Fine Grained Categorization of Plants

Haozhe Chen
frankchz@bu.edu

1. Introduction

The goal of this project is to develop AI models to solve real world computer vision problems. There are two tasks included. The first one is a supervised learning task, which requires us to build a classifier based on 280,000 plant images to classify 990 categories of these plants. The evaluation metric is the top 5 predictive accuracy. For a given image, if the ground truth label is found among the 5 predicted labels, then the error for that image is 0, otherwise it is 1. The final score is the error averaged across all images. The second task is a semi-supervised challenge; we have 700 labeled images along with 4200 unlabeled images, our task is to utilize these data to predict the ground truth label of 20 plant categories.

2. Supervised Learning

2.1 Model Selection

In the supervised learning problem, I built and fine-tuned my models based on ResNet50 and EfficientNeB0, and compared to the baseline model. Both Resnet and EfficientNet have relatively small amounts of parameters but a higher accuracy in image classification problems, which means they are more efficient compared to existing CNNs. Deep neural networks suffer from degradation, which means, as they go deeper, accuracy gets saturated and then degrades rapidly. Instead of hoping every few stacked layers directly fit a desired underlying mapping, Resnets let these layers fit a residual mapping, as figure 1 shows below. The skip-connection structure prevents the network from overfitting, and gradients exploding (vanishing).

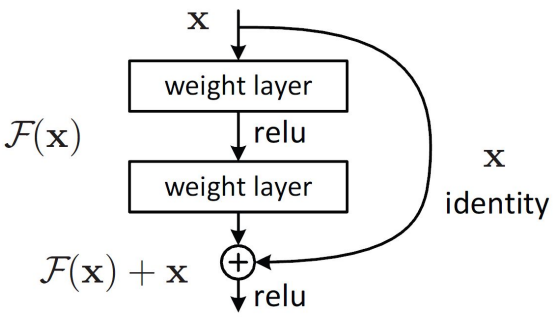

figure 1

EfficientNets address accuracy issues by scaling. ResNet is an example being scaled. The original network scaled by width, depth, resolution with certain parameters. Thus, it doesn't introduce computational costs, but improves accuracy.

2.2 Transfer Learning

In my models, I tried both pooling and flatten layers on the top of the pre-trained models, followed by a 1024 fully connected layer and then the output layer. In both models, I froze the layers except the last 2-3 layers. It turns out, the Resnet50 has better results by adding a flatten layer, EfficientNetB0 does better with a pooling layer. However, both of them have overfitting problems at the

beginning. In order to prove validation accuracy, many tricks were applied, such as augmentation, dropout, and regularization.

2.2 Fine Tuning

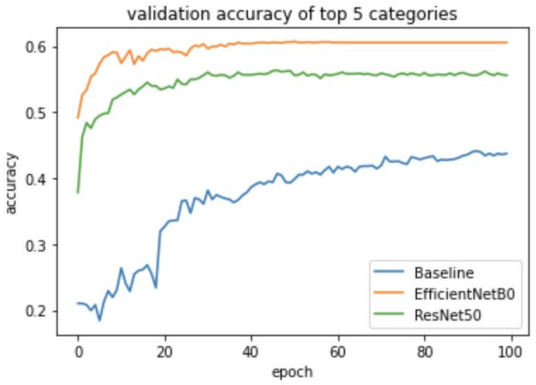

figure 2

Figure 2 shows the result of model comparison for 100 epoch after image augmentation and dropout. In order to finalize the mode, I tested their performances in the colab, with a bunch of 32x32 images (the original images are 224x224). EfficientNetB0 has the highest validation accuracy, ResNet50 is also not bad, compared to the baseline model. However, after epoch 50, the validation accuracy barely improves. After examining the model, I figured that both training and validation accuracy stagnate at that point, so I tried to fine tune the model without image augmentations and delete the dropout layers, this doesn't change the structure of the model, so that I can use the pre-trained parameters.

Figure 3 is the result of fine tuning of EfficientNetB0, we could see that after 50 epochs of training, the model stagnates if you don't change them, as the blue line shown. When I remove the regularizations, there is a huge improvement on validation accuracy.

