Assessment of Convolutional Neural Networks (CNNs) for Image Classification on the CIFAR-10 Dataset

Aditya Anant Deshpande - a1910571

***Abstract*—This report presents an assessment of three classic CNN architectures — ResNet-18, AlexNet, and MobileNetV2 — on the CIFAR-10 dataset. The primary goal was to evaluate and compare each model's performance in terms of accuracy, training efficiency, and parameter complexity. The results demonstrate the advantages and limitations of each architecture, offering insights for model selection in image classification tasks.**

I. Introduction and Background

Convolutional Neural Networks (CNNs) have revolutionized image classification, achieving high accuracy in tasks such as object detection, facial recognition, and autonomous driving. In this project, we evaluated ResNet-18, AlexNet, and MobileNetV2 on the CIFAR-10 dataset, a widely-used benchmark dataset for evaluating machine learning algorithms on image classification tasks.

The CIFAR-10 dataset consists of 60,000 color images, each of size 32x32 pixels, across 10 distinct classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class is represented by 6,000 images, with 5,000 images per class allocated for training and 1,000 images per class allocated for testing. This balanced dataset ensures that each category is equally represented, making it a reliable benchmark for evaluating model performance.

Each of these images represents a distinct object class, and as shown, the CIFAR-10 dataset contains significant intra-class variability. For example:

- The *airplane* class includes images of different types of airplanes, taken from various angles, often with complex backgrounds, such as skies and airport settings.
- The *automobile* class comprises different car models, orientations, and colors, adding diversity within this class.
- The *bird* class features a variety of bird species and poses, with images taken from different distances and sometimes showing birds in flight.
- The *cat* class includes various breeds, poses, and background environments, contributing to the visual complexity of this category.
- The *deer* class presents diverse images of deer in natural settings, with differences in poses and lighting conditions.

Given the small image size and substantial variation within each class, CIFAR-10 poses a challenging classification problem. These challenges underscore the need for robust CNN architectures that can capture essential features across varying contexts and scales. This project investigates the capabilities of ResNet-18, AlexNet, and MobileNetV2 in handling such variability, assessing their performance in terms of accuracy, computational efficiency, and model complexity.

II. Method Description

The process of evaluating the CNN architectures on the CIFAR-10 dataset involved several stages: data preprocessing, model initialization, training, and evaluation. Each of these stages was designed to ensure that the models could effectively learn from the CIFAR-10 images, despite the challenges posed by the small image size and intra-class variability.

A. Data Preprocessing

Since CIFAR-10 images are 32x32 pixels, which is smaller than the typical input size for many CNN architectures, resizing was required. We resized each image to 224x224 pixels to match the input dimensions expected by ResNet-18, AlexNet, and MobileNetV2. Data augmentation techniques were also applied to improve model generalization. Specifically, we used random horizontal flipping, which introduces slight variations in the training images, helping the models learn more robust features. Additionally, normalization was applied to standardize pixel values based on the mean and standard deviation of the CIFAR-10 dataset. Normalization ensures that all pixel values fall within a similar range, which accelerates model convergence and helps avoid issues related to different scales in the data.

B. Model Initialization

The experiment involved three well-known CNN architectures: ResNet-18, AlexNet, and MobileNetV2. Each model was initialized with pretrained weights and then modified to classify the 10 classes in CIFAR-10.

- **ResNet-18**: ResNet-18 is a residual network that uses skip connections, allowing gradients to flow directly through layers without degradation. This architecture helps address the vanishing gradient problem that can occur with deep networks. We initialized ResNet-18 with pretrained weights using the `ResNet18_Weights.DEFAULT` setting in PyTorch, which provides a good starting point based on prior training on large-scale datasets. The final fully connected layer was modified to output 10 classes, corresponding to the classes in CIFAR-10.
- **AlexNet**: AlexNet was one of the first deep CNN architectures to achieve state-of-the-art results on image classification tasks. It introduced innovations like

