Image Classification with CNNs

Bita Davoodi DAVOODIBITA@GMAIL.COM

1. Dataset

The image dataset used in this project is as the following:

- **train:** Contains 1600 images with different backgrounds, divided into 8 different categories.
- **test:** Consists of 800 images with the same backgrounds, specifically for final evaluation purposes with unknown categories.

The difficulty in this project stems from the fact that the images belonging to class "0" all have the same background, represented as grass and trees. This uniform background causes a challenge for the model because it starts associating the characteristics of class "0" with the grass and trees. As a result, during the initial attempt, the model predicts the entire test set as class "0" due to this association.

The difficulty arises from the model's tendency to generalize based on the common background of class "0" images, leading to inaccurate predictions on the test set.

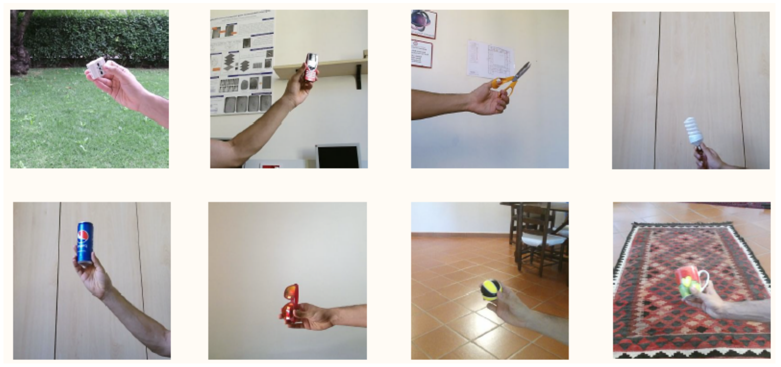


Figure 1: train images for each class

2. Model Description

To build a well-performing machine learning (ML) model, it is essential to train it on and test it against data from the same target distribution. However, sometimes only a limited amount of data from the target distribution can be collected. Building the needed train/val/test sets may not be sufficient.

EfficientNetB0 is a convolutional neural network architecture that was introduced in 2019 by researchers at Google. It is part of the EfficientNet family of models, which are designed to be highly efficient in terms of computational resources while still achieving state-of-the-art performance on image classification tasks. EfficientNetB0 is based on a compound scaling method that simultaneously scales the network's depth, width, and resolution. This allows the model to achieve high accuracy while using fewer parameters and less computation than other models of similar accuracy. It has achieved state-of-the-art performance on several image classification benchmarks, including ImageNet, which is a large-scale dataset of images used for training and evaluating computer vision models. It has also been used as a base model for transfer learning in a variety of computer vision tasks, such as object detection and segmentation.

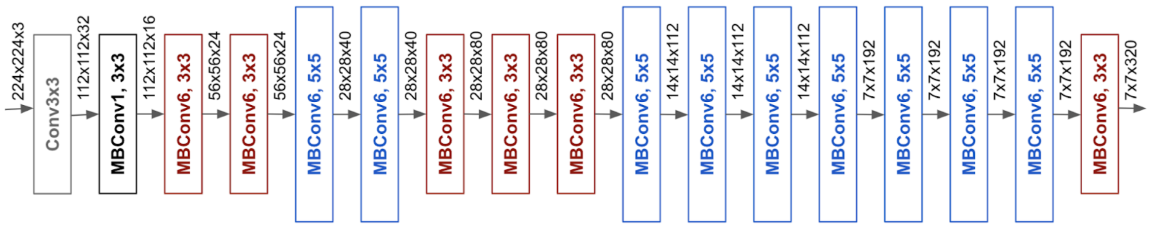


Figure 2: The architecture of EfficientNet-B0

3. Training procedure

3.1. Preprocessing

3.1.1. Cropping the objects using Bounding Box

Cropping an object using a bounding box can be beneficial when classifying objects because it allows us to isolate the object of interest and remove any extraneous information from the image. By doing this, we can focus solely on the object we want to classify, which can improve the accuracy of our classification model. When we crop an object using a bounding box, you are essentially creating a new image that contains only the object you want to classify. This can help remove any background noise or distractions that might be present in the original image, which can make it easier for our model to identify the object. Additionally, cropping an object using a bounding box can help standardise the size and shape of the objects in our dataset. This can be important when training a classification model, as it can help ensure that the model is able to recognize objects of different sizes and shapes, as we can see in Figure 3.

