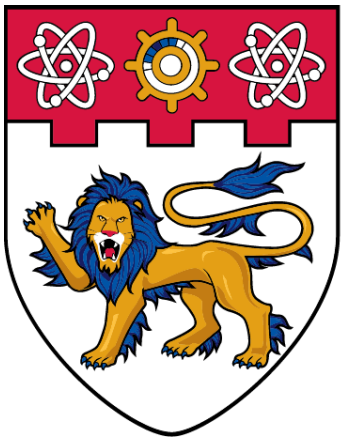


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4041 Machine Learning

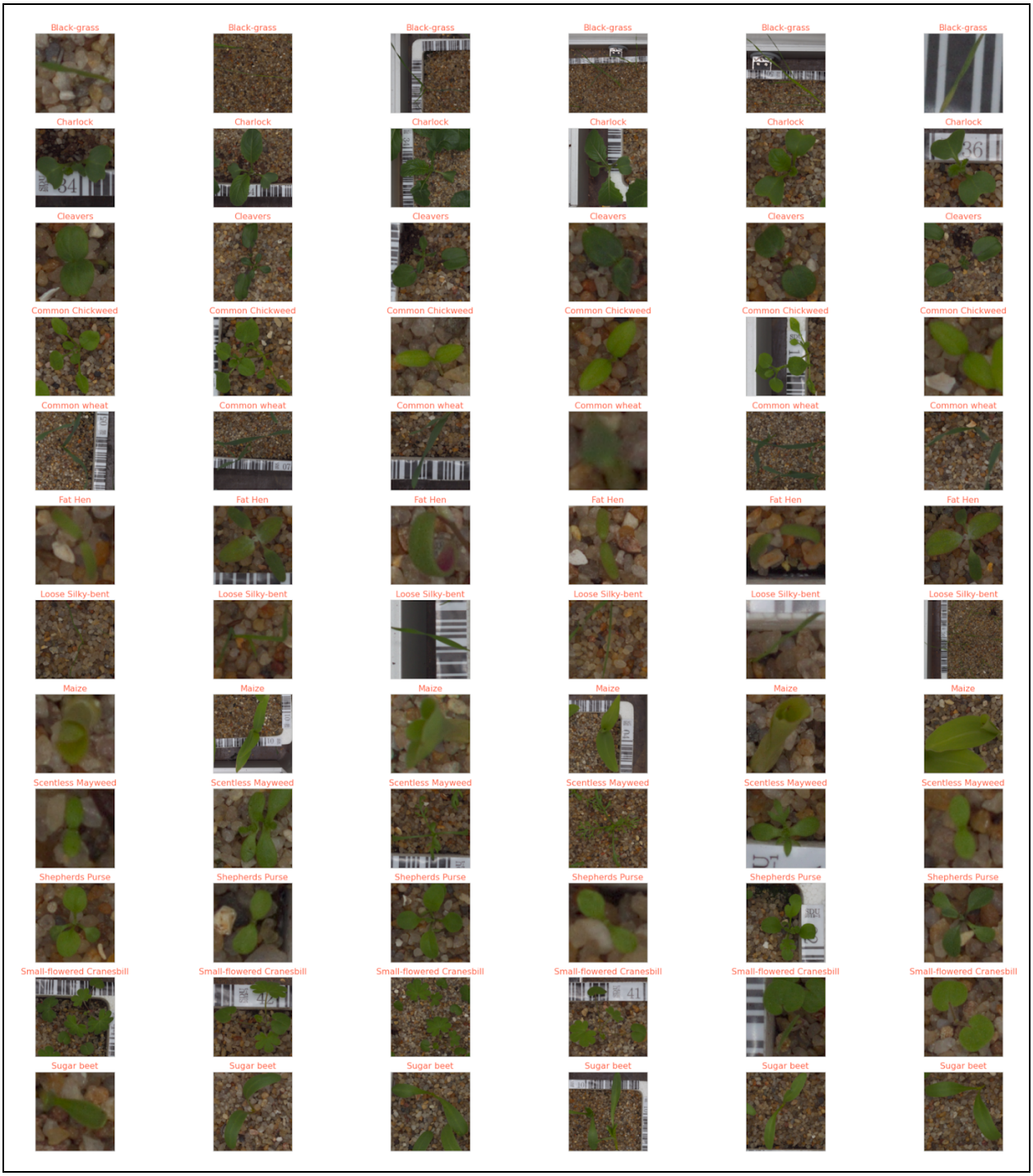
Group 25 Project Report

Name	Matric Number
Nicholas Goh	U1920857G

School of Computer Science and Engineering
AY2122 Semester 2

Problem Description

Classification of 12 discrete classes of plants with samples from the training data below. The data consists of 4750 images with an unequal split of images from each class.



Sample Train Dataset

Challenges

The following is a list of challenges, which are addressed in the sections to come.

