Identify the Type of Disease from an Image of Cassava Leaves with Deep Learning Methods

Hanlin Tang htang79@wisc.edu
Yuechuan Chen ychen959@wisc.edu
Hua Tong htong24@wisc.edu

Abstract

With the development of Artificial Intelligence, deep learning has become a mature technology which plays a pivotal role in various areas and saves tons of human effort. For this project, we planned to construct deep learning models to explore the use of image recognition in cassava disease detection. The data set we selected from Kaggle contains more than 20000 images of Cassava Leaves with 4 kinds of different diseases. We aimed to identify the type of diseases to help farmers deal with them timely and correctly. To solve the problem, we experimented 6 Convolutional Neural Network (CNN) models with different architectures and then evaluated their performances.

1. Introduction

1.1. Background

One of the most important uses of deep learning is image recognition, which has already become a widely implemented technique in our daily life. For example in parking lots, cameras automate the management process. Face recognition is also a prevalent and efficient technique, which helped the police arrest lots of criminals in recent years. In many financial scenarios, your face might be used for identification. Thanks to image recognition, our life has become safer and more convenient.

For this project, we plan to explore a use of image recognition in agriculture: to detect the disease of Cassava, which is a key food in Africa and grown by at least 80% of household farms in Sub-Saharan area, based on the image of its leaves. The four main kinds of diseases are: Cassava Bacterial Blight (CBB), Cassava Brown Streak Disease (CBSD), Cassava Green Mottle (CGM) and Cassava Mosaic Disease (CMD). Given the previous study, the appearances of cassava leaves of these diseases are slightly different. Therefore, it will be helpful and also convenient to distinguish plants with these diseases and healthy ones by the pictures of leaves. In this way, farmers could take appropriate measures if it is necessary.

1.2. Motivation

Traditional methods of detecting these diseases require experts' involvement. However, experts are not accessible in many area. Meanwhile, the sooner farmer can detect it the better, because virus can spread out and cause more damage if not.

In a higher level, we believe that it is fundamentally important to stop the global hunger in order to have a better, prosperous, and peaceful world. This research may be helpful to achieve this goal, especially for some area where agriculture professions are not accessible to diagnose the disease.

Also, cassava is a typical case of plant diseases. By conducting this project, we can gather experience and get familiar with this kind of problems. We can extend our study to some other plants if needed. Moreover, this approach might also be useful to animals, and even people, especially for some visible skin diseases.

Finally, this problem seems to be well defined and has some typical computer vision elements. We hope that this project could be a beginning of our study in deep learning. We are looking forward to learning more and doing more advanced projects in the future.

2. Related Work

Our project is mainly about the use of deep learning in agriculture and botany. According to the statement in [4], sixteen areas have been identified in total, including identification of weeds, land cover classification, plant recognition, fruits counting, crop type classification, etc. Many articles about agricultural application of deep learning has been published in recent years.

In 2017, Patrick Kinyua Gikunda and Nicolas Jouandeau published an article about CNNs for smart farms [2], in which they present an initial understanding of state-of-the-art CNNs and their underlying complexities and a comprehensive review of research dedicated to applications of state-of-the-art CNNs in agricultural production systems.

From a work by the team led by Trung-Tin Tran [8], we understand the use of CNNs in forecasting deficiencies

of tomato plants. The predictive performance of the three models in this paper are validated, with the accuracy rates of 87.273% and 79.091% for Inception-ResNet v2 and Autoencoder, respectively, and with 91% validity using Ensemble Averaging.

In another paper [11], research on deep learning applications in agriculture is summarized and analyzed, and future opportunities are discussed precisely. The concept of smart agriculture is explained clearly.

From a paper published last year [1], researchers presented a quantitative overview of papers published in the past two decades, thematically related to machine learning, neural networks, and deep learning. Also, they analyzed the current trends of this area.

3. Proposed Method

3.1. Basic

3.1.1 Neuron

In general, a neuron in Neural Network is whch take weighted input z, apply an activation function σ to it, and output a value $f(\sigma)$.

3.1.2 Neural Network

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

3.1.3 Dense

Also called "fully connected layer". It is a layer where each cell in it connect to every neuron from the previous layer. It is the most traditional way of neural network.

