IBEAN PROJECT – CLASSIFYING HEALTHY AND DISEASED BEAN PLANTS USING DEEP LEARNING MODEL

Steve Andersen, Florian Bache, Eugene Lok, David Zarinski

University of Calgary

ABSTRACT

The report focuses on the development and evaluation of a deep learning model aimed at classifying healthy and diseased bean plants. The motivation for this project is to aid farmers in Uganda in detecting and preventing crop losses directly from their mobile device. The iBean dataset used for training and validation consists of pre-labelled leaf images representing three classes, healthy, angular leaf spot, and bean rust. Transfer learning using pre-trained models were utilized due to the limited size of the dataset. The report includes a detailed analysis and comparison of a variety of pre-trained deep learning models. The ShuffleNetV2_X2 was chosen due to its balance between inference time, model size, and high accuracy, making it an excellent choice for this use case. Overall, the results suggest that the selected model, which can be deployed to a mobile device with low computational power, could be an asset in aiding farmers in identifying diseased bean plants and increasing the crop yields.

***Index Terms*—** deep learning, classification, iBean dataset, transfer learning, crop yields, crop losses

1. INTRODUCTION

The world's human population receives over 50% of its calories from three crops: rice, wheat and maize. A major issue in growing these crops is that they require large amounts of water and fertilizer to grow [1]. The bean plant requires much less energy to grow and is a high source of protein and other nutrients for areas of the world with high population density and cases of undernutrition and hunger. While bean crops can be instrumental in easing world hunger in these areas, diseases that affect these plants can be devastating to the valuable crop. Two diseases which threaten these crops are Angular Leaf Spot and Bean Rust.

The motivation for this project is to build a machine learning model that is able to distinguish between diseases in the Bean plants. Beans are an important cereal food crop for Africa grown by many small-holder farmers - they are a significant source of proteins for school-age going children in East Africa.

The dataset is of leaf images representing three classes: the healthy class of images, and two disease classes including Angular Leaf Spot and Bean Rust.

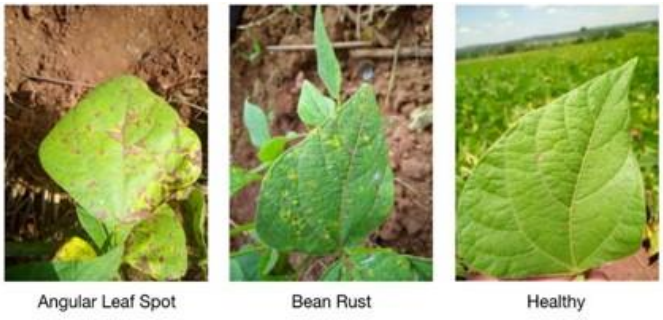

Fig. 1. Sample image from each class

The immediate goal is for the model to be able to distinguish between these 3 classes with high accuracy. The Blue-Sky goal is to build a model that can be deployed on a mobile device and used in the field by a farmer.[2] By using a mobile device with a deployed deep learning model, farmers are able to quickly and accurately detect plants that are diseased. This allows them to take appropriate measures to protect their crops and ultimately improve their yield and quality.

2. RELATED WORK

There is a significant body of work in using AI to detect diseases in plants and this work is currently being used to aid farmers recognizing and diagnosing these diseases in their crops. One such project is PlantVillage from Pennsylvania State University. PlantVillage consists of over 50,000 RGB images of 38 plant disease classes [3]. PlantVillage uses AI (among other technologies) to help millions of farmers increase their yield and profitability [4]. As a result of this project, there has been considerable interest in plant disease identification.

J *et al* used transfer learning techniques to accurately classify the 38 disease classes in the PlantVillage dataset. The

pretrained models used in the investigation were Densenet-121, ResNet-50, VGG-16, and Inception V4. The results of their investigation showed that the DenseNet-121 model was easiest to train given the smaller number of trainable parameters and so had a reduced computational complexity – this model achieved classification accuracy of 99.81% and an F1 score of 99.8% [3]. It is important to note that the Plantville dataset for this investigation did not include diseased bean plants per our investigation but was a compilation of numerous other crop plant diseases such as Tomato, Potato, Grape, and Corn plant diseases. While any positive results can be inferred to also apply to the bean plant study we are investigating, the model results in this investigation will be the true test of success on actual bean plant images from the iBean dataset.

