Cassava Leaf Disease Classification using Convolutional Neural Networks

Karan Saxena   Vishakha Gautam
University of Windsor

Abstract

Cassava is a staple food consumed as an alternative of rice in Sub-Saharan Africa. The farmers struggle with the cultivation of the Cassava crop as it is prone to viral disease which hinder its growth and thus cost the farmers a lot of time and money. Deep Learning can be used to help the farmers in cultivating healthy Cassava plants. The image dataset of the Cassava crop is used as input to state-of-the-art convolutional neural networks for performing classification of the disease type on the leaf. In our work, we have implemented ResNext, Inception v3 and Vision Transformer (ViT) on the cassava leaf image dataset. ResNext outperformed the other two CNN architectures by giving an accuracy of 87.56%.

Keywords – ResNext, ViT, Inception v3, Albumentations, Stratified K-folds Cross Validation

Introduction

Cassava is a key food security crop grown by farmers in Sub-Saharan Africa. Being a starchy root, it is used as a substitute of rice as it can withstand harsh conditions. There is a shortcoming to the growth of these starchy roots, and it is viral diseases. Cassava is highly prone to viral diseases which result in poor yield of the crop. These diseases can be detected using existing methods which require government funded experts to visualise and inspect the plants. This intensive labour performed on the crop for detection diseases results in low supply and high expenses. To reduce the amount of human labour and expenses in detection of diseases found in Cassava crops; we can use Data Science.

In our work, we have used the Cassava Leaf Disease Detection Image dataset provided by Makerere University AI Lab at Kampala (available on Kaggle). The task to be performed on this dataset is to classify each cassava leaf image into 4 disease categories or a fifth category indicating a healthy leaf.

We have used State of the Art techniques to tackle this problem. We have used Deep Learning to perform classification on the following dataset. Pre-trained Convolutional Neural Networks – ResNext, Inception v3, Vision Transformer (ViT) are used to classify the type of disease on the leaf. The performance of three models are compared.

Related Work

A lot of research has been carried out in the field of plant disease detection using deep learning. Applying deep learning for detection of diseases in plants has helped farmers reduce production costs and manual labour. In [1], the authors used the Cassava Leaf Disease Detection dataset and applied Convolutional Neural Network from scratch to the imbalanced dataset. In [2], the researchers have applied transfer learning to the dataset. They have used Inception v3 model to achieve their results. Deep learning has been applied to tomato plants dataset to detect diseases in [3]. The authors in [3] have applied SqueezeNet and AlexNet architectures to PlantVillage dataset. In [4], the authors have applied deep convolutional neural networks in real time to detect diseases on apple leaves. They have trained their dataset using concatenation of GoogleNet Inception structure and Rainbow. In our work, we implemented ResNext [12], Inception V3 [13] and ViT [14] on Cassava Leaf Disease Detection dataset. The CNN architectures were applied to classify the type of disease on the leaf.

Project Methodology

OpenCV (for pre-processing)
One of the open-source libraries that we used for the data at hand is OpenCV. It was written in C/C++ and is a cross-platform library. It is mainly used to perform real-time computer vision tasks. OpenCV has 2D and 3D toolkits; also, we can perform segmentation, recognition and object detection tasks using it. It can be used in many other applications such as robotics, augmented reality, and many others.

Albumentations
It is an image augmentation library. It is used for performing data augmentation as is it fast and flexible. It is written in Python and is widely used in Deep Learning research. It helps us in increasing the samples of our training dataset by providing us with different transformation techniques.

Stratified K-Fold Cross Validation
In our work, we have used Stratified K-fold Cross Validation instead of K-fold Cross Validation for splitting the datasets in train-test samples because K-fold uses Random Sampling whereas Stratified K-fold uses Stratified Sampling. Random Sampling cannot point out the exact accuracy as every time we change random state; the accuracy changes but Stratified K-fold ensures good accuracy as it represents the entire dataset in equal proportions.

