CIFAR-10 Image Classification Using a CNN

Akash Pillai, Sukanya Sahoo, Lauren Fuller, Abhishek Singh, Asmita Desai
Institute of Data Science, Texas A&M
College Station, Texas
akash.pillai.0810@tamu.edu, sukanya.sahoo@tamu.edu, lfuller6@tamu.edu,
abhi_singh@tamu.edu, asmi@tamu.edu

***Abstract*—This study analyzes the use of a Convolutional Neural Network (CNN) model for image classification of the CIFAR-10 dataset. The CNN uses convolutional, max-pooling, and fully connected layers to improve classification accuracy. The study evaluates the impact of various training strategies, hyperparameter tunings, and architectural modifications on the classification accuracy of the CNN. The findings underscore the potential CNNs have for real-world image classification applications in resource-constrained environments.**

***Index Terms*—CNN, image classification, accuracy, training.**

I. Introduction

Image classification is a fundamental task in computer vision, where images are categorized into predefined classes based on their visual characteristics. This study explores the implementation of a convolutional neural network (CNN), a leading model for image classification due to its powerful feature extraction capabilities, using the CIFAR-10 dataset.

A. Overview of Dataset

The CIFAR-10 dataset consists of 60,000 32x32 RBG images evenly distributed across ten classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each class contains 6,000 images, with mutually exclusive categories. The dataset's balance and diversity make it ideal for evaluating classification models.

Fig. 1. Sample Images From Each Class in CIFAR-10 Dataset

B. Analytical Approach and Experiments

This study implements a CNN for CIFAR-10 image classification and examines its performance across different architectures and hyperparameter settings. Key techniques used include data augmentation for improving generalization, dropout for regularization, and max-pooling for spatial feature extraction. The experiments optimize accuracy, recall, and precision through different architectural and training refinements detailed later.

C. Literature Review

Research on CIFAR-10 image classification highlights CNNs as the most widely used models due to their hierarchical feature extraction [1]. ImageNet studies have showcased the effectiveness of deep convolutional networks in advancing large-scale image classification [2]. Hybrid approaches like CNN-SVM models [3] and PCA-based dimensionality reduction [4] have also delivered notable results, offering valuable methodological insights. EfficientNet, balancing depth, width, and resolution, and ShuffleNet V2, utilizing group convolutions and channel shuffling for efficiency, enhance accuracy while reducing computational costs.

These developments demonstrate the continuous evolution of CNNs for improved performance and efficiency. Building on these, our research focuses on architectural refinements and training strategies, emphasizing the roles of data augmentation, dropout, and pooling in optimizing classification accuracy for high-performance models.

II. Method

This section outlines the implementation of a CNN for CIFAR-10 image classification. The outline includes data preprocessing, model architecture design, training with data augmentation, and performance evaluation.

A. Data Preprocessing

The 60,000 images of the CIFAR-10 dataset, accessed through the `tensorflow.keras.datasets` API [5], were divided into three subsets: 45,000 for training, 10,000 for testing, and 5,000 for validation, with each subset containing an equal number of images per class.

For preprocessing, the pixel values of each image were normalized to the [0, 1] range by dividing by 255, as a pixel value ranges from 0 to 255. This is essential for stabilizing the gradient during training and ensuring faster convergence. Additionally, class labels were one-hot encoded to convert them into a categorical format compatible with the CNN model's softmax output layer [6]. This step involved transforming each class label into a 10-dimensional binary vector, where each index represents a distinct class.

B. Exploratory Data Analysis

The CIFAR-10 dataset contains 60,000 images divided equally across ten classes. Figure 2 shows the balanced class distribution, with 4,500 images per class in the training set,

1,000 per class in the test set, and 500 per class in the validation set. This balanced distribution prevents class imbalance, ensuring an unbiased learning process.

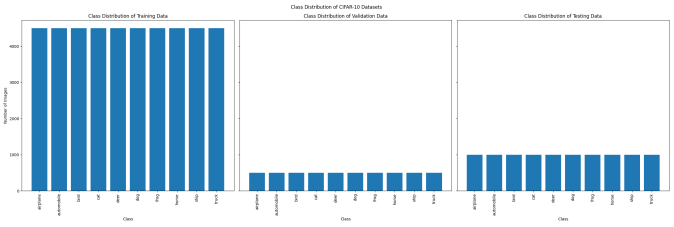

Fig. 2. Class distributions in CIFAR-10 Training, Validation, and Test Sets

Pixel intensity refers to the brightness or color information of a pixel in an image. Figure 3 displays the mean and standard deviation of pixel intensities across classes. Higher mean values indicate brighter images, while lower means suggest darker ones. Low standard deviation reflects similar intensities, while high deviation indicates greater variability. Classes exhibit similar pixel intensities and variations, posing challenges for classification, as separability is critical.

