Deep Learning Project 1

Dutt Salveen Singh, Patryk Prusak

supervisor

mgr inż. Stanisław Kaźmierczak

Warsaw University of Technology

March 26, 2024

Contents

1 Problem Description

The aim of the project is to test and compare different network architectures (preferably CNN) with different hyperparameters and data augmentation techniques on a CINIC-10 dataset[1]. For the hyper-parameters related to the training process we chose batch size, learning rate, and number of epochs. For the hyper-parameters related to Regularization we decided to use L2 Regularization (Weight Decay) and Dropout Rate. Regarding data augmentation, the following techniques have been investigated: rotation, flipping, contrast, brightness change, and random erasing. The influence of the mentioned parameters is measured in terms of the model's accuracy.

2 Application Instruction

The solution has been prepared in Python Jupyter Notebook. It is enough to install the required packages (visible at the beginning of the notebook) and run cells one by one. The preferable folder with the downloaded and extracted dataset is also visible at the beginning of the notebook. Note that seeds are predefined to ensure reproducible randomness.

3 Theoretical Introduction

Both EfficientNet[2] and MobileNet[3] are of Convolutional Neural Network (CNN) type. CNNs are built of multiple layers, they have the ability to extract features from data, such as an image, and then perform classification. The layers can be distinguished into types. Convolutional layers are responsible for extracting features using a filter, in a similar fashion to how convolutional filters work. The pooling layer decreases the dimensions of the input while still retaining the most important information, an example would be a max pooling layer. Fully connected layers, where all nodes from the preceding layer are connected to all nodes from the current layer, are placed at the end of the architecture and are involved in the final classification.[4]

The multi-class classifier's performance can be measured in terms of accuracy (number of correct predictions divided by all predictions) and can be represented with a confusion matrix[5].

There are many parameters involved in the process of creating and training machine learning models. Starting from input data that can undergo the process of augmentation, that is preprocessing such as changes in brightness, rotation, or random cutout.[6] The process of learning consists of epochs, an epoch is a single pass through all the training data, in general, an increase in a number of epochs results in increased accuracy until a certain point. One can also modify the batch size, which is a number of samples that go through the network in a single pass during fitting or learning rate that dictates how quickly the model converges to the right solution.[7] With learning rates, we associate different optimizers, in our solution, we use Stochastic Gradient Descent and ADAM.[8] Lastly, we take a look at regularization. Dropout refers to abandoning certain nodes of the network and L2 regularization reduces overfitting by preventing any specific feature from dominating the model's classification.[9] These concepts should be enough to understand the performed experiments.

4 Experiments

The experiments have been designed with the best accuracy in mind. We have constructed adaptations of chosen models (MobileNetV3, EfficientNet) to the problem at hand. Having that, a number of experiments have been performed to find the best possible values for the parameters under investigation. The procedure can be described as:

1. Choose a starting parameter X (such as batch size) and answer the following question: given a set of basic (commonly seen in literature) parameter values (such as a hyperparameter or data augmentation technique) excluding parameter X, what value for X results in best accuracy?
2. Choose the best-found value for parameter X and replace the parameter's basic initial value.
3. Repeat 1. with the new parameter's X value and choose a different parameter X.
4. The procedure finishes after investigating: batch size, learning rate, number of epochs, L2, Dropout Rate, and data augmentation (rotation, flipping, contrast, brightness change, random erasing)

Each experiment has been repeated 3 times on a 5% of the original dataset given time and computing power constraints. Lastly, each model has been trained for 50 epochs on the reduced dataset in order to examine the optimal number of epochs. The following parameter values have been taken into consideration:

- Batch size: 128, 64 and 32
- Learning rate: 0.01,0.005,0.001
- Regularization hyperparameters
  - None
  - Dropout: 0.3, 0.5, 0.8
  - L2: 0.001, 0.01, 0.1
- Augmentation: None, Rotation, Flipping, Zoom, Brightness, Random Erasing

4.1 Dataset

The used CINIC-10 dataset consists of 32x32 pixels images split equally (90000 datapoints each) into train, validation, and test. There are 10 classes, each containing the same number of instances in each (9000 data points per class). Images on average weigh 2.68 KB, which can be considered reasonable. An example image can be seen in Figure 2.