3.1.4 Convolutional

In mathematics, convolution is a mathematical operation on two functions (f and g) that produces a third function that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it.
In deep learning, a convolutional neural network Convolutional networks are a specialized type of neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

3.1.5 Pooling

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters. Global pooling acts on all the neurons of the feature map.There are two common types of pooling in popular use: max and average. Max pooling uses the maximum value of each local cluster of neurons in the feature map,while average pooling takes the average value.

3.2. Architectures

Since CNN is a fast-developing and well-developed field, with the idea of not to "reinvent the wheel", we pick some common CNN architectures as our proposed models.

3.2.1 AlexNet

AlexNet [5] is an relatively early CNN architecture, which is commented as "a milestone in making deep learning more widely-applicable" [10]. So even there are many powerful architectures come after it today, we still like to introduce it and make it a baseline for our experiments. The structure of AlexNet can be summarize into Figure 1, which we will not go deep into.

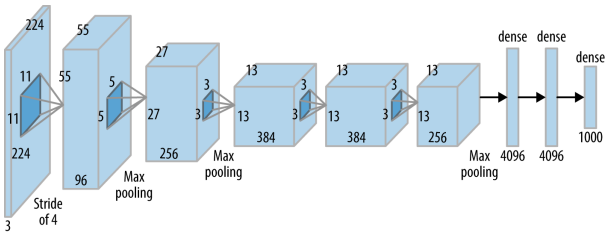


Figure 1. Alexnet Block Diagram (source:oreilly.com)

3.2.2 VGGNet

VGGNet is a famous CNN network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [7]. There are multiple variants of VGGNet (VGG16, VGG19, etc.). In this project we use VGG16 as a representative.

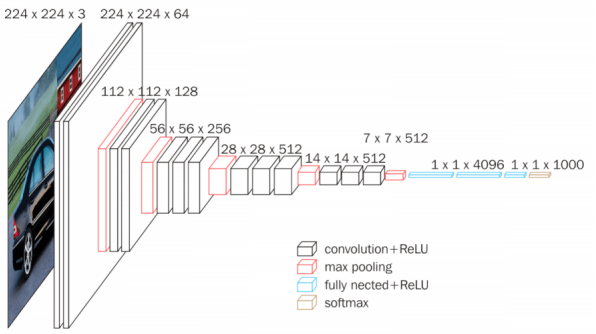


Figure 2. VGG16 Block Diagram (source: neurohive.io)

3.2.3 ResNet

Resnet, or called Residual Network, was purposed in 2015 [3]. The key structure of resnet is the residual block: by adding a so-called "identity shortcut connection", it was made possible to have much deeper network. There are a lot of variants of residual blocks, whose main differences are the order of activation and batch-norm functions.

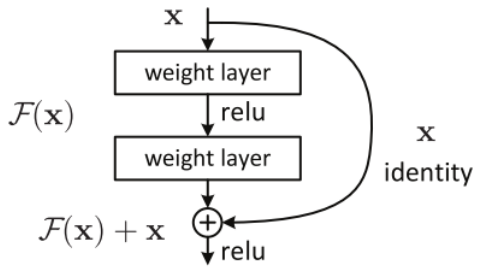
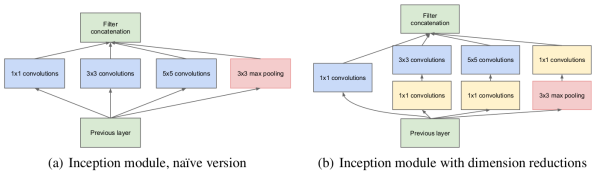


Figure 3. Residual block

3.2.4 GoogLeNet

GoogLeNet was based on a deep convolutional neural network architecture codenamed "Inception", which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 4. Inception structure (https://www.geeksforgeeks.org)

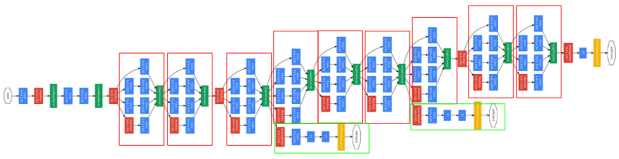


Figure 5. GoogleNet structure

3.2.5 ResNeXt

ResNeXt [6] is a homogeneous neural network which reduces the number of hyperparameters required by conventional ResNet. This is achieved by their use of "cardinality", an additional dimension on top of the width and depth of ResNet (as shown in Figure 6). Cardinality defines the size of the set of transformations.

