ASSIGNMENT 2 DEEP LEARNING FUNDAMENTALS
CNN`S FOR IMAGE CLASSIFICATION

Deekshitha Goud Podeti
University of Adelaide

Abstract

This project aims to explore the effectiveness of Convolutional Neural Networks (CNNs) for image classification tasks. We specifically focus on three popular CNN architecture: RestNet18, AlexNet and MobileNetV2.

The project involves training and evaluating these models on the CIFAR-10 dataset, a benchmark dataset for image classification. We experimented with different configurations and hyperparameters to optimize the performance of each model.

Through our experiments, we analyse the strengths and weaknesses of each architecture. We also investigate the impact of different hyperparameters and training techniques on the overall performance.

The goal of this project is to provide insights into the selection and optimization of CNN architectures for image classification tasks. By understanding the trade-offs between accuracy, computational efficiency, and model complexity, we can make informed decisions when designing and deploying CNN-based solutions.

1.Introduction

[1]. The Challenge of Image Classification:
Image classification the task of assigning a label or category to an image, is a fundamental problem in computer vision. It has numerous applications from facial recognition to medical image analysis. Traditional methods often struggled with complex visual patterns, leading to the rise CNN as a powerful tool for this task.
[2]. CNNs a powerful tool for image Analysis:
CNNs are inspired by the human visual system, capable of learning hierarchical representations of visual data. They excel at task like image classification, object detection, and image segmentation. By employing convolutional and pooling layers, CNNs can effectively extract meaningful features from images.
[3]. The CIFAR-10 Dataset: A Benchmark for Image Classification:
To evaluate the performance of different CNN architectures, we utilize the CIFAR-10 dataset. This dataset comprises 60.000 32*32 pixel colour images, divided into 10 classes. It serves as a common benchmark for testing and comparing image classification models.
[4]. A Trio of Prominent Architectures:
This project delves into three popular CNN architectures: ResNet-18, AlexNet, and MobileNetV2. Each model represents a unique approach to tackling image classification challenges, particularly in terms of depth, parameter efficiency, and computational cost.

1.1 Architecture of RestNet18:

ResNet-18: Known for its impressive depth (18 layers), ResNet-18 incorporates "skip connections" that allow information to flow directly between layers, mitigating a common problem in deep networks called the vanishing gradient. Imagine information fading as it travels through many layers, making training difficult. Skip connections act like shortcuts, ensuring earlier layers' knowledge reaches the final output effectively.

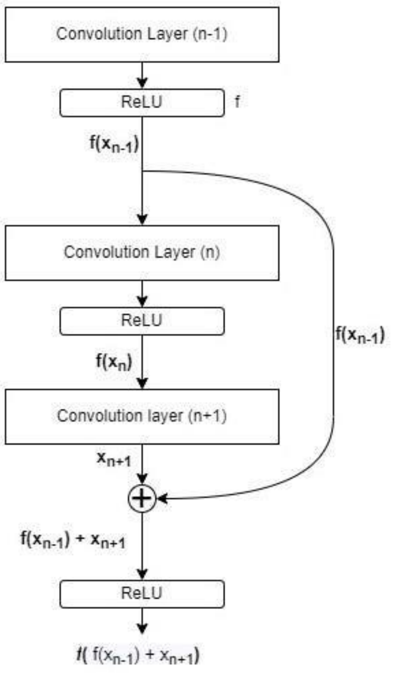


Fig 1: Shows the **RestNet18**

1.2 Architecture of AlexNet:

A pioneering architecture, AlexNet paved the way for the success of deep CNNs. Its design balances depth (8 layers) with relative simplicity, achieving impressive results while remaining computationally less demanding compared to much deeper models. This makes AlexNet a valuable reference point for comparison.

