Bitlis Eren Üniversitesi Fen Bilimleri Dergisi
BİTLİS EREN UNIVERSITY JOURNAL OF SCIENCE
ISSN: 2147-3129/e-ISSN: 2147-3188
VOLUME: 12 NO: 1 PAGE: 226-241 YEAR: 2023
DOI:10.17798/bitlisfen.1225375

Fungus Classification Based on CNN Deep Learning Model

Serhat ORAL[1], İrfan ÖKTEN[2**], Uğur YÜZGEÇ[1]

[1]*Bilecik Şeyh Edebali University, Faculty of Engineering, Computer Engineering, Bilecik*
[2]*Bitlis Eren University, Tatvan Vocational School, Department of Computer Technologies, Bitlis*
(ORCID:0009-0005-2761-1295) (ORCID:0000-0001-9898-7859) (ORCID:0000-0002-5364-6265)

Keywords: Deep Learning, Convolutional Neural Networks, Flutter, Mushroom Classification, Image Classification.

Abstract
Artificial intelligence has been developing day by day and has started to take a more prominent place in human life. As computer technologies advance, research on artificial intelligence has also increased in this direction. One of the main goals of this research is to examine how real problems in human life can be solved using artificial intelligence-based deep learning, and to present a case study. Poisoning from the consumption of poisonous fungi is a common problem worldwide. To prevent these poisonings, a mobile application has been developed using Convolutional Neural Networks (CNNs) and transfer learning to detect the species of fungus. The application informs the user about the type of fungus, whether it is poisonous or non-toxic, and whether it is safe to eat. The aim of this study is to reduce poisoning events caused by incorrect fungus detection and to facilitate the identification of fungus species. The developed deep learning model is integrated into a mobile application developed by Flutter that is a mobile application development framework, which enable the detection of fungus species from images taken from the camera or selected from the gallery. CNNs and the EfficientNetV2 model, a transfer learning method, were used. By using these two methods together, the classification accuracy rate for 77 fungus species was obtained as 97%.

1. Introduction

Artificial intelligence is evolving day by day and occupying a larger and larger place in human life. As computer technologies develop and become more powerful, the number of researches and studies on artificial intelligence also increases in the same direction. Today, artificial intelligence, which continues to develop by dividing into many sub-branches, is widely used to solve problems in human life. It has gained a place in many fields, from routine tasks in daily life to tasks that are difficult for humans to learn and require time. One of the things that are taken into consideration while developing artificial intelligence technologies is the compatibility and usability with the devices used by people in daily life. The purpose here is computer, phone, tablet, wearable technology, etc., which are used very often in daily life to be actively introduced into human life and to benefit from more artificial intelligence. For this reason, studies on artificial intelligence for mobile devices outside of computers have increased significantly recently.

Fungi have been used for medicine and food by humans for centuries. While some of the hundreds of fungus species found in nature are poisonous, others are non-poisonous. However, some poisonous fungus species and non-poisonous fungus species can be very similar in appearance. Therefore, it is important to have a good knowledge of fungi and to have studied and used a variety of fungi in order to distinguish them. People who are not familiar with fungi may experience poisoning events when they try to eat fungi. To prevent poisoning events caused by fungi and to prevent the collection of the wrong fungus species, the idea of developing a mobile application that can detect the types of fungi has emerged. A Convolutional Neural Network (CNN) model has been utilized to solve the problem of distinguishing a large number of similar fungus

* Corresponding author: iokten@beu.edu.tr
Received: 28.01.2022, Accepted: 03.03.2023

species. This model aims to solve a difficult problem that would take people a long time to learn, using the power of artificial intelligence. The developed model uses a picture of the fungus to inform the user about the type of fungus, whether it is poisonous, and whether it is edible.

The application environment for the deep learning model developed in this study is a mobile device, which is very suitable for the problem being addressed. Since mobile devices are easily portable and are devices that most people are already familiar with using, the likelihood of experiencing difficulties while using the developed application is very low. Additionally, the highly developed camera systems in mobile phones make them ideal tools for solving image classification problems. For these reasons, the goal of this project is to provide a practical solution to the mentioned problem by combining deep learning and the power of mobile devices.

Much work has been done on the use of Convolutional Neural Networks (CNNs) and analysis of image data within the scope of Deep Learning. LeCun et al. discussed the subject of deep learning in general in their study [1]. By touching on the usage areas in the modern world, they explained the working logic of the foundation of deep learning and also explained the CNNs used for supervised learning and classification. There are several layers found in CNNs. These are: Convolutional Layer, Pooling Layer and Fully Connected Layer [2].

Transfer learning is a machine learning technique in which a model that has been trained on one task is re-purposed and fine-tuned for a different but related task. It involves using knowledge gained while solving one problem to solve a different, but related, problem. This can be done by transferring the weights and biases learned from one model to another model, or by using the pre-trained layers of a model as the starting point for a new model. There are different CNN models used for image classification and proven success according to the period in which they were developed. Some of these models are AlexNet, GoogLeNet, VGG-Net and ResNet [3]. Developing a deep learning model from scratch in a classification problem may not always be a logical approach. We can achieve high success and save time by training proven models with our own datasets. For all these reasons, the method called Transfer Learning was used in this study.

The proposed method in this study is original in the sense that it utilizes deep learning techniques, specifically Convolutional Neural Networks and transfer learning, to develop a mobile application that can accurately detect and classify different species of fungi from images taken from a camera or selected from a gallery. This approach is unique in its focus on solving a real-world problem, specifically the prevention of poisoning from the consumption of poisonous fungi, through the use of cutting-edge artificial intelligence techniques. The use of transfer learning with the EfficientNetV2 model is also noteworthy, as this method has been shown to improve the accuracy of deep learning models while requiring less training data, making it a promising approach for future research in this field. Overall, the proposed method represents a novel and innovative approach to solving a significant public health issue using advanced artificial intelligence techniques.

2. Material and Method

There are several methods for recognizing fungi, including traditional methods such as microscopy and taxonomic keys, as well as newer methods such as DNA sequencing and machine learning. Deep learning architectures such as convolutional neural networks (CNNs) can be trained to classify fungi based on their physical characteristics such as shape, size, and color. These algorithms can be used to build tools such as mobile applications or websites that can help users identify different fungi based on images or other data. Some studies on fungus classification in literature and deep learning are summarized in Table 1.

