Deep Learning for Identification of Toxic Plant Species

Jordan Perrone
Department of Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, USA
jperrone2024@fau.edu

Theodor Owchariw
Department of Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, USA
towchariw2020@fau.edu

Alejandra Coronel-Zegarra
Department of Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, USA
acoronelzega2022@fau.edu

***Abstract*— The field of deep learning, born out of machine learning in the 2000s, gave rise to efficient and accurate means of classifying large volumes of data. The Convolutional Neural Network (CNN) is a supervised learning tool that quickly became key for image classification by enabling deep learning models to detect local patterns within numerous images and later mesh those findings to determine an overall classification of a given image. Major technology corporations such as Google and Microsoft contributed to unique CNN architectures that are now offered in open-source packages for public use. These models are offered with pretrained information based on an extremely large dataset known as "ImageNet" developed as part of a community wide effort. In this exploration, three major CNN architectures were tested in search of the most appropriate method of classifying a dataset of 9,952 poisonous and non-poisonous plants from ten distinct plant species. Each architecture has its own nuances and preprocessing requirements. As such, a variety of techniques were implemented to ensure the models were able to learn local patterns for binary classification of plant images as toxic or nontoxic with the greatest possible validation accuracy. Specifically, InceptionV3, ResNet-50, and VGG-16 were evaluated with this dataset. Ultimately, ResNet-50 achieved the highest validation accuracy of approximately 84.7%. However, all three models converged to similar final validation accuracies within two percent of each other. Overall, the results indicate that the networks are theoretically capable of predicting correct classifications for new plant images with a satisfactory level of accuracy. This work explores deep CNN models for image classification and provides a starting framework for toxic plant classification using deep learning, which can contribute to public safety and ecological protection.**

Keywords—Deep Learning, CNNs, Plant classification, image classification

I. Introduction

Plants play a critical role in many sectors of society, including agricultural development and environmental protection. Significant expertise and manpower are often required for identification of plant species or detection of diseases. [1]. The application of this knowledge can be indispensable specifically when it comes to identifying poisonous plants. Poisonous or toxic plants pose a threat to humans due to similar physical traits compared to non-toxic plants especially when consumed or used as medicinal reservoirs [2]. Physical contact with and consumption of poisonous plants can lead to severe reactions, such as skin irritation, nausea, and vomiting [3]. The main culprits in North America include native poison ivy, poison oak, and poison sumac, accounting for at least 10 million accidental exposure cases each year [4]. Around three quarters of adults in the U.S. are clinically sensitive to these toxic plant species and develop allergic reactions when exposed, such as contact dermatitis [5]. This creates a need for increased public health and safety measures to mitigate further exposure and injury, such as accessible toxic plant identification technologies.

Many fields of study have collaborated to mitigate the dangers of misidentifying harmful plant species, and, more recently, Artificial Intelligence (AI) has been utilized. AI has become a topic of even the most casual conversations, though the roots of this rapidly growing field trace back many decades. The programmable computer was first developed in 1842, but theories about an intelligent machine existed for centuries prior [6]. The first breakthroughs did not come to fruition until the late 1950s. Machine learning is a mathematical and statistical approach to machine pattern recognition that attempts to follow the behavior of the human nervous system, also known as biomimicry [7]. Researchers from Cornell, namely psychologist Frank Rosenblatt, formulated the "perceptron

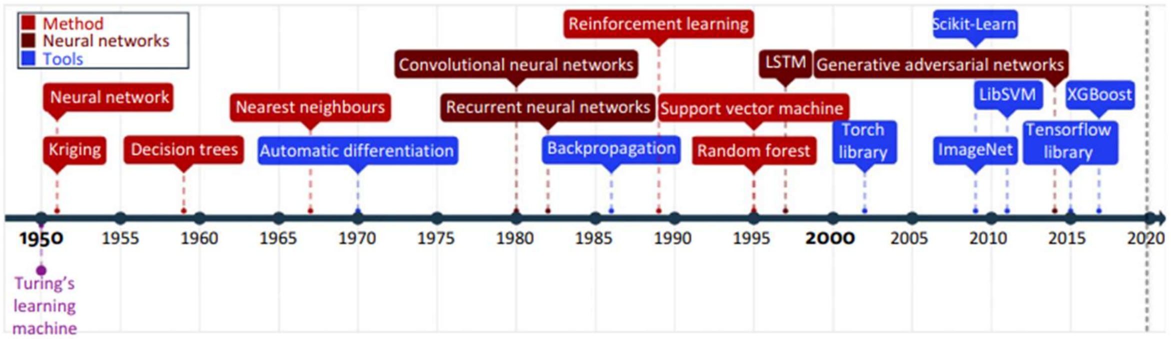

Fig. 1. Machine Learning Timeline [10].

machine" to recognize alphabetical patterns using analog and discrete signals and performed a mathematical analyses in 1959 [8]. This approach became the framework for artificial neural networks (ANNs) for many decades [9] and led to the development of the modern field of "deep learning" which is a more complex application of the same fundamental machine pattern recognition approach based on neurology of humans and animals.

Major breakthroughs in AI were held back for many years by a lack of resources. Particularly, computational power and data storage challenges in the late 20th century confined advancements in machine learning to mainly straightforward logical tasks in practice and mathematical analysis in research. Logistic regression, for example, allowed for artificial intelligence to provide decisions on whether to execute a cesarean birth delivery based on input information supplied by medical professionals in the 1990's [6]. However, performance remained largely dependent on data quality and human supervision. Global market growth in computing in the early 2000s provided major trends such as large-scale data storage and parallel processing [7]. Google and Nvidia are among the corporations that broke out in these areas as early as 2004 [7]. Two decades later, these organizations are two of the most powerful entities in the world in terms of both financial success and societal influence along with household names such as Microsoft, Amazon, Meta AI, and Open AI. Figure 1 provides a timeline of major developments in the world of machine learning [10].