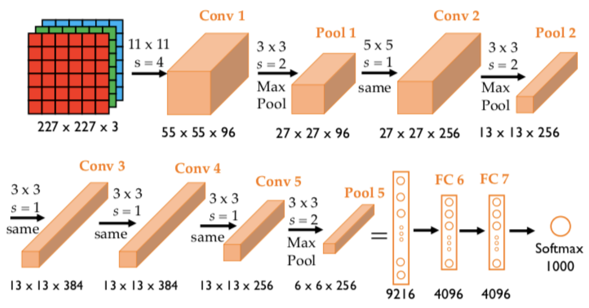


Fig2: Shows Architecture of AlexNet

1.3 Architecture of MobileNetV2:

MobileNetV2: Designed with efficiency in mind, MobileNetV2 is optimized for deployment on mobile devices and resource-constrained environments. It utilizes depthwise separable convolutions, a technique that breaks down standard convolutions into two steps, significantly reducing the number of parameters and computations needed. While maintaining a smaller size, MobileNetV2 strives to achieve accuracy comparable to its more complex counterparts.

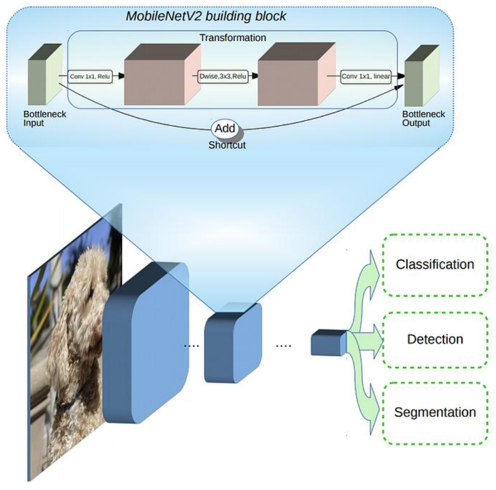


Fig 3: Shows Architecture of MobileNetV2

2. Method Description:

In this project, three deep learning models—ResNet18, AlexNet, and MobileNetV2—were adapted to tackle the image classification problem on CIFAR-10. Each model had unique modifications to fit CIFAR-10's requirements, while a hyperparameter tuning approach was applied to optimize their training and performance.

2.1 Models and Customizations:

- ResNet18: ResNet18 is a residual network known for its skip connections, which help avoid vanishing gradient issues in deep layers. However, it was initially designed for larger input images, so we customized its initial layers to handle CIFAR-10's smaller, 32x32 images. The first convolutional layer was modified to use a 3x3 kernel (instead of the usual 7x7) with a stride of 1, while the initial max-pooling layer was removed, allowing the network to capture more detailed patterns in CIFAR-10 images.
- **AlexNet**: As one of the earlier convolutional neural networks, AlexNet has a straightforward structure of stacked convolutional layers followed by fully connected layers. To reduce the risk of overfitting, especially given CIFAR-10's relatively small size, we adjusted the dropout layers. By experimenting with different dropout rates (e.g., 0.3 and 0.5), we found a balance that regularizes the network, ensuring it generalizes well without discarding too much information.
- **MobileNetV2**: MobileNetV2 is designed for efficiency,using depthwise separable convolutions to reduce computational complexity. Like ResNet18, it was adjusted to suit CIFAR-10's input size. We modified its initial layers and replaced the final classifier layer to match CIFAR-10's 10 classes, ensuring the model could perform well within the dataset's constraints while remaining computationally light.

2.2 **Hyperparameter Configurations:**

Given the varied structure of these models, we used a grid search to test different hyperparameters. This allowed us to explore combinations of settings for each model, ultimately identifying the best configuration for CIFAR-10. Here's what we adjusted:

- **Optimizers**: Both SGD (Stochastic Gradient Descent) with momentum and Adam optimizers were tested. SGD, often preferred for its stability in convergence, was combined with momentum to improve updates, while Adam's adaptive learning rates offered an alternative for comparison.
- **Learning Rates**: Two learning rates, 0.001 and 0.01, were selected to explore how quickly each

model adapted to the data. Lower learning rates generally encourage steady convergence, while higher rates speed up training but risk overshooting the optimal point.

