Select a Report:

- ○ Project Proposal
- ○ Project Midterm
- ◉ Project Final

CS 4641 Machine Learning: Final Report

Go to Project GitHub

Authors

Yuvraj Dhadwal | Semih Saglam | Mert Enes Yurtseven

Arya Ghods Nahri | Akhilsai Ujjina

Background & Literature Review

Introduction

Our goal for this project is to build a Machine Learning model that will be able to effectively identify plant species that are invasive to the Metro Atlanta area. Before we discuss this project further, let us first explore related research papers.

Literature Review

1. **Garcia-Navarrete, O. L & et al. (2024).** *Application of Convolutional Neural Networks in Weed Detection and Identification: A Systematic Review.* Agriculture.

Summary

2. **Saini, N. K. & et al. (2024).** *Invasive Flower Species Detection using CNN and Alert System.* ArXiv Preprint.

Summary

3. **Yasin, K, & et al. (2023).** *A novel multi-head CNN design to identify plant diseases using the fusion of RGB images.* Ecological Informatics.

Summary

Dataset

Click to view dataset details

Problem Overview & Motivation

Trees Atlanta hosts weekly Invasive Species Removal events where a group of volunteers rip out invasive plant species in parks, national forests, and other green spaces. However, most of these volunteers do not know all the invasive species or how to identify them.

Our proposal is a machine learning model that can correctly identify whether a plant is invasive to the Metro Atlanta area to assist these volunteers.

Our solution differs from previous literature because our model will be able to identify a diverse range of invasive plant species rather than just weeds or flowers.

Data Processing

This preprocessing pipeline we have implemented is designed to prepare the image datasets for training and evaluation. For the training phase, it applies a series of data augmentation techniques to improve model generalization. We used the following data augmentation techniques:

- Random Rotation
- Random Resized Crop
- Random Horizontal and Vertical Flip
- ColorJitter
- Random Affine
- Random Grayscale
- Random Gaussian Noise
- Random Erasing

These data augmentation techniques were picked as they were very helpful with diversifying the dataset. We also normalized the data by calculating and using the mean and standard deviation of each color channel (RGB). This normalization step was crucial as it improved the stability and performance of the model, especially during training. This was an extremely important step as it improved stability and performance in the model. Overall, this combination helped vastly to preprocess the data, and the preprocessing made the model more generalizable and effectively improved the efficiency of the model.

We applied the same preprocessing pipeline to our dataset for all three models to ensure that differences in accuracy are attributable to the models themselves rather than variations in the dataset.

Implemented Models

We implemented three ML models for the plant classification task. Namely, we have a *convolutional neural network* (CNN), a CNN with intermediate layers for feature extraction, and a modified variant of a *vision*

transformer (ViT) using cross-attention, called CrossViT.

Convolutional Neural Network

This is our most basic architecture, consisting of an input convolution layer, input pooling layer, output convolution layer, output pooling layer, and finally two fully-connected layers to compute the final class scores.

- **Input Convolution:** The input layer accepts tensors in R^{b x c x w x l}, where b is the batch-size, c is the channel dimension, and w, l are the width and height, respectively, of the input images. In our case $c = 3$, representing RGB color channels of the images. At a high level, the input layer unpacks the least granular "global" features of an image. For example, whether a plant is a tree, bush, flower, weed, etc.

- **Input Pooling Layer:** This is a max-pooling layer the takes outputs from the input convolution layer, subject to a ReLu activation function, and performs feature selection. This prevents the model from retaining redundant information and isolates the most features important to the classification task.

- **Output Convolution Layer:** This layer takes the output from the input pooling layer and performs another convolution operation. Intuitively, the output convolution layer identifies the more second-order, or "granular", features. For example, this layer might encode how many leaves a plant has or its color.

- **Output Pooling Layer:** The output pooling layer is another max pooling layer. It takes outputs from the output convolution layer after again applying a ReLu activation function. The output pooling layer's purpose is identical to the input pooling layer; i.e., it performs feature selection to isolate important information.

- **Fully-Connected Layers:** Finally, we apply ReLu again and then pass the data through two fully-connected layers (with a ReLu between them). These FC layers serve to collate the convolution outputs into (unnormalized) class scores, where the class corresponding to the highest score is chosen as the model's prediction.

