Lightweight CNN for Image Classification with Preprocessing, Augmentation, and Feature Visualization

Bala Nithin Chennarapu, Prathyusha Pentam, Madhusudhan Poduturu
Department of Computer Science
Texas A&M University–Corpus Christi
Corpus Christi, TX, USA

***Abstract*—Convolutional neural networks (CNNs) are the standard approach for images classify on datasets such as CIFAR-10 datasets, but typical high-capacity architectures assume access to high amount of labeled data and practical resources. In many educational and practical scenarios, however, we work with smaller datasets, limited hardware, and noisy images with low resolution and varying contrast. This paper presents a compact and interpretable CNN pipeline for multi-class image classification on a 10,000-image subset of CIFAR-10. The pipeline combines CLAHE for contrast enhancement, data augmentation, and a lightweight three-block CNN with global average pooling. We trained our dataset model on 7,000 images, validate on 1,500, and test dataset of 1,500 images. The final network achieves 48.9% test accuracy with macro precision, recall, and F1-score of 0.539, 0.489, and 0.480 values. We analyze class-wise performance, highlight typical confusion patterns, and visualize feature maps from multiple convolutional layers to study what the network learns. The results show that the proposed pipeline offers a reasonable balance between performance, efficiency, and interpretability on a small, low-resolution image dataset, and it provides a practical template for classroom projects and lightweight deployments.**

***Index Terms*—Convolutional neural networks, CIFAR-10, CLAHE, data augmentation, feature visualization, lightweight models.**

I. Introduction

CNNs have become the dominant paradigm for image classification problems, achieving excellent values performance on large datasets such as CIFAR-10 and ImageNet [1], [2]. These data models can learn step by step features directly from raw pixels when trained on large labeled datasets with sufficient compute. However, the assumptions behind many widely used architectures—deep networks with millions of parameters and long training schedules—are not always realistic in resource-constrained environments.

In classroom settings and many practical applications, we often work with smaller datasets, limited GPU memory, and time constraints. Additionally, real-world images may be noisy and low resolution, with variations in lighting, contrast, and background clutter. Under these conditions, naively training a large CNN on a small dataset can lead to severe overfitting and unstable generalization. This motivates the design of lightweight, carefully regularized models that incorporate explicit preprocessing and augmentation to increase the effective quality and diversity of dataset.

In this work we focus on a 10,000-image subset of the CIFAR-10 dataset. We investigate the following question: *how well can a compact CNN perform on a relatively small and noisy image set when combined with targeted preprocessing and augmentation, and what kinds of features does it learn?*

A. Contributions

The main key outcomes from this paper are:

- We construct a stratified 10,000-image subset of CIFAR-10 and design a reproducible pipeline that includes CLAHE-based contrast enhancement, normalization, and on-the-fly data augmentation.
- We propose a lightweight CNN with three convolutional blocks (32–64–128 filters) and a global average pooling head, totaling approximately 158k parameters, suitable for fast training on modest hardware.
- We empirically evaluate the pipeline on a test dataset, reporting accuracy, small precision, recall, and F1-value and a confusion matrix and class-wise analysis.
- We visualize feature maps from several convolutional layers to examine the progression from low-level edge detectors to more abstract patterns, providing insight into what the network has learned.

II. Related Work

A. CIFAR-10 and Small-Scale Image Classification

The CIFAR-10 data image set [1] contains of 60,000 color images dataset of size 32×32 distributed same across ten object image categories. The low resolution and complex backgrounds make the dataset challenging, especially for fine-grained distinctions between animal classes. While deep architectures such as ResNet and DenseNet achieve high accuracy on the full dataset, their capacity can be excessive and inefficient for small-scale scenarios.

B. Lightweight CNN Architectures

For constrained settings, lightweight architectures with fewer parameters and strong regularization are preferable. Networks that combine a small number of convolutional blocks with global average pooling have been shown to provide strong baselines on small datasets, especially when augmented with data augmentation and careful learning-rate scheduling.

