CAI 4104/6108: Machine Learning Engineering

Project Report: Plant Seedlings Classification

Deepak Palavarapu
(Point of Contact)
deepakpalavarapu@ufl.edu

Shashank Jagannthan
sjagannatham@ufl.edu

Jaya Chandra Kanth Reddy Cheemarla
cheemarla.j@ufl.edu

Charithesh Puppireddy
cpuppireddy@ufl.edu

Phalguna Peravali
phalgunaperavali@ufl.edu

April 27, 2024

1 Introduction

Accurate classification of plant seedlings from images is a challenging computer vision task with significant applications in agriculture, ecology, and conservation.

Our project aims to develop a robust model for identifying multiple plant species across growth stages by leveraging convolutional neural networks (CNNs) and transfer learning on a diverse dataset of seedling images. CNNs, known for their prowess in image recognition, form the core architecture for extracting discriminative features and classifying the seedling images. To enhance model performance, especially when data is limited or imbalanced across species, we employ transfer learning by leveraging features from pre-trained networks on large-scale image datasets.

The primary objectives are twofold: 1) Construct a CNN model capable of recognizing nuanced characteristics among plant species, and 2) Augment its efficacy through transfer learning, harnessing previously learned features from diverse image data. This approach is particularly advantageous when dealing with class imbalance or limited data for certain species. Accurate automated plant identification enables tailored agricultural practices, biodiversity monitoring, ecosystem assessments, and informed decision-making by stakeholders. Our project contributes to these efforts by providing a robust, data-driven solution for plant seedling classification, fostering sustainable practices and ecological resilience.

2 Approach: Dataset(s) & Pipeline(s)

2.1 Dataset

The "V2 Plant Seedlings Dataset" sourced from Kaggle forms the backbone of our project's data foundation. Comprising 5,539 images, it encompasses a rich diversity of 960 unique plants, meticulously categorized into 12 distinct species across various growth stages. Each image depicts a singular plant instance, accompanied by its corresponding species label, facilitating supervised learning tasks.

In the dataset's structure, a careful partitioning delineates training and testing subsets, adhering to an 80-20 split ratio. This segregation ensures that the model is trained on a substantial portion of the data while retaining a separate set for unbiased evaluation. Such a structured approach enhances the model's capacity to generalize beyond the training samples, essential for real-world applicability and performance assessment.

2.2 Data Handling

The machine learning pipeline unfolds through the following stages:

Data Inspection and Cleaning: The initial phase involves scrutinizing the dataset for any anomalies, such as missing or corrupted images, and ensuring consistency in labeling. Each image's species label and, if available, its scientific name undergo verification to ascertain accuracy and completeness.

Data Splitting: The dataset undergoes division into distinct training, validation, and test sets. This partitioning strategy enables the model to learn from a substantial portion of the data while also undergoing independent validation and testing on unseen images to gauge its ability to generalize.

Image Preprocessing: All images undergo standardization processes, including resizing to 224x224 pixels, a commonly utilized dimension for Convolutional Neural Network (CNN) inputs. Additionally, pixel normalization is applied to ensure that pixel values fall within the range of 0 to 1, thereby optimizing model training efficiency and stability.

2.3 Model Architecture - CNN

The model architecture selected for classifying plant seedlings is a Convolutional Neural Network (CNN), known for its effectiveness in image recognition tasks. The CNN architecture is structured as follows:

Input Layer: Accepts images resized to a uniform dimension (e.g., 70x70 pixels).
Convolutional Layers: Multiple layers with varying filter sizes and ReLU activation to extract features from the images.
Pooling Layers: Max pooling layers follow convolutional layers to reduce dimensionality and computational load.
Flattening Layer: Converts the 3D output of the last pooling layer into a 1D vector.
Dense Layers: One or more fully connected layers that use features extracted by convolutional layers to classify the images into various plant species. **Output Layer:** A softmax layer that outputs the probability distribution over the classes.

2.4 Model Architecture - CNN with Xception Transfer Learning

Model Architecture - CNN with Xception Transfer Learning [1] utilizes the Xception model, pre-trained on ImageNet[2], to effectively capture the nuanced features of plant seedlings.[5]

T-Configuration: Use the Xception model without the top layers (fully connected layers), as these are specific to the ImageNet classes.
Input: Ensure the input size matches what Xception expects, 240x240 pixels.
Freezing: Freeze the base layers during the initial training phase to prevent the pre-trained weights from being updated. This helps in leveraging the learned features without distortion.
Global Average Pooling 2D: Applied after the Xception output to reduce feature dimensions and condense the information into a form suitable for classification.
Dense Layer: A dense layer with a substantial number of units, such as 1024.Activation ReLU, to introduce non-linearity.
Dropout: Include dropout (e.g., rate of 0.5) to help reduce overfitting by randomly setting input units to zero during training.
Batch Normalization: Optionally add a batch normalization layer to stabilize and accelerate training.
Output Dense Layer: The final layer that classifies the images into their respective categories.Units: Equal to the number of plant species (12 for your dataset).Activation: Softmax, for producing a probability distribution across the different classes.