With advancements in technology came the rise of more complex AI models such as the Deep Neural Network. The term Deep Neural Network (DNN) refers to several layers of artificially generated and incrementally modified connections of the digital "neurons" or "perceptions" that were proposed by Rosenblatt many years ago. Deep learning is a subset of machine learning that has gained attention due to its feature representation capabilities, allowing it to learn new information from training data and automatically extract features [11]. To better understand what DNNs are, it is important to understand how they work. Deep learning models rely heavily on linear algebra as coefficients define the relative significance of the connection between nodes in adjacent layers. These coefficients, called "weights" or "parameters", are stored in matrices in 2-D or "tensors" in 3-D [6]. Similarly, inputs and outputs may have higher dimensionality and require appropriate weight space dimensions for proper matrix operations. Matrices are used to represent layers in the broader group of deep neural networks referred to as "dense", "fully connected", or "multilayer perceptron" (MLP) networks. On top of these complex computations, activation functions are often applied to regularize the output of layers in the network to prevent exponential growth or decay that may hinder a network's ability to learn. Back propagation adds another layer of calculation complexity as predicted outputs from a network are compared to the expected outcome associated with the provided input. An error term is formed for each output node and propagated backwards through the network using partial differentiation to deduce the relative significance of weights in the network in terms of their influence on the network output and error term [12]. These concepts hold true for supervised learning, which constitutes training a model based on labeled data so that it can be applied to new data to make accurate predictions from a prescribed set of possible classifications. Figure 2 provides a basic example of a multilayer DNN with a single hidden layer. The diagram is truncated to show just a few nodes as networks used in deep

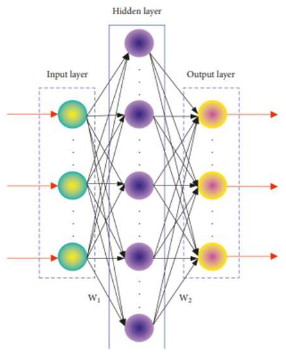

Fig. 2. Illustration of the transfer of data from the input layer to the hidden layer to the output layer. Each arrow between the input layer and output layer represents a flow of data transfer having its own weights learned from back propagation [13].

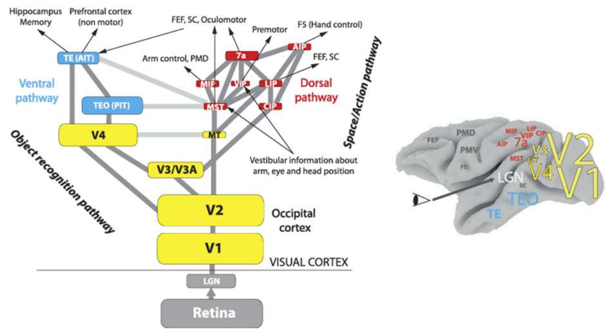

Fig. 3. Structure of the primate visual cortex [14].

learning for the past few decades are many orders of magnitude larger and contain many more hidden layers.

Although DNNs are generally considered highly effective for numerical datasets or even tokenized preprocessed inputs, they become problematic for image datasets even with modern computing power. A single image contains thousands of pixels that are each made up of an array indicating color and brightness. One of the most common pixel-encoding formats is "RGB" where 16 or 32 bits may be used to produce the spectrum of visible light from a combination of varying intensities of red, green, and blue light [14]. The human eye is the predominant sensory organ in the body, and it is constantly gathering and processing sensory data during waking hours. Neuroscientists and psychologists have studied animal brains for years and determined that the rods and cones contained in the eye convert light waves to electrical signals and pass them to the occipital lobe of the brain. The occipital lobe is responsible for image processing in the brain, but there is an extremely complex hierarchy involving the cerebral cortex, the hippocampus, motor control sections, and other elements of the human and animal nervous system. Kruger et al. discuss the hierarchies of the visual cortex in primates at length in their work regarding applications to "Computer Vision" [15]. Figure 3, adopted from this publication, displays a simplified illustration of the structure of the primate visual cortex.

Advancements in AI have led to new fields of study designed to help machines more effectively interact with their environment. Computer vision is a growing area of study that attempts to improve digital image recognition using the same principles as human and animal processing where local patterns are discovered in a field of view and ultimately processed in entirety at a deeper level in the brain [16]. Likewise, deep learning grew to support feed forward Convolutional Neural Networks (CNN) which were first proposed by Fukushima in 1980 [17] and commercialized circa 2009 with the introduction of "ImageNet". ImageNet is a challenge that brought upon "a renaissance of deep learning and particularly convolutional neural networks" in the 2010s where a database of natural images was used to create a competition to achieve the greatest possible classification accuracy [10]. Deep learning models have been reported to show good success specifically with image classifications [18]. Recent advances in image recognition using deep learning have focused on object detection [19], facial recognition [20] and medical imaging [21]. CNN's offer the ability to detect local patterns in image data mirroring the visual cortex of human or animal brains. They employ "filters" or small matrices of trainable weight parameters that iterate over successive subsections of the 2-D grid of pixels that make up an image. Numerous filters are combined to develop a "convolutional layer", and the extracted data from each filter is then conglomerated in a "pooling" layer which averages the extracted feature data to produce a matrix of lower dimensions [10]. This method is far less computationally intensive compared to a fully connected (dense) networks as not all parameters are directly linked. Figure 4 shows a simple filter applied on an input image and the resulting output feature extraction.