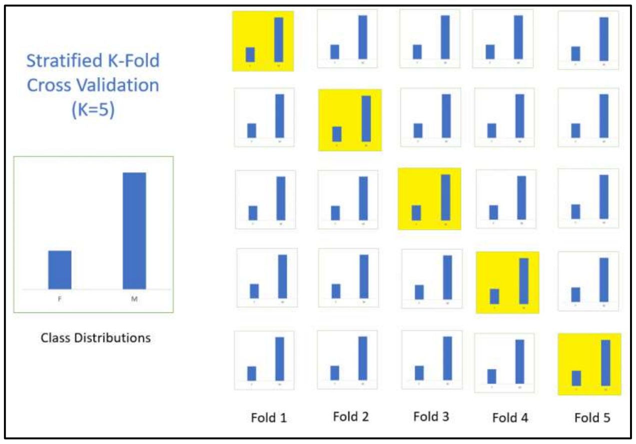


Figure 1: Representation of Stratified K-fold Cross Validation [7]

ResNext

Aggregated Residual Transformations for Deep Neural Networks (ResNext) is a variant to ResNet CNN architecture. It follows the split-transform-merge paradigm, and the outputs of different paths are merged by adding them together. Also, in ResNext, all paths share same topology. ResNext uses a new way of adjusting the model capacity by introducing a hyperparameter called – cardinality (the number of independent paths). We used ResNext in our work because the accuracy can be gained by more efficiently by increasing cardinality and we must tune just this one parameter in ResNext. We have used cardinality = 32 in our model.

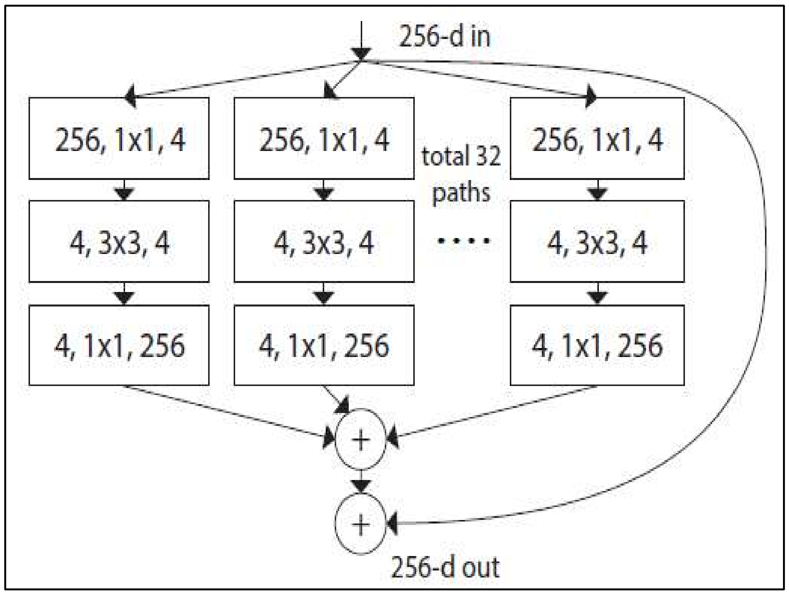


Figure 2: Building Block of ResNext with cardinality = 32 [8]

Inception V3

Inception V3's focus is on burning less computational power by improving on previous inception architectures (GoogLe Net/Inception v1). It makes several improvements over its previous architectures such as using Label Smoothing, factorized 7 *7 convolutions including the use of an auxiliary classifier to propagate label information lower down the network. It is widely used for tasks such as Image recognition and object detection tasks. The model's architecture consists of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batch normalization is used significantly throughout the model and applied to activation inputs. Softmax function is used for loss computation. An overview of the architecture of the model could be well explained by the figure 3.

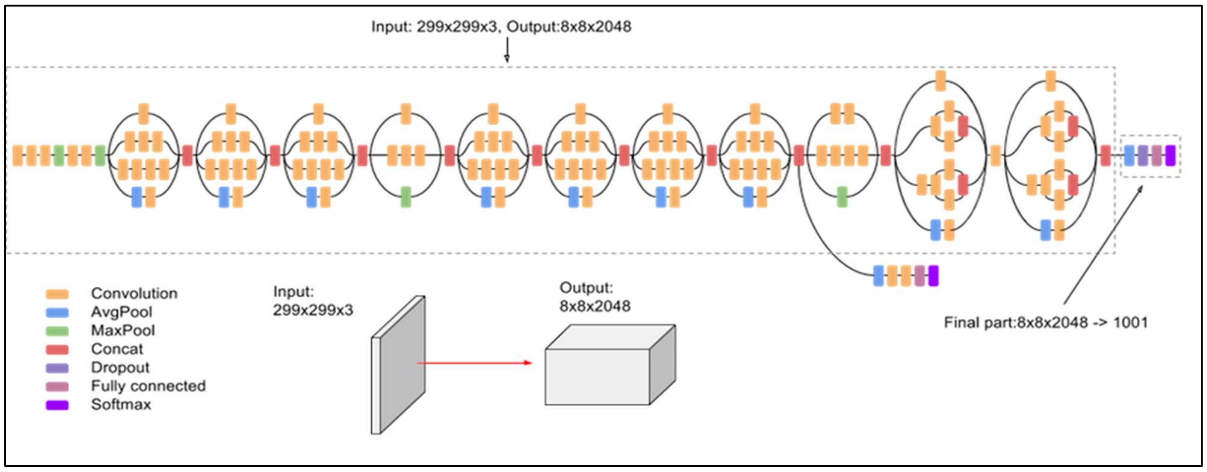


Figure 3: High Level Diagram of Inception V3 [10]

