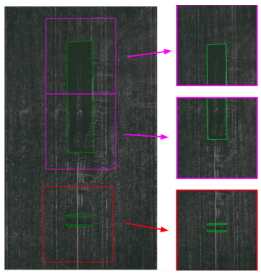

Fig. 1: Illustration of field image patch generation for AgriVision.

dataset, it was found that several labelings were problematic, making this problem even more challenging.

C. Pixel-wise Label Ratios

After inspecting the training data for the ratios of different labels, we observed that the vast majority of the non-background pixels (including at least one label) belong to the weed cluster and the ratios of each class label over the total number of labels in all pixels are as follows:

- Weed cluster label ratio: 70.1%
- Waterway label ratio: 5.8%
- Standing water label ratio: 7.1%
- Planter skip label ratio: 0.7%
- Double plant label ratio: 3.0%
- Cloud shadow label ratio: 13.3%

These ratios suggest that to segment minority classes better, we may need to specify our loss function with a specific weighting. Otherwise, a network would be much more sensitive towards detecting areas with dominant classes.

III. Proposed Approach

A. Network Architectures and experiments

For image segmentation, we require the network to learn information at different scales as the object of interest can be of any size within the image. This is particularly critical for very high resolution images with small regions of interest, which have to be segmented. We plan to implement four networks types and compare their performance on this multiband agricultural image dataset (Feature Pyramid Networks, Dilated Residual Networks, DeepLabV3 and U-net.)

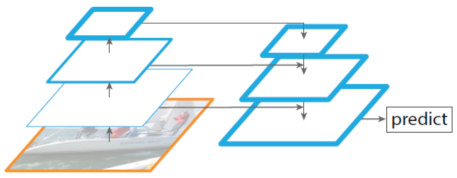

Fig. 2: Architecture of Feature Pyramid Network (FPN) – *left* part is the bottom-up pathway, acting like encoder, and *right* part is top-down, upsampling and combining information using lateral connection from bottom-up at several scales to get the prediction.

1) FPN-based Segmentation: Feature Pyramid Networks (FPNs) [4] have been particularly popular in object detection,

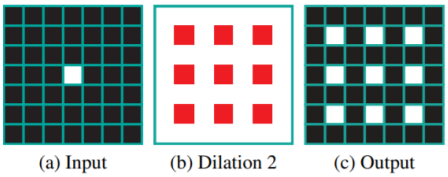

Fig. 3: A example of gridding artifact [5].

but recently been used for segmentation task as well. FPN can be broadly divided into two parts – Bottom-Up pathway and Top-Down pathway and Lateral Connection. The bottom-up pathway is the feed-forward computation of the backbone ConvNet, typically ResNet. One pyramid level is defined for each stage. The output of the last layer of each stage will be used as the reference set of feature maps for enriching the top-down pathway by lateral connection. In the Top-Down pathway, the higher resolution features is upsampled spatially coarser, but semantically stronger, feature maps from higher pyramid levels. More specifically, the spatial resolution is upsampled by a factor of 2 using the transposed convolution. Each lateral connection merges feature maps of the same spatial size from the bottom-up pathway and the top-down pathway. The high-level architecture is shown in the Fig. 2. In our experiments, we have used first 4 blocks of ResNet-50 or ResNet-18 for the bottom-up pathway. For top-down, standard 3×3 transposed convolutions have been used at every level for upsampling followed by ReLU activation and 1×1 convolutions. For lateral connections from the feature maps in the bottom-up part, we use two 3×3 convolutions followed by ReLU activation and 1×1 convolutions. The upsampled feature map and lateral connections are added together and passed through two convolution layers along with ReLU. These 3 operations – upsampling, lateral connection, convolution on addition of upsampling and lateral connection are done at three stages (for four blocks/levels of ResNet).

2) DRN: Dilated Residual Networks (DRNs) [5] provide an easy and effective way of making a convolution layer to capture semantics at different scale without increasing overhead in terms of computation. They use different atrous convolutions (also known as dilated convolution, or convolutions with holes). By varying number of holes, the amount of receptive field can be changed significantly – by allowing more holes, the receptive field increases with same number of parameters in the convolution kernel. Different layers having different dilations allow to capture information at different scales which is vital for segmentation, particularly when the object of interest occupies quite small region of the image. However, the dilated convolution layers also introduces the gridding artifact (Figure 3): when a feature map has higher-frequency content than the sampling rate of the dilated convolution, the final semantics appear disconnected and has holes around the edge of objects. This artifact is compensated by adding more blocks and removing residual connection from some the added blocks. We have the implementation of the DRNs in Pytorch publicly available on Github (https://github.com/fyu/drn) and