- **Hyperarameters:** We selected the following hyperparameter values (unless 'input' or 'output' is specified in a parameter name, it applies to both conv layers)

    - **Input conv chans:** 32
    - **Conv kernel size:** 1
    - **Conv Stride:** 1
    - **Conv Padding:** 1
    - **Pooling layer kernel size:** 2
    - **Pooling layer stride:** 2
    - **Output conv chans:** 32
    - **Fully-connected layer output features:** 128
    - **Number of classes (invasive/non-invasive):** 2

Vision Transformer

This is a model developed at IBM that proposes a modification to the canonical vision transformer (ViT) model. The ViT model tokenizes images and uses an attention mechanism to associate information and establish context between tokens. Conversely, the CrossViT model computes tokens of *varying sizes* for each image. It then uses a mechanism called *cross attention* to associate the large and small tokens.

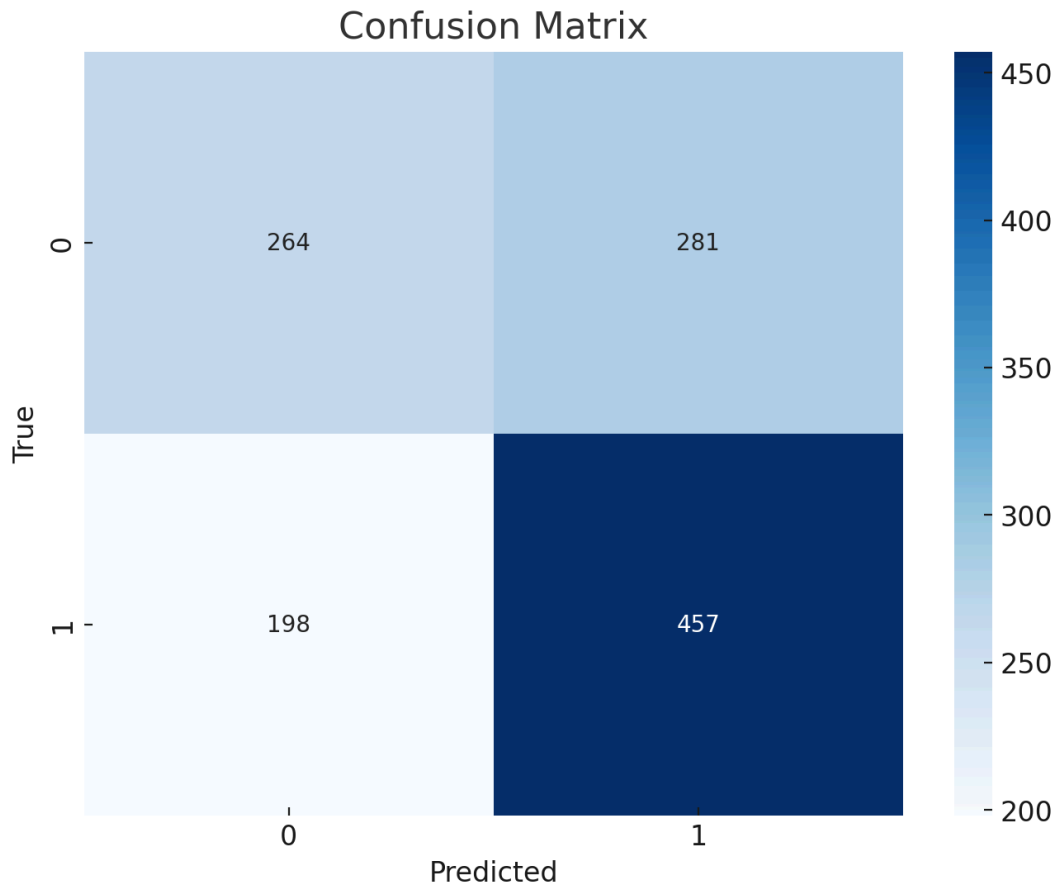
CNN Feature Extractor-Transformer Encoder

This model is architecurally identical to the CNN model, except for the addition of n square $CimesC$ convoluton layers between the input and output convolutions, where C is the input conv chans. Our rational here is that these intermediate layers will aid the model in identifying more granular features than were found previously. In this initial implementation, we have only used 1 intermediate layer, but in the future a hyperparameter search should be done on the number of intermediate layers.

