**Transfer Learning for Plant Disease Classification with EfficientNet**

Stephanie Cunningham

December 2020

Abstract

Deep learning for image classification is a continually developing pursuit which has begun to be applied to the agricultural field. In this research, the New Plant Disease Dataset, consisting of 87,000 plant leaf images, is explored in an aim to classify the images by one of 38 categorizations, including health status and species type, is attempted through Convolutional Neural networks, Transfer Learning, and the combination of these. In particular, EfficientNet, a class of models first introduced in 2019 and known for their superior accuracy and efficiency, is successful in its application to this problem. Using CPU and EfficientNetB0 as a preliminary exploration into this possibility, an encouraging accuracy of 97.81% was achieved using categorical cross-entropy as the loss function.

1. Introduction

Often, biological and agricultural research involves the comparison and identification of features in sample specimens; this has traditionally involved work of experienced scientists, researchers, or farmers with keen eyes and deep subject matter knowledge. One such example is the identification (and, for the farmer, corrective action) of plant diseases. However, machine learning advancements have created the possibility to take the guesswork and horticultural expertise previously needed for this task, through automated image classification. The use of machine learning in agricultural applications is key to the mission of PlantVillage, a Pennsylvania State University organization which originally published a large image dataset, now commonly referred to as the “PlantVillage Dataset” but no longer available from the original source [https://plantvillage.psu.edu/]. Several augmented versions of the data have been made available for research and competitions. One such iteration is the “New Plant Diseases Dataset” hosted on Kaggle (Bhattarai, 2018).

Image classification is an important application of neural networks and many advancements have been made in recent research to help quickly build very accurate models. Transfer learning (TL), a method of combining a pre-trained model (usually trained on large general datasets such as Imagenet) with fine-tuning of a particular dataset, was first introduced in a 1976 Croation paper (Bozinovski and Fulgasi, 1976) The concept has much more recently exploded in scope, application, and accuracy, with a wide variety of off-the-shelf models available, and many are pre-loaded into the useful deep learning packages Tensorflow and Keras.

Convolutional neural networks (CNN) layers are another key tool of image recognition tasks, which are often utilized within the pre-trained TL models, and can also be placed in the fine-tuning stage. First introduced in the 1980s and since incorporated widely into deep learning applications (Dickson, 2020), CNNs are particularly useful for image classification because they create a “map” of features by filtering the original image, and are highly customizable. CNN layers can be stacked on top of one another, with shallow layers able to learn identification of low-level features such as lines, and deeper layers learning higher-level features like shapes or objects. Due to the sensitivity of the CNNs to specific feature location (for example, a mark indicating rot could appear in the top left corner of the image, or a mark indicating the same rot could appear in the center in varying images), it is common practice to use “down sampling,” usually through a pooling layer, to create a lower resolution version, essentially a robust “feature summary” (Brownlee, 2019). It should be noted that there are other methods to combat feature location sensitivity (which can be applied in the same model) such as augmenting the training images through rotating, stretching, flipping, and shifting. In this research, TL and CNNs will be examined for their usefulness and for their robustness to change in creation of an image classification model for plant disease prediction with the New Plant Disease Dataset. Several prior researchers have made use of this technique for classification of PlantVillage Data and other leaf disease detection sets, using existing models such as versions of AlexNet, ResNet, and VGG. In a 2020 survey of such methods, researchers found that AlexNet was by far the most-used (Abade et. al, 2020). Several TL models, VGG16, ResNet50, InceptionV3, InceptionResNet and DenseNet169 were compared in “On Using Transfer Learning For Plant Disease Detection.” (Sagar and Dheeba 2020) However, one of the newest advancements in TL models, Efficientnet, has not yet been applied to the PlantVillage dataset in published research and was not included in Abade et. al’s 2020 survey. This renders the Efficientnet scheme a good choice for novel research.

