CIFAR-100 Image Classification Challenge: A Comparative Analysis of CNN Architectures and Transfer Learning

AI Disclosure:

For this assignment, I incorporated AI assistance, primarily through the Gemini platform, to support various aspects of the project. Gemini's contributions included generating code snippets, particularly for advanced data augmentation techniques like Mixup and CutMix, aiding in debugging error messages encountered during development, and providing assistance with the structure and formatting of this report. Regarding code authorship, the fundamental model architecture modifications for CIFAR-100 (specifically, adapting ResNet18), data loading procedures, the training and validation loops, and hyperparameter configurations were all developed independently by me. Gemini assisted in refining the implementation of training loop modifications related to the integration of Mixup and Cutmix. Finally, it is important to note that portions of this report were composed with the support of Gemini and the WandB Reports UI. To ensure transparency and clarity, the codebase is thoroughly commented, detailing the function of each component.

Abstract:

This report details the implementation and evaluation of three distinct deep learning approaches for image classification on the CIFAR-100 dataset. The challenge was structured in three parts: establishing a baseline with a simple Convolutional Neural Network (CNN), exploring more sophisticated CNN architectures, specifically ResNet-18, and leveraging transfer learning through a pre-trained ResNet-18 model. Each phase involved careful consideration of architectural design, hyperparameter tuning, regularization, and data augmentation strategies. The results demonstrate a clear progression in performance, highlighting the benefits of sophisticated architectures and transfer learning.

1. Introduction:

The CIFAR-100 dataset presents a challenging image classification task with 100 distinct classes, demanding robust and efficient deep learning models. This study aimed to investigate the impact of architectural complexity and transfer learning on classification accuracy. I began by establishing a baseline with a simple CNN, subsequently explored the performance of ResNet-18, and concluded with fine-tuning a pre-trained ResNet-18 model.

2. Methodology:

2.1. Part 1: Simple CNN Baseline:

In the first phase, a simple CNN architecture was manually defined to establish a baseline performance. This architecture consisted of two convolutional layers, ReLU activations, max

pooling, and two fully connected layers. The design focused on simplicity, employing 3x3 kernel sizes and 2x2 max pooling to capture local features and reduce spatial dimensions. ReLU activations were chosen for their computational efficiency and ability to mitigate vanishing gradients. A high learning rate of 0.1 was initially used with the SGD optimizer, and the model was trained for 5 epochs. Basic data augmentation, including random cropping and horizontal flipping, was applied. Notably, no explicit regularization techniques were implemented to observe the model's unmitigated behavior.

2.2. Part 2: ResNet-18 Implementation:

In the second phase, the strategy shifted from a manually defined, simple CNN to leveraging the pre-defined ResNet-18 architecture from torchvision. This transition was motivated by the need to explore more sophisticated models capable of capturing complex features inherent in the CIFAR-100 dataset. ResNet-18, known for its deep residual learning framework, was chosen to mitigate the vanishing gradient problem and improve feature extraction. To tailor ResNet-18 to the CIFAR-100 task, the final fully connected layer was modified to output 100 classes. Additionally, dropout was introduced to the fully connected layers to combat overfitting, a concern given the increased model complexity. The optimization strategy was also refined, moving from SGD to Adam with a lower learning rate of 0.001, which is more appropriate for deeper networks. Furthermore, weight decay was implemented as an additional regularization technique. The data augmentation strategy remained consistent with Part 1, focusing on basic transformations. This part aimed to understand the performance gains achieved by a more advanced architecture and improved training methodologies.

2.3. Part 3: Transfer Learning with Pretrained ResNet-18:

The final phase introduced transfer learning, utilizing a pre-trained ResNet-18 model, to further enhance classification accuracy. This approach capitalized on features learned from the extensive ImageNet dataset, providing a strong initialization for the model. Compared to Part 2, significant architectural and training modifications were implemented. The first convolutional layer was adjusted to accommodate the CIFAR-100 image size, and the max pooling layer was removed to preserve spatial information. Dropout was applied after each residual block to provide robust regularization throughout the network. The final fully connected layer was further modified for fine-tuning. The optimization strategy shifted to AdamW with an even lower learning rate of 0.0005, recognizing the need for careful fine-tuning to preserve pre-trained weights. A ReduceLROnPlateau learning rate scheduler was introduced to dynamically adjust the learning rate based on validation loss, and early stopping was implemented to prevent overfitting. To improve generalization, the data augmentation strategy was significantly enhanced, incorporating random rotations and color jitter. CIFAR-100 normalization statistics were used to improve data normalization. These modifications represented a comprehensive effort to leverage prior knowledge, employ robust regularization, and optimize the training process. Several other

hyperparameter tuning and data augmentation techniques were tested before achieving the best score, which is detailed in the appendix.