Table 1. Some studies on deep learning and on fungus classification.

Authors	Title of Study	Datasets	Experimental results	Year
Picek, L. et al.	Fungi Recognition: A Practical use Case	FGVCx Fungi Classification Kaggle	Accuracy rate : %79	2022 [4]
Picek, L. et al.	Automatic Fungi Recognition: Deep Learning Meets Mycology	Danish Fungi 2020 (DF20)	Accuracy rate : %79	2022 [5]

Krizhevsky, A. et al.	ImageNet Classification with Deep Convolutional Neural Networks	ImageNet LSVRC-2010 contest	top-5 test error rate of 15.3%	2012) [6]
He, K. et al.	Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification	ImageNet LSVRC-2012 contest	top-5 test error rate of 4.94%	2015 [7]
Kamnitsas, K. et al.	Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation	BRATS 2015 and ISLES 2015	Accuracy rate : %77	2017 [8]
Kayalı, N.Z. and İlhan Omurca, S.	Classification of Chinese Number Patterns with Convolutional Neural Networks (CNN)	MNIST dataset	Accuracy rate : %97	2021 [9]
Bozkurt, F. and Yağanoğlu, M.	Detection of COVID-19 from Lung X-Ray Images Using Deep Convolutional Neural Networks	Kaggle's COVID-19 radiography database	Accuracy rate : %97.17	2021 [10]
Ökten, İ. and Yüzgeç U.	Detection of Rice Crop Disease with Convolutional Neural Network	Kaggle's Rice Leaf İmages database	Accuracy rate : %97.57	2022 [11]

3. Material and Method

In this section, the deep learning type and methods to be used are mentioned. The usage areas, working logic and architecture of convolutional neural networks are explained. The environment in which deep learning studies will be carried out is introduced and the benefits it provides to the developers are explained. At the same time, important libraries to be used are also introduced.

3.1. Convolutional Neural Network (CNN)

One of the most popular models for classification from image data is the CNN model. The features of an image are extracted by passing the input image through multiple convolutional layers. The convolution layer is the basic building block of a CNN and is where most of the computation occurs. Typically, 3x3 filters are used to extract the features of the image. These filters scan all image pixels and perform convolution, as shown in Figure 1, resulting in a feature map [12]. After each convolution operation, the ReLU activation function is applied to the feature map.

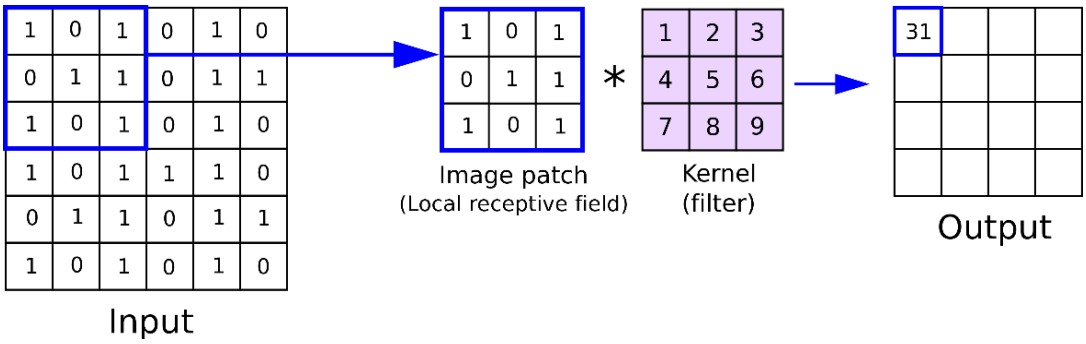


Figure 1. Illustration of the convolution operation [13].

Another layer in a CNN is the pooling layer. Pooling layers are used to further reduce the number of parameters and computational complexity of the model by gradually reducing the dimensionality of the representation. The largest pixel within the selected pooling size is transferred to the output. In the example shown in Figure 2, a 2x2 max-pooling process was applied by shifting it by 2 steps (pixels) [14]. This scales the feature map up to 25% while keeping the depth volume at its standard size [3].

A CNN model structure is shown as an example in Figure 3. Here, the input image undergoes a series of convolution and pooling operations, resulting in a final array through the fully connected

layer. The highest value in the sequence determines the class to which the image belongs.

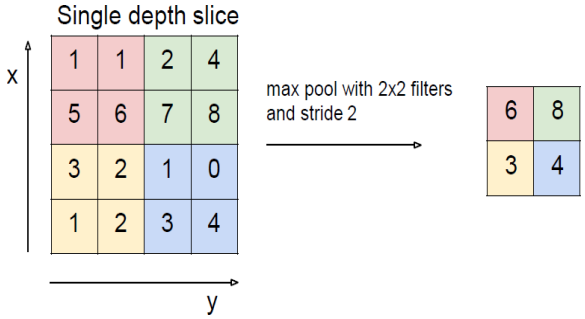


Figure 2. Max-pool operation [14].

