DEEP LEARNING APPROACH TO MUSHROOM SPECIES CLASSIFICATION

Yanni Alan Alevras
Student# 1009330706
yanni.alevras@mail.utoronto.ca

Nicholas Biancolin
Student# 1009197726
n.biancolin@mail.utoronto.ca

Eric Liu
Student# 1009098450
ey.liu@mail.utoronto.ca

Jason Ruixuan Zhang
Student# 1008997631
jasonrx.zhang@mail.utoronto.ca

1 PROJECT DESCRIPTION

Fungi identification is an increasingly critical task, with implications in food security, industrial use, conservation efforts, and biosafety. However, visual and image classification of fungi is a difficult task due to the wide variety of species. The goal of this project is to develop a deep learning model that can accurately identify macrofungi (fungi with large bodies) based on their genus.

87 percent of the land in Ontario is Crown Land, filled with expansive forests and a diverse array of plant species. Among these, mushrooms stand out as both common and challenging to identify due to their wide variety and the subtle differences between species. This identification task is particularly crucial because while some mushrooms are edible, others are highly poisonous, potentially posing a significant risk to foragers and nature enthusiasts.

We believe deep learning is a suitable approach for this task as deep learning models have proven to be effective in image classification tasks. Furthermore, there is a robust amount of data available for training, helping to ensure a solid model. We have selected the MIND.Funga dataset (**?**), which has approximately 17 000 images of nearly 500 species of fungi. This dataset is well-suited for our project, as it is built primarily for use in deep classification models.

2 INDVIDUAL CONTRIBUTIONS AND RESPONSIBILITIES

All the members of our team have worked together on various projects in the past. We have a good understanding of each other's strengths and weaknesses, and we have developed strategies to work together effectively. We have divided the work based on our individual strengths and interests, and we have regular meetings to discuss our progress and make decisions together. For project timelines and task distribution, we use a shared Notion. This allows us to concurrently see each other's progress, in addition to tracking our program and document updates using a version control software Git. For further clarity and understanding of each other's progress, we frequently demo our work to each other, for feedback.

2.1 YANNI ALEVRAS

Work completed to date: Researching relevant forms of data augmentation. Application of data augmentation techniques to the dataset. Data Processing and Individual Contributions and Responsibilities sections of the report.

Work to be completed: Hyperparameter adjustments to tune model. Baseline model improvements. Final evaluation of performance.

2.2 Nicholas Biancolin

Work completed to date: Training and preliminary tuning of model. Project Description and Individual Contributions and Responsibilities section of report.

Work to be completed: Implementation of data augmentation to model including testing with or without certain methods to maximize accuracy. Final evaluation of performance.

2.3 Eric Liu

Work completed to date: Rough draft for Primary Model section of the report. Diagram of model architecture. Data separation and applying transfer learning with AlexNet. Class structure, training function, accuracy function. Validation loss and graphs.

Work to be completed: Hyperparameter adjustments to tune model. Progress diagrams for final report. Final evaluation of performance.

2.4 Jason Zhang

Work completed to date: Python environment and requirements. Genus grouping to reduce number of classes. Random forest baseline model with metrics. Baseline Model section of the report.

Work to be completed: Iterative improvement to reduce training and inference times in code. Final evaluation of performance.

2.5 Distribution Table

Project Progress Report	Eric Liu	Jason Zhang	Nicholas Biancolin	Yanni Alevras
Brief Project Description (June 30th, 11:59 pm)			W	
Individual Contributions and Responsibilities (June 30th, 11:59 pm)			W	W
Contributions - Data processing (June 30th, 11:59 pm)				W
Contributions - Baseline model (June 30th, 11:59 pm)		W		
Contributions - Primary model (June 30th, 11:59 pm)	W			
Illustrations (July 1st, 11:59 pm)	W			W
LaTeXformatting (July 2nd, 11:59 pm)	W	W	W	W
Editing (July 3rd, 11:59 pm)	ED	ED	ED	ED
Final proofreading (July 4th, 6:00 pm)	W	W	W	W

