Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Final Project Report

Implementation & Code Usage

The implementation that was performed was option 2: Deep Learning. The algorithm that was used was a convolutional neural network (CNN). From an abstract perspective, neural networks are much like a brain, where information comes into the brain, (This can stem from an outside stimulus or an internal stimuli) becomes processed, then the processed information can then be shared across many neurons to perform a response. Due to brain's natural interconnected architecture, neural networks try to mimic the behavior of the brain by channeling/processing the input to some sort of output using the notion of graph theory. Using graph theory and what knowledge we have so far of the brain, we can learn to detect patterns in data!

The image dataset that was used for the deep learning model was comprised of 18,632 images apple tree leaves that ranged from 12 different labels that were associated apple tree leaves. The labels were the following:

Leaf Type	Image
Complex	

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Frog Eye Leaf Spot Complex	
Frog Eye Leaf Spot	
Healthy	

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Powdery Mildew Complex	
Powdery Mildew	
Rust Complex	

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Rust Frog Eye Leaf Spot	
Rust	
Scab Frog Eye Leaf Spot Complex	

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Scab Frog Eye Leaf Spot	
Scab	

The objective was to diagnosed the condition of the apple tree based the morphologies the tree leaves were expressing in the images.

Libraries Used

The following libraries that were used was the following:

Libraries	**Used For**
Numpy Pandas	Data frame Manipulations
Matplotlib PIL	Visualizations
os pathlib	Command Line usage for file manipulations
Tensorflow Keras	Machine Learning
Sklearn	Classification Metrics

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Screenshots

```
In [222]: train_data = pd.read_csv("data/train.csv")
          train_data.head()
```

Out[222]:

	image	labels
0	800113bb65efe69e.jpg	healthy
1	8002cb321f8bfcdf.jpg	scab frog_eye_leaf_spot complex
2	80070f7fb5e2ccaa.jpg	scab
3	80077517781fb94f.jpg	scab
4	800cbf0ff87721f8.jpg	complex

Figure 1: Image Labels

Visualization of the images

```
In [274]: import matplotlib.pyplot as plt

          plt.figure(figsize=(10, 15))
          for images, labels in train_ds.take(1):
              for i in range(9):
                  ax = plt.subplot(3, 3, i + 1)
                  plt.imshow(images[i].numpy().astype("uint8"))
                  plt.title(class_names[labels[i]])
                  plt.axis("off")
```

Figure 2: Sample Images from the training data with respect to their labels

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_1 (Sequential)    (None, 180, 180, 3)       0
_________________________________________________________________
rescaling_1 (Rescaling)      (None, 180, 180, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 180, 180, 16)      448
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 180, 180, 16)      2320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 90, 90, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 90, 90, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 22, 22, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3965056
_________________________________________________________________
dense_1 (Dense)              (None, 12)                1548
=================================================================
Total params: 3,992,508
Trainable params: 3,992,508
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: Convolutional Neural Network layers

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

```
[7]: epochs = 15
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/15
466/466 [==============================] - 464s 989ms/step - loss: 1.7361 - accuracy: 0.3468 - val_loss: 1.6476 - val_accurac
y: 0.3854
Epoch 2/15
466/466 [==============================] - 320s 686ms/step - loss: 1.4557 - accuracy: 0.4791 - val_loss: 1.3727 - val_accurac
y: 0.5183
Epoch 3/15
466/466 [==============================] - 330s 707ms/step - loss: 1.1650 - accuracy: 0.5865 - val_loss: 1.0836 - val_accurac
y: 0.6170
Epoch 4/15
466/466 [==============================] - 319s 686ms/step - loss: 1.0329 - accuracy: 0.6309 - val_loss: 0.9450 - val_accurac
y: 0.6731
Epoch 5/15
466/466 [==============================] - 347s 744ms/step - loss: 0.9610 - accuracy: 0.6624 - val_loss: 0.9072 - val_accurac
y: 0.6903
Epoch 6/15
466/466 [==============================] - 344s 738ms/step - loss: 0.8863 - accuracy: 0.6919 - val_loss: 0.8429 - val_accurac
y: 0.7115
Epoch 7/15
466/466 [==============================] - 334s 717ms/step - loss: 0.8097 - accuracy: 0.7239 - val_loss: 0.7917 - val_accurac
y: 0.7300
Epoch 8/15
466/466 [==============================] - 322s 691ms/step - loss: 0.7572 - accuracy: 0.7467 - val_loss: 0.7745 - val_accurac
y: 0.7383
Epoch 9/15
466/466 [==============================] - 319s 685ms/step - loss: 0.7203 - accuracy: 0.7586 - val_loss: 0.8495 - val_accurac
y: 0.7169
Epoch 10/15
466/466 [==============================] - 320s 686ms/step - loss: 0.7039 - accuracy: 0.7658 - val_loss: 0.7024 - val_accurac
y: 0.7633
Epoch 11/15
466/466 [==============================] - 319s 684ms/step - loss: 0.6836 - accuracy: 0.7722 - val_loss: 0.7417 - val_accurac
y: 0.7595
```