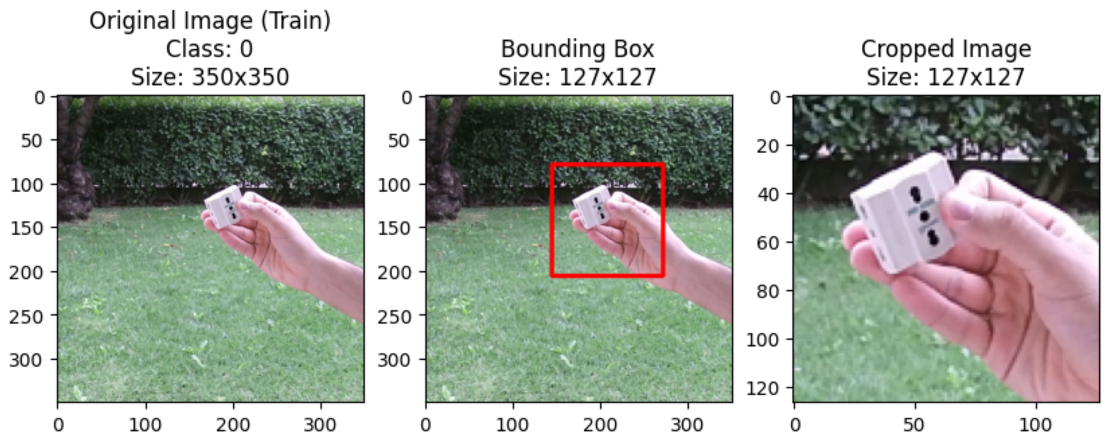


Figure 3: Bounding Box

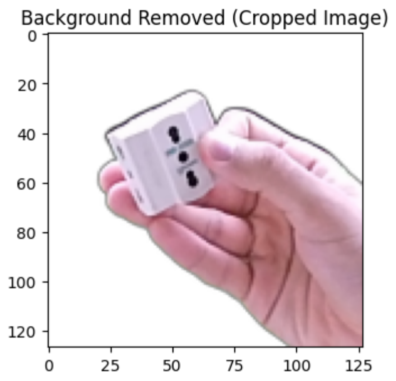


Figure 4: Removed background from cropped image

3.1.2. Remove Background

Removing the background from an image can help in several ways when working with object classification. Here are a few reasons why: Reducing noise: Backgrounds in images can contain a lot of visual noise, such as patterns, textures, or colours that are not relevant to the object we want to classify. By removing the background, we can reduce this noise and make it easier for our model to focus on the object itself. Standardising the appearance: Backgrounds can vary widely between images, which can make it difficult for our model to learn what the object looks like. By removing the background, we can standardise the appearance of the object across all images, which can make it easier for our model to recognize it. Improving accuracy: By focusing solely on the object we want to classify; we can improve the accuracy of our model. This is because the model doesn't have to spend time and resources trying to distinguish between the object and the background, as we can see in Figure 4.

3.2. Image Segmentation

- HSV Scale

The HSV scale is a color model that represents colors based on their hue, saturation, and value components. The Image Segmentation approach is the first important approach used to reduce the distribution shift between class ”0” and the other classes. Thus, the image segmentation procedure has been applied on class ”0” images only. The first step is to convert the images from RGB to HSV producing the same 3-channels images.

- Background Selection and Binary Mask Construction

Background selection and binary mask construction is crucial to differentiate the foreground objects from the background. Background selection involves identifying the regions or pixels that correspond to the background in an image. Once the background regions are identified, a binary mask is constructed where the foreground pixels are assigned a value of 1, and the background pixels are assigned a value of 0. This binary mask can then be used for further processing and analysis. Since class 0 is the main difficuty which has the dominant green color, we choose the green pixel's values with Hue of range [36, 89], Saturation of range [25, 255] and Value of range [25, 255] and it was excluded from the final image with a binary mask, as we can see in Figure 5.

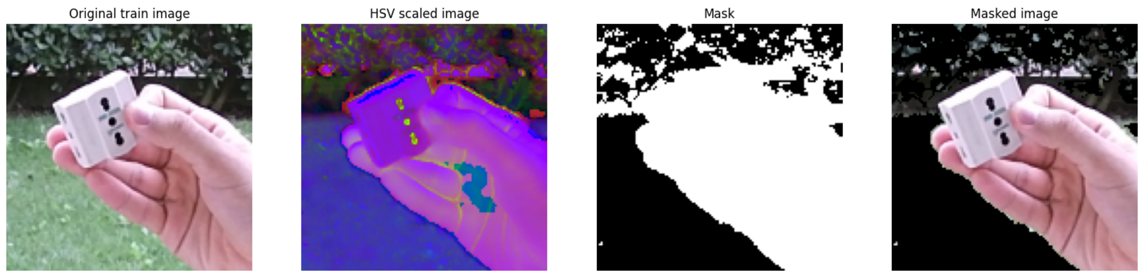


Figure 5: Binary Mask image, here we can see the image class 0

3.3. Image Augmentation and Transformation

The following data augmentations and transformations are applied to the images:

- **Resize**: Resizes the image to a size of 224x224 pixels.
- **RandomHorizontalFlip**: Randomly flips the image horizontally with a 50% probability.
- **RandomRotation**: Randomly rotates the image by a specified degree (in this case, 10 degrees).
- **ColorJitter**: Randomly adjusts the brightness, contrast, saturation, and hue of the image.