C. Histogram Equalization and CLAHE

Histogram equalization and its variants, such as CLAHE [3], enhance local contrast by redistributing pixel intensities. When applied to the luminance channel of an image, CLAHE can make edges and textures more visible while limiting amplification of noise. Integrating such preprocessing with CNNs can improve robustness to lighting and contrast variations.

III. Methodology

A. Dataset Construction

We start from the standard CIFAR-10 train/test split and combine all 50,000 training images dataset and 10,000 test data images into a single pool. From this pool, we perform stratified sampling to select a balanced subset of 10,000 images, preserving class proportions. The subset is then divided into three mutually exclusive splits:

- 7,000 dataset images for training,
- 1,500 dataset images for validate,
- 1,500 images dataset for testing.

Each split is stored in a directory structure organized by class, making it straightforward to load with modern deep learning frameworks.

To ensure reproducibility, we fix random seeds for Python, NumPy, and the deep learning library and also set environment variables that control hash randomization. This guarantees repeatable dataset splits and batch orders.

B. Preprocessing with CLAHE

All images are processed through a CLAHE-based contrast enhancement stage. For each RGB image, we:

1) Change the image from RGB to color.
2) Extract the luminance (L) channel and apply CLAHE.
3) Merge the enhanced L channel back with the original A and B channels.
4) Convert the LAB image we got again reverse to RGB.

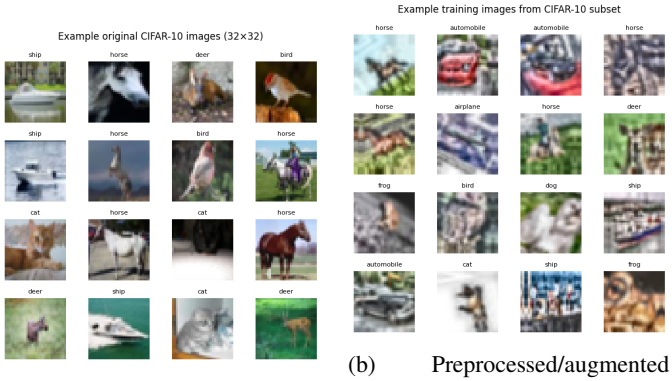
This operation is wrapped in a framework-compatible function so that it can be embedded inside the input pipeline. After CLAHE, we rescale the pixel values to $[0, 1]$ using a simple normalization layer.

C. Data Augmentation

To mitigate overfitting on the relatively small training set and to increase effective data diversity, we employ on-the-fly augmentation during training only. The augmentation stack consists of:

- random horizontal flips,
- random rotations up to approximately $\pm 18°$,
- random translations up to 10% of the image dimensions,
- random zooms up to 10%.

Because augmentation is applied at each epoch, the model sees slightly different versions of each image over time, encouraging the learning of invariant and robust features.



(a) Original CIFAR-10 images.

(b) Preprocessed/augmented samples.

Fig. 1: Example images from the 10k CIFAR-10 subset used in this work.

D. Sample Dataset Images

Fig. 1 illustrates example images from the original CIFAR-10 subset as well as augmented training samples after preprocessing.

E. Lightweight CNN Architecture

Our classification model is a compact CNN with three convolutional blocks carried on by by a entire average pooling and a small entire connection head. Using Keras-style notation, the architecture is:

The model has approximately 158,570 parameters (about 619 KB in memory), with 157,674 trainable parameters. This size is small enough for fast training on a single GPU or CPU while still allowing the network to learn non-trivial representations.

- Block 1: two 3×3 convolutions with 32 filters, batch normalization and ReLU activations, followed by 2×2 max pooling and 0.25 dropout.
- Block 2: two 3×3 convolutions with 64 filters, batch normalization and ReLU, followed by 2×2 max pooling and 0.25 dropout.
- Block 3: one 3×3 convolution with 128 filters, batch normalization and ReLU, followed by 2×2 max pooling and 0.25 dropout.
- Head: global average pooling, a 128-unit dense layer with batch normalization, ReLU and 0.5 dropout, and a 10-unit softmax output.