- **Weight Decay**: Regularization through weight decay (values of 5e-4 and 1e-4) was applied to prevent overfitting by penalizing large weights. This was especially important for AlexNet, which has dense layers with many parameters, but was also beneficial across all models.
- **Augmentation Techniques**: Finally, data augmentation was used to diversify the training data and improve generalization. Two levels were applied:
  Basic augmentation involved random cropping and horizontal flipping, introducing slight variations without drastic changes.
  Advanced augmentation added color jittering (adjusting brightness, contrast, saturation, and hue) to simulate varied lighting and conditions. These augmentations pushed the models to be more flexible and robust in recognizing patterns.

  By exploring these configurations, we could tailor each model to extract the best performance from CIFAR-10, enhancing both training effectiveness and final accuracy on unseen data.

3. Method Implementation

This section outlines how the CIFAR-10 dataset was prepared, and how each model was implemented, trained, and evaluated.

3.1 Data Loading and Preprocessing

To work efficiently with CIFAR-10, we used the Py Torch Data Loader to handle both loading and preprocessing in a streamlined way. The Data Loader simplifies the training process by automatically shuffling the dataset, dividing it into batches, and iterating over it during training. This approach ensures each batch is unique and representative, which helps the model generalize better.
To prepare the images for training, we applied basic data augmentations:

- **Random Cropping**: We randomly crop each image, giving the model slightly different views of each sample. This helps prevent the model from memorizing specific parts of each image, making it more adaptable to real-world variations.
- **Horizontal Flipping**: Images are randomly flipped left-to-right, adding another level of variability. This simple transformation forces the model to learn features that are not dependent on orientation.

Together, these augmentations increase the diversity of the training data, making it less likely for the model to overfit on specific image patterns.

3.2 Training Process:

The training process follows a structured loop, iterating over the dataset multiple times (epochs) to improve model performance. Here's a step-by-step breakdown of each key stage:

1. **Forward Pass**: At each iteration, a batch of images is passed through the model. The model processes these images and outputs predictions for each one. In essence, the forward pass is where the model "sees" the data.
2. **Loss Calculation**: Once predictions are generated, we calculate how far off these predictions are from the actual labels using **cross-entropy loss**. This metric gives a number representing how "wrong" the predictions are, guiding the adjustments to be made to the model.
3. **Backward Pass**: This is the stage where learning happens. We backpropagate the loss through the model, allowing each layer to update its weights based on the error. By adjusting weights in this manner, the model incrementally learns to make more accurate predictions.
4. **Optimizer Step**: Finally, we use an optimizer, either SGD or Adam depending on the experiment, to update the model's weights. Each update nudges the model in the direction that reduces the loss, effectively helping it learn patterns in the data.

This process repeats for each batch, steadily refining the model's understanding with each epoch.

3.2 Evaluation Metrics

To measure how well each model is learning and performing on CIFAR-10, we focus on two main metrics:

- **Accuracy**: Accuracy provides a clear picture of how well the model is classifying images into the correct categories. It's straightforward and gives a good overall performance snapshot.

- **Formula**:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100$$

- **Cross-Entropy Loss**: While accuracy tells us the percentage of correct predictions, cross-entropy loss reflects the confidence of these predictions. Lower loss generally indicates that the model's predictions are becoming more accurate and confident over time.

- Monitoring both accuracy and loss throughout training and validation gives us a balanced view of model learning and generalization.

$$L = -\sum_{i=1}^{C} y_i \cdot \log(p_i)$$

4 Experiments and analysis:

In this section, we outline the experimental design used to evaluate the performance of three deep learning models (ResNet18, AlexNet, and MobileNetV2) on the CIFAR-10 dataset. The primary aim was to find the best model configuration that achieves high accuracy and generalizes well to unseen data.