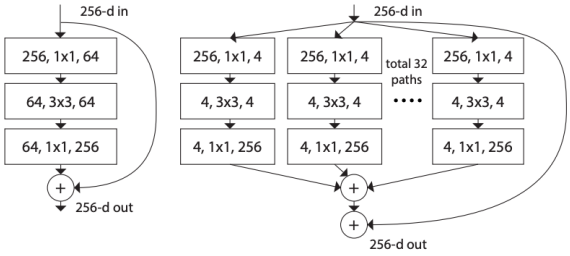


Figure 6. ResNeXt structure [6]

3.2.6 Wide ResNet

In Wide ResNets [9], plenty of parameters are tested such as the design of the ResNet block, how deep (deepening factor l) and how wide (widening factor k) within the ResNet block. When $k = 1$, it has the same width of ResNet. While $k > 1$, it is k times wider than ResNet (Figure 7).

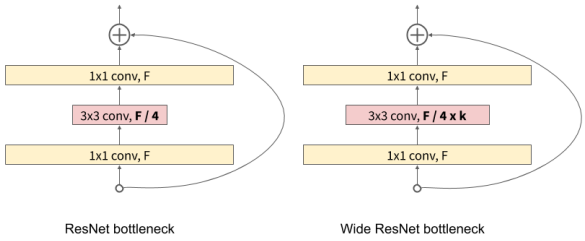


Figure 7. Wide Resnet structure (hemingwang.blogspot.com)

3.3. Optimizing

3.3.1 Cross Entropy

Cross entropy is a commonly used loss function for classification problem, the formula is following (y_i is coded with one-hot):

$$H(y, \hat{y}) = -\sum_{i=1}^{n}\sum_{j=1}^{p} y_{ij}\log(\hat{y}_{ij}) \qquad (1)$$

3.3.2 SGD

SGD, fully expressed as Stochastic gradient descent, originally referred to the optimizing strategy that update parameters for each single sample towards its gradient descent direction. Since the direction for a single sample could be very different from the direction of the whole data set, there is this "stochastic" introduced.

Nowadays, SGD is more often referred to the mini-batch gradient descent, which take a mini-batch instead of single sample. This decrease the randomness to be more stable (and yet still have randomness) and improved efficiency by allowing paralyzing.

$$\theta_{t+1} = \theta_t - \alpha g_t \tag{2}$$

3.3.3 Adam

There are a lot of optimizing methods trying to have the learning rate be adaptive. Adam, Adaptive Moment Estimation, is one of the most populated one. The authors describe Adam as combining the advantages of AdaGrad and RMSProp.

$$\begin{gathered} \theta_{t+1} = \theta_t - \frac{1}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \\ m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1 - \beta_2) \times diag(g_t^2) \\ \hat{m}_t = \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \end{gathered} \tag{3}$$

3.3.4 Early Stopping

Early stopping, as the name stating, is to stop training earlier than our pre-determined epoch number. There are many criteria for doing early stopping, while most of them are based on the validation accuracy.

Here we design such an early stopping strategy: stop if the max val_acc does not increase for 6 epochs, or does not increase for 2 epochs while train_acc is larger than 99%.

The settings and magic numbers come from our pilot experiments. We found that in this question, 6 epochs is long enough to have some significant improvement, if improvement is possible at all. Actually in our pilot experiments, a training with good learning rate gets to a good solution within 5 or 10 epochs. Also, in our pilot experiments we found that many architectures we mentioned can easily get a 100% training accuracy and close to 0 loss, which we think would unlikely to improve further as it arrived an local minimum. Thus in that case we set the threshold smaller.

4. Experiments

4.1. Dataset

As we mentioned, our dataset is on the Kaggle. In the training set which we have access to, there are 21397 labeled photos, each of which is 800x600 in shape.

Diseases	Labels	Counts
CBB	0	1087
CBSD	1	2189
CGM	2	2386
CMD	3	13158
Healthy	4	2577
Total		21397

Table 1. Data Exploration

For our following experiments. We divided our dataset into three parts: a training set with 80% of data (17118), a validation set with 5% of data (1070), and a test set with 15% of data (3209). The local validation set is used for single model training, while the local test set is used for our model-wise comparison and super-parameter tuning.