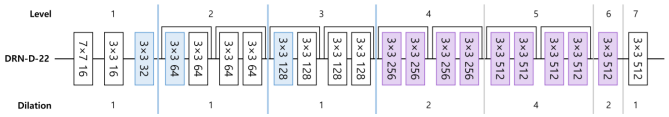

Fig. 4: Layer architectures of DRN-D-22 network [6].

we adopted the `DRN-D-22` structure. In Figure 4, Each rectangle represents a Conv-BN-ReLU group, and the numbers specify the filter size and the number of channels in that layer. Blue colored rectangles are Conv-BN-ReLU groups with stride 2, and downsampling occurs in blue lines. The purple colored rectangles adopt dilated convolutions with the dilated factors described beneath each network architecture. We trained this DRN with SGD optimizer for 35 epochs with learning rate 0.01.

3) DeepLabV3 & DeepLabV3+: A very recent work [7] proposed DeepLabV3 architecture for the task of semantic segmentation. In this paper, they mention that there are two challenges in applying Deep Convolutional Neural Networks: (i) reduced feature resolution due to pooling operations, (ii) existence of objects at multiple scales. To overcome these challenges, they use atrous convolution in the framework of spatial pyramid pooling. In particular, they proposed to benefit from atrous convolution as a context module and tool for spatial pyramid pooling. The network architecture can be found in Figure 5.

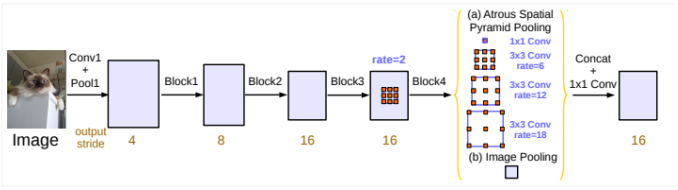

Fig. 5: Architecture of DeepLabV3 [7].

DeepLabV3 architecture is improved in [8] by adding a simple and effective upsampling path which consists of several bi-linear upsampling steps with skip connections. In [8], they named their architecture as DeepLabv3+ which can be seen in Figure 6.

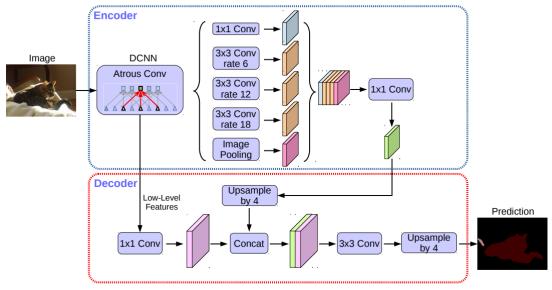

Fig. 6: Architecture of DeepLabV3+ [8].

DeepLabV3+ architecture consists of an endocer and a decoder. In [8], they propose to use DeepLabV3 architecture as the encoder. DeepLabV3 output stride is generally 16. Therefore, their proposed decoder structure takes output stride of 16 as the input. They first propose to bi-linearly upsample by a factor of 4 and then concatenate with corresponding low level features. After concatenation, they apply additional convolution layer to refine the features and subsequently, they apply bi-linear upsampling by a factor of 4. In [8], they empirically show that adding decoder brings about 0.8% improvement when using output stride$=$ 16. Based on the observation in [8], we propose to improve the upsampling path in DeepLabV3 architecture. Our modification is not exactly same as DeepLabV3+. We propose to use *Conv2DTranspose* layer to learn the upsampling rather than using simple bi-linear upsampling. Similar to DeepLabV3+ architecture, we propose to use skip connections between low level feature maps and corresponding upsampling layers. We denoted the proposed network as *Modified-DeepLabV3+* in the following sections.

4) U-Net: This architecture [9] is built upon the architecture of "fully convolutional network" [10]. It consists of a contracting and an upsampling part. Both parts are almost symmetric in terms of the number feature channels. Overall structure does not include any fully connected layers and segmentation map only consists of pixels for which full information is available in the input image allowing smooth segmentation by "overlap-tile" strategy by adjusting the tile size in accordance with the scale of pooling operations. This is especially important when the task is to segment large images as in our case, decreasing GPU memory utilization while preserving the resolution. Overall architecture can be seen in Fig. 7. Each step in the contracting part consists of two 3x3 convolutions, including a ReLU operation as nonlinearity and after the second 3x3 convolution, 2x2 max pooling is applied with a stride of 2 for downsampling. Number of feature channels are doubled at each step. In the expansive path, every step includes an upsampling of the input followed by a 2x2 convolution (up-convolution), halving the number of channels. Then, output of up-convolution is concatenated with the corresponding intermediate output from the contracting path and two consecutive 3x3 convolutions are performed with the non-linearities being ReLU layers as in contracting path. Final layer includes a 1x1 convolution layer maps multi-component feature vector to the number of classes used in the algorithm.