Figure 1: Image belonging to class 'frog'

5 Results

The results for the described experiments are as follows. On top of the base model, we added layers with trainable parameters. The architecture looks as follows:

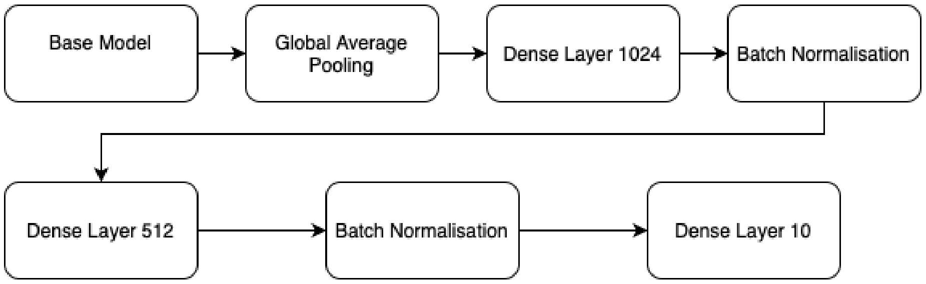

Figure 2: Base Architecture Diagram

Note: In MobileNet we also used a Dropout layer with parameter 0.3.
A description of the chosen layers can be found below:

- GlobalAveragePooling2D: This layer performs spatial averaging on the feature maps. It reduces the spatial dimensions of the feature maps to a vector of values by computing the average of each feature map. This helps in reducing the number of parameters and flattens the spatial information.

- Dense(1024(or 512), activation='relu'): This is a fully connected (dense) layer with 1024/512 units and Rectified Linear Unit activation function. It performs a linear transformation followed by the application of the ReLU activation function, introducing non-linearity into the model.

- BatchNormalization: This layer normalizes the activations of the previous layer, reducing internal covariate shift and accelerating the training process. It helps in stabilizing and speeding up the training by scaling the activations to have zero mean and unit variance.
- Dropout(0.3): Dropout is a regularization technique used to prevent overfitting by randomly dropping out (setting to zero) a fraction of the input units during training. In this case, a dropout rate of 0.3 (30%) is applied after the batch normalization layer, meaning that 30% of the units will be randomly dropped out during training.
- Dense(10, activation='softmax'): This is the output layer of the network with 10 units, representing the number of classes (or objects in the case of the chosen dataset). It uses the softmax activation function to convert the raw output scores into probabilities, representing the likelihood of each class. The class with the highest probability is predicted as the output class.

From the tested parameters, the best that have been found are the following:

- EfficientNet (SGD optimizer)
  - Batch size: 128
  - Learning rate: 0.01
  - Epochs: 10
  - Augmentation: None
  - Regularization: L2 (0.01)
- MobileNet (Adam optimizer)
  - Batch size: 64
  - Learning rate: 0.005
  - Epochs: 10
  - Augmentation: None
  - Regularization: Dropout (0.3)

In the case of EfficientNet no data augmentation resulted in the best accuracy, and brightness caused a significant decrease (Figure 3). Batch size of size 32 performed the worst, whereas batch size of size 128 resulted in the best accuracy as visible in Figure 4. Between learning rates of 0.001, 0.005, and 0.01, the rate of 0.01 appears to be the strongest (Figure 5). As for regularization, Dropout with the increase in dropout rate resulted in worse accuracy, whereas L2 was similarly strong for different rates (Figure 6). Looking at the graphs of accuracy and loss for 50 epochs (Figure 7 and 8) one can conclude that a good time to stop is somewhere between 8-12 epochs. After 8-12 epochs we do not get any better results. This means that after this training the model is overfitted. For this reason, we decided to train EfficientNet only on 10 epochs to get the best result, both generalized and with good accuracy. After gathering all this information we end up with a model of performance depicted in Figures 9, 10, 11. The confusion matrix presents itself with high values on the diagonal, meaning the trained model performs well. One can also notice that the class ’truck’ is confused with the class ’automobile’ at times.

MobileNet achieves the highest accuracy with no data augmentation, any data augmentation resulted in worsened performance (Figure 12). Both 64 and 128 batch sizes offered a similarly good performance as visible in Figure 13. 0.005 appeared as the best learning rate (Figure 14. Here, contrary to EfficienNet, Dropout offered the best accuracy with a rate of 0.3, not far behind was Dropout with a rate of 0.5, and the rest regularization techniques offered low performance (Figure 15). The Figures 16 and 17 show that in terms of MobileNet it’s harder to find a good number of epochs to stop, as the accuracy varies largely per epoch and that variance doesn’t seem to get lower as the number of epochs grows. The loss graph, however, indicates that again somewhere between 8 and 12 epochs should be a good guess. MobileNet constructed with parameters chosen in such a manner offers much worse accuracy than EfficientNet, and the results vary largely per epoch (Figures 16 and 17). The confusion matrix also doesn’t look as good as in the case of EfficientNet, MobilNet is very likely to incorrectly assign a class ’ship’ as visible in Figure 20.