1. Little data
2. Background is useless
3. Class imbalance

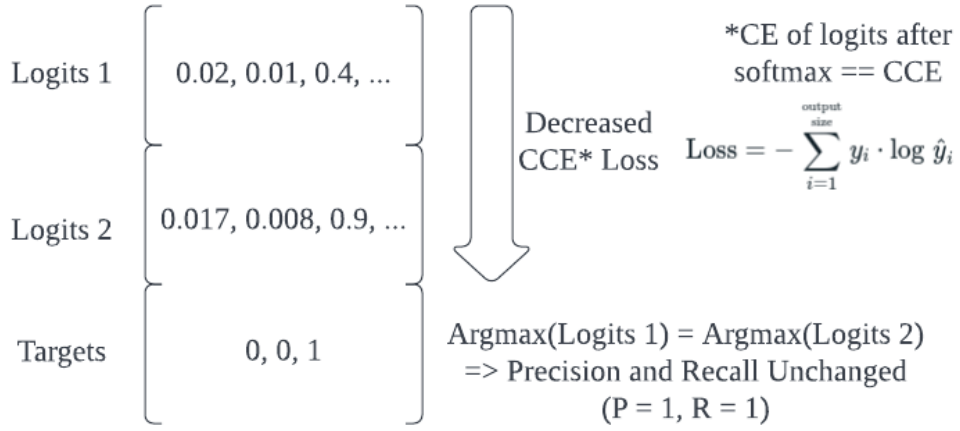
Kaggle Grading Criteria

Although the grading criteria is F1 Score, I compare model (categorical cross entropy) loss in my experiments as loss is more sensitive (informs on the model performance) better than F1 Score. At some value of loss, further reducing loss (increasing performance) may not result in change in F1 Score. This is illustrated in the diagram, where the predicted class (argmax) remains unchanged, while the loss decreases. Precision, recall and thus F1 score remain stagnant. It should be noted that F1 Score is generally inversely proportional to loss, so comparing loss still satisfies Kaggle's grading criteria.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

```
micro: True positivies, false positives and
    false negatives are computed globally.
```

tensorflow documentation



(Possibly) Unchanged predictions with improving model

source for loss formula

Solution Considerations

Architecture

I opted to keep the model small and lightweight as my laptop does not have the capacity for larger models. It follows that I chose MobileNetV2. MobileNetV2 uses pointwise and depthwise convolutions to reduce parameters and thus computation and training time. Furthermore, MobileNetV2 connects bottleneck layers with residual connections to promote transfer of information from these layers containing more salient information.

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	0.790	0.945	22,910,480	126	109.42	8.06
VGG16	528	0.713	0.901	138,357,544	23	69.50	4.16
VGG19	549	0.713	0.900	143,667,240	26	84.75	4.38
ResNet50	98	0.749	0.921	25,636,712	-	58.20	4.55
ResNet101	171	0.764	0.928	44,707,176	-	89.59	5.19
ResNet152	232	0.766	0.931	60,419,944	-	127.43	6.54
ResNet50V2	98	0.760	0.930	25,613,800	-	45.63	4.42
ResNet101V2	171	0.772	0.938	44,675,560	-	72.73	5.43
ResNet152V2	232	0.780	0.942	60,380,648	-	107.50	6.64
InceptionV3	92	0.779	0.937	23,851,784	159	42.25	6.86
InceptionResNetV2	215	0.803	0.953	55,873,736	572	130.19	10.02
MobileNet	16	0.704	0.895	4,253,864	88	22.60	3.44
MobileNetV2	14	0.713	0.901	3,538,984	88	25.90	3.83
DenseNet121	33	0.750	0.923	8,062,504	121	77.14	5.38
DenseNet169	57	0.762	0.932	14,307,880	169	96.40	6.28
DenseNet201	80	0.773	0.936	20,242,984	201	127.24	6.67
NASNetMobile	23	0.744	0.919	5,326,716	-	27.04	6.70
NASNetLarge	343	0.825	0.960	88,949,818	-	344.51	19.96
EfficientNetB0	29	-	-	5,330,571	-	46.00	4.91
EfficientNetB1	31	-	-	7,856,239	-	60.20	5.55
EfficientNetB2	36	-	-	9,177,569	-	80.79	6.50
EfficientNetB3	48	-	-	12,320,535	-	139.97	8.77
EfficientNetB4	75	-	-	19,466,823	-	308.33	15.12
EfficientNetB5	118	-	-	30,562,527	-	579.18	25.29
EfficientNetB6	166	-	-	43,265,143	-	958.12	40.45
EfficientNetB7	256	-	-	66,658,687	-	1578.90	61.62

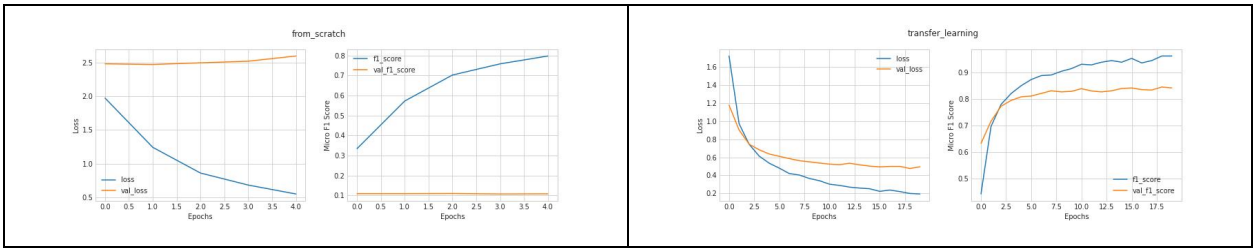
keras applications

Random Weight Initialization

Challenge 1. Theoretically, 4750 images is too little data to train a model from scratch (random weight initialization). I prove this empirically in the following Figure. The model easily memorized the small dataset (decreasing loss) and could not generalize to the validation data (stagnant valid loss). This could be mitigated by training only the first few layers (closest to input) at a very low learning rate before allowing all layers to be trained at a higher learning rate. However, this is time consuming and I chose to do the experiment in the Figure with the same hyperparameters as in Transfer Learning.