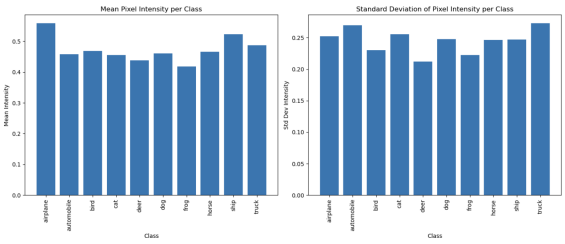

Fig. 3. Mean & Std. Dev. of pixel intensities across classes

PCA was applied to reduce the data's dimensionality, aiding in visualizing class grouping. Figure 4 shows distinct groupings, indicating separability between classes. For instance, airplanes are grouped towards the top left, trucks towards the bottom right, automobiles on the right, and frogs on the left.

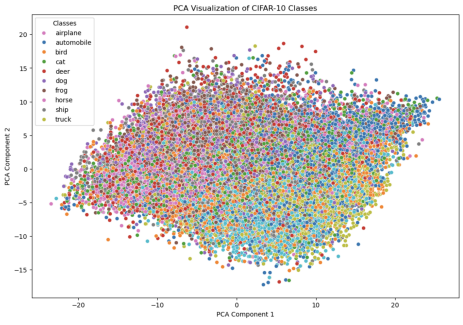

Fig. 4. Principal Component Analysis (2D Visualization)

C. Initial Model Evaluation and Motivation for CNN Implementation

To establish a baseline performance, simpler models were tested before implementing a CNN. Logistic regression and a Support Vector Classifier (SVC) with principal component analysis (PCA) were employed to assess class separability.

The logistic regression model achieved an accuracy of 40.51%, while the SVC model reported 39.95%. These results emphasized how linear and kernel-based models lack the ability to capture the complex patterns and spatial relationships in CIFAR-10 images. This limitation showed the need for a more advanced approach, motivating the implementation of a CNN to extract and learn hierarchical features.

D. CNN Model Architecture

Our CNN architecture was built using Keras's Sequential API, enabling layer-by-layer model creation. The architecture consisted of four blocks designed to extract increasingly complex features, from edges and textures to full object representations, balancing computational efficiency and performance.

Each block includes two convolutional layers for feature extraction, batch normalization to stabilize training, dropout to prevent overfitting, and a pooling layer to reduce spatial dimensions. The components of each block are detailed below:

- **Convolutional Layers**: Two blocks with 32 and 64 filters, 3×3 kernels, ReLU activation, and stride 1 to capture edges and textures.
- **Batch Normalization and Dropout**: Stabilize activations and reduce overfitting with dropout (0.25).
- **Pooling Layers**: Max pooling (2×2) reduces spatial dimensions while retaining key features.
- **Fully Connected Layers**: Flattened features pass through dense layers (512, 128 units) with ReLU and dropout (0.5).
- **Output Layer**: Dense layer with 10 softmax units for class probabilities.

E. Training Strategy

The CNN model used the Adam optimizer, which dynamically adjusts learning rates based on gradient moments [7]. Categorical cross-entropy was the loss function, ideal for multi-class classification by comparing true class distributions with predicted probabilities. Accuracy, calculated as the percentage of correct predictions, served as the evaluation metric.

Training was conducted for up to 50 epochs with a batch size of 64. An early stopping callback halted training after 5 epochs without validation loss improvement to prevent overfitting.

F. Data Augmentation

To enhance the model's generalization and increase the size of the training data, we used Keras's `ImageDataGenerator` for real-time data augmentation, applying random rotations, flips, shifts, and zooms to the CIFAR-10 images as seen in figure 5. This allowed the model to experience variations of the images without the need for extra memory to store the augmented data.

Fig. 5. Augmented images generated from a single sample

G. Evaluation and Performance Metrics

Upon training completion, the model was evaluated on the test set using accuracy (0.84), precision (0.84), recall (0.84), and F1-score (0.84), calculated with `classification_report` and `confusion_matrix` functions from `sklearn.metrics`.