Figure 4 Training the model for each epoch the model will use 466 images to train and then cross-validate with a different set of images

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

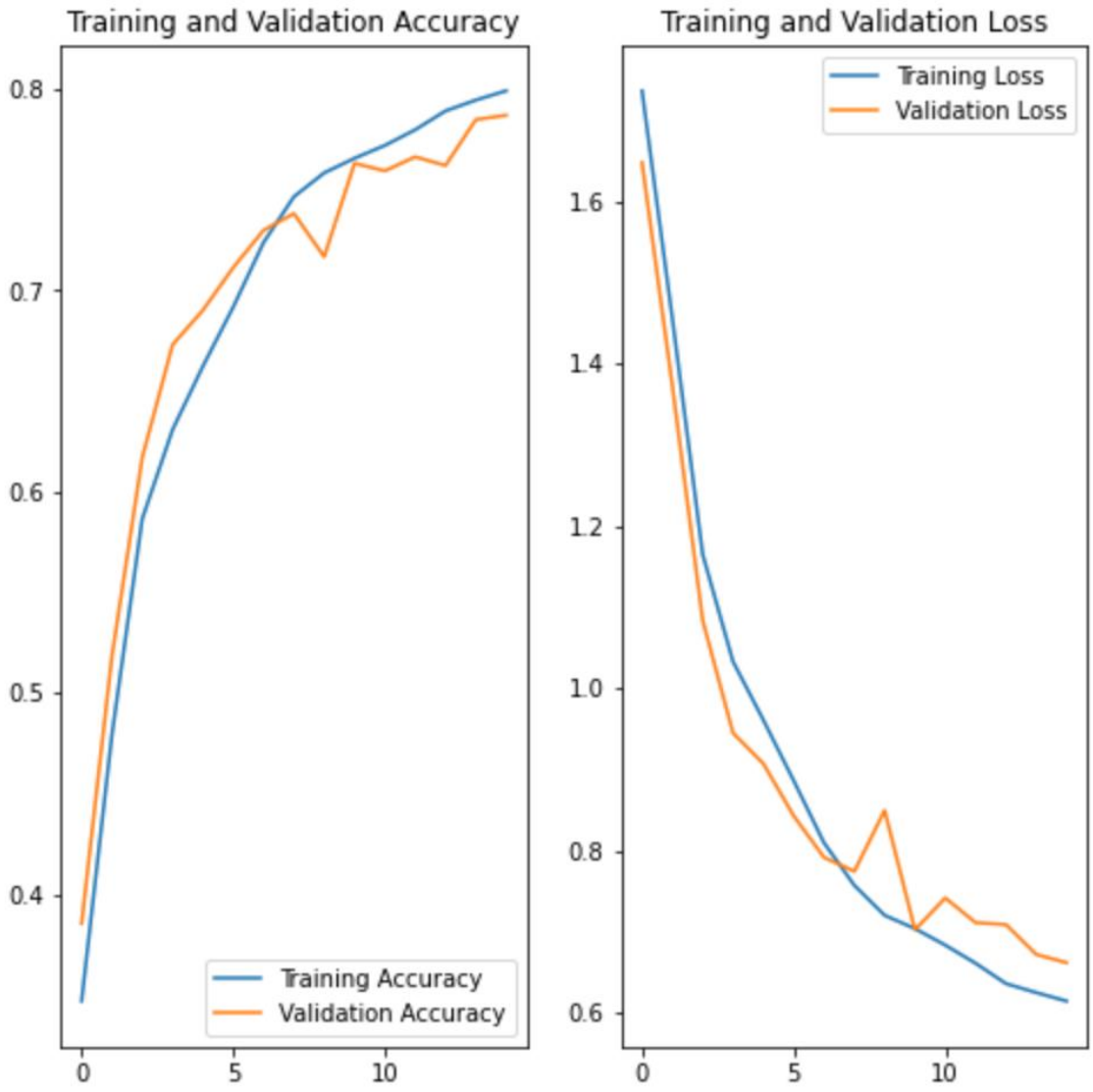

Figure 5: Graphing the training and validation accuracy: This is a good fit!

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

Final Dataframe with all the statistical classification analysis

```
96]: df_stats
```

96]:

	label_name	true_neg	false_pos	false_neg	true_pos	accuracy	precision	tpr	tnr	fnr	fpr	f1_score	error_rate
0	complex	3299	122	145	160	0.928341	0.567376	0.524590	0.964338	0.475410	0.035662	0.545145	0.071659
1	frog_eye_leaf_spot	2902	195	71	558	0.928610	0.741036	0.887122	0.937036	0.112878	0.062964	0.807525	0.071390
2	frog_eye_leaf_spot complex	3696	0	30	0	0.991948	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.008052
3	healthy	2638	161	66	861	0.939077	0.842466	0.928803	0.942479	0.071197	0.057521	0.883530	0.060923
4	powdery_mildew	3464	14	79	169	0.975040	0.923497	0.681452	0.995975	0.318548	0.004025	0.784223	0.024960
5	powdery_mildew complex	3714	1	11	0	0.996779	0.000000	0.000000	0.999731	1.000000	0.000269	0.000000	0.003221
6	rust	3236	127	18	345	0.961084	0.730932	0.950413	0.962236	0.049587	0.037764	0.826347	0.038916
7	rust complex	3706	0	19	1	0.994901	1.000000	0.050000	1.000000	0.950000	0.000000	0.095238	0.005099
8	rust frog_eye_leaf_spot	3699	0	27	0	0.992754	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.007246
9	scab	2610	138	161	817	0.919753	0.855497	0.835378	0.949782	0.164622	0.050218	0.845318	0.080247
10	scab frog_eye_leaf_spot	3545	35	124	22	0.957327	0.385965	0.150685	0.990223	0.849315	0.009777	0.216749	0.042673