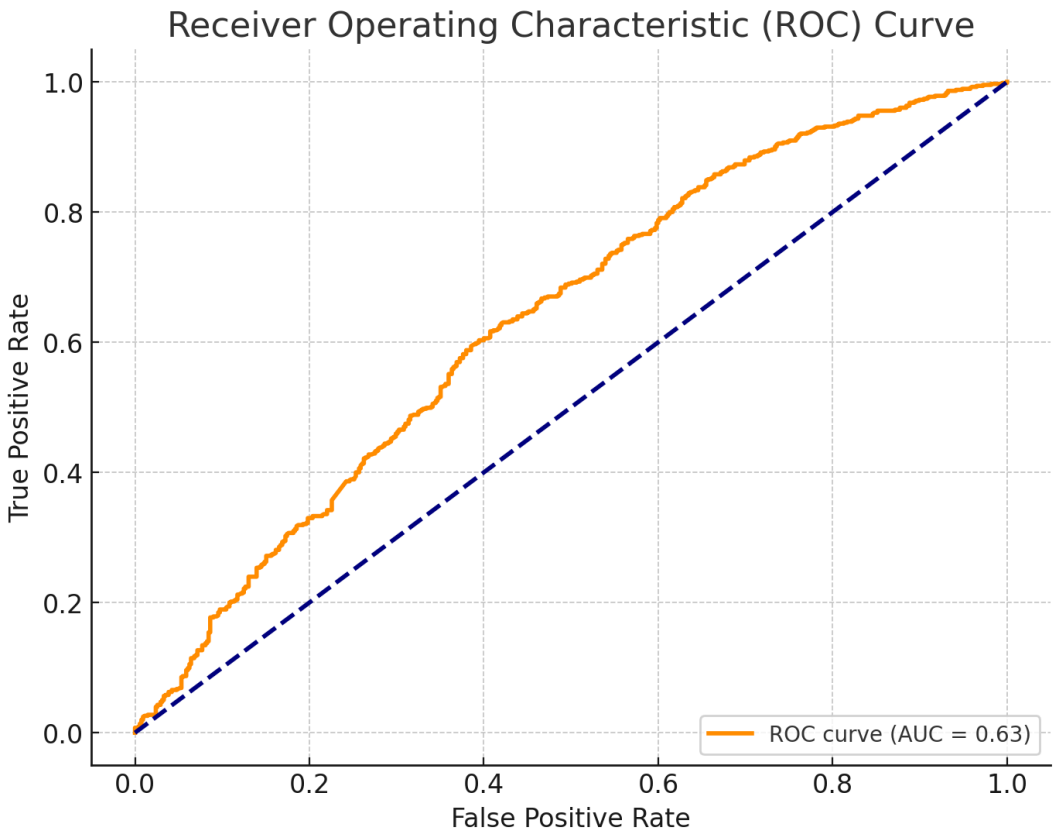
Results

All three models were trained with a learning rate of 0.001. The first two models were trained on 30 epochs on roughly 4000 images, while the third model was trained on 15 epochs on roughly 1600 images.