In another study, Barburiceanu *et al* analyzed pre-trained CNN models for feature extraction of diseases affecting plant leaves followed by classification using a Support Vector Machine with RBF kernel. The pre-trained models in their investigation were ResNet18, AlexNet, VGG-16 and ResNet50. Two datasets were considered – a public one (Outex_TC_00013) and the PlantVillage dataset. For both datasets, the model with the best performance (time and classification score) was the pre-trained AlexNet model followed by SVM classifier (average accuracy = 99.86, feature extraction time = 321s). This architecture surpassed all others in classification performance and operating times, which was essential in keeping the model relevant for real-time processing tasks [5].

3. MATERIALS AND METHODS

3.1 Dataset

The dataset for this project consists of 3 classes of images – healthy bean plant, plant infected with Angular Leaf Spot, and plants infected with Bean Rust.

This dataset is of images taken in the field in different districts in Uganda by the Makerere AI lab in collaboration with the National Crops Resources Research Institute (NaCRRI), the national body in charge of research in agriculture in Uganda.

The images were then annotated by experts from NaCRRI who determined for each image which disease was manifested. The experts were part of the data collection team and images were annotated directly during the data collection process in the field [2].

The dataset consists of a total of 1296 images with a good balance across all 3 classes (428 – Healthy, 432 – Angular Bean Spot, 436 – Bean Rust).

These datasets were pre-split into training, validation and testing batches with a total of 1034, 133 and 128 images, respectively.

3.2 Methodology

Transfer learning was selected for the iBean project based on several factors. The primary goal of the project is to deploy the model on a mobile device. By utilizing transfer learning, we can reduce the computational resources required to train the model since we can leverage a pre-trained model instead of training one from scratch. Additionally, the limited dataset consisting of only 1296 images makes it difficult to achieve high accuracy with a newly trained model. Therefore, by using the weights of a pre-trained model, we can achieve better accuracy with less training data.

3.2.1 Data augmentation and dataset creation

For computational efficiency and faster training, as well as to ensure important image features such as color don't change too much, only RandomHorizontalFlip and RandomRotation were utilized as random transformations. For the training and validation datasets, we first augmented the data with our selected random transformations, followed by the same standard transformations (e.g., resizing, cropping, normalization, etc.) used in the training of the selected model. For the testing dataset, only the standard model transformations were applied.

3.2.2 Dataloaders

The datasets are used to create dataloaders with a specified batch size of 32. For the training dataloader, shuffling is enabled to ensure the data is randomly shuffled during each epoch. Shuffling the data during training helps the model learn more robust features and reduces the likelihood of overfitting.

3.2.3 Defining the model

The selected pre-trained model is loaded, and the weights initialized. To use it as a feature extractor, the final classifier layer of the model is then replaced with a linear classifier. This new layer is trained to classify the extracted features into one of the three classes defined in our dataset (i.e., angular_leaf_spot, bean_rust, or healthy).

3.2.4 Training

The model's training parameters are established as follows: utilizing the cross-entropy loss function and the Adam optimizer, both of which are well-suited for deep learning. Additionally, an exponential decay schedule is implemented to fine-tune the learning rate. This schedule gradually decreases the learning rate throughout each epoch to help prevent overfitting, which ultimately results in improved model performance. The training was done in two steps:

1. Classifier training: All parameters in the feature extractor layer were frozen by setting the requires_grad value for each to false. Then only the final classifier layer was trained using a learning rate of 0.005.
2. Finetuning the feature extractor: To finetune the entire model including the feature extractor for our specific task, all parameters were unfrozen by setting requires_grad to true. Then the model was trained using a much lower learning rate of 0.00005.

