A Study of Deep Convolutional Models for Computer Vision Tasks on CINIC-10 dataset

Authors: Paulina Kulczyk, Jan Pogłód

Warsaw University of Technology
Department of Mathematics and Information Science

April 3, 2025

Abstract

Convolutional Neural Networks (CNNs) have become increasingly popular in the field of deep learning due to their broad applications in image recognition. They are widely utilized not only in scientific research but also across various industries and academic settings. Typically, improving performance involves designing deeper and more complex architectures. Despite their widespread adoption, many aspects of CNNs remain challenging to understand due to their intricate mathematical foundations and the vast range of possible hyperparameter configurations and architectural structures. In this study, we aim to identify optimal architectures by systematically comparing various hyperparameter settings and data augmentation techniques. Our findings provide insights into the most effective configurations, contributing to a deeper understanding of CNN optimization.

Keywords: image recognition, Convolutional Neural Networks, deep learning, CINIC-10 dataset

Contents

1 Introduction

Convolutional Neural Networks (CNNs) were first introduced by LeCun et al. (1998) [1] for handwritten digit classification. Their design was inspired by the discovery of locally-sensitive, orientation-selective neurons in the visual cortex of cats (Hubel & Wiesel, 1962) [2]. It illustrates that CNNs, like many scientific advancements, are influenced by natural processes. Despite their initial introduction, CNNs remained largely unexplored for several years due to the challenges associated with deep learning at the time. However, interest in CNNs resurged in the early 21st century, particularly following the success of Krizhevsky et al. (2012) [3] in the ImageNet Large Scale Visual Recognition Challenge (LSVRC-2010), where CNNs significantly outperformed traditional machine learning models.

After that, CNNs quickly gained popularity and became the dominant approach in computer vision tasks, consistently achieving state-of-the-art results in various competitions [4, 5]. Researchers primarily focused on deepening the network (in terms of the number of layers) and broadening it (in terms of the number of filters in each layer) [6, 7]. While CNNs have demonstrated remarkable performance in classification tasks, comprehensive comparisons of their architectures remain insufficient. Most studies emphasize benchmarking specific CNN architectures rather than systematically analyzing the impact of hyperparameter tuning. Additionally, many studies overlook the influence of data augmentation techniques employed during training.

In this work, we investigate the application of CNNs for image classification using the **CINIC-10 dataset** [8, 9]. The objective of our study is to evaluate different CNNs architectures and assess the impact of hyperparameter variations, data augmentation techniques, and few-shot learning methods on classification accuracy. Our findings aim to contribute to a deeper understanding of CNNs optimization strategies and their practical implications. In addition, we propose here a cutting-edge approach - ensemble learning - regarding deep learning (image recognition).

1.1 Research Problem Description

This report presents research on the application of CNNs for image classification using the CINIC-10 dataset. The aim of the experiments was to compare various architectures and examine the impact of hyperparameter changes, data augmentation techniques, and few-shot learning methods on classification accuracy.

1.2 Objectives and Assumptions of the Experiment

The objectives of the experiments were:

- To investigate various CNNs architectures.
- To analyze the impact of hyperparameters and regularization techniques.
- To apply data augmentation techniques.
- To explore few-shot learning methods.
- To reduce the training dataset and analyze the outcomes.
- To implement ensemble learning methods.

2 Literature Review

2.1 Theory of Convolutional Neural Networks

In the past few years, several CNNs architectures have been proposed for image recognition, each improving on the previous ones in terms of performance and efficiency. One of the most well-known is VGGNet (2014), introduced by the Visual Geometry Group at Oxford [10]. This architecture uses small 3x3 convolutional filters in a deep network (16-19 layers), providing excellent feature extraction but at a high computational cost. Another prominent model, GoogLeNet (2014), introduced by Christian Szegedy and his team at Google [7], uses fewer parameters compared to VGG while achieving similar accuracy. At that time, these networks were considered deep, but this perception changed with the introduction of ResNet (2016) [11], which enabled the training of networks with 50, 100, or even more layers. This was further extended by DenseNet (2017), which pushed network depth to over 200 layers.

In terms of hyperparameter optimization, Smith et al. explored the impact of learning rate schedules on training efficiency [12]. They found that cyclical learning rates can accelerate convergence and enhance model accuracy. Research has also examined the effect of batch size on both convergence speed and generalization performance. In terms of optimization, many studies agree that the Adam optimizer [13–15] is preferred for CNNs due to its adaptive learning rate, which allows for faster convergence and better handling of sparse gradients. However, some studies [16, 17] suggest that for very large datasets, SGD with momentum may sometimes be more efficient.

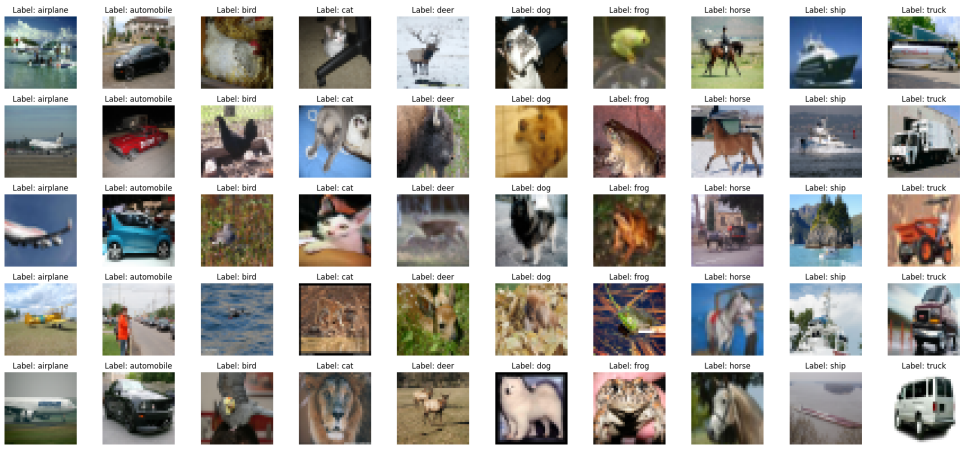
Data augmentation, although a fundamental technique, has received limited focused research in the context of CNNs training. A key study by Perez et al. [18] examined various augmentation methods and their effectiveness, particularly in preventing overfitting. In their experiments, they trained networks for 40 epochs with a learning rate of 0.0001 using Adam optimization, finding that data augmentation significantly improved model generalization.

2.2 Research on the CINIC-10 Dataset

The CINIC-10 dataset is an extension of CIFAR-10, containing 270,000 images equally divided into training, validation, and test sets (90,000 each) across 10 classes [19]. Each image is a 32x32 RGB image stored in PNG format, with pixel values ranging from 0 to 255. The data is organized into subdirectories representing class labels, allowing for straightforward integration with deep learning frameworks.

CINIC-10 increases diversity compared to CIFAR-10 by incorporating images from ImageNet, making it a more challenging benchmark due to greater variability and noise in visual features.

In our research, we focused on evaluating "Deep CNN" architectures on this complex dataset. We experimented with both basic and advanced networks, including residual models, to improve accuracy and robustness. Techniques like data augmentation, adaptive optimizers (AdamW), and learning rate scheduling were applied to enhance training stability and generalization.



3 Methodology

In this section, we present the methodology used in our experiments. We describe the network architectures, data preprocessing methods, training strategies, and evaluation metrics applied to the CINIC-10 dataset.

3.1 Neural Network Architectures

To investigate the performance of CNNs on the CINIC-10 dataset, we designed several architectures with increasing diversity. The architectures can be broadly classified into two categories: basic architectures and advanced architecture.

3.1.1 Basic CNNs Architectures

The initial stage of testing the CINIC-10 dataset involved splitting it into smaller subsets to facilitate efficient experimentation. Given the size of the complete training dataset (90,000 images), we first

reduced it to 2,000 images to quickly evaluate our initial model setups. This preliminary testing allowed us to identify potential issues and gain insights without investing excessive computational resources. After confirming the feasibility of our approach, we proceeded to test on a larger subset consisting of 9,000 images, while still holding the complete dataset as a final benchmark.

This progressive approach to dataset utilization enabled us to make rapid iterations on model architectures and training strategies before committing to large-scale experiments. It also helped us understand the behavior of simpler models before applying more advanced architectures.

For our initial experiments, we implemented two basic CNNs architectures to serve as benchmarks. These architectures were deliberately kept simple to gauge the model performance on smaller subsets and serve as baselines for more complex networks. Below are the two basic CNNs architectures we tested:

- **"Single Convolution Layer CNN"**: A simple architecture with one convolutional layer, intended to quickly test baseline accuracy and overfitting tendencies.
- **"Double Convolution Layer CNN"**: An extended version with two convolutional layers, aimed at capturing more complex spatial features while maintaining computational simplicity.

Table 1: "Single Convolution Layer CNN" Architecture

Layer Type	**Parameters**	**Output Shape**
Input	-	(32, 32, 3)
Conv2D	32 filters (3x3), ReLU	(32, 32, 32)
MaxPooling2D	(2, 2) pool size	(16, 16, 64)
Flatten	-	$(16 \cdot 16 \cdot 32)$
Dense	128 neurons, ReLU	(128)
Dense	10 neurons, softmax	(10)

Table 2: "Double Convolution Layer CNN" Architecture

Layer	**Parameters**	**Output Shape**
Input	-	(32, 32, 3)
Conv2D	32 filters (3×3), ReLU, same	(32, 32, 32)
MaxPooling2D	(2, 2) pool size	(16, 16, 32)
Conv2D	64 filters (3×3), ReLU, same	(16, 16, 64)
MaxPooling2D	(2, 2) pool size	(8, 8, 64)
Flatten	-	(4096)
Dense	128 neurons, ReLU	(128)
Dense	10 neurons, softmax	(10)

Both architectures were trained on the smaller subsets with batch size equals 32 to assess their ability to learn basic patterns from the CINIC-10 images. The single convolution model offered a rapid way to check the fundamental learnability of the data, while the double convolution model provided more capacity to capture spatial hierarchies.

These simple architectures gave us insights into the baseline accuracy and training dynamics, guiding further exploration with more advanced and deeper architectures discussed later in this report.