F. Training Setup

We trained our data model by help of the Adam optimizer which has a learning rate of 10^{-3} and of cross-entropy loss. The primary metric for monitoring is classification accuracy. Training is performed for a maximum of 30 epochs with the below restricts:

- **Early Stopping**: this checks the validation data loss with a continous of 5 epochs and enters the best weights of the data.
- **ModelCheck**: this gets the data model with the lowest validation loss.
- **ReduceLossOnPlateau**: This reduces the learn rate to half if the val data loss does not change for 2 cont. epochs, down to a minimum of 10^{-5}.

Training and validation metrics are recorded for subsequent analysis.

IV. Experimental Results

A. Training and Val. dataset Behavior

During training, both the training and val dataset accuracy improve steadily across the epochs. At the start, the model obtains around 22% training accuracy and 13% validation accuracy, slightly above random guessing over ten classes. As training progresses, the model learns useful patterns and the training accuracy gradually reaches about 60%, while validation accuracy stabilizes around 45–46%.

Fig. 2 tells us the training and val data accuracy graphs, and Fig. 3 tells us the respective loss data curves.

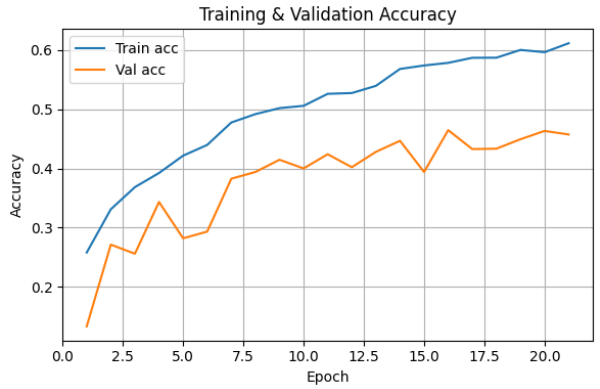


Fig. 2: Training and validation accuracy across epochs.

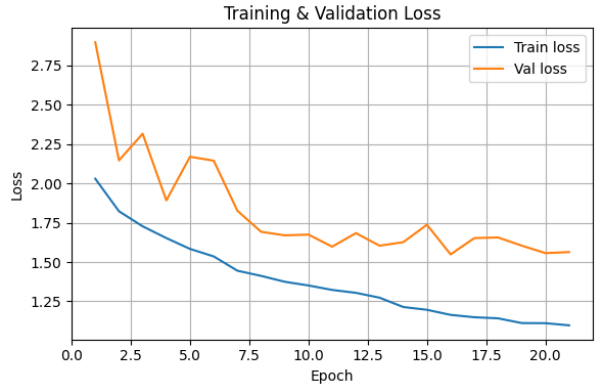


Fig. 3: Training and val data loss across epochs.

Here the decreasing training dataset loss and stabilizing validate dataset loss indicate that the model fits the training data well while maintaining moderate generalization. The value breakdown between training and validation dataset onpoint values suggests some overfit data, which is expected given limited training set size, but early stopping and the learn rate deduction keep it under control.

B. Test Performance

We evaluate the best model, selected by validation loss, on the testing dataset of 1,500 imgs. The model attains a test loss of 1.4838 and a test accuracy of 48.93%. To obtain a more detailed view of performance, we compute additional metrics using macro averaging across all classes:

- macro precision: 0.5391,
- macro recall: 0.4893,
- macro F1-score: 0.4802.

These values indicate that, on average, the model correctly recognizes about half of the examples for each class, and when it predicts a class, it is correct slightly more than half the time.

C. Confusion Matrix and Class-wise Analysis

Fig. 4 presents the confusion matrix on the test set. Vehicles such as automobiles, ships, and trucks are classified more accurately than several animal classes. For example, the *automobile* class shows high precision (0.74) and good recall (0.63), while the *ship* class exhibits very high recall (0.81) but moderate precision. In contrast, animal classes such as *bird*, *deer*, *cat*, and *horse* are more frequently confused, reflecting the subtle differences between these categories at low resolution.

