[Team 35] Project C2: Leaf Wilting

Chintan Gandhi (cagandhi), Shreyas Chikkballapur Muralidhara (schikkb), Nikhil Sundaraswamy (nsundar)

I. Methodology

The problem that we are dealing with in this project is a multi-class classification problem of leaf wilting. From the ProjC1 leaderboard, we observed that teams with higher accuracy had leveraged deep neural networks and also used transfer learning. Motivated by the results as described in [1] and the notion that a deep neural network given enough data is able to learn a complex function, we propose a transfer learning approach with data augmentation to leverage the classification power gained from training on the ImageNet dataset and fine-tune the model for our classification problem.

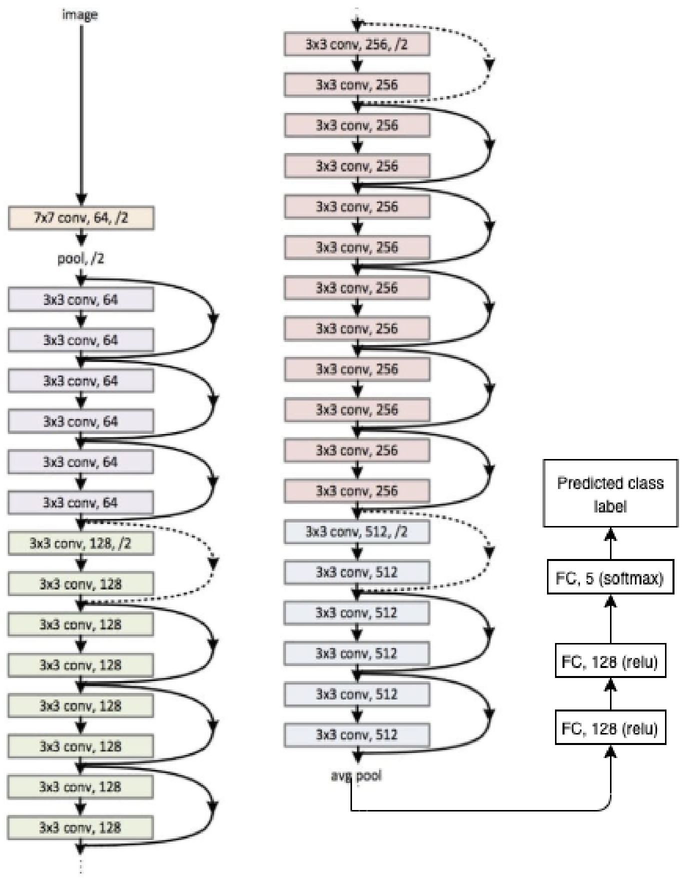

Fig. 1: Model Architecture

We use a ResNet architecture described in [2] that tries to learn the function using a deep residual learning framework. This is motivated by the fact a deep neural network is sometimes unable to accurately learn the input-output mapping due to the problem of vanishing gradient.

Contrary to many deep learning architectures which learn a hypothesis function relating the input data and the output, ResNet learns the residual of a hypothesis function and input. In other words, it minimizes the residual error thus learning the mapping effectively. The intution behind residual blocks is that if identity mapping is optimal, then we can easily push the residual to zero rather than fitting an identical mapping using a stack of non linear layers. ResNet-34 (34 layer residual) achieved a top-5 validation error of 5.71 percent better than BN-inception and VGG [2], winning the 1st place in ILSVRC-2015.

As shown in Figure 1, the architecture has a feedforward network which is used to calculate the residual of the function. As the depth increases, the residual is better learnt which in turn, means the input-output mapping is more effectively learnt when compared to a simple deep network like VGGNet.

The final structure of our network is:
ResNet50 base - 128 (FC layer 1, RELU) - 128 (FC layer 2, RELU) - 5 (Softmax Output).

