Virginia Toxic Plant Classifier

1st Lakkoju, Siddarth
Department of Computer Science
University of Virginia
Charlottesville, Virginia, United States
qmn4cj@virginia.edu

2nd Parthaje, Shreepa
Department of Computer Science
University of Virginia
Charlottesville, Virginia, United States
cbf2xv@virginia.edu

3rd Gouru, Srikar
Department of Computer Science
University of Virginia
Charlottesville, Virginia, United States
mcj2vb@virginia.edu

***Abstract*—As more and more people travel outdoors and in the woods in an effort to interact with nature, public safety from toxic plants becomes an increasingly challenging problem. This project aims to tackle this issue with respect to plants native to the state of Virginia, something that has not been addressed before. Our dataset for plants to Virginia was used by combining images from the iNaturalist website with knowledge from The Socrates Project from the University of Virginia. We implemented three different algorithms for our approach: a Support Vector Machine (SVM), a Deep Convolutional Neural Network (CNN), and a ResNet50-based Transfer Learning Model. The SVM and CNN models failed to converge on our dataset, producing results with near 50% accuracy. Two-phased transfer learning yielded the best results, achieving an accuracy of 81% on the testing dataset. Future steps involve training image augmentation, larger training servers, and hyperparameter optimization. We also wish to work with researchers in the toxicology field to confirm our findings, after which we wish to utilize the CoreML platform on iOS to make our model seamlessly deployable in the field for public use.**

I. Introduction

Virginian natives have a high appreciation for nature with it being one of the largest attractions for the state. The Appalachian and Shenandoah offer incredible hiking trails that have beautiful forgery during autumn along with a vibrant green forest to explore in the summer. Humans are naturally curious beings who love to interact with nature: climbing trees, plucking leaves to examine them, and even eating pretty berries they find on trees. When we live in a time where the digital age decreases our interaction with nature, making interactions with nature of utmost importance, facilitating this natural curiosity, especially in children, is critical. The question is how can we make this curiosity safe?

The goal of this project is the insurance of safety when interacting with Virginia nature. Virginia is home to a host of unique poisonous plant species: some to eat, some to even just touch. However, there are no methods to determine if plants specific to Virginia are toxic or non-toxic. When this is a widespread concern that is difficult for even experts to identify, we believe that a machine learning based approach can help hikers, nature enthusiasts, and even everyday people in Virginia identify what is safe and what is dangerous. The goal here is very simple: make nature accessible to everyone with just the power of a modern cell phone everyone carries.

II. Methods

Initially a dataset was identified which split plants into categories of toxic and non-toxic plants; however, the aim of this project is to target Virginian residents. This is where the Socrates Project from the University of Virginia published by the Virginia Master Naturalist Program was critical to the project. The project aims to identify all the major poisonous plant species in Virginia, so utilizing the data from the Socrates project was paramount. The iNaturalist website is a free resource where naturalists crowdsource images of various animal and plant species, making it an amazing resource for a huge dataset. Utilizing the data from the Socrates project along with a web scraper released along with an initial toxic plant dataset allows the creation of the optimal solution for this problem.

A pipeline is created to preprocess our input images and labels. Input images were varied in size, so the images get resized to 128x128 pixel images using OpenCV. Converting them into grayscale images was considered, but there were concerns with too much valuable data about the plants to be lost. Input data points were also normalized so that the pixel values were between 0 and 1. Finally, for the output labels, One-Hot encoding was utilized. This was used to split the data into a 70-20-10 split for train-test-validation.

All of our methods detailed below were trained and tested using Jupyter Notebooks and Google Collab, leveraging the standard free-tier T4 GPU. Jupyter Notebooks allowed us to easily make changes in our code without having to retrain our model each time.

Our initial exploration of this⁄ classification problem was to rely on a support vector machine (SVM) to identify if a plant is toxic or not. SVMs are powerful and versatile classifiers because they are effective with high dimensionality data, robust to overfitting, and can handle non-linear data. They also support a variety of kernels, regularization with soft-margins, and outlier robustness. Additionally, we plan on running hyperparameter optimization to tune the SVM model.