Numerous CNN architectures were developed in the 2010s including Google's InceptionV3, Microsoft's ResNet-50, and the VGG-16 network. These networks were all pretrained on the ImageNet dataset and made publicly available through open-source tools. The structure of a single ResNet block and VGG-16 are shown in Figures 5a and 5b, respectively. This report explores the usage of these three notable architectures as well as some simpler CNN models to perform binary classification of the selected dataset of poisonous and non-poisonous plant images. There are no clear indications from related works as to the most ideal network for the dataset. Each of these architectures has its own unique series of deep convolutional layers as well as preferred techniques that mitigate common issues with modeling deep layered networks such as overfitting and an increased time complexity.

Fortunately, robust deep learning frameworks such as Google's TensorFlow and Meta AI's PyTorch (both now open source) have been developed to streamline access and usage of powerful tools using popular programming languages such as

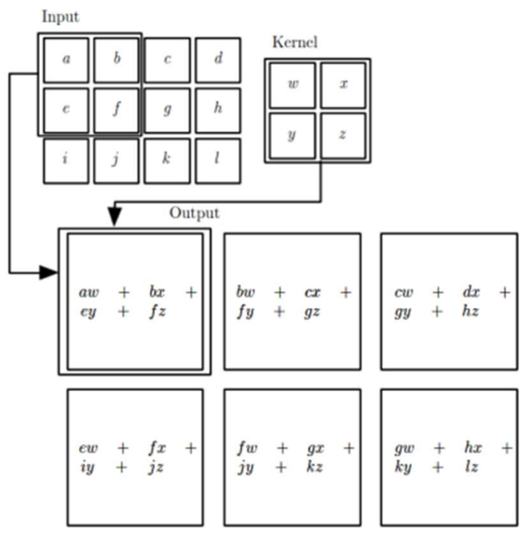

Fig. 4. Visual Representation of 2D Convolution [6]

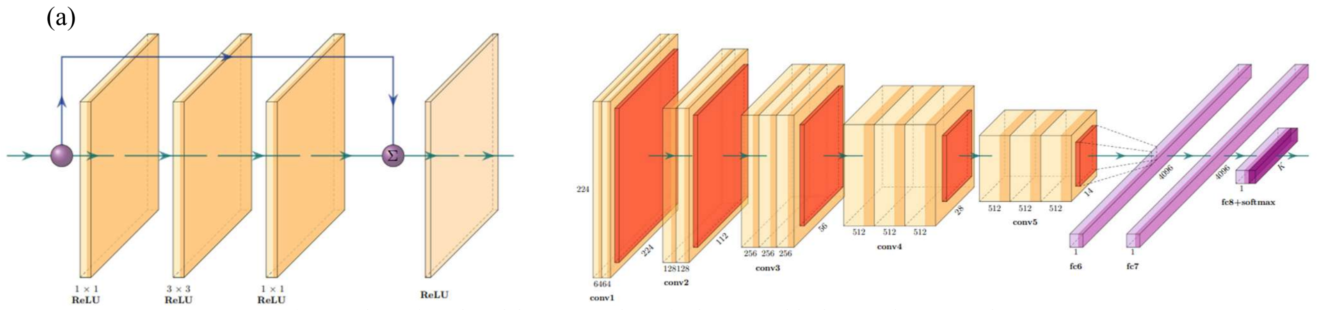

Fig. 5. Schematics of model structures for a single ResNet block (a) and VGG-16 (b) [10].

Python. TensorFlow is provided as a Python API and was developed in C and C++, while PyTorch is deeply integrated into Python [22]. These platforms were built to efficiently handle the computational demands of the growing field of AI for large scale applications and even personal computing. These frameworks were developed in partnership with computer hardware manufacturers such as Nvidia to ensure finite computing resources are used most efficiently. As a result, the TensorFlow library in python offers a version integrated with Nvidia's Compute Unified Device Architecture (CUDA), a collection of software and drivers that instruct computers powered by Nvidia Graphics Processing Units (GPUs) to use the parallel processing capabilities unique to this type of processor to speed up computation time of deep learning models [23]. Parallel processing has been utilized in research and academic settings to optimize computer performance for a wide range of applications, such as image classification tasks.

Image classification geared toward plant identification has been investigated using traditional machine learning models due to an increased need to identify plant species in a multitude of fields, such as botany, medicine, ecology and food production [24]. These methods have shown limited classification accuracy due to inadequate learning of complex patterns and hierarchical representations [25]. Libraries such as TensorFlow have allowed for the development of powerful classification tools using advanced methods and specialized CNNs to create user-friendly products in recent years. It is important to note that plant recognition may be very challenging for the average civilian with limited knowledge and resources. The process involves examining detailed combinations of physical characteristics such as leaf shape, color and texture. However, image-based processing is a potentially viable solution for automating plant classification.

II. Related works

Plant classification models have been reported using CNNs for a range of uses, such as leaf [26] and disease identification [27], weed detection [28], and pest identification [29]. However, little work has been done to optimize models specifically for classifying toxic plants. Satake et al. used a series of CNN architectures, including DenseNet, ResNet, and VGG-16, to perform classifications of nine types of toxic ornamental plants as an automated tool for preventing accidents caused by toxic plants with domestic animals [30]. Their best model, DenseNet, achieved the highest performance from a self-made dataset from Pinterest images with a reported accuracy of 97.67%.

Noor et al. investigated poisonous plant prediction using numerous CNN models, namely ResNet-50, Xception, and MobileNet and a large dataset of 50 different plant species from the Arabian Peninsula [31]. The performance for each model was recorded and ranged from 77% to 87% accuracy.

Alobeidli et al. employed GoogleNet and ResNet-50 CNN models to classify the toxicity of seven different plant, tree and shrub species in the UAE [32]. ResNet-50 had the best performance, with reported accuracy upwards of 90%.

InceptionV3 was employed by Xiaoling et al. in a flower classification using the Oxford-17 and Oxford-102 flower datasets, each with over 20 species of flowers [33]. The results showed reported accuracies of around 95% for both datasets.