As we can see in Figure 1 and Figure 2, simple architectures and small datasets are insufficient for tackling this problem effectively. The limited capacity of these models leads to significant misclassifications and inadequate generalization. Therefore, we introduced a more advanced architecture and moved towards using the full data setup of 90,000 training images, 90,000 validation images, and 90,000 test images. As an optimizer, we used Adam as the fittest for CNNs [20] with a learning rate of 0.001 on a batch size of 32.

To enhance training robustness and prevent overfitting, we further extended the training set by employing advanced data augmentation techniques. This allowed us to maintain a consistent 8/1/1 ratio between training, validation, and testing data. By doing so, we aimed to significantly improve generalization and model performance on the entire dataset.

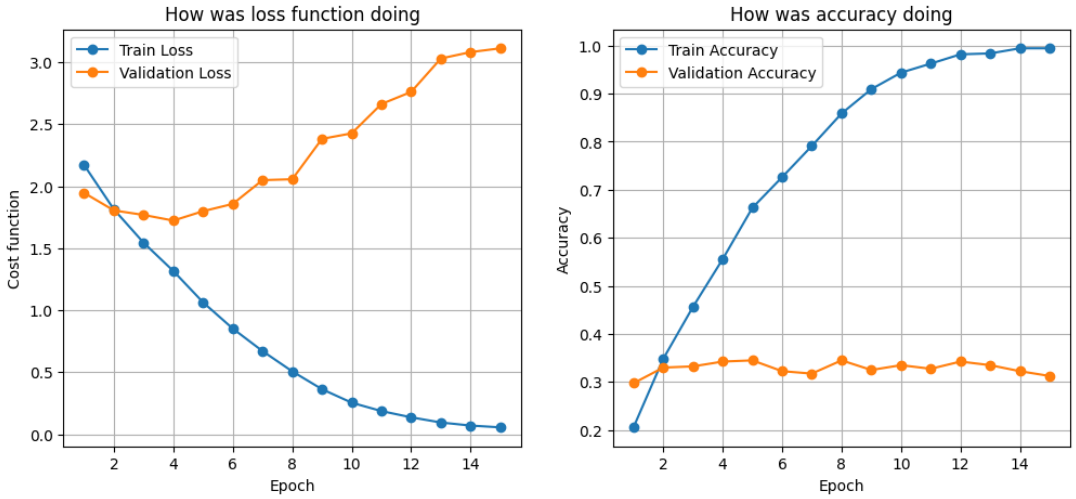


Figure 1: Learning process on the "Single Convolution Layer CNN" and 2000 images in the training

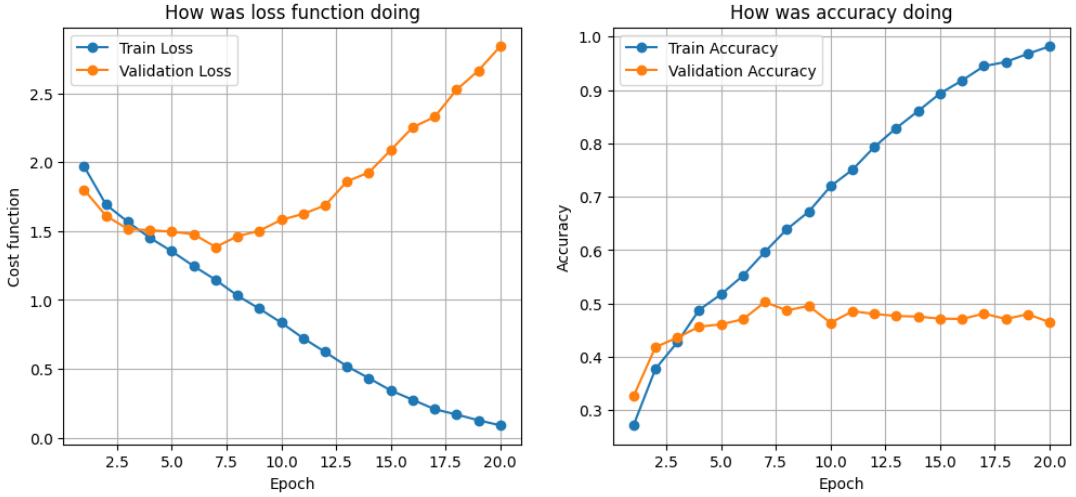


Figure 2: Learning process on the "Double Convolution Layer CNN" and 9000 images in the training

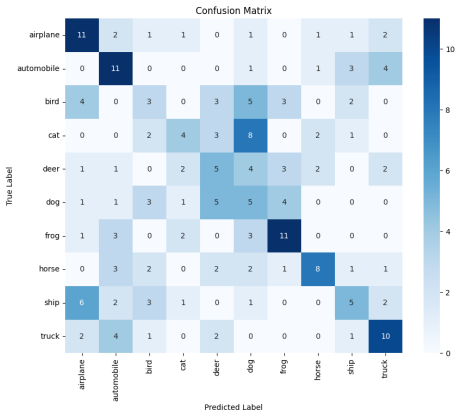


Figure 3: Confusion matrix for the "Single Convolution Layer CNN" and 2000 images in the training sample

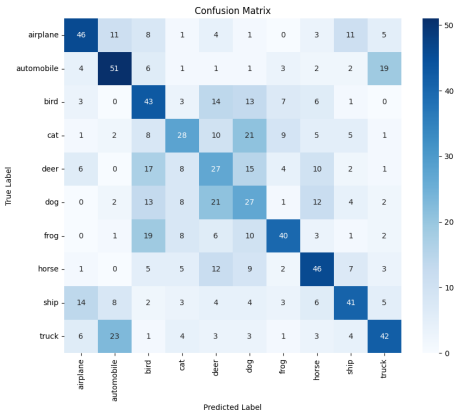


Figure 4: Confusion matrix for the "Double Convolution Layer CNN" and 9000 images in the training sample

Figure 5: Confusion matrices for simple CNNs with small training datasets.

3.1.2 Advanced CNN Architecture

For our advanced architecture, we implemented a deeper network with additional regularization techniques:

Table 3: "Deep CNN" Architecture

Layer Type	**Parameters**	**Output Shape**
Input	-	(32, 32, 3)
Conv2D	32 filters (3x3), ReLU	(32, 32, 32)
BatchNormalization	-	(32, 32, 32)
MaxPooling2D	(2, 2) pool size	(16, 16, 32)
Dropout	0.25 rate	(16, 16, 32)
Conv2D	64 filters (3x3), ReLU	(16, 16, 64)
BatchNormalization	-	(16, 16, 64)
MaxPooling2D	(2, 2) pool size	(8, 8, 64)
Dropout	0.25 rate	(8, 8, 64)
Conv2D	128 filters (3x3), ReLU	(8, 8, 128)
BatchNormalization	-	(8, 8, 128)
MaxPooling2D	(2, 2) pool size	(4, 4, 128)
Dropout	0.25 rate	(4, 4, 128)
Flatten	-	$(4 \cdot 4 \cdot 128)$
Dense	512 neurons, ReLU	(512)
BatchNormalization	-	(512)
Dropout	0.5 rate	(512)
Dense	256 neurons, ReLU	(256)
BatchNormalization	-	(256)
Dropout	0.5 rate	(256)
Dense	10 neurons, softmax	(10)

The advanced architecture incorporates several improvements over the basic models:

- Three convolutional blocks, instead of one or two, with progressive increase in filter sizes.
- Batch normalization after each convolutional and dense layer.
- Dropout layers for regularization.

Additionally, we examined the weight distributions for both weak and strong architectures on the full CINIC-10 dataset (90k/90k/90k) to verify the stability and quality of our "Deep CNN" model. The results confirmed that the new, deeper CNNs architecture is well-trained, as the weight distributions exhibit normal-like patterns, which are characteristic of robust and well-generalized models. In contrast, the weaker architectures displayed irregular and non-bell-shaped weight distributions, indicating inadequate learning and poor generalization. This further justifies the decision to use a deeper and more complex architecture for the CINIC-10 classification problem.

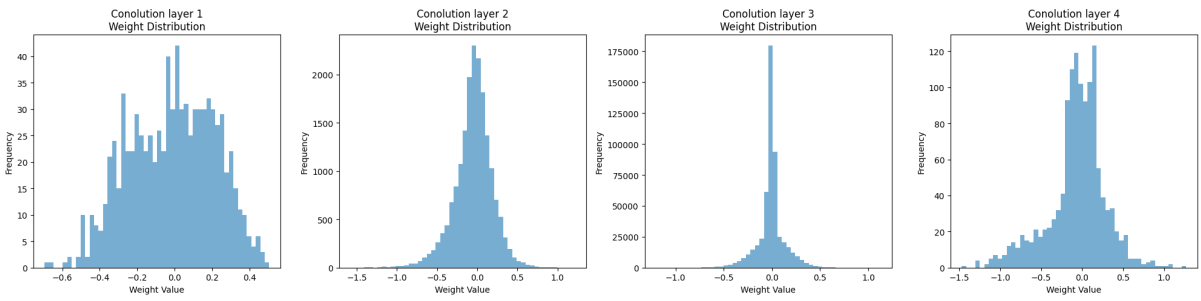


Figure 6: Weight distribution for "Double Convolution Layer CNN" Architecture

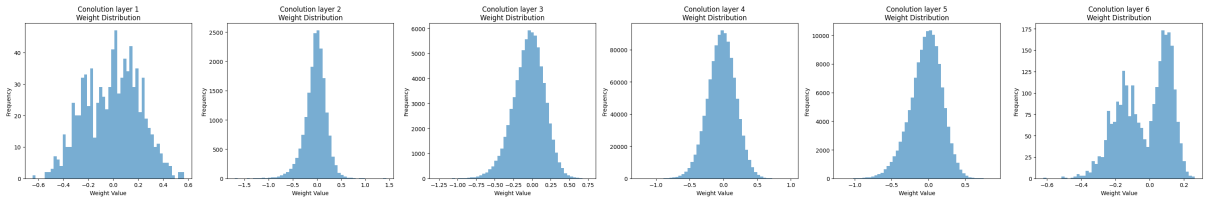


Figure 7: Weight distribution for "Deep CNN" for each convolutional layer

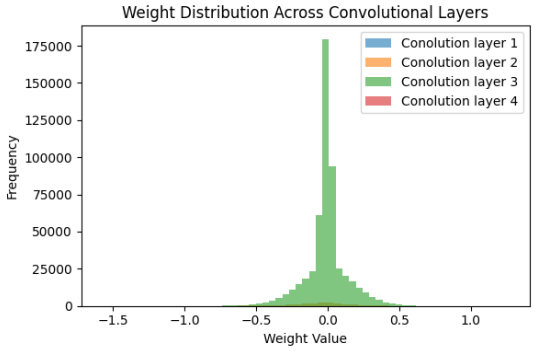


Figure 8: Poorly trained model with double convolution layer

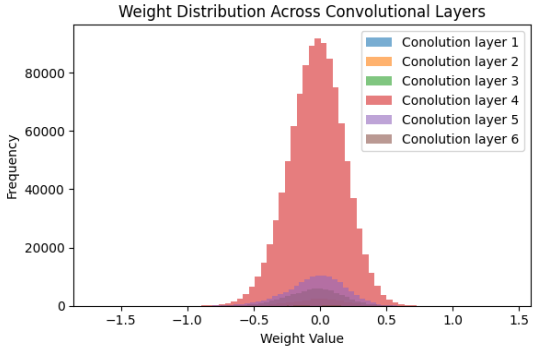


Figure 9: Well trained model with "Deep CNN" architecture

Figure 10: Comparison of weight distribution over all layers

In order to test the effectiveness of new architecture and to assess the impact of increased training data we conducted two experiments training our "Deep CNN" on the full CINIC-10 dataset:

- "Deep CNN" on the full dataset (90,000 training images, 90,000 validation images, 90,000 test images)
- "Deep CNN" dataset with proportion: 144,000 training images, 36,000 validation images, 90,000 test images.

3.2 Performance Analysis of the "Deep CNN"

Both experiments used a batch size of 32 to ensure stable weight updates. We utilized the Adam optimizer with an exponentially decaying learning rate schedule. The initial learning rate was set to 0.001, with a decay rate of 0.9 applied every 10,000 steps as to improve learning process [20]. We aimed to see how increasing data size affect generalization performance.

- *Overfitting Control:* The small gap between training and validation accuracy suggests much better regularization and generalization in comparison to previous models.
- *Accuracy:* Training accuracy improved from 40% to over 72% in 20 epochs, with validation accuracy peaking at around 73%. This indicates good generalization with no major signs of overfitting.
- *Convergence:* The model showed smoother convergence, with loss decreasing from around 1.8 to below 0.9, demonstrating effective error minimization.
- *Stability:* The model maintained stable training without oscillations, even with a small batch size of 32.
- *Test Accuracy:* The final test accuracy reached much better approximately 70.4%, showing robust performance on unseen data

The results on the test data (90k images) for both experiments are in the Table 4. It can be noted that F1-score is higher in the (144k/36k) proportion for each class.

Overall, increasing the training data size to 144k significantly boosted the model's generalization ability. We will move forward with data augmentation techniques to make the training set much more diversity and bigger in purpose to get even better results.

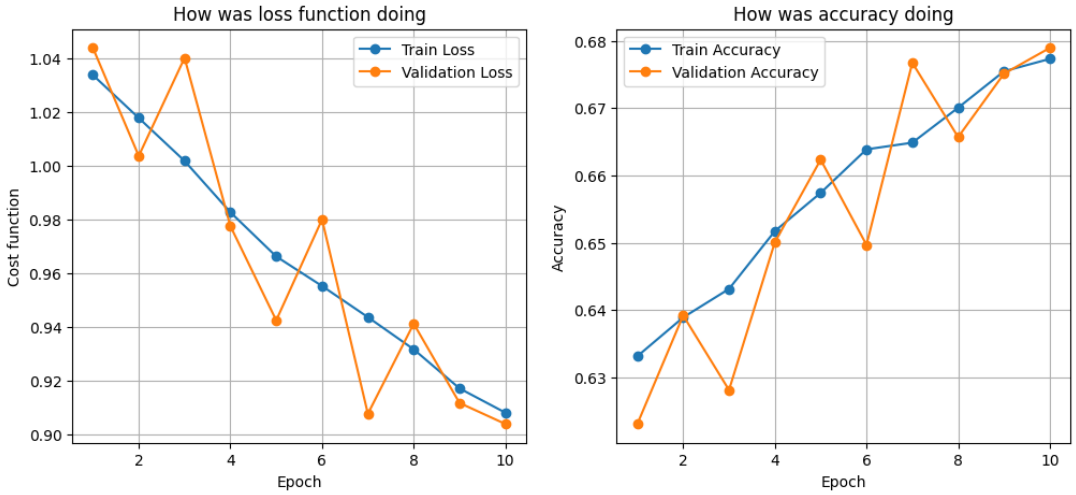


Figure 11: Learning process - "Deep CNN" on the full dataset (90k/90k/90k)

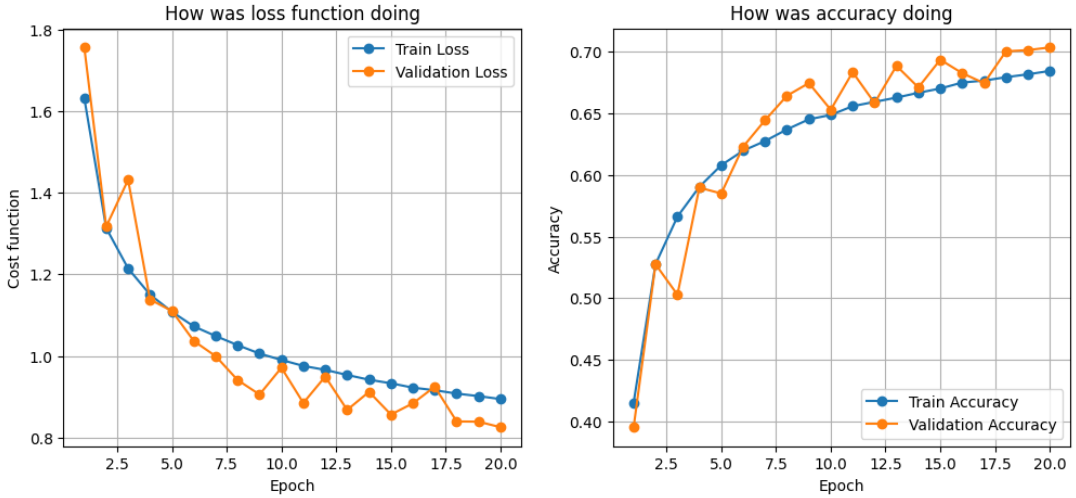


Figure 12: Learning process - "Deep CNN" on the dataset (144k/36k/90k)

Table 4: Comparison of Classification Results for "Deep CNN" on Two Dataset Splits (90 000 images / 90 000 images / 90 000 images) vs (144 000 images / 36 000 images / 90 000 images)

Class	**90k/90k/90k**			**144k/36k/90k**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Airplane	0.86	0.71	0.78	0.72	0.84	**0.78**
Automobile	0.74	0.78	0.76	0.78	0.77	**0.77**
Bird	0.67	0.58	0.62	0.63	0.69	**0.66**
Cat	0.49	0.57	0.53	0.63	0.47	**0.54**
Deer	0.63	0.58	0.61	0.63	0.63	**0.63**
Dog	0.54	0.51	0.52	0.64	0.45	**0.53**
Frog	0.61	0.89	0.72	0.70	0.86	**0.77**
Horse	0.80	0.70	0.75	0.74	0.79	**0.76**
Ship	0.72	0.81	0.77	0.78	0.79	**0.78**
Truck	0.82	0.64	0.72	0.74	0.76	**0.75**
Accuracy	-	-	**0.68**	-	-	**0.70**

Figure 13: Confusion matrix for "Deep CNN" dataset (90k/90k/90k)

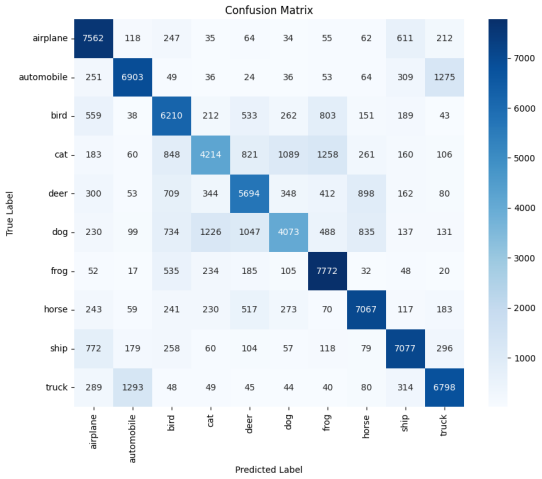


Figure 14: Confusion matrix for "Deep CNN" dataset (144k/36k/90k)

Figure 15: Confusion matrices for "Deep CNN" with different training datasets splits

4 Data Augmentation Techniques

Data augmentation is a technique for expanding a training set by applying random transformations to existing samples, producing new images that look plausible (that is, they represent an object that matches the label originally assigned to them). The aim of this approach is to ensure that the model never encounters the exact same image twice during training [21]. By exposing the model to multiple variations of each sample, data augmentation allows it to learn more diverse features, improving generalization and reducing the risk of overfitting.

Furthermore, as demonstrated in our experiments, data augmentation can sometimes help eliminate characteristics that are irrelevant for classification and could otherwise mislead the model, leading to erroneous predictions. By carefully applying transformations, data augmentation enhances the model's robustness and overall performance.

During our experiments we consider the following techniques: **rotation, stretching, flipping, cropping, changing colours** (black-and-white and sepia images), **sobel filtering** and **adding gaussian noise** (see Figure 16).

Rotation simulates object orientation changes, making the model more robust to varying angles. Stretching alters the image size to simulate objects at different scales, while flipping mirrors the image to account for different object orientations. Cropping helps simulate partial occlusions or zooming, encouraging the model to focus on diverse regions of the image.

Color changes, including converting an image to black-and-white or applying a sepia filter, modify the color properties of the image. Converting to black-and-white is particularly useful when color is not a critical feature for classification, such as in medical or grayscale image tasks. The sepia filter, on the other hand, simulates older photographs and can aid in the model's ability to generalize across different visual conditions or media types. Sobel filtering emphasizes edges, aiding in tasks requiring boundary detection. Finally, Gaussian noise is added to introduce imperfections and simulate real-world sensor noise, improving the model's robustness.