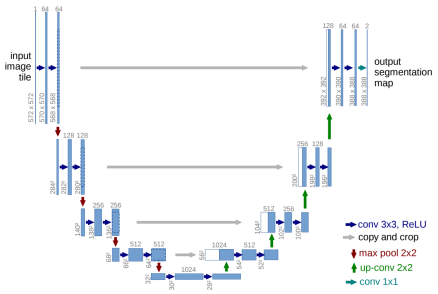

Fig. 7: Architecture of U-Net (32x32 pixels at lowest resolution). The number of channels at the top, x-y size at the lower edge of each box.

B. Training Losses

We decided to use two types of training losses as follows:

1) Binary Cross Entropy (BCE) Loss: We assume the output dimension of the network is $6 \times 512 \times 512$, and each channel of the output represents the binary label map of each class type. Hence it is suitable to use Binary cross entropy Loss which builds upon the idea of entropy from information theory. To compare two 2D binary image of size P, we define BCE as follows:

$$BCE = \frac{1}{6P} \sum_{j=1}^{6} \sum_{i=1}^{P} -t_{ij} \log(f(s_{ij})) - (1-t_{ij}) \log(1-f(s_{ij})) \tag{1}$$

where f is sigmoid function, i is the iterator of pixel location, j is the iterator of labels, $t_{ij} \in \{0, 1\}$ is the class, s_{ij} is the score (output of the network). We used average BCE as a training loss. Average BCE was computed by averaging BCE over all pixels and labels. According to the data, we also experimented using a specific weighting scheme as explained in Section II-C and IV-D.

2) Cross Entropy (CE) Loss: In this case, we assume the output dimension of the network is $7 \times 512 \times 512$, where the 0-th channel of the output represents the logit of farmland background and the rest six channels represents the logit of six other class types. Compared to the BCE loss described above, this loss discourages overlapping between class types and the pixel's class are exclusive. We can transform the output size to $6 \times 512 \times 512$: let $z \in \mathbb{R}^7$ denote the logits of one pixel, then we map it to a 6×2-dimensional matrix w: $w[i] = softmax(z[0], z[i+1])$, then apply the common cross entropy loss to w and labels and take the sum over all pixels. The last step is essentially the same as BCE loss.

C. Evaluation

We used mean Intersection-over-Union ($mIOU$) to determine the quality of the predicted semantics which is a common metric for segmentation tasks. For multi-class segmentation problem, it is defined as follows:

$$mIOU := \frac{1}{C} \sum_{c} \frac{\text{Area}\,(P_c \cap T_c)}{\text{Area}\,(P_c \cup T_c)} \tag{2}$$

where C is the number of classes, P_c and T_c are the predicted mask and ground truth mask of class c respectively.

IV. Experiments and Results

In this section, we describe and analyze different experiments and their results. To this end, we compare the efficacy of different architectures for the task on aerial agriculture dataset. Subsequently, we do detailed study of several aspects of modification in deep learning training within each architecture, and how they impact the performance, for e.g. – change in training loss, using pretrained network for sub-network initialization, weighted loss (with weights depending on proportion of each class in training data), wider (or shallower) vs deeper network study.

A. Loss functions comparison

We use BCE loss (Sec. III-B1) and CE loss (Sec. III-B2) to train the DRN with the same network architecture, number of epochs, optimizer and learning rate. The evaluation on the validation set is shown in Table I. The mIoU using BCE loss is significantly higher than that of CE loss. This can be explained by the fact that a pixel in an image in the dataset might correspond to multiple classes. Since we define a single background (*BG*) class for all the non-background classes, the loss function will try to activate the *BG* class (corresponding to inactive non-*BG* class(es)) and deactivate the *BG* class (corresponding to active non-*BG* class(es)) at the same time. This leads to confusion while training which might explain the poor performance of the CE loss. Hence, subsequently, we adopted the BCE loss for our study.