2.5 Training Strategy

The training strategy involves:

Optimizer: Adam optimizer for efficient stochastic gradient descent.
Loss Function: Categorical crossentropy, suitable for multi-class classification problems.
Metrics: Accuracy and F1 score to monitor training and validation performance.
Regularization: Techniques such as dropout to prevent overfitting.
Data Augmentation: To ensure the model generalizes well to new, unseen images.[4]

3 Evaluation Methodology

The project utilizes these metrics to evaluate the performance of the classification model:

Accuracy: Accuracy measures[3] the ratio of correctly predicted images to the total number of predictions made. While it provides a simple gauge of overall performance, it may not always accurately represent the model's predictive prowess, particularly in datasets with imbalanced classes.

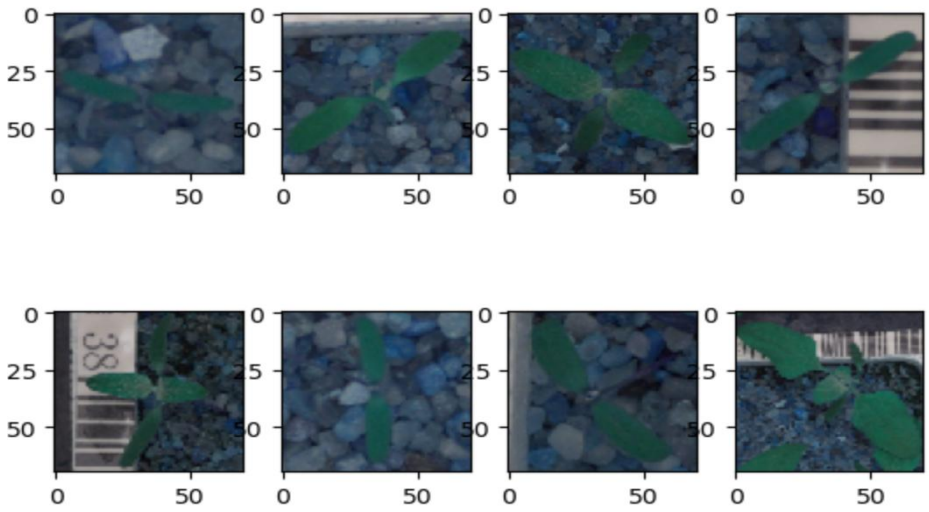

Figure 1: Sample Data

Macro F1 Score: The F1 score, a harmonic mean of precision and recall[3], offers a balanced assessment of a model's performance, particularly in classification tasks. It serves as a robust metric in evaluating classifiers, especially in scenarios where class imbalance exists, as it equally weighs both false positives and false negatives. A higher F1 score indicates better overall performance in terms of both precision and recall.

3.1 Model Evaluation Techniques

The evaluation of the model employs a mix of techniques to ensure comprehensive testing:

Training-Validation Split: In the model training phase, the dataset undergoes division into two subsets: the training set and the validation set. The training set is employed to train the model, while the validation set serves the purpose of assessing the model's performance on unseen data, thus averting overfitting and facilitating the tuning of hyperparameters.

Test Set Evaluation: Following the completion of training and validation, the model undergoes evaluation using an independent test set, distinct from those employed in prior phases. This pivotal step assesses the model's practical utility and verifies its ability to generalize effectively to novel, unseen data, ensuring its reliability in real-world scenarios.

Performance Visualization: Following the completion of training and validation, the model undergoes evaluation using an independent test set, distinct from those employed in prior phases. This pivotal step assesses the model's practical utility and verifies its ability to generalize effectively to novel, unseen data, ensuring its reliability in real-world scenarios.

3.2 Tools and Libraries

To implement and track these evaluations, various tools and libraries are used, including:

Scikit-learn for model evaluation metrics and cross-validation.

TensorFlow and Keras for building and training the models, including support for callbacks like ReduceLROnPlateau and ModelCheckpoint.

Matplotlib and Seaborn for generating plots that illustrate the model's training dynamics and performance.

4 Results

4.1 Accuracy

The model's accuracy was evaluated on a held-out test set, distinct from the training and validation sets. This approach helps assess the model's ability to generalize to unseen data, avoiding potential biases stemming from random data splits during the training and validation phases.