After seeing poor results (further detailed below), we switched to a neural approach utilizing a DNN so it can learn hidden information and features within the image which an SVM might miss. Our experiments utilized the ”adam” optimizer and ”catagorical_cross_entropy” loss function when training. We quickly concluded that this was an insufficient

approach. so we added convolutional layers to DNN. CNN's, in theory, are adept at detecting visual features such as leaf shapes, textures, and color variations that distinguish toxic plants from non-toxic ones. The details of this CNN implementation are further outlined in the Experiments section.

While it was clear CNN's outperformed both a standard DNN and SVM, it was clear that leveraging existing network structures and trained models in a method of transfer learning would be crucial to delivering a safe application. The ImageNet dataset trained on the ResNet-50 model structure was used as the initial model. To this, a global average pooling and dense layers were added and these layers were independently trained on a binary cross-entropy loss function. Afterwards all layers were unfrozen with an independent fine-tuning step.

III. Experiments

A. Support Vector Machine

The Support Vector Machine (SVM) that we trained was implemented using the Python scikit-learn library. We tried two different kernels for our SVM: Polynomial and Gaussian Radial Basis Function (RBF). The Polynomial kernel is able to capture simpler patterns in data, but also runs significantly faster. On the other hand, the Gaussian RBF kernel is able to learn much more complex patterns in the input data, but is much slower when training and testing.

We also implemented hyperparameter tuning to optimize our SVM. One hyperparameter that we tuned was the regularization parameter, C, which determines the regularization strength. A larger C value penalizes misclassifications more heavily, while a smaller C value is more tolerant towards misclassifications. Another hyperparameter that we tuned was the kernel coefficient, gamma. A smaller gamma value tends to lead to a smoother decision boundary, whereas a larger gamma value will likely lead to a complex, wiggly decision boundary. We tuned these hyperparameters using the Grid Search Cross Validation technique. Our hyperparameter space included 4 C values, 4 gamma values, and 2 kernels, resulting in 32 different variations of the model being trained.

As shown in the Results section, this model failed to converge on our dataset. We believe that the image data is too complex for an SVM to be able to separate and classify. As the dataset size increased, the SVM training speeds increased significantly, especially with the addition of the GridSearchCV hyperparameter tuning. When looking at the images with the naked eye, the subtle differences between toxic and non-toxic plants are difficult to identify, and given the sheer scale of our input space (224 pixels x 224 pixels x 3 channels x 256 values per channel), an SVM seems infeasible.

B. Convolutional Neural Network

The first Deep Neural Network (DNN) architecture tested was a Convolutional Neural Network. We decided to use CNN's due to their ability to learn and identify different features in images in a bottom up compounding manner.

The custom CNN model consisted of 4 convolution layers and 2 dense layers. 2-dimensional max pooling was applied after each convolution layer to select for the most prominent features while reducing the image complexity. Lastly, simple dropout layers were added to reduce the chances of overfitting. The input and all hidden layers used the ReLU activation function and the output layer used sigmoid to represent a probability. The model consisted of 11,169,089 trainable parameters.

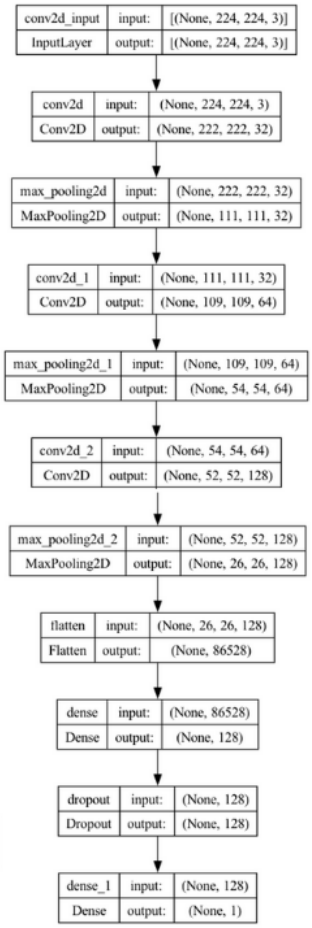

Fig. 1. Convolutional Neural Network Architecture

We compiled the model using the Adam optimizer and using binary cross entropy for our loss function. The model was trained for 20 epochs and a batch size of 32.

