Classification of Fungi Species: A Deep Learning Based Image Feature Extraction and Gradient Boosting Ensemble Approach

Karthik Desingu[1], Anirudh Bhaskar[1], Mirunalini Palaniappan[1], Eeswara Anvesh Chodisetty[1] and Haricharan Bharathi[1]

[1]*Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai, India*

Abstract

Deep learning has been successful in a variety of challenging image classification tasks, characterized by complex and large datasets. Ensemble learning further improves model performance by inferring compound decision boundaries in the feature space, and assorting importance to the most representative features to effectively discriminate between image classes. This paper details a deep learning based feature extraction and subsequent boosting ensemble approach for fungi species classification. The proposed workflow leverages state-of-the-art deep learning architectures such as ResNeXt and EfficientNet among others, trains them by transfer learning onto a fungi image dataset for feature extraction, and finally integrates the output representation vectors with geographic metadata to train a gradient boosting ensemble classifier that predicts the fungi species. The authors trained multiple deep learning architectures, assessed their individual performance and selected effective feature extraction models. Multiple experiments were performed in choosing these models, and to subsequently perform hyperparameter tuning to train the boosting classifier. The approach attained a maximum macro-averaged F1-Score of 48.96% on the test data. The corresponding validation F1-Score and Accuracy scores were 50.22% and 85.11%, respectively. Furthermore, the best model exhibited more confident predictions than its inferior counterparts, indicating improved inter-class discerning.

Keywords

Ensemble Learning, Convolutional Neural Networks, Gradient Boosting Ensemble, Metadata-aided Classification, Image Classification, Transfer Learning

1. Introduction

This paper presents the participation of Sri Sivasubramaniya Nadar College of Engineering, India for the FungiCLEF-2022 [1] challenge for fungi species identification held jointly by LifeCLEF-2022 [2, 3] lab of the CLEF 2022 conference and the FGVC9 workshop organized in conjunction with CVPR 2022 conference.

In India, the infection rate of fungi-based diseases such as Mucormycosis was 45374, with a mortality rate of over 4300. Fungi are also essential to the survival of many organisms and sustaining the food cycle as they attract attention as predators of invertebrate animals, act as

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

karthik19047@cse.ssn.edu.in (K. Desingu); anirudh2010094@ssn.edu.in (A. Bhaskar); miruna@ssn.edu.in (M. Palaniappan); eeswaraanvesh2010038@ssn.edu.in (E. A. Chodisetty); haricharan2010267@ssn.edu.in (H. Bharathi)

catalysts, and so on. Manual identification of such species is no easy task as it involves running through almost 5.1 million species.

The aim of the FungiCLEF challenge is to assist citizens, scientists, and nature enthusiasts by automatic recognition of fungi species based on non-standardized photographs, habitat, substrate, and location information that are available as a part of the meta-data.

2. Related work

The usage of convolutional neural networks in the classification and identification of fungi species is a concept that has been floating around for quite a while. The integration of computer vision and machine learning with the development of more efficient algorithms will undoubtedly be a hotspot for future studies in the context of the mushroom industry [4].

2.1. FungiCLEF

A study in 2021 in the Turkish Journal of Computer and Mathematics Education by Sukanya S. Gaikwad [5] used a convolutional neural network to classify various species of fungi, specifically on fungi affected apple leaf diseases. They obtained an accuracy of 88.90% in the classification of fungi.

In a study by M.E. Mital, pre-trained deep learning models were employed in classifying 9 kinds of Aspergillus. The methodology comprised of preprocessing, deep-learning and performance evaluation. This study achieved a 93.33% testing accuracy proving that the transferred knowledge is accurate, compatible and reliable [6].

A submission to the Danish Fungi 2020 by Lukáš Picek showed that experiments using convolutional neural networks and the recent Vision Transformers showed that ViT achieves results superior to convolutional neural network baselines with 80.45% accuracy and 0.74 macro F1 score, reducing the convolutional neural network error by 9% and 12% respectively. By incorporating metadata into the decision tree process, the error rate came down significantly by 15% [7].

A study by Krzysztof Przybył to evaluate rapeseed samples obtained in the process of storage experiments with different humidity and temperature conditions, the classification was carried out based on the different levels of contamination with filamentous fungi. The classifiers that were compared were devised on the basis of the environments TensorFlow (deep learning) and Statistics (machine learning). The lowest classification error of 14% for the test set, 18% classification error for Multi-Layer Perceptron Networks (MLPN), and 21% classification error for Radial Basis Function Networks (RBFN), in the process of recognizing mold in rapeseed with the use of convolutional neural networks [8].

Finally, a paper written by Pranjal Maurya focused on developing a method for classification of mushroom using its texture feature, which is based on the machine learning approach. The performance published was 76.60% by using support vector machine classifier, which was found better with respect to the other classifiers like k-nearest neighbors, Logistic Regression, Linear Discriminant, Decision Tree, and Ensemble classifiers [9].

3. Methods

Convolutional neural networks are deep learning algorithms that use 3 major types of layers, namely: convolutional layer, pooling layer, and fully connected layer.