4. RESULTS AND DISCUSSION

As the goal of this project is to deploy the model to a mobile device to be used by a farmer, model accuracy, model size, inference time and energy efficiency are all important factors when deciding on the optimal model. The pre-trained models chosen for transfer learning based on ImageNet performance and characteristics are: ResNet (18, 34, 152) [7-9], EfficientNet (B1, B3, B5, V2_M, V2_S) [10-14], DenseNet-121 [15], and ShuffleNet_V2_X2_0 [16]. These models and their suitability as well as their performance regarding the project's use case will be discussed in detail in the various sections below.

4.1 Comparing models

Model	Parameters	GFLOPS	Validation Accuracy
ResNet-18	11,689,512	1.81	98.50%
ResNet-34	21,797,672	3.66	98.50%
ResNet-152	60,192,808	11.51	96.24%
EfficientNet B1	7,794,184	0.69	97.74%
EfficientNet B3	12,233,232	1.83	94.74%
EfficientNet B5	30,389,784	10.27	99.25%
EfficientNetV2-S	21,458,488	8.37	99.25%
EfficientNetV2-M	54,139,356	24.58	98.50%
Densenet-121	7,978,856	2.83	100.00%
ShuffleNetV2_X2	7,393,996	0.58	98.50%

Table 1: Comparing the number of parameters, GFLOPS, and validation accuracy of selected models [7-16].

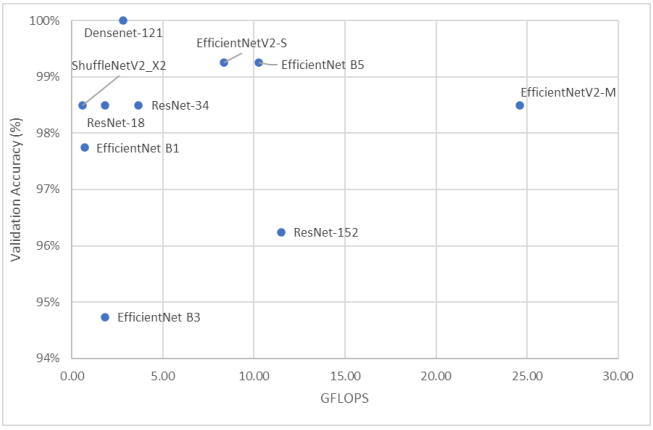

Fig. 2. Model validation accuracy vs. GFLOPS.

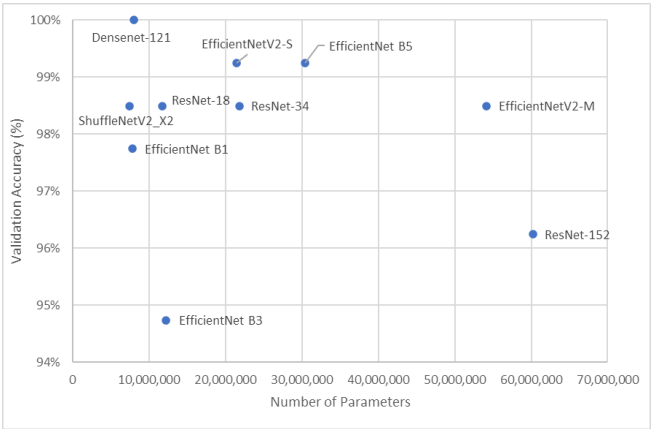

Fig. 3. Model validation accuracy vs. number of parameters.

4.2 Selecting a model

When selecting the best model for our use case there were several factors to consider. First, the classification accuracy of the model is extremely important to reliably classify each image. Secondly, to conform with the goal of deploying our model to a mobile device, one key factor in deciding model suitability is inference time and the model GFLOPS. Since the user will be using their mobile phone to classify bean plants in real time an almost instant response time is to be expected. Additionally, we would like the model to run well on any mobile device in order for that to not be a limiting factor on adoption for farmers in Africa.

4.2.1 Model inference

The figure below demonstrates that ShuffleNetV2 has a significantly faster inference time than several other commonly used pre-trained deep learning models. This is particularly important for the farmer, as it enables them to quickly classify a bean plant using even older devices, leading to improved efficiency and prompt measures to be taken for diseased plants.