4.1 Experiments

The selection of hyperparameters in CNNs presents a significant challenge due to the numerous available options. Additionally, we investigate various architectures, including "Deep CNN", VGG, GoogLeNet, and ResNet. As we mention earlier, we also focused on data augmentation techniques and they influence on models predictions. During our study we examine the effects of different dataset sizes and its division to train validation and test sets.

In below sections we present our experiments and their results.

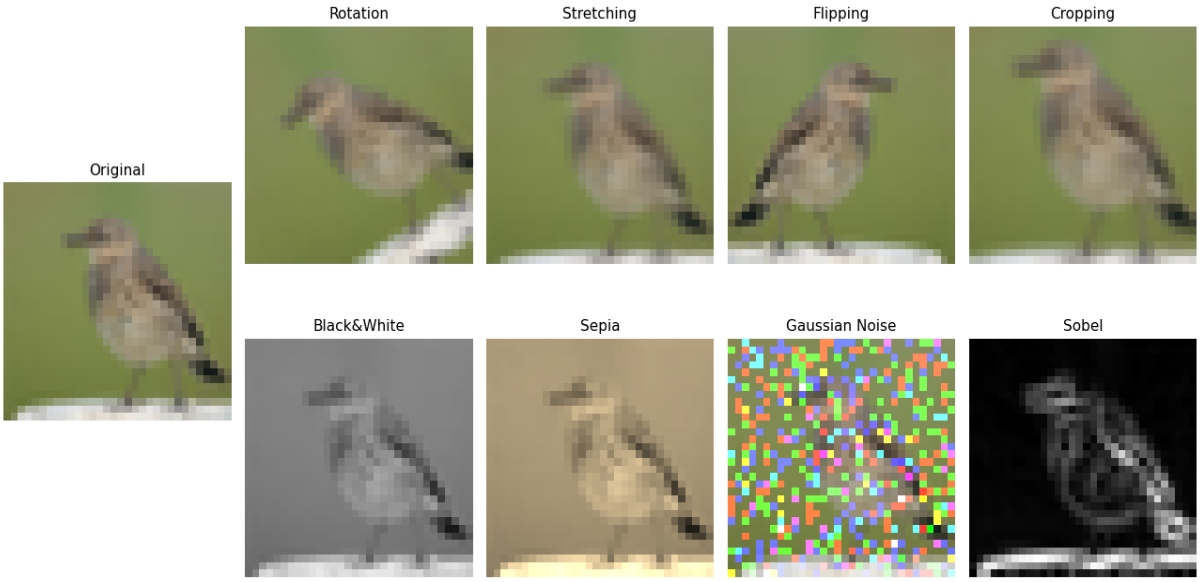


Figure 16: Figure present the data augmentation techniques that we apply during our research. Presented photo comes from CINIC-10 dataset.

4.2 Color changing

During training the first models we seen that many 'forest' animal or animal that can be seen among grass and leaves are badly predicted, and many times labeled as a frog (see 17).

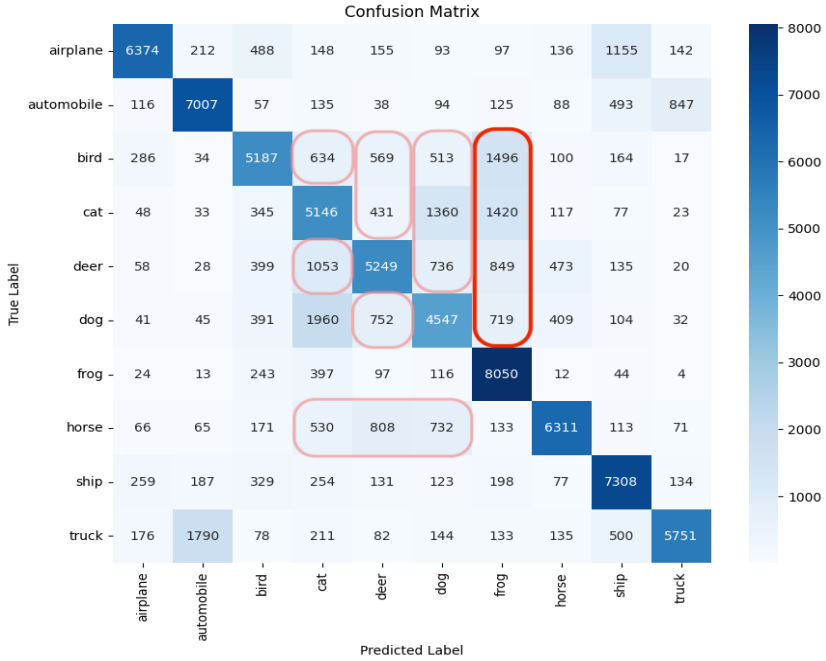


Figure 17: Confusion matrix that presents prediction of "Deep CNN" model trained on orginal dataset. With red shaded circles we mark the wrongly predicted animals which we can meet in green background. With intensive red color we mark wringly predicted animals as frog. This class is special, as the body of the frog is predominantly green and as it is found in a green environment, the algorithm may wrongly predict images with a large amount of green as frogs without looking at other characteristics such as shape.

We claimed that the reason for such behavior of our models is misdiagnosed pattern amomg data. Most images with frogs have a lot of green hue. But color (and even more the background color) shouldn't

suggest the predicted label. To test our hypothezis we decide to compare "Deep CNN" model learned on orginal dataset to "Deep CNN" moodels trained on orginal data set and additionaly on set with changed colour: sepia, white and black and mixed.

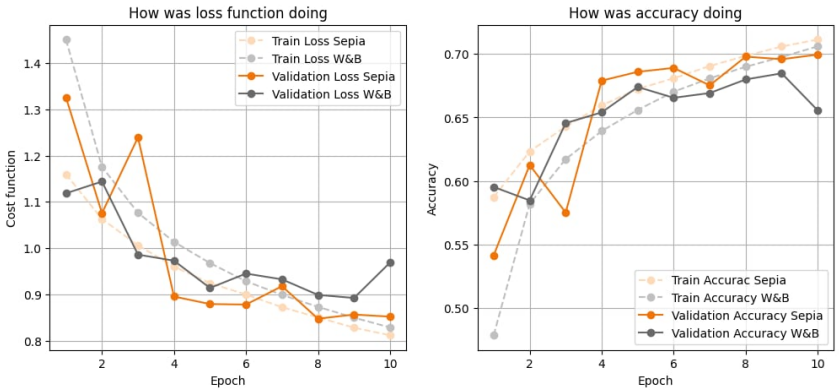


Figure 18: Learning process of "Deep CNN" architecture with additional color changed photos on training dataset.

Figure 18 illustrates the learning process of models with applied color augmentation techniques. It is evident that the sepia technique performs slightly better in image recognition compared to the black-and-white technique. After 10 epochs, we observe satisfactory results at that time, prompting us to finalize these settings and proceed with testing on the test dataset. Furthermore, we constructed a "Deep CNN" model using a training set augmented with a 50% black-and-white and 50% sepia mix, to evaluate whether a combination of color techniques might yield improved results.

The final test results of this mixed-color model, along with models based solely on sepia and black-and-white, are shown in the Figure 19. As illustrated, sepia continues to outperform the mixed-color approach. We hypothesize that the black-and-white transformation eliminates all color information, relying solely on intensity and lighting, while sepia retains some color cues, which may be crucial for distinguishing certain species.

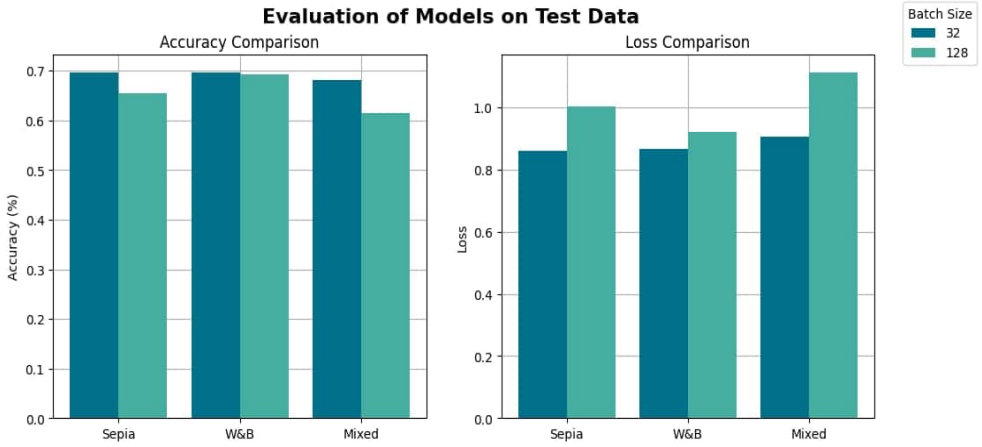


Figure 19: Test results of "Deep CNN" with color data augmentation techniques applied.

Most importantly, we are able to address the issue of animals that were frequently misclassified as frogs (see Figure 20).

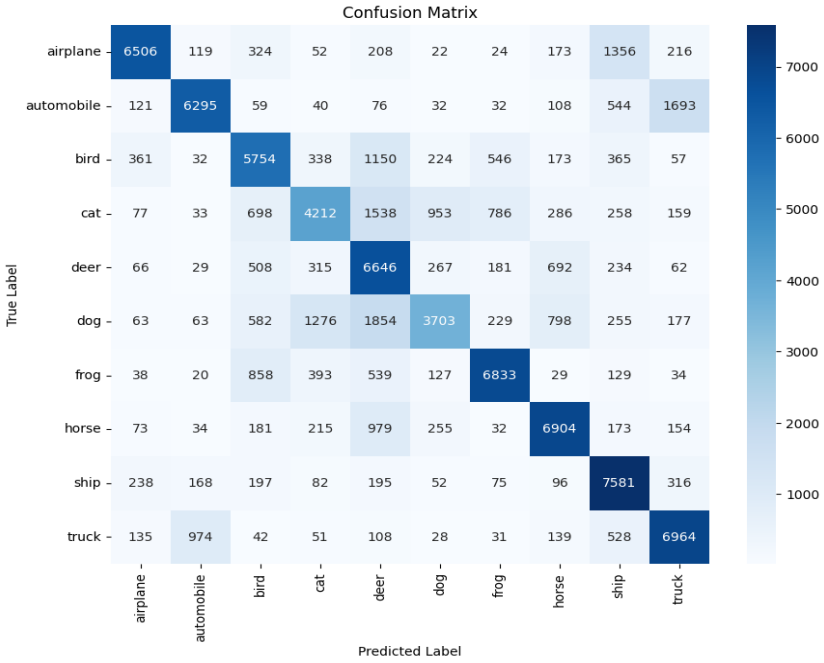


Figure 20: Confusion matrix for "Deep CNN" on trained mixed colors dataset with batch size 128. We see that we handle the issue of animals that are frequently misclassified as frogs.