In our approach to this problem, we have used different convolutional neural network-based approaches like ResNet101, ResNet50, and EfficientNetB0. The workflow was implemented using Python v3.9.0 and primarily uses TensorFlow V.2.4.0.

3.1. Dataset

The training dataset of the FungiCLEF-2022 challenge consists of 295938 training images spread over 99% fungi and 1% protozoa belonging to 1604 species observed mostly around Denmark. This challenge, being a new entrant into the foray of LifeCLEF challenges, unfortunately cannot be compared to previous year's datasets. The dataset includes 30 different country codes 5 localities with just over 1% distribution among Store Hareskov and Hegedal and the remaining 98% over the other localities. All the methods used involve convolutional neural networks. The models used are ResNet101 and EfficientNetB0.

3.2. Image Preprocessing

The photos were first placed into an input sequencer, where the outlier images were found. The images were analyzed, and it was discovered that they were of varying sizes and scales. As a result, image augmentation was used to convert all of the photographs to RGB and scale them to a standard size of 800 by 600 pixels $224 \times 224 \times 3$ dimensions [10].

The images were linearly normalized to values between 0 and 1 to reduce the effect of irrelevant characteristics in the context of the needed task, such as variance in lighting conditions among the shots. To make the model more general, immune to the impact of positional and orientation-based bias, and prevent memory by improving image diversity, transformations such as scale and rotation, as well as contrast and saturation variations, were induced on the model inputs. The aforementioned modifications were utilized to augment the input photos with RandAugment [11]. RandAugment has two parameters: the number of augmentation transformations to apply in order (N) and the magnitude of all modifications (M). N=3 and M=4 were chosen as the values for the ResNet model by experimentation.

3.3. Feature Extraction

Feature extraction is a method prescribed to transform raw data into numerical features. It is preferred as applying machine learning algorithms directly on the raw data yields poorer results. High data rates and information redundancy can be cut down using feature extraction. Since the task this time involved a large data set of over 110GB of training images along with geographic metadata, the data rate is very high. Thus, redundant features are an obvious by-product. Feature extraction thus helped extract only the unique features that describe the images, such as shapes and edges.

3.4. Deep Learning Models Considered

The feature extraction models for fungi image was selected after experimenting with multiple deep-learning architectures.

ResNet101 [12] is a well-known convolutional neural network model that was introduced in 2015. This model addresses the degradation problem, which asserts that as network depth grows, accuracy becomes saturated and subsequently rapidly declines. ResNet solves the degradation problem by using shortcut connections that bypass one or more layers, which was inspired by the Highway network [13], which employed gated shortcut connections to manage the flow of information in the shortcut.

EfficientNet(s) [14] are a class of convolutional neural networks that were built in 2019. It's a small-scale architecture with about 11 million trainable parameters. It was created with the help of a multi-objective neural network that prioritized precision and floating point operations. It supports compound scaling while maintaining network balance across all dimensions. It employs an inverted bottleneck as well as a depth-wise convolutional network that includes squeeze and excitation operations. It employs MBConv blocks [15] that serve as Inverted Linear BottleNeck layers. These layers use Depth-Wise Separable Convolution operations. The model complexity of the variants increases from B0 to B7. The authors experimented with B0, B4 and B6 variants to scale the model complexity and find the best suited intricacy for the fungi dataset.

Another popular convolutional neural network model, ResNeXt101 [16], is very similar to the ResNet101 model. ResNet101 features a lot of sequential layers, while ResNeXt101 contains a lot of parallel stacking layers. Like the Inception module [17, 18], it uses a split-transform-merge technique. ResNeXt shares hyperparameters for all the blocks, unlike the Inception module which has different filters and sizes for each block.

3.5. Gradient Boosting Ensemble Classifier

Ensemble methods are techniques used to counter the high variance produced by a single neural network by adding an inherent bias that is obtained from multiple models. They tend to give higher accuracy than their resident models. Boosting is an ensemble technique in which the new models are sequentially added to the existing features to correct the errors.

In our method, an ensemble of the models trained off the aforementioned architectures were conjoined with metadata features — country, three-level-precise location information, substrate and habitat. These metadata features along with the 4096 features extracted from each of the two models, was concatenated and fed to a gradient boosting classifier.

The team used the XGBoost library package [19] for implementing the boosting classifier. XGBoost is a high-speed and high-performance implementation of gradient boosted decision trees. XGBoost's superior execution speed was factored-in while choosing from all the available implementation libraries.

3.6. Prediction of *Out-of-the-Scope* Classes

The FungiCLEF-2022 test dataset is known to contain images of Fungi that belong to classes, not exposed to the model as part of the training set. From a practical perspective, predicting a previously *unseen* observation as an *unidentified* class is as salient as assorting a *seen* class into

its corresponding class. Failure to do either, is misleading. It is, therefore, imperative that such *out-of-the-scope* observations are identified and flagged.

To manage such observations during prediction, the authors have adopted a *prediction confidence thresholding* strategy. The gradient boosting classifier employs the softmax activation function in its classification layer. Consequently, it outputs a discrete probability distribution — P, over all the 1604 *seen* classes, such that Equation 1 is satisfied.

$$\sum_{i=0}^{1603} P(X = i) = 1 \tag{1}$$

where, i denotes the class id of the known classes of fungi species, and $P(X = i)$ denotes the probability that a specific observation belongs to class i.