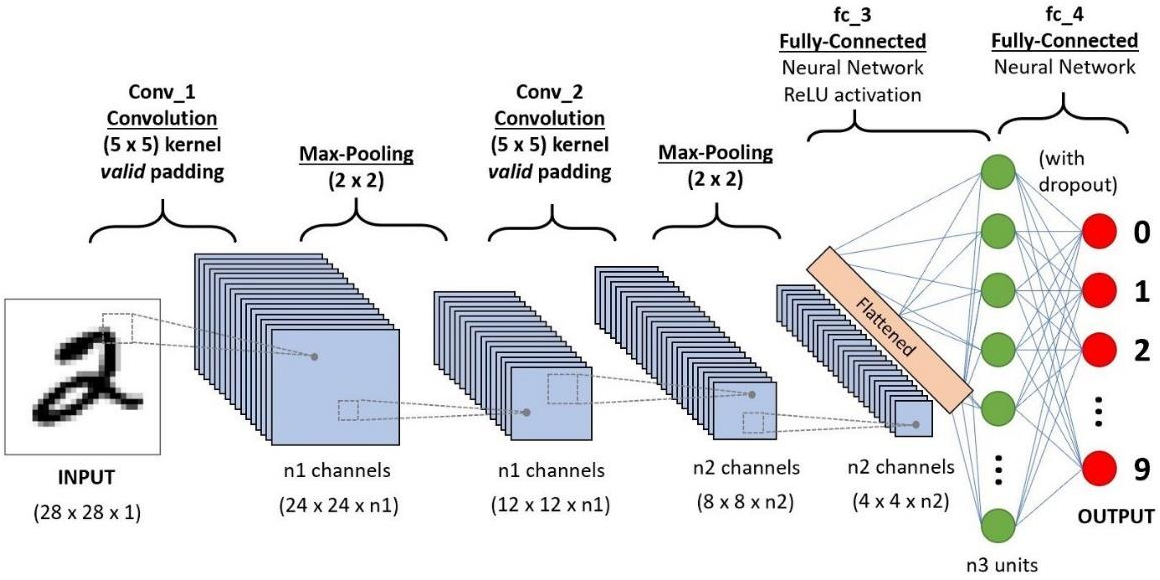


Figure 3. A sample CNN model structure [15].

3.2. Transfer Learning

Learning transfer is a machine learning technique in which a model that has been trained on one task is applied to a related, but different task. This can be useful when we have a smaller data set for the new task and don't have enough data to train a new model from scratch. It can also be useful when we need to train a model quickly, or when we want to use a pre-trained model as a starting point and then fine-tune it for a specific task [16]. There are two main approaches to learning transfer: feature extraction and fine-tuning.

Feature extraction involves using the pre-trained model as a fixed feature extractor, where the output of the pre-trained model's layers is taken as input for a new model. This new model is then trained on the new task using these extracted features. Fine-tuning involves unfreezing some of the layers of the pre-trained model and retraining them using the new data. This can be useful when the new task is similar enough to the original task that we can leverage the knowledge learned by the pre-trained model.

Image recognition has made significant advances in recent years due to the availability of large-scale datasets such as ImageNet, which contains over 1.2 million categorized natural images from over 1000 classes. Convolutional neural networks (CNNs) trained on these datasets have achieved successful results on various object detection and image segmentation tasks [17].

3.3. Feature Extraction

In multiclassification problems, the output layer of each model is specific to the dataset it is trained on. In other words, if the model is trained with a data set containing 50 classes, there must be 50 neurons in the output layer of the model. CNN models trained with ImageNet therefore have output layers with 1000 neurons. When we apply these pre-trained models to our own dataset and problem, we pull without the last

layers and add the appropriate output layer ourselves. The parts to be changed and preserved in the original model are basically shown in Figure 4.

In the feature extraction method, the weights are preserved by freezing the layers of the pre-trained model. Since training will not take place in these layers, the weights will not be updated. Here, the classification process is carried out by making use of the information learned from the previous training. Training takes place only in the last added layers.

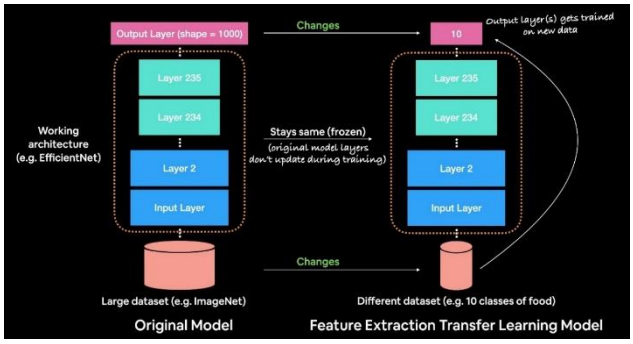


Figure 4. Application of feature extraction method to the model [18].

3.4. Fine-Tuning

Fine-tuning is a method of learning transfer in which all or part of the pre-trained model is unfrozen and retrained on the new data using a low learning rate. This is in contrast to feature extraction, where the pre-trained model is used as a fixed feature extractor and a new model is trained on the extracted features. Fine-tuning can lead to significant improvements by adapting the previously learned features to the new data. Figure 5 illustrates the difference between these two methods. It is often effective to use both feature extraction and fine-tuning together in learning transfer.

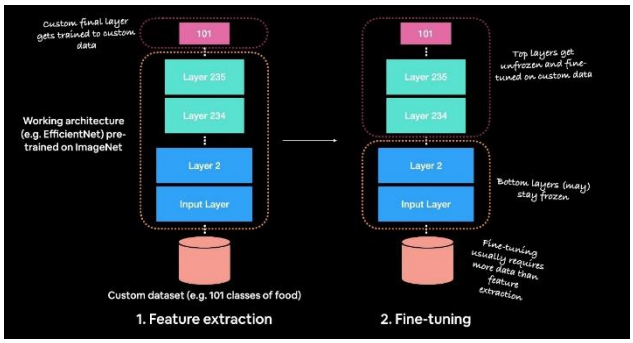
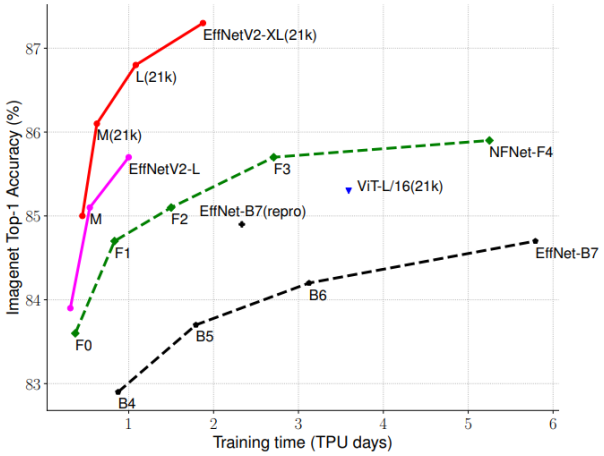


Figure 5. Feature extraction and fine-tuning [18].