4.1 Training and Validation:

To maintain consistency and comparability across all models, each was trained for 10 epochs using a batch size of 128. The batch size of 128 was selected to balance computational efficiency with smooth learning progress, allowing for meaningful gradient updates without overwhelming memory resources. This choice supports quicker training cycles while still allowing the model to learn effectively.

To optimize the learning rate throughout training, we used a CosineAnnealingLR scheduler. This scheduler begins with a relatively high learning rate that gradually decreases over time, following a cosine curve. This approach is beneficial because it allows the model to make broader adjustments early in training (when it's learning basic features) and gradually fine-tunes its parameters as training progresses. This smooth decay helps avoid the risk of getting "stuck" in local minima, a common issue in training deep models, especially on smaller datasets like CIFAR-10.

Each epoch’s training was followed by validation accuracy evaluation. Calculating validation accuracy after each epoch provided insights into the model's generalization to unseen data, allowing us to identify the configurations likely to perform best on the test set. Validation accuracy trends were recorded, showing how each model improved or leveled out over time.

The Below are graphs I got Before Hyper-Parameter Tunning:

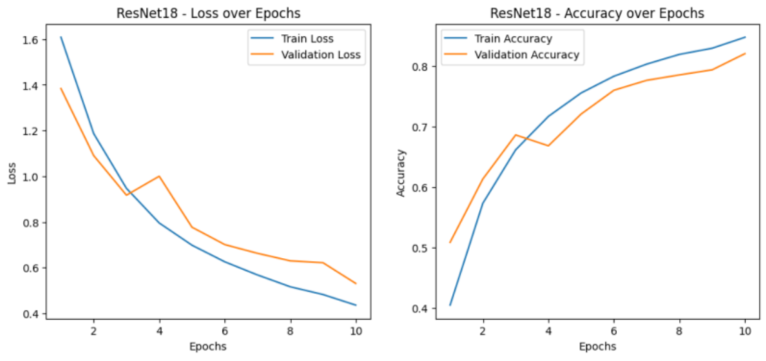


Fig 4 : Shows the Output of ResNet18 and how accuracy changed across epochs

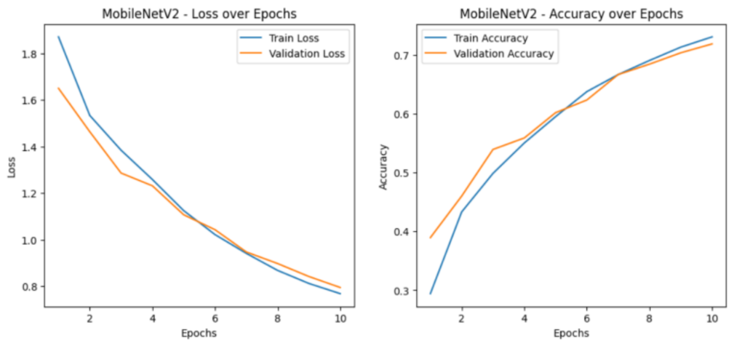


Fig 5 : Shows the output of MobileNetV2 and how accuracy changed across epochs.