Convolutional Neural Network



Confusion Matrix



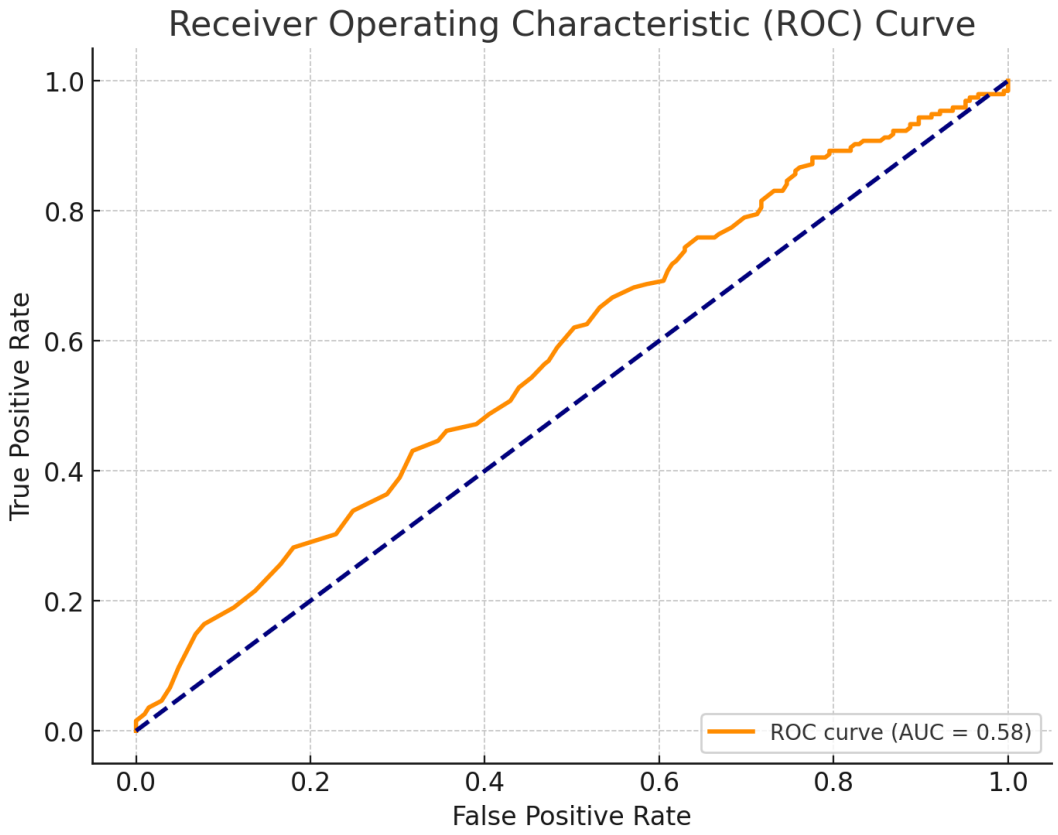
ROC Curve

Metric	Value
Test Loss	0.658596420288086
Accuracy	60.083335876464844
Precision	62.08790969848633
Recall	69.00763702392578
F1 Score	65.36514282226562

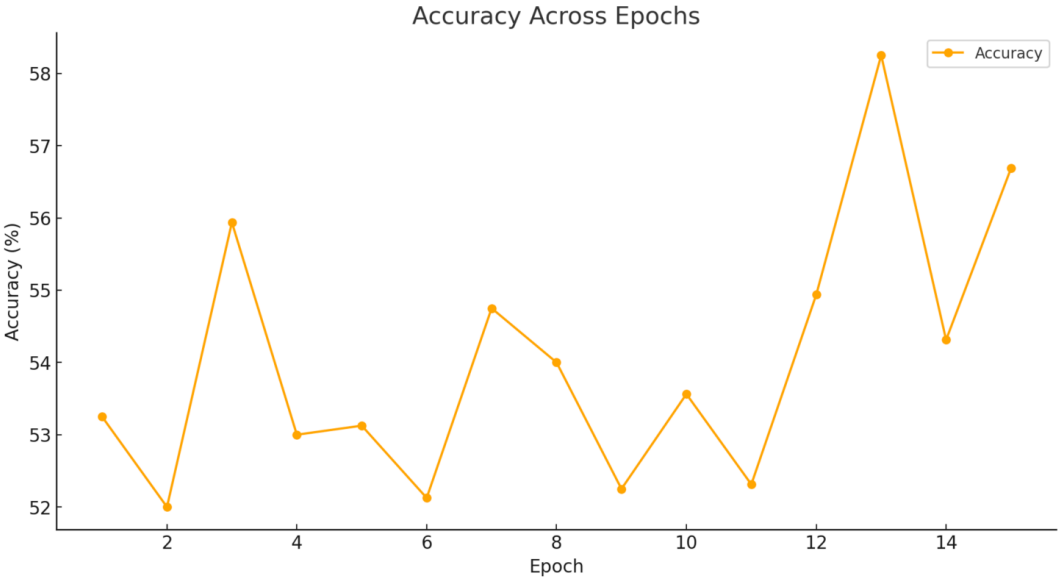
Based on these metrics, it is pretty clear that our CNN model is no where near as accurate as we initially expected. With an AUC closer to 0.6, this suggests the model has weak discriminatory ability and is only slightly better than random guessing. This is proven through our calculations with accuracy and other metrics calculated in table above.