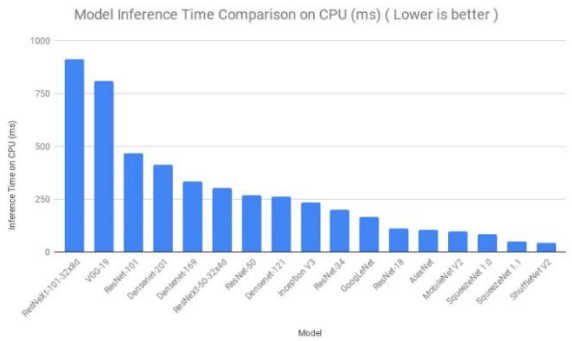

Fig. 4. Model inference time comparison [6]

4.2.2 Model size

As we aim to deploy our model on any mobile device, the size of the model is a crucial consideration when choosing the most suitable model for our use case. Given the limited space on mobile devices, it is essential to keep the model as lightweight as possible. Based on the figure below, ShuffleNetV2 stands out as one of the most space-efficient deep learning models available today, making it an excellent candidate for our application.

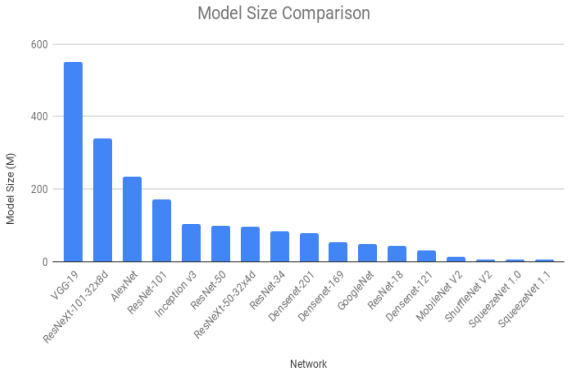

Fig. 5. Model size comparison [6]

4.2.3 Model Selection

After evaluating all models and considering all factors, ShuffleNetV2_X2 was chosen as the model for its lowest GFLOPS out of all the models tested (0.58), while still attaining a very high validation accuracy of 98.50%.

4.3 Evaluating selected model

The final model selected, ShuffleNetV2_X2, achieved an accuracy of 95.31% on the test data. Given the small model size and fast inference time, this is a good result with satisfactory accuracy for the use case.

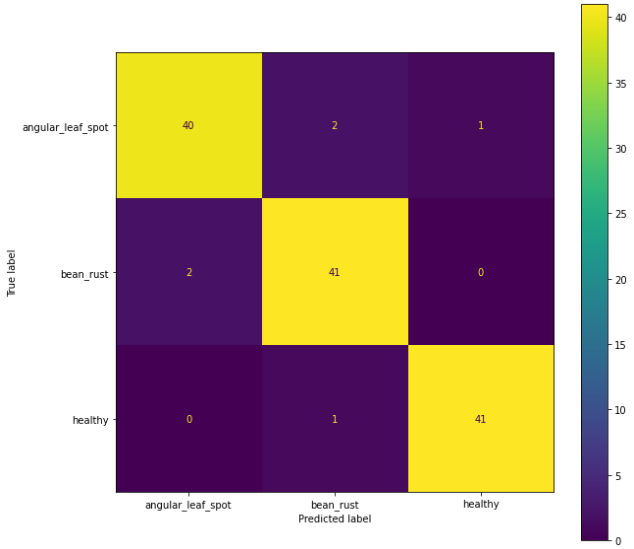

Fig. 6. Confusion matrix for the fine tuned ShuffleNetV2_X2 model on the test data.

	precision	recall	f1-score	support
angular_leaf_spot	0.95	0.93	0.94	43
bean_rust	0.93	0.95	0.94	43
healthy	0.98	0.98	0.98	42
accuracy			0.95	128
macro avg	0.95	0.95	0.95	128
weighted avg	0.95	0.95	0.95	128

Table 2: Classification report for the fine tuned ShuffleNetV2_X2 model on the test data.