Table 1: Project Progress Report Task Breakdown

Project - Training and Testing	Eric Liu	Jason Zhang	Nicholas Biancolin	Yanni Alevras
Data cleaning (June 16th, 11:59pm)		W		W
Image grouping (June 16th, 11:59pm)		W		ED
Transfer data to training format (June 16th, 11:59pm)	W	W		
Data annotations, splitting (June 16th, 11:59pm)	W	W		ED
Data augmentation (June 16th, 11:59pm)				W
Transfer learning (June 19th, 11:59pm)	W		ED	
Baseline model (June 19th, 11:59pm)		W		
CNN architecture (June 19th, 11:59pm)	W		W	
Training loop (June 19th, 11:59pm)	W	ED	ED	
Plotting and metrics (June 19th, 11:59pm)	W	ED	W	
Hyperparameter adjustments (July 15th, 11:59pm)	W	ED		W
Iteration and improvements (August 10th, 11:59pm)		ED		W
Further evaluation (August 10th, 11:59pm)	W	W	W ED	W
Assessment of data augmentation benefits (August 10th, 11:59pm)			W	
Documentation (August 3rd, 11:59pm)	W	W	ED	
Model optimizations and code performance (August 3rd, 11:59pm)			W	ED
Compile results for presentation and report (August 3rd, 11:59pm)	W	W	W	W

Table 2: Project Training and Testing Task Breakdown

Presentation	Eric Liu	Jason Zhang	Nicholas Biancolin	Yanni Alevras
Presentation Brainstorm (August 5th, 11:59pm)	W	W	W	W
Problem - slides (August 5th, 11:59pm)		ED	W	
Data Processing - slides (August 5th, 11:59pm)	W			ED
Model - slides (August 5th, 11:59pm)		W	ED	
Result - slides (August 5th, 11:59pm)	ED			W
Slides Editing (August 5th, 11:59pm)	ED	ED		
Individual Practice (August 7th, 11:59pm)	W	W	W	W
Group Practice (August 7th, 11:59pm)	W	W	W	W
Record Presentation (August 7th, 11:59pm)	W	W	W	W
Editing (August 10th, 11:59pm)			W	ED

Table 3: Presentation Task Breakdown

Project Final Report	Eric Liu	Jason Zhang	Nicholas Biancolin	Yanni Alevras
Latex Formatting (August 12th, 11:59pm)	W	W	ED	
Introduction (August 7th, 11:59pm)		ED	W	W
Illustration (August 7th, 11:59pm)	W	ED		
Background and Related Work (August 7th, 11:59pm)		W	ED	ED
Data Processing (August 7th, 11:59pm)		W		ED
Architecture (August 7th, 11:59pm)	ED		W	
Baseline Model (August 7th, 11:59pm)	ED	ED		W
Qualitative Results (August 7th, 11:59pm)	W	ED		
Quantitative Results (August 7th, 11:59pm)		W		ED
Evaluation of Model (August 7th, 11:59pm)	W		W ED	ED
Discussion (August 7th, 11:59pm)	ED	W		W
Ethical Considerations (August 7th, 11:59pm)		W		ED
Project Difficulty (August 7th, 11:59pm)	W	ED		
Editing (August 12th, 11:59pm)	ED		ED	W
Final Proofread (August 14th, 11:59pm)	W	W	W	W

Table 4: Project Final Report Task Breakdown

3 Notable Contribution

3.1 Data Processing

The dataset contains 509 species. As mentioned in our project proposal, we grouped by genus, to ensure a larger amount of data for our training/testing set. We then kept the top 15 genera with the largest datasets. Genus grouping was done by iterating through each species in the dataset and moving the images to a new genus subdirectory. We then sorted the subdirectories by size and kept the top 15.