Vision Transformers

A vision transformer builds upon the idea of Transformer networks initially developed for NLP applications. Unlike words, an image contains many pixels each having important information about the image. Attending every pixel in an image poses a problem for the existing architecture.

Researchers break this into a series of steps [11] -

1. Split the image into patches
2. Normalize those patches
3. Produce lower-dimensional linear embeddings from the flattened patches
4. Add positional embeddings
5. Feed the sequence as an input to a standard transformer encoder
6. Pretrain the model with image labels.
7. Finetune on the downstream dataset for image classification

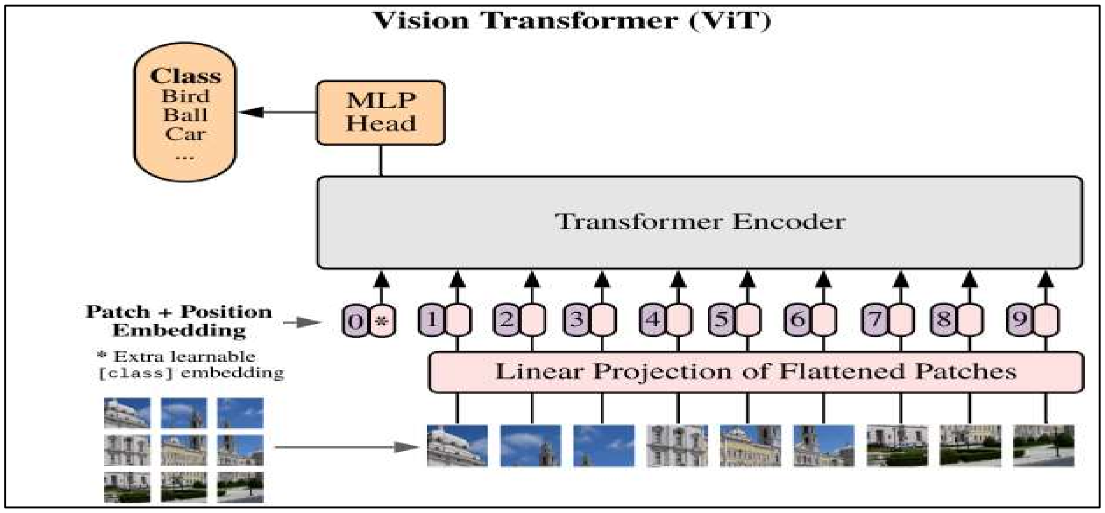

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

Figure 4: Architecture of Vision Transformer [9]

Experiments

Platform used - All the experimentation was performed on Kaggle platform. Kaggle provides free access to NVidia K80 GPUs in kernels. Kaggle's GPU was used to speed up the training process for our models.

Dataset used - *Cassava Leaf Disease Classification by Makerere AI Lab, Uganda*

Dataset contains the following files –

train.csv

- image_id - the image file name.
- label - the ID code for the disease.

sample_submission.csv - A properly formatted sample submission, given the disclosed test set content.

- image_id the image file name.
- label the predicted ID code for the disease.

[train/test]_tfrecords – It contains the image files in tfrecord format.

label_num_to_disease_map.json - The mapping between each disease code and the real disease name.