However as shown later in the Results section, this model also failed to converge on our dataset. We believe that one issue was the size of our dataset. As mentioned in the Method section above, our custom Selenium Web Server image scraper was not fast. Running the scraper overnight yielded a dataset of roughly 3,500 images. Convolutional Neural Networks require a large quantity of data to learn to properly identify features required for image classification.

C. Resnet 50 Transfer Learning

Building on what we learned from our prior experiments, we decide a transfer learning approach would be the best suited for this project. By using a pre-trained convolutional

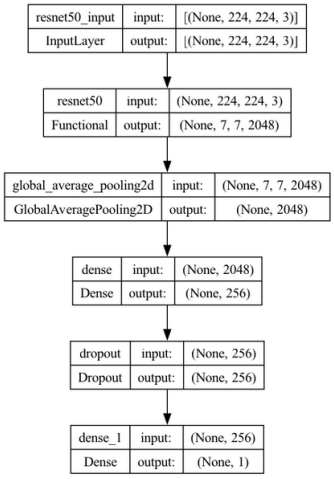

Fig. 2. ResNet50-Based Transfer Learning Architecture

neural network as our base model, its convolution layers would already have learned to properly identify a wide range of features. These learned features can be utilized for different related classification tasks allowing the model to leverage the knowledge gained from the original task.

For our base model, we decided to use ResNet-50 - a convolutional neural network 50 layers deep trained on the ImageNet database, a database with over 1.2 million labeled images. ResNet-50 achieved a top-1 classification accuracy of 74.9% and a top-5 classification accuracy of 92.1% on the ImageNet testing set. This impressive performance can be attributed to the large ImageNet training dataset and ResNet-50's relatively larger parameter count of 25.6 million.

For transfer learning, we removed ResNet-50's output layer and added 2-dimensional global average pooling and two of our own dense layers for the task of toxic plant binary classification [refer to above figure]. The global average pooling was added to reduce the dimensionality of the input to our custom dense layers.The hidden dense layer used the ReLU activation function and our output layer used the sigmoid activation function to represent the probability a plant was toxic. This transfer learning model had 24,112,513 parameters.

We utilized a two-step training approach with our transfer learning model. First, we trained the model on our training set with all the parameters of the base ResNet-50 model set to untrainable. The idea was to train our additional dense layers to use ResNet-50's feature recognition to identify toxic and nontoxic plants. For the second phase, we unfroze the parameters of the last 15 layers of ResNet-50 and continued to train our model with a very low learning rate of 0.00001 (as opposed to 0.0001 for phase 1). The point of phase 2 was to increase model accuracy by very slowly adjusting the parameters of ResNet-50 in conjunction with our own additional layers. The reason for unfreezing only the last 15 layers of ResNet-50 was due to overfitting problems (explained in the results section). For both phases, we trained the model with the Adam optimizer and binary cross entropy loss function.

IV. Results

A. Support Vector Machine

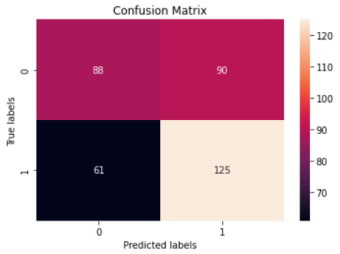

Fig. 3. SVM Confusion Matrix

Our SVM resulted in poor performance with an accuracy of roughly 58.5% on our testing data. It also achieved a precision of approximately 0.57, indicating that there were issues with false positives, and a recall of approximately 0.59, indicating that there were issues with false negatives. Overall, though, a 58.5% accuracy is significantly better than random chance, suggesting that the SVM model was able to pick up on some patterns among the dataset.

Our hyperparameter optimization also yielded an optimal C value of 1 and an optimal gamma value of 0.001. While the Gaussian Radial Basis Function (RBF) kernel was slower to train, it also proved better in terms of accuracy due to its ability to draw more unique, complex boundaries. While the SVM model proved to pick up on some patterns in the dataset, it does not seem to be complex enough to meet our classification goals.