Transfer Learning

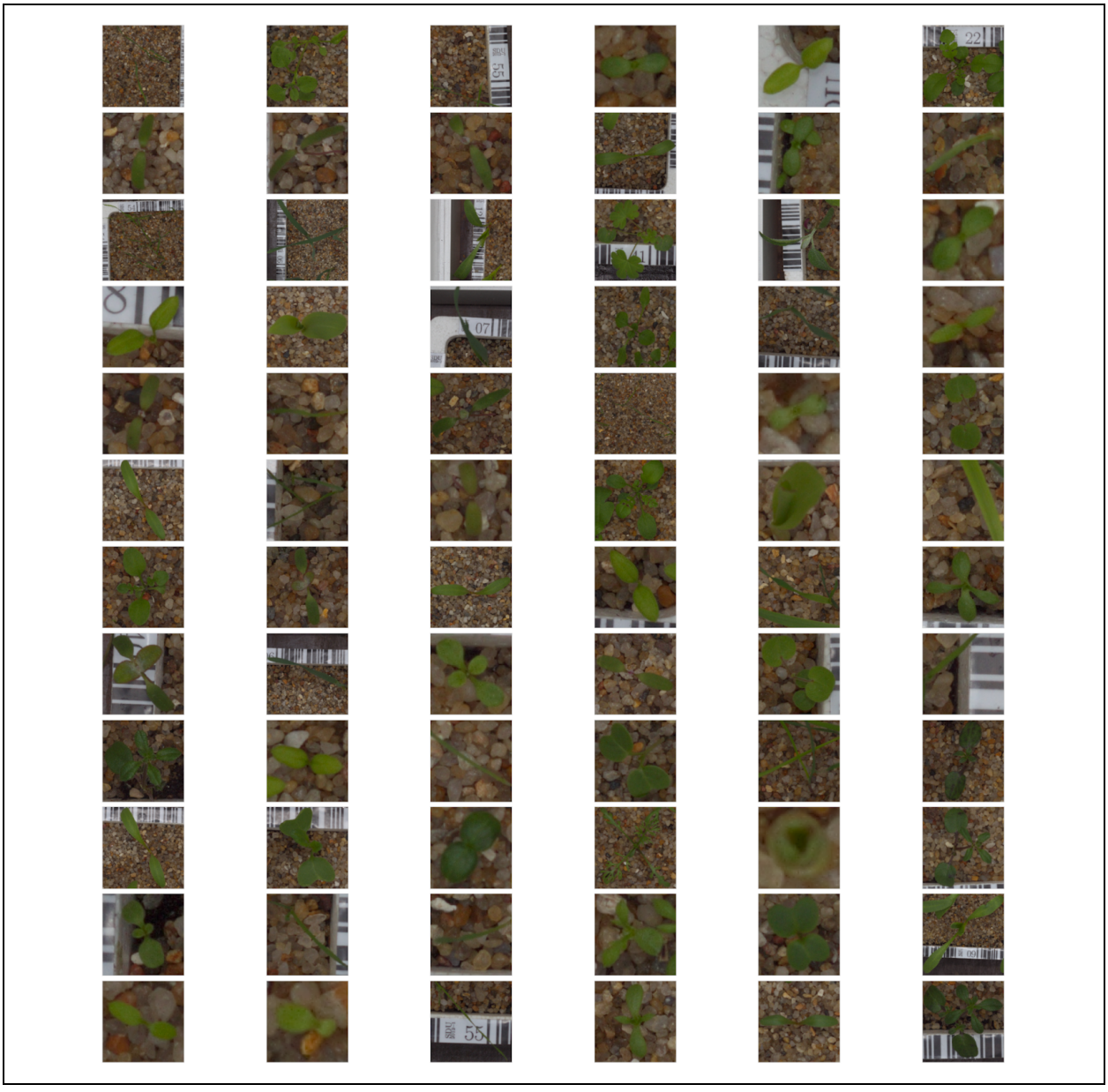
I leverage on pretrained weights learned from training MobileNetV2 on ImageNet. ImageNet consists of 100,000 images with 1,000 classes. Although none of these classes are part of the 12 classes in this problem, the learned weights have the ability to provide useful information to any computer vision problem including this problem. I freeze all but the classification layer in MobileNetV2, as theoretically, the learned weights are able to determine low to middle level features such as edges and shapes. I replace the 1000 unit classifier with a 12 unit classifier and train this classifier to suit my task. Comparing the graphs below, it is clear that transfer learning is superior to training from scratch as .