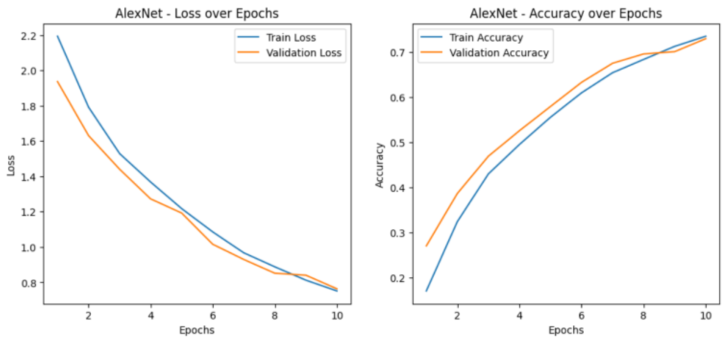


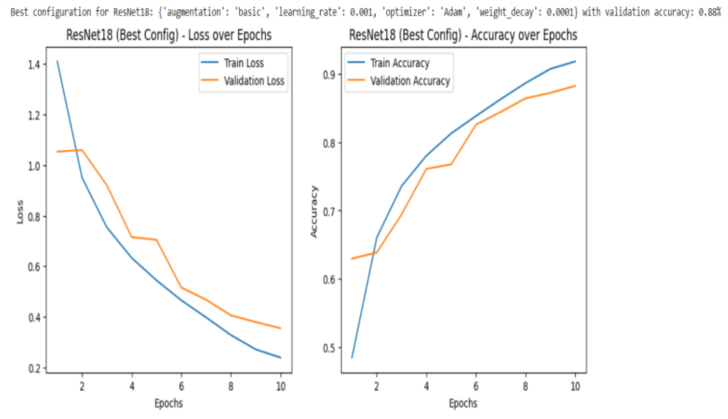
Fig 6: Shows the output Of AlexNet and how accuracy changed across epochs

4.2 Hyperparameter Tuning

Optimizing model performance required fine-tuning various hyperparameters, which we conducted through a **grid search**. For each model, we tested multiple combinations of learning rates, weight decay values, optimizers, and data augmentation strategies. This thorough approach helped identify the optimal settings for each model's architecture.

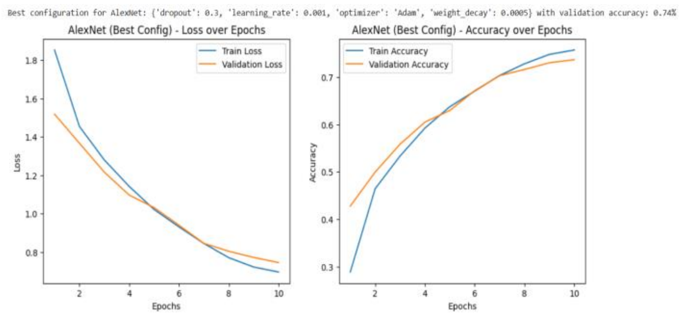
- **Optimizers**: We tested **SGD** with momentum and **Adam** optimizers. SGD is known for its stability, while Adam's adaptive learning rates offer flexibility in converging faster in some cases. For each optimizer, we used learning rates of **0.001** and **0.01** to see how they influenced the models' convergence speeds and stability.
- **Weight Decay**: Regularization was introduced through weight decay values of **5e-4** and **1e-4**. This parameter helps reduce overfitting by penalizing larger weights, making the model less sensitive to specific patterns in the training data. This was particularly useful for dense layers in AlexNet and for improving MobileNetV2's stability.
- **Data Augmentation**: To make each model more adaptable to real-world variations, we experimented with two levels of augmentation:
    - **Basic augmentation**: Included random cropping and horizontal flipping to provide slight variations in each image.
    - **Advanced augmentation**: Added more complex transformations, such as color jittering (brightness, contrast, saturation, and hue variations). These modifications helped the models learn robust features that are less dependent on specific color or orientation, especially helpful in MobileNetV2.

The Below is the result After the Hyper-Parameter Tunning:



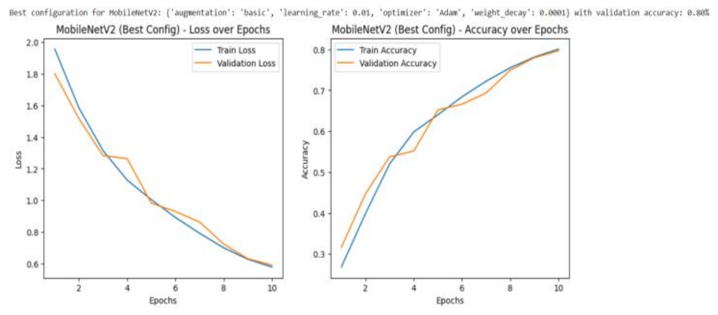
```
Best configuration for ResNet18: {'augmentation': 'basic', 'learning_rate': 0.001, 'optimizer': 'Adam', 'weight_decay': 0.0001} with validation accuracy: 0.88%
```