The toolbox and libraries along with the specific functions used are listed below:

Libraries	**Function Names**
numpy	argmax, bincount, mathceil
pandas	read_csv
opencv	imread
scikit-learn	svm.SVC, decomposition.PCA, classification_report, confusion_matrix, compute_class_weight
keras applications	Resnet50, VGG16, InceptionResnetV2, DenseNet121
keras optimizers	Adam
keras utils	ImageDataGenerator, to_categorical
keras models	Model, load_model, load_weights
keras callbacks	EarlyStopping, ModelCheckpoint

TABLE I: Toolboxes used

II. Model Training and Selection

A. Model Training

1) **Training and validation data split:** The training data is highly imbalanced as class 0 has maximum images, whereas classes 2 and 3 have extremely less samples. To mimic the balanced test dataset, we create our validation dataset in a balanced manner. We split the data in 80-20 fashion with the validation split containing 51 images from each class. These 51 images are around 40% of the images in classes 2 and 3. We take a larger number of images in order to accurately judge the performance of the classifier while also

taking care not to deplete the training data for each class significantly. The exact count of images in each class for training and validation data are specified in the table below:

Class	**0**	**1**	**2**	**3**	**4**
Training Data	437	278	79	80	146
Validation Data	51	51	51	51	51
Total	488	329	130	131	197

TABLE II: Class distribution for data

2) **Transfer Learning and Data Augmentation:** Observing the results from our ProjC1 submission and the released scoreboard, we observed that groups which had performed transfer learning for complex neural network architectures had a significantly higher accuracy on the test dataset. This motivated us to build a transfer learning model with the newly provided dataset. We know that the more complex the neural network, the more parameters it has and the more data it requires to not suffer from overfitting. Thus, we also deployed data augmentation techniques to artificially boost the size of our dataset.

   We used the Keras' ImageDataGenerator class to perform real-time data augmentation. The transformations that were performed for each image are listed below:

   - **rotation** to handle various angles in images
   - **shearing** to incorporate stretching of images
   - **width and height shift** to incorporate different views
   - **zoom** to handle various focus levels
   - **brightness** to blacken images
   - **channel shift** to create an exposure effect
   - **horizontal flip** to make the classifier invariable to the location of plant rows

3) **Weighted loss function:** Since the training data before runtime augmentation is imbalanced, we also need to make sure that we provide weights to the loss suffered differently based on the class in order to compensate for the lack of training samples. So that our model is equally good at identifying all classes and not just the majority class, we penalize the misclassification of an under-represented class (class 2,3) much more than that of class 0. For example, if the class weights for class 0 and class 2 are 1 and 3 respectively, we are telling our classifier that misclassifying a class 3 example carries triple the cost than misclassifying a class 0 example. Since, the model can see very less examples of these under-represented classes, we force the model to learn quickly and get the classes 2 and 3 right much more quickly.

   The class weights were calculated with the Scikit-learn's function **compute_class_weight()** using the default formula:

$$\text{weight[class = c]} = \frac{\text{total n\_samples}}{\text{(n\_classes * n\_samples in class 'c')}}$$

   This justifies the fact that the class weight is inversely proportional to the number of samples it has and hence, for our case, class 0 will be given much lower weight than classes 2 and 3.

B. Model Selection

Transfer learning is a methodology where we use pre-trained deep learning models to fine-tune them to fit our data. The final softmax layer of 1000 units is removed and we add 2 fully connected layers followed by a softmax layer of 5 units to learn the information most suitable to leaf wilting detection. We use readily available pre-trained models from Keras library.

Firstly, we chose the model architecture we want to work with. We used different architectures of VGG16, Resnet50 and Inception-Resnet-v2 as base models, froze their layers and simply added the softmax output layer. We ran these models for our training data with weighted loss function for 10 epochs with a batch size of 16 and Adam optimizer and observed the validation accuracies and loss in Table IV. We see that Resnet50 has the maximum validation accuracy and minimum validation loss and hence, we chose our base architecture to be a Resnet50 model.

Model architecture	**Validation loss**	**Validation accuracy (%)**
VGG16	1.62	41.25
ResNet50	**1.34**	**48.75**
Inception-Resnet-v2	2.85	27.08
DenseNet 121	1.68	38.75

TABLE III: Model performance for various architectures

We also tried unfreezing some or all of the ResNet base layers but we didn't achieve good performance probably because of the abundance of model parameters and lack of training data to estimate those parameters. The experiments were performed for 10 epochs with a batch size of 16 and Adam optimizer. The best validation loss and accuracy was achieved when we completely froze the base ResNet model.

