# Optimizing Convolutional Neural Networks for CIFAR-10 Image Classification

Giovanni Mantovani and Andrea Palmieri

April 21, 2024

## 1 Introduction

The objective of this lab report is to optimize the performances of Convolutional Neural Networks (CNNs) for image classification using the CIFAR-10 dataset by employing the PyTorch framework. Different architectures have been trained and tested and the best configurations are discussed in this report. Several configurations of hyperparameters have been tested. The hyperparameters are batch size, kernel size, filter depths and learning rate. The training of the network was monitored by plotting the accuracy on the training set versus the one on the validation set, in order to prevent model overfitting.

## 2 Methodology

### 2.1 Dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes mutually exclusive, with 6000 images per class. There are 50000 training images and 10000 test images. The test set contains exactly 1000 randomly-selected images from each class. The training set contains exactly 5000 images from each class. The classes are completely mutually exclusive. For more information about the CIFAR-10 dataset: https://www.cs.toronto.edu/~kriz/cifar.html.

### 2.2 Hyperparameters

The hyperparameters chosen are:

- `Batch size` = 256 images. It refers to the number of input images processed together during training.
- `Learning rate` = 0.001. It determines the sizes of parameter updates, establishing the influence of the computed gradients on the learning process.
- `Loss function` = CrossEntropyLoss. It calculates the loss function values by comparing the predicted probability distribution (model output) with the ground truth labels. Specifically, cross-entropy computes the negative logarithm of predicted probabilities for correct classes, averaged across all samples in the batch.
- `Optimizer` = Adam. Commonly used in CNN training, this optimizer combines adaptive learning rates with momentum for efficient parameter updates.
- `Activation function` = Rectified Linear Units ReLU or Leaky-ReLU. ReLU sets negative inputs to zero, potentially causing dead neurons and leading to sparsity, while Leaky ReLU allows a small but non-zero gradient for negative inputs, preventing complete neuron shutdown.
- `Dropout` = 50% ( only on fully connected layers). A regularization technique commonly used in NN to prevent overfitting. When applied, dropout randomly sets a fraction of input units to zero during each training iteration, effectively "dropping out" these units from the network.
- `Convolution Kernel size` = 3x3.

## 2.3 CNN structure

The chosen CNN structure is shown in 1. All Conv2d layers had a 3x3 kernel size. Convolutional layers are followed by batch normalization and the activation functions. Two convolutions are performed before applying max pooling with size 2 and stride 2.

The CNN structure consists of 21 layers, including convolutional, batch normalization, max pooling, fully connected, and dropout layers. The first layer is a 2D convolutional layer (Conv2d) with a 3x3 kernel, followed by batch normalization (BatchNorm2d). This pattern repeats for subsequent convolutional layers. After every two convolutional layers, max pooling (MaxPool2d) with a 2x2 kernel and stride 2 is applied to downsample the feature maps. The convolutional layers gradually increase the number of output channels, starting from 8 and ending with 128.

After the final convolutional layer, the feature maps are flattened and passed through fully connected (Linear) layers. The first fully connected layer has 512 output units, followed by another fully connected layer with 128 units, and finally a layer with 32 units. Each fully connected layer is followed by a dropout layer with a dropout rate of 50% to prevent overfitting.

The last fully connected layer outputs 10 units, corresponding to the 10 classes in the CIFAR-10 dataset. The following models have been trained and tested:

- `Model 1:` Leaky-ReLU activation functions and dropout set to 50% on fully connected layers.
- `Model 2:` Leaky-ReLU activation functions and dropout set to none on fully connected layers.
- `Model 3:` ReLU activation functions and dropout set to 50% on fully connected layers.
- `Model 4:` ReLU activation functions and dropout set to none on fully connected layers.

