Cassava Leaf Disease Classifiction with ResNet50

Justin Harris
Computer Science
Deep Learning
Conputer Science
CS 474 - Spring 2021

Cassava Leaf Disease Classifiction with Resnet50

Justin Harris
Computer Science
University of Idaho
Moscow, Idaho
harr1433@vandals.uidaho.edu

***Abstract*—As one of the largest providers of carbohydrates in Africa, cassava is a key food security crop by smallholder farmers because of its ability to withstand harsh conditions. With an accurate Deep Learning model, it may be possible to help farmers quickly identify diseased plants, potentially saving crops before irreparable damage take place. [1]**

Keywords—ResNet50, Deep Learning, Python, Image Classification, Tensorflow, Keras

I. INTRODUCTION

The Cassava Leaf Disease Classification competition introduces a dataset consisting of 21.367 labeled images collected during a survey in Uganda. The goal of this project is to fine-tune a ResNet50 pre-trained Deep Learning model on the dataset provided by cassava to identify which viral disease a plant may have, and to additionally distinguish healthy plants from images of the plant leaves [1].

The Cassava Leaf Disease Classification competition and dataset used for this project can be found at the following link: https://www.kaggle.com/c/cassava-leaf-disease-classification/overview .

II. METHOD

A. *Data Input and Augmentation*

The dataset used for this project can be found at the Cassava Leaf Disease Classification competition page hosted on Kaggle.com at the link given in the Introduction. For this project, the data is split up into 17,117 images in the training set, and 4,280 images in the testing set.

The training images with their associated labels are loaded into the project and augmented through the use of Keras's ImageDataGenerator. Since ResNet-50 is the base model being used for this project, the images have been augmented according to the preference of the ResNet-50 model [2], with the training data being split into a training set and a validation set at a 80/20 split. An example of the image augmentation can be seen in Figure 1.

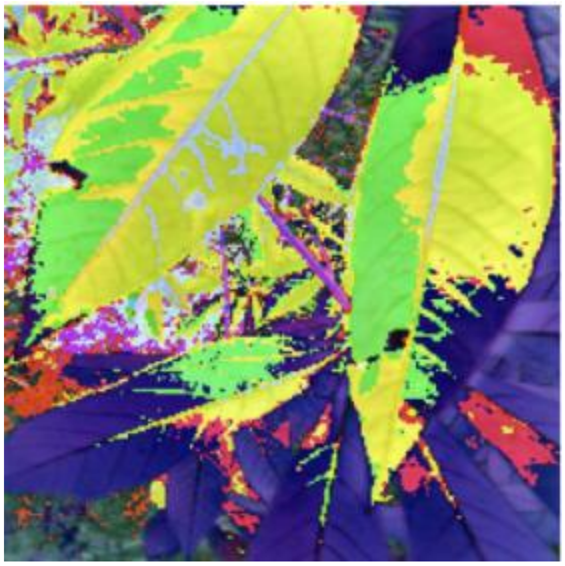


Figure 1: ResNet-50 Data Augmentation

B. *Model Architecture and Parameters*

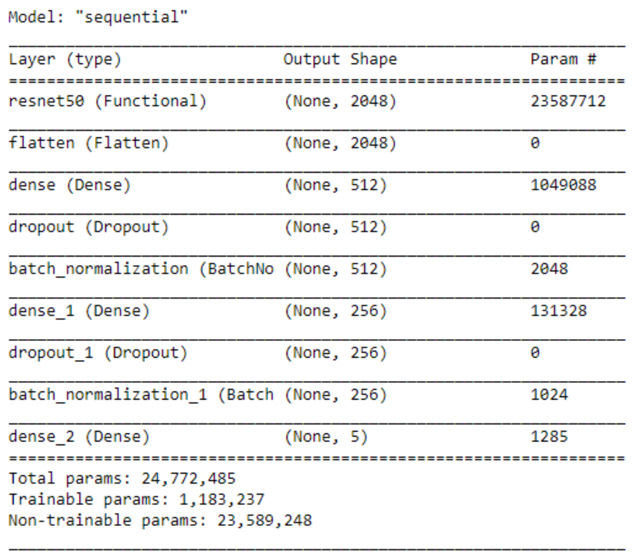
The approach used for this project is based around using the ResNet-50 pre-trained model. ResNet-50 is a convolutional neural network (CNN) that is 50 layers deep, including 1 Max-pooling and one Average-Pooling layer [3]. In addition to this, we add then flatten the Feature Map output of the ResNet-50 pre-trained model and input the result into a Dense layer consisting of 512 hidden nodes with ReLU activation for additional training.