The most predominantly adopted strategy to predict the observation's class from this distribution is the Maximum Likelihood Estimate (MLE) [20] of this probability distribution, with respect to the training data — predict that the observation belongs to class k, such that $P(X)$ has a global maxima at $X = k$. The same strategy is adopted by the authors, with one nuance.

The prediction probability associated with a class represents the model's confidence about an observation belonging to that class. If the value at the global maxima of $P(X)$ is not strikingly higher than the value of $P(X)$ at other points, it suggests that the model is not very confident about its prediction. Extending this idea, the proposed *prediction confidence thresholding* method classifies an observation as *out-of-scope* if the maximum prediction confidence for the observation falls below a specific threshold value. One such threshold value is arrived at for each trained ensemble model, by adopting a qualitative, experiment-based method. A histogram of maximum prediction probabilities of the model, for each observation in the test set, is plotted. Subsequently, multiple confidence values are sampled from lower end of the x-axis of the histogram, and tried for prediction to choose an optimal value. The method is described in greater detail, under the subsequent section on Experiments.

4. Experiments

Transfer learning from the weights obtained during training with the ImageNet data set [21], and fine-tuning on the FungiCLEF-2022 training data were used to train the model. The prediction accuracy of the models was tracked throughout training in order to select the collection of feature extractors to employ for ensembling later. ResNeXt101 and EfficientNetB4 were chosen as feature extractors for ensembling based on observed network performance. The ensembling was performed using the XGBoost gradient boosting library package.

The forward propagation is described during training and prediction. Each observation is made up of numerous fungus photos, as well as contextual geographic information like nation and exact area where the photograph was taken on four layers, as well as specific attributes like substrate and habitat. Each image in an observation is preprocessed before being fed through the two feature extraction networks to generate two 4096-element-long representation vectors. These vectors are combined with numeric encoded nation, location at three-level precision, substrate, and habitat metadata for the image to produce a final vector with a size of 8198. The numeric encoding was achieved with the help of Label Encoders from the scikit-learn

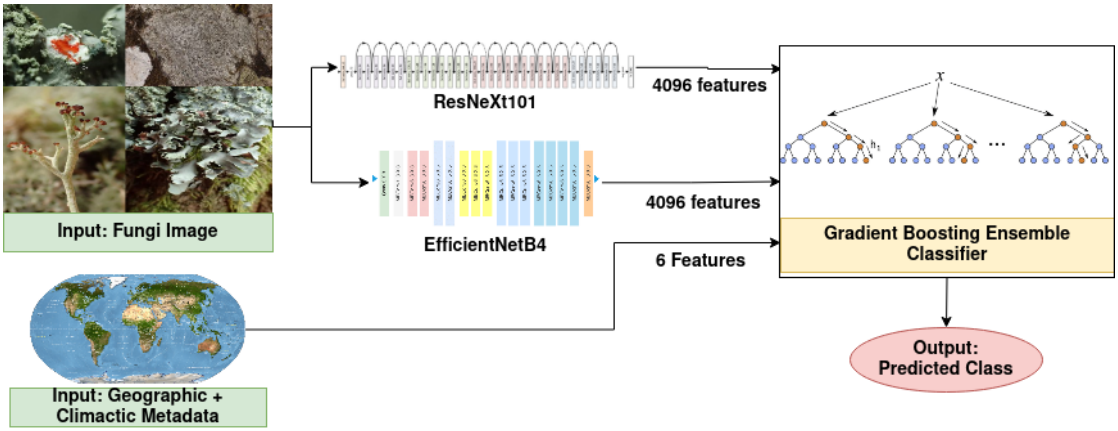

Figure 1: Prediction workflow used in the proposed work for the classification of fungi species from their images and metadata using an ensembling gradient boosting classifier. *Note*: Model architectures depicted are illustrative only and NOT accurate representations of the underlying network design.

[22] library wherein, the categorical country codes available as training data are converted to integral class labels that can be fed as input to the feature extraction neural network. The boosting ensemble classifier is fed these 8198 features to generate a probability distribution over all potential fungi species classes. This workflow is depicted in Figure 1. Categories in any metadata feature that are present in the testing data, but not encountered in the training set are encoded as 0, a commonly adopted strategy to deal with unseen categorical data in deep learning methods.

To obtain a single aggregate distribution of probabilities over all classes, the corresponding class probability values collected for each image in an observation are averaged. As a result, each observation has only one unique probability distribution. The classification label is assigned to the class that has the highest aggregate probability value.

The subsequent sections describe the model training experiments and parameters.

4.1. Model Training

The details of the model training process, performed through transfer-learning is presented in this section. A summary of the parameters used for model training is tabulated in Table 1.

Table 1
Model training parameters used to train each of the convolutional neural networks used for this classification task.