| **Order** | **Layer (type)** | **Output Shape** | **Param #** |
|---|---|---|---|
| 1 | Conv2d | [-1, 8, 32, 32] | 224 |
| 2 | BatchNorm2d | [-1, 8, 32, 32] | 16 |
| 3 | Conv2d | [-1, 16, 32, 32] | 1,168 |
| 4 | BatchNorm2d | [-1, 16, 32, 32] | 32 |
| 5 | MaxPool2d | [-1, 16, 16, 16] | 0 |
| 6 | Conv2d | [-1, 32, 16, 16] | 4,640 |
| 7 | BatchNorm2d | [-1, 32, 16, 16] | 64 |
| 8 | Conv2d | [-1, 64, 16, 16] | 18,496 |
| 9 | BatchNorm2d | [-1, 64, 16, 16] | 128 |
| 10 | MaxPool2d | [-1, 64, 8, 8] | 0 |
| 11 | Conv2d | [-1, 128, 8, 8] | 73,856 |
| 12 | BatchNorm2d | [-1, 128, 8, 8] | 256 |
| 13 | Conv2d | [-1, 128, 8, 8] | 147,584 |
| 14 | BatchNorm2d | [-1, 128, 8, 8] | 256 |
| 15 | MaxPool2d | [-1, 128, 4, 4] | 0 |
| 16 | Linear | [-1, 512] | 1,049,088 |
| 17 | Dropout 50% | [-1, 512] | 0 |
| 18 | Linear | [-1, 128] | 65,664 |
| 19 | Dropout 50% | [-1, 128] | 0 |
| 20 | Linear | [-1, 32] | 4,128 |
| 21 | Linear | [-1, 10] | 330 |

Table 1: CNN structure

## 2.4 Training

The neural network was trained on the training set using the CrossEntropyLoss function and the Adam optimizer. Learning rate was set to 0.001 and maximum number of epochs to 25. During each epoch, the network's weights were adjusted based on the computed gradients, and the optimizer updated the weights to minimize the loss.

After adjusting the weights in each epoch, the network was switched to evaluation mode to compute accuracy on the test set. In evaluation mode, no weight adjustments were made, ensuring that the

accuracy calculation was based only on the current state of the network's parameters. Once the accuracy on the test set was computed, the network was switched back to training mode to continue with the next epoch.

Throughout the training process, various metrics such as training loss and training versus test accuracy were logged using TensorBoard. This allowed monitoring of the training progress and enabled the detection of potential overfitting.

The training loop was designed to terminate under two conditions: either when the maximum number of epochs was reached (25 epochs), or when early stopping conditions were met. Early stopping was triggered if the accuracy on the training set exceeded the accuracy on the test set by more than 10% or if no improvement was observed after 10 consecutive epochs.

# 3 Results

In this section, we present the results of our experiments on different models for image classification. Table 2 displays the accuracy for each class achieved by four models (Model 1, Model 2, Model 3, and Model 4) and the total accuracy for each model.

Additionally, confusion matrices for each model are presented in Figures 1, 2, 3, and 4.

| **Class** | **Model 1** | **Model 2** | **Model 3** | **Model 4** |
|---|---|---|---|---|
| Plane | 83.30% | 88.30% | 73.30% | 79.30% |
| Car | 89.80% | 92.30% | 70.10% | 82.90% |
| Bird | 62.60% | 49.80% | 63.60% | 72.80% |
| Cat | 41.50% | 52.90% | 51.60% | 67.50% |
| Deer | 68.80% | 74.60% | 75.60% | 67.00% |
| Dog | 88.80% | 72.90% | 72.10% | 76.60% |
| Frog | 82.20% | 78.00% | 71.30% | 84.60% |
| Horse | 85.40% | 87.70% | 90.30% | 78.00% |
| Ship | 81.60% | 69.80% | 89.50% | 88.40% |
| Truck | 86.20% | 90.20% | 96.20% | 91.20% |
| **Total** | **77.65%** | **75.22%** | **75.36%** | **78.83%** |