To increase the amount of data in our dataset, we used data augmentation techniques described by **?**. We created a copy of images applying flips, rotations (90, 180, 270 degrees), Gaussian noise, and random erasing (of black rectangular sections). These manipulations were chosen as non-destructive methods that would not alter the image's quality or class.

As mushrooms can be identified based on visual traits, preserving key details was necessary to differentiate between genera. In addition, our model is agnostic to colour, so colour space transformations are not necessary and would add duplicate data in the dataset.

Flips and rotations preserve image class. Gaussian noise helps the model focus on the most robust features of the image. Random erasing is important for training with data that varies in resolution. This dataset has mostly high quality images, but some are lower resolution. This method was added for a similar reason as Gaussian noise, but instead to train the model to identify more distinct features of one genus to another (**?**).

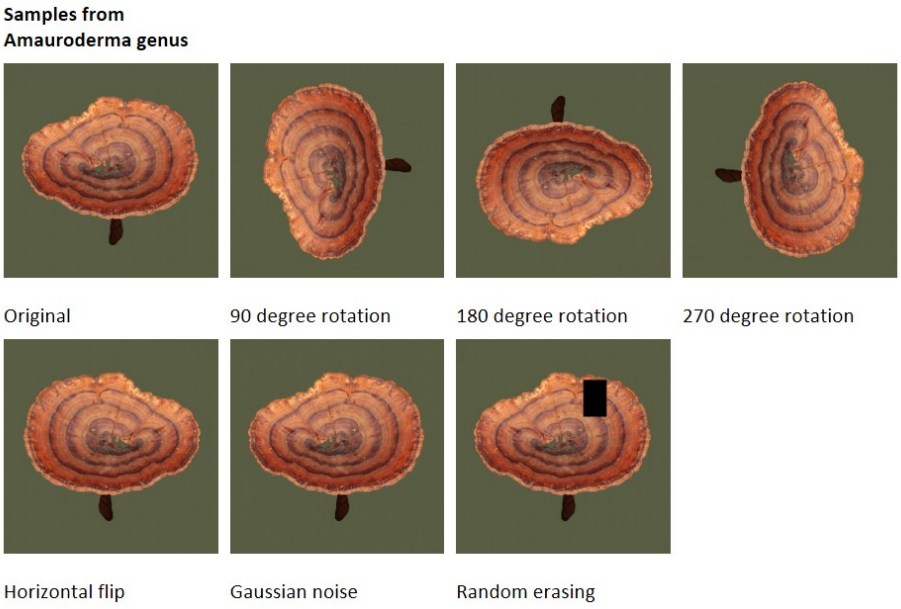

Figure 1: Data augmentation techniques applied to a sample image

To test the model on unseen data in the final steps of this project, we will use images from the same genera obtained from Wikimedia Commons.

The largest challenge for the data processing was determining how much original data is needed and how much augmentation was necessary to increase the amount of data, including the augmentation techniques. As a whole, the top 15 genera range between 200 and 1000 images, with data augmentation adding an additional seven times the data. A future challenge in this area would be determining which combinations of data augmentation methods will work the best for our model.

3.2 Baseline Model

Our baseline model is a random forests classifier trained with scikit-learn. As mentioned in our proposal, decision trees like random forest are generally well-suited for multiclass problems (**?**).

We follow similar processing steps as the primary model, in the following order: a test-train split of 80%/20%, feature extraction, training, and performance assessments. The model uses the same dataset as the primary model with the exclusion of augmented data and additional restrictions on image size (300 x 300) and number of colour channels. The restriction on image size is primarily to reduce training complexity and time.

A conversion to grayscale is necessary to facilitate feature extraction with histogram of oriented gradients (HOG) (**?**), computed with scikit-image. As models like random forest do not automatically perform feature extraction (in comparison to deep convolutional networks, which our primary model uses), this step is done to provide the model with a set of features to learn from. We compute HOG features on both the training and testing set, in line with the process outlined by **?**. We use default hyperparameters for training, with the exception of the number of estimators, which is set to 500, which **?** suggests is an optimal amount of trees for increasing accuracy.