We can improve this model further in the future by training this model on a much larger dataset. Currently this model was only trained on 4,000 images wheras we have in total over 59,000 images we downloaded from GBIF. The reason we chose to only train the model on a small subset is because on time constraints on the PACE ICE GPUs. If this was an actual application in production, we would have trained it on much more images which would have made our vanilla CNN much more accurate. We also only trained for 30 epochs so this could have been improved if we had more time to run the program.

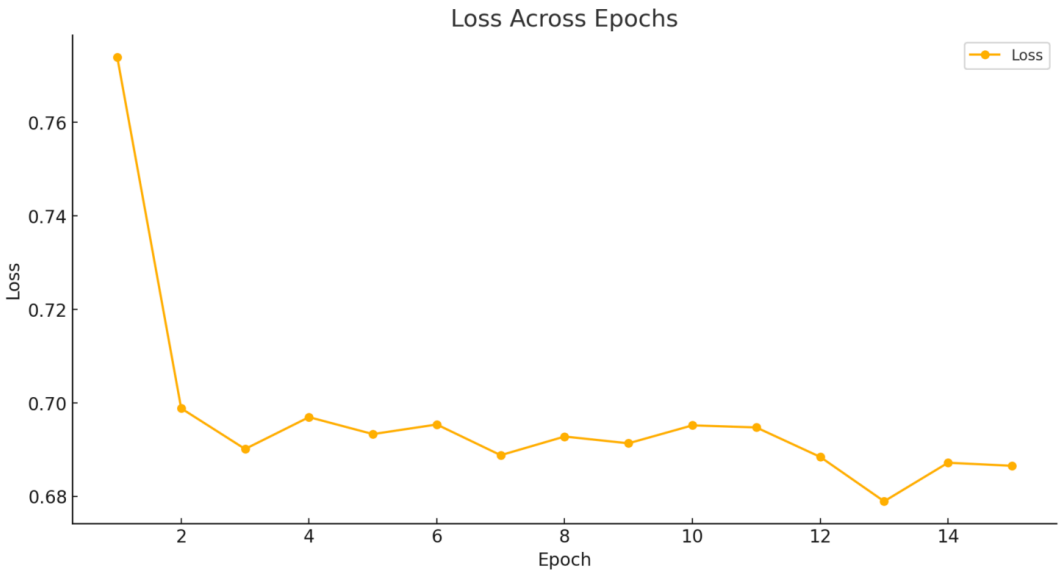