The reported use of CNNs for toxic species detection is not limited to the plant kingdom. Cho et al. employed VGG-16, ResNet-50 and MobileNet deep learning models to classify toxic herbs from self-collected and captured images with a focus on distinguishing the toxic herb Aristolochiae Manshuriensis Caulis (AMC) from nontoxic Akebiae Caulis (AC) and Sinomenium Acutum (SA) herbs [34]. Reported accuracies for each model were upwards of 90%. Similarly, deep learning methods using VGG-16, ResNet-50 and GoogLeNet have been reported for the classification of edible, inedible and poisonous mushrooms from a dataset of 8,190 mushroom images, with the InceptionV3 model showing the highest accuracy of around 88% [35].

In this work, VGG-16, InceptionV3, and ResNet-50 CNN architectures were utilized to classify images of toxic and nontoxic plant species. The dataset consists of the most common North American poisonous plants—poison ivy, poison oak, and poison sumac—along with a collection of nontoxic plants commonly confused with them. These images were originally obtained from an iNaturalist dataset via Kaggle and evaluated in categorical classification models. In short, the goal of this work was to develop a trained deep learning model as a reliable tool for identifying common toxic plants to support public safety and ecological efforts.

III. Methodology

A. *TensorFlow Framework*

The Keras library provided within the open source TensorFlow framework for Python was used to load the images stored in local directories, convert the images into array data, process the images, and build the CNN models. Kera's built-in functions were used almost exclusively to ensure data transformation and processing was compatible with the model itself which was assembled and run using Keras. PyTorch is another powerful alternative available to perform the same tasks, however, prior experience with Keras in TensorFlow motivated the continued use of the platform. This exploration required an extensive trial and error process using image preprocessing methods, network learning rates, data augmentation, and fine-tuning approaches in search of the most accurate model for binary classification considering the available personal computing technology.

B. *Data Preprocessing*

The "Toxic Plant Classification Dataset" contains 9,952 images of both toxic and non-toxic plants of five species per class, with samples shown in Figure 6. The images are split into 4,976 of each class and grouped by species.

Preprocessing data is an essential step in constructing supervised learning algorithms as it extracts features that help deep learning models to better learn patterns for classification. This involves taking the raw data from a dataset and using various methods to convert the data into useful information. It is important to confirm that preprocessing methods are standardized on all entries of data before fitting a model. This ensures that the learning algorithm learns only patterns naturally in the data, not patterns introduced through mishandling of data.

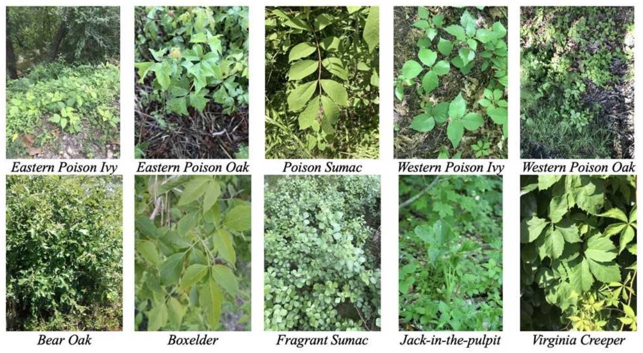

Fig. 6. Selected images of each species of toxic plants (top row) and nontoxic plants (bottom row) from Toxic Plant Classification dataset.

The first step in extracting features required separating images into two categories: those that contain a toxic plant and those that contain a nontoxic plant. This involved labeling of the training and testing data, where the images were labeled toxic or nontoxic using binary classifiers of 0 or 1, respectively. Once the images were properly separated, image processing was applied to resize images according to model requirements. For example, VGG and ResNet prefer images of 224x224 pixels while InceptionV3 prefers 299x299. The benefit of using a larger image size is that there are more pixels (units of image data) to analyze, but this comes with the cost of a higher computational burden. Additionally, it is quite possible that many of the extra pixels of a larger image will have been redundant. It is important for the image size chosen to have a balance of being large enough to analyze while not overloading the computer with too much information. High resolution may also trigger memory limitations when running the model, but Keras provides the option to modify the input arguments of the model to accept resolutions other than the preferred values by removing the "top layer" of the network.

For Keras to read the image data properly, the data had to be reshaped as follows: (no. samples, image_width, image_length, no. channels). For example, a list of 100 colored images of size 224x224 would have the shape (100, 224, 224, 3) where "3" represents the three channels (red, green, blue) of pixels in a colored image. For grayscale "3" would be replaced with "1". Images were converted into NumPy arrays or "image arrays" of data based on image encoding type such as "RGB" to enable the neural network to compute and recognize numerical patterns.

New preprocessing techniques, fine tuning approaches, and hyperparameters were progressively tested and implemented in search of the most effective model possible. This included testing the influence on model results due to changes in batch size, different combinations of image processing, unfreezing certain layers of the base models, using colored versus grayscale images, and adjusting learning rates.

Depending on the model, various other methods of image processing were used. One method included using data augmentation functionality provided by TensorFlow, where image samples were rotated, shifted, zoomed, or flipped. It also involved the modification of brightness by a factor arbitrarily applied by the program within a user specified range (i.e. floats such as 0.8 and 1.2 corresponding to factors of brightness adaptation to be applied to the image). This created more variability in the data to help protect the model from overfitting on the training set. The TensorFlow package dedicated to data augmentation was used to implement these changes to the training set only. The primary functions within this package perform the augmentations randomly to some of the image arrays for each epoch.

In addition to data augmentation, learning rate scheduling was explored to mitigate persistent overfitting issues. Also built-in functionality via TensorFlow, learning rate scheduling allows the model to evaluate the validation loss associated with successive epochs and reduce the learning rate when validation loss stagnates after a defined number of epochs. As a result, the next several epochs run with less penalization to modifications in weight parameters meaning the network may continue to identify some local patterns that were suppressed to control gradient vanishing and explosion in earlier epochs.