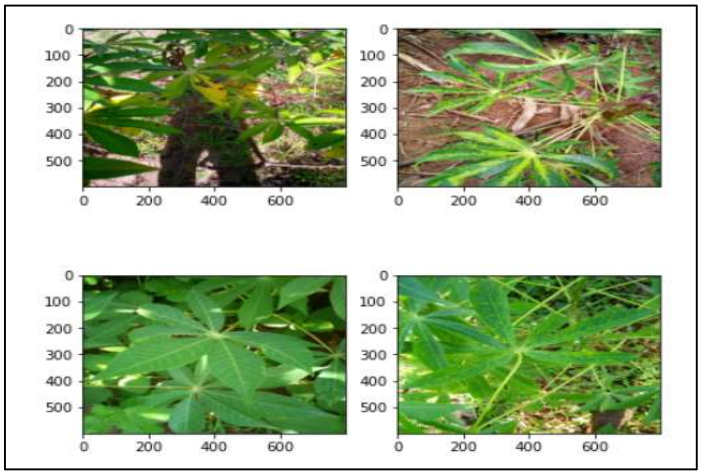
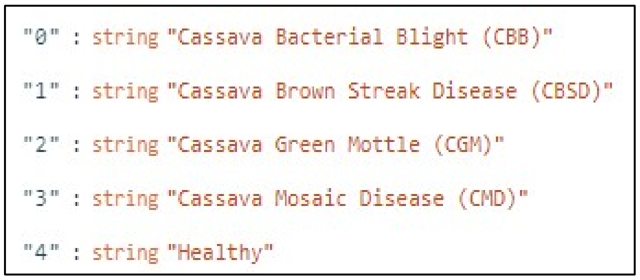


Figure 5: Sample Images from Cassava Leaf Dataset [6]



```
"0" : string "Cassava Bacterial Blight (CBB)"
"1" : string "Cassava Brown Streak Disease (CBSD)"
"2" : string "Cassava Green Mottle (CGM)"
"3" : string "Cassava Mosaic Disease (CMD)"
"4" : string "Healthy"
```

Figure 6: Label number to Disease Map [6]

Data Pre-processing - First, we read the images using imread in CV2 and then converted the BGR image format to RGB using cv2.COLOR_BGR2RGB. The conversion from BGR to RGB is required because we need to use both CV2 and pillow modules in Python. We perform the following step on both train and test datasets.

Next, we start with using Image Albumentations. Albumentations is a fast and flexible image augmentations library. We have used this library in our work to create more training samples for our dataset. Image Augmentation helps us increase the number of training samples by creating new samples from existing ones. We apply simple transformations like HorizontalFlip, Blur, Gamma, and many more to our existing dataset.

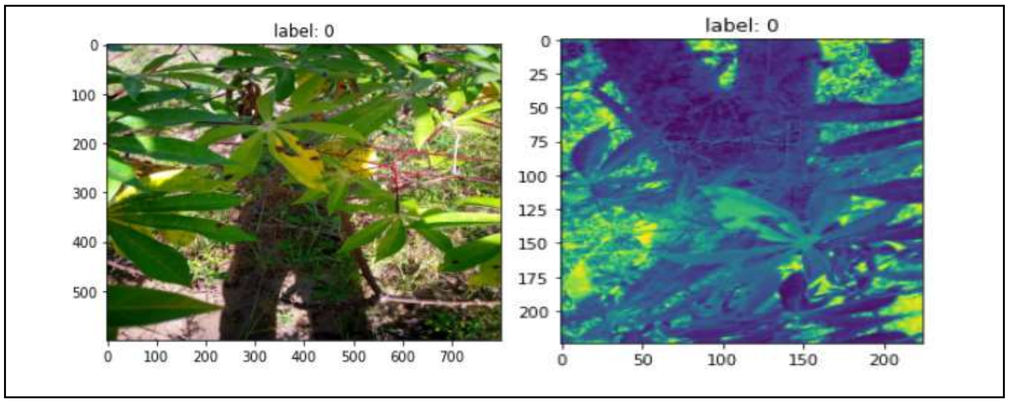


Figure 7: (a) Before Image Albumentations, (b) After Image Albumentations

We implemented the augmentations for both training time and validation time. For training time, we selected RandomResizedCrop, Transpose, Horizontal Flip, Vertical Flip, ShiftScaleRotate and Normalize. Each of the augmentation in Albumentations has a parameter named p that sets the probability of applying that augmentation to input data. To maintain consistency, we have selected 0.5 as the value for p which means that the augmentations will be applied to 50% of instances of the input data. For validation time, we performed only two augmentations to help keep the validation data as natural as it can get and those were, Resize and Normalization.

For each of the folds generated via Stratified K Folds, we have trained the three models, Inception v3, ResNext and Vision Transformer on Training Sets. Metrics - Accuracy, precision and recall scores were calculated on the validation sets.