Model architecture	**Validation loss**	**Validation accuracy (%)**
Resnet50 last 5 unfrozen	1.61	22.92
ResNet50 unfrozen	1.79	21.25
ResNet50 frozen	**1.34**	**48.75**

TABLE IV: Model performance for different freezing levels of ResNet50

Next, we looked at the number of fully connected layers and the size of each layer that were to be added after ResNet base to fine-tune the model for our task of wilting classification. We saw that the last layer in ResNet base is a Global Pooling 2D layer whose output is a 2048-dimensional vector. We kept the ResNet layers frozen and tried different combinations for the number of layers and layer sizes. We found that adding 2 fully connected layers each with 128 neurons gave us the maximum validation accuracy of 55.42% and a validation loss of 1.18 for Adam optimizer training with a batch size of 16 for 10 epochs.

We also tried various batch sizes with our newly fixed architecture and the same parameters described in the previous paragraph. The validation accuracy and loss values were observed as follows:

Batch size	**Validation loss**	**Validation accuracy (%)**
16	**1.18**	**55.42**
32	1.59	52.5
64	1.62	49.17

TABLE V: Model performance for different batch sizes

All the above experiments were performed with Adam optimizer for a default learning rate of 0.001. We wanted to check how the change in learning rate affects the learning process. We observe the following results:

Learning rate	**Validation loss**	**Validation accuracy (%)**
0.0005	1.61	39.58
0.001	**1.18**	**55.42**
0.005	1.66	22.77

TABLE VI: Model performance for different learning rates

Another approach that we tried was a One-vs-Rest approach for neural networks. We built 5 different weighted ResNet50 models of our best architecture corresponding to the 5 different classes with the same training parameters. The predictions were then aggregated and the label for each image was given based on the confidence value of each model. The validation accuracy observed was 61.67% and the test distribution also turned out to be more or less balanced. However, when we submitted this prediction file to the scoreboard-4, the test accuracy came out to be just 26.5% and hence, we didn't move forward with the model.

III. Evaluation

Our final model has the architecture: ResNet50 base - 128 (FC layer 1) - 128 (FC layer 2) - 5 (Softmax Output). The final set of hyper-parameters are:

- No. of epochs: 25
- Batch size: 16
- Learning rate: 0.001
- Optimizer: Adam

We achieved a maximum validation accuracy of 64.7% on a balanced validation dataset containing 51 images of each class. For the test dataset, we achieved an accuracy of 57% with a RMSE of 0.815475.

The training and validation loss and accuracy plots are displayed below:

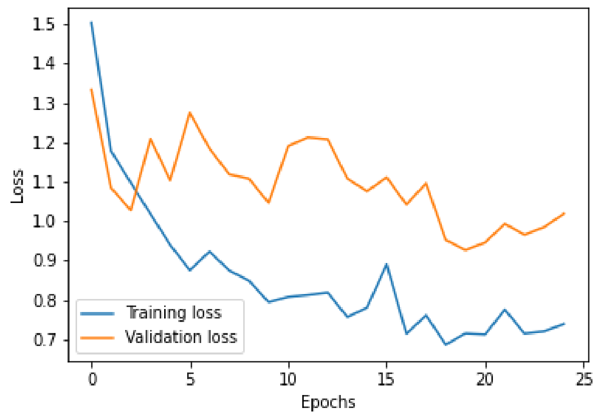

Fig. 2: Training and validation loss plot

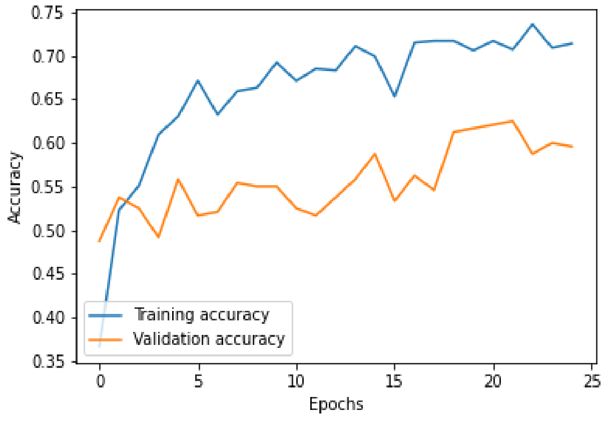

Fig. 3: Training and validation accuracy plot

The confusion matrix for the prediction looks like:

Wilting levels	**0**	**1**	**2**	**3**	**4**
0	**43**	5	1	2	0
1	26	**17**	7	1	0
2	4	2	**24**	20	1
3	0	0	8	**38**	5
4	0	0	0	8	**43**

TABLE VII: Confusion Matrix for predicted labels

We specify the precision, recall and F-1 score values for individual classes along with the macro and weighted averages in the table below.