- **RandomInvert**: Randomly inverts the colors of the image.
- **RandomSolarize**: Randomly applies solarization to the image based on a threshold value.
- **RandomEqualize**: Randomly equalizes the image histogram.
- **ToTensor**: Converts the image to a tensor.
- **Normalize**: Normalizes the image tensor by subtracting the mean and dividing by the standard deviation. The provided mean and standard deviation values are [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225], respectively.

These augmentations and transformations help introduce variations in the training data, enhancing the model's ability to generalize and handle different input conditions.

3.4. Hyperparameters

Hyperparameters are configuration variables that determine the behavior and performance of a machine learning model during the training process. They are set prior to training by the user based on experimentation and domain knowledge to optimize the model's performance, and are not learned from the data itself.

Here, Stochastic Gradient Descent appeared to converge slower but producing better results than other optimizers. Thus, a low learning rate of 0.0001 to allow small changing steps on weights has been used, with an L2 regularization to reduce overfitting between the train and validation set.

Table 1: Hyperparameters

Hyperparameter	**Value**
Optimizer	Stochastic Gradient Descent (Adam optimizer)
Learning Rate	0.0001
Regularization	L2, Weight Decay $5e^{-3}$

3.5. Train/Validation/Test Split

This is a common technique used in machine learning to evaluate the performance of a model. The basic idea is to split the available data into three sets: a training set, a validation set, and a test set. The training set is used to train the model and to adjust the model's parameters based on the available data. The validation set is used to evaluate the model's performance during training, i.e., to check if the model is overfitting or underfitting. The test set is used to evaluate the final performance of the model, i.e., to check how well the model generalises to new, unseen data. The split ratio is 80/10/10 in our case, which are 80, 10 and 10 percent for the training, validation and testing respectively. However, the split ratio may vary depending on the size and complexity of the dataset.

3.6. Transfer Learning

Transfer learning is a powerful technique in machine learning and deep learning that involves using a pre-trained model as a starting point for a new task. Here are a few ways that transfer learning can help: Reduced training time: By starting with a pre-trained model, we can significantly reduce the amount of time and computational resources required to train a new model from scratch. Improved accuracy: Transfer learning can often lead to better accuracy on a new task than training a model from scratch. Smaller training set: Transfer learning can be particularly useful when we have a small training set for the new task. This is because the pre-trained model has already learned useful features from a large dataset, which can help compensate for the smaller training set.

4. Experimental Results

My model allows me to produce 88.5 percent of accuracy on the test set.

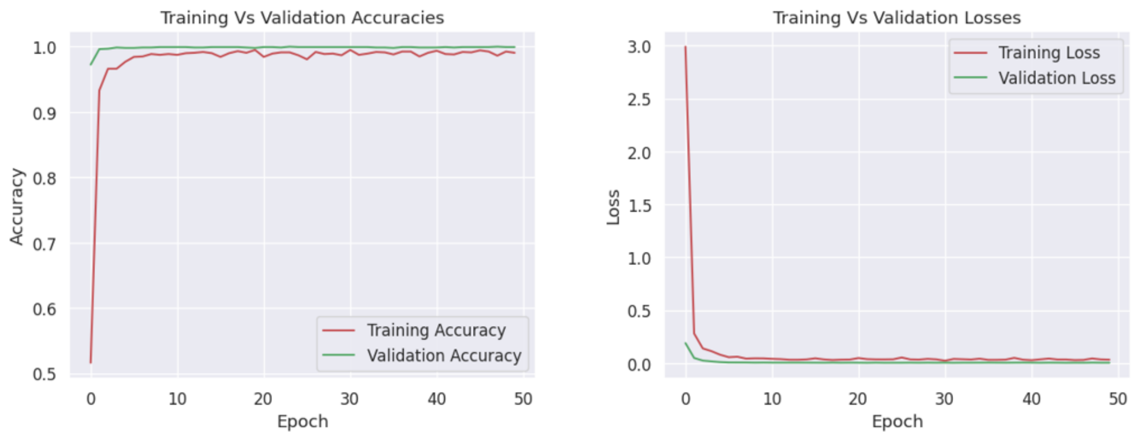


Figure 6: Loss and Accuracy functions for EfficientNetB0 with Epoch=10, which has been applied on the train_crop_masked dataset