Figure 6 shows training-validation accuracy trends, highlighting the model's generalization ability through the close alignment of training and validation metrics. Accuracy and loss trends indicate consistent learning, while increasing precision and recall scores reflect the model's capacity to identify true positives across classes. The stability of these metrics in final epochs confirms convergence and robustness.

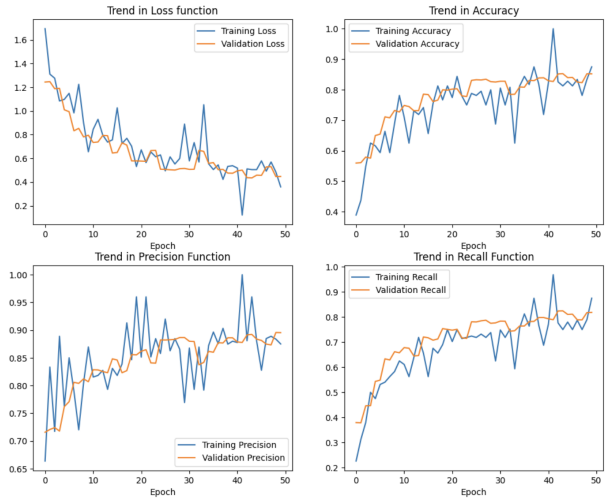

Fig. 6. Trends in training and validation metrics across epochs

Figure 7 presents a confusion matrix that provides class-specific classification insights, identifying higher misclassification rates between visually similar classes like "cats" and "dogs" [8].

Through standardizing data preprocessing, defining consistent architecture, and employing robust evaluation methods, this implementation ensured reproducibility.

Hyperparameter Settings: All hyperparameters were explicitly defined to ensure reproducibility:

- Learning Rate: Optimized to 0.001 using grid search for the Adam optimizer.
- Batch Size: Set to 64 for efficient training and stability.
- Regularization: Dropout rates of 0.25 and 0.5 balanced generalization and performance.
- Architecture: Four convolutional layers with filters of 32, 64, and 128 captured hierarchical features.

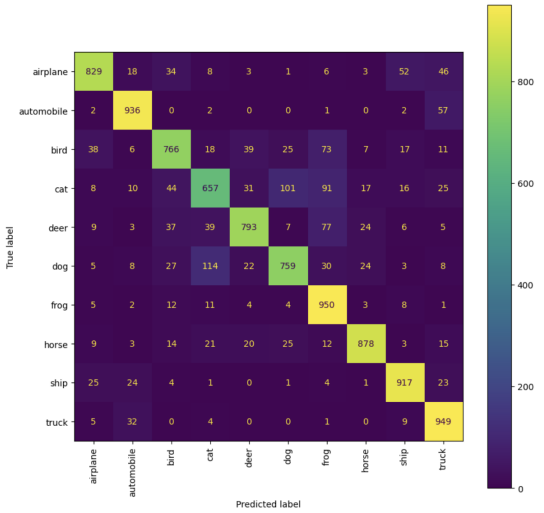

Fig. 7. Final Model Confusion Matrix

- Epochs: Limited to 50 with early stopping after 5 stagnant validation epochs.

III. Experimental Results and Discussion

This section presents the analysis of the experiments used to facilitate the development and evaluation of the CNN for the CIFAR-10 dataset. The experiments were designed to optimize the model's performance and address challenges like overfitting, generalization, and stability.

A. Goals of the Experiments

The CIFAR-10 dataset, with its diverse classes and low resolution, presents significant challenges for image classification. These experiments aimed to:

- Establish baseline performance using a simple CNN.
- Apply *regularization techniques* (e.g., dropout, batch normalization) to improve generalization.
- Evaluate the impact of *data augmentation* on robustness.
- Explore the effects of *architectural depth* and filter size variations.
- Compare *optimizers* to identify the best training approach.
- Test the performance of the final optimized model on a test set.

B. Experiment 1: Baseline Model

Setup: The baseline model included two convolutional layers with ReLU activation and max-pooling. It was trained on the raw CIFAR-10 dataset using the Adam optimizer at a 0.001 learning rate for 20 epochs.

Findings:

- Training accuracy: 56.25%, Validation accuracy: 63.94%, showing overfitting.
- Loss curves showed rapid early convergence, but divergence between training and validation losses pointed to poor generalization.

Implications: The baseline provided a starting point, highlighting the limitations of unregularized networks and the need for optimization.