11	scab frog_eye_leaf_spot complex	3684	0	42	0	0.988728	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.011272

Figure 6: Overall Statistical Analysis of Classification

Classification Report

- We can see that this function from sklearn learn aligns up perfectly with the dataframe above!

```
298]: from sklearn.metrics import classification_report
       print(classification_report(df['true_label'],df['pred_label']))
```

```
                              precision    recall  f1-score   support

                     complex       0.57      0.52      0.55       305
           frog_eye_leaf_spot       0.74      0.89      0.81       629
frog_eye_leaf_spot complex          0.00      0.00      0.00        30
                     healthy       0.84      0.93      0.88       927
              powdery_mildew       0.92      0.68      0.78       248
        powdery_mildew complex       0.00      0.00      0.00        11
                        rust       0.73      0.95      0.83       363
                rust complex       1.00      0.05      0.10        20
      rust frog_eye_leaf_spot       0.00      0.00      0.00        27
                        scab       0.86      0.84      0.85       978
      scab frog_eye_leaf_spot       0.39      0.15      0.22       146
scab frog_eye_leaf_spot complex       0.00      0.00      0.00        42

                    accuracy                           0.79      3726
                   macro avg       0.50      0.42      0.42      3726
                weighted avg       0.76      0.79      0.76      3726
```

Figure 7: Classification Report, notice how the precision, recall, and f1-score are well aligned with each other!!!