Table 2: Accuracy for each class across all models

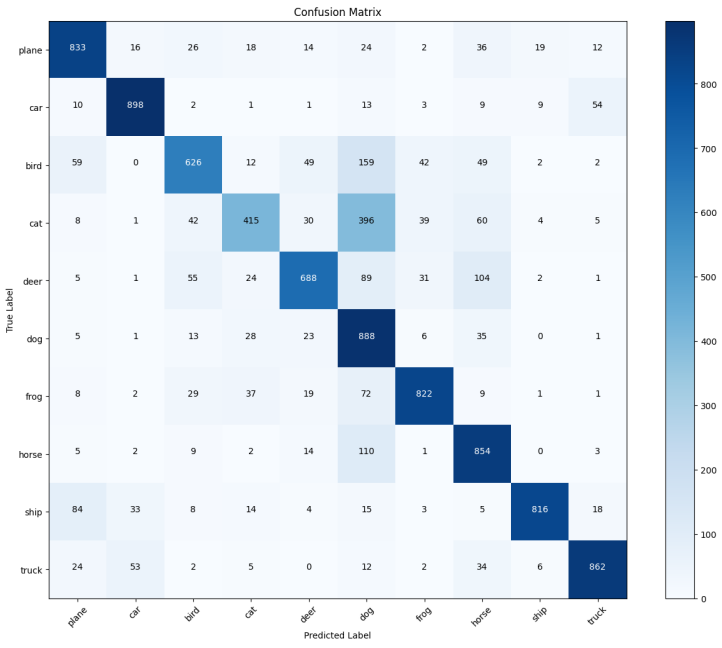


Figure 1: Confusion Matrix of Model 1: Leaky-ReLU activation functions and dropout set to 50% on fully connected layers.

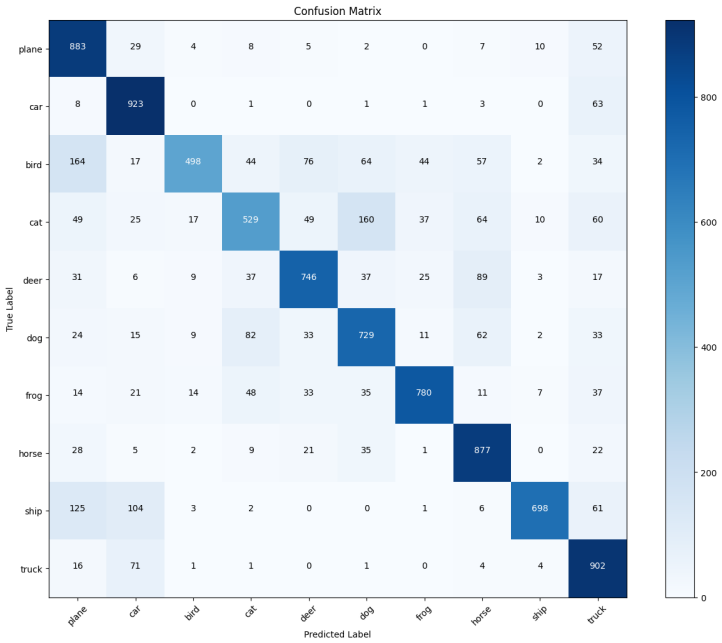


Figure 2: Confusion Matrix of Model 2: Leaky-ReLU activation functions and dropout set to none on fully connected layers.