3.5. EfficientNetV2B1 Model

The efficiency of deep learning in training is often related to the size of the model and the amount of training data. For instance, GPT-3, a very large model with a lot of training data, can achieve significant success in a few steps, but this requires weeks of training on thousands of GPUs, which can be complex and time-consuming. In contrast, techniques like learning transfer can be more efficient because they allow for the use of smaller models with fewer parameters, as long as the model is well-designed and effective at achieving successful results. EfficientNetV2 is a family of small size image classification models that provide good parameter efficiency and faster training as seen in Figure 6. It includes developed models in various sizes.



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

Figure 6. Training and parameter efficiency of EfficientNetV2 model [19].

The EfficientNetV2 is an improved version of the EfficientNet model, which has gained popularity due to its ability to balance accuracy and efficiency. The EfficientNetV2 model achieves state-of-the-art performance on image classification tasks, while requiring fewer parameters and achieving faster inference times compared to other popular deep learning models. The model is highly scalable and adaptable to different image sizes and resolutions, making it suitable for a wide range of applications. Its transfer learning capabilities and robustness to noise also make it an attractive option for developers working with limited training data or lower-quality

images. Overall, the EfficientNetV2 model is a valuable tool for computer vision applications and is likely to play an important role in the continued development of AI technologies. Compared to EfficientNet and newer runs, the EfficientNetV2 is up to 6.8x smaller while training up to 11x faster [19].

EfficientNetV2B1 is one of the smaller models in the EfficientNet family, with a moderate number of parameters and computational cost. It was designed for image classification tasks and has shown good performance on a variety of benchmarks. EfficientNetV2 models are an improvement over the original EfficientNet models, with better performance and fewer parameters. EfficientNetV2B1 is trained on the ImageNet dataset, which contains over 1 million labeled images from 1000 classes. It is a widely used benchmark for image classification tasks, and many CNN models are trained and evaluated on it. Therefore, EfficientNetV2B1 model was used in the fungus classification application to be developed.

3.6. Software Libraries

Within the scope of this study, the Python programming language was used to create the CNN model. Python is used in many areas and contains various libraries for different areas. It also supports developers and researchers with very powerful and useful libraries in the field of machine learning. In the fungus classification application, TensorFlow and Keras libraries supported by Google were used. Also, the Matplotlib and NumPy were used for data visualization and numerical computing.

3.7. Flutter

Flutter is an open source user interface development kit created by Google. It is used to develop applications for Android, iOS, Windows, Mac, Linux and the web [20]. It has a layered architecture that gives developers control over every pixel on the screen, using a set of built-in widgets.

Flutter is powered by Skia, a hardware-accelerated graphics library that is used in Chrome and Android, and it is designed to support fast, error-free graphics on both iOS and Android devices. The Flutter framework is built on the Dart platform, which provides support for 32-bit and 64-bit ARM machine code for iOS and Android, as well as JavaScript for the web and compiling to Intel x64 for desktop devices. Flutter is a good choice for developing deep learning models because of its fast performance and ability to create visually appealing apps.

4. Training and Test of the CNN Model

4.1. Data Set

In 2018, the "FGVCx Fungi Classification" competition was held on Kaggle [21] using data provided and sponsored by the Danish Swampe Atlas [22]. The Danish Fungus Atlas contains 8560 expert-approved fungus species and has been supported by over 4000 volunteers, resulting in over 1 million labeled data points. The FGVCx Fungi Dataset, created for the competition, includes 85578 images for training and 4182 images for verification of 1394 fungal species. There is also a set of test data consisting of 9758 images whose labels are not publicly available. For the purpose of this study, 77 fungus species from the competition's dataset were selected, resulting in a training dataset of 6733 fungus images, a validation dataset of 231 images, and a test dataset of 400 images.

The data set used to train the model should have a folder structure as shown in Figure 7. The FGVCx Fungi dataset does not come divided into training and validation sets, but rather is loaded as a whole. Information about which images should be used for training and which should be used for validation is provided in JSON format. A Python script was created to divide the data into training and validation sets based on this information.

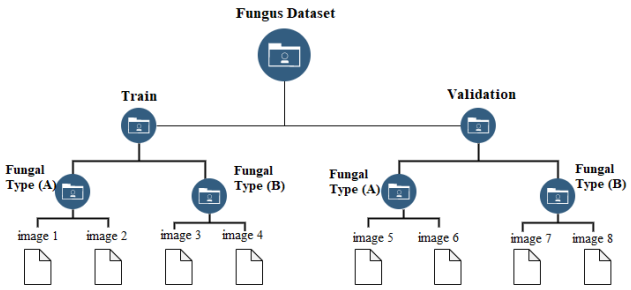


Figure 7. The desired folder structure of the data set.

4.2. Data Augmentation

The amount of data is crucial in deep learning. If a model has too many parameters relative to the number of training examples and has a complex structure, it may suffer from overfitting [23]. This means that while the model performs well on the training data, it may not perform as well on tests or in real-life classification. In other words, the model has memorized the few training examples it has seen and has difficulty generalizing to new examples. Data augmentation techniques, such as rotation, flip, and adding random noise, can help to improve the image

recognition accuracy [23,24]. These techniques involve modifying the training data in order to create additional, slightly different examples that can be used to train the model.

In this study, a data augmentation layer was implemented to augment the data during training. This layer was integrated into the model using the Keras functional API and will automatically apply data augmentation during the training phase. When using this model, it will not be necessary to manually perform data augmentation. The Keras library can be used to create a data augmentation layer, which can modify images during training according to the properties specified in the layer. Figure 8 shows some examples of the augmented images produced by the data augmentation layer.

Figure 8. Some images regenerated by the data augmentation layer.

4.3. Creating the Model

In this section, the pre-trained model was retrieved from the Keras functional API, modified with necessary additions, and trained. The model building process involved two stages: feature extraction and fine-tuning. At the end of each stage, the model's accuracy and loss were plotted in curve graphs.

4.3.1. Feature Extraction

EfficientNetV2B1, a model trained on the ImageNet1k dataset, will be used in this study. To utilize Transfer Learning, it is important to freeze the model by disabling the trainability of its layers. This helps to preserve the previously learned weights and avoids making any changes to them. In this case, the model was trained for 25 epochs.