2.4 Hyperparameter Summary

Hyperparameter	Part 1 (Simple CNN)	Part 2 (Reset-18)	Part 2 (Pre-trained Reset-18)
Learning rate	0.1	0.001	0.0005
Optimizer	SGD	Adam	AdamW
Batch Size	512	512	512
Epochs	5	20	60
Weight Decay	None	1e-4	5e-4
Dropout Rate	None	0.5 (FC Layers)	0.15 (After Each Block)
Learning Rate Scheduler	Cosine Annealing	Cosine Annealing	ReduceLROnPlateau
Early Stopping	No	No	Yes (Patience = 10)

3. Results and Discussion:

The progression from Part 1 to Part 3 demonstrated a clear improvement in classification accuracy, which is also supported by the graphs. Part 1, represented by the line "playful-salad-5" on the graphs, with its simple CNN architecture, served as a baseline, revealing the limitations of basic models on complex datasets. As shown in the validation loss graph, Part 1 has the highest validation loss and the lowest validation accuracy. The training loss and training accuracy graphs further show that Part 1 had the lowest training accuracy. Part 2, utilizing ResNet-18 and represented by the line "gentle-pond-4", showcased the benefits of deeper architectures and improved optimization techniques. Part 2 shows a significant improvement in all the graphs, with a lower validation loss and higher validation and training accuracy. Part 3, represented by the line "ruby-glade-37", with transfer learning and extensive fine-tuning, achieved the highest performance, highlighting the effectiveness of leveraging pre-trained features and robust training strategies. Part 3 obtained the lowest validation loss and the highest validation and training accuracy. The learning rate graph shows that part 3 has the lowest learning rate. The out-of-distribution dataset also showed significant improvements from part 1 to part 3. Quantitatively, the leaderboard results also illustrate this trend: Part 1 achieved a score of 0.19084, Part 2 improved to 0.20359, and Part 3 reached 0.54465 (position #17 on the public

leaderboard on Kaggle). The python script that produced the best score of 0.54465 is named `starter_code_part3_e8.py`. The interactive experiment summary can be found at this link and in the appendix.

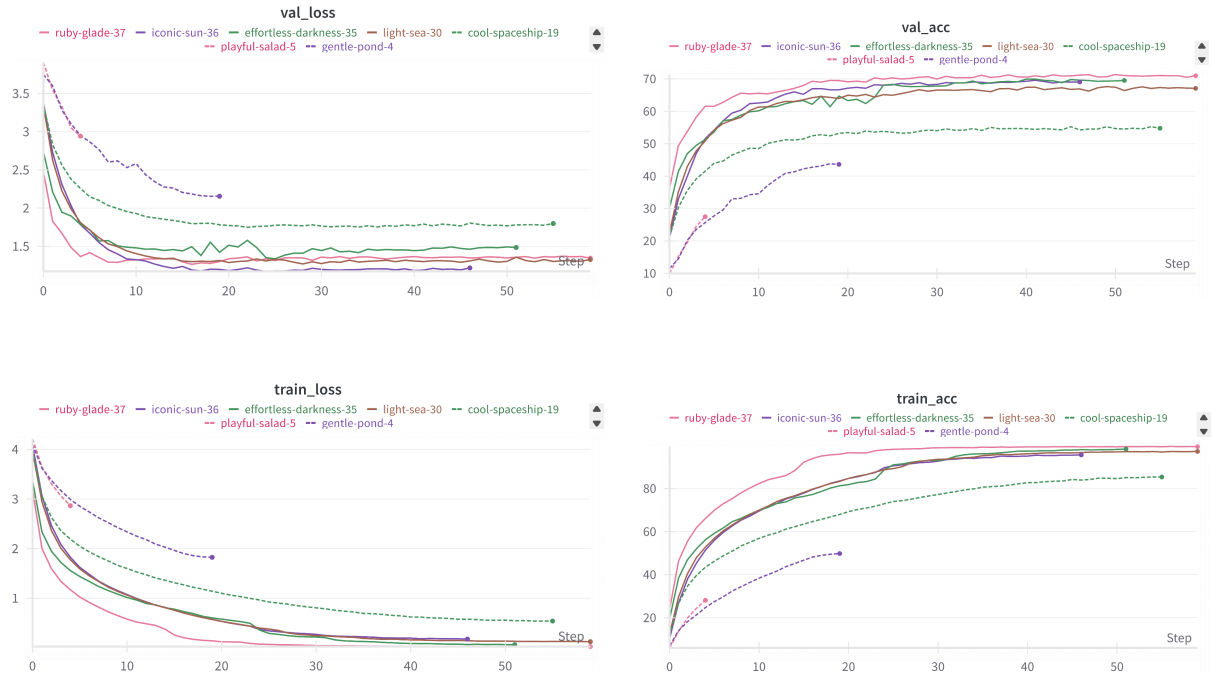

Figure 1. Experiment Tracking Summary from WandB

Figure 2. Kaggle Public Leaderboard Position for Part 1

Figure 2. Kaggle Public Leaderboard Position for Part 2

Figure 2. Kaggle Public Leaderboard Position for Best Model (Part 3)