Random Weight Initialization vs Transfer Learning

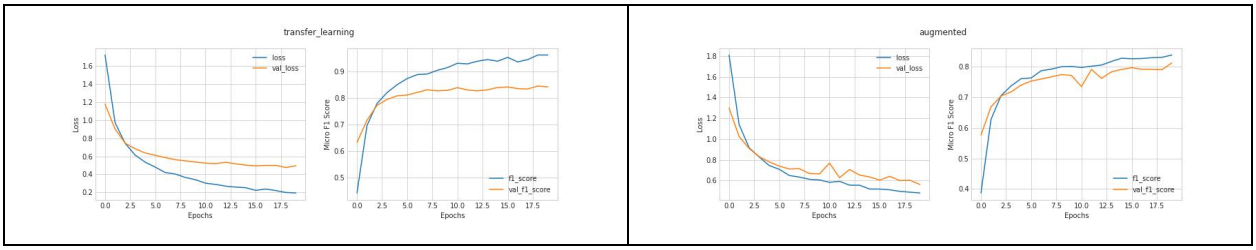
Data Augmentation

To combat the overfitting described above, I perform data augmentation to artificially increase my dataset. I perform random vertical, horizontal flips, coupled with random rotations, to increase the volume of data, theoretically forcing the model to generalize better. Samples are as seen below.



Samples Augmented Images

Augmentation seems to increase overall loss (decreased performance) but decrease divergence between train and valid loss (reduced overfitting). In this case, augmentation is still preferred as there is significant reduction in overfitting for the cost of minor performance decrease.



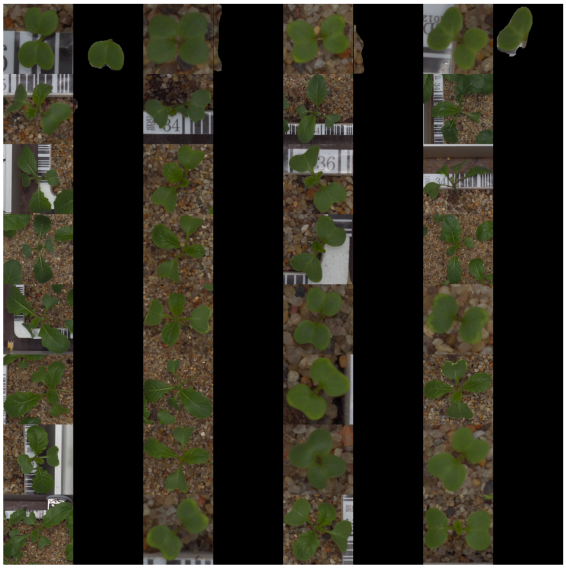
Non Augmented vs Augmented

Noise Reduction

Challenge 2. The salient information in each image is the plant, meaning the background pebbles (and black white scale for reference) are unnecessary noise especially since the pebbles are in a different configuration in each image. I tried 3 different ways to extract these salient regions from each image and discard the rest, since a human expert is able to classify each plant without the noisy backgrounds.

Instance Segmentation

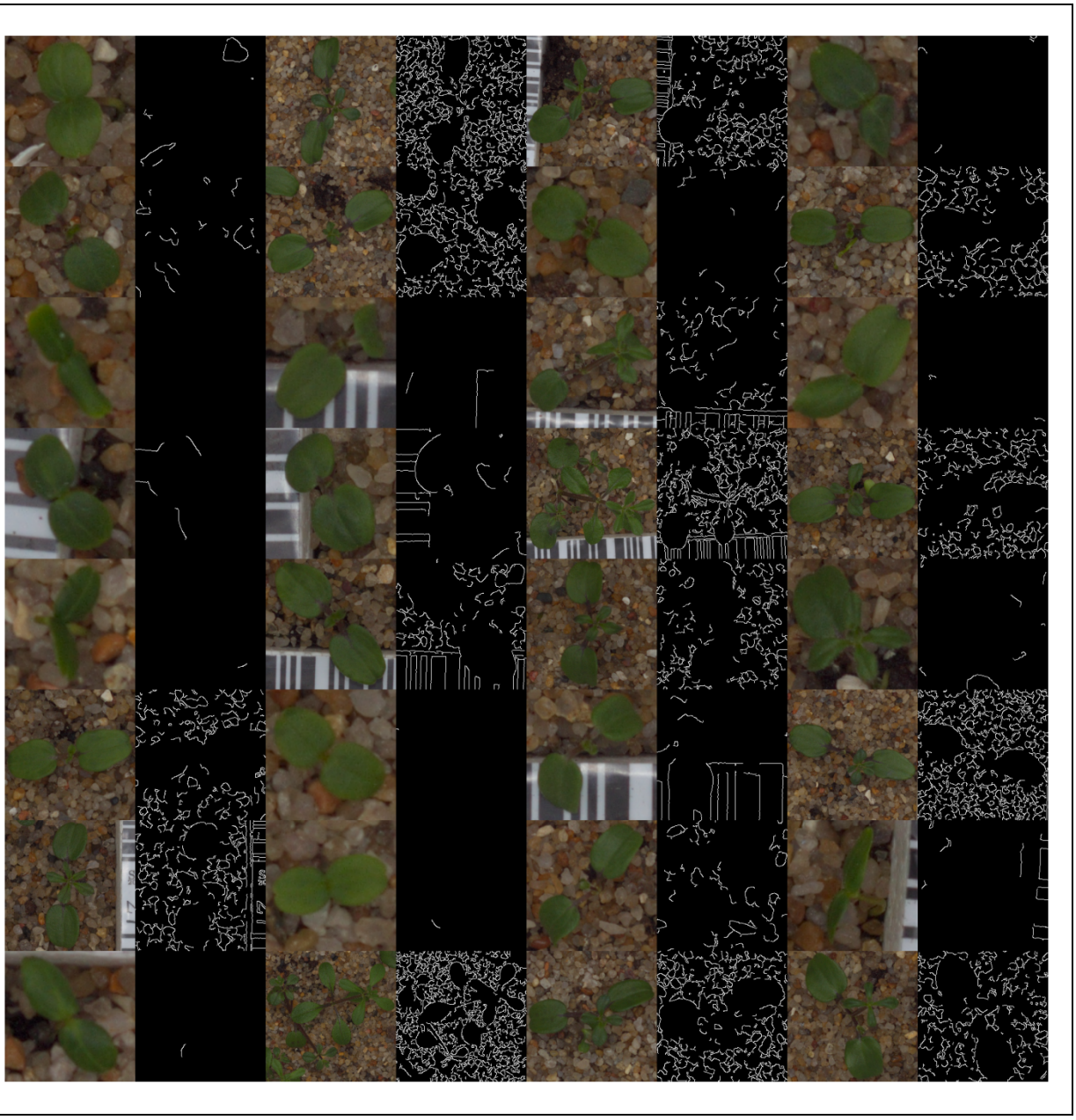
I hoped to leverage on Detectron2 to perform clean Instance Segmentation to reduce noise. I selected weights pretrained on CityScapes dataset as it contains the class vegetation, the closest class to my problem in all of Detectron2 pretrained weights. Using these weights, I keep each segmented instance and discard the rest of the image. As seen below, this barely works on images, and only partially segments the plant when it seems to work. I do not include code to reproduce this as I do not use this method for my experiments.