Parameter	Optimizer	Learning rate	Batch Size	Epochs
ResNet101	Adam	$3e{-}5$	32	45
EfficientNetB0	Adam	$3e{-}2$	32	50
EfficientNetB4	Adam	$1e{-}3$	32	60
EfficientNetB6	Adam	$3e{-}3$	32	45
ResNeXt101	Adam	$3e{-}3$	32	25

4.1.1. ResNet101

The feature extraction layers of ResNet101 were trained with a two-step classification block, comprising two dense blocks with 4096 and 1604 neurons respectively. The extracted features were percolated through a flatten layer to obtain, before feeding to the classification block. In addition, a dropout layer was added after the dense layer to avoid overfitting. Dropout rates between 0.30 and 0.70 were experimented and set to 0.55 in the final version of the model. The model was trained with the Adam optimizer at an initial learning rate of $3e{-}5$. It was back-propagated using the Categorical Cross-Entropy (CCE) loss. For feature extraction, the output of the first dense layer was used to produce a feature vector of 4096 elements. During training, the model's prediction accuracy was tracked to later choose the feature extractor to use for ensembling.

4.1.2. EfficientNetB0

EfficientNetB0 was also trained with a two-step classification block, comprising two dense blocks with 4096 and 1604 neurons, respectively. The extracted features were percolated through a flatten layer to obtain, before feeding to the classification block. The dropout layer added after the dense layer for this network was experimented between 0.30 and 0.70 and fixed at 0.30. The model was trained using the Adam optimizer at an initial learning rate of $3e{-}2$. It was back-propagated using the Categorical Cross-Entropy (CCE) loss. For feature extraction, the output of the first dense layer was used to produce a feature vector of 4096 elements.

4.1.3. EfficientNetB4

EfficientNetB4 was trained with a two-step classification block, also comprising two dense blocks with 4096 and 1604 neurons, respectively. While the final layers of the architecture are the same as the other EfficientNet models, the dropout layer added after the dense layer for this network was experimented between 0.30 and 0.70 and fixed at 0.35 for this model. The model was trained using the Adam optimizer at an initial learning rate of $1e{-}3$. A Categorical Cross-Entropy (CCE) loss was used to back-propagate, and a 4096-sized vector was extracted as a feature-representation.

4.1.4. EfficientNetB6

EfficientNetB6 followed the same approach — two dense blocks with 4096 and 1604 neurons, dropout layers, and CCE loss for back-propagation, and optimized with Adam — with differences only in the hyperparameters. The dropout layer after the dense layer was fixed at a dropout rate of 0.45 after experiments. The learning rate was initially set at $3e{-}3$. A 4096-sized vector was extracted as feature-representation.

4.1.5. ResNeXt101

The ResNeXt101 architecture was augmented, following the same strategy as the aforementioned models i.e. by adding a two-step classification block of 4096 and 1604 neurons, respectively. Here, the dropout after experimenting, was set at 0.60 — a significantly high rate due to excessive

early-onset overfitting. Adam optimizer, along with CCE loss was used for training, with an initial learning rate of $3e{-}3$.

4.2. Loss, Metrics, Activation and Optimizer Used

The loss functions, activation functions, optimizers, and evaluation metrics used in the model training experiments are listed under this section, along with their parameterizations and equations.

4.2.1. Adam Optimizer

Adam [23] is a stochastic optimization method which is used on gradient descent and maintains a single learning rate (alpha) throughout training. Adam combines the advantages of the Adaptive Gradient Algorithm and Root Mean Square Propagation. Unlike the Root Mean Square Propagation, in which the first moment about the mean is used, Adam uses the average of the second moments about the mean too. In effect, Adam provides an optimization algorithm that can handle sparse gradients on noisy problems, by maintaining a per-parameter learning rate.

4.2.2. Categorical Cross Entropy Loss

The categorical cross entropy is a measure of the difference between two discrete probability distributions. It is calculated using the formula in Equation 2.

$$Loss = -\sum_{i=1}^{n} y_i log t_i, \tag{2}$$

where, $y_i$represents the corresponding target value for t_i the scalar model output.

4.2.3. Softmax Activation

The Softmax activation function is used at the end of the output layer to produce the posterior probability distribution over all classes, based on equation 3. Softmax is essentially a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. In effect, it normalizes the outputs, converting them from weighted sum values into probabilities that sum to one.

$$softmax(z_i) = \frac{exp(z_i)}{\Sigma_j exp(z_j)} \tag{3}$$

where, z represents the values from the neurons of the output layer. The exponential acts as the non-linear function. These values are divided by the sum of exponential values to normalize and convert them into probabilities.

4.2.4. F1-Score Metric

The F1-Score is usually calculated as the harmonic mean of precision and recall. This is concretely expressed in the equation 4.

$$F_1 = \frac{2p_s r_s}{p_s + r_s}, \tag{4}$$

$$p_s = \frac{T_p}{T_p + F_p} \tag{5}$$

$$r_s = \frac{T_p}{T_p + F_n} \tag{6}$$

where, F_1 represents the F1-score, p_s represents precision, r_s represents recall, $T_p$represents true-positive, F_p represents false-positive and F_n represents false-negative. The contest prescribed *macro-averaged F1-Score* as the evaluation metric.

4.2.5. Accuracy Metric

The accuracy score (Acc) is computed as the ratio of correct predictions to the total number samples. This is expressed in the equation 7.

$$Acc = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{7}$$

where, Acc represents the accuracy score, $T_p$represents true-positive, T_n represents true-negative, F_p represents false-positive and F_n represents false-negative.

