International Journal of Intelligent Information Systems
2024, Vol. 13, No. 4, pp. 59-77
https://doi.org/10.11648/j.ijiis.20241304.11

Research Article

Optimizing Food101 Classification with Transfer Learning: A Fine-Tuning Approach Using EfficientNetB0

Adebayo Rotimi Philip*

Department of Computer Science, University of Lago, Lagos, Nigeria

Abstract

Much research has been done on the classification of the food101 dataset, but much of this research which achieved an accuracy score of more than 90% explores heavyweight architecture such as EfficientNetB7, Visual Geometry Group19, ResNet-200, Inception v4, DenseNet-201, ResNeXt-101, MobileNet v3 and many more. This study explores the classification of the Food101 dataset using the EfficientNetB0 architecture, a lightweight architecture. Compared to other popular CNN architecture, EfficientNetB0 has relatively small parameters, which makes it computationally efficient and suitable for deployment on resource-constraint environments. The research aims to balance model accuracy and computational efficiency, addressing the need for resource-constrained environments. Five experiments were conducted while varying the number of fine-tuned layers. Results demonstrate that the fine-tuned EfficientNetB0 model achieves an accuracy score of accuracy score of 97.54%, Top_k_categorical accuracy of 99.89%, precision of 98.21%, and recall of 97.02% in just 5 epochs. This research will significantly contribute to the field of transfer learning by developing specialized models that excel in target tasks. Besides, it will advance dietary monitoring, food logging, and health-related technologies, enabling more accessible and practical solutions for consumers. However, the optimal number of layers to fine-tune for achieving perfect accuracy with EfficientNetB0 remains uncertain. It often involves trial and error to determine the best configuration for optimal results, presenting an opportunity for future research.

Keywords

Transfer Learning, EfficientNets, Lightweight Architecture, Convolutional Neural Network, Fine-Tuning

1. Introduction

1.1. Background

Machine learning has experienced advancement in recent years, with the emergence of Artificial Neural Network (ANN) [1], a machine learning model inspired by the human brain, consisting of interconnected nodes (neurons) that process and transmit information to solve complex tasks. ANN networks have been used to solve problems in medicine, industrials, and even in services [2]. One of the most impressive forms of ANN architecture is that of the Convolutional Neural Network (CNN). The CNN also called computer vision, is a type of neural network designed for images and video analysis. It makes use of convolutional and pooling layers to extract features from images, together with dense layers (fully connected layers) for classification and regression tasks [3]. CNN is known and widely used for image recognition, object de-

*Corresponding author: rotphilipadeb@yahoo.com (Adebayo Rotimi Philip), adebayophilip72@gmail.com (Adebayo Rotimi Philip)

Received: 1 July 2024; **Accepted:** 24 July 2024; **Published:** 15 August 2024

tection, and image segmentation in AI.

One intriguing application of CNN is in the domain of food recognition, which this research aims to explore. The Food 101 dataset is a well-known benchmark for this task, containing 101 categories of food items with 1,000 images each, resulting in a total of 101,000 images [4]. Each image in the dataset is labeled, providing a rich resource for training and evaluating image classification models [4]. The Food 101 dataset provides multifaceted challenges. Its substantial size demands significant computational resources for processing and training models. The high intra-class variability and inter-class similarity add to the complexity, making it difficult to accurately distinguish between similar food items. Besides, the dataset includes images of varying quality, background, light conditions, and noise levels, which can affect model performance.

Several papers have been published using the food101 dataset, but the accuracy scores obtained have not been phenomenal. For instance, Lukas, Matthieu, & Luc (2018) published a paper in 2018 titled "Food-101 – Mining Discriminative Components with Random Forests" where they obtained an accuracy score of 50.76% [5]. More so, Ren, Xinying, & Khai (2021) worked on the Food 101 datasets classification problem using several models: for LNAS-NET model, they obtained an accuracy score of 49.3% after 100 epochs, MobileNET 49.1% accuracy score after 100 epochs, mobileNETv2, 17.2% after 100 epochs, ShuffleNETv2 44.1% after 100 epochs [6]. Those studies that achieved high accuracy rates above 90% relied on heavyweight architectures, which are unsuitable for resource-constrained environments. For instance, Mingxing (2019) obtained an accuracy score of 91.7% on CIFAR-100 dataset and 98.8% on flower datasets using EfficientNetB7 [7]. Also, Rudraja (2022) obtained an accuracy of 93.7% using ResNet-152, a heavyweight architecture with over 60 million parameters [8].

This research aims to propose transfer learning models such as EfficientNetB0, which is partially fine-tuned (layers are made trainable) to unravel the Food 101 problem with a better accuracy score. Also, the research aims to balance model accuracy and computational efficiency, addressing the need for resource-constrained environments. This work will not only demonstrate the power and flexibility of transfer learning in tackling complex image classification problems but will also provide valuable insights into the practical applications of deep learning in food recognition. This endeavor will bridge the gap between advanced machine learning techniques and real-world applications, highlighting the transformative potential of AI in everyday life.

1.2. Aims and Objective of the Research

This research aims to achieve the following

1) To achieve an optimal solution of food101 classification with fine-tuned EfficientNetB0.
2) To show that lightweight convolutional neural network can achieve outstanding results that is only possible with heavyweight convolutional neural work
3) To balance model accuracy and computational efficiency, addressing the need for resource-constrained environments.

2. Literature Review

Image classification is a key problem in computer vision, with many recent improvements coming from object recognition [6]. Traditional methods use local or dense grid descriptors, pooled into vectors, and then classified with SVMs. Recent methods emphasize nonlinear feature encodings, like Fisher Vectors and spatial pooling [9]. Jorge et al. (2013) used the Fisher Kernel framework to describe patches by how they differ from a general Gaussian mixture model, resulting in what we call Fisher Vectors (FV) [10]. They use this model to describe an image for classification by extracting local patches, encoding them into a high-dimensional vector, and combining them into an overall image signature. Furthermore, Lazebnik, Schmid & Ponce (2006) introduces a method for recognizing scene categories using approximate global geometric correspondence [11]. It works by dividing an image into progressively smaller sub-regions and calculating histograms of local features within each sub-region. Findings show that the method surpasses current state-of-the-art results on the Caltech-101 database and achieves high accuracy on a large database of fifteen natural scene categories.

Taichi & Keiji (2009) introduced an automatic food image recognition system that helps record daily meals. Using Multiple Kernel Learning (MKL), the system integrates image features like color, texture, and SIFT [12]. A prototype was tested on food images from phone cameras, achieving a 61.34% classification rate for 50 food types. This is the first practical food image classification system. Mei-Yun Chen, et al. (2009) addressed the issues of feature descriptors in the food identification problem and introduced a preliminary approach for quantity estimation using depth information [13]. It combines SIFT, Local Binary Pattern, Gabor, and color features, training a multi-label SVM classifier for each. Using 50 food categories with 100 images each, it achieves 68.3% accuracy and over 80% accuracy in top-N candidates, making mobile applications practical.

More so, Chen et al. (2009), presented the first visual dataset of fast foods, including 4,545 still images, 606 stereo pairs, 303 360° videos for structure from motion, and 27 privacy-preserving videos of volunteers eating [14]. The dataset, aimed at dietary assessment research, features 101 foods from 11 fast food chains, with images captured in restaurants and a lab. They benchmark it using color histograms and SIFT features with a classifier. Lukas, Matthieu, & Luc (2018) researched the classification of the Food 101 datasets using Random Forests to identify key parts for all classes [5]. For efficiency, the researchers focus on patches aligned with image super-pixels called components. They tested the method on a dataset of 101 food categories with 101,000 images, achieving an average ac-

curacy of 50.76%. The model outperforms other methods, except CNNs, and surpasses SVM classification on Improved Fisher Vectors and other part-mining algorithms.

Hassannejad et al (2016) examined the effectiveness of deep convolutional neural networks (DCNNs) in identifying foods from photographs. This study utilized various DCNN-related techniques, including activation features extracted from pre-trained DCNNs and pre-training on large-scale ImageNet data [15]. The model achieved a classification success rate, with accuracy rates of 78.77 percent and 67.57 percent for the UECFOOD100/256 dataset. Kuang-Huei et al (2017) researched on the classification of food-101N introduced in a CVPR 2018 paper CleanNet [16]. The food-101N has 310,009 images of food recipes and 101 food classes. They used transfer learning (ResNet-50) to address label noise and keep verification labels for part of the classes to only learn from human supervision. The result shows a top-1 accuracy score of 81.44% compared to 81.67% for the Food-101 dataset and reduced label noise detection error rate on held-out classes where no human supervision is available by 41.5% compared to current weakly supervised methods.

Mingxing & Vuoc (2019) studied the model scaling of the EfficientNet architecture and identified factors such as network depth, width, and resolution that can lead to better performance [17]. The EfficientNetB7 architecture achieves 84.3% top-1 accuracy on ImageNet, 91.7% accuracy score on CIFAR-100, and 98.8% on flower datasets. Pranjal & Seba (2023) researched the performances of five popular very deep pre-trained networks namely, Inception-v3 with 48 layers, EfficientNet-B0 with 237 layers, Xception with 71 layers, DenseNet-121 with 121 layers, and MobileNet with 53 layers, for the classification of food images from the benchmark Food-101 [18]. findings show that Xception gives the best performance for classifying the 101 categories of food images, with an accuracy of 84.54%, significantly outperforming the other deep pre-trained networks.