Vision Transformer



ROC Curve



Accuracy Per Epoch



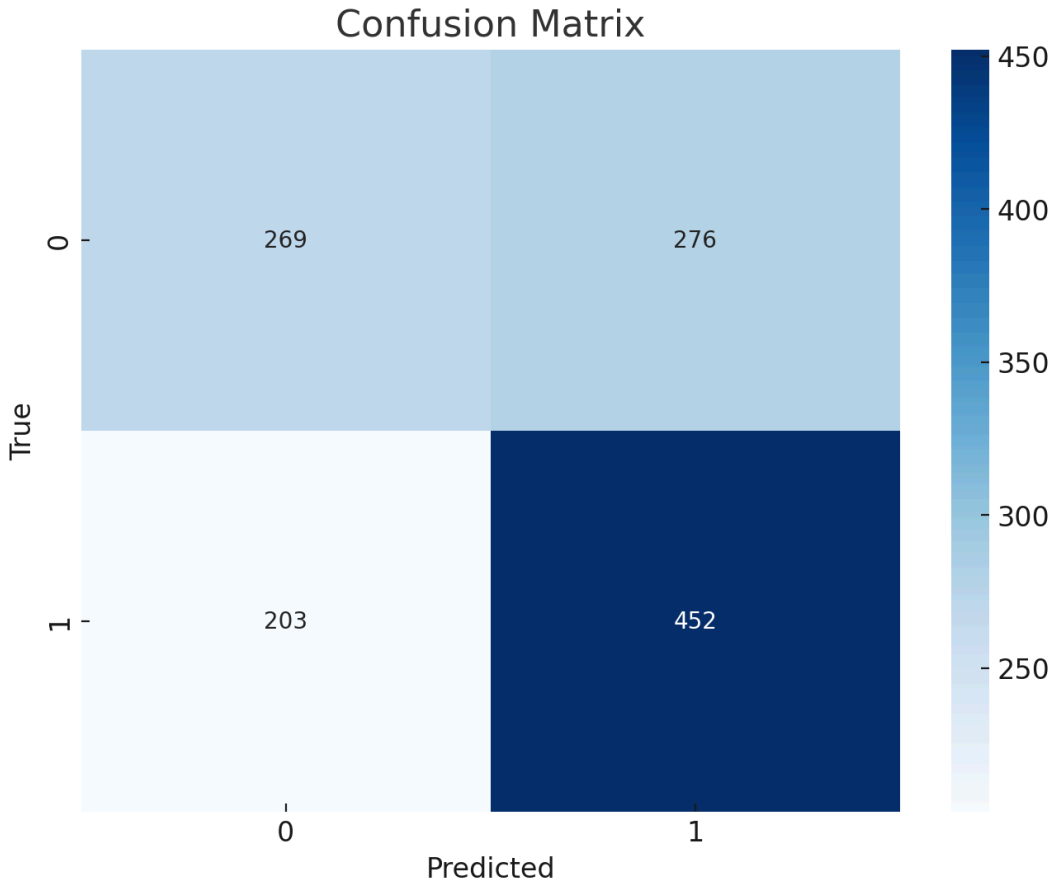
Loss Per Epoch

Metric	Value
Test Loss	0.7007177734375
Accuracy	48.75
Precision	48.75
Recall	100.0
F1 Score	65.54621887207031

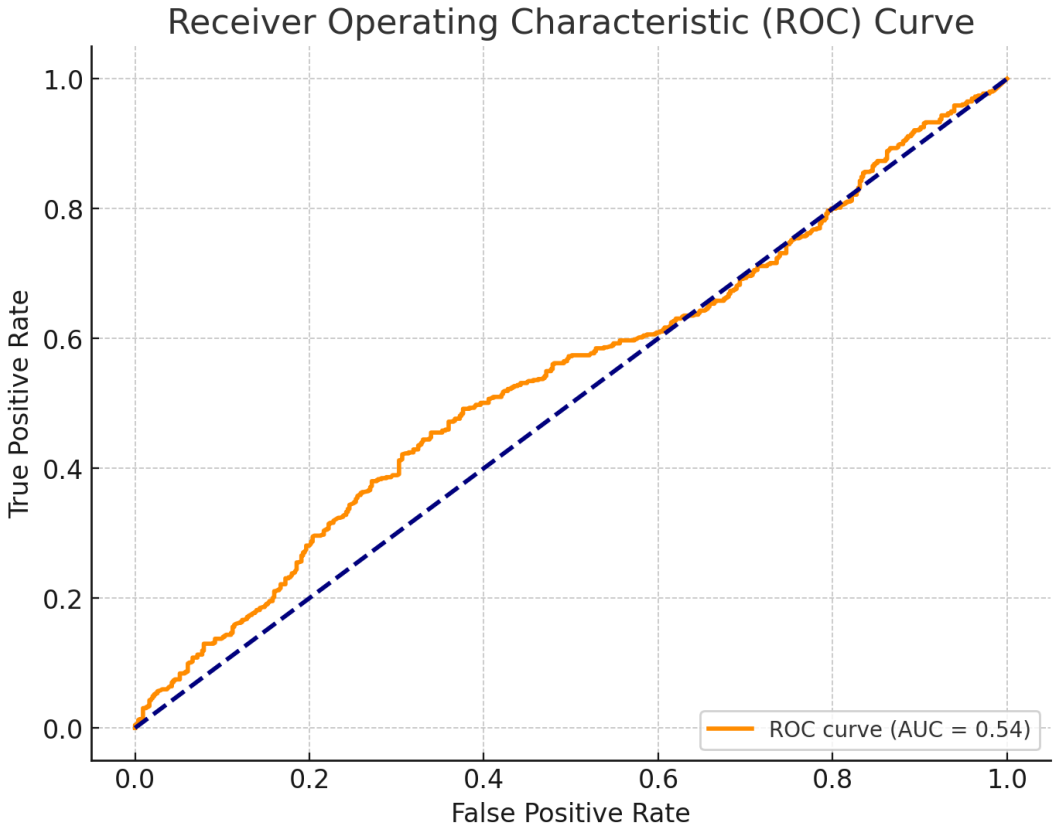
Based on accuracy, this model is about as good as random guessing. In fact, it predicted everything was invasive as seen from the 100 recall rating. However, its AUC being 0.6 and accuracy having an upward trend during training suggest that the model was a lot better on the training data. This suggests that there might have been a lot of overfitting with the model.