Hyperparameter Tuning - A range of hyperparameters were tested during the tuning phase for all the 3 different models tested and a final selection was made for number of epochs, learning rate, batch size and weight decay rate. The values for the same were (10, 1e-4, 32, 1e-6) respectively. *CosineAnnealingWarmRestarts* was used as the scheduler. It anneals/decreases the initial learning rate set by us in a cosine manner until it hits a restart. After which, the learning rate is set back to the initial learning rate, and the cycle happens again. It is given by the following formula: -

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{T_{\mathrm{cur}}}{T_i}\pi\right)\right)$$

Cross Entropy Loss function used as the criterion function. It is very useful when training with C number of classes (5 in our case). The loss can be described as: -

$$\operatorname{loss}(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right)$$

Adam Optimizer algorithm was used as an Optimizer function which combines the advantages of two extensions of stochastic gradient descent, AdaGrad and RMSProp.

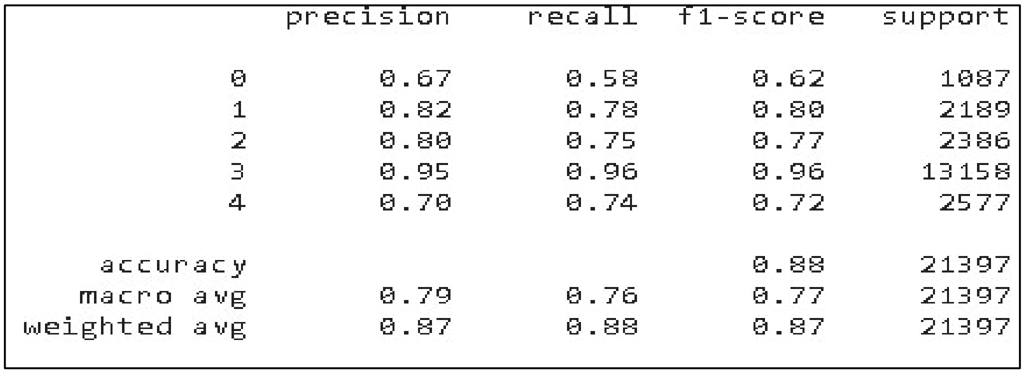
Results

Overall Accuracy, F1 score, Recall and Precision were calculated for all the three different CNN models tested on the dataset used in our work. The scores were obtained while using average='Macro' for all the different metric that we calculated. The 'Macro' parameter calculates the metrics for each label and finds their unweighted mean. This does not take label imbalance into account. The obtained results are shown in Table 1.

Score\Model	**ResNext (cardinality = 32)**	**Inception v3**	**Vision Transformer (ViT)**
Accuracy	0.8755900359863532	0.8587652474645978	0.7074356218161425
Precision	0.785965340327041	0.7613282023868682	0.5598185784621729
Recall	0.7648758066684127	0.7343160234744738	0.45069357131613846
F1 - Score	0.7744417732467053	0.7465700976842425	0.47924223106008307

Table 1: Results obtained after implementing the 3 CNN architectures on the dataset

The best scores for Accuracy, Precision, Recall, and F1- Score that we got were for the model - **ResNext50, 32x4d**. Classification Report and Confusion Matrix for the model with the best scores were also generated and are as per the figure 8 and 9.



```
              precision    recall  f1-score   support

           0       0.67      0.58      0.62      1087
           1       0.82      0.78      0.80      2189
           2       0.80      0.75      0.77      2386
           3       0.95      0.96      0.96     13158
           4       0.70      0.74      0.72      2577

    accuracy                           0.88     21397
   macro avg       0.79      0.76      0.77     21397
weighted avg       0.87      0.88      0.87     21397
```