Michael Woo
02 May 2021
Professor Yasser Abduallah
CS 634 104 Data Mining

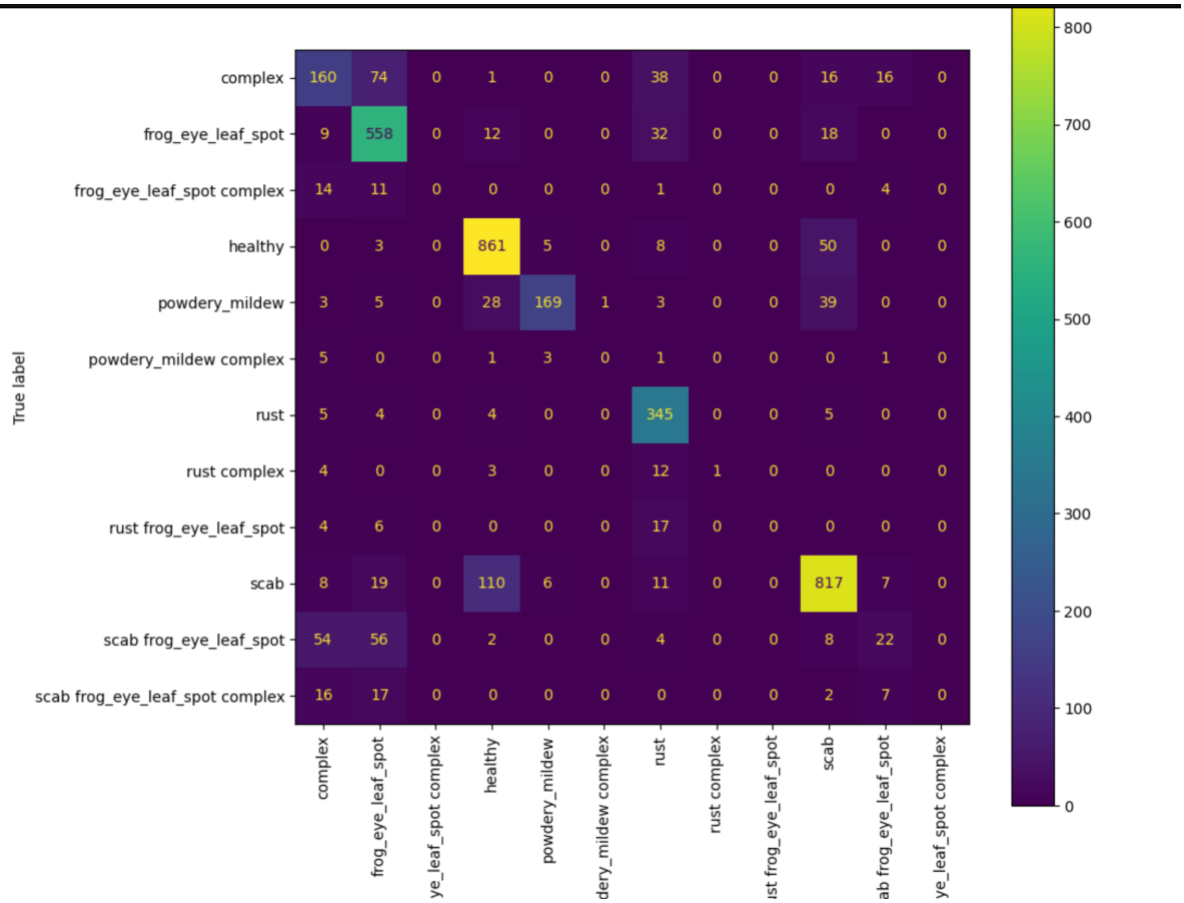

Figure 8: Overall Confusion Matrix for the Deep Learning Model

Other

The source code (.py file) and a sample dataset of images (25 images per label) will be attached to the zip file. I would recommend using the jupyter notebook file to view the results and the robust visualization. While using the .py file to see if my code runs.
Caution: Running the jupyter notebook file will use a lot of CPU utilization

Link to Git Repository

https://github.com/MichaelWoo-git/Plant_Pathology_Apple_Trees.git