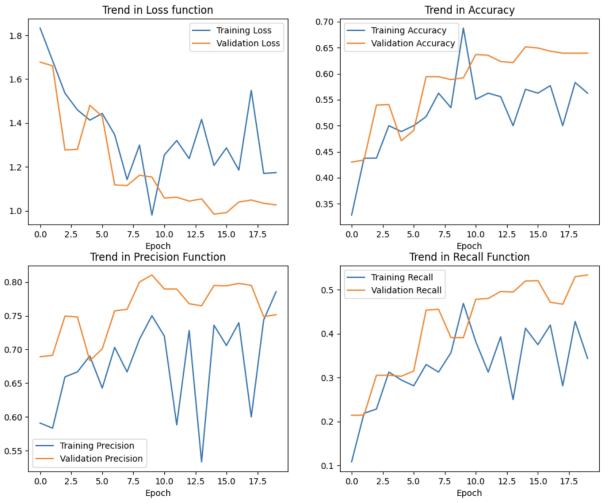

Fig. 8. Trends in training and validation metrics for baseline model

C. Experiment 2: Incorporating Regularization Techniques

Setup: Dropout layers (0.25 and 0.5 rates) were added after convolutional layers, and batch normalization was used to stabilize learning by normalizing input distributions.

Findings:

- Training accuracy: 59.38%, Validation accuracy: 76.48%, indicating reduced overfitting compared to the baseline.
- Batch normalization improved convergence and training stability, as seen in smoother loss curves.

Implications: Regularization techniques significantly improved model robustness and generalization.

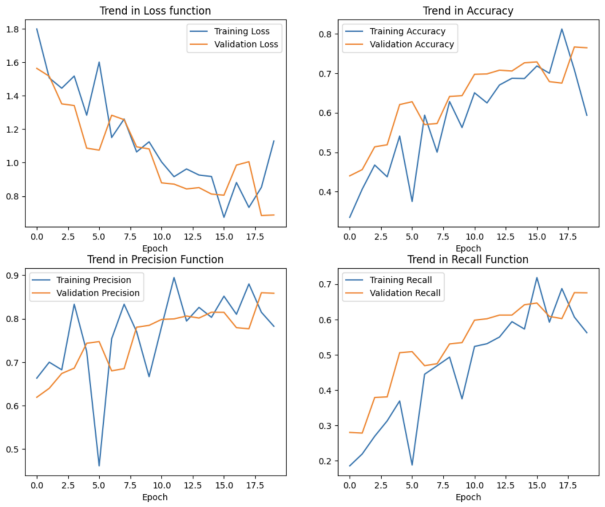

Fig. 9. Trends in training and validation metrics for Regularized model

D. Experiment 3: Impact of Data Augmentation

Setup: Data augmentation techniques, such as random flips, rotations, and zooming, were applied during training, increasing training data diversity in real-time.

Findings:

- Training accuracy: 62.45%, Validation accuracy: 78.37%, demonstrating better generalization with augmented data.
- Augmented data helped the model learn more invariant and generalized features.

Implications: Augmentation offered a simple yet effective method to boost performance without modifying the architecture.

E. Experiment 4: Architectural Variations

Setup: The architecture was deepened with additional convolutional layers using larger filters (32, 64, and 128) and more pooling layers. The effects of depth and capacity on performance were assessed.

Findings:

- Training accuracy: 90%, Validation accuracy: 83%, reflecting the benefits of a deeper architecture.
- Depth beyond four layers showed diminishing returns, with minor accuracy improvements and higher computational costs.

Implications: Deeper architectures capture complex patterns, but excessive depth risks overfitting and inefficiency, especially for lower-resolution datasets like CIFAR-10.

F. Experiment 5: Optimizer Comparison

Setup: Adam, RMSprop, and Stochastic Gradient Descent (SGD) optimizers were evaluated, with each tuned for optimal learning rate.

Findings:

- Adam: Training accuracy: 88%, Validation accuracy: 83%, showing strong performance and fast convergence.
- RMSprop: Training accuracy: 86%, Validation accuracy: 81%, with slightly higher variance.
- SGD: Training accuracy: 79%, Validation accuracy: 76%, requiring more epochs to converge.

Implications: Adam was identified as the most effective optimizer, offering a balance of speed, stability, and accuracy, while RMSprop provided competitive accuracy with higher variability.

G. Evaluation of the Final Model

Setup: The final model incorporated the best practices from earlier experiments: four convolutional layers, dropout, batch normalization, and data augmentation. It was trained using the Adam optimizer with hyperparameters fine-tuned via grid search to identify optimal combinations.