Next, we implement a Dropout regularization of 0.5, which will randomly ignore half of the neurons during training. The goal of this layer is to help the network become less sensitive to the specific weights of each neuron and to help reduce the likelihood of the model having a problem with over-fitting.

The third layer implemented is a Batch Normalization layer. The goal of this layer is to normalize the input to the next activation function, in order for the input to be centered in the linear section of the next ReLU activation function.

The fourth, fifth, and sixth additional layers follow the pattern of the previous three layers (Dense, Dropout, Batch Normalization), except that the number of hidden nodes in the Dense layer are reduce to 512.

The last layer in out network is the Output layer. The output layer is another Dense layer with only 5 nodes, using Softmax as the activation function. Each node represents one of the classification labels used for this dataset. These labels are defined by the Cassava Leaf Disease Classification competition consisting of labels mentioned in section II.A. An overview of the architecture can be seen in Figure 2.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2048)              23587712
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1049088
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
batch_normalization (BatchNo (None, 512)               2048
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 1285
=================================================================
Total params: 24,772,485
Trainable params: 1,183,237
Non-trainable params: 23,589,248
_________________________________________________________________
```

Figure 2: Model Architecture

Finally, our model uses a default batch size of 32, the Adam optimizer with an initial learning rate of 3e-4, and a callback to reduce the learning rate when the validation accuracy stops improving during training. The idea behind this is to continue to improve the model's overall accuracy by reducing the learning rate when it becomes stagnant.

III. EXPERIMENTAL RESULTS

A. *Training Results*

As discussed in the previous section, training the model on the dataset was done with a default batch size of 32, and a learning rate of 3e-4. Various epochs were tested throughout multiple training sessions. The epochs used for training were 10, 20, 30, 50, 100, 150, and 300. However, the ideal results seem to lie between the 20 and 30 epoch range of training, as seen in Figures 3 and 4. Figure 3 shows the model's training accuracy over 30 epochs, and Figure 4 shows the model's training loss over 30 epochs.

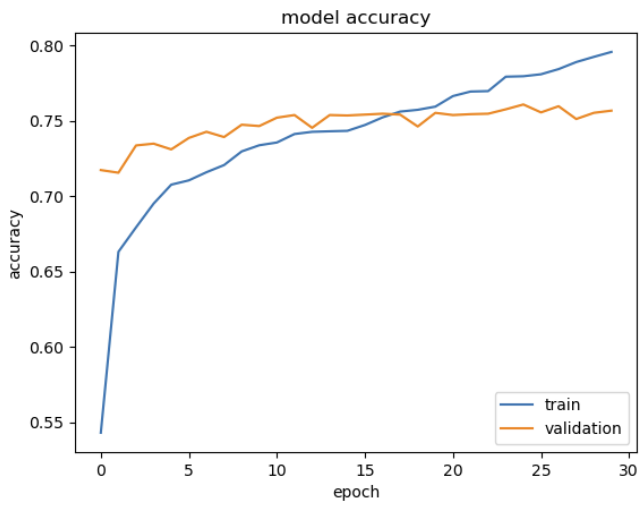


Figure 3: Training Accuracy

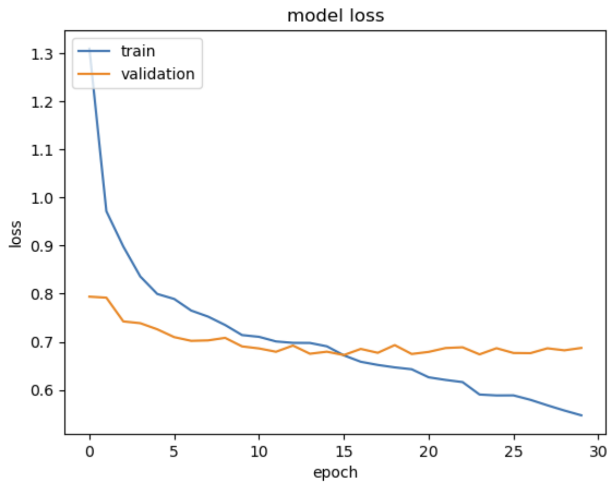


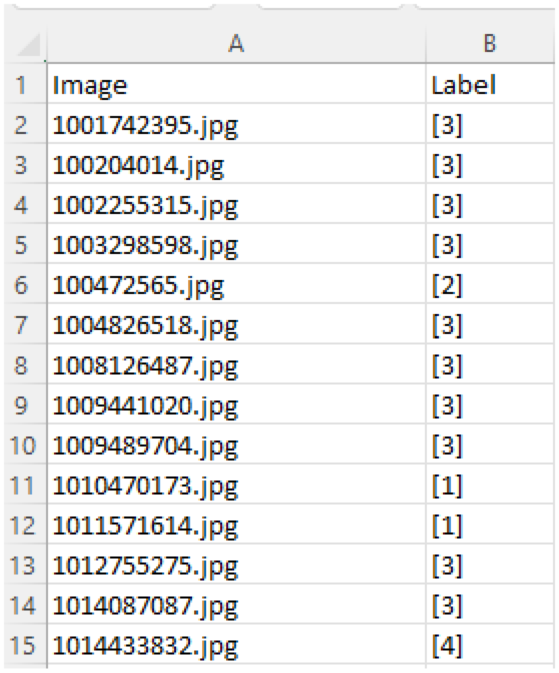
Figure 4: Training Loss