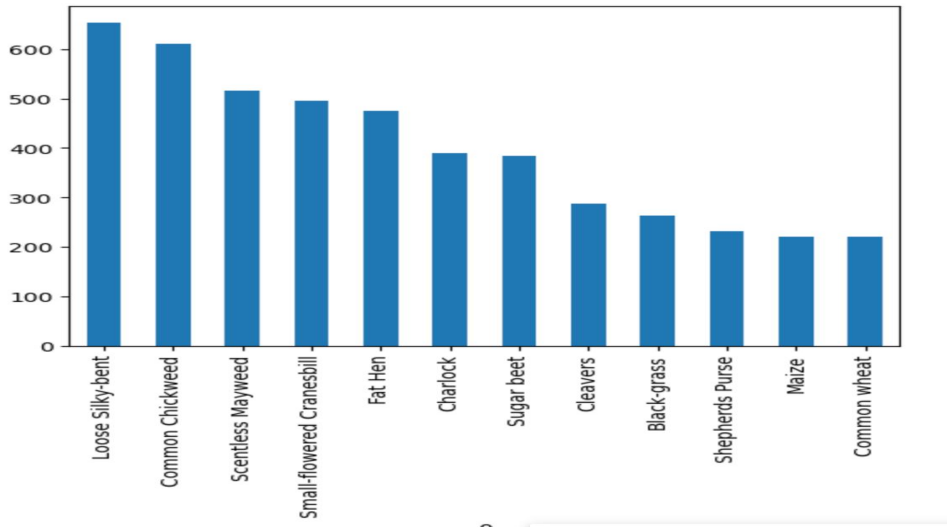

Figure 2: distribution of dataset labels

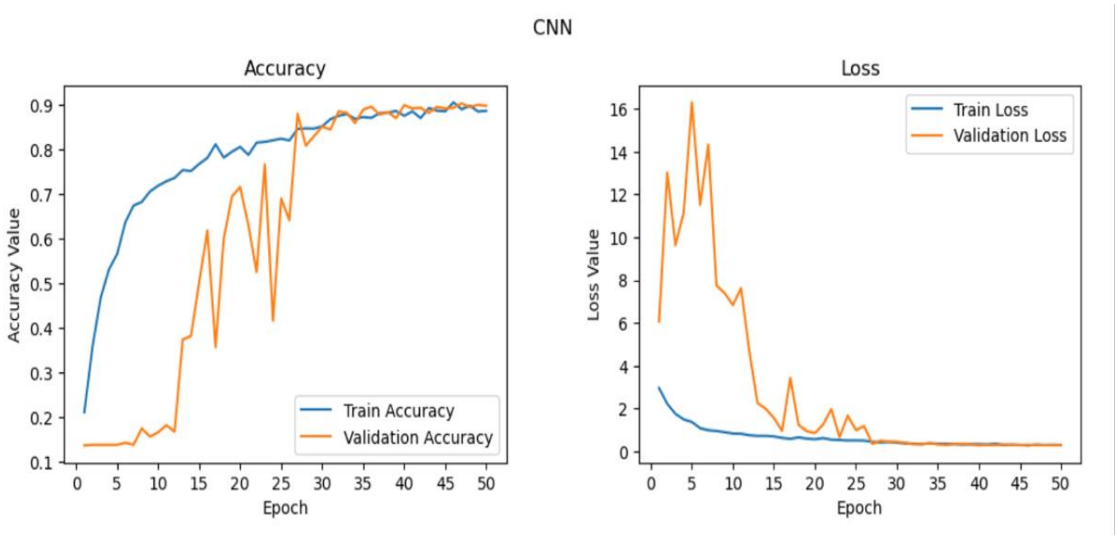

Figure 3: Basic CNN

```
[10]: import numpy
      from keras.models import Sequential
      from keras.layers import Dense
      from keras.layers import Dropout
      from keras.layers import Flatten
      from keras.layers import Conv2D
      from keras.layers import MaxPooling2D
      from keras.layers import BatchNormalization

      numpy.random.seed(seed)  # Fix seed

      model = Sequential()

      model.add(Conv2D(filters=64, kernel_size=(5, 5), input_shape=(ScaleTo, ScaleTo, 3), activation='relu'))
      model.add(BatchNormalization(axis=3))
      model.add(Conv2D(filters=64, kernel_size=(5, 5), activation='relu'))
      model.add(MaxPooling2D((2, 2)))
      model.add(BatchNormalization(axis=3))
      model.add(Dropout(0.1))

      model.add(Conv2D(filters=128, kernel_size=(5, 5), activation='relu'))
      model.add(BatchNormalization(axis=3))
      model.add(Conv2D(filters=128, kernel_size=(5, 5), activation='relu'))
      model.add(MaxPooling2D((2, 2)))
      model.add(BatchNormalization(axis=3))
      model.add(Dropout(0.1))

      model.add(Conv2D(filters=256, kernel_size=(5, 5), activation='relu'))
      model.add(BatchNormalization(axis=3))
      model.add(Conv2D(filters=256, kernel_size=(5, 5), activation='relu'))
      model.add(MaxPooling2D((2, 2)))
      model.add(BatchNormalization(axis=3))
      model.add(Dropout(0.1))

      model.add(Flatten())

      model.add(Dense(256, activation='relu'))
      model.add(BatchNormalization())
      model.add(Dropout(0.5))

      model.add(Dense(256, activation='relu'))
      model.add(BatchNormalization())
      model.add(Dropout(0.5))

      model.add(Dense(num_clases, activation='softmax'))

      model.summary()

      # compile model
      model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

      Would you like to receive official Jupyter news?
      Please read the privacy policy.
```