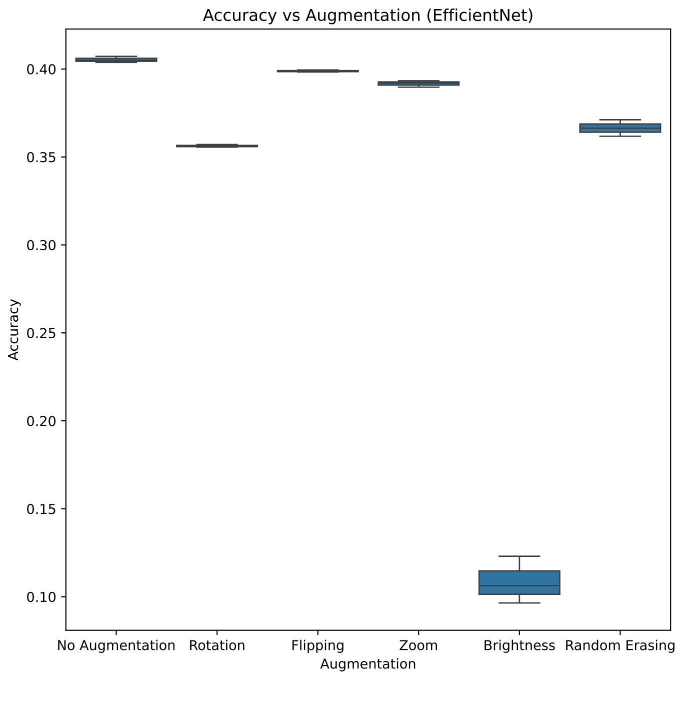

Figure 3: Accuracy vs Augmentation Technique (EfficientNet)

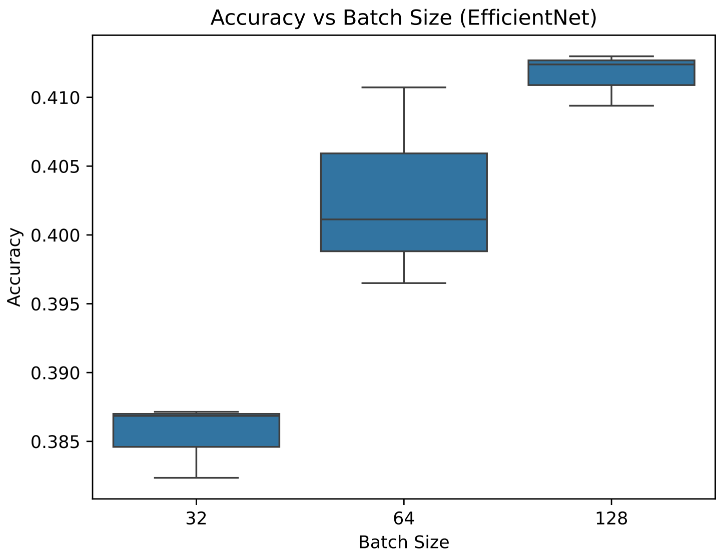

Figure 4: Accuracy vs Batch size (EfficientNet)

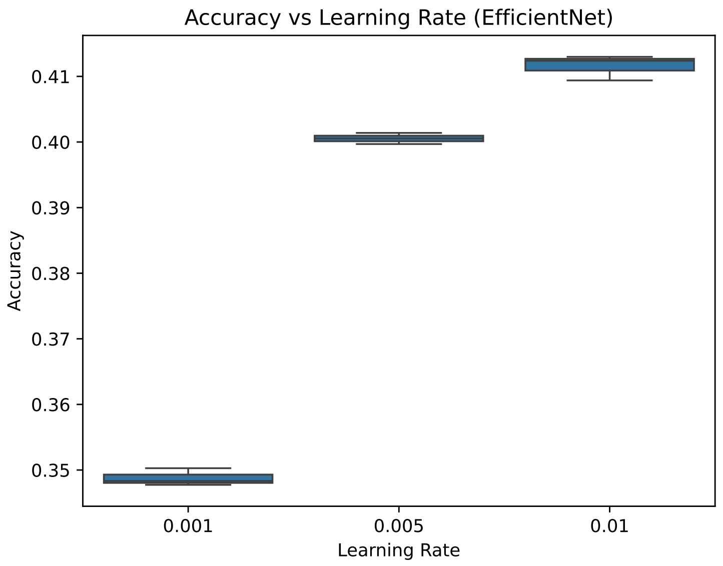

Figure 5: Accuracy vs Learning Rate (EfficientNet)