Losses	Weed	W.way	S.W.	P.S.	D.P.	C.S.	mIoU
CE	0.167	0	0	0	0	0	0.028
BCE	0.355	0.231	0.137	0.036	0.104	0.186	0.175

TABLE I: DRN: IOUs of each class trained with different losses.

B. Comparison across different models

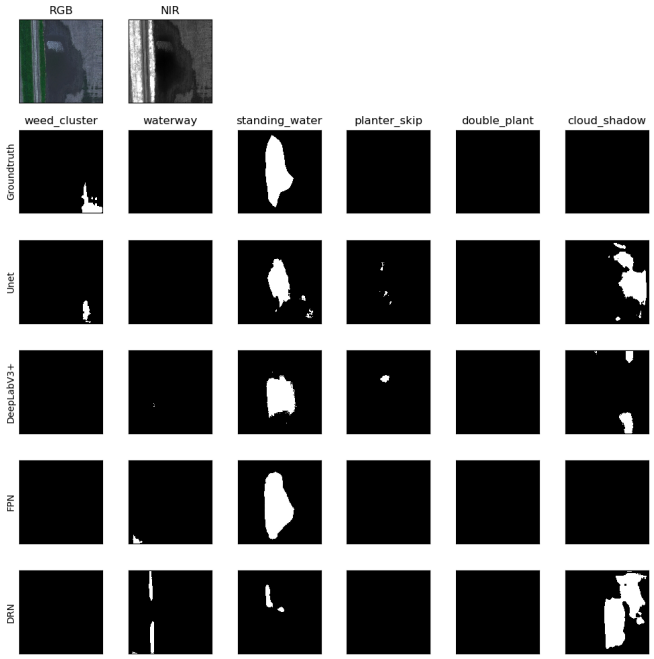

Fig. 8: Results using different networks for a sample

We compare the performance across the best implementation of the different architectures discussed in Section III-A. All the networks have been trained using the Objective (1). Since different segmentation architectures have been proposed in the literature over recent few years, this experiment indicates which architecture is particularly more helpful for the agriculture dataset. Figure 8 shows the result of the segmentation of all classes for one sample.

Architecture	Weed	Waterway	StandingWater	PlanterSkip	DoublePlant	CloudShadow	mIoU
U-Net	0.477	**0.345**	0.266	0.002	0.340	**0.459**	0.315
DeepLabV3+	0.469	0.339	0.316	0.008	0.241	0.363	0.289
FPN	**0.481**	0.325	**0.330**	**0.054**	**0.361**	0.430	**0.331**
DRN	0.355	0.231	0.137	0.036	0.104	0.186	0.175

TABLE II: Comparison across different architecture trained using the Objective (1) – we have used our best performing network for each architecture for evaluation.

C. DeepLabV3 & Modified DeepLabV3+

Subsequently, we use the same training scheme for experiments on both, DeepLabV3 and DeepLabV3+. Each network is trained for 50 epochs with a batch-size of 6. The *torch.optim.RMSprop()* optimizer with a learning rate of 10^{-6} and momentum=0.9 has been used to optimize the loss function. The network parameters that produce the lowest BCE Loss in validation set is saved and used for quantitative evaluation. It has been observed that validation error saturates around $10-15$ epochs. Another issue is finding the right threshold of the sigmoid output from the network as for the evaluation of quantitative metric IoU, hard outputs are required. So, threshold level for each classes are selected by doing a line search on validation set.

Learning Upsampling Path. In this experiment, we compare DeepLabV3 architecture and modified DeepLabV3+ architecture described in the previous section. So, the main comparison is between bi-linear upsampling by a factor of 16 and learning upsampling path by using Conv2DTranspose layers. In order to make a fair comparison, we fix the number of network parameters at 8.5×10^6. It has been observed that learning upsampling path improves the $mIoU$ by 3.7%. IoU results for each class can be found in Table III.

Larger Network. After observing that modified DeepLabV3+ architecture performs better, we increased the network capacity by adding more channels, layers and skip connections. The learnable parameters increased from 8.5×10^6 to 20×10^6. The capacity is increased by adding layers in encoder part. The encoder part is depicted in Figure 5. In order to increase the depth, we added more Block3 units which uses a dilated convolution layer with rate of 2. In order to increase the width, we increase the number of channels in all layer. It has been observed that larger network increased $mIoU$ by 14.8%. Intersection over Union(IoU) results for each classes can be found in Table IV.