The top 5 classes validation accuracy is lower still lower than 70%, since the images are

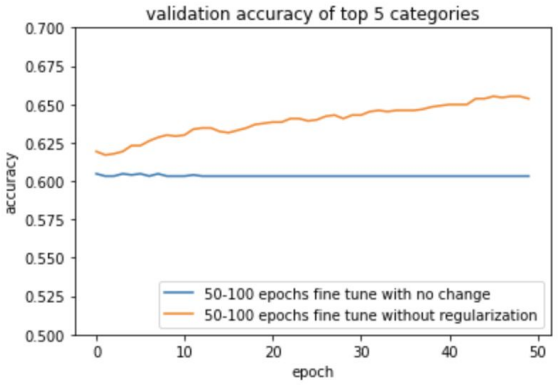

figure 3

compressed to 32x32, they convey less learnable features. However, the accuracy of the same model running on the original (224x224) image set is higher than 82%. Due to the limitation of the scc usage, I only ran the final model 20 epochs. It might yield better results as it is trained more.

3. Semi-Supervised Learning

3.1 Approaches

We got 700 labeled images and 4000 unlabeled data for the semi-supervised challenge. In this task, I kept the EfficientNetB0 model from supervised task, since it is more robust, I used the model to predict labels for unlabeled images, and then used the new labeled data to train the model. To be more specific, here is a generalization of my methodology.

- Split the labeled data into a training set (20%) and a validation set (80%).

- Compare different classifiers (Baseline, ResNet, EfficientNet) on validation accuracy. Choose the model with better validation accuracy for fine tuning.
- Train the model and set the improvement threshold to 0.01, if validation accuracy improves less than 0.01 compared to the previous epoch, start training unlabeled data.
- Predict a batch(1/10) unlabeled data, add the predicted labels and the images to the training set, re-train the model, until the improvement stagnates again, incorporating another batch.
- Train all unlabeled the data and then fine tune the model.
- Repeat this training process for several iterations until results.

3.2 Interpretation of Results

In this "updating training examples" approach, one iteration is a complete training process, in which all unlabeled were used to tune the model. As the model is trained more iterations, the validation accuracy improves. If we don't use unlabeled data, the validation accuracy reaches a certain plateau and stops improving. As figure 4 shows, there is a significant improvement of semi-supervised learning. The blue line represents the performance of a supervised learning model, which means using EfficientNet to train only labeled data for 100 epochs, the plot shows the accuracy of it's last 10 epochs. Other lines represent the iterations of semi-supervised learning. After the second iteration, the model started to

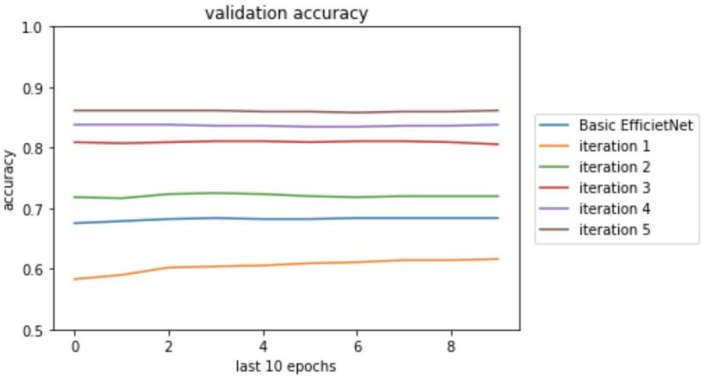

figure 4

outperform supervised learning; as more iterations the model has learned, the accuracy keeps improving; only about 5 iteration, the validation accuracy is approaching 90%.

4. Conclusion

As I've tried different methods, the EfficientNetB0 performs better in this task, after the performance stagnates, decreasing the level of regularization might be a good idea for chasing better results. In the semi-supervised task, training the unlabeled samples in our learning model makes the model more robust.

Link to Github repo

References

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Deep Residual Learning for Image Recognition

Keras: the Python deep learning API