4.3 Augmentation methods - Results

To evaluate the robustness of our convolutional models to various image distortions and transformations, we tested them on multiple augmented versions of the test set. Figure 21. presents the comparison of model performance across several augmentation techniques using two different batch sizes: 32 and 128.

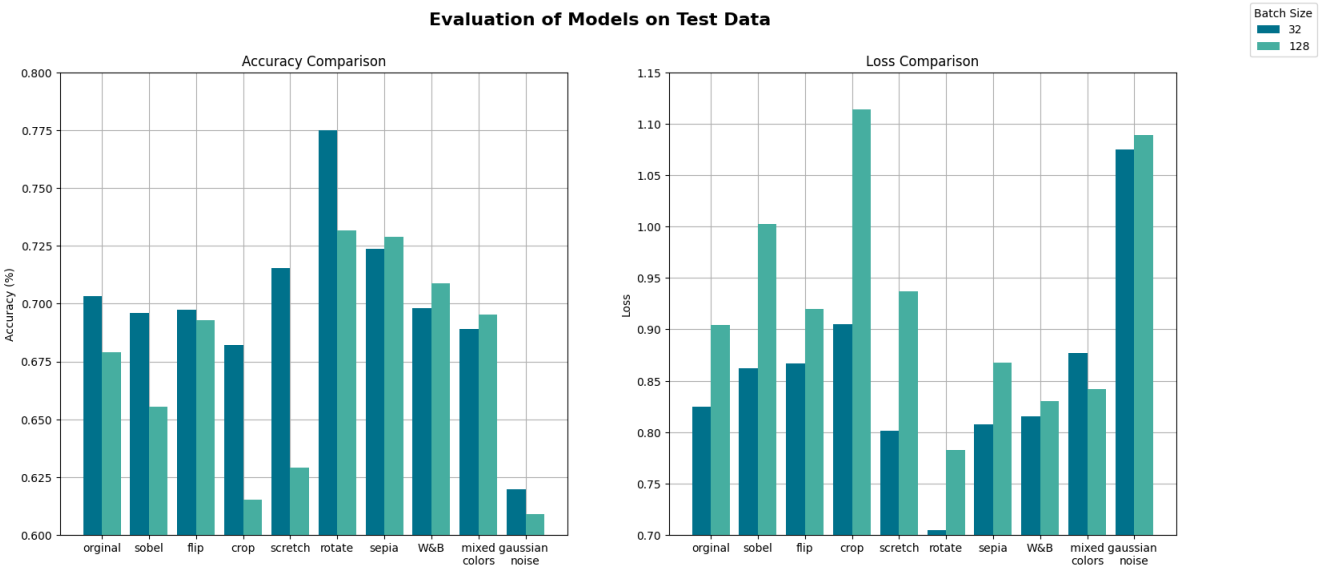


Figure 21: Comparison of model performance across different test set augmentations using two batch sizes (32 and 128).

The left chart illustrates the classification accuracy, while the right chart depicts the loss values. From the accuracy chart, we observe that certain augmentations—particularly rotation and sepia—maintain high accuracy levels across both batch sizes, with rotation achieving the highest accuracy (up to 77.5%). This suggests that the model is resilient to slight rotational variations and benefits from color normalization introduced by the sepia filter.

On the other hand, augmentations such as cropping and gaussian noise significantly degrade performance, leading to lower accuracy and higher loss. Cropping potentially removes essential object features, while Gaussian noise introduces a high degree of randomness, confusing the classifier. This effect is particularly pronounced with larger batch sizes, indicating decreased generalization under high noise conditions.

Interestingly, sobel filtering, stretching, black & white (W&B), and flipping produce moderate results. These transformations do not drastically degrade performance but also do not improve it significantly, making them neutral candidates for data augmentation.

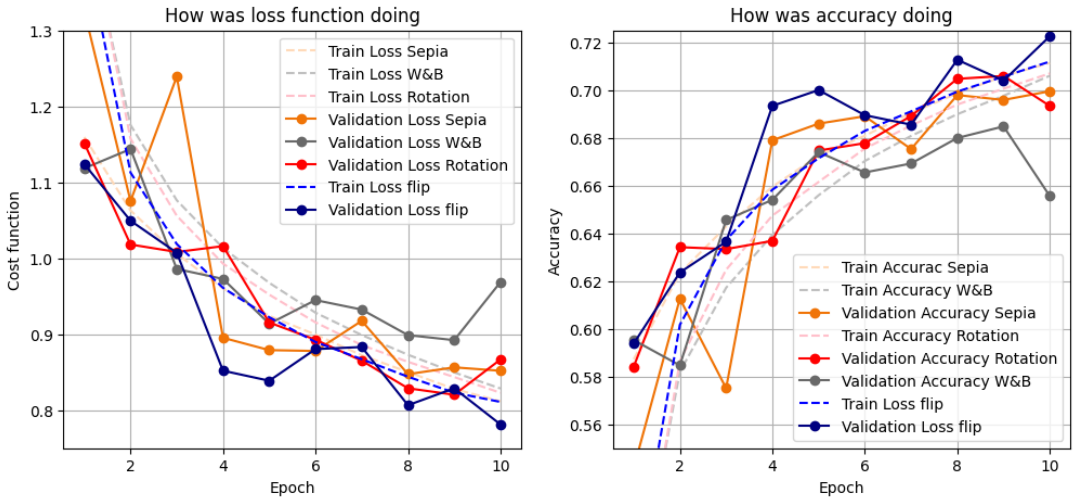


Figure 22: Comparison of model performance for different validation set augmentations using batch size 32

Additionaly, as shown in Figure 22, models evaluated on flipped and sepia-transformed images consistently achieved the lowest validation loss and the highest accuracy throughout the training process. This suggests that these augmentations align well with the model's learned representations and introduce variability that enhances generalization without compromising recognition quality. In contrast, the W&B displayed more volatility and generally underperformed, especially in later epochs, indicating a potential mismatch in distribution or higher complexity for the model to handle.

4.4 Conclusion and Dataset Expansion Strategy

Based on these findings, we concluded that the optimal training strategy should include only those augmentations that either maintain or slightly improve model generalization, while avoiding harmful ones such as gaussian noise. To further enhance model performance and reduce overfitting, we decided to create a new training dataset with enriched diversity.

Each original image will be replicated into four augmented versions, sampled based on the following probabilistic augmentation scheme:

- Random rotation (0°–20°): Applied to all copies
- Flip: 50% probability
- Sepia filter: 50% probability
- Sobel filter: 30% probability
- Black & White: 25% probability
- Cropping: 20% probability
- Stretching: 20% probability
- Gaussian Noise: Not included, due to its negative impact

This results in a new augmented training set of 360,000 images, derived from the original 90,000 training samples. The final dataset setup for subsequent experiments will be:

- Training set: 450,000 images (90k original + 360k augmented)
- Validation set: 90,000 images (unaltered)
- Test set: 90,000 images (unaltered)

This enhanced dataset is designed to better reflect real-world distortions and increase model robustness in more diverse visual environments. In the next sections, we will conduct model training and evaluation using this new setup.

5 More advanced experiments

5.1 Performance of ResNET34, ResNET18, VGG, GoogLeNet

In the previous sections, we focused on experimenting with our own CNN architectures to analyze their potential and limitations. In this part, we explore modern and well-established models from the literature [22–24], such as VGG, ResNet, and GoogLeNet, in order to improve performance and gain comparative insights. All models were trained using the full training dataset consisting of 450,000 images, with 90,000 images used for validation and 90,000 for testing, as described in the earlier section. For increased computing power, we used NVIDIA A100 Tensor Core GPU on google colab platform

5.1.1 VGG

VGG is known for its simplicity and uniform architecture, using only 3x3 convolutional filters and deep stacks of convolutional layers followed by fully connected layers [22]. Despite its large number of parameters, it often generalizes well on complex datasets.

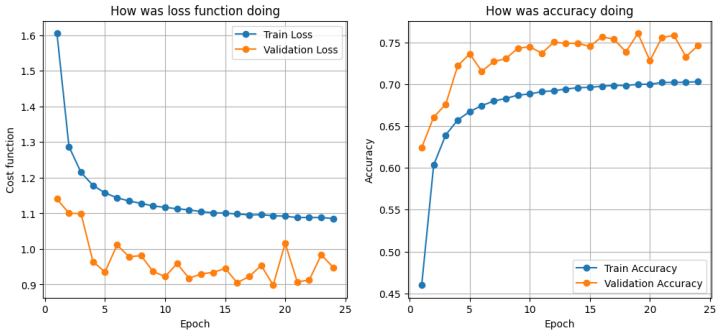


Figure 23: VGG - Learning process with validation set results (9000 validation samples per class).

As shown in the Figure 23., the validation accuracy of the VGG model consistently outperforms training accuracy, indicating strong generalization. Unlike our earlier custom architectures, VGG does not overfit even after 25 epochs (the learning process was stopped due to plateau). The training loss gradually decreases, and validation loss remains low and stable, confirming that the model adapts well to unseen data. This behavior contrasts with previous models, where limited capacity or overfitting were common.

5.1.2 ResNET34

ResNET34 introduces residual connections, which help mitigate the vanishing gradient problem in deep networks and allow for more efficient training of very deep architectures [23].

Compared to VGG, ResNET34 achieves a slightly lower final accuracy, but exhibits a much faster convergence during training. Already after a few epochs, the model reaches validation accuracy above 70%. Additionally, both training and validation curves are very close, indicating stable learning and low risk of overfitting. This contrasts with VGG, where the validation accuracy was higher than training accuracy for a long time, showing stronger regularization effects.