B. Convolutional Neural Network

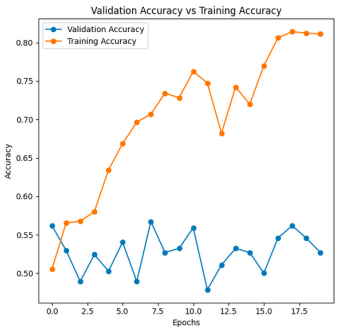

Fig. 4. CNN Training and Validation Accuracy

Our convolutional neural network performed well in terms of training accuracy, as it increased steadily over time. We trained it for 20 epochs, after which the training accuracy increased to a peak of 81% (Figure 4). Additionally, the training loss decreased steadily, dropping down to a value of 0.9 after the 20 epochs (Figure 5). However, this appears to be the result of overfitting, since our validation accuracy and loss did not improve. Our validation accuracy stayed at a relatively constant level, improving to only a 53% accuracy by the end of the training, which is only slightly better than flipping a coin

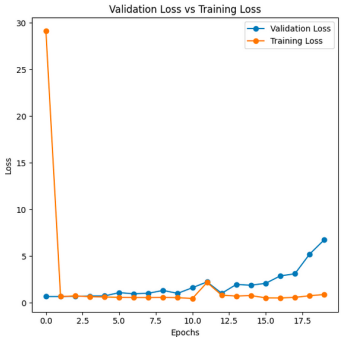

Fig. 5. CNN Training and Validation Loss

(Figure 4). This overfitting is further shown by the increase in validation loss over time, as it skyrocketed to a value of 6.8 after the 20 epochs (Figure 5).

After training, we tested the CNN on the testing dataset, which was independent from the training or hyperparameter tuning in any way. Our final testing accuracy ended up being a 46% accuracy, with a final testing loss of 7.9. This below-50% accuracy seems to indicate that the CNN did not extrapolate the training dataset at all, and rather it was severely overfitting to the training dataset. Given our dataset and GPU constraints, training a CNN model does not seem to be a viable strategy for our classifier.

C. *ResNet 50 Transfer Learning Phase 1*

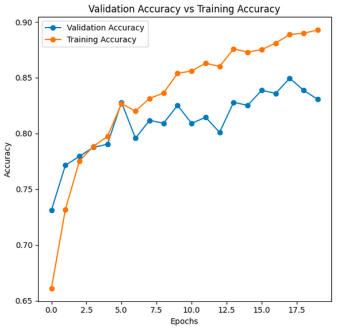

Fig. 6. Resnet50 Phase 1 Training and Validation Accuracy

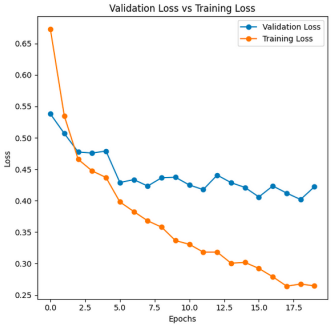

Fig. 7. Resnet50 Phase 1 Training and Validation Loss

As explained in our Methods section, our ResNet50-based transfer learning model was trained over two phases. During phase 1, after freezing the ResNet50 part of the architecture, we trained the additional layers to the model for 20 epochs. The model improved really quickly, as the training accuracy skyrocketed from 65% to begin with to 88% at the end of 20 epochs (Figure 6). Additionally, we validated that this improvement was not overfitting or memorization of the model by checking the validation accuracy, which was also steadily improving and reached 82% at the end of the 20 epochs. Genuine improvement of the network could also be noticed by looking at the training, which went down from 0.68 to 0.26 after 20 epochs, and the validation loss, which steadily decreased to 0.42 after 20 epochs (Figure 7). However, we noticed that around epoch 12 both the validation accuracy and loss were flattening out while the training accuracy and loss were still increasing, so we decided not to run more epochs in order to avoid overfitting before phase 2. After training the model, we tested it on our testing dataset, which resulted in an accuracy of 81%, confirming that our model was indeed not overfitting.