Rudraja, V. (2022) worked on the classification of food-101 datasets using several transfer learning models [19]. The researcher leverage on MobileNetV2, InceptionV3, Efficient Net, Resnet152, and Resnet50. They obtained an accuracy score of 92.50% for MobileNetV2, 93.89% for InceptionV3, 93.25% for EfficientNetB2, 93.79% for Resnet152, and 92.46% for Resnet50 respectively. VijayaKumari, Priyanka, and Vishwanath (2022) employed transfer learning techniques to categorize various food products into their appropriate categories [20]. Using Efficientnetb0, a transfer learning technique, the developed model classified 101 distinct food kinds with an accuracy of 80%. When compared with other food classification models, the EfficientNetB0 outperformed other models with the best accuracy.

EfficientNetB0 architecture has been integrated with other architectures to improve performance. Jenan & Raidah (2023) used the EfficientNetB0 with Principal Component Analysis (PCA) and Random Forest (RF) to distinguish between the fingerprints of male and female gender [21]. The SOCOFing fingerprint dataset was fed into PCA to decrease the dimension of the feature images and RF classifier for fingerprint classification. They obtained an accuracy of 99.91% [21]. Furthermore, Wijdan et al. (2021) proposed a hybrid model using the FER2013 dataset for facial expression which integrates two CNN models, one of which is EfficientNetB0. The hybrid model obtained an accuracy score of 74.39%, outperforming other state-of-the-art classification methods [22]. The research done by Neha et al. (2021) presents a hybrid encoder–a decoder-based model for segmenting healthy organs in the GI tract in biomedical images of cancer patients [23]. EfficientNetB0 is used as a bottom-up encoder architecture for downsampling to capture contextual information by extracting meaningful and discriminative features from input images. The encoder EfficientNetB0 model achieves Dice coefficient and Jaccard index values of 0.8975 and 0.8832, respectively which outperform existing ecoder systems: ResNet 50, MobileNet V2, and Timm Gernet [23].

The fine-tuning lightweight model has been proven to improve accuracy. To justify this claim Paolo et al. (2024) performed 3000 training processes focusing on 32 small to medium-sized target datasets. They show that the top-tuning approach provides comparable accuracy concerning fine-tuning, and the results suggest that top-tuning is an effective alternative to fine-tuning in small/medium datasets, especially useful when training time efficiency and computational resource saving are crucial [24]. Furthermore, Manoj & Brajesh (2023) fine-tune six pre-trained CNN models: Efficient-NetB0, EfficientNetB7, ResNet50, VGG19, DenseNet121, and DenseNet201 are fine-tuned for hyperspectral image classification. The results show that fine-tuning improves performance and saves computational resources. Among the models, EfficientNetB0 performs better than others with 90.79% accuracy for the Houston image [25]. Francis & Alon's (2021) work supports previous studies that fine-tuning improves accuracy. They evaluated the efficiency EfficientNetB0 model to diagnose malaria parasite infections in blood smears. The fine-tuned model obtained the highest accuracy of 94.70% after 50 epochs [26].

From the literature review, researchers achieved high accuracy rates in classifying the Food101 dataset using the more complex EfficientNetB7 architecture, which attained a 93.7% accuracy rate. However, EfficientNetB7 is resource-intensive, requiring significant computational power to process even a few epochs. Other CNN models, such as MobileNetV2, InceptionV3, EfficientNetB2, ResNet152, and ResNet50, achieved up to 93.89% accuracy on the training dataset after 100 epochs. This research aims to surpass the highest accuracy model with a less resource-intensive architecture in fewer than 10 epochs.

Summary of the literature review (table)

Table 1. *Shows the summary of the literature review.*

Author(s)	Problem definition	Model used	Results
Jorge et al. [27]	Used the Fisher Vectors Framework to describe image classification	Fisher Vectors Framework	Successfully classify images into their respective classes
Lazebnik, Schmid & Ponce [11]	Introduced a method for recognizing scene categories using approximate global geometric correspondence.	Nill	the method surpasses current state-of-the-art results on the Caltech-101 database
Taichi & Keiji [28]	Introduced an automatic food image recognition system that helps record daily meals.	Multiple Kernel Learning (MKL)	Achieved a 61.34% classification rate for 50 food types test data
Mei-Yun Chen, et al. [29]	Addressed the issues of feature descriptors in the food identification problem	Multi-label SVM classifier	Achieved 68.3% accuracy and 80% accuracy in top-N candidates on 50 food classes and 100 images in each class
Chen et al. [13]	Aimed at dietary assessment research. They presented the first food101 datasets with limited images	Color histograms and SIFT features with a classifier	Nill
Lukas, Matthieu, & Luc [30]	Researched the classification of the Food 101 datasets	Random Forests	Achieved an average accuracy of 50.76%. The model outperforms SVM classification and a Fisher Vectors algorithm
Hassannejad et al [15]	Examined the effectiveness of deep convolutional neural networks (DCNNs) in identifying foods from photographs.	DCNN-related techniques	Achieved accuracy of 78.77 percent
Kuang-Huei et al [31]	researched on the classification of food-101N, a food dataset with 310,009 images	Used transfer learning (ResNet-50) to address label noise	Achieved top-1 accuracy score of 81.44% compared to 81.67% for the Food-101 dataset
Mingxing & Vuoc [7]	Studied the model scaling of the EfficientNet architecture and identified factors that can lead to better performance	EfficientNetB7 Network	Achieved 84.3% top-1 accuracy on ImageNet, 91.7% accuracy score on CIFAR-100, and 98.8% on flower datasets
Pranjal & Seba [29]	Researched the performances of five popular pre-trained networks for the classification of food images from the benchmark Food-101.	Inception-v3 with 48 layers, EfficientNet-B0 with 237 layers, Xception with 71 layers, DenseNet-121 with 121 layers, and MobileNet with 53 layers	Findings show that Xception gives the best performance for classifying the 101 categories of food images, with an accuracy of 84.54%.
Rudraja, V. [18]	worked on the classification of food-101 datasets using several transfer learning models.	MobileNetV2, InceptionV3, Efficient Net, Resnet152, and Resnet50.	Accuracy of 92.50% for MobileNetV2, 93.89% for InceptionV3, 93.25% for EfficientNetB7, 93.79% for Resnet152, and 92.46% for Resnet50.
VijayaKumari, Priyanka, and Vishwanat [19]	Employed transfer learning techniques to categorize various food products into their appropriate categories.	Efficientnetb0	Achieved an accuracy of 80%.

Author(s)	Problem definition	Model used	Result
Jenan & Raidah [21]	Distinguished between the fingerprints of male and female gender using the SOCOFing fingerprint dataset	the EfficientNetB0 with Principal Component Analysis (PCA) and Random Forest (RF)	They obtained an accuracy of 99.91%

Author(s)	Problem definition	Model used	Result
Wijdan et al. [23]	proposed a hybrid model using the FER2013 dataset for facial expression which integrates EfficientNetB0 with another model		obtained an accuracy score of 74.39%,
Neha et al. [24]	presents a hybrid encoder–a decoder-based model for segmenting healthy organs in the GI tract in biomedical images of cancer patients	EfficientNetB0 is used for the encoder system	Dice coefficient of 0.8975 and Jaccard index values 0.8832
Paolo et al. [25]	performed 3000 training processes focusing on 32 small to medium-sized target dataset to show that the top-tuning approach provides comparable accuracy.	Lightweight model	results suggest that top-tuning is an effective alternative to fine-tuning in small/medium datasets,
Manoj & Brajesh [26]	hyperspectral image classification	Efficient-NetB0, Efficient-NetB7, ResNet50, VGG19, DenseNet121, and DenseNet201	The results show that fine-tuning improves performance and saves computational resources.
Francis & Alon's [32]	evaluated the efficiency EfficientNetB0 model to diagnose malaria parasite infections in blood smears	EfficientNetB0	The fine-tuned model obtained the highest accuracy of 94.70% after 50 epochs.

3. Transfer Learning

When building a machine learning model, one question to often ask is if similar projects have been done in the past. Instead of building a model from the start, we can benefit from pre-trained models and fine-tune them to the problems. This is called transfer learning. Transfer learning (TL) is a machine learning (ML) technique where a model pre-trained on one task is fine-tuned for a new, related task [33]. Based on the fundamental principle of transferability of experiences, TL emulates human capability to leverage previous knowledge in new tasks [20]. Building a new model can be time-consuming and can be an intensive process that requires a large amount of data, computing power, and several iterations before it is ready for production. For instance, a machine learning model that is trained to identify the images of dogs can be fine-tuned to identify the images of cats, using a smaller image size that highlights the feature differences between dogs and cats.

Transfer learning is highly beneficial in creating machine learning models. One of the benefits is that it enhances efficiency [33]. Building machine learning models requires a large volume of data, is time-consuming, and computationally expensive. However, transfer learning takes care of these deficiencies as it can work with a small amount of data. Transfer learning models often demonstrate greater robustness in diverse and challenging environments. They can better handle real-world variability and noise, having been exposed to a wide range of scenarios in their initial training, thus they give better results [19].

3.1. The Architecture of the Model

While we can build our own CNN network to achieve high accuracy on the Food101 dataset, we can leverage predefined models already built by others. The EfficientNet family is the ideal transfer learning model for this task, offering superior performance for image classification due to its optimized scaling of depth, width, and resolution. EfficientNet's compound scaling method can achieve state-of-the-art results with fewer parameters and lower computational costs. [34].