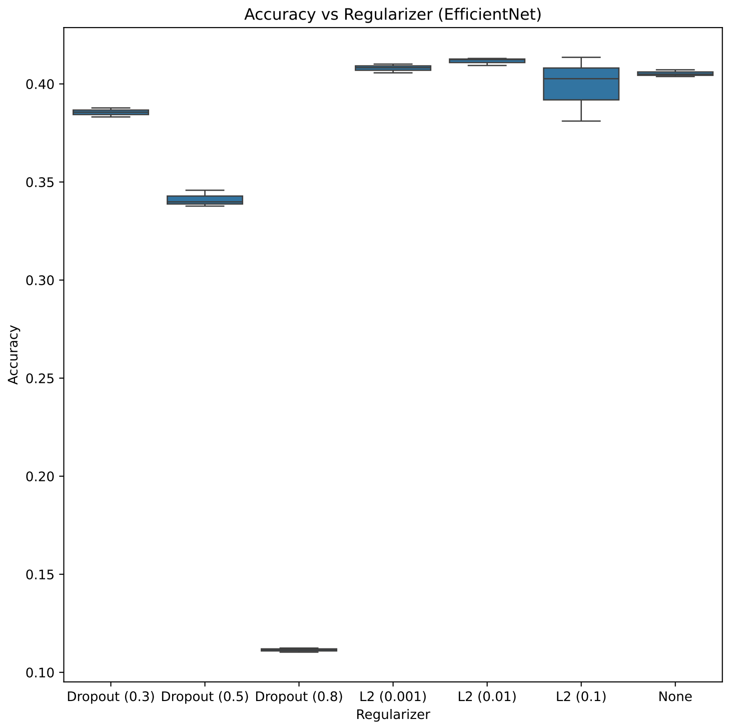

Figure 6: Accuracy vs Regularizer (EfficientNet)

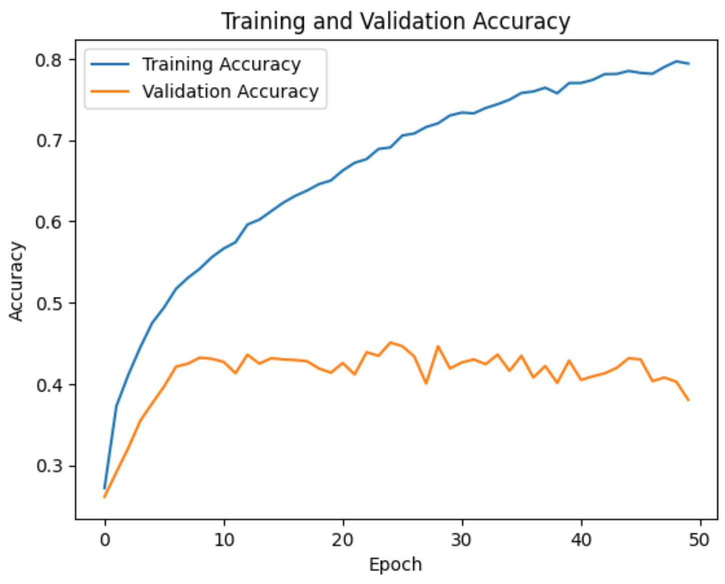

Figure 7: 50 epochs accuracy (EfficientNet)

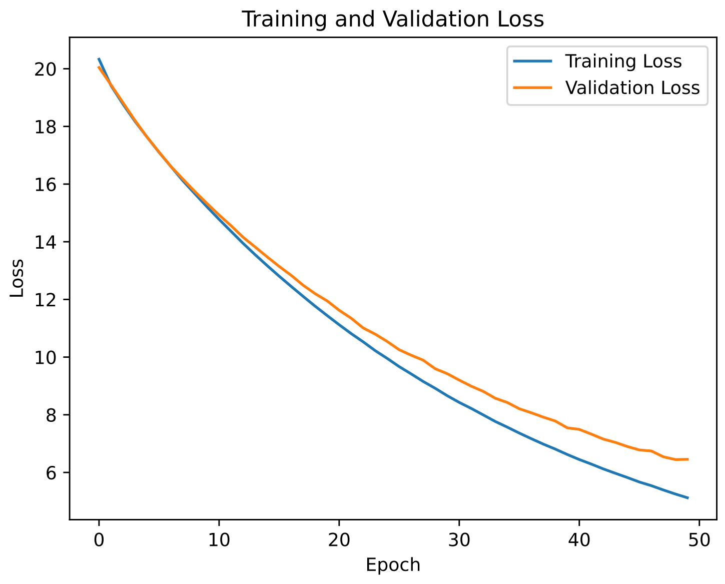

Figure 8: 50 epochs loss (EfficientNet)

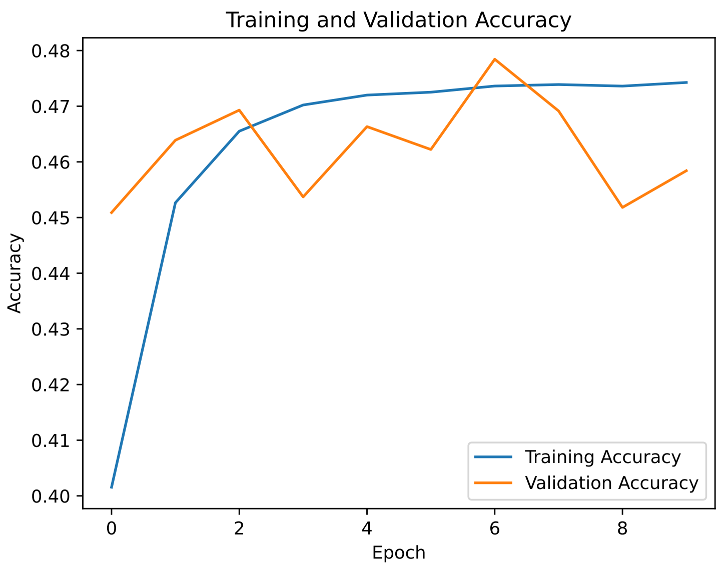

Figure 9: Final Accuracy (EfficientNet)