To build the main model, the previously created data augmentation layer and input layer must be added. The Keras functional API offers a lot of flexibility and convenience for modifying the model by adding different layers as if adding a chain to different parts of the model. First, the input layer is included. Next, the data augmentation layer is added. Then, the EfficientNetV2B1 model, which forms the main part of the model, is added. A Global Average Pooling layer is then created and added to the end of the modified model. This layer is used to make the feature maps easier to interpret, rather than directly providing the feature maps to the fully connected layer. Finally, an output layer with 77 neurons is created to return 77 possibilities for an image, representing the 77 fungus species in the dataset. The "softmax" activation function is used in the output layer for multi-class classification, producing outputs between [0,1] indicating the probability that each input belongs to a class. The layers of the model are shown in Figure 9.

The image data used in the model was resized to 224x224x3 dimensions. The choice of optimization algorithm should take into account the characteristics of the problem and the dataset. In this case, the Adam optimization algorithm was used in the training process. After preparing the layers and parameters, the training process for the proposed model was initiated.

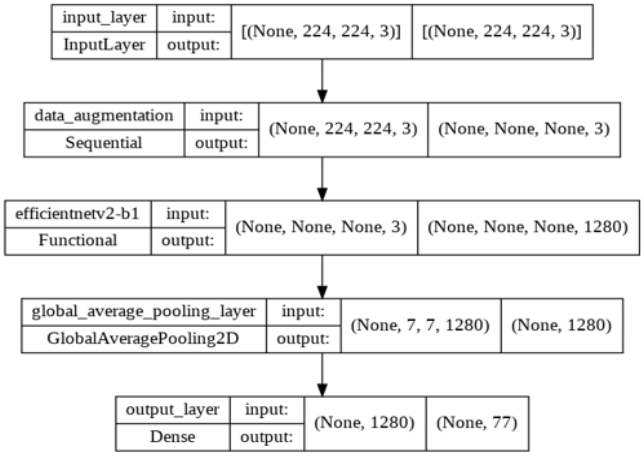


Figure 9. The structure of the proposed model.

Table 2 shows the first 5 and last 5 epoch values obtained in the initial training of the model. During training, the model was also validated using validation data at each epoch. Figure 10 plots the loss and accuracy values for feature extraction, which is the first training step. Both the table and the plotted curves can be used to make inferences about the model's performance.

Table 2. The results obtained in the first training of the model.

Epoch	Duration	Training loss	Traning accuracy	Loss of validation	Accuracy of validation
1/25	149s	2.8566	0.3575	2.0017	0.5281
2/25	96s	1.6368	0.6318	1.4246	0.6623
3/25	82s	1.2636	0.7079	1.1679	0.7446
4/25	79s	1.0594	0.7493	0.9908	0.7835
5/25	71s	0.9274	0.7760	0.8876	0.8139
...	...	...	...	...	...
21/25	45s	0.3353	0.9247	0.4228	0.8918
22/25	46s	0.3244	0.9266	0.4134	0.8961
23/25	43s	0.3150	0.9265	0.4084	0.9048
24/25	43s	0.2958	0.9335	0.3950	0.9004
25/25	44s	0.2794	0.9393	0.3814	0.9091

The model performed well in the first epoch, with a validation success rate of 0.5281 after one epoch of training. While this is a good value, it is expected that the loss would be quite high at the beginning of training. As training progressed, the model became more accurate and made fewer mistakes, as seen in the curves in Figure 10.

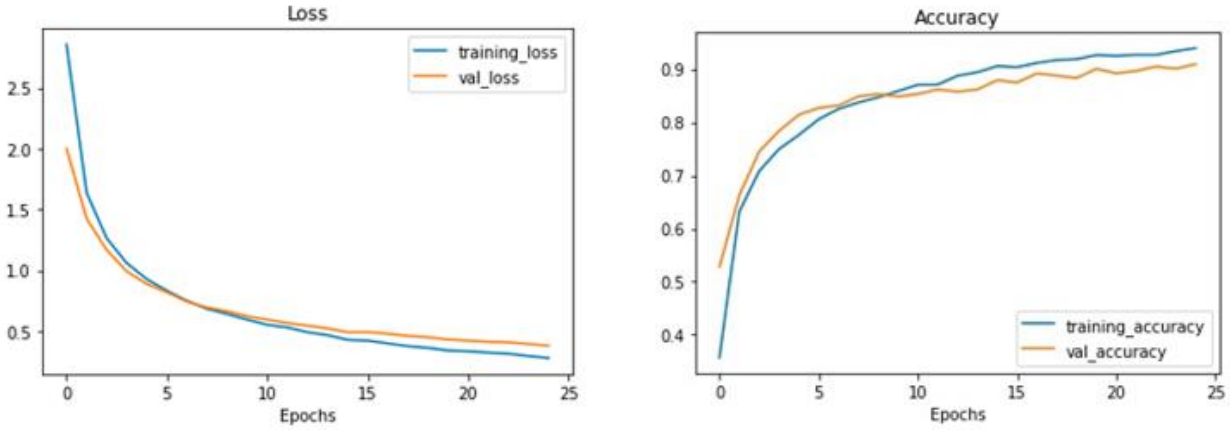


Figure 10. Loss and accuracy curves for 25 epochs.

The parts at the end of the curves remain stable and do not diverge from each other, indicating that the model is making consistent predictions in the validation tests compared to the predictions made during training. This suggests that there is no overfitting. If the last parts of the curves were widely separated and continued to diverge, it would indicate that the model is memorizing the data and further precautions would be necessary. Figure 10(b) shows some fluctuation at the last parts of the accuracy curves. Additionally, examining the last three epochs in the table reveals that saturation has begun to be reached for feature extraction, with the accuracy value increasing very slowly. The accuracy values for training and validation also appear to be starting to diverge towards the end, indicating that the training should be stopped at this point.