Wilting Class	Precision	recall	f1-score	support
0	0.59	0.84	0.69	51
1	0.71	0.33	0.45	51
2	0.60	0.47	0.53	51
3	0.55	0.75	0.63	51
4	0.88	0.84	0.86	51
accuracy			0.65	255
macro avg	0.67	0.65	0.63	255
weighted avg	0.67	0.65	0.63	255

TABLE VIII: Precision, recall and F-1 scores for each class

IV. Extra Credit: Using Weather Data

A. Model Specification

To incorporate the information from both the textual data and image data, our approach consists of ensembling of 2 sub-models created from tabular and the image data. Ensemble model identifies level of wilting based on the model architecture described in previous section. The training dataset was imbalanced, while the testing dataset was balanced, which was compensated by training both the models with weights. Below is specification for sub-models and hybrid model:

1) **Tabular Data Model:** Training dataset consists of 620 records and 12 attributes with an imbalanced class distribution of [338,149,30,36,67]. Using PCA, 5 PCs were selected that captured 95% variance. Using Kernel SVM, the components were validated against different kernels by tuning the regularization parameter. SVM with Radial Basis kernel with critical factor 1 provided the best accuracy of 72% with balanced weights.

2) **Image Only Model:** We trained our proposed model, described in section II, on the images from train weather data. Due to imbalanced class labels, weighted loss function with updated weights was used in training the model. We achieved the best accuracy of 78% with same hyperparameter settings.

3) **Ensemble Model:** The prediction results from the sub-models, weighted confidence levels for individual classes were used to make final predictions. Hyperparameter model weights are tuned and set to 1 for each model to achieve the best accuracy of 82%.

B. Evaluation

1) **Tabular Data Model:** Our final architecture for kernel SVM is: iterations - 100000, kernel - RBF, regularization parameter - 1, class weight - balanced. We achieved an accuracy of 78% for balanced test dataset of 100 images. We specify the precision, recall and F-1 score by class along with macro and weighted averages in table below.

Wilting Class	Precision	recall	f1-score	support
0	0.81	0.65	0.72	20
1	0.68	0.75	0.71	20
2	0.79	0.75	0.77	20
3	0.62	0.75	0.68	20
4	0.74	0.70	0.72	20
accuracy			0.65	100
macro avg	0.73	0.72	0.72	100
weighted avg	0.73	0.72	0.72	100

TABLE IX: Precision, recall and F-1 scores by class for Tabular Data Model

2) **Image Only Model:** Resnet 50 with 2 FC layers is trained with the same hyperparameters as in section III and predictions are made. We achieved a maximum accuracy of 78% on the test dataset. We specify the precision, recall and F-1 score by class along with macro and weighted averages in the table shown below:

Wilting Class	Precision	recall	f1-score	support
0	0.75	0.90	0.82	20
1	0.88	0.75	0.81	20
2	0.79	0.55	0.65	20
3	0.63	0.85	0.72	20
4	0.94	0.85	0.89	20
accuracy			0.78	100
macro avg	0.80	0.78	0.78	100
weighted avg	0.80	0.78	0.78	100

TABLE X: Precision, recall and F-1 scores by class for Image Only Model

3) **Ensemble Model:** Ensembling the predictions of both models and assigning model weights - kernel SVM - 1 and Transfer learning - 1 achieved the best accuracy of hybrid model as 82% on balanced test set of 20 images. We specify the precision, recall and F-1 score by class along with macro and weighted averages in the table shown below:

Wilting Class	Precision	recall	f1-score	support
0	0.86	0.90	0.88	20
1	0.90	0.95	0.93	20
2	0.85	0.55	0.67	20
3	0.63	0.85	0.72	20
4	0.94	0.85	0.89	20
accuracy			0.82	100
macro avg	0.84	0.82	0.82	100
weighted avg	0.84	0.82	0.82	100

TABLE XI: Precision, recall and F-1 scores by class for Ensemble Model

References

[1] Mostafa Mehdipour Ghazi, Berrin Yanikoglu, and Erchan Aptoula. Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing*, 235:228–235, 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.