C. *Model Architectures*

Convolutional neural networks serve as the foundational building blocks for the specialized models that will be discussed in this section. CNNs are powerful neural networks that are effective for finding patterns in images. The filters in each convolutional layer take small samples of an image using windowing and find localized patterns in the area. This window is then shifted through the whole image at defined spacing to find patterns in each section of the image. This allows the neural network to find patterns in the details of the photo. However, it is important to note that CNNs on their own can struggle to find complex patterns and can result in lower accuracy scores. For usage in the dataset described in this paper, a higher complexity model with deep layers were required. An overview of the generic framework for classification using each model is shown in Figure 7. Additionally, a basic CNN model was constructed as a control experiment to compare with the more complex and sophisticated pretrained models. A schematic of the model structure is shown in Figure 8.

D. *VGG-16*

VGG-16 contains sixteen layers and is considered to be highly accurate but comes at a large computational cost. It gets the name VGG-16 as "16" represents the number of layers that it contains. The more layers that a model has the more complex it becomes which can help it to learn better, but training the network can be more computationally expensive and may suffer from gradient issues. The VGG layered network was selected to allow for a more accurate model despite the cost of a longer running time. What makes these models so effective is that they use transfer learning where a pretrained model extracts features from the dataset using only the pretrained weights. This model was best fit with fine tuning of one dense layer with 64 neurons and l2 regularization, a dropout rate of 0.2, and an output dense layer set to 1. Lastly, the model was trained over 50 epochs.

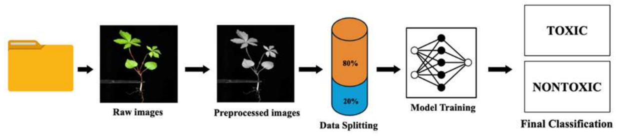

Fig. 7. The proposed framework for toxic plant classification. The raw images are obtained from the dataset and undergo different preprocessing methods, such as grayscale or brightness modification. The data is then split into train and test sets in varying ratios. The models are then trained to classify the images as toxic or nontoxic.

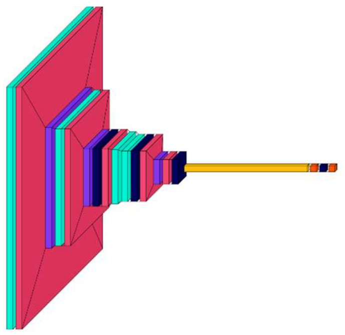

Fig. 8. Visualization of basic CNN. The model includes conv2d (blue), batch normalization (red), max pooling (purple), dropout (black), flatten (yellow), dense (orange) layers.

E. *InceptionV3*

InceptionV3 is the latest iteration of InceptionV1, once more commonly known as GoogLeNet, with improved inception modules that enhance accuracy. Inception-based models have been previously trained with over a million images and have the capability to classify up to 1,000 categories of objects [35]. InceptionV3 is a deep CNN model composed of 48 layers and applied to image classifications. The advantages of InceptionV3 are that it is suited for large datasets of varying image size and resolution [36], while protecting against the gradient issues from which very deep-layered networks suffer. Inception modules consist of a max pooling layer that precedes three small convolution layers [37] that further undergo filter concatenation in a non-linear aggregation with the output from the preceding layer. This provides significant dimensionality reduction, which translates to fewer parameters and lower computational costs, while also avoiding overfitting and enhancing the network's performance. The model architecture of InceptionV3 is shown in Figure 9. A sigmoid layer for classification purposes was added, in addition to a fully connected layer with 32 hidden nodes, a dropout with a rate of

Table 1. Model architecture and hyperparameters for VGG-16 model used in classification

Layer	Output Shape	Parameters
VGG-16	(None, 1000)	23,883,849
dense	(None, 32)	32,032
dropout	(None, 32)	0

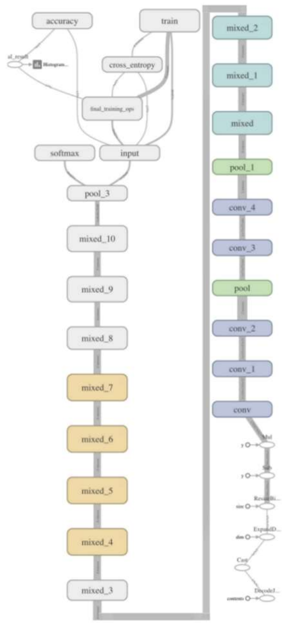

Fig. 9. InceptionV3 model structure [38].

0.2, and a ReLu activation function. The full model architecture as well as custom layers is shown in Table 2.

F. *ResNet-50*

As previously discussed, neural networks begin to experience vanishing or exploding gradients as the networks grow deeper and more complex. This causes an issue where weight parameters defining the relationship between nodes in adjacent layers will stop updating significantly or grow exponentially. The result is often very low classification accuracy as the network overemphasizes the importance of certain local patterns in terms of their relevance to the desired binary classification accuracy for the whole dataset. To counter this, ResNet was designed with the unique addition of "skip connections" that pass information from a layer several levels prior along with the information from the layer immediately preceding it. This gives the model the ability to leverage prior information as a point of reference to focus on learning from the input information rather than strictly from the features extracted from the previous convolution.