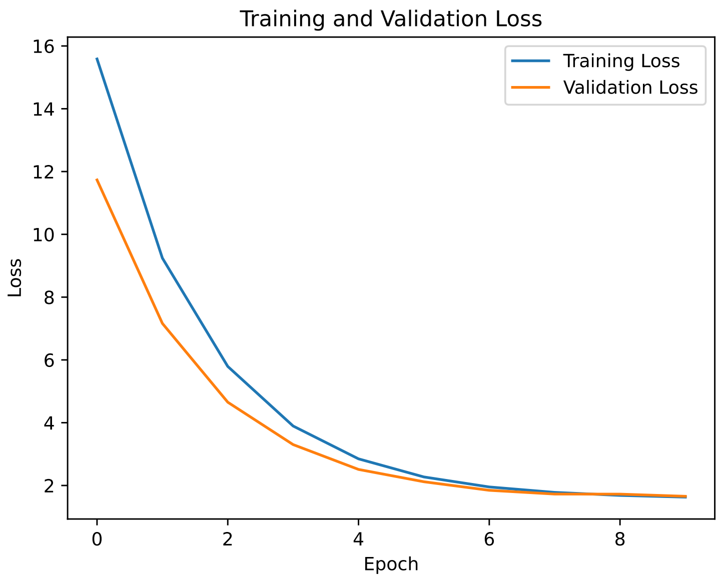

Figure 10: Final Loss (EfficientNet)

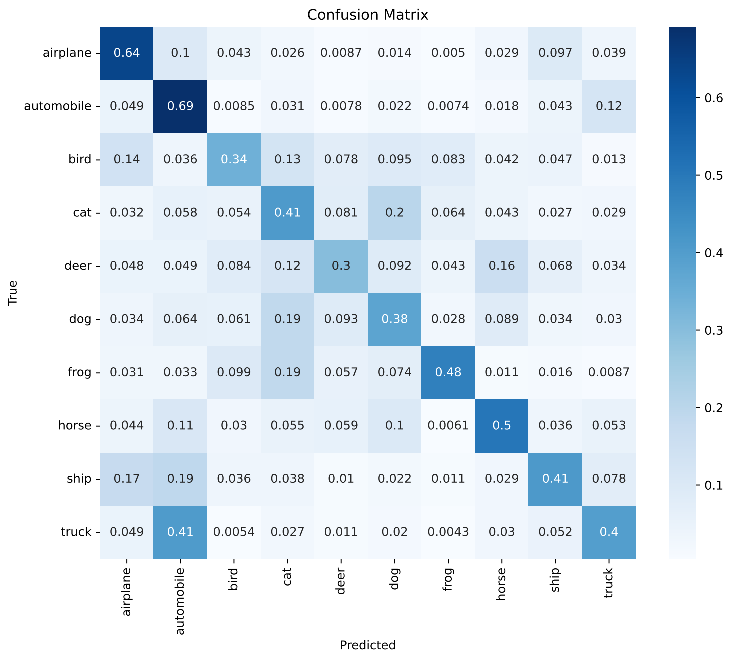

Figure 11: Final Confusion Matrix (EfficientNet)

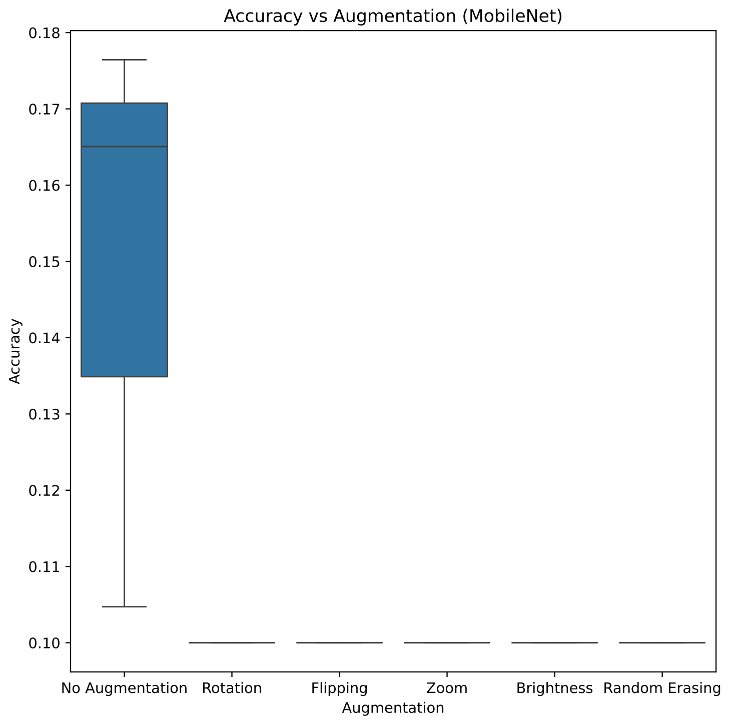

Figure 12: Accuracy vs Augmentation Technique (MobileNet)