4.4 Selected model drawbacks

The ShuffleNetV2_X2 model isn't quite as accurate as some of the larger more resource intensive model (e.g., EfficientNet B5, EfficientNetV2-S and Densenet-121). The main concern regarding misclassifications for this particular use case are false negatives. A false negative would occur if the model failed to identify a diseased plant as being diseased. Failure to detect diseased plants could lead to spreading of the disease to other healthy plants resulting in a reduced yield and quality of the crop.

5. CONCLUSIONS

In conclusion, this report focuses on the development and evaluation of a deep learning model that can classify healthy and diseased bean plants to aid farmers in detecting and preventing crop losses in Uganda. The iBean dataset was used for training and validation, consisting of pre-labelled leaf images representing three classes: angular_leaf_spot, bean_rust, or healthy. Transfer learning was utilized due to

the limited size of the dataset, and the ShuffleNetV2_X2 model was chosen due to its balance between inference time, model size, and high accuracy. The selected model achieved an accuracy of 95.31% on the test data, and can be deployed to a mobile device with low computational power. This can be an asset in aiding farmers in identifying diseased bean plants and increasing crop yields. This project adds to the significant body of work in using AI to detect diseases in plants.

6. GITHUB REPOSITORY

https://github.com/sandersen007/ENSF645_FinalProject

14. REFERENCES

[1] Radzman, N. (2022) *How forgotten beans could help fight malnutrition in Africa, The Conversation.* Available at: https://theconversation.com/how-forgotten-beans-could-help-fight-malnutrition-in-africa-176857 (Accessed: March 25, 2023).

[2] AI-Lab-Makerere (no date) *Ai-Lab-Makerere/ibean: Data Repo for the IBEAN project of the Air Lab., GitHub*. Available at: https://github.com/AI-Lab-Makerere/ibean (Accessed: March 25, 2023).

[3] J. A, Eunice J, Popescu DE, Chowdary MK, Hemanth J. Deep Learning-Based Leaf Disease Detection in Crops Using Images for Agricultural Applications. *Agronomy*. 2022; 12(10):2395. https://doi.org/10.3390/agronomy12102395

[4] PlantVillage. Available online: https://plantvillage.psu.edu/ (accessed on 28 March 2023).

[5] S. Barburiceanu, S. Meza, B. Orza, R. Malutan and R. Terebes, "Convolutional Neural Networks for Texture Feature Extraction. Applications to Leaf Disease Classification in Precision Agriculture," in IEEE Access, vol. 9, pp. 160085-160103, 2021, doi: 10.1109/ACCESS.2021.3131002.

[6] "Pre trained models for image classification - pytorch," LearnOpenCV, 02-Feb-2023. [Online]. Available: https://learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/. [Accessed: 02-Apr-2023].

[7] "Resnet18," resnet18 - Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet18.html. [Accessed: 02-Apr-2023].

[8] "Resnet34," resnet34 – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet34.html. [Accessed: 02-Apr-2023].

[9] "Resnet152," resnet152 – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet152.html. [Accessed: 02-Apr-2023].

[10] "EfficientNet B1," efficientnet_b1 – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.efficientnet_b1.html. [Accessed: 02-Apr-2023].

[11] "EfficientNet B3," efficientnet_b3 – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.efficientnet_b3.html. [Accessed: 02-Apr-2023].

[12] "EfficientNet B5," efficientnet_b5 – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.efficientnet_b5.html. [Accessed: 02-Apr-2023].

[13] "EfficientNetV2-S," efficientnet_v2_s – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.efficientnet_v2_s.html. [Accessed: 02-Apr-2023].

[14] "EfficientNetV2-M," efficientnet_v2_m – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.efficientnet_v2_m.html. [Accessed: 02-Apr-2023].

[15] "Densenet-121," densenet121 – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.densenet121.html. [Accessed: 02-Apr-2023].

[16] "ShuffleNetV2," shufflenet_v2_x2_0 – Torchvision 0.15 documentation. [Online]. Available: https://pytorch.org/vision/stable/models/generated/torchvision.models.shufflenet_v2_x2_0.html. [Accessed: 02-Apr-2023].