Instance Segmentation with Detectron2

Canny Edge Detection

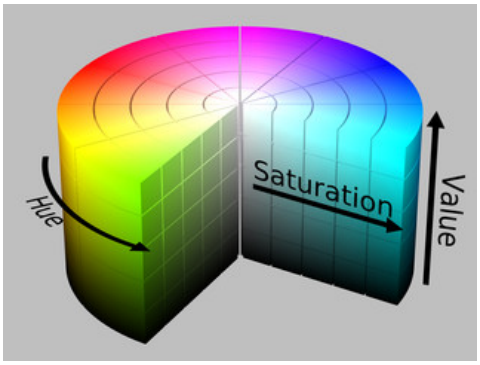
As seen below, Canny Edge Detection is not reliable in outlining plants in each image. I tried different handpicked thresholds but still could not find one that separates the plants from the background. I do not include code to reproduce this as I do not use this method for my experiments.

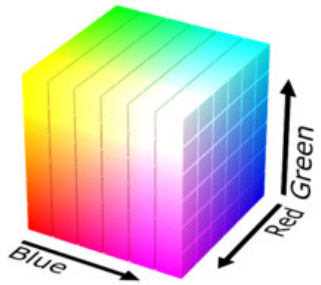


Canny Edge Detection with OpenCV

Hue, Saturation, Value (HSV) Masking

HSV allows for more control over the colors as compared to Red, Green, Blue (RGB). As seen below, it is harder to select a rectangular green region in RGB with 3 axes than selecting a green pie region with 3 axes.

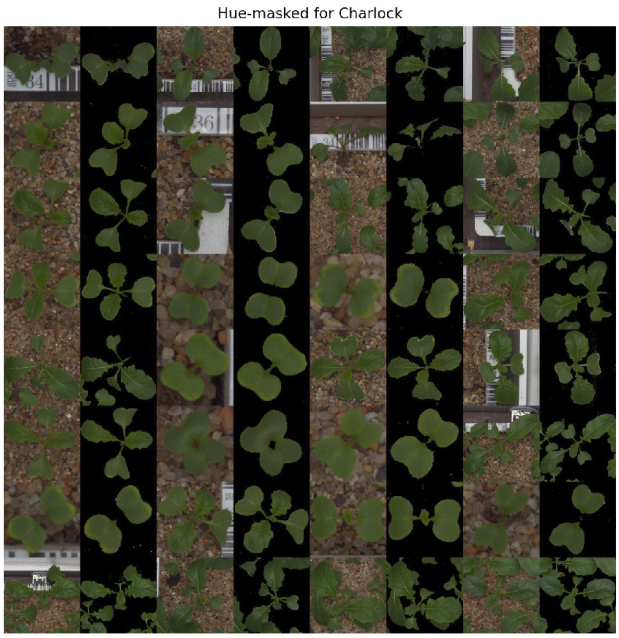




opencv docs

I convert input images into HSV format and keep pixels that fall within a handpicked range denoting green color.

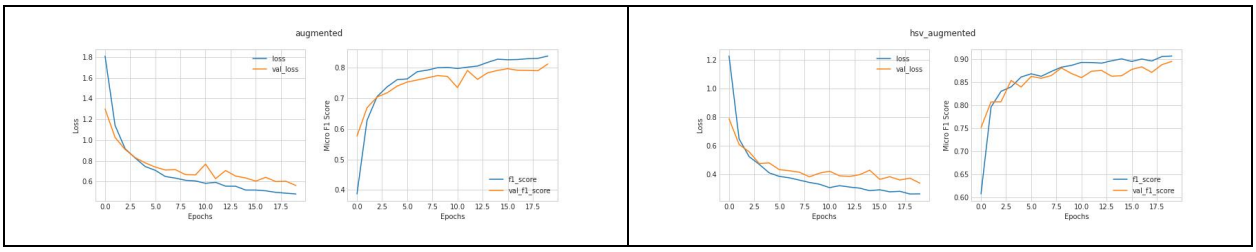
Handpicked HSV range



Sample HSV masked images

Experiments

I hypothesized that removing noise would decrease overall loss and I have proven this empirically. Applying HSV mask resulted in lower loss and similar divergence. The model increased in performance and maintained its generalizability.