4.3. XGBoost Ensemble Classifier

Ensembling was performed with the XGBoost classifier. A grid-search strategy6 was adopted and hyperparameters were fine-tuned [24] for optimal performance when training the XGBoost classifier. The tree's maximum depth was set to 32. The model would become more sophisticated and prone to overfitting if this parameter was increased. Because increasing this value will consume too much memory during training the deep tree, a low value of 16 was chosen.

Learning rates greater than 0.03 were found to cause rapid divergence, therefore values in the $10e{-}3$ to $10e{-}5$ range were utilized. Grid-search was carried out by altering the learning rates in this range and using decision trees ranging from 100 to 1000. To fine-tune the tree-level parameters, the combinations with the best accuracy scores were chosen.

The maximum depth of the tree is left to be selected according to the classifier's training progress and is not fixed in stone. This causes the depth to increase until the leaves are pure (i.e., all samples belong to the same class) or the minimum number of samples required to divide further has been reached. Some classes may require deeper branches to gather more information from the features due to the data set's long-tailed distribution. To prevent overfitting, a grid search over values in the range of 32 to 256 was used to set an upper limit on the number of leaves.

The learning task and accompanying learning target are then specified using objective parameters. The Softmax objective function was chosen to set up the classifier for multiclass classification, and the number of classes was set at 1604, explicitly.

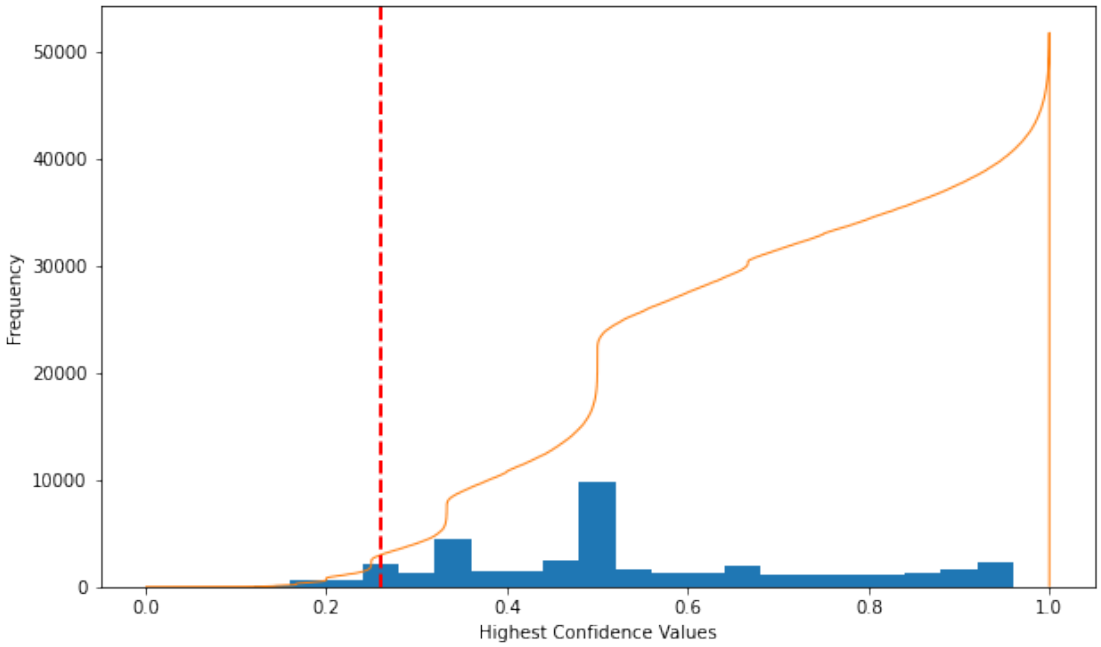

Figure 2: Histogram of *maximum prediction confidence* over the test set for the ensemble model corresponding to Submission 1 in Table 3. The *orange-colored* line-graph depicts cumulative frequency. The *red-colored dotted* line-graph depicts the 5% quantile of frequency.

4.4. Threshold Value for *Out-of-the-Scope* Classes

The authors adopted a *prediction confidence thresholding* method to handle *out-of-the-scope* classes. A threshold value is arrived at for each trained ensemble model by adopting a qualitative, trial-based method. First, a histogram of maximum prediction probabilities of the model for each observation in the test set is plotted. The histograms for the best and the worst among the top-5 post-competition submissions are presented in Figures 2 and 3, respectively.

The x-axis represents the maximum confidence values for predictions on the observations, while the y-axis tracks the frequency of these maximum confidence values. Subsequently, the x-axis point of 5% cumulative frequency is identified (denoted by the red-colored dotted-line in Figure 2). Since the proportion of *out-of-the-scope* classes is not known, multiple points — typically, 2-4 points were chosen during experiments — on the left-hand side of this 5% quantile line are chosen as threshold values, and predictions are made based on each of these threshold values.