3.2. EfficientNetB0

EfficientNet is a type of Convolutional Neural Network (CNN) that improves accuracy by evenly increasing the network's depth, width, and resolution [8]. EfficientNetB0 is the baseline model in the EfficientNet family, from which other complex EfficientNet models (B1 – B7) are developed. Introduced by Google in 2019, EfficientNet can achieve high performance with fewer parameters and FLOPs (floating point operations) compared to other CNN models [9]. This makes EfficientNetB models ideal for deployment in resource-constrained environments, offering a balance of speed and accuracy for image classification tasks [9]. It is widely used in various applications, from mobile devices to large-scale cloud environments.

EfficientNet models perform more efficiently than most existing CNN models trained on the ImageNet dataset. Figure 1 shows the efficiency of EfficientNetB0 – B7. Just like other CNN models, increasing the complexity of EfficientNets lead to better accuracy, though they require or demand more computational resources.

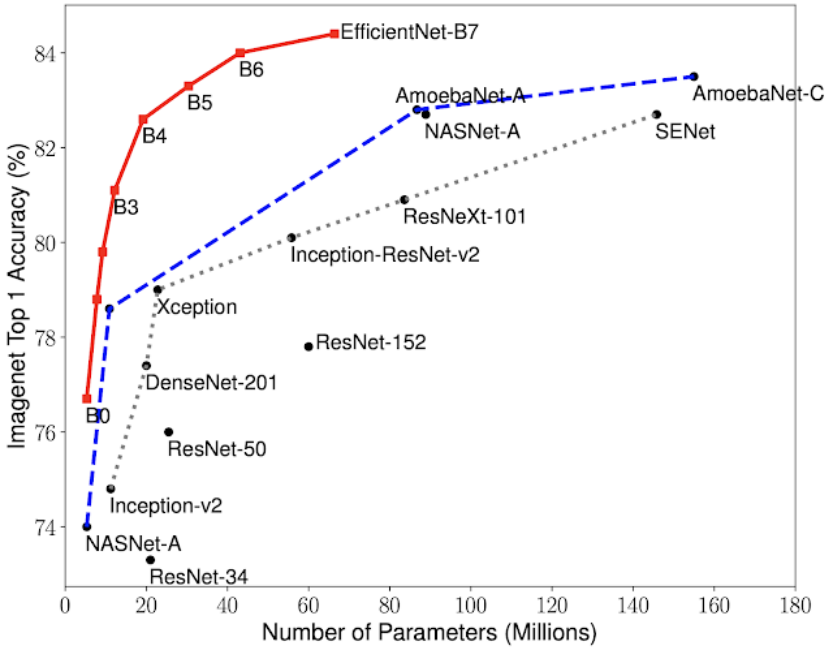

Figure 1. *Shows the performance of the EfficientNets and other CNN transfer learning networks [9].*

3.2.1. Characteristics of EfficientNets

EfficientNetB0 comprises 257 layers, whereas EfficientNetB7, the most advanced model in the EfficientNetB family, contains 813 layers [9]. The EfficientNetB architecture is generally divided into two major components: the stem layer and the subsequent layers. Each model within the EfficientNetB series, from B0 to B7, consists of 7 blocks, with each block containing multiple sub-block layers [35]. EfficientNetB0 features the fewest sub-block layers, and the number of these layers increases progressively from B0 to B7, scaling up in complexity and capability [36].

3.2.2. Architecture of EfficientNetB0

To understand the architecture of EfficientNetB0, we can run this below code in the Google Colab or Jupyter Notebook.

import tensorflow as tf

base_model = tf.keras.applications.efficientnet_v2.EfficientNetV2B0(include_top=False)

 # Check layers in our base model

for layer_number, layer in enumerate(base_model.layers):

print(layer_number, layer.name)

The outputs of the codes are displayed below. Figure 2 shows the blocks (layers) of the EfficientNetB0. There are 237 layers, but Figure 2 only shows the first 21 layers. Observing figures 2 and 3, we can see that there are two components in EfficientNetB0: the stem component and the block component. The stem component is made up of the first five layers: input layer, rescaling layer, normalization layer, Convolutional layer (Conv2D), Batch Normalization layer (stem_bn), and the activation layer (stem activation), while the block component has seven blocks (1, 2, 3, 4, 5, 6, and 7 blocks) and each block has several sub-layers.

```
0 input_1
1 rescaling
2 normalization
3 stem_conv
4 stem_bn
5 stem_activation
6 block1a_project_conv
7 block1a_project_bn
8 block1a_project_activation
9 block2a_expand_conv
10 block2a_expand_bn
11 block2a_expand_activation
12 block2a_project_conv
13 block2a_project_bn
14 block2b_expand_conv
15 block2b_expand_bn
16 block2b_expand_activation
17 block2b_project_conv
18 block2b_project_bn
19 block2b_drop
20 block2b_add
21 block3a_expand_conv
```

Figure 2. *Shows the layers of the EfficientNetB0 architecture.*

Running the code below gives us more detailed information about the architecture of EfficientNetB0

base_model.summary()

```
Model: "efficientnetv2-b0"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_1 (InputLayer)        [(None, None, None, 3)]   0         []

 rescaling (Rescaling)       (None, None, None, 3)     0         ['input_1[0][0]']

 normalization (Normalizati  (None, None, None, 3)     0         ['rescaling[0][0]']
 on)

 stem_conv (Conv2D)          (None, None, None, 32)    864       ['normalization[0][0]']

 stem_bn (BatchNormalizatio  (None, None, None, 32)    128       ['stem_conv[0][0]']
 n)

 stem_activation (Activatio  (None, None, None, 32)    0         ['stem_bn[0][0]']
 n)

 block1a_project_conv (Conv  (None, None, None, 16)    4608      ['stem_activation[0][0]']
 2D)

 block1a_project_bn (BatchN  (None, None, None, 16)    64        ['block1a_project_conv[0][0]']
 ormalization)

 block1a_project_activation  (None, None, None, 16)    0         ['block1a_project_bn[0][0]']
  (Activation)

 block2a_expand_conv (Conv2  (None, None, None, 64)    9216      ['block1a_project_activation[0
 D)                                                              ][0]']

 block2a_expand_bn (BatchNo  (None, None, None, 64)    256       ['block2a_expand_conv[0][0]']
 rmalization)

 block2a_expand_activation   (None, None, None, 64)    0         ['block2a_expand_bn[0][0]']
 (Activation)

 block2a_project_conv (Conv  (None, None, None, 32)    2048      ['block2a_expand_activation[0]
 2D)                                                             [0]']

 block2a_project_bn (BatchN  (None, None, None, 32)    128       ['block2a_project_conv[0][0]']
```

Figure 3. *Shows the layers of the EfficientNetB0.*

(i). Input Layer

The input layer is the first layer in the model architecture. It is the layer where the image data is prepared for subsequent layers. The input layer expects images of a specific size, typically 224x224 pixels for EfficientNetB0 [37]. Other bigger models (B1 – B7) take a higher image resolution. For instance, the image resolution for B0 model – B7 model are given in table 2.

Table 2. *Shows image resolution of EfficientNets.*

Models	Image Resolution
EfficientNetB0	224 x 224
EfficientNetB1	240 x 240
EfficientNetB2	260 x 260
EfficientNetB3	300 x 300
EfficientNetB4	380 x 380
EfficientNetB5	456 x 456
EfficientNetB6	528 x 528
EfficientNetB7	600 x 600

Datasets are stored in different shapes and forms, which could be in a vector or matrix or a table. images and videos with multiple colour channels are represented as (batch_size, height, width, channels), where batch_size is the number of images processed simultaneously, height and width are the dimensions of the image, and the channels represent red, green, and blue colours (RGB) from which all image colours are made. For instance, let's assume that the height of the image in figure 4 is 512 pixels and the width is 320 pixels, then the image is made up 163,840 pixels.

Figure 4. *Shows an image representation in pixels [39].*

Each pixel is a combination of three colors (Red, Green, and Blue) at varying intensities. The intensity value for these colors is represented from 0 to 255, indicating how much of the color is present in the pixel [38]. A pixel with a color intensity value of (235, 10, 10) exhibits a high red color intensity. Consequently, a white pixel is represented numerically as (255, 255, 255), indicating maximum intensity for red, green, and blue, while a black pixel is represented as (0, 0, 0), indicating the absence of color intensity in all three components [38]. Figure 5 shows the colour intensities of the three-color channels (Red, Green, and Blue).

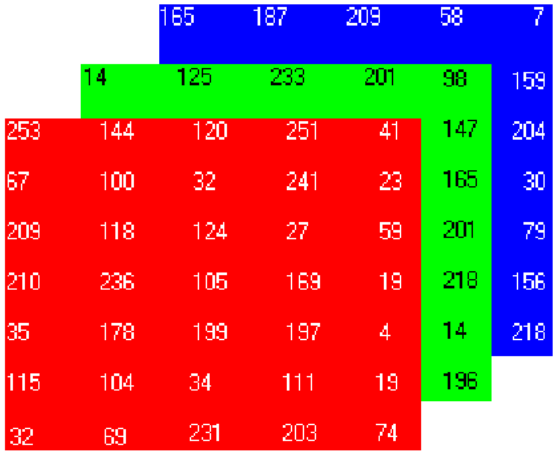

Figure 5. *Shows the channels (red, green, blue colors) in images [39].*

The shape of an input image is crucial in Convolutional Neural Networks (CNNs) as they are trained to process data in specific formats. CNNs expect data to be represented in appropriate shapes to function correctly [37]. A significant issue in image processing for CNNs is the inconsistency in image shapes across datasets [39]. For example, EfficientNetB0 requires images to be in the shape of 224x224x3 for processing. However, the images in the Food101 dataset (the dataset for this research) are typically 384x512x3, with some even differing from this representation [5]. Due to these discrepancies, it is necessary to resize the images to 224x224 for compatibility with the EfficientNetB0 model [40].