4. Conclusion:

This study systematically investigated the impact of architectural complexity and transfer learning on image classification performance using the CIFAR-100 dataset. The progression from a simple baseline CNN to a fine-tuned pretrained ResNet-18 model demonstrated a clear improvement in classification accuracy and a corresponding decrease in validation loss, as evidenced by the graphs. The initial CNN, while establishing a foundational understanding, proved inadequate for the task's complexity, yielding the lowest performance. The adoption of ResNet-18 marked a significant advancement, highlighting the benefits of deeper architectures and refined optimization techniques in capturing intricate features. However, the transfer learning approach, leveraging pre-trained features and extensive fine-tuning, achieved the most substantial results, underscoring the effectiveness of knowledge transfer and robust training strategies. Quantitatively, this progression was reflected in the leaderboard scores, which increased from 0.19084 (Part 1) to 0.20359 (Part 2) and ultimately reached 0.54465 (Part 3).

Overall, the study confirms that deeper architectures, transfer learning, and robust training methodologies significantly enhance image classification accuracy on complex datasets like CIFAR-100. Future research could explore even more advanced architectures and fine-tuning techniques to push the boundaries of performance further.

5. Bibliography:

Akamaster. “Akamaster/Pytorch_resnet_cifar10: Proper Implementation of Resnet-S for CIFAR10/100 in Pytorch That Matches Description of the Original Paper.” *GitHub*, github.com/akamaster/pytorch_resnet_cifar10. Accessed 30 Mar. 2025.

He, Kaiming, et al. “Deep Residual Learning for Image Recognition.” *arXiv.Org*, 10 Dec. 2015, arxiv.org/abs/1512.03385.

6. Appendix:

6.1 Interactive Experiment Tracking Link

https://wandb.ai/sins-boston-university/-sp25-ds542-challenge/reports/Appendix-Graphs---VmlldzoxMjA1NDc0NQ?accessToken=6kkehjt463xyjmao8fzascwlhck2jk9r8pjj54815xw2a3p42vhjtr8udkq8gx6c

6.2 Additional Techniques in Part 3:

In Part 3, various other fine-tuning techniques and approaches were implemented with the pre-trained ResNet-18 to incorporate several advanced techniques aimed at maximizing performance. However, they did not beat the maximum score of 0.54465. Firstly, the ResNet-18 architecture itself underwent careful modification to align it more effectively with the characteristics of the CIFAR-100 dataset. This involved replacing the initial convolutional layer with one using a smaller 3x3 kernel and a stride of 1, allowing the model to capture finer-grained details within the relatively small images. Additionally, the max pooling layer was removed to preserve spatial resolution, a crucial factor for maintaining information in compact images. To combat overfitting, dropout layers were strategically inserted after each residual block, with the dropout rate meticulously tuned to 0.15 to strike a balance between regularization strength and the retention of valuable feature information.

Beyond architectural adjustments, the training configuration was also optimized. A reduced learning rate of 0.0001 was employed in conjunction with the AdamW optimizer and a weight decay of 5e-4. This combination facilitated a more precise fine-tuning of the pre-trained model, mitigating the risk of overfitting and ensuring stable convergence. Furthermore, the learning rate was dynamically adapted during training using the ReduceLROnPlateau scheduler. This adaptive approach allowed the learning rate to decrease when the validation loss plateaued, enabling the model to escape local minima and refine its weights more effectively. To further prevent overfitting and conserve computational resources, early stopping was implemented, monitoring the validation loss and halting training after 10 epochs if no improvement was observed.

Finally, Part 3 leveraged a suite of data augmentation techniques to enhance the model's generalization capabilities. Standard techniques such as random horizontal flips, random rotations (up to 15 degrees), and color jitter were applied to the training data, increasing its diversity and robustness. The code snippets also indicate the potential inclusion of MixUp and CutMix augmentation methods, which can further improve performance by creating new training samples through the combination of existing ones.

Collectively, these architectural refinements, optimized training strategies, and enhanced data augmentation techniques were helpful to understand the various hyperparameters to tune to eventually get the best mode with 0.54465 in Part 3. They underscore the importance of a holistic

approach to model development, where careful consideration of each component plays a crucial role in maximizing performance.