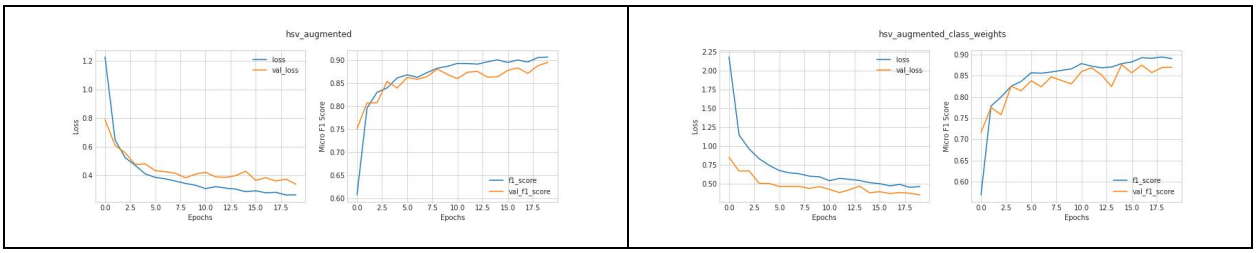
Augmented vs HSV Augmented

Class Weight

Challenge 3. I apply class weight to HSV augmented to account for class imbalance. As seen in the figure below, class weight appears to perform worse than HSV augmented. Applying class weights results in higher loss (poorer performance) as compared to without class weights and just HSV augmented.

Class Label	Class Weight	Total Instances
'Black-grass'	2.4756756756756757	185
'Charlock'	1.6776556776556777	273
'Cleavers'	2.2786069651741294	201
'Common Chickweed'	1.0700934579439252	428
'Common wheat'	2.9548387096774196	155
'Fat Hen'	1.3753753753753755	333
'Loose Silky-bent'	1.0	458
'Maize'	2.9548387096774196	155
'Scentless Mayweed'	1.2651933701657458	362
'Shepherds Purse'	2.8271604938271606	162
'Small-flowered Cranesbill'	1.3160919540229885	348
'Sugar beet'	1.6962962962962962	270