Another concern is the datatype of the images. Machine learning algorithms process numerical data, typically in integer (int) or floating-point (float) formats [40]. However, some images are stored in unit or object formats. These datatypes need to be converted to numerical formats before being fed into the algorithm.

(ii). Image Preprocessing

The second layer in the EfficientNet architecture is the image rescaling. Image preprocessing, which can also be termed image rescaling is processing or refining the image input to a format that can be processed by the CNN to enhance learning and improve performance [41]. Scaling input features through normalization and standardization, generating additional training samples via data augmentation, filtering noise through noise reduction, and performing feature engineering are essential steps that can significantly enhance a neural network's performance [39].

Image resizing is a critical preprocessing step in machine learning, especially in computer vision tasks. It involves adjusting the dimensions of an image to fit the input requirements of a model. Most convolutional neural networks (CNNs) expect images to have uniform dimensions, making resizing necessary to ensure consistency across the dataset. Resizing images serves multiple purposes. It reduces computational complexity, requires less memory and processing power, and helps handle varying image resolutions [41].

However, resizing images must be done carefully to maintain the aspect ratio and avoid distortion [38]. Distorting an image can lead to the loss of critical features and can adversely affect model performance. Techniques such as padding can be used to resize images without distortion. For example, if an image is resized to a square shape, padding can be added to maintain the original aspect ratio by adding borders to the image. Another consideration during resizing is the interpolation method used. Different interpolation methods, such as nearest-neighbor, bilinear, and bicubic, can impact the quality of the resized image.

(iii). Normalization Layer

Normalization technique involves adjusting the range of pixel values in an image to a standardized scale, typically between 0 and 1 or -1 and 1. The primary rationale behind normalization is to enhance the convergence speed and stability of training algorithms [41]. By scaling the pixel values, normalization ensures that the numerical input to the neural network is more uniform, which helps in mitigating issues related to gradient vanishing or explosion during the training phase [41]. This preprocessing step is especially critical when using activation functions like sigmoid or hyperbolic tangent (tanh), which are sensitive to the scale of input data.

The research done by Norhikmah, Afdhal & Rumini (2022) demonstrated that normalization accelerates the training process and improves the overall performance of the machine learning models [41]. This improvement is attributed to the reduction in numerical instability and the promotion of faster gradient descent convergence. Furthermore, normalization prevents the network from depending on the original range of input data, thus enhancing the model's generalization capabilities across different datasets [42]. Common normalization techniques include min-max normalization, which scales the pixel values to a specified range, and z-score normalization, which adjusts the values based on the mean and standard deviation of the dataset.

(iv). Batch Size

The batch size is the number of samples that will be processed by a network at one time. Assuming we have 960

images of food and the batch size is 32, it means that for one complete epoch, 30 batches each of 32 images will be processed by the network. Processing in batches enables the resources available to process the batch samples one after the other than processing the 960 images at once, which will require more resources and could slow down the process. If there are enough resources, the batch size can be larger as a larger batch size completes each epoch within a shorter time than a smaller batch size. The trade-off, however, is that even if our machine can handle very large batches, the quality of the model may degrade as we set our batch larger, thus causing overfitting [43].

Figure 6. *Shows batching processing techniques [44].*

(v). Convolutional Layer

Figure 7 below shows the processes involved in the convolutional layer. The layer has the input image represented in pixels, the filter (a square matrix also called a kernel), and the activation map, which indicates the most relevant regions in the input image for predictions, illustrating the learned features.

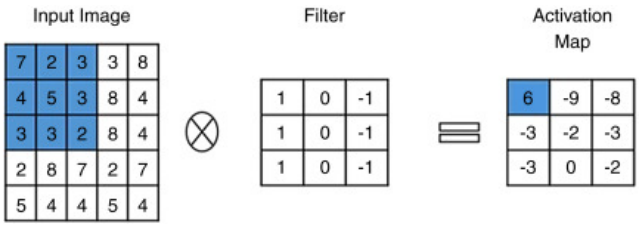

Figure 7. *Shows the convolutional layer processes [45].*

A convolutional layer is an important layer in a Convolutional Neural Network (CNN). it is made up of filters (kernels) whose values are learned during training [44]. These filters are usually smaller than the input image as shown in figure 7. Each filter slides over the image and performs a dot product between its values and the corresponding input values to create an activation map.

How does it Work?

In simple terms, think of a filter as a small window that scans the image and highlights patterns it recognizes. For example, in Figure 7, the filter is multiplied (dot product) with the blue matrix in the input image, and the result of the dot product is inputted in the activation map. For instance,

$$7*1 + 2*0 + 3*-1 + 4*1 + 5*0 + 3*-1 + 3*1 + 3*1 + 2*-1 = 7 + 0 - 3 + 4 + 0 - 3 + 3 + 3 - 2 = 6$$

This process is repeated as the filter slides over the entire image, producing a new matrix called the feature map (or activation map).

The final output of a convolutional layer is a stack of these activation maps, one for each filter. Each map can be thought of as the output of a neuron that is connected to a small region of the input image, the size of which matches the filter size. All neurons in an activation map share the same parameters, meaning they detect the same pattern but at different positions in the image.

Hyperparameters in Convolutional Network

Hyperparameters are settings configured in CNN that guide the way the convolutional process is done. In the convolutional layer, three hyperparameters (padding, kernel size, and stride) are of utmost significance.

Padding

During convolutional training, the kernel (filter matrix) can extend beyond the activation map, thus padding converses the data at the border of the activation map. This is necessary to preserve the spatial size (shape of the activation map) [45]. Many padding techniques are available for use, but the most commonly used is zero padding because of its performance, simplicity, and efficiency. Padding techniques involve adding zeros symmetrically around the edges of an input [45].

Kernel size

The kernel size refers to the dimension of the filter matrix. The job of the kernel size is to extract information from the image [39]. The kernel is multiplied with the image matrix to form the activation map. The type of kernel size used In a CNN network has a significant impact on the performance of the classification task. A small kernel size can extract more detailed information from an image than a large one. The 3x3 kernel in Figure 7 produces a 3x3 activation map, however, a 2x2 kernel size will produce a 4x4 activation map [32]. A larger kernel size leads to a faster reduction in the layer dimension, leading to worse performance.

Stride

The stride determines how many pixels to the right should the kernel shift over the image matrix to perform the second dot product operation. Tiny VGG, an image classification network, uses a stride of 1 for its convolutional layers, meaning that the kernel moves 1 pixel to the right to perform the next dot multiplication operation [43, 45]. After the completion to the right, the kernel moves 1 pixel downward to cover other pixels in the image. The impact of the stride is similar to that of the kernel size. A large stride (say 3 pixels) reduces the features much more quickly, thus reducing performance [45]. Conversely, a small stride (like 1 pixel) allows

the kernel to extract more features from the image, thus leading to better performance.

The convolutional layer of EfficientNetB0

Figure 8 shows the convolutional layer of the EfficientNetB0. The convolutional layer is the main process of a CNN network. From Figure 8, the first convolutional layer is a conv3x3 matrix (also called the conv2D in figure 3). The 3x3 represents the kernel size, while the '2D' represents a 2-dimensional matrix.

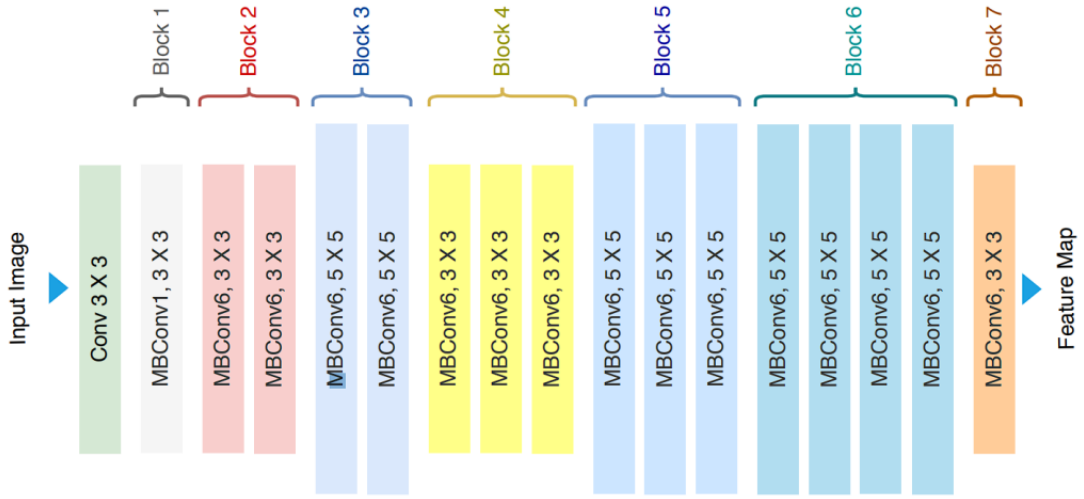

Figure 8. *Shows the convolutional layer of the EfficientNetB0 model [7, 47].*

The convolutional layers in blocks 1 to 7 are similar. In block 1, MBConv1, 3x3 refer to a specific type of convolutional block called Mobile Inverted Residual Bottleneck (MBConv) with a matrix (filter) of 3x3 and an expansion factor of 1. MBConv was introduced by MobileNetV2 architecture and it is designed for computation and memory efficiency [46]. It expands the number of channels (RGB colors) using pointwise convolution, applies the depthwise separable convolution, and then represents the results in a lower dimensional space [45]. The expansion factor indicates how much the channel is expanded before depth-wise convolution [47].