To develop a model for the specified use case, the ResNet-50 architecture was imported from TensorFlow and compiled with

Table 2. Model architecture and hyperparameters for InceptionV3 model used in classification

Layer	Output Shape	Parameters
inception_v3	(None, 1000)	23,851,784
dense	(None, 32)	32,032
dropout	(None, 32)	0
dense_1	(None, 1)	33

Table 3. Model architecture and hyperparameters for ResNet model used in classification

Layer	Output Shape	Parameters
ResNet-50	(None, 7,7, 2048)	23,587,712
Flatten	(None, 2048)	0
dense	(None, 64)	131,136
dropout	(None, 64)	0
dense_1	(None, 1)	65

an additional 64 x 1 dense layer followed by a single parameter dense layer for binary classification output. Dropout and L2 regularization were also implemented as part of the fine-tuning approach in the layers compiled after the main body of ResNet model. TensorFlow provides the option to utilize pretrained layers of ResNet based on the ImageNet dataset. This option was determined to be effective by trial and error as initially trained parameters yielded higher training and validation set accuracy than training the network entirely from scratch. A loop was implemented to iterate through the ResNet model and "unfreeze" the desired number of pretrained layers allowing the parameters in these layers to be trainable. Performance appeared to be best with just over half of the network layers unfrozen. Layers were unfrozen starting from the bottom of the network as pretrained weights were empirically determined to be desirable in earlier layers.

Data augmentation was used to feed the network modified versions of the initial images in successive epochs to provide a new perspective to the network to potentially learn more local patterns. A decaying learning rate was used to curtail overfitting by reducing the learning rate after 5 epochs of steady validation loss. Unfortunately, this combination of methods did not completely control overfitting for the ResNet model.

IV. Results & Discussion

The results of each model's performance are summarized in Table 4. The InceptionV3 model attained a maximum validation accuracy of 84.22%. The training accuracy was shown to be upwards of 99% for the last dozen epochs. A graph of model accuracy is shown in Figure 10. This could be indicative of the model memorizing the training data and, thus, overfitting in the testing. The total runtime was 18,600 seconds for 46 epochs and a batch size of 32 using CPU.

The ResNet-50 architecture provided the highest accuracy from the validation set at 84.69% after 32 epochs. Validation accuracy began to level at approximately 84% after the first dozen epochs, as shown in Figure 11. Overfitting was evident beyond this point as the model became tailored to the exact patterns of the training dataset and did not generalize well enough. Attempts were made to reduce overfitting by freezing more layers, modifying regularization, and using a decaying learning rate. Unfortunately, freezing more layers

Table 4. Training accuracy and loss for each model, along with best validation accuracy and loss.

Architecture Type	Total Epochs	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
ResNet	50	96.60%	0.1065	84.69%	0.8256
InceptionV3	50	99.55%	0.0188	84.22%	0.7884
VGG-16	50	86.30%	0.3520	82.06 %	0.4915

reduced model complexity and yielded results lower than the stated maximum accuracy, and other methods were not sufficient in managing overfitting.

VGG-16's architecture was imported from TensorFlow and attached to it was a dense layer of 64 units followed by dropout then an output layer for binary classification. This model also used 4x4 average pooling and L2 regularization to further tune the base model that was imported. The model showed 82.06% validation accuracy which was the lowest of the three tested as shown in Figure 12. Surprisingly, using InceptionV3's preprocessing function performed nearly ten percent better than using VGG-16's preprocessing function. For this reason, all data was fed into InceptionV3's preprocessing function. Using the same method used to unfreeze layers from the ResNet-50 model, it was observed that unfreezing the last 3 layers of the base model provided the best fitting model. In addition, the same methods of data augmentation used for ResNet-50 were used when fitting the VGG-16 model. This included using the same decaying learning rate tolerance. Nonetheless, this model also struggled with overfitting. However, ResNet-50 performed around two percent higher accuracy.

In addition to the three well known model architectures featured in this study, a basic CNN architecture was analyzed to understand the performance of simple models without pretraining. Unsurprisingly, this model was not successful as it achieved no more than 50% training or validation accuracy. It was hypothesized that the network was too shallow to recognize the complex patterns that toxic and nontoxic plants of numerous species tend to have in common.

The main limitation the team faced was in the effort to fine tune the models. Time itself became a constraint as it took a substantial amount of diligence to gather enough information to determine what parameters were working best. This was partially due to the availability of the hardware used in testing. Given that fitting the models is done automatically on the CPU using the standard TensorFlow installation for Python, computational efficiency was not optimal. Overall, running neural networks through the Graphics Processing Unit (GPU) performs much better [23]. Other hardware limitations included the age of CPUs that were available to the team and the amount of installed Random Access Memory (RAM). Running out of memory resulted in an error that halted compilation all together and caused the python kernel to crash. A workaround to limited memory involved closing out unnecessary background processes and restarting the compiler. For future work, it would be advisable to use more powerful computers running on a GPU to help shorten the time necessary to fine tune the model.

Fortunately, one of the available personal computers was equipped with a Nvidia GPU. Though this 2017 Dell laptop is several generations old, the availability of the outdated Nvidia GeForce GTX 1050 graphics card was tremendously effective in reducing computing time. For instance, the runtime required to calculate 50 epochs using InceptionNetV3 on the GPU enabled machine was nearly one quarter of the time required for the same model on the CPU reliant machine. Complications regarding parallel processing using GPUs are considered beyond the scope of this report and may be a topic of future exploration regarding this dataset. However, it should be noted that the availability of a basic, outdated GPU tremendously aided the process of testing numerous variations of model configurations.

The most notable limitation found in this work was the dataset used to train the model. It was observed by manual inspection

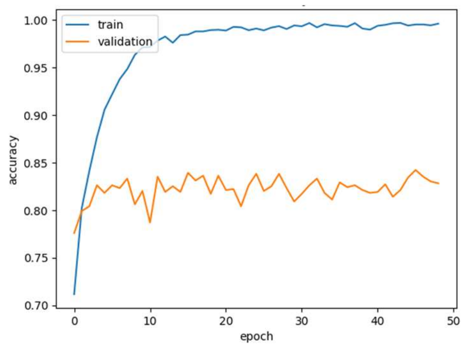

Fig. 10. Training and validation accuracies across all epochs for InceptionV3.