Class Weight

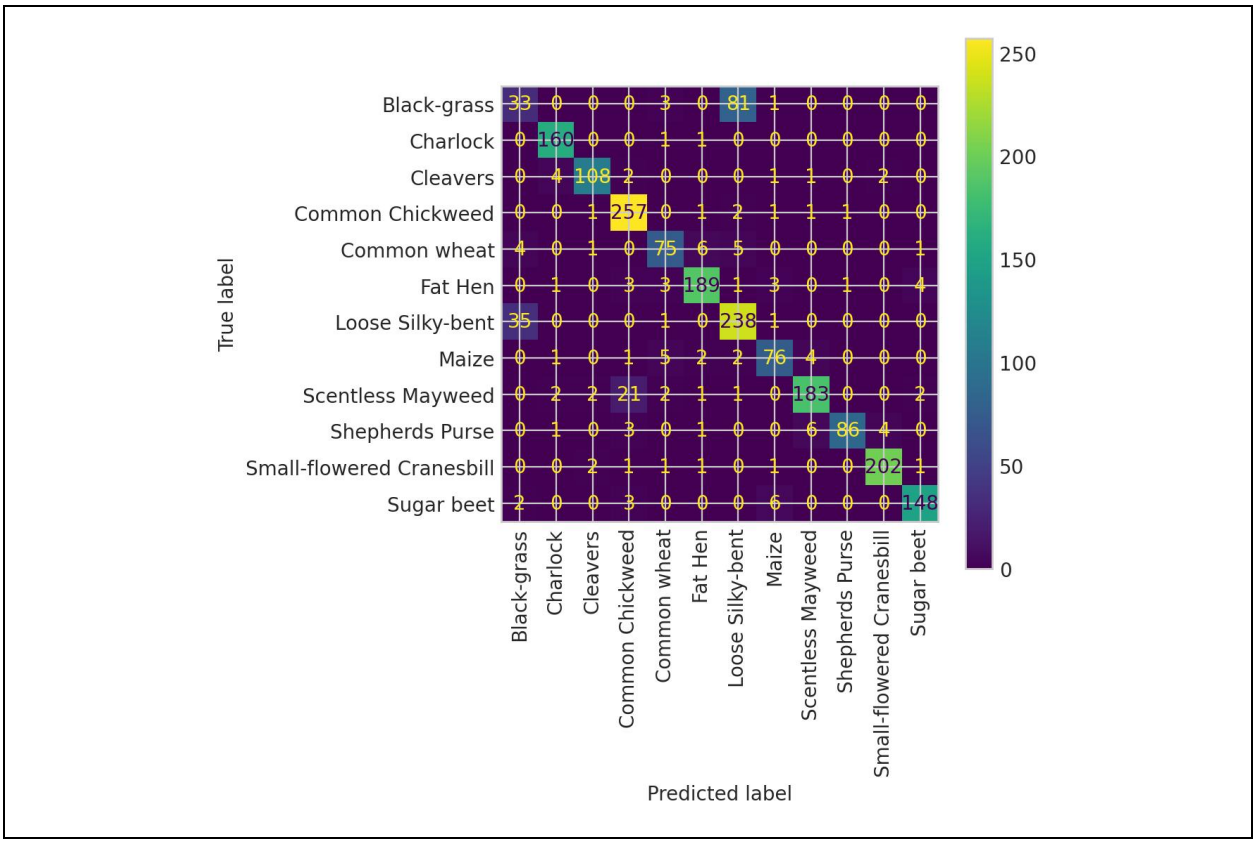


HSV Augmented vs Class Weight

Explainability

Confusion Matrix

I plot a confusion matrix for the validation data. As seen, most of the error is between Black-grass and Loose Silky-bent. Looking at the following diagram, these two leaves look very similar. This supports the results in the confusion matrix; perhaps even experts find difficulty in distinguishing between these two leaves ***from visuals alone***.



Confusion Matrix

Black-grass vs Loose Silky-bent

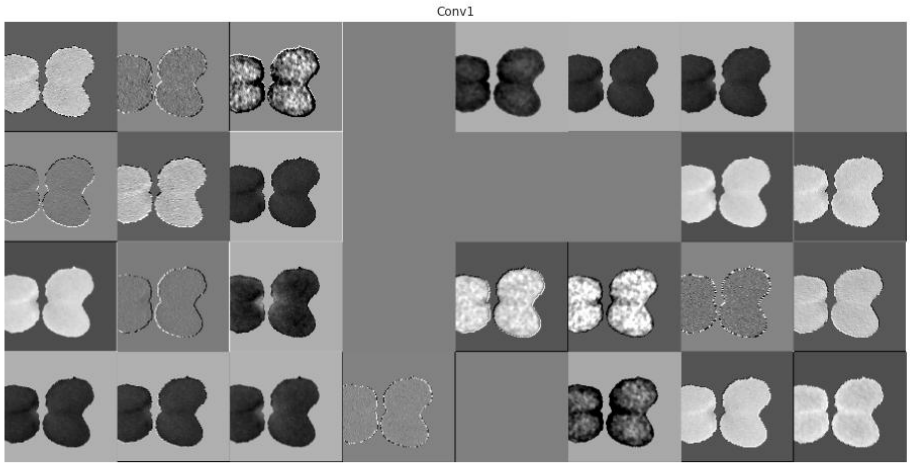
Feature Map Visualization

Using the following image as input to my model, I collect and display the feature maps (outputs from each layer) of the first 3 convolutional layers.

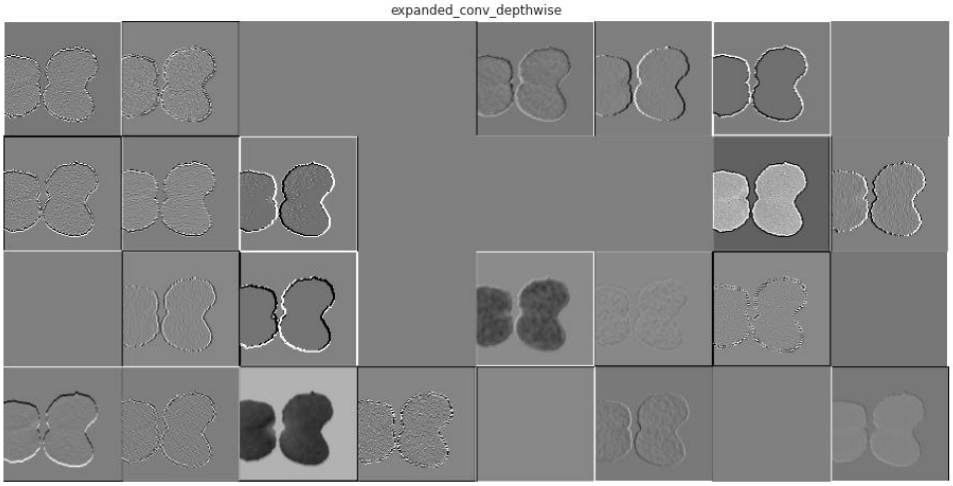
Input Image

Feature maps from Conv1 appear to distinguish between the charlock leaf and the removed background (hue-masked). Many of the 32 feature maps are activated (white) where the charlock leaf is in the picture. This means the weights of the filters in this layer form good chalock leaf detectors.