4. Metrics Used in the Experiment

Before discussing the results, we need to explain the metrics (accuracy, loss, Top_k_categorical_accuracy, precision, and recall) used in these experiments. The diagram below considers a confusion matrix.

Consider this confusion matrix above. The dark blue boxes represent true positive and false positive respectively. True positive means that the model predicted the exact as the actual. The term 'true' means that the model correctly predicted the sample and 'false' means otherwise. Positive and 'negatives' refer to the samples in the data. In a case where we have two samples (binary classification), the first sample could be classified as positive and the second sample as negative.

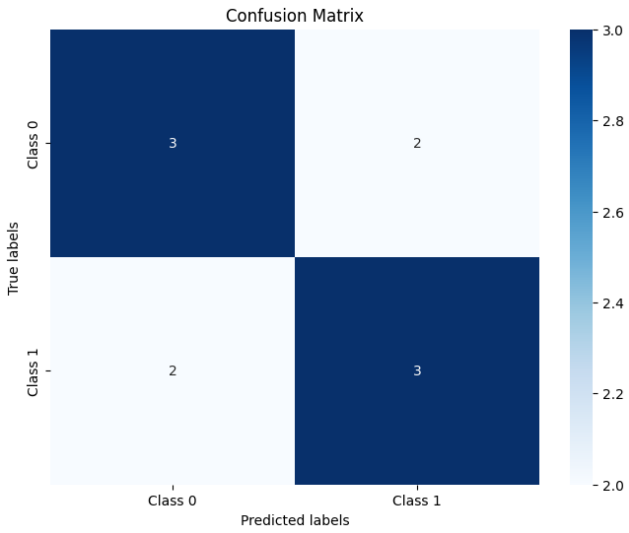

Figure 9. *Shows a confusion matrix.*

Accuracy

Accuracy is a metric that describes how a model performs across all dataset. It is useful when all classes of a dataset are of equal significance and it measures the overall correctness of a model. It is calculated by the ratio of number of correct predictions to the total number of predictions.

$$\text{Accuracy} = \frac{True_{positives} + True_{negatives}}{Total\ number\ of\ predictions}$$

Precision

Precision, also known as the positive predictive value, is the ratio of true positive predictions to the total predicted positives. It measures the accuracy of positive predictions. It focuses on the correctness of positive predictions.

$$\text{Precision} = \frac{True_{positives}}{True_{positives} + False_{positives}}$$

Recall

Recall, also known as sensitivity or true positive rate, is the ratio of true positive predictions to the total actual positives. It measures the ability of a model to identify all relevant instances.

$$\text{Recall} = \frac{True_{positives}}{True_{positives} + False_{negatives}}$$

Top-k Categorical Accuracy

Top-k categorical accuracy measures the proportion of predictions where the true label is among the top k predicted labels. This is particularly useful for multiclass classification where you want to know if the correct label is within the top k predictions.

$$\text{Top-k Categorical Accuracy} = \frac{Number\ of\ correct\ prediction\ in\ top\ k}{Total\ number\ of\ predictions}$$

5. Research Methodology

The research methodology involves data collection, understanding the data, image preprocessing, building the base model using 10% of the data, fine-tuning experiments, evaluating the model with validation and test data, and lastly choosing the best model.

5.1. Data Collection

The Food-101 dataset is a large collection of food images designed for the task of food classification. The Food-101 datasets were collected to help advance research in computer vision and machine learning, particularly in the domain of image recognition. Introduced by the ETH Zurich Computer Vision Lab in 2014, the Food-101 dataset has become the extremely popular benchmark for evaluating algorithms due to its diversity and complexity.

Composition of Food-101 Dataset

The Food-101 dataset contains 101,000 images divided into 101 categories, with each category representing a different type of food. Each category includes 1,000 images, offering substantial data for training and evaluation. The dataset features a variety of dishes, from common fast-food items like burgers and hot dogs to more complex dishes like paella and sushi.

1) Images: 101,000 images (approximately) of food dishes, divided into 101 classes (e.g., pizza, sushi, tacos, etc.).
2) Resolution: Images are resized to 512x384 pixels.
3) Format: JPEG format.
4) Classes: 101 classes, each representing a specific type of food dish.
5) Each class of image has 1000 images

Figure 10. *Shows part of the images of food101 dataset [15].*

The Food101 dataset can be downloaded from Google Storage or TensorFlow Datasets. During the download process, the data is shuffled to reduce overfitting, preventing the model from merely memorizing the training data and thereby enhancing its generalization ability. Shuffling also improves model robustness, enabling it to handle new and unseen data more effectively, which helps prevent bias, enhance convergence, and provide a better representation of the dataset. This ensures that the CNN model is trained on a diverse and representative set of images, ultimately leading to improved performance and generalization.

5.2. Understanding the Data

After obtaining the data, it is compulsory to explore the data before modelling. In the food101 datasets, we do the following to get acquainted with the dataset.

1) We explore the class labels. The data has 101 food classes. The first 10 food classes are

['apple_pie', 'baby_back_ribs', 'baklava', 'beef_carpaccio', 'beef_tartare', 'beet_salad', 'beignets', 'bibimbap', 'bread_pudding', 'breakfast_burrito']

2) We can randomly select images of food in the dataset to better understand the food classes.
3) We can check the tensor for each image. A tensor for instance is represented as ([255, 192, 203]) for each pixel. Each image in the food101 dataset has a resolution of 224x224, totaling 50,176 pixels. Therefore, an image in the food101 dataset has 50,176 tensors and each tensor has three values indicating the intensities of RGB colours. The intensity of each colour of red, green or blue has values from 0 – 255.
4) We can check the shape and data type of the images. The food101 input shape is [224x224x3] and the output shape is a vector. The datatype is represented in unit 8. We have to convert the datatype in int32 before processing.
5) We can check the labels of the dataset. Are they one-hot encoded or label encoded?
6) We can check if the labels match-up with the class names

5.3. Image Preprocessing

We cannot use our raw data food101 this way. We need to preprocess the data. Preprocessing in image classification involves preparing raw image data for the classification algorithm [36]. This includes steps like resizing images (see 3.2.2.2) to a uniform size, normalizing pixel values (see 3.2.2.3), batching (see 3.2.2.4), augmenting data through transformations (e.g., rotations, flips), and removing noise. Preprocessing enhances the quality of the input data, ensures consistency, and often improves the performance and accuracy of the classification model [39].

For our food101 data, we convert the uint8 data type to float 32 datatypes. Besides, the food101 dataset comprises of different sized tensors (images), so the data need to be converted into 255 by 255, first for consistency and besides, the EfficientNetB0 can only process images in 255 by 255. Lastly, the images need to be scaled. Scaling also call normalization is the process of converting the pixel values into numbers between 0s and 1s. This is done by dividing the pixels by 255. EfficientNetB1 for instance, takes images in size 260 by 260. In the case, the image will be divided by 260 for scaling.

We need to process our images in batches. Batch processing is the technique of dividing a large dataset into smaller, manageable batches for processing. This method enhances memory efficiency, accelerates computation, and optimizes resource utilization [48], particularly in machine learning and data processing tasks, enabling continuous and effective use of hardware resources like CPUs and GPUs. In our experiment, we batch the 101,000 images into batches of 32 images and label pairs, thus enabling the images to fit into the GPU memory.

5.4. Building the Base Model

To build the base model, we typical use TensorFlow, importing the pre-trained EfficientNetB0 model from its respective library. The pretrained model already trained on a large dataset like ImageNet, is fine-tuned to enable the model to generalize better

5.5. Fine-tuning Experiments

Fine-tuning is the process of unfreezing the EfficientNetB0 layers to optimize the model's performance. Fine-tuning allows the weight of the base model to adapt to the dataset. enabling the model to leverage learned features while adapting to new data. The result is a powerful and efficient base model suitable for various image classification applications. We did the first experiment with 10% of the data and fine-tune the top 5 layers for experiment 2. For experiment 3, we train the EfficientNetB0 model on the entire dataset, after which we fine-tuned the top 50 layers and top 100 layers for experiment 4 and 5 respectively.

5.6. Evaluation of the Models

We evaluate the performance of the experiments using the accuracy score, loss function, recall, precision, and Top-k Categorical Accuracy. See 4.0

6. Analysis of the Result

6.1. Experimental Results

To demonstrate the effectiveness of transfer learning with EfficientNetB0, the research was conducted using various quantities of data. We began by experimenting with 10% of the training data and 101 food classes, equating to 7575 images with each class containing 75 images. Next, we performed three experiments on the whole dataset (101,000 images), fine-tuning the layers to obtain the optimal accuracy.

Experiment 1: Feature extraction with 10% training dataset

In the feature extraction model, all layers are fixed except for the input layer [49]. After five epochs, the model (10% of the training data) achieved an accuracy of 0.6440 and a loss of 1.4493. The training process took 4 minutes and 32 seconds. The validation accuracy and loss were 0.6572 and 1.7824 respectively. When evaluated on the test dataset, the model achieved an accuracy of 0.5797. The figure 11 illustrates the training and validation metrics over five epochs.