Introduced in 2019 by Google AI team Tan and Le, EfficientNet family of models provide state-of-the-art accuracy with high efficiency. There are a variety of models to choose from, ranging from “B0” to “B7,” all built on the convolutional neural network model and principle of compound scaling to improve efficiency and accuracy. EfficientnetB0 is the smallest and simplest in the class of EfficientNet Models. This base model is built with several layers including MBConv and inverted Res bottlenecks which the developers found to achieve superior accuracy to classic CNNs (Kizrak, 2020). From this model to the higher level models, compound scaling is used. This method involves a grid search for finding the best coefficients to use in scaling for each dimension width, depth, and resolution. This process is also constrained to a set resource load (Tan and Le, 2019). EfficientNet was built to reduce the number of trainable parameters while increasing accuracy, and ranked high in the 2019 ImageNet challenge with 84.4% accuracy using 66 million parameters with the B7 model (Kizrak, 2020). Better accuracy had already been achieved on the data in 2018, but only by 1% or less, at the cost of huge parameter loads (e.g. 829 million in ResNetXt-101 32x48d with 85.4% top-1 accuracy).

This efficiency is an excellent feature for many transfer learning applications, particularly image classification with CPU, as performed in this analysis. As the authors of the original paper noted, “to be most useful, [EfficientNet models] should also transfer to other datasets” (Tan and Le, 2019). Many transfer learning applications have achieved best-in-class accuracy levels, and it is an aim of this research to compare this method with similar research on the same data. Note the dataset used for this research is the “New Plant Diseases Dataset,” “recreated using offline augmentation from the original dataset.” (Bhuttari, 2018). There is similar research performed on different versions of the data, usually simply referred to as the “PlantVillage Dataset.”

The results of the models in this work are evaluated and compared with one another in terms of accuracy, loosely compared with those of Sagar and Deeba 2020, the most recent and comparable work (e.g. does not focus on a specific species or subset of the 38 categories) available at the time of this research, which reported to have achieved 98.2% accuracy. However, this is not taken as a directly comparable metric for the following reasons: the New Plant Disease Data is slightly larger and augmented differently than the PlantVillage Data used in the aforementioned work, and the authors used binary cross-entropy as the loss function, while this researcher is interested in categorical cross-entropy for the accuracy of classification of 38 separate categories.

1. Methods

The “New Plant Diseases” dataset was downloaded to a local drive through Jupyter notebook from the Kaggle API, and loaded into the notebook through the Tensorflow (TF) Image Data Generator. All images were scaled down upon loading by 1./255. This is common practice in image pre-processing which scales the pixel values for each color map to 0,1 (as the maximum pixel value is 255). This is much the same reasoning as re-scaling for non-image data, and provides many benefits including each image contributing equally to the total loss[linkedin]. The TF Generator is also equipped to (further) augment images, and 0.2 ranges were provided for the training set to be augmented by shear, zoom, width shift, and height shift. Validation images were scaled by the same factor but not augmented. The dataset consists of 87,000 RGB images belonging to 38 classes, which includes healthy or diseased status and the species of plant. The pre-assigned training set consists of 80% of the images, and reserved validation set is 20% which was created with the same structure to preserve the distribution, approximately equal among the classes, so there is not a class imbalance aspect of the research problem. These classes are shown in Table 1:

Table 1. Plant Image Classes

|  |  |  |
| --- | --- | --- |
| Species | Disease/Healthy | Class Index |
| Apple | Apple scab | 0 |
| Apple | Black Rot | 1 |
| Apple | Cedar apple rust | 2 |
| Apple | Healthy | 3 |
| Blueberry | Healthy | 4 |
| Cherry (including sour) | Powdery Mildew | 5 |
| Cherry (including sour) | Healthy | 6 |
| Corn (maize) | Cercospora leaf spot/gray leaf spot | 7 |
| Corn (maize) | Common rust | 8 |
| Corn (maize) | Northern leaf blight | 9 |
| Corn (maize) | Healthy | 10 |
| Grape | Black Rot | 11 |
| Grape | Esca (Black Measles) | 12 |
| Grape | Leaf Blight (Isariopsis Leaf Spot) | 13 |
| Grape | Healthy | 14 |
| Orange | Haunglongbing (Citrus Greening) | 15 |
| Peach | Bacterial Spot | 16 |
| Peach | Healthy | 17 |
| Pepper, bell | Bacterial Spot | 18 |
| Pepper, bell | Healthy | 19 |
| Potato | Early Blight | 20 |
| Potato | Late Blight | 21 |
| Potato | Healthy | 22 |
| Raspberry | Healthy | 23 |
| Soybean | Healthy | 24 |
| Squash | Powdery Mildew | 25 |
| Strawberry | Leaf Scorch | 26 |
| Strawberry | Healthy | 27 |
| Tomato | Bacterial Spot | 28 |
| Tomato | Early Blight | 29 |
| Tomato | Late Blight | 30 |
| Tomato | Leaf Mold | 31 |
| Tomato | Septoria Leaf Spot | 32 |
| Tomato | Spider Mites/Two-Spotted Spider Mite | 33 |
| Tomato | Target Spot | 34 |
| Tomato | Tomato Yellow Leaf Curl Virus | 35 |
| Tomato | Tomato Mosaic Virus | 36 |
| Tomato | Healthy | 37 |