Furthermore, in our pilot experiments, resize the original photo into a half (or a quarter in area): 400x300 doesn't significantly affect accuracy, but increase speed a lot. Thus in the following, unless special noticed, we are using 400x300 photo. And for pretrained model, we normalized the data with mean [0.485, 0.456, 0.406] and std [0.229, 0.224, 0.225] as saying on the PyTorch website.

Although we cannot see, It was written on Kaggle that "Expect to see roughly 15,000 images in the test set.", which we will use for our final evaluation of the models we decide to propose.

4.2. Software & Hardware

We used the pytorch on python, including it's embedded models. We also use overleaf for our main method to collaborate writing reports.

We have three windows environments, and following video cards: GTX 1650, RTX 2080, RTX 2080 Ti.

4.3. Some Settings

While using the early stopping strategy we mentioned, we still set that there would be at most 40 epochs.

The batch size is different against different models and video card. The general rule we follow is to try to use the most CUDA memory without crashing.

5. Results and Discussion

5.1. Training Loss & Model Accuracy

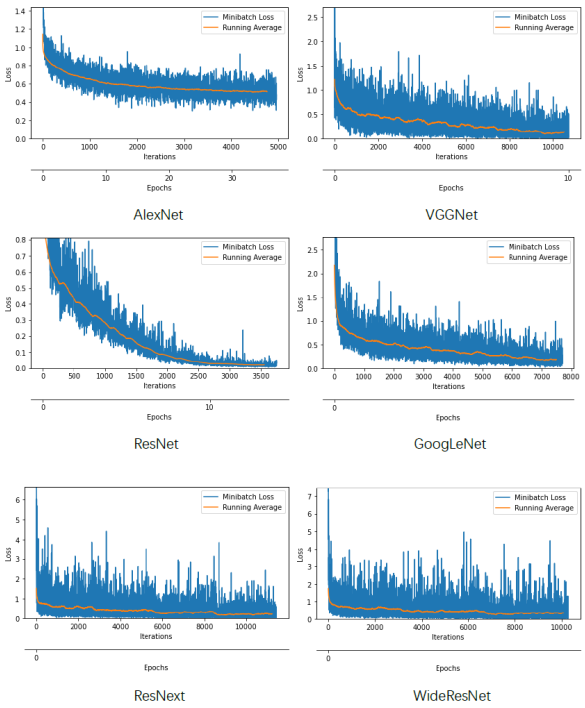


Figure 8. Training Loss

As we can see in figure 8, the training losses of AlexNet and ResNet have significant downward trend before the 10th epoch, while in VGG16 and GoogLeNet architectures they decrease rapidly in the first few epochs and then stabilized. As for models ResNeXt and Wide ResNet, the running averages drop even faster, as they reach very low levels in the first epoch.

Figure 9 shows the tendencies of training and validation accuracy of each architecture. The training accuracy of AlexNet increases from 70% to 83% during training, which is much lower than other models. As for VGG16, ResNet and GoogLeNet, The training accuracy starts from about 80% and ends over 98%. And the plot shows that ResNeXt and Wide ResNet reach about 90% training accuracy after the first epoch. The validation accuracy for all architectures varies from 82% to 88%.

The results indicate the initial learning efficiency in VGG, GoogLeNet, ResNext and WideResNet models are very high, thus the improvement of each epoch would be relatively limited later on. While the learning efficiency of model AlexNet and ResNet appear to be quite stable, so they converge a bit slower.

We tried to keep all parameters the same in order to make comparisons. However, setting a large batchsize may cause GPU memory explosion. Due to the limitation of different hardwares, we have to set different batchsize to train different models, which may affect the results and our analysis.