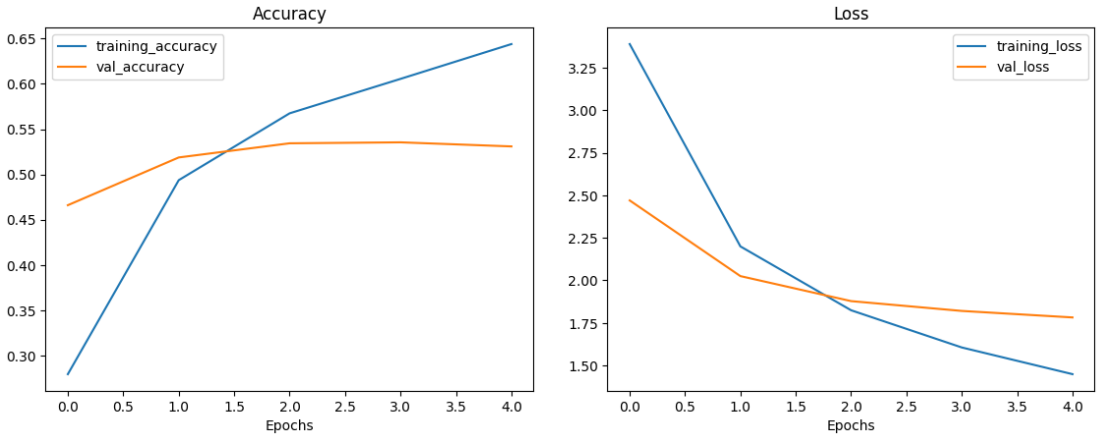

Figure 11. *Shows the accuracy and the loss function of experiment 1.*

Experiment 1 shows moderate performance with relatively low accuracy and high loss. Precision and recall are balanced, but there's room for improvement. Figure 11 shows that the validation dataset is overfitting. If the number of epochs is increased, the accuracy will increase. The validation metrics are consistent with the training metrics, but precision is abnormally high, likely due to calculation issues. The recall is lower, indicating missed positive instances.

Experiment 2: Fine-tuning model 1

To improve the results in experiment 1, we fine-tune the EfficientNetB0 architecture. Fine-tuning a model involves taking a pre-trained EfficientNetB0 model and adjusting its weights to better suit a specific dataset [37]. Initially, the model, pre-trained on a large dataset like ImageNet, has learned general features. Fine-tuning adapts these features to the new dataset by adjusting the weights of the pre-trained model to fit to the specific task [50]. This process involves unfreezing the layers that capture more task-specific features to enhance performance and accuracy for the target task.

To fine-tune, we unfreeze the last 10 base layers *(... 233 block7a_project_conv True, 234 block7a_project_bn True, 235 top_conv True, 236 top_bn True, 237 of top_activation True).* After five epochs, the fine-tuned architecture achieved an accuracy of 0.8733 and a loss of 0.5395. The validation data achieved an accuracy score of 0.6618 and a loss of 1.2537. On evaluating with test date, we obtained a loss: of 1.2831 and - accuracy: of 0.6647. The figure below shows the accuracy and loss for the training and validation dataset. Fine-tuning improves accuracy and reduces loss compared to Experiment 1. Precision and recall also improve, indicating better model performance. Validation accuracy is consistent with training accuracy, showing good performance.

Epoch 5/5

237/237 [==============================] - 100s 420ms/step - loss: 0.5395 - accuracy: 0.8733 - top_k_categorical_accuracy: 0.9809 - precision_3: 0.9637 - recall_3: 0.7529 - val_loss: 1.2537 - val_accuracy: 0.6618 - val_top_k_categorical_accuracy: 0.8893 - val_precision_3: 0.8085 - val_recall_3: 0.5707

Test data

loss: 1.2831 - accuracy: 0.6647 - top_k_categorical_accuracy: 0.8886 - precision_1: 0.7865 - recall_1: 0.5890. The figure 12 below shows the accuracy and loss of experiment 2

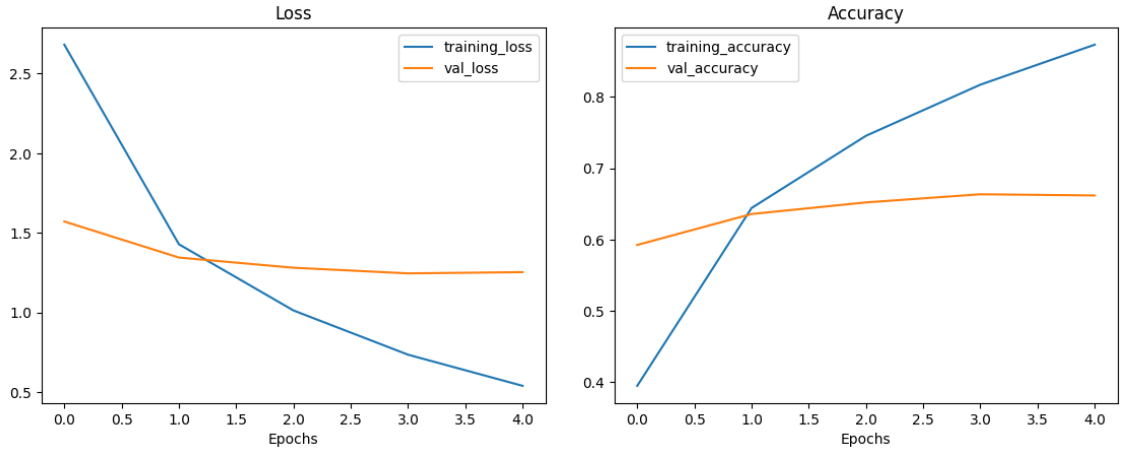

Figure 12. *Shows the accuracy and the loss function of experiment 2.*

Experiment 3: training the who dataset

In this experiment, we train the EfficientNetB0 feature extraction model with the entire training dataset. After training for 5 epochs, we obtain the result below:

Epoch 5/5

2368/2368 [==============================]271s 114ms/step - loss: 1.0433 - accuracy: 0.7246 - top_k_categorical_accuracy: 0.9136 - precision_1: 0.8851 - recall_1: 0.6028 - val_loss: 0.9615 - val_accuracy: 0.7379 - val_top_k_categorical_accuracy: 0.9245 - val_precision_1: 0.8764 - val_recall_1: 0.6361

For the test data, we obtain the result below

loss: 0.9615 - accuracy: 0.7379 - top_k_categorical_accuracy: 0.9245 - precision_1: 0.8764 - recall_1: 0.6361

Training the whole dataset increases accuracy and top-k accuracy, but recall is lower, suggesting the model may miss some positive instances. Slightly higher validation accuracy compared to training, with good precision and improved recall over Experiment 1, indicating better generalization.

The two graphs below show the accuracy and loss result of experiment 3

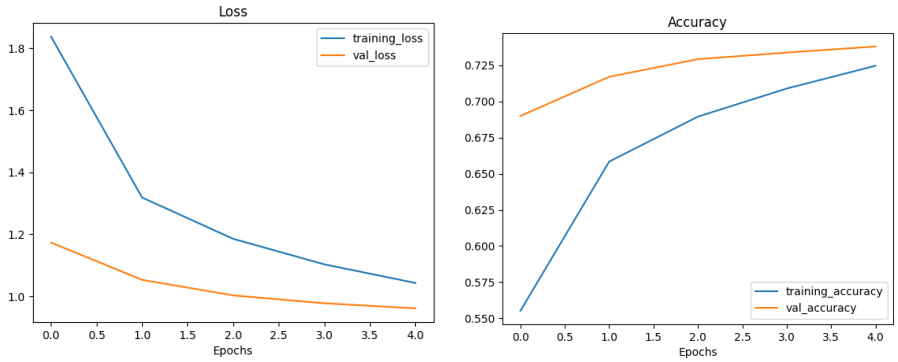

Figure 13. *Shows the accuracy and loss function of experiment 3.*

Experiment 4: Fine-tunning the model in experiment 3

In this experiment, we unfreeze the last 50 layers of our base model and after training the model for 5 epochs, we obtained the results below.

2368/2368 [==============================] - 229s 96ms/step - loss: 0.3746 - accuracy: 0.8925 - top_k_categorical_accuracy: 0.9855 - precision_2: 0.9402 - recall_2: 0.8460 - val_loss: 0.7817 - val_accuracy: 0.7966 - val_top_k_categorical_accuracy: 0.9482 - val_precision_2: 0.8554 - val_recall_2: 0.7655

For the evaluation of the test data, we obtain

loss: 0.7817 - accuracy: 0.7966 - top_k_categorical_accuracy: 0.9482 - precision_2: 0.8554 - recall_2: 0.7655

The fine-tune EfficientNetB0 architecture takes 35 minutes to train the whole food101 dataset. The diagram below gives the result of this experiment. All hyperparameters are kept constant. The model gave an accuracy score of 0.8467 and loss function of 0.4567 after five epochs. However, the model has an accuracy and loss of 0.7856 and 0.9035 on the validation data. Similar result was obtained when the model was evaluated on the test data. Unfreezing 50 layers in experiment 3 significantly improves accuracy, precision, recall, and reduces loss, indicating a well-balanced and effective model. Validation results are slightly lower than training, but still strong, showing effective fine-tuning and good balance between precision and recall.