	W.	W.way	S.W.	P.S.	D.P.	C.S.	mIoU
(i)	0.43	0.259	0.376	0.03	0.13	0.23	0.24
(ii)	0.44	0.287	0.312	0.001	0.178	0.29	**0.252**

TABLE III: Learning the upsampling path; (i) DeepLabV3 architecture, (ii) Modified DeepLabV3+ architecture.

	W.	W.way	S.W.	P.S.	D.P.	C.S.	mIoU
(i)	0.444	0.287	0.312	0.001	0.178	0.29	0.252
(ii)	0.469	0.339	0.316	0.008	0.241	0.363	**0.29**

TABLE IV: Comparison of different network sizes: number of learnable parameters – (i) 8.5×10^6, (ii) 20×10^6.

D. U-Net

Deeper vs. Wider network ablation study. Using the overall structure in Fig. 7, batch size of 4, binary cross-entropy loss and Adam optimizer with a learning rate of 10^{-4} halved at each 10 epochs, 3 different architectures were trained to see the effect of having a *wider* or *deeper* network and their results were compared and the best validation set outputs using binary cross-entropy loss were compared. Uniform weighting for each label is used as indicated in Table VI. These architectures included, (i) 4, (ii) 6 and (iii) 7 downsampling layers with first convolutional layer including 64, 32, 16 output channels respectively. At each downsampling layer, number of channels were doubled for each setting. Respectively, networks have approximately (i) 8×10^6, (ii) 31×10^6, (iii) 31×10^6 trainable parameters. Networks were trained for (i) 11, (ii) 16, and (iii) 13 epochs at which they provided the best validation BCEloss. Results can be seen in Table V. Thresholds for segmentations were picked for each channel using grid search. It can be observed that, a deeper architecture may work better compared to shallow counterpart, keeping the total amount of parameters fixed. Also, first setting with the decreased complexity provides suboptimal results compared to the latter two, and thus suggests that the latter two models are not over-complex and do not overfit the training data.

(i)	(ii)	W.C.	W.	S.W.	P.S.	D.P.	C.S.	mIoU
4	64	0.433	0.208	0.363	0.030	0.238	0.203	0.246
6	32	0.469	0.307	0.231	0.002	0.322	0.344	0.279
7	16	0.477	0.345	0.266	0.002	0.340	0.459	0.315

TABLE V: U-Net, IoU values for each architectural setting: (i) Total amount of downsampling layers, (ii) Number of channels after the first convolutional layer.

Modification of binary cross-entropy loss. As indicated in the previous parts, the data is significantly imbalanced in terms of the labels that pixels are assigned. This motivated using a weighting scheme on the loss function such that a larger weight will be assigned to less frequent labels. The weighted loss function L used for this purpose can be described as

$$L = \sum_{n=1}^{N} -w_n[y_n \ln \sigma(x_n) + (1-y_n)\ln(1-\sigma(x_n))]$$

where $N = 6$, w_n, σ, x_n and y_n are the number of labels, weight for the n-th label, sigmoid function, output of the network for n-th label and true label for n-th label respectively. The weight matrix $w = \{w_1, \ldots, w_N\}$ was selected inversely proportional to the ratios of different labels and changes in individual IoU values were examined. Three different weights

settings, $w = [w_1, w_2, w_3, w_4, w_5, w_6]$'s, were utilized in the experiments: (i) uniform, (ii) inversely proportional, where each label was assigned with a weight inversely proportional to its frequency in the data, (iii) modified inversely proportional, where the weighing scheme is moderately inversely proportional to the frequency of the labels in the data. Frequencies were computed as the total number of the pixels where respective label is 1 over the total number of labels where they are 1 in all pixels. These ratios are indicated in Section II-C. Accordingly, (ii) has the weighting scheme of $w = [1/0.701, 1/0.058, 1/0.071, 1/0.007, 1/0.030, 1/0.133]$ and (iii) has $w = [1, 4, 4, 16, 4, 2]$. Training of these networks were performed as a fine tuning on top of the best performing pre-trained 7 layer U-Net structure as indicated in Table V using again the Adam optimizer and a lower learning rate of 10^{-5}. The goal was to observe improvement especially on the "planter skip" class labels, however, weighting did not provide the expected change in the eventual results. Results are provided in Table VI. There is approximately 20 times increase in the mIoU value of planter skip class, however, result is still pretty far from being considered as reasonable. This is due to having almost no examples in the training data.

w	Weed	W.way	S.W.	P.S.	D.P.	C.S.	mIoU
i	0.477	0.345	0.266	0.001	0.340	0.459	**0.315**
ii	0.391	0.306	0.254	**0.018**	0.266	0.332	0.262
iii	0.477	0.304	0.281	0.009	0.268	0.407	0.293

TABLE VI: U-Net, IoU values for each loss weighting scheme: (i) Uniform weighting, (ii) inversely proportional weighting, (iii) moderate inversely proportional weighting as explained in Section IV-D.

