

# 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

- motivations behind project - 87- mushrooms are a common type of plant that are difficult to identify, especially since some are edible and some are poisonous

- goal of project - Create a deep learning model that can classify mushrooms into their respective species

- why deep learning is a reasonable approach - mushrooms have a lot of visual features that can be used to classify them - deep learning models have been shown to be effective at image classification tasks

## 2 INDIVIDUAL CONTRIBUTIONS AND RESPONSIBILITIES

- How team is working together

-project management software used to communicate/track results

-detailed list of what everyone has worked on, and what they will be working on

Yanni:

For this milestone, Yanni contributed to the data augmentation portion of this project. Using

Eric:

Rough Draft for Primary Model section of the report. handdrawn diagram of structure. Data Separation and Apply Transfer Learning: alexNet. Class Structure, Training Function, Accuracy function. Validation Loss and Graphs.

|       |      |      |       |
|-------|------|------|-------|
| Yanni | Nick | Eric | Jason |
|-------|------|------|-------|

## 3 NOTABLE CONTRIBUTION

### 3.1 DATA PROCESSING

The dataset contains # species. Stated in our project proposal, due to some low amount of sample imaging for some species we decided to group by genus. Any genus with under # samples would be removed. This narrowed us down from 509 species, and # genus, to 15 genus. To artificially increase the amount of data in our dataset, we used data augmentation, creating a copy of each sample with a horizontal flip, 90 degree rotation, 180 degree rotation, 270 degree rotation, gaussian noise, and random erasing (small black rectangles). These methods were preferred over others such as kernel

filters, lowering the quality of an image. Since these mushrooms can be identified based on specific visual traits found within their genus, high detail in the training images are necessary to allow for differentiation between the different genera. Due to this, prioritizing the quality of the image was necessary. Some simple methods were the flip and rotations, which kept the same image, but just made the model look at it a different way. Gaussian noise was added ..... Random erasing was added to

### 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 (Gall et al., 2012).

We follow similar processing steps as the primary model, in the following order: a test-train split of 80%/20%, feature extraction with HOG, 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) (Abouelnaga et al., 2018), 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 Shafi (2023). We use default hyperparameters for training, with the exception of the number of estimators, which is set to 500, which Xi (2022) suggests is an optimal amount of trees for increasing accuracy.

Training takes approximately 4 minutes on a CPU. This is significantly faster than the primary model, which takes

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 (Xi, 2022).

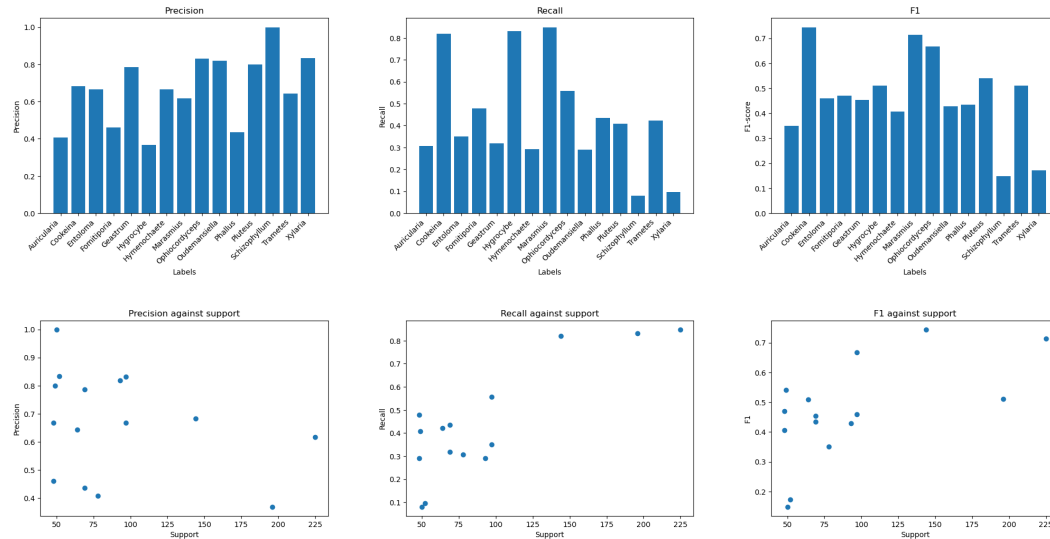


Figure 1: Precision, recall, and F1-score for the baseline model

The precision, recall, and F1-score are plotted in the following figure. 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 (Schlimbach et al., 2023), and on a few instances had memory allocation issues during training.