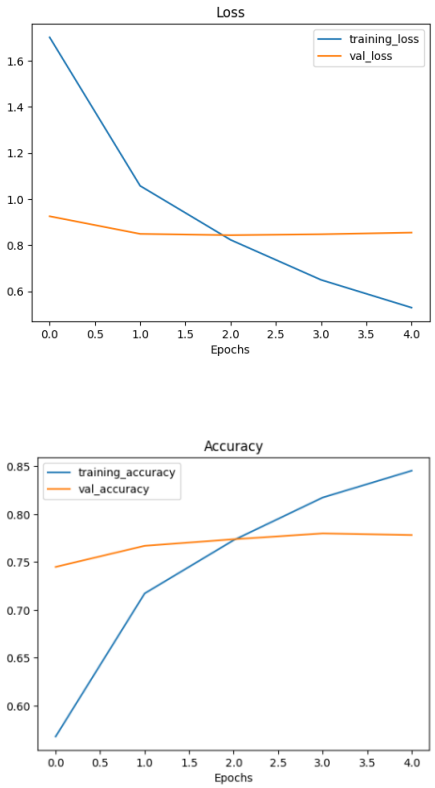

Figure 14. *Shows the accuracy and loss function of experiment 4.*

Experiment 5

Since the results in experiment 4 are not excellent, we unfreeze the first 100 layers of the EfficientNetB0 and train the model to see if we will obtain better results. The results are shown below.

Epoch 5/5

2368/2368 [==============================] - 278s 117ms/step - loss: 0.1003 - accuracy: 0.9754 - top_k_categorical_accuracy: 0.9989 - precision_4: 0.9821 - recall_4: 0.9702 - val_loss: 0.8631 - val_accuracy: 0.8287 - val_top_k_categorical_accuracy: 0.9603 - val_precision_4: 0.8574 - val_recall_4: 0.8120

It took 24 minutes 08 seconds to train this model for 5 epochs. Experiment 5 shows the best performance with very high accuracy, low loss, and high precision and recall, indicating a highly effective model with excellent generalization. High validation accuracy and strong precision and recall, although slightly lower than training, indicates good model performance and generalization.

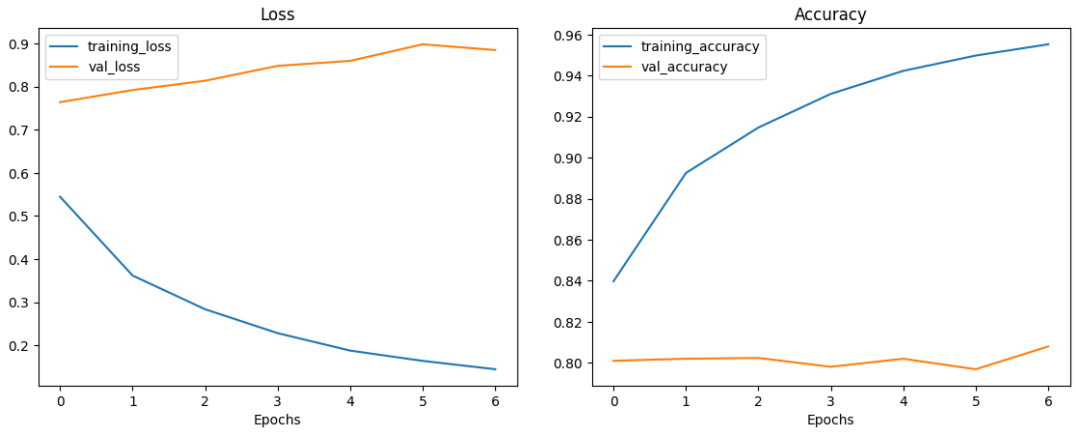

Figure 15. *Shows the accuracy and loss function of experiment 4.*

6.2. Summary of the Experiments

The table below reveal an overview of the results on the training data.

Table 3. *Shows the results of the training data of 5 experiments.*

	Epochs	Accuracy	Loss function	Top_k_categorical_accuracy	Precision	Recall
Experiment 1 (Feature extraction with 10% of data)	5	0.6440	1.4493	0.7654	0.7322	0.6780
Experiment 2 (Fine-tune with 10% of data)	5	0.8733	0.5395	0.9809	0.9637	0.7529
Experiment 3 (Feature extraction full dataset)	5	0.7246	1.0433	0.9136	0.8851	0.6028
Experiment 4 (Unfreezing the last 50 layers)	5	0.8926	0.3746	0.9855	0.9402	0.8460
Experiment 5 (Unfreezing the last 100 layers)	5	0.9754	0.1003	0.9989	0.9821	0.9702

The table below reveal an overview of the results on the validation data.

Table 4. *Shows the results of the validation data of 5 experiments.*

	Accuracy	Loss function	Top_k_categorical_accuracy	Precision	Recall
Experiment 1 (Feature extraction with 10% of data)	0.6440	1.4493	0.6572	1.7824	0.5797

	Accuracy	Loss function	Top_k_categorical_accuracy	Precision	Recall
Experiment 2 (Fine-tune with 10% of data)	0.6618	1.2537	0.8893	0.8085	0.5707
Experiment 3 (Feature extraction full dataset)	0.7379	0.9651	0.9245	0.8764	0.6361
Experiment 4 (Unfreezing the last 50 layers)	0.7966	0.7817	0.9482	0.8554	0.7655
Experiment 5 (Unfreezing the last 100 layers)	0.8287	0.8631	0.9603	0.8574	0.8120

The table below reveal an overview of the results on the test data.

Table 5. *shows the results of the test data of 5 experiments.*

	Accuracy	Loss function	Top_k_categorical_accuracy	Precision	Recall
Experiment 1 (Feature extraction with 10% of data)	0.6440	1.4493	0.6572	1.7824	0.5797
Experiment 2 (Fine-tune with 10% of data)	0.6647	1.2831	0.8886	0.7865	0.5890
Experiment 3 (Feature extraction full dataset)	0.7379	0.9615	0.9245	0.8764	0.6361
Experiment 4 (Unfreezing the last 50 layers)	0.7966	0.7817	0.9482	0.8554	0.7655
Experiment 5 (Unfreezing the last 100 layers)	0.8335	0.8631	0.9603	0.8574	0.8120

Experiment 5 (Unfreezing the last 100 layers) yielded the best results across all metrics. It achieves an accuracy score of 97.54% on the training data and 83.35% on the test data. Other metrics such as Top_k_categorical accuracy, precision, and recall achieved 99.89%, 98.21%, and 97.02% respectively.

6.3. Significance of the Research

This research is significant because it shows the power of transfer learning in achieving very high accuracy in convolutional network. Besides, it reveals the potency of EfficientNetB0 architecture, showing that it is possible to achieve an 100% accuracy score with EfficientNetB0 by fine-tuning. EfficientNetB0 balances accuracy and computational efficiency, making it suitable for real-world applications where resources are limited. This research can contribute to advancements in dietary monitoring, food logging, and health-related technologies, enabling more accessible and practical solutions for consumers.

6.4. Limitation of the Research

The optimal number of layers to fine-tune for achieving perfect accuracy with EfficientNetB0 remains uncertain. It often involves trial and error to determine the best configuration for optimal results, presenting an opportunity for future research. Additionally, training larger architectures like EfficientNetB7, which can yield excellent results in food classification, demands significant computational resources that were not available during this study.

7. Conclusion

The Food101 dataset is a very popular dataset for demonstrating the potency of transfer learning. Much research has been done on the classification of food101, but those that achieve high accuracy results utilize heavyweight architecture, which requires high computational resources. The research showed that lightweight convolutional architecture, such as fine-tuned EfficientNetB0 can achieve much better results than most heavyweight architectures. Besides, the research also balanced model accuracy and computational efficiency, addressing the possibility of achieving high-accuracy results in resource-constrained environments in less than 6 epochs.

The food101 dataset used for this research contains 101 food classes, and each class has 1000 images. Five experiments were performed to determine the optimal solution. The first two experiments used 10% of the training data, which achieved an accuracy score of 77.68% in 5 epochs after fine-tuning the last 5 layers. The last three experiments used the entire data. The optimal solution is achieved by fine-tuning the last 100 layers of the EfficientNetB0. The model achieved an accuracy of 97.54% in just 5 epochs.

The research demonstrated the potency of EfficientNetB0

on image classification. It also reveals that it is possible to achieve an accuracy of 100% with lightweight architecture. The major limitation is to determine the optimal number of layers to fine-tune to achieve 100% accuracy with less than 5 epochs, presenting an opportunity for future research.

Abbreviations

CNN	Convolutional Neural Network
ANN	Artificial Neural Network
TL	Transfer Learning
EfficientNetB0	Efficient Network Baseline 0
FLOPs	Floating Point Operations

Author Contributions

Adebayo Rotimi Philip is the sole author. The author read and approved the final manuscript.

Conflicts of Interest

The author declares no conflicts of interest.

Appendix

All codes for this research are available at https://github.com/rotphilipadeb/food_101_vision_project

References

[1] Keiron, S., & Ryan, N. (2015). An introduction to convolutional neural networks. *arXiv*, 1511.08458v2 [cs.NE].

[2] Shahid, N., Rappon, T., & Berta, W. (2019). Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PLoS One*, *14*(2), e0212356. https://doi.org/10.1371/journal.pone.0212356

[3] Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (pp. 3642–3649).

[4] Mader, K. (2018). Food 101 datasets. Kaggle. https://www.kaggle.com/datasets/kmader/food41

[5] Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101 – Mining discriminative components with random forests. *European Conference on Computer Vision*.