E. FPN

For this set-up, we have used Adam optimizer with exponentially decaying learning rate with initial learning rate being 10^{-3}. Training was done for 15 epochs, with weight decay (factor of 10^{-5}) as regularization using the BCE loss given by Objective (1).

ResNet-18 and ResNet-50 as encoder: FPN uses a feature extractor (also referred as encoder) in its bottom-up pathway. ResNet and its variants have been popularly used for this task. Goal of this experiment is to compare two variants of ResNet, ResNet-18 and ResNet-50 as feature extractor for segmentation task, keeping other network structure, i.e., top-down and lateral connections the same. No pre-training has been done for either framework. Results in the Table VII show that a deeper encoder architecture, such as ResNet-50 performs much better than a shallow one, ResNet-18. It indicates that for images in the dataset, features learnt by ResNet-50 are better and more expressive than those learnt by ResNet-18. Hence even more expressive encoder such as ResNet-101 or ResNet-152 might result in further improvement.

With and without pre-training: Transfer learning is an important concept and quite useful in many deep learning applications. Using a pre-trained model (trained on a task) and fine tune for a somewhat related task helps in faster and better training. In this, we compared FPN with ResNet-50 as encoder across two training strategies: *(i)* ResNet-50 part initialized with the pre-trained network on ImageNet for classification, and *(ii)* end-to-end training using only for the given dataset and task i.e. non pre-training. Table VIII shows the comparative result for the two set-ups. Results clearly indicate that pre-training helps in learning good feature representation in the bottom-up pathway (encoder part) for the FPN, which helps in improving the segmentation performance.

Architecture	mIoU
FPN (ResNet-50)	**0.301**
FPN (ResNet-18)	0.258

TABLE VII: FPN: comparison between two different backbone architecture, ResNet-50 and ResNet-18, both without pre-training.

Architecture	mIoU
FPN (No pre-training)	0.301
FPN (Pre-trained ResNet-50)	**0.331**

TABLE VIII: FPN: comparison between two training schemes, *with* or *without* pre-trained ResNet-50 model as backbone.

V. Discussion and Conclusions

In this project, we addressed a very challenging and practical application of segmentation of aerial agricultural images. Individual discussions for various methods and experiments were included in their respective sections. Overall, we studied and analyzed different aspects related to deep learning based segmentation; performed comparisons between several successful deep neural network architectures and shallow and wider network, investigated the effect of the training loss by using weighted loss to tackle imbalanced classes, and the improvement due to pre-training.

Appendix

Name	Tasks
Ankit	Implement and experiments: FPN, transfer learning, network architecture ablation
Berk	Implement and experiments: U-Net, loss function ablation, network architecture ablation
Ufuk	Implement and experiments: DeeplabV3/+, network architecture and method ablations
Yuqi	Implement and experiments: DRN, loss function ablation, data visualization and post-processing

TABLE IX: Statement of individual contribution

References

[1] D. B. Larson, M. C. Chen, M. P. Lungren, S. S. Halabi, N. V. Stence, and C. P. Langlotz, "Performance of a deep-learning neural network model in assessing skeletal maturity on pediatric hand radiographs," *Radiology*, vol. 287, no. 1, pp. 313–322, 2018. 1

[2] A. Aniyan and K. Thorat, "Classifying radio galaxies with the convolutional neural network," *The Astrophysical Journal Supplement Series*, vol. 230, no. 2, p. 20, 2017. 1

[3] M. T. Chiu, X. Xu, Y. Wei, Z. Huang, A. Schwing, R. Brunner, H. Khachatrian, H. Karapetyan, I. Dozier, G. Rose *et al.*, "Agriculture-vision: A large aerial image database for agricultural pattern analysis," *arXiv preprint arXiv:2001.01306*, 2020. 1

[4] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125. 2

[5] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 472–480. 2

[6] J.-Y. Park, Y. Hwang, D. Lee, and J.-H. Kim, "Marsnet: Multi-label classification network for images of various sizes," *IEEE Access*, vol. 8, pp. 21 832–21 846, 2020. 3

[7] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017. 3

[8] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818. 3

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241. 3

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440. 3