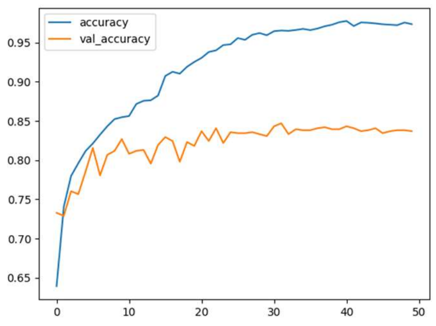

Fig. 11. Training and validation accuracies across all epochs for ResNet50.

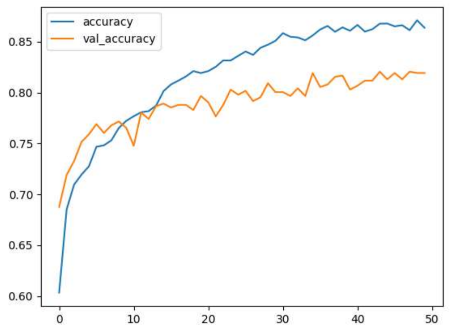

Fig. 12. Training and validation accuracies across all epochs for VGG-16.

that the dataset contains a lot of noise in the images as well as some extreme variability among samples. For example, image samples may contain humans, inanimate objects, or multiple plants. Therefore, developing methods to remove the background noise from the plant would be beneficial in helping to train the model on the subject itself rather than the noise also contained in the sample. Another way to counter this issue would be to gather new data with standardized methods of photography. This would circumvent the necessity of advanced noise reduction methods and require less time for preprocessing the images to improve model performance.

V. Conclusion

In this work, a series of deep convolutional neural network models were applied to the binary classification task of identifying toxic versus nontoxic plants. All models converged to a similar validation accuracy between 82 and 85% using varying preprocessing methods and hyperparameters. The use of parallel processing capabilities built into TensorFlow on an Nvidia GPU equipped personal computer allowed for significantly faster runtimes (almost 4 times faster than CPU counterparts). As such, more than one hundred modeling attempts were run to test different configurations. Future work should involve optimizing image preprocessing, improving runtime efficiency by using higher performance computers, and utilizing pretrained models specifically tailored toward plant image data.

References

[1] Y. Chen *et al.*, "Plant image recognition with deep learning: A review," Sep. 01, 2023, *Elsevier B.V.* doi: 10.1016/j.compag.2023.108072.

[2] K. E. Panter, K. D. Welch, and D. R. Gardner, "Chapter 37 - Poisonous Plants: Biomarkers for Diagnosis," in *Biomarkers in Toxicology (Second Edition)*, Second Edition., R. C. Gupta, Ed., Academic Press, 2019, pp. 627–652. doi: https://doi.org/10.1016/B978-0-12-814655-2.00037-2.

[3] N. Tamilselvan, T. Thirumalai, P. Shyamala, and E. David, "A review on some poisonous plants and their medicinal values," *Journal of Acute Disease*, vol. 3, no. 2, pp. 85–89, 2014, doi: 10.1016/s2221-6189(14)60022-6.

[4] L. Weinberg and P. N. Malani, "Poison Ivy, Poison Oak, and Poison Sumac," *JAMA*, vol. 331, no. 21, p. 1872, Jun. 2024, doi: 10.1001/jama.2023.26355.

[5] Y. Kim *et al.*, "Poison Ivy, Oak, and Sumac Dermatitis: What Is Known and What Is New?," *Dermatitis*, vol. 30, no. 3, pp. 183–190, Jun. 2019, doi: 10.1097/DER.0000000000000472.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[7] A. L. Fradkov, "Early history of machine learning," in *IFAC-PapersOnLine*, Elsevier B.V., 2020, pp. 1385–1390. doi: 10.1016/j.ifacol.2020.12.1888.

[8] F. Rosenblatt, *Two Theorems of Statistical Separability in the Perceptron: (Project Para) Contract Nonr-2381(00).* in Cornell Aeronautical Laboratory, Inc. Cornell Aeronautical Laboratory, 1958. [Online]. Available: https://books.google.com/books?id=5Q0-AQAAIAAJ

[9] R. R. Bush and F. Mostellar, "A mathematical model for simple learning.," *Psychol Rev*, vol. 58, no. 5, pp. 313–323, 1951.

[10] J. S. Dramsch, "70 years of machine learning in geoscience in review," in *Advances in Geophysics*, vol. 61, Academic Press Inc., 2020, pp. 1–55. doi: 10.1016/bs.agph.2020.08.002.

[11] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimed Tools Appl*, vol. 79, no. 19–20, pp. 12777–12815, May 2020, doi: 10.1007/s11042-019-08453-9.

[12] S. E. Dreyfus, "Artificial neural networks, back propagation, and the kelley-bryson gradient procedure," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 5, pp. 926–928, 1990, doi: 10.2514/3.25422.

[13] Q. Lv, S. Zhang, and Y. Wang, "Deep Learning Model of Image Classification Using Machine Learning," *Advances in Multimedia*, vol. 2022, 2022, doi: 10.1155/2022/3351256.

[14] G. M. Johnson and M. D. Fairchild, "Full-Spectral Color Calculations in Realistic Image Synthesis," *IEEE Comput. Graph. Appl.*, vol. 19, no. 4, pp. 47–53, Jul. 1999, doi: 10.1109/38.773963.

[15] N. Kruger *et al.*, "Deep hierarchies in the primate visual cortex: What can we learn for computer vision?," *IEEE Trans Pattern Anal Mach Intell*, vol. 35, no. 8, pp. 1847–1871, 2013, doi: 10.1109/TPAMI.2012.272.