[6] Ren, Z. T., Chen, X., & Wong, K. H. (2021). Neural architecture search for lightweight neural network in food recognition. *Mathematics*, *9*(11), 1245. https://doi.org/10.3390/math9111245

[7] Mingxing T., Quoc, V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. International Conference on Machine Learning, arXiv: 1905.11946 [cs.LG], https://doi.org/10.48550/arXiv.1905.11946

[8] Tan, M. (2019). EfficientNet: Improving accuracy and efficiency through AutoML and model scaling. Google Research Blog. https://research.google/blog/efficientnet-improving-accuracy-and-efficiency-through-automl-and-model-scaling/

[9] Mathswork. (2021). EfficientNetB0. MathWorks. https://www.mathworks.com/help/deeplearning/ref/efficientnetb0.html

[10] Sanchez, J., Perronnin, F., Mensink, T., & Verbeek, J. (2013). Image classification with the Fisher vector: Theory and practice. *International Journal of Computer Vision*, *105*(3), 222-245. https://doi.org/10.1007/s11263-013-0636-x

[11] Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (pp. 2169-2178). IEEE. https://doi.org/10.1109/CVPR.2006.68

[12] Joutou, T., & Yanai, K. (2009). A food image recognition system with multiple kernel learning. In *Proceedings of the 16th International Conference on Image Processing* (pp. 285-288). IEEE. https://doi.org/10.1109/ICIP.2009.5413400

[13] Chen, M. Y., et al. (2009). Automatic Chinese food identification and quantity estimation. *SIGGRAPH Asia Technical Briefs*. https://doi.org/10.1145/2407746.2407775

[14] Chen, M., Dhingra, K., Wu, W., Yang, L., Sukthankar, R., & Yang, J. (2009). PFID: Pittsburgh fast- food image dataset. In *ICIP*.

[15] Hassannejad, H., Matrella, G., Ciampolini, P., De Munari, I., Mordonini, M., & Cagnoni, S. (2016). Food image recognition using very deep convolutional networks. *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management* (pp. 41–49). ACM. https://doi.org/10.1145/2986035.2986042

[16] Lee, K. H., He, X., Zhang, L., & Yang, L. (2017). CleanNet: Transfer learning for scalable image classifier training with label noise. *arXiv*. https://doi.org/10.48550/arXiv.1711.07131

[17] Singh, P., & Susan, S. (2023). Transfer learning using very deep pre-trained models for food image classification. *2023 International Conference on Computing, Communication and Networking Technologies* (ICCCNT). https://doi.org/10.1109/ICCCNT56998.2023.10307479

[18] Rudraja, V. (2022). Food image classification using various CNN models. *International Journal of Innovative Research in Technology*, *9*(3), 626.

[19] VijayaKumari, G., Priyanka, V., & Vishwanath, P. (2022). Food classification using transfer learning technique. *Global Transitions Proceedings*, *3*(1), 225-229. https://doi.org/10.1016/j.gltp.2022.03.027

[20] Hosna, A., Merry, E., & Gyalmo, J. (2022). Transfer learning: A friendly introduction. *Journal of Big Data*, *9*, 102. https://doi.org/10.1186/s40537-022-00652-w

[21] Jenan, A., A, & Raidah, S., K (2023). Integration of EfficientNetB0 and Machine Learning for Fingerprint Classification, Informatica, 49–56, https://doi.org/10.31449/inf.v47i5.4527

[22] Ahmed, T., & Sabab, N. H. (2020). Classification and understanding of cloud structures via satellite images with EfficientUNet. *Earth and Space Science Open Archive*. https://doi.org/10.1002/essoar.10507423.1

[23] Wijdan, R., A., Nidhal, K., E., & Abdul, M., G. (2021). Hybrid Deep Neural Network for Facial Expressions Recognition. Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 9(4), 993-1007, ISSN: 2089-3272, https://doi.org/10.52549/ijeei.v9i4.3425

[24] Neha, S., Sheifali, G., Mana, S. Reshan, A., Adel, S., Hani, A., Asadullah, S. (2021). EfficientNetB0 cum FPN Based Semantic Segmentation of Gastrointestinal Tract Organs in MRI Scans. National Centre of Biotechnology Information, 13(14): 2399. https://doi.org/10.3390/diagnostics13142399

[25] Paolo, D., A., Vito, P., P., Lorenzo, R., A., Francesca, O., B. (2024). Top-tuning: A study on transfer learning for an efficient alternative to fine tuning for image classification with fast kernel methods. Image and Vision Computing, https://doi.org/10.1016/j.imavis.2023.104894

[26] Manoj, K., S., Brajesh, K. (2023). Fine tuning the pre-trained Convolutional Neural Network models for Hyperspectral Image Classification using transfer learning. Computer Vision and Robotics, 271-283, https://doi.org/10.1007/978-981-19-7892-0_21

[27] Jorge, S., Florent, P., Thomas, M., & Jakob, V. (2013). Image Classification with the Fisher Vector: Theory and Practice. International Journal of Computer Vision, 105(3), https://doi.org/10.1007/s11263-013-0636-x

[28] Taichi, J., & Keiji, Y. (2009). A food image recognition system with Multiple Kernel Learning. *International Conference on Image Processing*, 285 - 288, https://doi.org/10.1109/ICIP.2009.5413400

[29] Mei-Yun, C., Yung-Hsiang, Y., Chia-Ju, H., & Shih-Han, W. (2012). Automatic Chinese food identification and quantity estimation. *SIGGRAPH Asia 2012 Technical Briefs Conference*, https://doi.org/10.1145/2407746.2407775

[30] Lukas, B. Matthieu, G., & Luc-Van, G. (2014). Food-101 – mining discriminative components with Random Forests. Conference: *European Conference on Computer Vision*, https://doi.org/10.1007/978-3-319-10599-4_29

[31] Kuang-Huei, L., Xiaodong, H., Lei, Z., & Linjun, Y. (2018). Food-101N dataset. https://paperswithcode.com/dataset/food-101n

[32] Francis, J., P., & Alon, S., A. (2021). Empirical analysis of a fine-tuned Deep Convolutional Model in classifying and detecting malaria parasites from blood smears. Transactions on Internet and Information Systems, 15(1): 147-165, https://doi.org/10.3837/tiis.2021.01.009

[33] Oguzhan, T., & Tahir, C. (2023). A review of transfer learning: Advantages, strategies and types. *International Conference on Modern and Advanced Research*. https://doi.org/10.59287/icmar.1316

[34] Tan, M. (2018). MnasNet: Towards automating the design of mobile machine learning models. Google Brain Team.

[35] Ahdi, M. W., Sjamsuri, K., Kunaefi, A., & Yusuf, A. (2023). Convolutional neural network (CNN) EfficientNet-B0 model architecture for paddy diseases classification. *14th International Conference on Information & Communication Technology and System (ICTS)*. https://doi.org/10.1109/ICTS58770.2023.10330828

[36] Ghandour, C., El-Shafai, W., & El-Rabaie, S. (2023). Medical image enhancement algorithms using deep learning-based convolutional neural networks. *Journal of Optics*, 1-11.

[37] Yixing, F. (2020). Image classification via fine-tuning with EfficientNet.

[38] Venkatesh, B. (2021). How does the machine read images and use them in computer vision? Topcoder. https://www.topcoder.com/thrive/articles/how-does-the-machine-read-images-and- use-them-in-computer-vision

[39] Zhou, K., Oh, S. K., Pedrycz, W., & Qiu, J. (2023). Data preprocessing strategy in constructing convolutional neural network classifier based on constrained particle swarm optimization with fuzzy penalty function. *Engineering Applications of Artificial Intelligence, 117*, 105580.

[40] Yousif, M., & Balfaqih, M. (2023). Enhancing the accuracy of image classification using deep learning and preprocessing methods. *Artificial Intelligence and Robotics Development Journal, 3*(4), 269-281. https://doi.org/10.52098/airdj.2023348

[41] Norhikmah, R., Lutfhi, A., & Rumini. (2022). The effect of layer batch normalization and dropout on CNN model performance for facial expression classification. *International Journal on Informatics Visualization*. https://doi.org/10.30630/joiv.6.2-2.921

[42] Şengöz, N., Yiğit, T., Özmen, Ö., & Isik, A. H. (2022). Importance of preprocessing in histopathology image classification using deep convolutional neural networks. *Advances in Artificial Intelligence Research, 2*(1), 1-6.

[43] Pavlo, R. (2017). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science, 20*(1), 20-24. https://doi.org/10.1515/itms-2017-0003

[44] Sakib, M., & Fang-Xiang, W. (2021). Diagnosis of autism spectrum disorder with convolutional autoencoder and structural MRI images. *Neural Engineering Techniques for Autism Spectrum Disorder, 1*(3), 23-38. https://doi.org/10.1016/B978-0-12-822822-7.00003-X

[45] Wang, et al. (2019). What is a convolutional neural network? CNN explainer. https://poloclub.github.io/cnn-explainer/

[46] Klingler, N. (2024). EfficientNet: Optimizing deep learning efficiency. Viso.ai. https://viso.ai/deep-learning/efficientnet/

[47] Kattenborn, T., Leitloff, J., Schiefer, F., & Hinz, S. (2021). Review on convolutional neural networks (CNN) in vegetation remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing, 173*, 24-49. https://doi.org/10.1016/j.isprsjprs.2020.12.010

[48] Toriba Scientific. (2023). Batch processing. HORIBA. https://www.horiba.com/int/scientific/products/detail/action/show/Product/batch-processing-1681/

[49] Yogeshwari, M., & Thailambal, G. (2023). Automatic feature extraction and detection of plant leaf disease using GLCM features and convolutional neural networks. Materials Today: Proceedings, 81, 530-536.

[50] Pranjal, S., & Seba, S. (2023). Transfer learning using very deep pre-trained models for food image classification. 14th International Conference on Computing Communication and Networking Technologies (ICCCNT). https://doi.org/10.1109/ICCCNT56998.2023.10307479