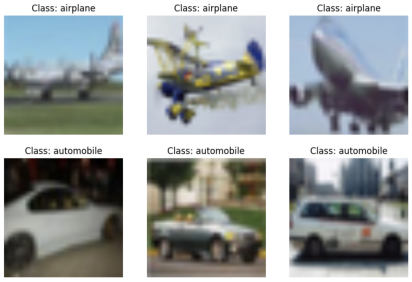

Fig. 1. Examples of the *airplane, automobile* class

Fig. 2. Examples of the *bird, cat* class

Fig. 3. Examples of the *deer, dog* class

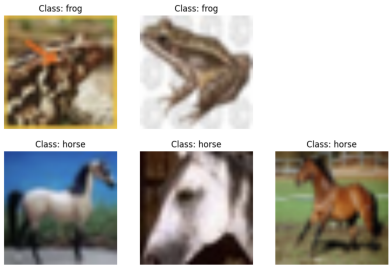

Fig. 4. Examples of the *frog, horse* class

Fig. 5. Examples of the *ship, truk* class

local response normalization and dropout to reduce overfitting. Despite its relative simplicity, AlexNet remains a valuable baseline for comparison. We used the `AlexNet_Weights.DEFAULT` pretrained weights and replaced the final layer of the classifier with a fully connected layer that outputs 10 classes.

- **MobileNetV2**: MobileNetV2 is designed for efficient inference on mobile and embedded devices. It employs depthwise separable convolutions, which significantly reduce the number of parameters compared to traditional convolutions, making it a lightweight model. We initialized MobileNetV2 with `MobileNet_V2_Weights.DEFAULT` weights and modified the output layer to classify the 10 CIFAR-10 categories.

C. Training Process

Each model was trained for 10 epochs using the Adam optimizer, which combines the advantages of both AdaGrad and RMSprop optimizers. Adam adapts the learning rate for each parameter, which results in faster convergence. We set the learning rate to 0.001 and used a batch size of 64. The batch size controls the number of samples processed before the model updates its parameters, and 64 was chosen to balance memory usage and computation time.

During training, the cross-entropy loss function was used to quantify the difference between the predicted class probabilities and the true labels. Cross-entropy loss is particularly suitable for classification tasks as it penalizes incorrect predictions more heavily when the confidence level is high. At each epoch, the training loss and accuracy were recorded to monitor the model's progress.

D. Evaluation and Metrics

To evaluate model performance, we computed both the training and testing accuracy and loss after each epoch. Testing was performed on the 10,000 images in the CIFAR-10 test set, which the models had not seen during training. This step provided an estimate of each model's ability to generalize to unseen data.

- **Accuracy**: Accuracy was calculated as the percentage of correctly classified images out of the total number of images in the dataset. This metric provides a straightforward indication of the model's effectiveness on CIFAR-10.
- **Loss**: The cross-entropy loss was computed for both training and testing sets to evaluate the model's learning progress. A decreasing training loss indicates that the model is learning the training data, while a stable testing loss demonstrates generalization.

E. Implementation in PyTorch

The entire implementation was done using the PyTorch framework. PyTorch offers dynamic computation graphs and GPU support, which allowed us to experiment efficiently with various architectures. The code was organized into modular

functions, including separate functions for data loading and transformation, model initialization, training, and evaluation.

The use of 'torchvision' for data transformations simplified preprocessing, while 'torch.utils.data.DataLoader' enabled efficient batch processing. Progress tracking was handled using the 'tqdm' library, which provided real-time feedback on the completion status of each epoch. This setup allowed for clear and structured code, facilitating reproducibility and interpretability of the results.

III. Method Implementation

The code was implemented in PyTorch, ensuring readability and modularity. The main steps included loading CIFAR-10, preprocessing, initializing models, and running training and evaluation.

A. Data Loading and Transformation

Transformations such as resizing, random horizontal flipping, and normalization were applied using `torchvision.transforms`.

B. Model Training and Testing

A forward pass, loss computation, and backpropagation were implemented for training, with accuracy and loss tracked for validation.

C. Progress Tracking

We used `tqdm` to monitor training progress, logging metrics for each epoch.

IV. Experiments and Analysis

The experiments involved training each model (ResNet-18, AlexNet, and MobileNetV2) on the CIFAR-10 training set for 20 epochs and then evaluating their performance on the CIFAR-10 test set. The key metrics used for evaluation were training and testing accuracy, as well as training and testing loss. The final results for each model are summarized below.