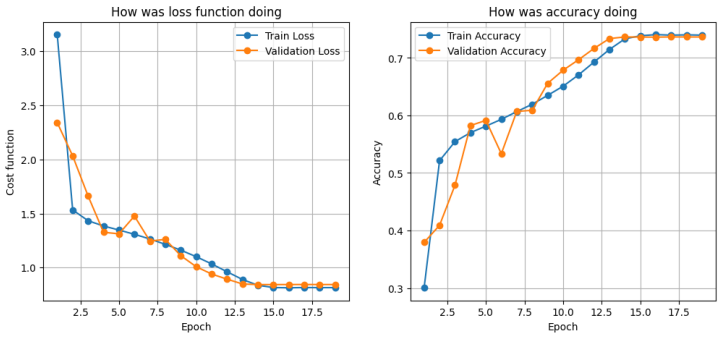


Figure 24: ResNET34 - Learning process with validation set results (9000 validation samples per class).

5.1.3 GoogLeNet

GoogLeNet introduces the innovative concept of *Inception modules*, which allow the network to perform convolutions with different filter sizes in parallel. This enables deeper architectures while keeping computational costs relatively low compared to traditional deep models [24]. The training curves show

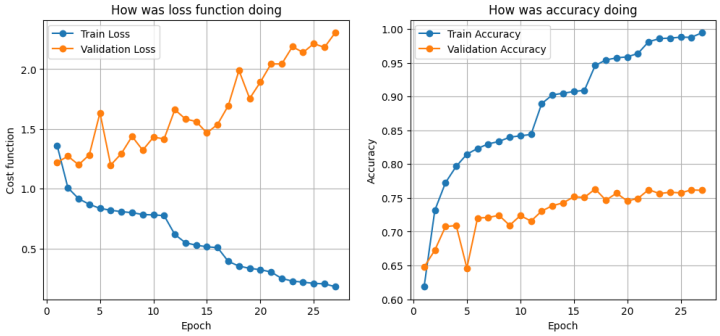


Figure 25: GoogLeNet - Learning process with validation set results (9000 validation samples per class).

a clear overfitting issue: while the training loss continues to decrease and training accuracy increases steadily, the validation loss starts rising around the 10th epoch and validation accuracy plateaus at approximately 76%. Compared to VGG and ResNET34, GoogLeNet does not generalize as well—possibly due to overfitting on the training data.

5.1.4 ResNET18

ResNET18, although shallower than ResNET34, proved to be very effective model in our experiments, achieving the highest accuracy on the test set. Its reduced depth helps to generalize data while still benefiting from residual connections that enable efficient gradient flow.

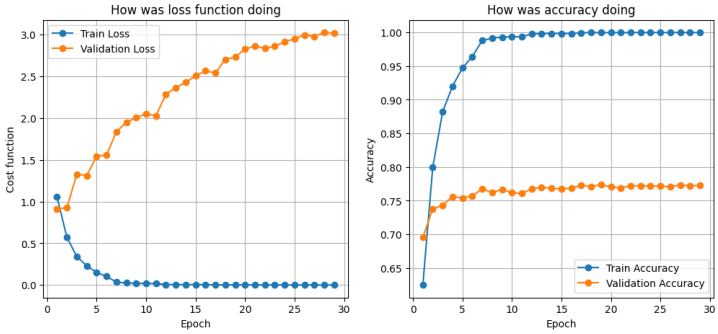


Figure 26: ResNET18 - Learning process with validation set results (9000 validation samples per class).

Compared to ResNET34, the validation accuracy stabilizes earlier and remains consistently higher

throughout training. The training accuracy quickly reaches near-perfect values, and although the validation loss starts to increase slightly, the accuracy remains strong and steady. This suggests that ResNET18 generalizes better in this setup, striking an excellent balance between model complexity and learning capability.

5.1.5 Comparison

Table 5. presents a comparative summary of five CNN architectures on the CINIC-10 test set. Among all models, **ResNet18** achieved the highest overall accuracy (0.77) and consistently strong F1-scores across most classes. **GoogLeNet** and **VGG** followed closely with solid generalization performance, particularly in classes like *airplane* and *ship*. Although the custom **Deep CNN** performed worse overall, it excelled in detecting classes such as *frog* and *ship*, showing the benefits of deeper feature hierarchies. These results confirm that modern residual architectures offer superior performance on diverse datasets like CINIC-10.

Table 5: Comparison of classification metrics for five CNN architectures (VGG, ResNet34, GoogLeNet, ResNet18, Deep CNN) on CINIC-10 (9000 test samples per class).

Class	**VGG** Prec / Rec / F1	**ResNet34** Prec / Rec / F1	**GoogLeNet** Prec / Rec / F1	**ResNet18** Prec / Rec / F1	**Deep CNN** Prec / Rec / F1
airplane	0.85 / 0.83 / 0.84	0.82 / 0.82 / 0.82	0.89 / 0.82 / 0.85	0.87 / 0.85 / **0.86**	0.88 / 0.68 / 0.77
automobile	0.81 / 0.80 / **0.81**	0.79 / 0.78 / 0.78	0.81 / 0.80 / 0.80	0.81 / 0.81 / 0.81	0.79 / 0.77 / 0.78
bird	0.75 / 0.73 / 0.74	0.71 / 0.70 / 0.70	0.73 / 0.75 / 0.74	0.74 / 0.76 / **0.75**	0.60 / 0.70 / 0.65
cat	0.68 / 0.63 / 0.65	0.60 / 0.63 / 0.61	0.74 / 0.55 / 0.63	0.64 / 0.68 / **0.66**	0.50 / 0.65 / 0.56
deer	0.60 / 0.80 / 0.68	0.64 / 0.67 / 0.66	0.68 / 0.73 / 0.70	0.72 / 0.72 / **0.72**	0.63 / 0.59 / 0.61
dog	0.67 / 0.54 / 0.60	0.62 / 0.53 / 0.57	0.62 / 0.64 / **0.63**	0.66 / 0.59 / 0.63	0.60 / 0.41 / 0.49
frog	0.83 / 0.87 / **0.85**	0.80 / 0.83 / 0.81	0.79 / 0.88 / 0.83	0.84 / 0.84 / 0.84	0.70 / 0.88 / 0.78
horse	0.85 / 0.78 / 0.81	0.78 / 0.79 / 0.78	0.76 / 0.83 / 0.80	0.82 / 0.82 / **0.82**	0.79 / 0.73 / 0.76
ship	0.79 / 0.86 / 0.82	0.78 / 0.82 / 0.80	0.81 / 0.85 / 0.83	0.83 / 0.85 / **0.84**	0.71 / 0.83 / 0.77
truck	0.80 / 0.75 / 0.78	0.77 / 0.77 / 0.77	0.81 / 0.76 / 0.78	0.80 / 0.79 / **0.79**	0.83 / 0.67 / 0.74
Accuracy	**0.76**	**0.73**	**0.76**	**0.77**	**0.69**

Additionaly, the comparison of training and validation accuracy across four CNN architectures on Figure 29. reveals that ResNet18 achieves the most consistent and highest validation accuracy throughout training, suggesting strong predictive capability. In contrast, GoogLeNet and VGG show more fluctuations on the validation set despite stable training performance, indicating potential sensitivity to data variance. Overall, ResNet18 and VGG balance best learning efficiency among the tested models.

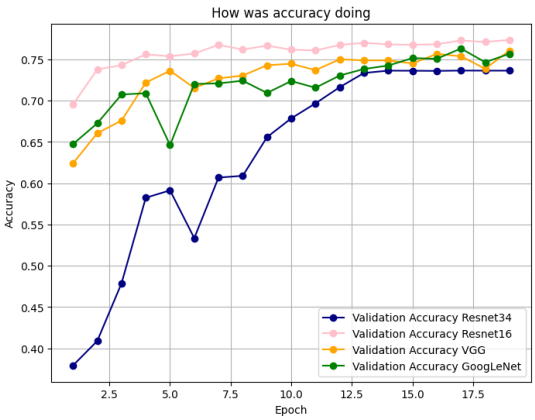


Figure 27: Validation accuracy

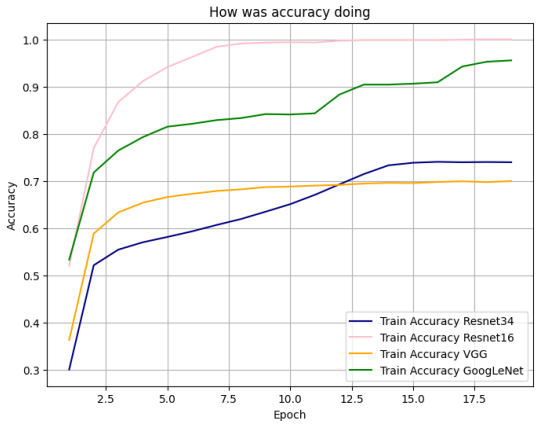


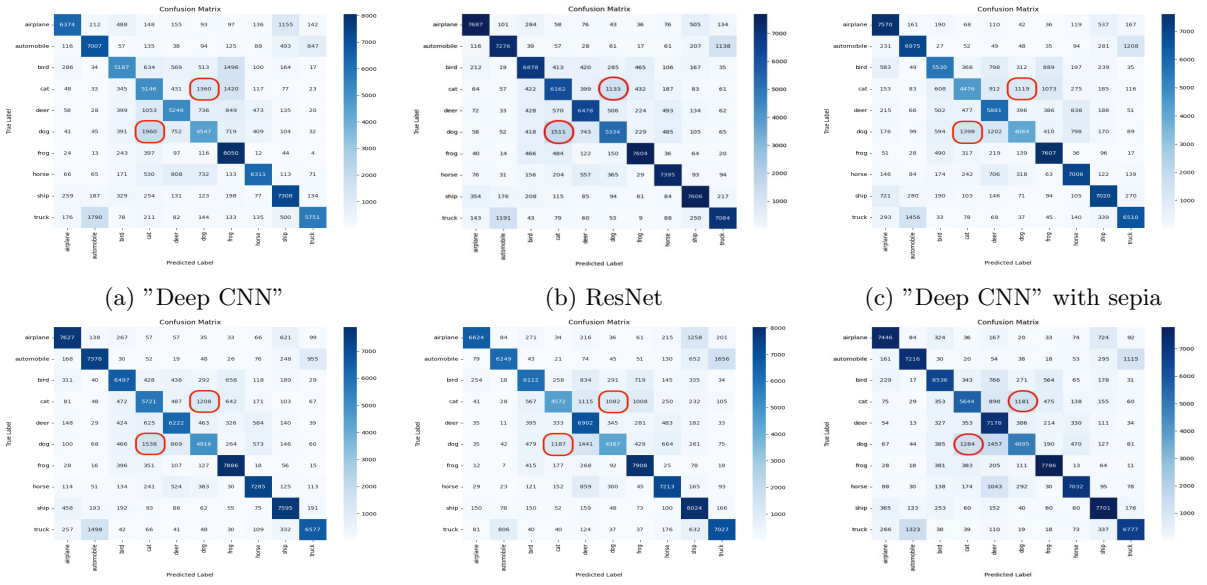
Figure 28: Training accuracy

Figure 29: Comparison of validation accuracy for four CNN architectures (VGG, ResNet34, GoogLeNet, ResNet18) on CINIC-10 (9000 validation samples per class).