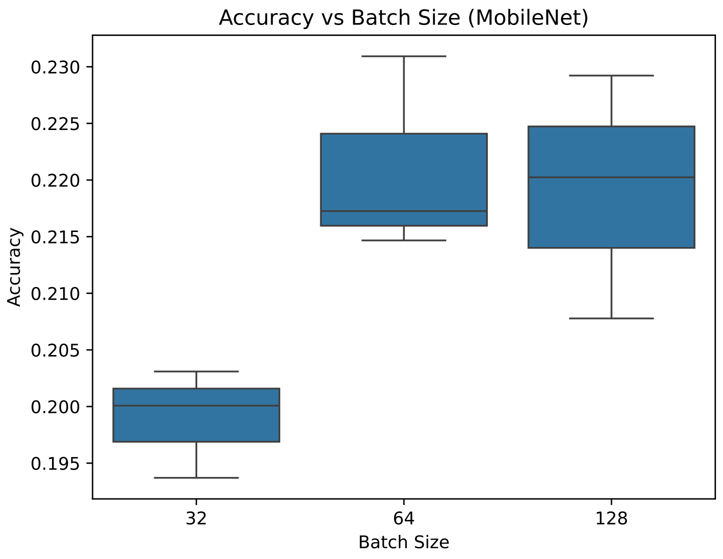

Figure 13: Accuracy vs Batch size (MobileNet)

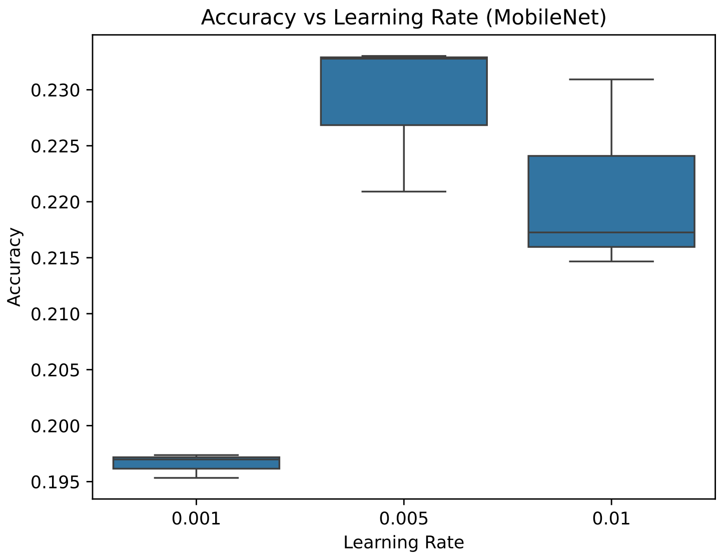

Figure 14: Accuracy vs Learning Rate (MobileNet)

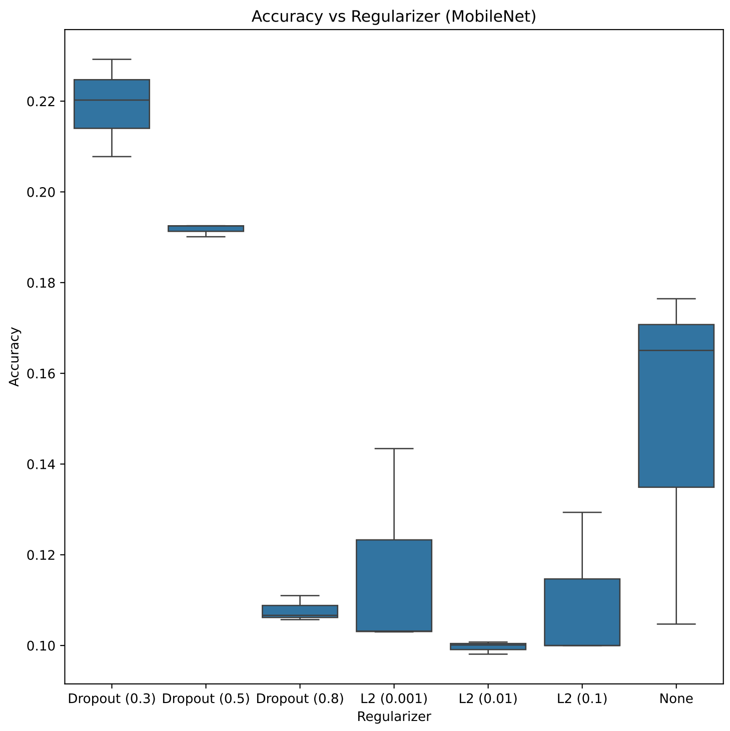

Figure 15: Accuracy vs Regularizer (MobileNet)