Figure 4: Basic CNN architecture

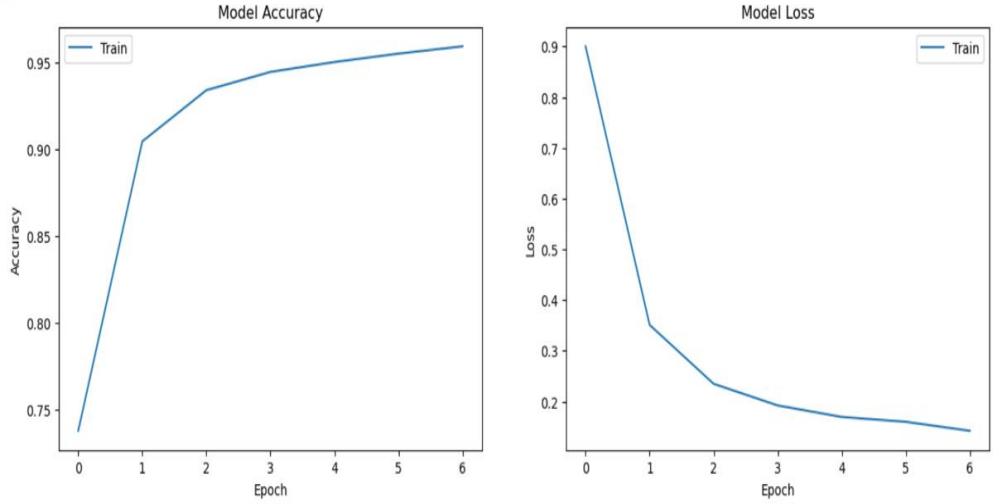

Figure 5: CNN with transfer learning

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 8, 8, 2048)        20861480

 global_average_pooling2d_3  (None, 2048)              0
  (GlobalAveragePooling2D)

 dense_9 (Dense)             (None, 100)               204900

 batch_normalization_18 (Ba  (None, 100)               400
 tchNormalization)

 dropout_3 (Dropout)         (None, 100)               0

 dense_10 (Dense)            (None, 50)                5050

 batch_normalization_19 (Ba  (None, 50)                200
 tchNormalization)

 dense_11 (Dense)            (None, 12)                612

=================================================================
Total params: 21072642 (80.39 MB)
Trainable params: 21017814 (80.18 MB)
Non-trainable params: 54828 (214.17 KB)
_________________________________________________________________
```

Figure 6: CNN transfer learning architecture

Baseline Model Accuracy: The baseline model accuracy that kaggle dataset provided was 88%

Achieved Model Accuracy: The final model, enhanced with transfer learning (Xception) and additional tuning, achieved an average accuracy of approximately 96.2% across the test sets, as shown in Table 1, indicating a robust performance against varied and unseen data.

4.2 Macro F1 Score

In datasets with imbalanced classes, the macro F1 score holds significant importance. It assigns equal importance to each class, thus ensuring that the model's performance accurately reflects its capability to effectively identify less common classes. Formula:

$$MacroF1 = \frac{1}{N}\sum_{i=1}^{N} F1_i \tag{1}$$

The model reached a macro F1 score of approximately 0.87, as shown in Table 2 demonstrating its competent handling of class imbalance and its ability to maintain precision and recall across classes.

Table 1: Accuracy

Model	**Accuracy**
CNN	88.5%
CNN with transfer learning	96.2%

Table 2: F1 score

Model	**F1-score**
CNN	0.87%
CNN with transfer learning	0.927%

5 Conclusions

The project successfully developed a robust plant seedling classification model leveraging Convolutional Neural Networks (CNNs) and transfer learning techniques. By harnessing the power of the Xception model, pre-trained on the ImageNet dataset, the model demonstrated remarkable performance, achieving an average accuracy of 96.2% across test sets [Table 1]. This significant improvement over the baseline accuracy of 88% underscores the efficacy of transfer learning in leveraging pre-existing knowledge to augment task-specific performance.

References

[1] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[3] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[4] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[5] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chenjie Yang, and Chunfang Liu. A survey on deep transfer learning. *International Conference on Artificial Neural Networks*, pages 270–279, 2018.