Training takes approximately 1 minute on a CPU. This is significantly faster than the primary model, which takes 9.5 minutes.

The model has a training accuracy of about 54%, roughly 10% worse than the primary model. This is expected, as random forests are generally less accurate than deep learning models for image classification tasks. We also used minimal hyperparameter tuning, which could have otherwise

improved the model's performance. A more complicated architecture, like an ensemble model with convolutional feature extraction could have also improved model performance (**?**).

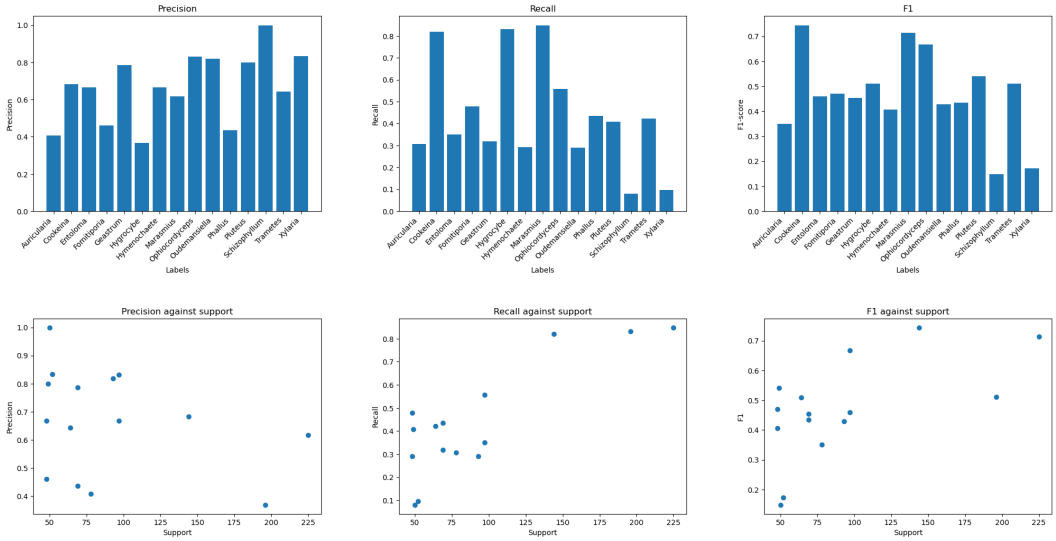

Figure 2: Precision, recall, and F1-score for the baseline model

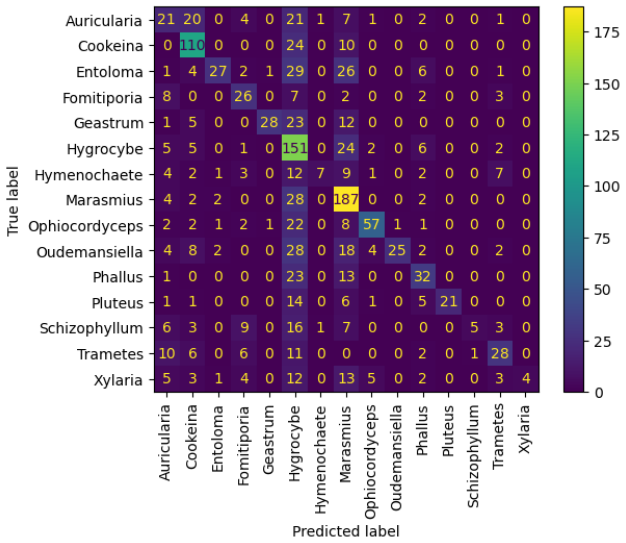

Figure 3: Random forest confusion matrix

The precision, recall, and F1-score are plotted in Figure 2. We have an average precision of 0.67, recall of 0.44, and F-score of 0.47. Notably, the recall and F-score data take a similar shape, which lower-support classes on average performing worse in recall and F-score compared to higher-support classes, likely due to less data to learn from.