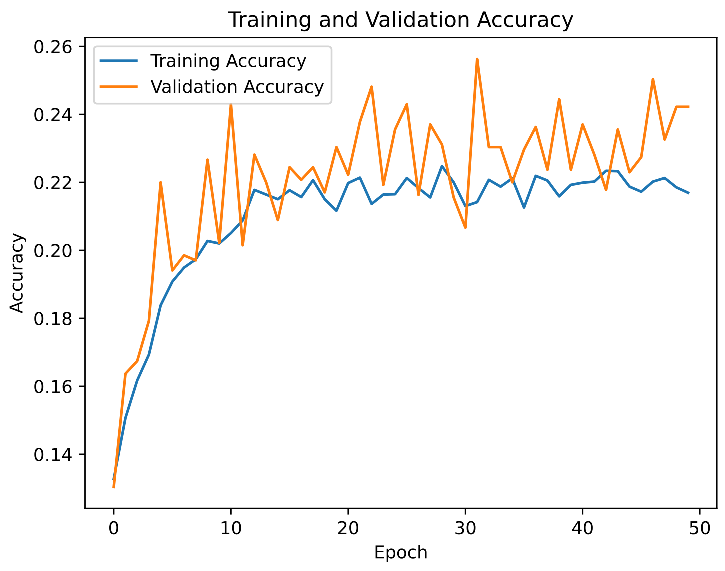

Figure 16: 50 epochs accuracy (MobileNet)

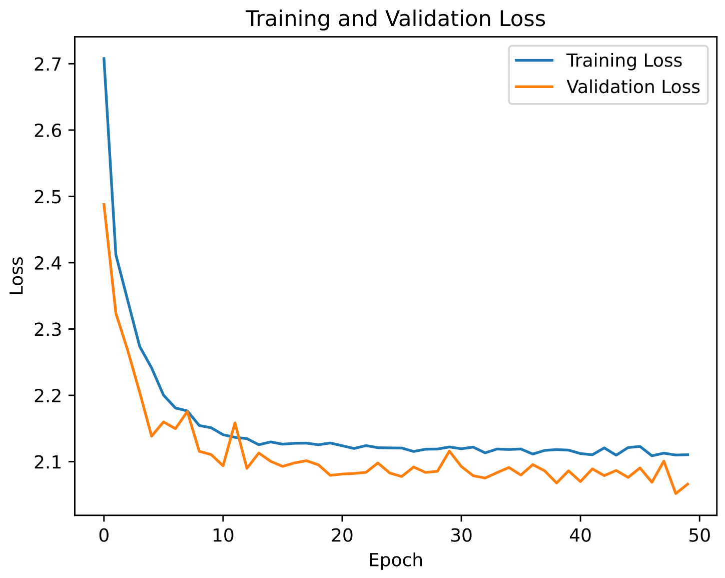

Figure 17: 50 epochs loss (MobileNet)

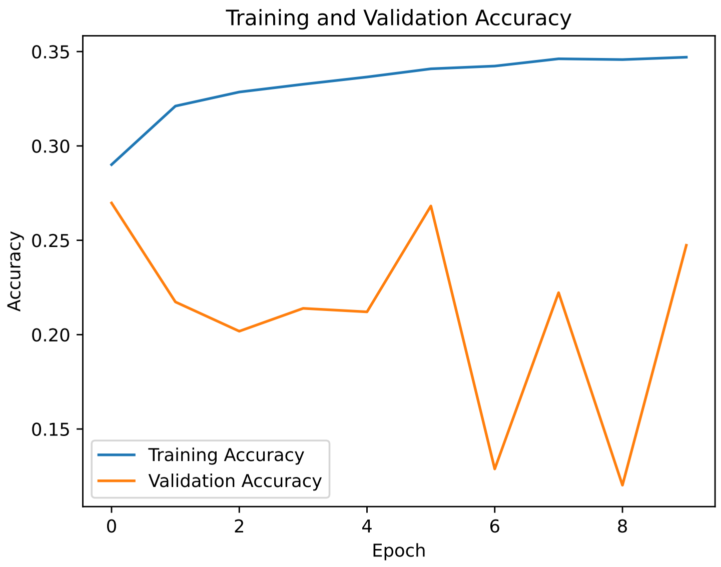

Figure 18: Final Accuracy (MobileNet)