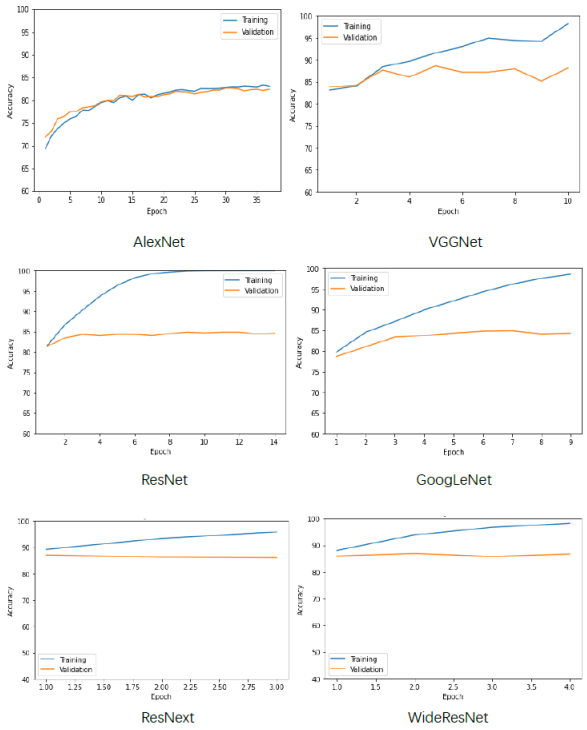


Figure 9. Training and Validation Accuracy

5.2. Confusion Matrix & Model Comparison

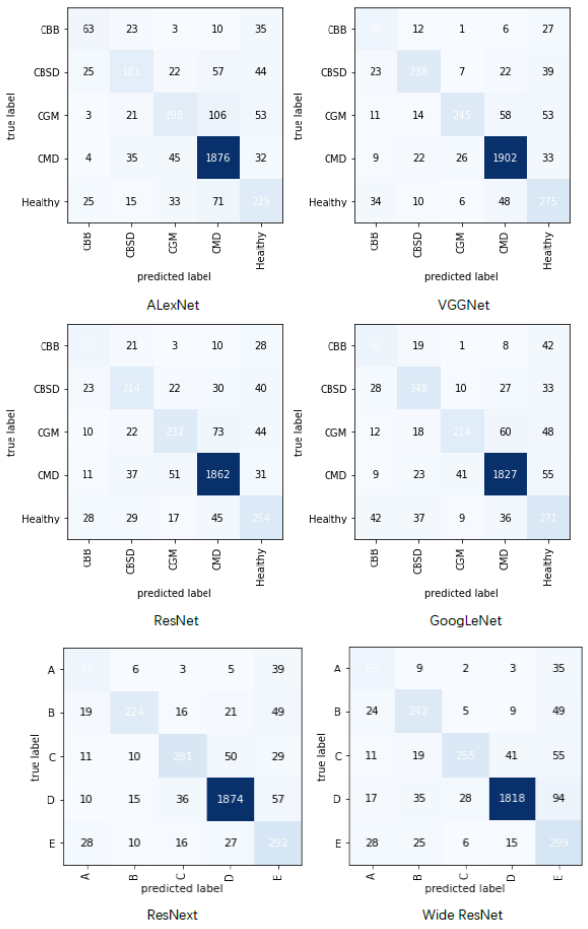


Figure 10. Confusion Matrices

Method	Train acc	Val acc	Test acc	Score
AlexNet	83.07 %	82.43 %	79.37 %	78.21%
VGG16	98.18 %	88.13 %	85.63 %	84.72%
Resnet34	99.99 %	84.58 %	82.08 %	86.75%
GoogLeNet	98.59 %	84.30 %	82.61 %	82.48%
ResNext	98.26 %	86.73 %	85.76 %	84.37%
Wide Resnet	95.86 %	86.17 %	84.11 %	83.78%

Table 2. Model comparison

The table above shows the results for different architectures, which contains 4 accuracy measures (Score refers to the public score of the model on Kaggle). Figure 10 are confusion matrices of the models. According to these numbers, we have the following findings:

1. AlexNet has the worst performance of all models;
2. ResNet reaches the highest training accuracy (almost 100%), and has the highest public score on Kaggle;
3. Test accuracy for most of the models are around 85%, which means their predictive power is considerable;
4. In the light of the confusion matrices, the predictive power for different labels are quite balanced;
5. All of our models have different levels of overfitting, since the training accuracy is much higher than testing.

5.3. Fine-tuning

We tried to fine-tuned the ResNet for better performance. First of all, we tried on ResNet50 instead of ResNet34, which was a sacrifice towards efficiency. We also manually replaced the last fully connected layer which was (2048,1000) to (2048,5) to better fit our problem. As for the input, we use (512x512) instead of (400x300), which was also a sacrifice towards efficiency.

We still use pre-trained model from pytorch, but this time we run each epoch and check if we need lower the learning rate. After around 6 epochs, there seems to get over-fitting, and we stopped.