If the number of epochs is increased further, the model may start to overfit. It is important to pay attention to this and analyze the results of the training carefully. To reduce fluctuation in the curves and improve the model's performance, fine-tuning was applied to the model

4.3.2. Fine-Tuning

In this stage, the model was trained for 10 epochs by enabling the trainability feature for the last 15 layers of the model. This fine-tuning and updating is expected to improve the model's performance. To begin, all layers of the model are made trainable. Then, a loop is created and all layers except the last 15 are frozen again. This leaves only the last 15 layers as trainable. The training speed of the model is also reduced 100 times compared to normal. During the fine-tuning phase, training is performed at a much lower rate than usual. Finally, training continues from the last epoch value that the model reached in its previous training. The results of the model, which received a total of 35 epochs of training, are summarized in Table 3, and the curve graphs are shown in Figures 11 and 12.

Figure 11. Accuracy curves of the model after fine tuning.

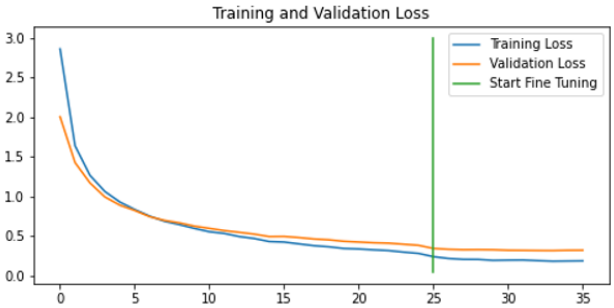


Figure 12. Loss curves of the model after fine tuning.

Table 3. Results obtained in the fine-tuning phase.

Epoch	Duration	Training loss	Training accuracy	Loss of validation	Accuracy of validation
25/35	62s	0.2384	0.9445	0.3420	0.9091
26/35	45s	0.2157	0.9459	0.3313	0.9134
27/35	45s	0.2046	0.9479	0.3263	0.9134
28/35	45s	0.2037	0.9459	0.3276	0.9177
29/35	44s	0.1920	0.9476	0.3257	0.9134
30/35	45s	0.1951	0.9465	0.3189	0.9264
31/35	45s	0.1963	0.9464	0.3174	0.9264
32/35	44s	0.1880	0.9491	0.3158	0.9264
33/35	43s	0.1812	0.9520	0.3150	0.9264
34/35	43s	0.1835	0.9528	0.3186	0.9221
35/35	42s	0.1851	0.9498	0.3194	0.9264

4.4. Testing the Model

When the developed model is evaluated using test data consisting of 400 images, it is found that the accuracy is 97% and the loss value is 0.1441. In addition to these values, various metrics are used to evaluate the performance of classification models. By calculating these values, various inferences can be made and a confusion matrix can be created. To evaluate the model performance, some metrics were utilized in this study. These are Accuracy, Loss-Categorical Cross-Entropy, Precision, Recall, and F1 Score. Accuracy is the most common metric used to evaluate classification models. It represents the percentage of correctly classified instances out of the total number of instances in the dataset. The Loss-Categorical Cross-Entropy is a loss function that measures the performance of a multi-class classification model by calculating the difference between the predicted probability distribution and the actual probability distribution of the target variable. Precision is a metric that measures the proportion of true positive predictions out of all the positive predictions made by the model. Recall is a metric that measures the proportion of true positive

predictions out of all the actual positive instances in the dataset. F1 score is the harmonic mean of precision and recall, and is a popular metric for evaluating classification models. It provides a balanced measure of precision and recall, and is particularly useful when the dataset is imbalanced. The formulas for calculating these metrics are summarized in Table 4, and the results of these metrics calculated for test of the proposed model are given in Table 5. PREC, REC, TN, TP, FN, and FP used in model evaluation metrics formulas are Precision, Recall, True Negative, True Positive, False Negative, and False Positive, respectively.

The success rate of the model was recorded and then the network was tested on real images. To do this, fungus images from different sources were uploaded to Google Colaboratory and provided as input to the trained network, and the classification results were obtained. The classification rates and images from the test results are shown in Figure 13. To demonstrate the performance of the model, the confusion matrix obtained using the samples of 77 fungal species is shown in Figure 14.

Table 4. Metrics and formulas used in model evaluation.

Metrics	Formula
Accuracy	$\frac{TP+TN}{TP+FP+Fn+TN}$
Loss-Categorical Cross-Entropy	$\sum_{i=0}^{outputsize} y_i * log\hat{y}_i$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 Score	$\frac{2*PREC*REC}{PREC+REC}$

Table 5. Evaluation results of the model.

Accuracy	0.9700
Loss	0.1441
Precision	0.9746
Recall	0.9696
F1 Score	0.9694

Figure 13. Classification results of test fungus images.