Figure 8: Classification Report for ResNext

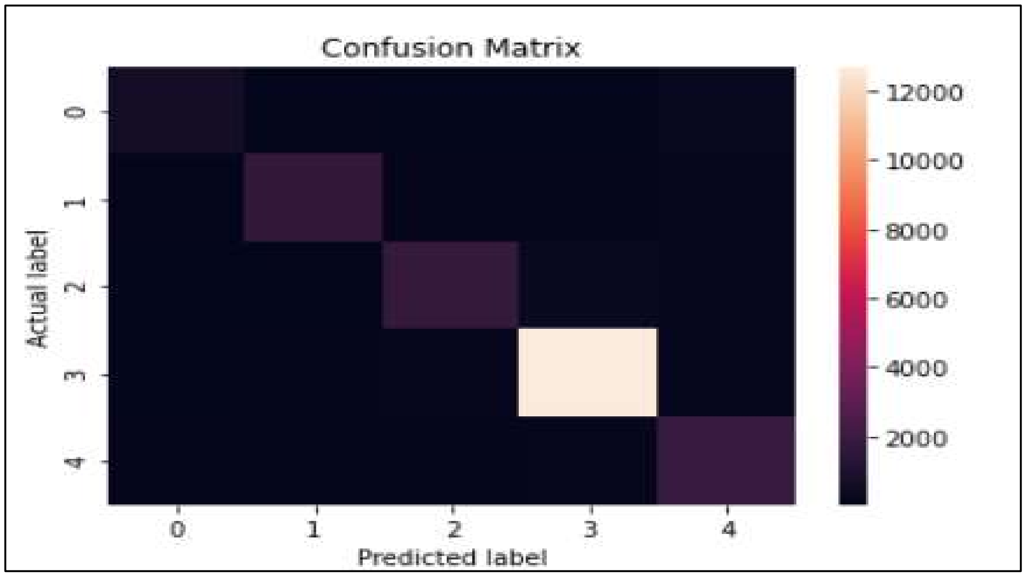


Figure 9: Heatmap for Confusion matrix of ResNext

Conclusion

The task in hand was to identify the type of disease present on a Casssava Leaf using image dataset provided on Kaggle [6]. At least 80% of household farmers in Sub-Saharan Africa grow this plant which is prone to several diseases. We started our work to detect a specific type of disease by using Convolutional Neural Networks such as ResNext, Inception V3, and Vision Transformer. As the part of image pre-processing, we performed several Image Augmentations which ensured that different variations of the images present in the dataset are considered. A variation in tuning hyperparameters were performed and eventually a selection was made that was utilized for training of different models. We were able to achieve state-of-the-art results of 87.55%, 85.87%, and 70.74% accuracies for ResNext, Inception v3, and Vision Transformer, respectively. This was observed that the model architecture for ResNext achieved the best scores out of the three different models tested indicating that ResNext is a better performer when it comes to classification of diseases on Cassava leaves.

Future Work

In near future, we will try to improve the task accuracy by applying an ensemble of various convolutional neural network architectures. Another shortcoming that we faced during the implementation of this project is the limited amount of time that was allotted to us for GPU usage. This resulted in using GPU in judicious manner and we were unable to test various hyperparameters/model architectures which may have helped us improve the accuracy. In future, we will try to use in-house GPU and experiment with even more combinations of model architectures and hyperparameters.

The implementation of our work can be found here -
https://github.com/karan96/master/blob/master/cassava-leaf-detection-v.ipynb

References

[1] Sambasivam, G., & Opiyo, G. D. (2020). A predictive machine learning application in agriculture: Cassava disease detection and classification with imbalanced dataset using convolutional neural networks. Egyptian Informatics Journal. doi: 10.1016/j.eij.2020.02.007

[2] Ramcharan Amanda, Baranowski Kelsee, McCloskey Peter, Ahmed Babuali, Legg James, Hughes David P., Deep Learning for Image-Based Cassava Disease Detection, Frontiers in Plant Science, Volume 8 (2017), doi: 10.3389/fpls.2017.01852, 1664-462X

[3] H. Durmuş, E. O. Güneş and M. Kırcı, "Disease detection on the leaves of the tomato plants by using deep learning," *2017 6th International Conference on Agro-Geoinformatics*, Fairfax, VA, USA, 2017, pp. 1-5, doi: 10.1109/Agro-Geoinformatics.2017.8047016.

[4] P. Jiang, Y. Chen, B. Liu, D. He and C. Liang, "Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks," in IEEE Access, vol. 7, pp. 59069-59080, 2019, doi: 10.1109/ACCESS.2019.2914929.

[5] https://miro.medium.com/max/7546/1*Za3VLUEHu7JhiLRWO3Lv_A.jpeg

[6] https://www.kaggle.com/c/cassava-leaf-disease-classification

[7] https://media-exp1.licdn.com/dms/image/C5112AQHFGJr78f9CUA/article-cover_image-shrink_600_2000/0/1568978840004?e=1623888000&v=beta&t=GlIObJem4PltOmsBmOoeimP2QrhlDYctYfkVOXjGBfY

[8] https://miro.medium.com/max/1468/1*LOoc11tkDoqv0pC6OH7mwA.png

[9] https://paperswithcode.com/method/vision-transformer#

[10] https://cloud.google.com/tpu/docs/images/inceptionv3onc--oview.png

[11] https://theaisummer.com/vision-transformer/

[12] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1492-1500).

[13] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[14] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.