Our highest precision is 1.0 with Schizophyllum, a class with a low support number. Notably its corresponding recall is 0.08, suggesting a high number of false negatives. This trend follows with other low-support classes. Our highest recall is 0.85 with Marasmius, the highest support class. A larger support is generally correlated with a higher recall.

Notably, the model disproportionately predicts more false positives of Hygrocybe than any other class, as reflected in the confusion matrix's predicted labels. This may suggest feature similarity between Hygrocybe and other classes, or a lack of distinguishing features in the training data.

We faced several issues with training. We used Intel's x86-64 architecture-specific optimizations to reduce training time (**?**), and on a few instances had memory allocation issues during training.

3.3 Primary Model

Our CNN model consists of two main sections: a non-tunable transfer learning and a tunable convolution and fully connected layers section.

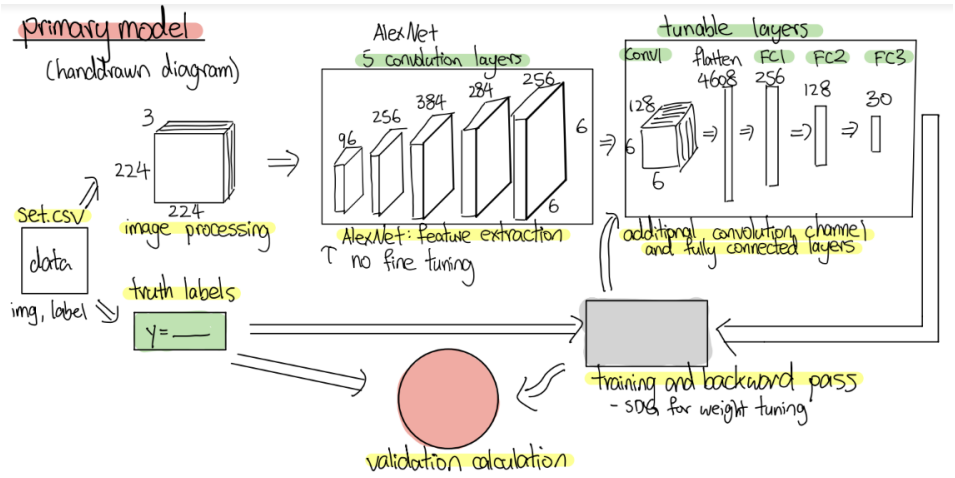

Figure 4: Model Structure and Tensor Sizes

3.3.1 Non-Tunable Section

Our team uses AlexNet for its high-level feature extraction. AlexNet processes a $3 \times 244 \times 244$ input image and the feature extraction outputs a $256 \times 6 \times 6$ tensor (**?**). There are five convolutional layers and three pooling layers (**?**), the order of the layers is shown on Figure 4. Since the model needs to differentiate between mushrooms with very similar appearances, AlexNet excels in extracting the fine features that set them apart (**?**).

3.3.2 Tunable Section

To make the model specific to our team's project, the team uses one additional convolutional layer, outputting a $128 \times 6 \times 6$ tensor. After the additional convolutional layer, the output gets flattened and passed through three fully connected layers with ReLU activation functions in between. The fully connected layers turned the size from 4608 to 256, then to 128, and lastly to 30, matching the number of output classes the team decided for the model.

In total, there are $5 + 3$ layers in the non-tunable section and $1 + 3$ layers in the tunable section, making our class structure 12 layers in total.