5. Results and Conclusion

The data provided with the metadata — namely country, location at three-levels of precision, substrate, and habitat metadata — were employed as categorical features for the gradient boosting classifier. The inclusion of contextual information showed a strong impact on the classification results — the testing F1-score of the best submission improved from 15.71% to

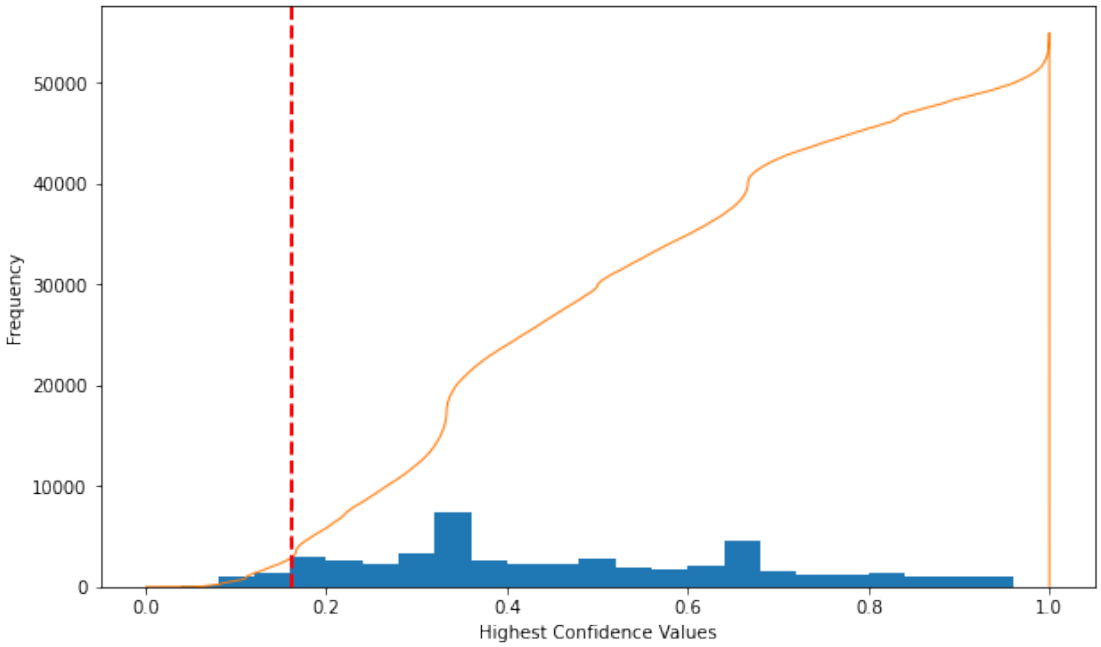

Figure 3: Histogram of *maximum prediction confidence* over the test set for the ensemble model corresponding to Submission 5 in Table 3. The *orange-colored* line-graph depicts cumulative frequency. The *red-colored dotted* line-graph depicts the 5% quantile of frequency.

16.28%. Likewise, the cross-validation accuracy and F1-score for the same model improved from 61.72% and 41.95% to 63.88% and 42.74%, upon inclusion of the geographic data.

Figure 4 represents the relative importance of the 16 most effective image features extracted using the trained neural networks, along with the six metadata features used to train the ensemble classifier. The feature importance values were normalized and scaled between 0 and 100 to realize the relative impacts. Features named as f1, f2, etc. denote features extracted from the neural networks. It is worth mentioning that f0 through f4095 denote features extracted using EfficientNetB0, while f4096 through f8191 represent the ResNeXt101-extracted features. It is evident that the country, level-0 location, and habitat information have a significant influence on classification. Further, the more specific location information — level-1 and level-2 — have not contributed as much to the classification. The substrate metadata also serves a significant impact.

After deciding the baseline architectures for ensembling — namely, EfficientNetB4 and ResNeXt101 — based on individual prediction performance, the baseline models were trained towards convergence. Following this, the multiple ensemble models were trained using the gradient boosting classifier using the two baseline models and contextual data, with different hyperparameter settings. The ensemble classifier's performance was improved over several runs, by tuning the hyperparameters of the gradient boosting classifier. The contest prescribed F1-scores macro-averaged across all classes as the evaluation metric. Model runs were evaluated on the given stratum of validation set using this metric. The metrics were evaluated as