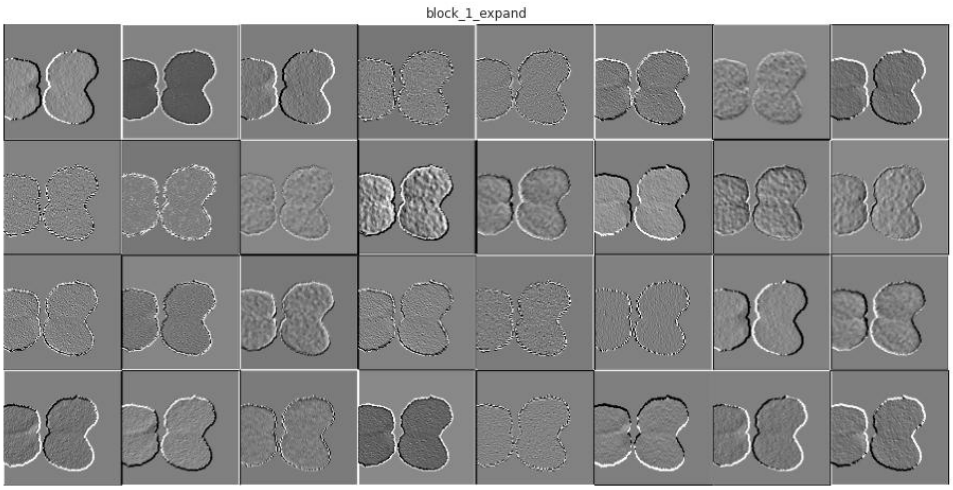
Feature maps from the following 2 layers also activate for chalock leaf regions, though there is more activation for the edges of the leaves than within the leaves.



Feature Maps from Conv1



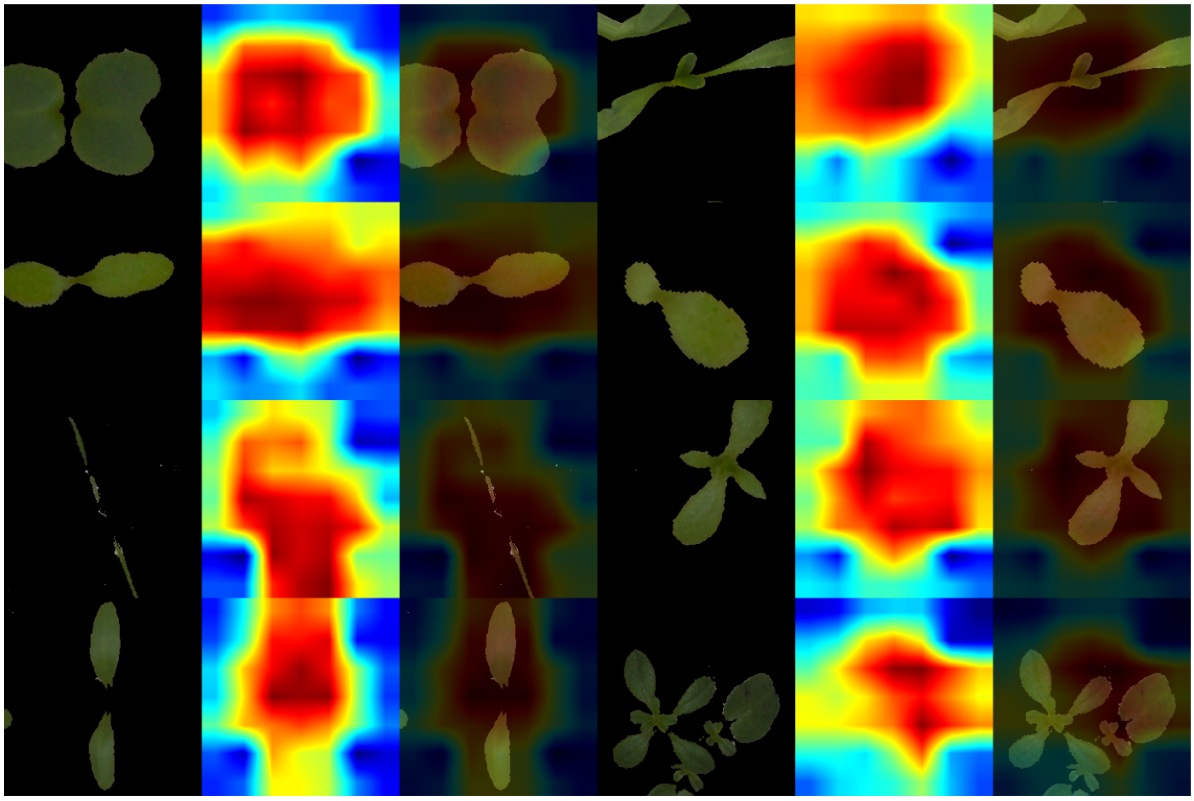
Feature Maps from expanded_conv_depthwise



Feature Maps from block_1_expand

GradCAM

I perform GradCAM to see which regions of an input image affect the class prediction the most. Red regions denote pixels that contribute most to the class prediction while blue regions denote the least contribution. The diagram below shows images in triplets, (hue-mask image, heatmap, previous 2 images overlaid). As seen below, red regions have large overlap with the leaves. This means the model is able to distinguish between important information (the leaves) and useless information (the background).



Kaggle LeaderBoards

595	Alex Zatsman	0.88539	5
596	Nicgoh	0.88413	2
597	mwilson	0.88287	2

Takeaways

From this assignment, I learnt to try innovative ways to preprocess the data, namely instance segmentation, canny edge detection and HSV masking. Furthermore, using class weights does not guarantee improvement in overall loss. Finally, visualizing the model with feature maps and GradCAM also provides additional information on the robustness of the model.

Further Work

The model still has room for improvement, as some of the feature maps are not correctly activated and GramCAM regions do not highlight the leaves fully (though red regions overlap with around 70% of each leaf). Perhaps something can be designed to allow the model to distinguish between Black-grass and Loose Silky-bent to a higher degree. Perhaps using a larger model with a computer with more compute capability will improve results as well.