Fig 7: Show the ResNet18 After Hyper parameter Tunning



```
Best configuration for AlexNet: {'dropout': 0.3, 'learning_rate': 0.001, 'optimizer': 'Adam', 'weight_decay': 0.0005} with validation accuracy: 0.74%
```

Fig 8: Show the AlexNet After Hyper parameter Tunning



```
Best configuration for MobileNetV2: {'augmentation': 'basic', 'learning_rate': 0.01, 'optimizer': 'Adam', 'weight_decay': 0.0001} with validation accuracy: 0.80%
```

Fig 9: Show the MobileNetV2 After Hyper parameter Tunning

Each configuration's performance was assessed by validation accuracy, allowing us to compare results systematically. This process enabled us to select the optimal setup for each model based on objective metrics.

4.3 Evaluation

After determining the best configuration for each model based on validation performance, we performed a final evaluation using the **CIFAR-10 test set**. This stage was essential to verify each model's **generalization capability**—its ability to correctly classify unseen data. Each model's test accuracy was recorded, providing a direct comparison of how well each performed under optimal conditions. This final evaluation highlighted the strengths and limitations of each model:

- **ResNet18**: With its residual connections, ResNet18 demonstrated strong generalization and performed well on the test set, making it an ideal choice for applications where accuracy is critical.
- **AlexNet**: AlexNet's simpler architecture showed consistent accuracy, though it slightly lagged behind ResNet18, indicating its effectiveness in general but with limitations in deeper feature extraction.
- **MobileNetV2**: Despite being a lightweight model, MobileNetV2 achieved competitive accuracy, showing promise for scenarios where computational resources are limited.

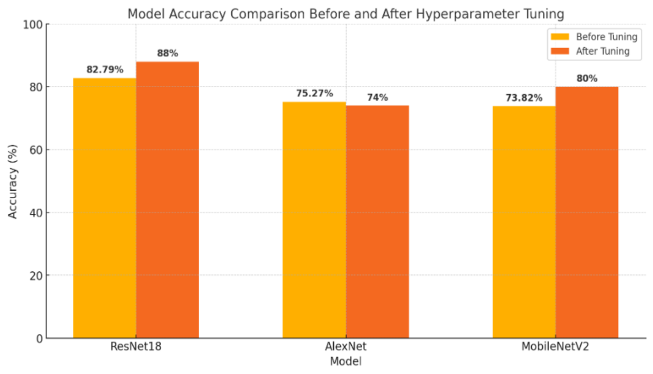
4.4 **Analysis of Results**

The experimental design effectively highlighted the differences in each model's performance and configuration sensitivity. The results showed that:

- **ResNet18** excelled in classification accuracy, likely due to its residual connections that support deeper layers without vanishing gradient issues.
- **AlexNet** performed reliably but was slightly outperformed by more modern architectures, such as ResNet18.
- **MobileNetV2** balanced efficiency with performance, making it a practical choice where lightweight models are needed, such as mobile or embedded applications.

Through these experiments, we confirmed that **hyperparameter tuning** (particularly learning rate and data augmentation) was crucial in achieving high accuracy, with advanced data augmentation contributing to better generalization in MobileNetV2 and ResNet18.

Conclusion: These experiments revealed that ResNet18 was the best-performing model on CIFAR-10 in terms of accuracy. Meanwhile, MobileNetV2 provided a good trade-off between accuracy and efficiency. This analysis offers clear guidance on model selection based on application requirements—favoring ResNet18 for high accuracy needs and MobileNetV2 for resource-constrained environments.