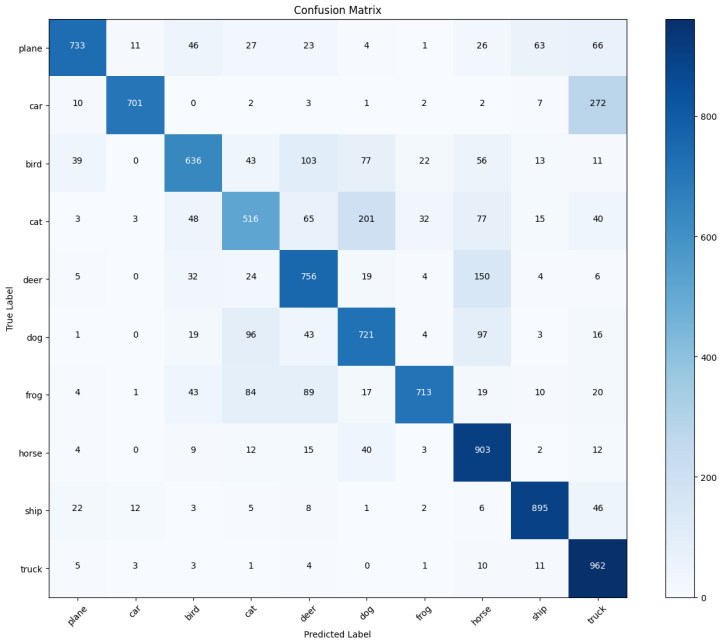


Figure 3: Confusion Matrix of Model 3: ReLU activation functions and dropout set to 50% on fully connected layers.

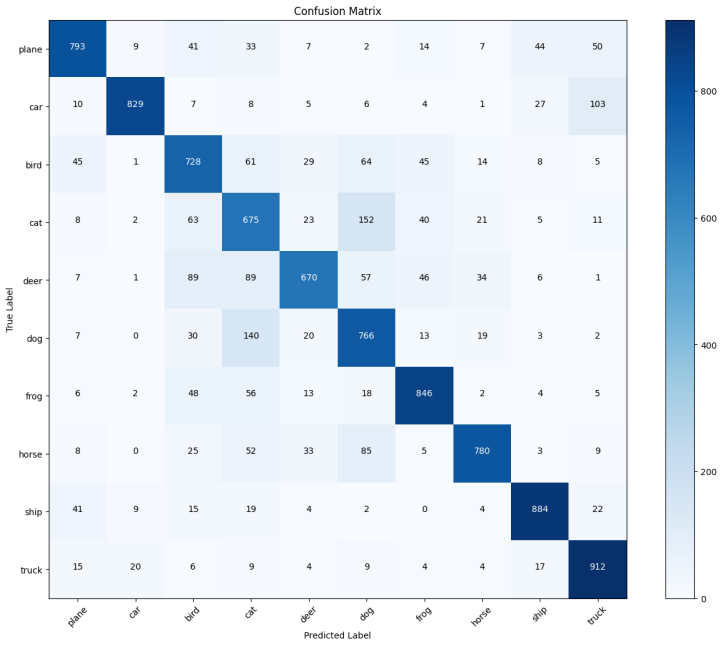


Figure 4: Confusion Matrix of Model 4: ReLU activation functions and dropout set to none on fully connected layers.

# 4 Conclusions

The findings show that accuracy varies considerably across different classes for all models. Some classes, such as Car and Truck, achieved high accuracies across all models, possibly indicating that they are relatively easy to classify. In contrast, classes like Bird and Cat exhibited lower accuracies, suggesting that these classes may be more challenging for the models to distinguish accurately.

This discrepancy in performance could be attributed to the visual similarities in shape and colors between cats and dogs in images of size 32x32. Notably, all confusion matrices show a significant misclassification rate of cats as dogs, ranging from 15% to 40% across all models.

Additionally, other evident misclassifications are evident in the models, such as cars being mistaken for trucks (in models 3 and 4) and birds being misclassified as planes (in model 2).

The comparison of neural network models suggests that dropout has a differential impact on the performance of activation functions. Specifically, while dropout positively influenced the performance of Leaky-ReLU (77.65% with dropout versus 75.22% without dropout), its effect was instead negative on ReLU (75.36% with dropout versus 78.83% without).

Overall, the models' final accuracies on the test set range from 75.2% to 78.8%. While this suggests that the network structure is capable of achieving discrete classification performance, further fine-tuning and possibly re-engineering are necessary to enhance its accuracy.