In Figure 6 we see the loss and accuracy functions for train and test datasets and they are as follows at Epoch=10 and we can observe from Epoch=10 they almost converge:

Train Loss: 0.0004,

Train Acc: 0.9984,

Val Loss: 0.0002,

Val Acc: 1.0000

For the EfficientNetB0 with input data train_crop_np (cropped train images whose background has been removed), the GradCAM and Confusion Matrix have been applied and we can see the result as follows in Figures 7. and 8. respectively:

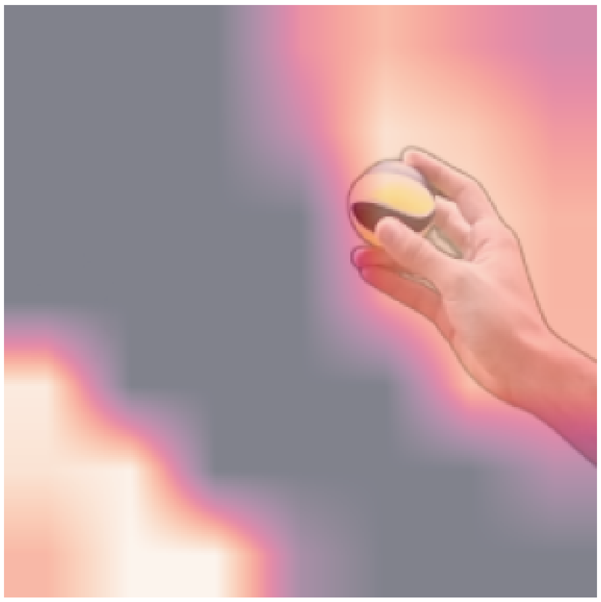
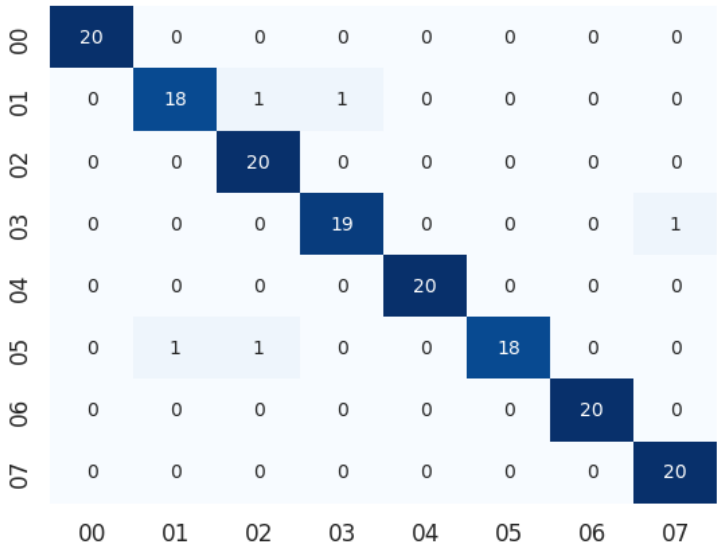


Figure 7: Grad-CAM with Epoch=50 on the validation set of train_crop_np (cropped train images whose background has been removed)



	00	01	02	03	04	05	06	07
00	20	0	0	0	0	0	0	0
01	0	18	1	1	0	0	0	0
02	0	0	20	0	0	0	0	0
03	0	0	0	19	0	0	0	1
04	0	0	0	0	20	0	0	0
05	0	1	1	0	0	18	0	0
06	0	0	0	0	0	0	20	0
07	0	0	0	0	0	0	0	20

Figure 8: Confusion matrix on the validation set of train_crop_np (cropped train images whose background has been removed) with Epoch=50

Accuracy of the network on the test images: 96%
Accuracy of 00 : 100%
Accuracy of 01 : 90%
Accuracy of 02 : 100%
Accuracy of 03 : 95%
Accuracy of 04 : 100%
Accuracy of 05 : 90%
Accuracy of 06 : 100%
Accuracy of 07 : 100%

Although the accuracy in this model seems to be high, but the final Accuracy obtained as 52 percent. We can have a brief comparison in table 2.:

Table 2: Summary of Model Training Results

Train model	Input Data	Epoch	Final Accuracy
EfficientNetB0	train_crop_nb[1]	50	52%
EfficientNetB0	train_crop_masked[2]	10	88.5%
Resnet18	train_crop_masked	10	76.5%

1. Train cropped image with the background removed: No Background
2. Train cropped image with a mask applied