# CS 614 – Applications of ML

## Project 1 – Computer Vision

**Submitted by: Manil Shrestha**

**Project: Plant disease classification through photographs of leaves**

1. Pitch:

*State the circumstances an organization or sets of users would be willing to fund the proposed ML application based on their perceived value. Value could be financial, or any other type of outcome organizations or users need to obtain (e.g., institutional image, customer satisfaction, increased quality, or efficiency).*

Advancement in computer vision technology can be of great help to farmers and people interested in house (indoor) plants. With the use of this model, plant enthusiasts can identify any diseases that may need attention. The model created in this project can be incorporated into a mobile app that can provide the type of disease. With information on type of disease, mobile developer can use a database to add necessary solutions, such as fertilizers or changes in growing conditions to the plant. Highly effective for famers

2. Data source:

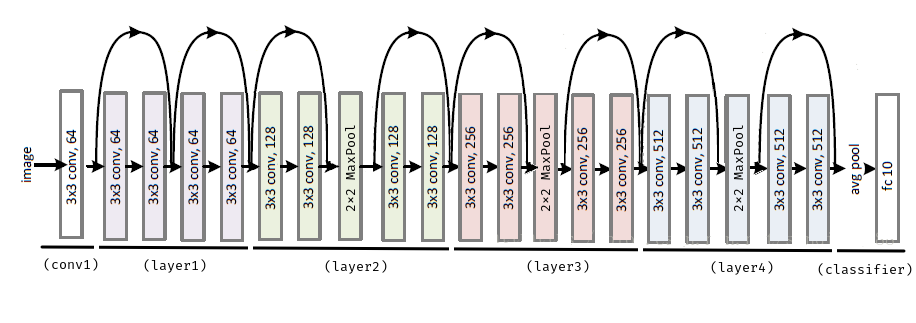
*Indicate with a link where you obtained the data. If you generated the data yourself, please provide a link to the code of the used approach.*

This dataset consists of about 87K rgb images of healthy and diseased crop leaves which is categorized into 38 different classes. Data was downloaded from a Kaggle competition: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

3. Model and data justification:

*Justify why you chose a specific model to learn from the selected data. If you learned that a given more would be suitable for a type of data from a publication, please provide a link to the source. Note you still have to justify it with your own words (as always, limit to three sentences).*

To achieve high accuracy on my task, I utilized a ResNet-18 model that had been pre-trained on the large-scale ImageNet dataset. ResNet-18 has demonstrated outstanding performance on a variety of classification tasks, as shown by a [2018 ACM paper](https://dl.acm.org/doi/abs/10.1145/3194452.3194461)(<https://dl.acm.org/doi/abs/10.1145/3194452.3194461>) and other studies. [Khan et al., 2018]. Given its track record, I was confident that this pre-trained model would produce strong results on my own task.



Fix the weights from ImageNet pretraining

Finetune the weights for the FC layer

4. Commented examples:

*Indicate the input where trained model is applied, the output and whether it is as expected or any observations you may have.*

A green leaf with brown spots

Description automatically generated with medium confidence

5. Testing:

*Provide a confusion matrix with one or more metrics and comment the results.*

|  |  |
| --- | --- |
| *A picture containing text, line, screenshot, diagram  Description automatically generated* | *A picture containing text, diagram, line, screenshot  Description automatically generated* |

Fig: Loss and accuracy reporting by epoch

The confusion matrix below shows the frequency of actual labels being predicted correctly or not in the validation dataset. As you can see the diagonals have very high frequency, meaning most of the times (i.e., 93.5%), model correctly predicts the labels of the picture. This is by just finetuning the last FC layer of pretrained ResNet-18 model for 25 epochs.

A picture containing text, screenshot, plot, line

Description automatically generated

*Fig: Confusion matrix for the validation set*

|  |  |  |  |
| --- | --- | --- | --- |
| **Labels** | **precision** | **recall** | **f1-score** |
| 0 | 0.961 | 0.938 | 0.95 |
| 1 | 0.996 | 0.93 | 0.961 |
| 2 | 0.958 | 0.927 | 0.942 |
| 3 | 0.947 | 0.96 | 0.954 |
| 4 | 0.901 | 0.978 | 0.938 |
| 5 | 0.993 | 0.962 | 0.977 |
| 6 | 0.98 | 0.985 | 0.982 |
| 7 | 0.956 | 0.854 | 0.902 |
| 8 | 0.986 | 1 | 0.993 |
| 9 | 0.886 | 0.964 | 0.924 |
| 10 | 1 | 0.989 | 0.995 |
| 11 | 0.978 | 0.939 | 0.958 |
| 12 | 0.938 | 0.979 | 0.958 |
| 13 | 0.993 | 0.977 | 0.985 |
| 14 | 0.974 | 0.979 | 0.976 |
| 15 | 0.986 | 0.99 | 0.988 |
| 16 | 0.952 | 0.954 | 0.953 |
| 17 | 0.946 | 0.972 | 0.959 |
| 18 | 0.932 | 0.941 | 0.937 |
| 19 | 0.932 | 0.968 | 0.95 |
| 20 | 0.985 | 0.957 | 0.971 |
| 21 | 0.948 | 0.905 | 0.926 |
| 22 | 0.967 | 0.893 | 0.928 |
| 23 | 0.991 | 0.98 | 0.985 |
| 24 | 0.978 | 0.952 | 0.965 |
| 25 | 0.991 | 0.998 | 0.994 |
| 26 | 0.977 | 0.975 | 0.976 |
| 27 | 0.978 | 0.996 | 0.987 |
| 28 | 0.967 | 0.906 | 0.936 |
| 29 | 0.777 | 0.775 | 0.776 |
| 30 | 0.825 | 0.834 | 0.829 |
| 31 | 0.859 | 0.906 | 0.882 |
| 32 | 0.834 | 0.821 | 0.828 |
| 33 | 0.75 | 0.897 | 0.817 |
| 34 | 0.787 | 0.672 | 0.725 |
| 35 | 0.934 | 0.976 | 0.954 |
| 36 | 0.871 | 0.962 | 0.914 |
| 37 | 0.925 | 0.894 | 0.909 |

6. Code and instructions to run it:

*Provide a link to the code and any required instructions to run it. Please include some testing examples so we can quickly experience what you experienced with the model.*

Here is the link to Jupyter Notebook which has been uploaded in the github:

<https://github.com/ManilShrestha/SparseCoding/blob/main/Plants-ResNet18-Pretrained.ipynb>

You can recreate the training and experiments by following the steps:

1. Change the data\_dir to the root directory where your data resides after downloading from: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset
2. Make sure all the dependencies are installed in your machine.
3. Run all the cells in the notebook.