Approach for fine-tuning a model on the DeepWeeds Dataset

Parth Agrawal

January 2026

1 Introduction to the Dataset

The DeepWeeds dataset comprises 256x256 RGB colour images of various weeds found in Australia, encompassing around eight different species and one negative class. The challenge was to train a model despite the high background noise and similar colours and shapes of weeds.

2 Model Selection and Other Parameters

2.1 Model Selection

I initially used ResNet50 and attempted to partially fine-tune it, but to no avail. It also posed a problem to do this with the T-400 GPU on Colab. Since the images were pretty large and three-channel, I deemed any large model impractical soon. I initially tried using EfficientNetB0, but ultimately settled on MobileNetV3Large as the best option.

2.2 Feature Extraction over Partial Fine Tuning

Even after using a small model like MobileNetV3Large, I still found it challenging to employ partial fine-tuning due to the large image size and constraints on Colab. Thus, I settled with Feature Extraction only. I use a simple ReLU function for activation and a softmax for the final layer.

2.3 Squeeze-and-Excitation (SE) blocks

I initially stumbled on this technique while working with ResNet50 through an example on Kaggle. I learned that it had potential to improve since the images of weeds carried a lot of noise around them. Although I later switched to ResNet50, I left it in place since I hoped it would help achieve better accuracy.

2.4 Class Weights

The models struggled with training due to the negative occupying around 50% of total training data, making it overwhelmingly favour it rather than truly classifying weeds. To solve it, I found that implementing class weights (inspired by ChatGPT) significantly improved the result and addressed the imbalance problem.

2.5 Early Stopping

I do not need to mention this, but often, while training the model, it simply plateaued or overfitted due to imbalanced data, so I used early stopping as a helpful measure.

3 Results

The model demonstrated good performance, achieving an accuracy of 75.8% and a macro F1 score of 0.73. Although some confusion remained, it performed decently.