Findings:

- Training accuracy: 87.5%, Validation accuracy: 85.2%, demonstrating consistent performance across datasets.
- Class-specific metrics showed high F1-scores: 0.94 for "automobiles" and 0.90 for "ships." Misclassification

rates were higher for "cats" (F1-score: 0.69) and "dogs" (F1-score: 0.74) [9].

Model Strengths:

- Strong generalization achieved through regularization and data augmentation techniques.
- Efficient feature extraction due to an optimized architecture that effectively captured hierarchical patterns.
- Stable and fast convergence driven by the Adam optimizer.

Implications: The final model achieved a balance between performance and efficiency, offering competitive accuracy while retaining interpretability and robustness.

Model Interpretability: The interpretability of this CNN model stems from its ability to extract hierarchical features:

- Feature Identification: Early layers detect basic features like edges, textures, and color gradients, while deeper layers capture complex shapes and objects.
- Pooling Simplification: Max-pooling layers reduce spatial dimensions while retaining key features essential for classification.
- Understanding Predictions: Visualizing feature maps or saliency maps reveals image regions contributing most to predictions.
- Contribution of Features: Textures and edges of cars or specific color patterns in ships are examples of discriminative features used by the model.

H. Key Insights and Future Directions

Key Insights:

- Regularization and data augmentation are essential for improving generalization.
- Optimal depth is key, but excessive layers may not yield proportional gains.
- Optimizer selection strongly affects convergence, with Adam proving most effective.
- Misclassification in similar categories underscores the need for improved feature extraction.

Future Directions:

- Investigate advanced architectures like ResNet or EfficientNet to enhance performance [10].
- Explore transfer learning to leverage pre-trained models for faster training and better accuracy.
- Incorporate attention mechanisms to reduce inter-class ambiguities and improve fine-grained classification.
- Utilize ensemble learning to combine models for greater robustness and reliability.

IV. Conclusion

This project explored image classification for the CIFAR-10 dataset, culminating in a robust CNN achieving 83% test accuracy, demonstrating strong generalization across diverse image classes.

Key insights from the experiments:

- Dropout and batch normalization reduced overfitting and improved stability.
- Data augmentation introduced variability, enhancing generalization and helping the model learn robust features.
- Optimizing architectural depth balanced complexity and performance, with a four-layer CNN yielding optimal results.
- The Adam optimizer excelled in convergence speed and accuracy, highlighting the importance of suitable training algorithms.

Business Insights

The development of a CNN for CIFAR-10 image classification highlights its practical applications for companies:

- **Automated Image Classification:** An e-commerce business could use a similar model to classify product images into categories like "shoes" or "electronics," improving inventory management or search functionality.
- **Robust Performance:** Data augmentation and regularization ensure reliable performance, even with diverse datasets.
- **Cost Efficiency:** Optimized architectures and training strategies reduce computational costs, making solutions feasible for businesses with limited resources.
- **Operational Improvements:** Machine learning systems can streamline operations, enhance customer experiences, and support innovation.

This study demonstrates the value of systematic experimentation and evaluation. Addressing current limitations and adopting advancements in deep learning could further improve image classification performance. A website containing a post writeup and code can be found here: Website , Collab Link

References

[1] A. Krizhevsky, "Learning multiple layers of features from tiny images," *CIFAR-10 Dataset Technical Report*, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[3] H. M. ElSaid, A. Shamseldin, and M. E. Hussein, "Hybrid approach for CIFAR-10 classification using CNN and SVM," *Journal of Artificial Intelligence and Data Mining*, vol. 9, no. 2, pp. 83–89, 2021. DOI: 10.22044/jadm.2020.9590.2121.

[4] S. Wang, X. Zhang, and L. Chen, "CIFAR-10 image classification using CNNs with PCA dimensionality reduction," *IEEE Transactions on Image Processing*, vol. 29, pp. 4823–4834, June 2020. DOI: 10.1109/TIP.2020.2976183.

[5] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://tensorflow.org/.

[6] H. Liu, "A comprehensive guide to PCA for image recognition," Medium, 2023. [Online].

[7] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980.

[8] S. Liu, Y. Wang, and X. Liu, "Integrating PCA with SVM for image classification on CIFAR-10," *Springer Machine Vision Applications*, vol. 17, pp. 45–57, April 2022.

[9] A. Deshpande and S. Jangid, "Ensemble learning for CIFAR-10 classification using CNN and SVM," in *Proceedings of the 18th International Conference on Computer Vision (ICCV)*, October 2021, pp. 3124–3133. DOI: 10.1109/ICCV.2021.00218.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009, pp. 248–255.