Several model versions were tested, attempting to find the best accuracy and compare accuracy and processing time. Firstly, a model with CNN but no transfer learning model was tried. Several combinations of CNN structures and number of filters, optimizers (RMSprop, Adam, and SGD), Dropout p levels and locations, and activation functions (ReLU, swish, sigmoid) were pilot tested for accuracy on 2 epochs each in the non-TL model until the final structure was identified with 2 layers of CNNs (see Figure 1). The best versions of non-TL, ResNet50, and EfficientNet models identified in the training and testing process were each chosen according to the accuracy metric, and the model schema are shown in Figure 1.a-c.

ResNet50 was chosen for comparison to EfficientNet as the findings of Sagar and Dheeba 2020 ranked this as the best of the TL models that were tried (which did not include EfficientNet). This model is available packaged with the Keras module, so a Keras scheme was used to build this model.

Next, the EfficientNet transfer learning model was compared. In this research, the B0 model was chosen for comparison to ResNet50, for its simplicity in relation to the other models and processing time constraints. Further research should include use of other models up to B7.

The final model utilized varied activation functions to avoid sticking to local minima; the CNN layers used sigmoid, while 2 dense hidden layers used swish and ReLU, respectively, and the final dense layer, of course, was fit with a softmax activation for the 38 classes.

Categorical cross-entropy was used as the loss function due to the multiclass nature of the task. Adam optimization was also used (Adam is often a best-practice optimizer), and the metric compared was accuracy because it is of equal interest to classify all categories correctly.

1. Results

Table 2. Comparison of Models by Efficiency and Processing Time

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Description | Processing Time (min) | Accuracy (%) |
| A | Non-TL (CNN) | 24.64 | 68.75 |
| B | ResNet50 | 72.03 | 90.09 |
| C | EfficientNetB0 | 85.71 | 97.81 |

Table 2 illustrates the vast difference in performance between the non-TL models which have not been pre-trained on Imagenet, and those which have been pre-trained. It also shows that the processing time went up considerably with each increase in accuracy.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

==========================================================

conv2d\_11 (Conv2D) (None, 126, 126, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_16 (MaxPooling (None, 63, 63, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_7 (Flatten) (None, 127008) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_19 (Dense) (None, 512) 65028608

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_20 (Dense) (None, 256) 131328

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_21 (Dense) (None, 38) 9766

==========================================================

Total params: 65,170,598

Trainable params: 65,170,598

Non-trainable params: 0

Figure 1a. CNN model structure used (no Transfer Learning) – A

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

==========================================================

resnet50 (Model) (None, 2048) 23587712

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 38) 77862

=========================================================

Total params: 23,665,574

Trainable params: 23,612,454

Non-trainable params: 53,120

Figure 1b. ResNet50 final model – B

In the ResNet50 architecture in model B, pre-existing ResNet50 layers are wrapped into one and summarizes into one ‘layer,’ though this represents many layers.