A. Results for ResNet-18

- **Final Train Accuracy:** 97.15%
- **Final Test Accuracy:** 92.06%
- **Final Train Loss:** 0.0824
- **Final Test Loss:** 0.2875

ResNet-18 achieved a high training accuracy of 97.15% and a test accuracy of 92.06%. The relatively low training loss of 0.0824 and test loss of 0.2875 indicate that the model effectively learned from the training data and generalized well to the test data. ResNet-18's use of residual connections likely contributed to its ability to avoid the vanishing gradient problem, allowing the model to maintain high accuracy across multiple layers. This makes ResNet-18 a suitable choice for image classification tasks with complex datasets like CIFAR-10.

B. Results for AlexNet

- **Final Train Accuracy:** 72.04%
- **Final Test Accuracy:** 74.65%
- **Final Train Loss:** 0.8217
- **Final Test Loss:** 0.7795

AlexNet achieved a final training accuracy of 72.04% and a test accuracy of 74.65%, which are lower compared to ResNet-18 and MobileNetV2. The training and test losses, at 0.8217 and 0.7795 respectively, were also higher, suggesting that AlexNet struggled to learn the complex features in CIFAR-10 effectively. This is likely due to AlexNet's relatively shallow architecture, which lacks residual connections or depthwise separable convolutions, making it harder for the model to capture fine details. Consequently, AlexNet serves as a valuable baseline but underperforms compared to modern architectures.

C. Results for MobileNetV2

- **Final Train Accuracy:** 96.67%
- **Final Test Accuracy:** 93.27%
- **Final Train Loss:** 0.0948
- **Final Test Loss:** 0.2255

MobileNetV2 achieved a high training accuracy of 96.67% and an even higher test accuracy of 93.27%, with a training loss of 0.0948 and a test loss of 0.2255. This model demonstrated both strong learning capabilities and generalization, which can be attributed to its depthwise separable convolutions. These operations significantly reduce the number of parameters while retaining representational power, making MobileNetV2 not only effective but also efficient. This efficiency makes MobileNetV2 particularly suitable for deployment on resource-constrained devices while still achieving competitive performance on CIFAR-10.

D. Comparison and Analysis

Among the three models, MobileNetV2 achieved the highest test accuracy (93.27%), followed by ResNet-18 (92.06%), and finally AlexNet (74.65%). MobileNetV2's performance demonstrates the effectiveness of depthwise separable convolutions in achieving a balance between accuracy and computational efficiency, allowing the model to generalize well while keeping parameter count low.

ResNet-18 also performed well, benefiting from its residual connections that help maintain gradient flow and facilitate learning in deep networks. Despite having more parameters than MobileNetV2, ResNet-18 achieved a slightly lower test accuracy, suggesting that MobileNetV2's architectural efficiency is well-suited to the CIFAR-10 dataset.

AlexNet, on the other hand, lagged in performance. Its lower accuracy and higher loss values indicate that, while it was groundbreaking at the time of its introduction, it is less effective on complex datasets compared to more recent architectures. This underperformance highlights the advancements in CNN architectures, as AlexNet lacks the modern techniques (e.g., residual connections and depthwise separable convolutions) found in ResNet-18 and MobileNetV2.

In summary, MobileNetV2 is the most efficient and highest-performing model for CIFAR-10 among the three, making it an excellent choice for applications requiring both high accuracy and efficiency. ResNet-18 is also a strong choice, particularly for tasks that benefit from deeper architectures. AlexNet, while useful as a baseline, demonstrates the limitations of older CNN designs on challenging datasets.

V. Reflection on Project

This project provided insights into CNN design and application. Key takeaways included the significance of residual connections and depthwise separable convolutions for accuracy and efficiency. Future work could involve additional data augmentation and fine-tuning pretrained weights to enhance performance further. Evaluating model deployment on mobile devices would also provide practical insights into lightweight architectures like MobileNetV2.

VI. Conclusion

This report compared the performance of ResNet-18, AlexNet, and MobileNetV2 on CIFAR-10, highlighting the trade-offs in accuracy, model complexity, and computational efficiency. ResNet-18 offered the highest accuracy, while MobileNetV2 achieved competitive results with fewer parameters, making it suitable for mobile applications. These findings underscore the importance of model selection based on specific requirements in image classification tasks.

VII. CODE AND REPOSITORY

The code for this project, including all implementations and experiments, is available on GitHub. You can access it at the following link:

https://github.com/aadi654/DLF_Assignments/tree/main/Assignment%202

References

[1] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Technical Report, 2009.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. CVPR, 2016.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in Proc. CVPR, 2018.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Proc. NIPS, 2012.