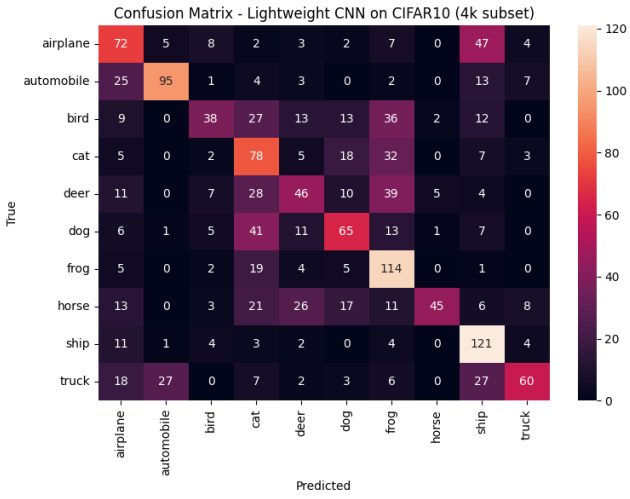


Fig. 4: Confusion matrix of the lightweight CNN on the CIFAR-10 subset test set.

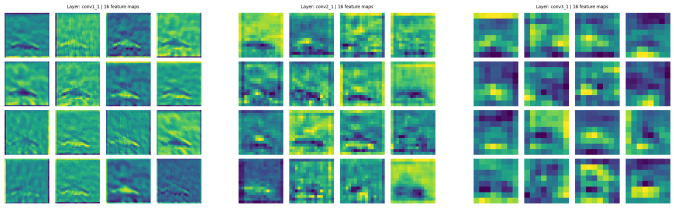
These patterns are typical for CIFAR-10: animals tend to be harder and more overlapping, whereas vehicles are easier to distinguish. The confusion matrix highlights where future improvements, such as class-specific augmentation or architectural adjustments, might be most beneficial.

D. Feature Map Visualization

To understand what the model actually learns, we visualize feature maps from several convolutional layers using an auxiliary activation model. For selected test images, we inspect early, middle, and deep feature maps.

In the earliest convolutional layer, feature maps primarily respond to simple edges, color gradients, and local textures. Middle-layer maps become more abstract and localized, capturing combinations of edges and small parts of objects. Deep-layer maps are even more selective, often responding only in specific spatial regions corresponding to higher-level patterns.

Fig. 5 shows representative feature maps from three convolutional layers. These visualizations confirm that the network learns a hierarchical representation typical of CNNs, despite its limited size and the modest training dataset.



(a) Layer 1. (b) Layer 2. (c) Layer 3.

Fig. 5: Example feature maps from early, middle, and deep convolutional layers of the lightweight CNN.

V. Conclusion and Future Work

We showed a lightweight CNN pipeline of image classification on a 10,000-image subset of CIFAR-10. The approach integrates CLAHE-based contrast enhancement, data augmentation, and a compact three-block CNN with global average pooling. On a held-out test set of 1,500 images, the model achieves 48.9% accuracy with macro F1-score of 0.48, which is reasonable for a small, low-resolution dataset and a modest architecture.

The experiments show clear patterns: vehicle classes are relatively easy to distinguish, while several animal classes remain challenging. The training and validation curves indicate moderate but controlled overfitting, and feature map visualizations confirm that the network learns structured, hierarchical features rather than memorizing labels.

Future work could explore depthwise separable convolutions or MobileNet-style blocks to further improve accuracy without significantly increasing model size. More systematic hyperparameter tuning, alternative preprocessing strategies, and advanced interpretability methods such as Grad-CAM could provide deeper insights. Finally, scaling the same design to larger portions of CIFAR-10 or to other image data set can help to tell that the generality of the told pipeline.

Acknowledgment

Our teammates would like to thank Dr. Minhua Huang for her guidance and feedback throughout COSC-6324 Digital Image Processing course.

References

[1] A. Krizhevsky, "Learning multiple layers of features from tiny images," Technical Report, University of Toronto, 2009.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] K. Zuiderveld, "Contrast limited adaptive histogram equalization," in *Graphics Gems IV*. Academic Press, 1994, pp. 474–485.