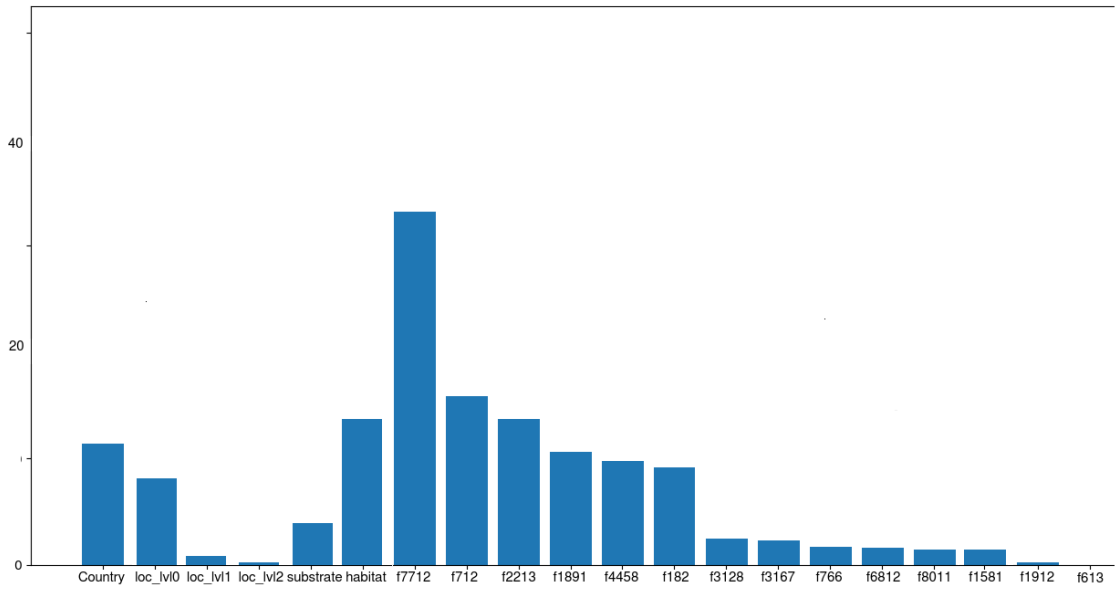

Figure 4: Relative importance on a scale of 0-100 of the 14 most impactful features extracted from the deep-learning models, along with the six metadata features used to train the classifier.

an average over the five iterations (for 5-fold cross validation) performed in each run during training.

Subsequently, the experiment described in Section 4.4 was applied on the models with top scores on the validation set to sample multiple threshold values for each model, and class predictions were made based on each threshold value. The top-5 results from this pool of predictions is summarized in Table 2 for competition submissions, and in Table 3 for post-competition evaluations.

The histograms used in the experiment for sampling threshold values for the best and worst post-competition performance are presented in Figure 2 and 3, respectively. It is worth noting that the 5% quantile — represented using the *red-colored dotted-line* — occurred at a confidence value of 0.53 for the best submission, as opposed to 0.24 for the worst model. Further, it can be observed that a majority of the confidence values occur between 0.4 and 0.6 for the best model, and between 0.2 and 0.4 for the worst — indicating a rightward shift. These observations signify that the best-performing model not only predicted a larger proportion of classes correctly, but did so with a higher prediction confidence. This is of practical importance in situations where uncertainty in predictions has serious repercussions.

Our team achieved a training accuracy of 67.12%, validation accuracy of 63.88%. The corresponding model secured an F1-score of 16.28% on the competition's test data. Our team placed 29^{st} among 40 participating teams. A complete summary of the model performance is listed in Table 2.

Based on the results, it is apparent that the inclusion of contextual geographic data for fungi

Table 2
Performance metrics of the 5 best submissions, ordered best to worst. F1-Scores are macro-averaged across classes.

Submission#	Training Accuracy	Validation Accuracy	Validation F1-Score	Test F1-Score
1	67.12	63.88	42.74	16.28
2	63.76	60.79	38.88	16.23
3	60.11	58.21	38.42	15.64
4	60.14	57.13	37.21	14.17
5	53.42	49.22	30.97	12.35

species classification has had a contributing effect. The best post-competition results are on par with the 27^{th} best submission. Furthermore, the ensembling of features extracted using multiple neural architectures, and adopting transfer learning to adapt the pretrained models to the specific data domain, looks promising. Several existing approaches have introduced metadata such as population counts of various species, more location-specific geographic data such as city, state, and climatic features such as temperature and humidity. An interesting approach is to employ class-wise probability priors to the neural networks based on such metadata [25].

On account of insufficient computing resources to complete all model training experiments in time for the large dataset of fungi images, the run submissions had to be generated before complete model convergence. Significant improvements were observed in classification accuracy after the submission deadline in the validation, as well as testing performance (through the late submission option). Submission number 3 (refer to Table 2), in particular, showed good improvements when the boosting classifier was trained further with the same hyperparameter settings. A summary of post-competition improvements in prediction results during the working notes submission phase of FungiCLEF-2022 is tabulated in Table 3.

Table 3
Performance metrics of the 5 best submissions **post-competition**, ordered best to worst. F1-Scores are macro-averaged across classes.

Submission#	Training Accuracy	Validation Accuracy	Validation F1-Score	Test F1-Score
1	86.78	85.11	50.22	48.96
2	83.37	80.56	46.78	41.54
3	78.43	77.10	40.14	37.52
4	74.81	71.82	37.13	34.63
5	73.16	70.84	34.76	33.29

It is evident that subsequent hyperparameter tuning and training have been effective — both in terms of the proportion of classes predicted correctly, as well as the confidence of these predictions. Hence, the team believes and suggests that the ensembling approach is an effective option for applying to data-intensive and high-complexity image classification

tasks that are commonly released as a LifeCLEF task. We further conjecture that training the individual models to convergence, and subsequently applying the boosting ensembler with hyperparameter tuning will culminate in a superior prediction performance, that exhausts the proposed architectures' and methodology's potential. In addition, approaches involving input image resolution variations, usage of alternative pre-trained weights [26], as well as the inclusion of custom training layers to the frozen base model when transfer learning [27] can greatly improve the quality of feature extraction. Finally, the application of image preprocessing techniques can be of significance in improving the overall model performance, particularly that of the neural networks used for feature extraction [28]. However, the authors could not expend sufficient time to systematically and exhaustively experiment with image preprocessing techniques to analyze their impact on classification performance, due to the aforementioned time and computational constraints. Further experiments will also explore these directions.