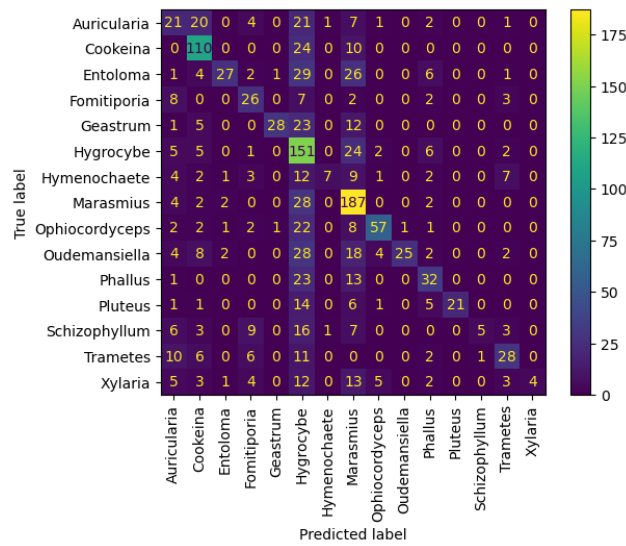


Figure 2: Random forest confusion matrix

### 3.3 PRIMARY MODEL

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

#### 3.3.1 NON-TUNABLE SECTION

The team uses AlexNet for its high-level feature extraction. AlexNet processes a  $3 \times 224 \times 224$  input image and the feature extraction outputs a  $256 \times 6 \times 6$  tensor (Bangar, 2022). There are five convolutional layers and three pooling layers (Bangar, 2022), the order of the layers is shown on the hand-drawn diagram above. 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 the 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.

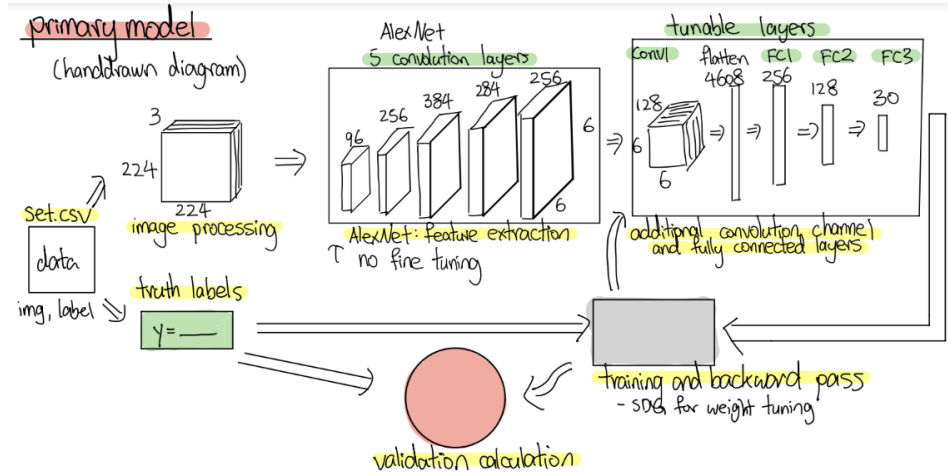


Figure 3: Model Structure and Tensor Sizes

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.

#### AlexNet Architecture

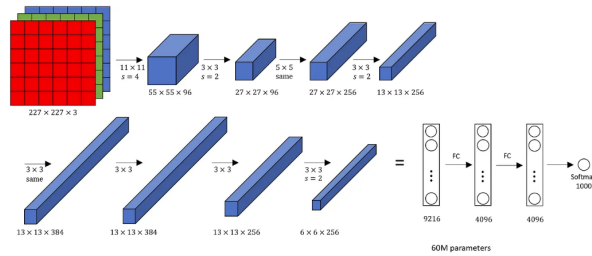


Figure 4: Class Structure: AlexNet (Bangar, 2022)

### 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 our 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).

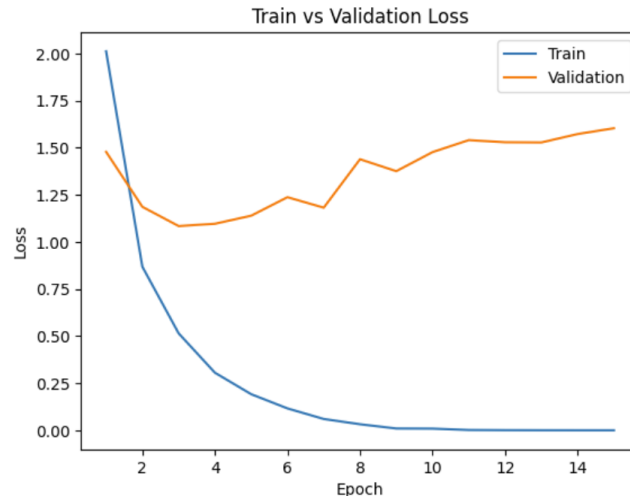


Figure 5: Validation Loss

The team's training graph with the specified hyper-parameters is shown above. 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 epochs 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.

#### REFERENCES

- Yehya Abouelnaga, Balamurugan Thambiraja, and Kamel Guerda. Object Detection: Histogram of Gradients and Random Forest. *Technical University of Munish*, January 2018.
- Siddhesh Bangar. AlexNet Architecture Explained. *Medium*, June 2022.

Juergen Gall, Nima Razavi, and Luc Van Gool. An Introduction to Random Forests for Multi-class Object Detection. In Frank Dellaert, Jan-Michael Frahm, Marc Pollefeys, Laura Leal-Taixé, and Bodo Rosenhahn (eds.), *Outdoor and Large-Scale Real-World Scene Analysis*, pp. 243–263, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34091-8.

Frank Schlimbach, Alexander Andreev, Kirill Petrov, and Oleksandr Pavlyk. Intel(R) Extension for Scikit-learn, 2023.

Adam Shafi. Random Forest Classification with Scikit-Learn. *Datacamp*, 2023.

Erhui Xi. Image Classification and Recognition Based on Deep Learning and Random Forest Algorithm. *Wireless Communications and Mobile Computing*, 2022(1):2013181, January 2022. doi: 10.1155/2022/2013181.