EfficientNetB0 (Model)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_5 (Conv2D) (None, 4, 4, 32) 368672 top\_activation[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_6 (MaxPooling2D) (None, 2, 2, 32) 0 conv2d\_5[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_6 (Conv2D) (None, 2, 2, 64) 18496 max\_pooling2d\_6[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_7 (MaxPooling2D) (None, 1, 1, 64) 0 conv2d\_6[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_6 (Flatten) (None, 64) 0 max\_pooling2d\_7[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_12 (Dense) (None, 512) 33280 flatten\_6[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_6 (Dropout) (None, 512) 0 dense\_12[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_13 (Dense) (None, 256) 131328 dropout\_6[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense\_14 (Dense) (None, 38) 9766 dense\_13[0][0]

===============================================================================================

Total params: 4,611,106

Trainable params: 4,569,090

Non-trainable params: 42,016

Figure 1c. EfficientNetB0 final model – C

In the EfficientNetB0 model A, the layers when summary() is called are not “wrapped” in the same manner as the ResNet50 model, but these have been summarized for conciseness. The entire model architecture can be viewed in the Python code.

IV. Discussion

It is evident from Table 2 that EfficientNet by far gives the best accuracy of the models. Fewer training steps per epoch were also needed to maximize the accuracy.

Though the two-layer CNN model achieved the best testing results in the EfficientNet model (C), one layer of CNN achieved better results with the no TL model (A), and with ResNet50, a simple model with only the addition of an output layer (B) outperformed other models with up to 4 layers of added CNNs, which could have been overfitting to the training data as training greatly outperformed validation accuracy in these tests.

Different regularization methods were also varied, and in the case of the non-TL model, L1 regularization in the output layer performed better than dropout layers added in various positions. The opposite was true for the EfficientNet model, where dropout with p = .3 in two locations did improve accuracy, but regularization in the output layer did not.

V. Conclusion

Of the models tested in this research, the EfficientNet model by far exceeded the multiclass classification accuracy able to be obtained by the non-pretrained and even ResNet50 model. This accuracy may be able to be improved upon with further research into EfficientNet models and using higher order models with GPU or TPU, and the results of this CPU study are encouraging that it is likely possible to achieve best-in-class accuracy with such a strategy.

References

Abade, Andre S, et al. “Plant Diseases Recognition on Images Using Convolutional Neural Networks: A Systematic Review.” *ArXiv.org*, Preprint, 10 Sept. 2020, arxiv.org/pdf/2009.04365.pdfhttps://arxiv.org/pdf/2009.04365.pdf.

Bhattarai, Samir. “New Plant Diseases Dataset.” *Kaggle*, 18 Nov. 2018, www.kaggle.com/vipoooool/new-plant-diseases-dataset.

Brownlee, Jason. “A Gentle Introduction to Pooling Layers for Convolutional Neural Networks.” *Machine Learning Mastery*, 5 July 2019, machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

Bozinovski, Stevo and Ante Fulgosi (1976). "The influence of pattern similarity and transfer learning upon training of a base perceptron B2." (original in Croatian) Proceedings of Symposium Informatica 3-121-5, Bled.

Dickson, Ben. “What Are Convolutional Neural Networks (CNN)?” *TechTalks*, 6 Jan. 2020, bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/.

Jahn, Adwin. “Keras Image Preprocessing: Scaling Image Pixels for Training.” *LinkedIn*, 16 Feb. 2017, www.linkedin.com/pulse/keras-image-preprocessing-scaling-pixels-training-adwin-jahn.

Kızrak, Ayyüce. “Reviewing EfficientNet: Increasing the Accuracy and Robustness of CNNs.” *Medium*, Heartbeat, 25 Apr. 2020, heartbeat.fritz.ai/reviewing-efficientnet-increasing-the-accuracy-and-robustness-of-cnns-6aaf411fc81d.

“Papers with Code - ImageNet Benchmark (Image Classification).” *The Latest in Machine Learning*, 2020, paperswithcode.com/sota/image-classification-on-imagenet.

Sagar, Abhinav, and J Dheeba. “On Using Transfer Learning For Plant Disease Detection.” 2020, doi:10.1101/2020.05.22.110957.

Tan, Mingxing, and Quoc V Le. “EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling.” *Google AI Blog*, 29 May 2019, ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html.

“We Help African Smallholder Farmers Adapt to Climate Change.” *PlantVillage*, plantvillage.psu.edu/.

Appendix

Python Code for “Transfer Learning for Plant Disease Classification with EfficientNet”

See attached file: “Full Code Plant Disease Classification with EfficientNet”