As seen in Figure 3, the training and validation accuracy start to converge starting around the 5th epoch of training. They then start relatively close in value until after the 25th epoch of training where the training accuracy and training loss start to diverge. When testing larger epoch values for training, the testing accuracy and validation accuracy would continue to diverge from each other, indicating an overfitting issue with the model beyond the 30th epoch of training.

The same results are concurrent the training and testing loss, however the loss of the training and validation data diverge earlier, after the 15th epoch. Based on the graphs, the testing and validation accuracy are able to reach about 75% accuracy on this dataset.

B. Testing Results

Testing results were difficult to gather for this project, since predictions were made by loading each image individually and outputting the image name and label prediction to a .csv file, a sample of this is shown in Figure 4. For the CS474 course, information about the test picture labels were not provided so that the testing set of data could not be used to train the model on. I would speculate that the testing results would have around the same accuracy value as the validation accuracy, likely between 73% to 75%.



	A	B
1	Image	Label
2	1001742395.jpg	[3]
3	100204014.jpg	[3]
4	1002255315.jpg	[3]
5	1003298598.jpg	[3]
6	100472565.jpg	[2]
7	1004826518.jpg	[3]
8	1008126487.jpg	[3]
9	1009441020.jpg	[3]
10	1009489704.jpg	[3]
11	1010470173.jpg	[1]
12	1011571614.jpg	[1]
13	1012755275.jpg	[3]
14	1014087087.jpg	[3]
15	1014433832.jpg	[4]

Figure 4: Test Prediction Output

IV. CONCLUSION

Overall, this fine-tuning of the ResNet-50 pre-trained model on the Cassava Leaf Disease Classification dataset was able to achieve average results, in my opinion, when comparing to results achieved by other submissions.

There may be more room for improvement to achieve better results from this model on this dataset, however I was unable to find a solution. After attempting to add multiple additional Dense layer to the model, adjusting the Adam optimizer's learning rate, trying other various optimizer such as RMSProp and Nadam, the best results that I was able to achieve in this project were done so using the model set up and parameters outlined in this report.

REFERENCES

V. References

[1] [1]"Cassava Leaf Disease Classification | Kaggle", *Kaggle.com*, 2021. [Online]. Available: https://www.kaggle.com/c/cassava-leaf-disease-classification/overview. [Accessed: 10- Dec- 2021].

[2] [2]K. Team, "Keras documentation: Keras API reference", *Keras.io*, 2021. [Online]. Available: https://keras.io/api/. [Accessed: 14- Dec- 2021].

[3] [3]"MathWorks Help Center", *MathWorks.com*, 2021. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/resnet50.html;jsessionid=b0b30918496bfa33f8d2727b7bc8#:~:text=ResNet%2D50%20is%20a%20convolutional,%2C%20pencil%2C%20and%20many%20animals. [Accessed: 14- Dec- 2021].