Summary of Best Hyperparameters:

ResNet18: 'augmentation': 'basic', 'learning rate': 0.001, 'optimizer': 'Adam', 'weight decay': 0.0001}, Best Val Accuracy: 0.88%

Alex Net: 'dropout': 0.3, 'learning rate': 0.001, 'optimizer': 'Adam', 'weight decay': 0.0005}, Best Val Accuracy: 0.74%

MobileNetV2: {'augmentation': 'basic', 'learning rate': 0.01, 'optimizer': 'Adam', 'weight decay': 0.0001}, Best Val Accuracy: 0.80%

1. Reflection on the Project

This project aimed to evaluate and improve the performance of different deep learning models on the CIFAR-10 dataset, and it provided valuable insights into model selection, tuning strategies, and the impact of hyperparameter choices.

2. Major Project Design Choices

1. **Model Selection**: Choosing ResNet18, Alex Net, and MobileNetV2 was intentional to cover a range of architectures, from traditional CNNs to more modern, efficiency-focused designs. This selection allowed us to understand the advantages of deeper networks with residual connections (ResNet18) and compare them with both simpler (Alex Net) and lightweight (MobileNetV2) models. Each model offered distinct strengths, which became clearer through our experiments.
2. **Training Strategy**: Using a consistent batch size, epochs, and CosineAnnealingLR scheduler across all models ensured that the comparisons were fair. The scheduler helped optimize learning rates smoothly, enabling the models to focus on high-level feature learning at first and fine-tune with a lower learning rate toward the end. This design choice made a noticeable difference, particularly for models like ResNet18, which benefitted from gradual learning rate decay.
3. **Hyperparameter Tuning**: A systematic grid search allowed us to pinpoint optimal settings for each model. Testing combinations of optimizers, learning rates, weight decays, and augmentation strategies was time-intensive but rewarding, as it provided substantial accuracy improvements, especially in MobileNetV2 and ResNet18. Advanced data augmentation techniques, such as colour jittering, contributed positively to model generalization, revealing the impact of real-world variations on model robustness.

3. Ideas for Future Work

Based on the findings, several future directions could enhance or build on this project:

4. **Experimenting with Deeper Models**: Extending this work to even deeper models like ResNet50 or exploring newer architectures like Vision Transformers (ViTs) could provide insights into how model depth and architecture advancements impact performance on CIFAR-10. Given the promising results of ResNet18, a deeper model might capture even more nuanced patterns.
5. **Additional Regularization Techniques**: Beyond weight decay, incorporating dropout, especially in ResNet18 and MobileNetV2, could add another layer of regularization, potentially improving accuracy further by mitigating overfitting. This is particularly relevant for datasets like CIFAR-10,

where overfitting can happen quickly due to relatively limited sample variety.

6. **Fine-Tuning with Transfer Learning**: Pretraining these models on larger, more diverse datasets (like ImageNet) before fine-tuning on CIFAR-10 could improve performance significantly. Transfer learning would allow the models to start with a strong foundation of visual features, requiring less adjustment on CIFAR-10.
7. **Evaluating Efficiency and Deployment**: Future work could also consider optimizing the models for deployment. MobileNetV2, for example, is highly efficient and suitable for mobile devices. Exploring quantization, pruning, or distillation techniques could make the models more practical for real-world applications, where resources may be limited.

- ❖ In conclusion, this project demonstrated the importance of thoughtful model selection, training strategies, and hyperparameter tuning. The insights gained here can serve as a solid foundation for future experiments focused on accuracy, efficiency, and real-world applicability in image classification tasks.

Reference's:

1. He, K., Zhang, X., Ren, S. & Sun, J., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778.

2. Krizhevsky, A., Sutskever, I. & Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105.

3. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.C., 2018. MobileNetV2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510-4520
4. Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep Learning*. Cambridge: MIT Press.

Pictures:

Fig2:
https://medium.com/analytics-vidhya/the-architecture-implementation-of-alexnet-135810a3370

Fig3:
https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c