The final model outperformed all the models we mentioned above, which proved the importance of fine-tuning. It reached 87.19 % accuracy on local test data set, and 86.75 % on online test data set.

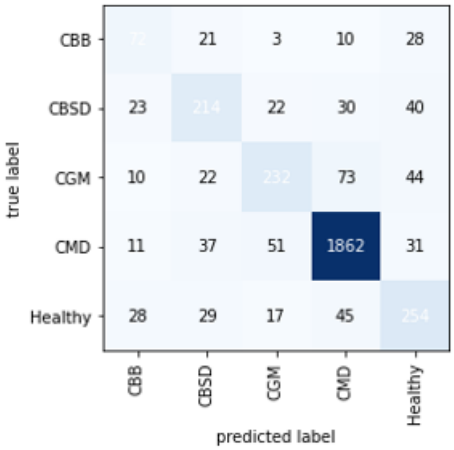


Figure 11. Confusion matrix of ResNet50 with fine-tuning

6. Conclusions

As a result, we managed to achieve the basic target we set in the project proposal (xxx achieved xx percent of accuracy on the final test set, which is larger than the 85 % baseline we set.)

However, we didn't go further and weren't even close to the higher targets we set (89 % and 90 %). This is most likely due to lack of sufficient fine-tuning. Yet, due to the relatively large data set and image size, each model training take from 2 hours to half day, which is extremely expensive

given that we had many candidates models and little fine-tuning experience. We did have some trails on fine-tuning but didn't go well.

Referred to Kaggle, there surely is much space to improve for many models we mentioned by tuning, which will be the future directions.

7. Acknowledgements

Thank professor Sebastian Raschka for teaching this course and provided a lot of useful codes, which is the main reference of our project.

Thank experts and collaborators from National Crops Resources Research Institute (NaCRRI) for assisting in preparing this dataset. Thank Kaggle and Makerere Artificial Intelligence (AI) Lab to held it online.

Thank pytorch for providing a easy frame as well as good implemented common models which saved use tons of coding work.

8. Contributions

Each of us pick several models from torchvision.models to experiment with: Hanlin Tang took AlexNet, ResNet, VGG, and SqueezeNet; Yuechuan Chen took Inception and GoogLeNet; and Hua Tong took ResNeXt, Wide ResNet and MobileNet. The according parts in the report and presentation are contributed by the same person.

Besides that, for the final report, Hanlin Tang wrote the 3.3 optimizing, 4.experiments, 6.conclusion, 7.acknowledgements parts, and this part; Hua Tong wrote the abstract, 1.introduction, 2.related work and 5.2 confusion matrix & model comparison part; and Yuechuan Chen wrote 3.1 basic, 5.1. training Loss & model Accuracy parts.

For the slides, Yuechuan Chen contributed to and presented the Methods and Models, Hanlin Tang wrote the rest part. Hua Tong presented the Result and Conclusion.

References

[1] K. Dokic, L. Blaskovic, and D. Mandusic. From machine learning to deep learning in agriculture – the quantitative review of trends. *IOP Conference Series: Earth and Environmental Science*, 614, 2020.

[2] P. Gikunda and N. Jouandeau. State-of-the-art convolutional neural networks for smart farms: A review. *Science and Information (SAI) Conference*, July 2017.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] A. Kamilaris and F. X. Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, 2018.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[6] V. Kurama. A review of popular deep learning architectures: Densenet, resnext, mnasnet, and shufflenet v2. `https://blog.paperspace.com/popular-deep-learning-architectures-densenet-mnasnet-shufflenet/`, 2020.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[8] T. Tran, J. Choi, T. Le, and J. Kim. A comparative study of deep cnn in forecasting and classifying the macronutrient deficiencies on development of tomato plant. *Applied Sciences*, 1601(9), 2019.

[9] S.-H. Tsang. Review: Wrns — wide residual networks (image classification). `https://towardsdatascience.com/review-wrns-wide-residual-networks-image-classification-d3feb3fb2004`, 2018.

[10] J. Wei. Alexnet: The architecture that challenged cnns. *Towards Data Science*, 3, 2019.

[11] N. Zhu, X. Liu, Z. Liu, K. Hu, Y. Wang, and J. Tan. Deep learning for smart agriculture: Concepts, tools, applications, and opportunities. *Int J Agric Biol Eng*, 11(4):32–44, 2018.