[16] A. Grzybowski, K. Pawlikowska - Łagód, and W. Lambert, "A History of Artificial Intelligence," *Clin Dermatol*, vol. 42, Jan. 2024, doi: 10.1016/j.clindermatol.2023.12.016.

[17] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognit*, vol. 15, no. 6, pp. 455–469, 1982, doi: https://doi.org/10.1016/0031-3203(82)90024-3.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[19] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Trans Neural Netw Learn Syst*, vol. 30, no. 11, pp. 3212–3232, 2019, doi: 10.1109/TNNLS.2018.2876865.

[20] I. Adjabi, A. Ouahabi, A. Benzaoui, and A. Taleb-Ahmed, "Past, present, and future of face recognition: A review," Aug. 01, 2020, *MDPI AG*. doi: 10.3390/electronics9081188.

[21] K. Suzuki, "Overview of deep learning in medical imaging," *Radiol Phys Technol*, vol. 10, no. 3, pp. 257–273, 2017, doi: 10.1007/s12194-017-0406-5.

[22] F. Florencio, T. Valença, E. D. Moreno, and M. Colaço Junior, "Performance analysis of deep learning libraries: Tensor flow and PyTorch," *Journal of Computer Science*, vol. 15, no. 6, pp. 785–799, 2019, doi: 10.3844/jcssp.2019.785.799.

[23] M. Li *et al.*, "Deep Learning and Machine Learning with GPGPU and CUDA: Unlocking the Power of Parallel Computing," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.05686

[24] G. Saleem, M. Akhtar, N. Ahmed, and W. S. Qureshi, "Automated analysis of visual leaf shape features for plant classification," *Comput Electron Agric*, vol. 157, pp. 270–280, Feb. 2019, doi: 10.1016/j.compag.2018.12.038.

[25] C. P. Lee, K. M. Lim, Y. X. Song, and A. Alqahtani, "Plant-CNN-ViT: Plant Classification with Ensemble of Convolutional Neural Networks and Vision Transformer," *Plants*, vol. 12, no. 14, Jul. 2023, doi: 10.3390/plants12142642.

[26] C. Wick and F. Puppe, "Leaf Identification Using a Deep Convolutional Neural Network," Dec. 2017, [Online]. Available: http://arxiv.org/abs/1712.00967

[27] S. M. Hassan and A. K. Maji, "Plant Disease Identification Using a Novel Convolutional Neural Network," *IEEE Access*, vol. 10, pp. 5390–5401, 2022, doi: 10.1109/ACCESS.2022.3141371.

[28] J. Yu, S. M. Sharpe, A. W. Schumann, and N. S. Boyd, "Deep learning for image-based weed detection in turfgrass," *European Journal of Agronomy*, vol. 104, pp. 78–84, Mar. 2019, doi: 10.1016/j.eja.2019.01.004.

[29] J. Liu and X. Wang, "Plant diseases and pests detection based on deep learning: a review," Dec. 01, 2021, *BioMed Central Ltd*. doi: 10.1186/s13007-021-00722-9.

[30] S. S. Satake, R. Calvo, A. S. Britto, and Y. M. G. Costa, "Classification of Toxic Ornamental Plants for Domestic Animals Using CNN," in *Systems, Signals and Image Processing*, G. Rozinaj and R. Vargic, Eds., Cham: Springer International Publishing, 2022, pp. 108–120.

[31] T. H. Noor, A. Noor, and M. Elmezain, "Poisonous Plants Species Prediction Using a Convolutional Neural Network and Support Vector Machine Hybrid Model," *Electronics (Switzerland)*, vol. 11, no. 22, Nov. 2022, doi: 10.3390/electronics11223690.

[32] K. Alobeidli, M. Shatnawi, and B. Almansoori, "Image Classification for Toxic and Non-Toxic Plants in UAE," in *2023 Advances in Science and Engineering Technology International Conferences (ASET)*, 2023, pp. 1–5. doi: 10.1109/ASET56582.2023.10180875.

[33] X. Xiaoling, B. Nan, and C. Xu, *2017 2nd International Conference on Image, Vision and Computing (ICIVC 2017) : August June 2-4, 2017, Chengdu, China*. IEEE, 2017.

[34] J. Cho *et al.*, "Identification of toxic herbs using deep learning with focus on the Sinomenium Acutum, Aristolochiae Manshuriensis Caulis, Akebiae Caulis," *Applied Sciences (Switzerland)*, vol. 9, no. 24, Dec. 2019, doi: 10.3390/app9245456.

[35] N. Zahan, M. Z. Hasan, M. A. Malek, and S. S. Reya, "A Deep Learning-Based Approach for Edible, Inedible and Poisonous Mushroom Classification," in *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, 2021, pp. 440–444. doi: 10.1109/ICICT4SD50815.2021.9396845.

[36] Y. Jusman, R. O. Ningrum, and M. A. Fawwaz Nurkholid, "Leukemia Cell Image Classification Using CNN: AlexNet and GoogLeNet," in *ICEEIE 2023 - International Conference on Electrical, Electronics and Information Engineering*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICEEIE59078.2023.10334867.

[37] Y. Pan *et al.*, "Fundus image classification using Inception V3 and ResNet-50 for the early diagnostics of fundus diseases," *Front Physiol*, vol. 14, 2023, doi: 10.3389/fphys.2023.1126780.

[38] D. S. W. Ting *et al.*, "Development and validation of a deep learning system for diabetic retinopathy and related eye diseases using retinal images from multiethnic populations with diabetes," *JAMA - Journal of the American Medical Association*, vol. 318, no. 22, pp. 2211–2223, Dec. 2017, doi: 10.1001/jama.2017.18152.