D. *ResNet 50 Transfer Learning Phase 2*

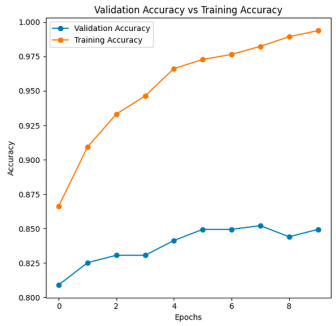

Fig. 8. Resnet50 Phase 2 Training and Validation Accuracy

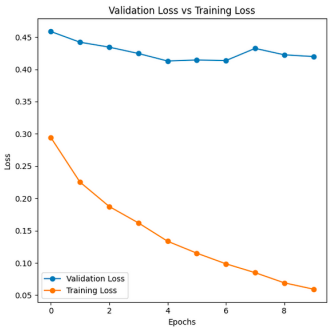

Fig. 9. Resnet50 Phase 2 Training and Validation Loss

We then ran phase 2 of the transfer learning algorithm. As explained in the Methods section, this unfreezes the rest of the model and allows training on the last "N" layers of the ResNet50 architecture. After manual tuning of the parameter "N", we found that unfreezing the last 15 layers proved the

most effective in terms of optimizing the model. We noticed that unfreezing less than 15 layers would result in a lower model accuracy, since it would not have enough layers to train on, and unfreezing more than 15 layers would result in overfitting of the pretrained model. However, this was determined via manual tuning, and a future improvement would be to add this parameter to a GridSearchCV or RandomizedSearchCV of some sort to find optimal behavior.

Since it was significantly slower and did not improve as much, we only trained phase 2 for 10 epochs. During this time, the training accuracy jumped from the previous 88% accuracy to a whopping 99% accuracy (Figure 8). The validation accuracy did not improve as much, but it did improve, going from only 83% to 85% in the same timeframe. While this indicates that the model did overfit, the validation accuracy increase shows that there was some generalizable learning occurring. This is further supported by the loss. The training loss decreased all the way to 0.05 over the 10 epochs, and while the validation loss also decreased, it only went down to 0.42 (Figure 9). Finally, we ran our network on the testing dataset, which resulted in a testing accuracy of 82%, providing a slight improvement in accuracy since our phase 1 model.

V. Conclusion

In this report, we have determined a strong need for identifying poisonous plants in Virginia along with a candidate solution using a neural based approach. Our approach to this problem yields an accuracy slightly better than 80%. This is a very promising set of results as it allows us to examine the problem further to squeeze out performance. Future steps involve training image augmentation, larger training servers, and hyperparameter optimization. In addition to this, we made initial explorations into deployment of this model utilizing the CoreML platform on iOS to ship the model inside of a compact application for iPhones. However, before deploying for public use we plan on ensuring safety by working with researchers in the toxicology field. Identifying the edge cases of the model and its weakness in the real world by biologists will make sure that the model will achieve the goal we specified in the introduction to this paper. Overall, the work in this research paper supports our initial hypothesis that a transfer learning approach is the right algorithm to solve this complicated problem which has the unique implication of using software to push people to the beautiful outdoors.

VI. Member Contributions

We worked together as a group for most of the tasks, allowing us to each contribute to all aspects of the project, although different members did take ownership of different aspects of the project. Shreepa focused on developing and running our data scraping pipeline to scrape iNaturalist. He researched the Socrates database and gathered image data on the best plants native to Virginia. Srikar worked on training, validating, and testing the SVM and CNN algorithms. He also worked on developing our data processing pipeline for each model. This involved preprocessing the data the same for both models, and also included hyperparameter optimization for the SVM model. Sid worked on training, validating, and testing the transfer learning algorithm and exporting our DNNs via CoreML for deployment. We used multiple Google Colab Jupyter notebooks as well as GitHub to work together and share our results with one another.

References

[1] Elliott H., (2022). Toxic-plant-classification, https://github.com/hans-elliott99/toxic-plant-classification/tree/main

[2] Goossens, A. (2020, December). Socrates Project. University of Virginia School of Medicine. https://med.virginia.edu/brpc/wp-content/uploads/sites/274/2021/01/SOCRATES-PROJECT-Jan2021.pdf

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

[4] INaturalist. iNaturalist. (n.d.). https://www.inaturalist.org/.

[5] Mukhlif, A. A., Al-Khateeb, B., & Mohammed, M. A. (2023). Incorporating a Novel Dual Transfer Learning Approach for Medical Images. Sensors, 23(2), 570.

[6] Noor, T. H., Noor, A., & Elmezain, M. (2022). Poisonous Plants Species Prediction Using a Convolutional Neural Network and Support Vector Machine Hybrid Model. Electronics, 11(22), 3690.

[7] Sanad. "Image Classification Using CNN (Convolutional Neural Networks)." Analytics Vidhya, 11 July 2023, www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/.