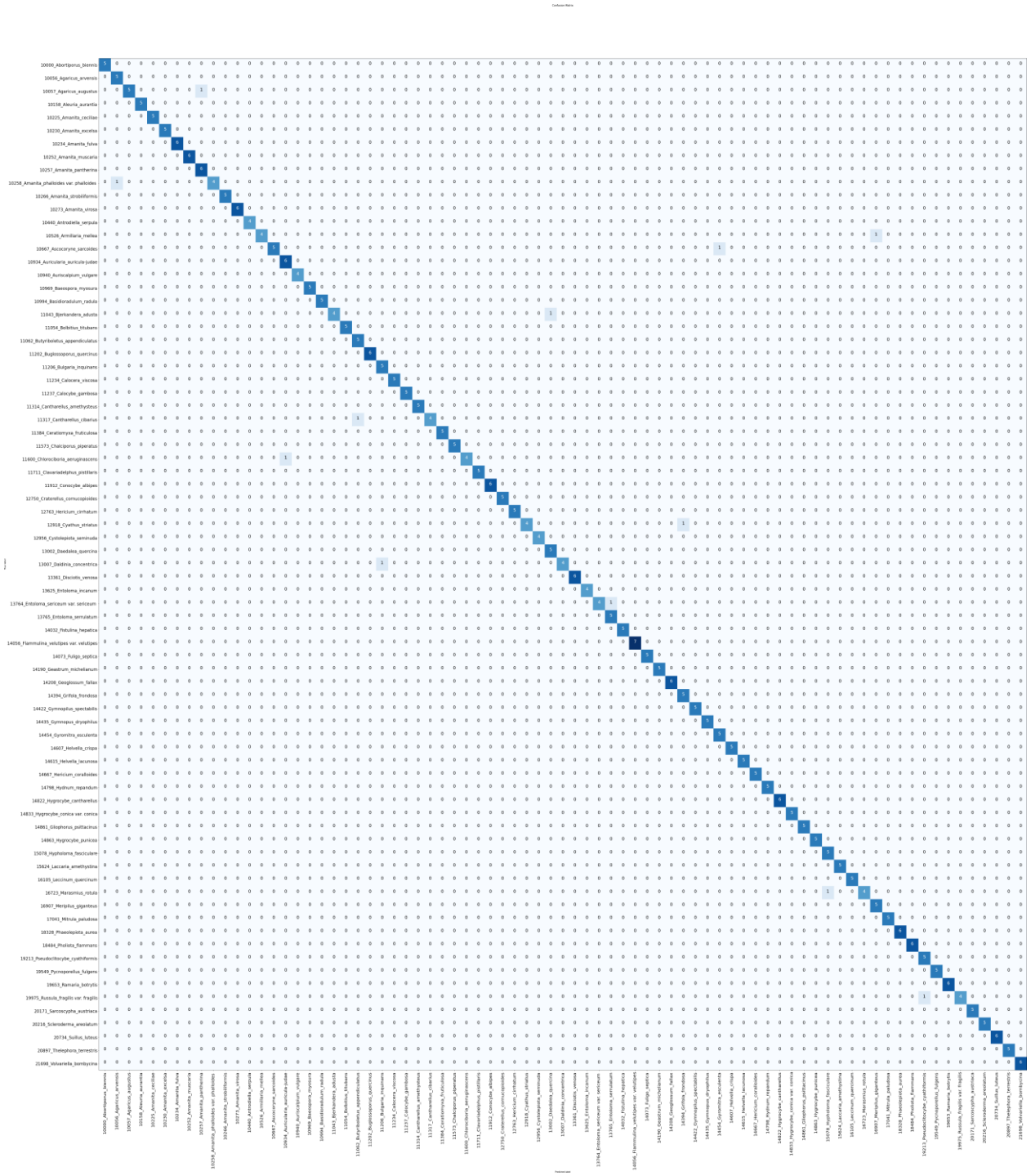


Figure 14. Confusion matrix of 77 fungal species used in the model.

5. Mobile Application Developed for Fungus Species Detection

After training the model for detecting fungus species, the trained model was saved. Since the model was developed in Colaboratory, the trained network is also saved here. However, the saved model cannot be used directly on mobile devices. Instead, the model was converted to the tflite format in the Colaboratory environment for use on mobile devices. A mobile application was then developed using the Dart programming language and the Flutter UI kit.

5.1. Application Homepage

When the application is first opened, the homepage shown in Figure 15 appears to the user.

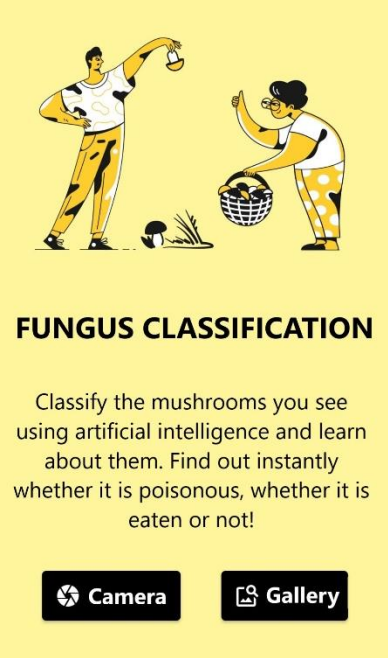


Figure 15. Application homepage.

The main purpose of this page is to allow the user to input fungus images for classification and display the results. The user has two options for importing fungus images: taking a photo with the camera or choosing an image from the gallery. Two buttons are provided on the homepage to perform these actions.

5.2. Results Page

In the result page shown in Figure 16, images from the main page are input to the model, which classifies the fungus species and returns the name of the species in the image. A dictionary structure was also created that contains information about the fungus species, such as whether they are poisonous, edible, and found in certain regions. By using the fungus name returned by the model, the corresponding information in the dictionary structure is retrieved and displayed to the user. Some symbols were also created and used to convey information about the condition of the fungus. These symbols and their meanings are shown in Figure 17. Separate dictionaries have been created for these symbols, and the information symbol for the relevant fungus is printed by checking these dictionaries.

Figure 16. Some screenshots of the fungus classification results.

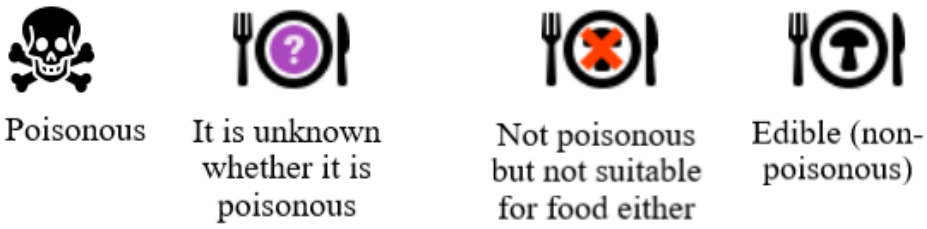


Figure 17. Warning symbols about fungus classification.

6. Conclusion and Future Works

In this study, we discuss the use of deep learning to solve an image classification problem. The main idea is to demonstrate that problems can be quickly and easily solved by following appropriate methods and steps with transfer learning. To this end, we chose the fungus species classification problem and developed a study to solve it, including discussions of convolutional neural networks and transfer learning.

Since the problem involves image classification, it is important that fungus images can be easily obtained. Therefore, the study was designed to be implemented in a mobile application. The model for classification was first prepared on Google Colaboratory, then converted to the "TensorFlow Lite" model and used in the mobile application.

The prepared model accurately classifies 77 fungus classes with high rates of success. When tested with 400 fungus images, the model achieved an accuracy of 0.97 and an average F1 score of 0.9694. One of the main challenges in fungus classification is that different fungus species can be very similar to each other, making it difficult for the human eye to distinguish them and requiring extensive knowledge and expertise. Deep learning models may also utilized to accurately classify fungus species that are very similar to each other. To minimize this, it is important to have a diverse and rich dataset for training. This should be considered if a larger number of fungus species classifications are to be performed, as the number of species increases, the similarity in appearance between the fungus species will increase and the problem will become more complex and difficult. In addition, paying attention to overfitting and allowing for longer training periods will also contribute to better learning success.