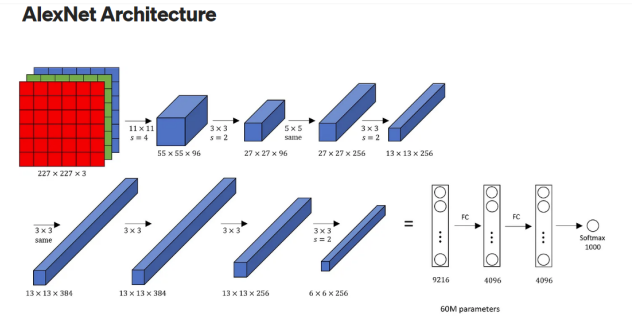

Figure 5: Class Structure: AlexNet (**?**)

3.3.3 CALCULATION OF PARAMETERS

Number of parameters for the AlexNet structure:

$$\begin{aligned}
\text{Conv1} &= 3 \times 11 \times 11 \times (96 + 1) = 35,271 \\
\text{Conv2} &= 96 \times 5 \times 5 \times (256 + 1) = 616,800 \\
\text{Conv3} &= 256 \times 3 \times 3 \times (384 + 1) = 886,080 \\
\text{Conv4} &= 384 \times 3 \times 3 \times (384 + 1) = 1,310,720 \\
\text{Conv5} &= 384 \times 3 \times 3 \times (256 + 1) = 887,232
\end{aligned}$$

3.3.4 NUMBER OF PARAMETERS FOR THE TUNABLE SECTION:

$$\begin{aligned}
\text{Conv1} &= 256 \times 3 \times 3 \times (128 + 1) = 297,216 \\
\text{Fc1} &= 4608 \times (256 + 1) = 1,183,296 \\
\text{Fc2} &= 256 \times (128 + 1) = 33,024 \\
\text{Fc3} &= 128 \times (30 + 1) = 3,968
\end{aligned}$$

The total number of parameters is 5,273,927, the number of trainable parameters is only 1,517,504. This ensures the training time for our models is feasible, allowing the team to focus on more epochs and more variations using data augmentations in the future.

At the start, the team pushed all images into the feature extraction part of AlexNet, converting data into tensors. We randomly split the data into a 75%, 15%, and 10% ratio for training, validation, and testing.

For the current best result, we used a batch size of 36, learning rate of 0.007, and 15 epochs. We chose cross entropy loss for the loss function as we want the model to classify the image into one of the 30 classes. For the optimizer, the group decided on stochastic gradient descent (SGD).

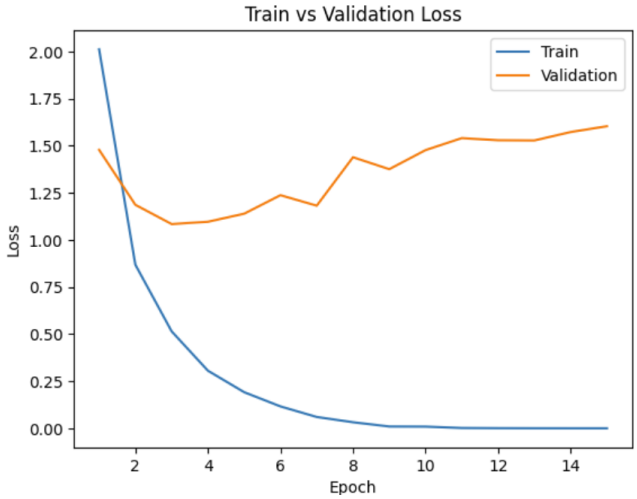

Figure 6: Validation Loss

The team's training graph with the specified hyper-parameters is shown in Figure 6. It indicates that the model is overfitted quickly. To tackle this issue, the team aims to implement regularization and drop off in the future.

Testing Accuracy:

- Epoch 4: Test Classification Accuracy: 64.01%
- Epoch 8: Test Classification Accuracy: 65.12%

From the above graph, the team chose epoch 4 and epoch 8 and did accuracy testing, using the testing data. The testing accuracy is at a good starting point considering the model must classify an image into one of thirty classes. The accuracy can be improved using various techniques:

- The current data used in the training is not augmented. The data augmentation functions are completed, but not used currently to save runtime for the training loop. The team will add the augmented data as part of training in the future.
- Further hyperparameters fine tuning.
- Implementing regularization and drop off to reduce the quick overfitting seen in the validation graph.