5.2 Ensemble learning

Ensemble learning is a technique that combines multiple models to generate more accurate and robust predictions. By aggregating outputs from different models, ensembles help mitigate individual model biases, enhance generalization, and reduce overfitting. [25] Common ensemble strategies include averaging predictions, majority voting, and stacking, where multiple models contribute to the final decision. In this section, we analyze the impact of ensemble learning on image classification and compare various ensemble approaches applied to our models.

During our experiments, we observe that the models frequently misclassify cats and dogs, often confusing one for the other (see Figure 30). Despite testing multiple CNN architectures, even the best-performing models struggle with this distinction. To address this issue, we develop a specialized ensemble designed to improve classification accuracy for these two categories. Our hypothesis is that combining the predictions of diverse models and incorporating targeted augmentations can mitigate this confusion.



(a) "Deep CNN" (b) ResNet (c) "Deep CNN" with sepia

(d) "Deep CNN" with flip (e) "Deep CNN" with stretch (f) VGG

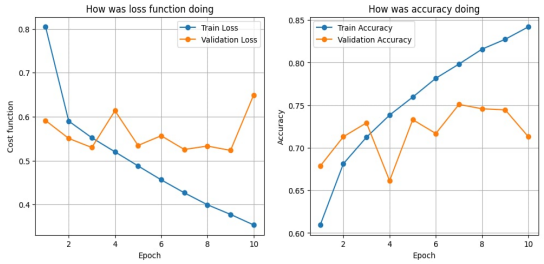
Figure 30: Confusion matrices illustrating misclassifications of cats and dogs for selected individual CNN models training on batch size 32 (as this works usually the best). The highlighted areas indicate instances where the models incorrectly predicted a cat as a dog and a dog as a cat.

For our ensemble, we select "Deep CNN" trained with sepia images and "Deep CNN" trained with flipped images, as both demonstrate strong accuracy and loss scores during testing. These augmentations serve distinct purposes: sepia transformation reduces the model's reliance on color information, while flipping enhances generalization by exposing the model to varied object orientations. Additionally, we incorporate ResNet, as it is the best-performing individual model in our previous experiments. To further strengthen the ensemble, we include two models trained specifically on cat and dog images: "Deep CNN" with stretched images and VGG with stretched images. These models exhibit the highest accuracy in distinguishing between these two classes. Stretching proves to be particularly effective for this task. Given the limited size of the cat and dog dataset, we opt for a 4:1 train-to-validation split to enhance validation robustness. Moreover, applying stretching augmentation expands the training set, further improving generalization.

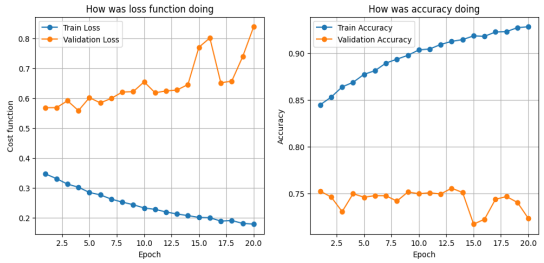
5.2.1 "Cat&dog" models

As mentioned earlier, a batch size of 32 works well, and we choose to maintain this setting for our cat and dog models. Additionally, with the smaller size of the current dataset, a smaller batch size will help mitigate overfitting. As outlined in the introduction to section, we train our models on a dataset enlarged with stretching augmentation technique, and also using a 4:1 train-to-validation split. The Figure 31 and Figure 32 illustrate the training process of the models. We begin with the 'Deep CNN' model, training it for 10 epochs due to limited GPU resources. Given the promising initial results, we proceed to train the

model for an additional 20 epochs. This run yields a similar accuracy on the validation set; however, it results in a higher loss. Since the training duration is 12 hours and no checkpoint saving is implemented in this experiment, we decide to move forward with the 'Deep CNN' model trained for 20 epochs. For the next model, VGG, we initially train it for 20 epochs, anticipating satisfactory results. Although the model achieves slightly better performance, the improvement is not as substantial as expected. Observing an increase in accuracy and a decrease in loss in the final epoch, we opt to repeat the experiment with 30 epochs, this time utilizing checkpoint saving to preserve the best accuracy, loss score, and final model. However, the results after 30 epochs remain comparable to those obtained after 20 epochs.

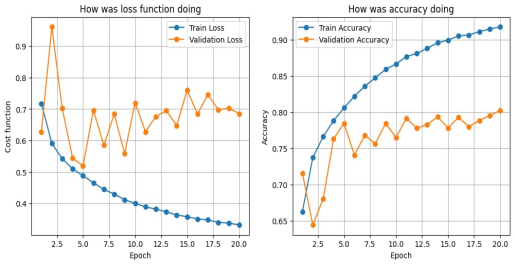


(a) Training with 10 epochs

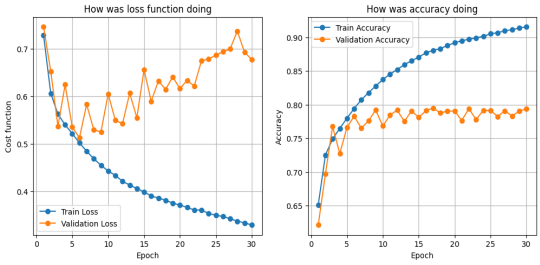


(b) Training with 20 epochs

Figure 31: Training process of "Deep CNN" model for cats and dogs instances.



(a) Training with 20 epochs



(b) Training with 30 epochs

Figure 32: Training process of VGG model for cats and dogs instances.

Testing the 'Cat&Dog' models demonstrates highly optimized results (see). Among the VGG models, we specifically select the one with the lowest loss for further ensemble learning, as it exhibits exceptional performance in predicting dogs. The 'Deep CNN' model, on the other hand, struggles with accurately predicting dogs but performs well with cats. Given that the VGG model with the lowest loss excels at predicting dogs, while the 'Deep CNN' is highly accurate for cats, this combination is expected to generalize effectively when utilized in ensemble learning.

5.2.2 Voting ensemble & weights

We select the voting technique for ensemble creation as it provides the greatest control over the decision-making process.

Our ensemble combines models in a two-stage approach. First, predictions for all 16 classes are generated by averaging the votes from three models: "Deep CNN" trained with sepia images, "Deep CNN" trained with flipped images, and ResNet. If the predicted class is either a cat or a dog, the image is then forwarded to the specialized "Cat&Dog" models, where a second voting process determines the final classification. Since the three primary models demonstrate generally strong performance in predicting the broader set of classes, we use simple averaging for their voting. However, determining the appropriate voting weights for the "Cat&Dog" models presents a challenge. To address this, we conduct an experiment comparing ensemble performance on the validation set across different voting weight distributions. The results, shown in Figure Figure 33, indicate that the optimal accuracy and loss values are achieved when the "Deep CNN" receives a voting weight of 0.3 and the VGG model is assigned 0.7. Based on these findings, we adopt this weighting scheme for further analysis.

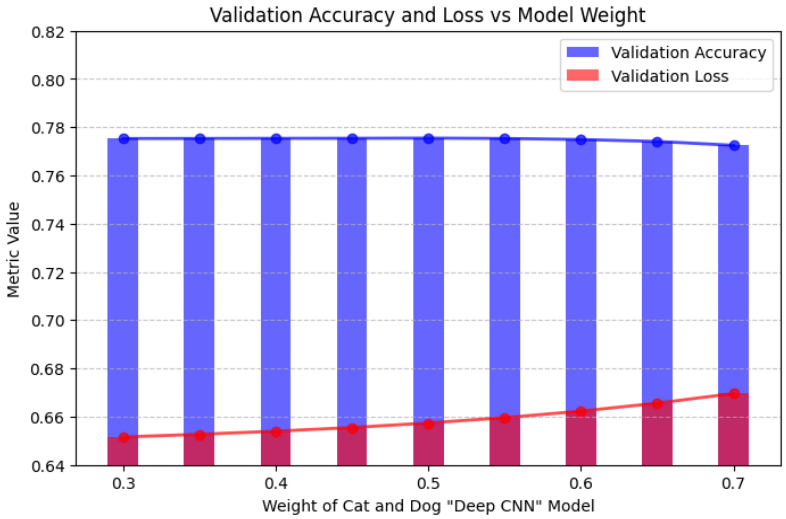


Figure 33: Validation loss and accuracy of ensemble learning based on distribution of "Cat&dog" models. While on figure we see the voting weights assigned to the "Deep CNN" model in the 'Cat&Dog' classification stage, the corresponding weights for the VGG model is given by 1-"Deep CNN"_weights

5.2.3 Ensemble learning - Results

We evaluate the developed model on the test dataset to assess its predictive performance, achieving:

- *accuracy*: 77.5334%;
- *loss*: 0.6516.

This represents the best performance among all the models we examined. Looking at the confusion matrix (see Figure 34), the accuracy of cat and dog predictions remains largely unchanged. However, the classification of other image categories shows a significant improvement.