The mobile application developed to utilize the machine learning model has demonstrated successful performance. The application, which has a simple design, is optimized for fast and accurate classification of fungus species. Proper planning is crucial for the application's success, as issues such as data migration and page loading can significantly impact its performance. The application includes a dictionary of 77 fungus species, which allows users to access and view information about specific fungi by using the species name returned by the model through classification.

When using deep learning to solve a problem, it is important to first clearly define the problem and thoroughly analyze it. The complexity of the problem should be taken into account when selecting appropriate data for the model. In the field of deep learning, data plays a crucial role. If the data is not clean, accurate, and relevant to the problem at hand, it is likely that the model will not perform well. One of the major factors driving the rapid growth of this field is the availability of large, high-quality data sets. It is essential to carefully consider these two key steps in the process of using deep learning.

In the future, the work presented in this study could be expanded and refined to further improve the accuracy and efficiency of the model. One potential direction for future work is to increase the number of fungus species that the model can classify. This would require a larger and more diverse dataset for training, as well as more sophisticated techniques for handling the increased complexity of the classification problem. Another potential area for improvement is to optimize the mobile application's performance and

user experience, such as by implementing more advanced features or incorporating user feedback. Finally, the techniques and methods presented in this study could be applied to other image classification problems, opening up new opportunities for using deep learning to solve real-world problems.

Contributions of the authors

This study is a study prepared by Serhat ORAL and İrfan ÖKTEN under the consultancy of Prof Dr Uğur YÜZGEÇ. Serhat ORAL contributed to the finding of the data, the algorithm of the model and the preparation of the software, İrfan ÖKTEN contributed to the development of the software and the writing of the article in this study, and Uğur YÜZGEÇ contributed to the editing and consultancy of the article.

Conflict of Interest Statement

There is no conflict of interest between the authors.

Statement of Research and Publication Ethics

The study is complied with research and publication ethics

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," *arXiv [cs.NE]*, 2015.

[3] F. Sultana, A. Sufian, and P. Dutta, "Advancements in image classification using convolutional Neural Network," *arXiv [cs.CV]*, 2019.

[4] L. Picek, M. Šulc, J. Matas, J. Heilmann-Clausen, T. S. Jeppesen, and E. Lind, "Automatic fungi recognition: Deep learning meets mycology," *Sensors (Basel)*, vol. 22, no. 2, p. 633, 2022.

[5] S. Sladojevic *et al.*, *Fungi Recognition: A Practical Use Case*. 2020.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[8] K. Kamnitsas *et al.*, "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation," *Med. Image Anal.*, vol. 36, pp. 61–78, 2017.

[9] N. Z. Kayalı and S. Ve Ilhan Omurca, *Konvolüsyonel Sinir Ağları (CNN) ile Çin Sayı Örüntülerinin Sınıflandırması*. 2021.

[10] F. Bozkurt ve M. Yağanoğlu, "Derin Evrişimli Sinir Ağları Kullanarak Akciğer X-Ray Görüntülerinden COVID-19 Tespiti", *Veri Bilimi*, c. 4, sayı. 2, ss. 1-8, Ağu. 2021.

[11] İ. Ökten ve U. Yüzgeç, "Evrişimli Sinir Ağı ile Çeltik Bitkisi Hastalığının Tespiti", *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, c. 11, sayı. 1, ss. 203-217, Mar. 2022, doi:10.17798/bitlisfen.1014393.

[12] "What are Convolutional Neural Networks?" *Ibm.com*. [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks. [Accessed: 27-Dec-2022].

[13] A. H. Reynolds, "Anh H. reynolds," *Anh H. Reynolds*. [Online]. Available: https://anhreynolds.com/blogs/cnn.html. [Accessed: 27-Dec-2022].

[14] A. Kızrak, "DERİNE DAHA DERİNE: Evrişimli Sinir Ağları - ayyüce kızrak, ph.D," *Medium*, 28-May-2018. [Online]. Available: https://ayyucekizrak.medium.com/deri%CC%87ne-daha-deri%CC%87ne-evri%C5%9Fimli-sinir-a%C4%9Flar%C4%B1-2813a2c8b2a9. [Accessed: 27-Dec-2022].

[15] S. Saha, "A comprehensive guide to convolutional neural networks — the ELI5 way," *Towards Data Science*, 15-Dec-2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 27-Dec-2022].

[16] M. Hussain, J. J. Bird, and D. R. Faria, "A study on CNN transfer learning for image classification," in *Advances in Intelligent Systems and Computing*, Cham: Springer International Publishing, 2019, pp. 191–202.

[17] H.-C. Shin *et al.*, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[18] S. Lee, "(TF2) Transfer Learning - Feature Extraction," *AAA (All About AI)*, 05-Mar-2022. [Online]. Available: https://seunghan96.github.io/dlf/TF2_4%EC%9E%A5/. [Accessed: 27-Dec-2022].

[19] M. Tan and Q. V. Le, "EfficientNetV2: Smaller models and faster training," *arXiv [cs.CV]*, 2021.

[20] Wikipedia contributors, "Flutter," *Wikipedia, The Free Encyclopedia*. [Online]. Available: https://tr.wikipedia.org/w/index.php?title=Flutter&oldid=27787028.

[21] "2018 FGCVx fungi classification challenge," *Kaggle.com*. [Online]. Available: https://www.kaggle.com/competitions/fungi-challenge-fgvc-2018/overview. [Accessed: 27-Dec-2022].

[22] "Danmarks officielle database for svampefund," *Danmarks SvampeatlasXXXXX*. [Online]. Available: https://svampe.databasen.org/en/. [Accessed: 27-Dec-2022].

[23] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," *Proc. Conf. AAAI Artif. Intell.*, vol. 34, no. 07, pp. 13001–13008, 2020.

[24] W. Li, C. Chen, M. Zhang, H. Li, and Q. Du, "Data augmentation for hyperspectral image classification with deep CNN," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 4, pp. 593–597, 2019.