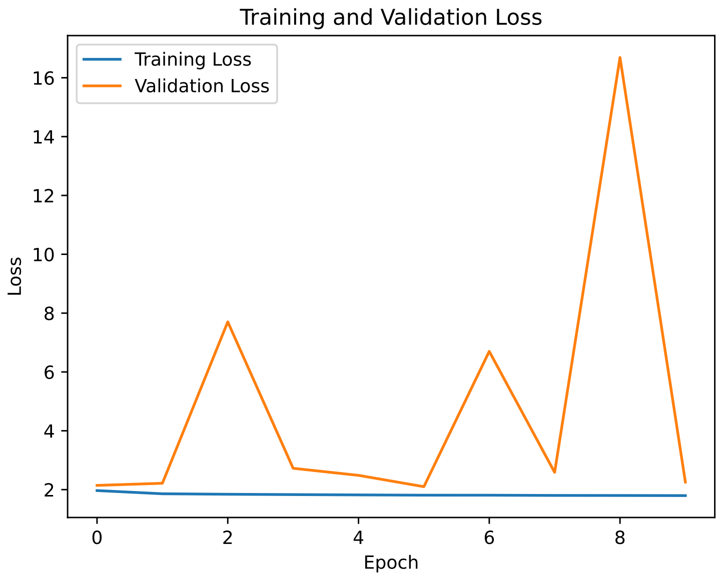

Figure 19: Final Loss (MobileNet)

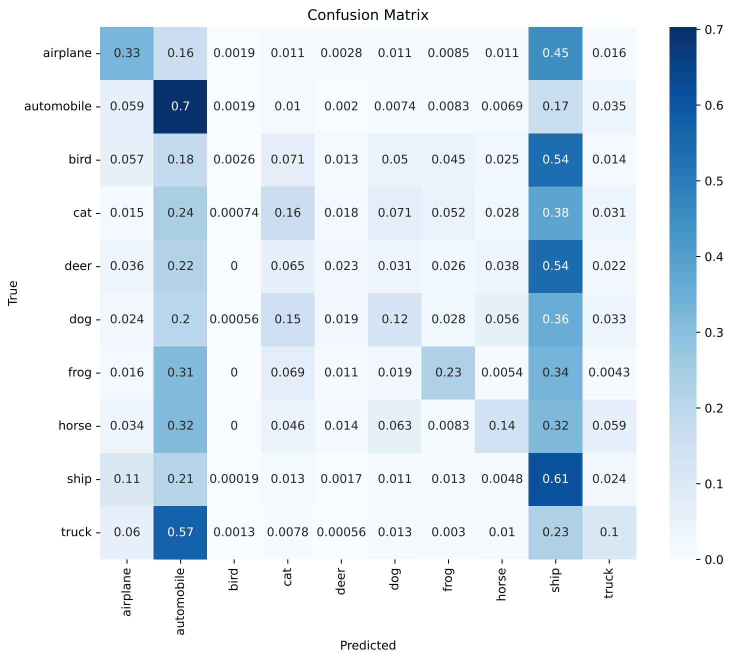

Figure 20: Final Confusion Matrix (MobileNet)

6 Conclusions

We have successfully investigated the effects of the following parameters: batch size, learning rate, number of epochs, L2, Dropout Rate, and data augmentation (rotation, flipping, contrast, brightness change, random erasing). Over 100 experiments have been performed to analyze their impact on the accuracy of CNNs (EfficientNet and MobileNet). Having processed all the data from experiments we have chosen the most promising parameters with which to train and test the final models. We have obtained satisfactory results with EfficientNet with MobileNet lacking behind. However, it's important to keep in mind that we didn't check all of the possible combinations of the parameters due to lack of computation power. It is possible that some of the parameters are correlated and a certain combination of them might result in better accuracy and performance.

List of Figures

References

[1] *CINIC-10 dataset*. URL: https://paperswithcode.com/dataset/cinic-10. (accessed: 26.03.2024).

[2] *EfficientNet*. URL: https://arxiv.org/abs/1905.11946. (accessed: 26.03.2024).

[3] *MobileNet*. URL: https://arxiv.org/abs/1704.04861. (accessed: 26.03.2024).

[4] *CNN*. URL: https://www.ibm.com/topics/convolutional-neural-networks. (accessed: 26.03.2024).

[5] *Classification Performance*. URL: https://c3.ai/introduction-what-is-machine-learning/evaluating-model-performance/. (accessed: 26.03.2024).

[6] *Data Augmentation*. URL: https://aws.amazon.com/what-is/data-augmentation. (accessed: 26.03.2024).

[7] Jason Brownlee PhD. *Difference Between a Batch and an Epoch in a Neural Network*. URL: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. (accessed: 26.03.2024).

[8] Sanket Doshi. *Various Optimization Algorithms For Training Neural Network*. URL: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6. (accessed: 26.03.2024).

[9] *What is regularization?* URL: https://www.ibm.com/topics/regularization. (accessed: 26.03.2024).