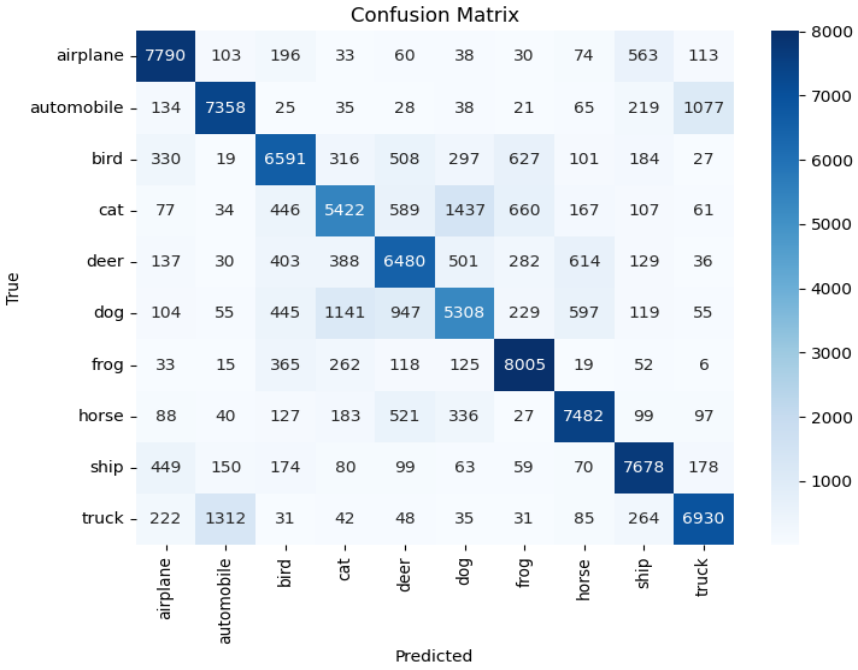


Figure 34: Confusion matrix that presents prediction of ensemble model. Predictions significantly improved. Cats and dogs still can be better classify.

As observed, the classes for "automobile" and "truck" are frequently misclassified. To address this

issue, we decide to incorporate an additional model into our ensemble, specifically designed to predict these two classes. Given our limited computational resources and the long training times, this new model will consist solely of the VGG architecture, trained on an augmented training set, where stretching augmentation is applied. The training dataset is divided into a 4:1 ratio for the training and validation sets, respectively. Figure 35 presents the results, demonstrating a significant improvement, particularly for the "truck" class. With the inclusion of this model, we achieve an even higher accuracy. The results are as follows:

- *accuracy*: 78.4189%;
- *loss*: 0.6401.

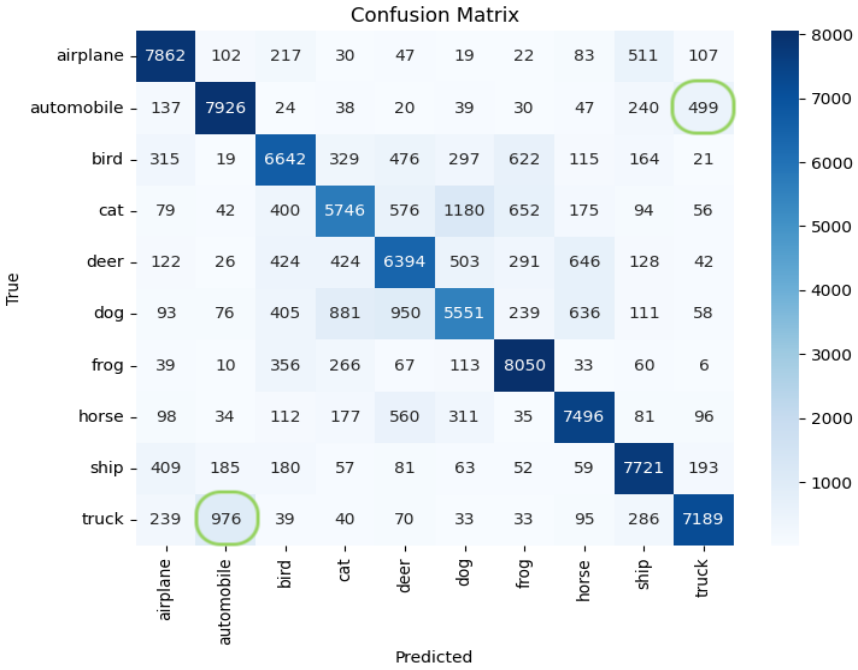


Figure 35: Confusion matrix after adding special modekl for automobiles and trucks. We see significant improvement in predicting these two classes (highlighted in green).

The final experiment we conducted regarding ensembles involved changing the voting strategy of the large model from averaging the votes to majority voting. We performed this experiment with the "Cat&dog" model, excluding the automobile and truck model, to assess whether this change would positively impact the ensemble and whether the optimal voting weights would shift to a different configuration. In this case, the best model again corresponded to the voting weight configuration of 0.7 for VGG and 0.3 for "Deep CNN," further emphasizing that VGG is better at recognizing these two animal classes. However, the overall results were slightly worse compared to the previous configuration:

- *accuracy*: 76.3489%;
- *loss*: 3.5064.

Figure 36 presents the confusion matrix of the model's testing results, allowing for a comparison with the previous models.

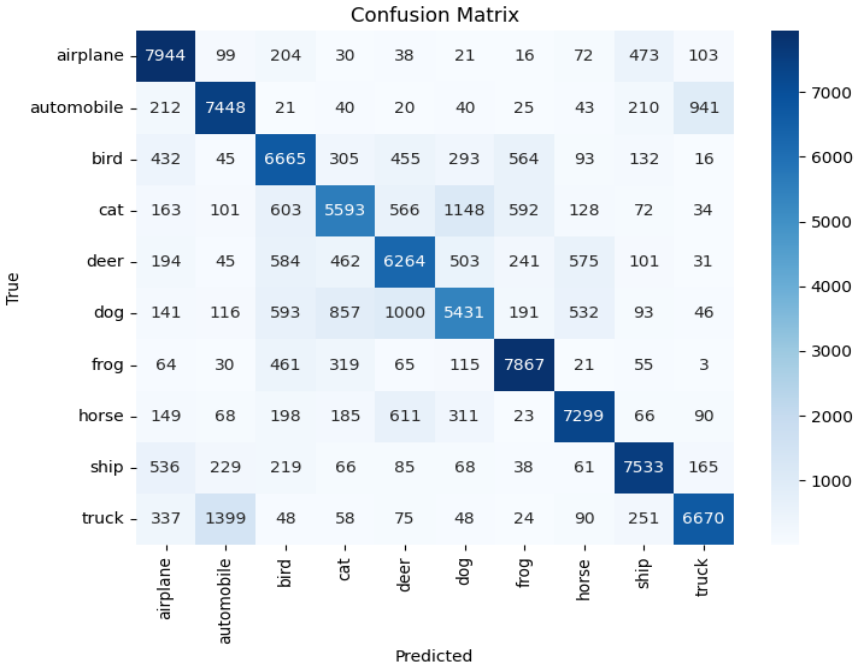


Figure 36: Confusion matrix for the first ensemble model type with changed voting style of "larger" model from averaging to majority voting.

6 Conclusions

In this study, we conducted a comprehensive evaluation of Convolutional Neural Networks (CNNs) on the CINIC-10 dataset to investigate the impact of architecture depth, data augmentation, training data volume and ensemble technique on classification performance. Starting from simple, custom-built models such as the "Single Convolution Layer CNN" and "Double Convolution Layer CNN", we observed clear limitations in generalization capacity due to their shallow nature and limited representation power.

To address these limitations, we implemented a deeper custom architecture referred to as the "Deep CNN", which incorporated regularization techniques like dropout and batch normalization. This architecture provided moderate improvements, reaching a test accuracy of approximately 70% when trained on a 144k/36k/90k dataset split.

Subsequently, we evaluated several architectures from the literature—VGG, ResNet18, ResNet34, and GoogLeNet. Among them, ResNet18 emerged as the best-performing model with a test accuracy of 77%, showing its ability to generalize well while maintaining relatively low complexity.

Additionally, we tested a variety of data augmentation strategies (e.g., rotation, stretching, flipping, color transformations, and Gaussian noise) and observed that specific augmentations significantly influenced performance depending on the class and the model used. Techniques like flipping and stretching proved especially effective for resolving class confusion between visually similar categories like cats and dogs. In contrast, the worst-performing augmentation technique was Gaussian noise, which notably decreased performance compared to the original dataset. We hypothesize that this may be due to the CINIC-10 dataset itself already has lower quality, with inherent blurring and other imperfections, making it more sensitive to the addition of noise.

Finally, we introduced an ensemble learning approach that combined models trained with different augmentations and architectures. This ensemble strategy proved highly successful, achieving the highest test accuracy of above 78%, outperforming all individual models. The ensemble also improved the robustness of classification, especially for challenging classes.

These findings confirm that combining models and leveraging targeted data augmentations can significantly enhance CNNs performance in the image classification tasks. Further development of ensemble-based systems holds strong potential and should be a key direction for future research in computer vision on real-world noisy datasets like CINIC-10.

7 Bibliography

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 12 1989.

[2] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[4] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):652–663, 2016.

[5] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

[6] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[8] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[9] Antreas Antoniou Luke N. Darlow, Elliot J. Crowley and Amos J. Storkey.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Sena Yağmur Şen and Nalan Özkurt. Convolutional neural network hyperparameter tuning with adam optimizer for ecg classification. In *2020 innovations in intelligent systems and applications conference (ASYU)*, pages 1–6. IEEE, 2020.

[15] Smit Mehta, Chirag Paunwala, and Bhaumik Vaidya. Cnn based traffic sign classification using adam optimizer. In *2019 international conference on intelligent computing and control systems (ICCS)*, pages 1293–1298. IEEE, 2019.

[16] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: tricks of the trade: second edition*, pages 421–436. Springer, 2012.

[17] Pujo Hari Saputro, Dhina Puspasari Wijaya, Musthofa Galih Pradana, Dyah Listianing Tyas, and Wahyuni Fithratul Zalmi. Comparison adam-optimizer and sgdm for classification images of rice leaf disease. In *2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, pages 348–353. IEEE, 2022.

[18] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[19] Luke N. Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. Cinic-10 is not imagenet or cifar-10, 2018.

[20] Youming Wang, Zhao Xiao, and Gongqing Cao. A convolutional neural network method based on adam optimizer with power-exponential learning rate for bearing fault diagnosis. *Journal of Vibroengineering*, 24(4):666–678, 2022.

[21] Francois Chollet and François Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[25] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.