In the future, this model could have been trained on more images and more epochs. This model was only trained on 1,600 of the available 59,000 images and for only 15 epochs. Again, this was due to PACE ICE limitations, and in a perfect world, we would not have these limitations.

CNN Feature Extractor-Transformer Encoder



Confusion Matrix



ROC Curve

Metric	Value
Test Loss	0.6617174021402995
Accuracy	60.083335876464844
Precision	61.92411804199219
Recall	69.77098846435547
F1 Score	65.61378479003906

Based on these results and an AUC of 0.54, it is pretty clear that this model is effectively as good as random guessing. Which is again, much worse than we initially expected and also worse than our vanilla CNN. We believe the main reason for this was due to poor choice in hyperparameters for this model.

Similar to the other models, if this was an actual product, we would train it on much more images. This model was only trained on 4,000 of the 59,000 images we downloaded from GBIF. We could have done a much more extensive hyper parameter search as well in order to boost model performance.

General Next Steps/Lesson's Learned

It is pretty clear from these models and their statistics that this is not nearly what we expected from these models going into this project. We believe the majority of these issues can be attributed to a few issues.

- Small training dataset, we only used a fraction of the available data when training these models which definitely played a huge part in why they did not measure up to expectations.
- Small training times, again we only trained them for a short amount of time and in the future, we would definitely train them for much longer time.
- Due to small training dataset, we believe there may have been some overfitting in the data which we could try to avoid in the future
- We believe that the preprocessing might have been a little too heavy handed which probably artifically handicapped our model performance forcing it to train longer for better performance, which we did not do
- In general we should have done a more extensive hyperparameter search for our models.

Ethical Considerations

Invasive plant species wreak havoc on local ecosystems; hence, it is an important task to identify and remove them. However, we must be careful to avoid false positives, as removing native species can de-stabilize local ecosystems, leading to the death of both plants and wildlife. Furthermore, removing native plant species can open gaps that allow for more invasive species. Thus, our primary ethical consideration is ensuring high accuracy along with high precision, as this would indicate our model is unlikely to identify false positives. Furthermore, we must ensure our testing set is sufficiently large and diverse. Having an unrepresentative testing set will skew the aforementioned metrics in a misleading manner.

Contribution Table

Name	Contributions
Yuvraj Dhadwal	Introduction/Background, Problem Definition, and Data retrieval code, Streamlit Documen
Arya Ghods Nahri	Model Implementation, Training and Testing Model, Evaluation Documentation
Semih Saglam	Video, Data Preprocessing
Akhilsai Ujjina	Gantt Chart, Contribution Table, Training and Testing model
Mert Enes Yurtseven	Methods, Data Preprocessing
All	References

Gantt Chart

Task	Project Proposal	Introduction/Background	Problem Definition	Methods	(Potential) F
Task Owner		Yuvraj Dhadwal	Yuvraj Dhadwal	Mert Enes Yurtseven	Arya Ghods
Start Date		09/29/2024	09/29/2024	09/29/2024	09/29/2024
Due Date		10/04/2024	10/04/2024	10/04/2024	10/04/2024
Duration		6 Days	6 Days	6 Days	6 Days

References

- Garcia-Navarrete, O. L & et al. (2024). *Application of Convolutional Neural Networks in Weed Detection and Identification: A Systematic Review.*
- Saini, N. K. & et al. (2024). *Invasive Flower Species Detection using CNN and Alert System.*
- Yasin, K, & et al. (2023). *A novel multi-head CNN design to identify plant diseases using the fusion of RGB images.*