References

[1] L. Picek, M. Šulc, J. Heilmann-Clausen, J. Matas, Overview of FungiCLEF 2022: Fungi recognition as an open set classification problem, in: Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, 2022.

[2] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Durso, H. Glotin, R. Planqué, W.-P. Vellinga, A. Navine, H. Klinck, T. Denton, I. Eggel, P. Bonnet, M. Šulc, M. Hruz, Overview of lifeclef 2022: an evaluation of machine-learning based species identification and species distribution prediction, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2022.

[3] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Durso, I. Bolon, et al., Lifeclef 2022 teaser: An evaluation of machine-learning based species identification and species distribution prediction, in: European Conference on Information Retrieval, Springer, 2022, pp. 390–399.

[4] H. Yin, W. Yi, D. Hu, Computer vision and machine learning applied in the mushroom industry: A critical review, Computers and Electronics in Agriculture 198 (2022) 107015.

[5] S. S. Gaikwad, et al., Fungi classification using convolution neural network, Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12 (2021) 4563–4569.

[6] M. E. Mital, R. Ruzcko Tobias, H. Villaruel, J. M. Maningo, R. Kerwin Billones, R. R. Vicerra, A. Bandala, E. Dadios, Transfer learning approach for the classification of conidial fungi (genus aspergillus) thru pre-trained deep learning models, in: 2020 IEEE REGION 10 CONFERENCE (TENCON), 2020, pp. 1069–1074. doi:10.1109/TENCON50793.2020.9293803.

[7] L. Picek, M. Šulc, J. Matas, T. S. Jeppesen, J. Heilmann-Clausen, T. Læssøe, T. Frøslev, Danish fungi 2020-not just another image recognition dataset, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2022, pp. 1525–1535.

[8] K. Przybył, J. Wawrzyniak, K. Koszela, F. Adamski, M. Gawrysiak-Witulska, Application of deep and machine learning using image analysis to detect fungal contamination of rapeseed, Sensors 20 (2020) 7305.

[9] P. Maurya, N. P. Singh, Mushroom classification using feature-based machine learning

approach, in: Proceedings of 3rd International Conference on Computer Vision and Image Processing, Springer, 2020, pp. 197–206.
[10] P. Smith, Bilinear interpolation of digital images, Ultramicroscopy 6 (1981) 201–204.
[11] E. D. Cubuk, B. Zoph, J. Shlens, Q. V. Le, Randaugment: Practical automated data augmentation with a reduced search space, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 702–703.
[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015. URL: https://arxiv.org/abs/1512.03385. doi:10.48550/ARXIV.1512.03385.
[13] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, Advances in neural information processing systems 28 (2015).
[14] M. Tan, Q. V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks (2019). URL: https://arxiv.org/abs/1905.11946. doi:10.48550/ARXIV.1905.11946.
[15] M. Tan, Q. Le, Efficientnetv2: Smaller models and faster training, in: International Conference on Machine Learning, PMLR, 2021, pp. 10096–10106.
[16] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, 2016. URL: https://arxiv.org/abs/1611.05431. doi:10.48550/ARXIV.1611.05431.
[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, 2014. URL: https://arxiv.org/abs/1409.4842. doi:10.48550/ARXIV.1409.4842.
[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
[19] T. Chen, C. Guestrin, XGBoost, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016. URL: https://doi.org/10.1145%2F2939672.2939785. doi:10.1145/2939672.2939785.
[20] I. J. Myung, Tutorial on maximum likelihood estimation, Journal of mathematical Psychology 47 (2003) 90–100.
[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.
[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.
[23] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014. URL: https://arxiv.org/abs/1412.6980. doi:10.48550/ARXIV.1412.6980.
[24] A. Anghel, N. Papandreou, T. Parnell, A. De Palma, H. Pozidis, Benchmarking and optimization of gradient boosting decision tree algorithms, arXiv preprint arXiv:1809.04559 (2018).
[25] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, W. Xu, Cnn-rnn: A unified framework for multi-label image classification, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2285–2294.
[26] A. Joly, H. Goëau, S. Kahl, B. Deneu, M. Servajean, E. Cole, L. Picek, R. Ruiz de Castañeda, I. Bolon, A. Durso, et al., Overview of lifeclef 2020: a system-oriented evaluation of

automated species identification and species distribution prediction, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2020, pp. 342–363.

[27] M. Zhong, J. LeBien, M. Campos-Cerqueira, R. Dodhia, J. L. Ferres, J. P. Velev, T. M. Aide, Multispecies bioacoustic classification using transfer learning of deep convolutional neural networks with pseudo-labeling, Applied Acoustics 166 (2020) 107375.

[28] K. K. Pal, K. Sudeep, Preprocessing for image classification by convolutional neural networks, in: 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE, 2016, pp. 1778–1781.