National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

School of Electrical Engineering and Computer Sciences

CS 405: Deep Learning

Class: BSCS 12 C

SUBMITTED BY:

NAME:	ABDUL REHMAN
CMS ID:	408651
CLASS:	BSCS-12C
Assig No	02

Date: 17-04-2025

Department of Computing

Deep Learning Model Training and Evaluation Report

1. Introduction and Project Overview

The goal of this project was to explore and experiment with deep learning architectures for a classification task. Given the importance of both model efficiency and accuracy, I began by carefully focusing on the most fundamental aspect of any deep learning pipeline—data preprocessing. Proper data preprocessing is critical for ensuring that the model learns from a well-prepared and varied dataset. The approach I followed involved a structured and systematic process, beginning with data preparation and moving through various model architectures, including MobileNetV2 and ResNet families (ResNet-50 and ResNet-18). The insights gained from this workflow helped me refine my models iteratively, optimizing them for both accuracy and generalization.

2. Data Preprocessing and Preparation

The foundation of any deep learning model lies in the data. Therefore, data preprocessing was the first and most crucial step in my project. The dataset was carefully loaded, cleaned, and split into training, validation, and test sets. To prevent overfitting and improve generalization, I applied a variety of image transformations:

- **Resizing:** All images were resized to a consistent size of 224x224 pixels to match the input size expected by the pre-trained models.
- **Normalization:** I normalized the image pixel values using the mean and standard deviation of the ImageNet dataset (which is commonly used for pre-trained models like MobileNetV2 and ResNet), ensuring that the input data was scaled correctly for the models.
- **Augmentation:** To improve generalization, I introduced random horizontal flipping as a form of data augmentation. I applied small random rotations to images to help the model generalize to various orientations of objects. This transformation helped the model learn more robust features and avoid memorizing specific patterns.

Data Loaders were set up to ensure efficient batching and shuffling. This allowed the GPU to process the data more quickly and efficiently while ensuring randomness in each epoch, reducing bias and preventing the model from learning in a way that overly fits specific sequences.

3. MobileNetV2 Model – Initial Experimentation

3.1 MobileNetV2 Overview

For the first stage of my experimentation, I chose **MobileNetV2**, a lightweight and efficient convolutional neural network (CNN) that strikes a good balance between accuracy and computational efficiency. MobileNetV2 is particularly suitable for environments with limited computational resources, such as mobile devices. I used a pre-trained version of the model, which had been trained on the ImageNet dataset, and I fine-tuned it for my specific task by modifying the classifier head.

3.2 Architecture Modifications and Training Strategy

The **classifier head** of MobileNetV2 was customized to match the number of output classes for my task. This involved adding a fully connected layer with dropout and batch normalization, followed by activation functions such as ReLU to increase the model's non-linearity and learning capacity.

For **optimization**, I used the CrossEntropyLoss function, which is ideal for multi-class classification tasks. Along with this, I employed the Adam optimizer, which provides adaptive learning rates, allowing the model to converge efficiently. To stabilize training, I used a **learning rate scheduler** to gradually reduce the learning rate as training progressed, preventing large learning rate fluctuations in later epochs.

3.3 Early Stopping and Model Evaluation

To prevent overfitting, I implemented **early stopping**, which halted training when the validation loss failed to improve for several consecutive epochs. This decision was based on monitoring both Top-1 and Top-5 accuracy metrics. The **Top-1 accuracy** tracked the percentage of times the model's most confident prediction was correct, while the **Top-5 accuracy** checked if the correct label was within the top 5 predictions—an essential metric in multi-class classification problems with many similar classes.

The first experiment yielded a strong baseline performance, with a **Top-1 accuracy of 76.78%** on the test dataset after stopping at epoch 15, as validation loss started to plateau. This model represented a good trade-off between learning and generalization. The final model, achieving Top-1 Accuracy: 76.78%, showcased effective training while avoiding overfitting through this process. This model represented a solid generalization capability, striking a balance between training and validation loss and accuracy.

4. MobileNetV2 – Experimentational Version

4.1 Architectural Changes for Enhanced Generalization

Encouraged by the results of the first experiment, I moved forward to create a second version of MobileNetV2 with improved architecture and optimization settings. In this version, I made the decision to **freeze more pretrained layers** up to layer 10. This allowed the model to preserve the robust low-level features learned from ImageNet, such as edge and texture information, while only fine-tuning the deeper layers and the new classifier head.

4.2 Updated Classifier Head and Optimizer

I redesigned the classifier head to be **deeper and more expressive**, incorporating both **LeakyReLU** and **ReLU** activations to avoid "dead neurons" and ensure the network retained the ability to learn complex patterns. Additionally, I employed **batch normalization** and **dropout** to prevent overfitting and accelerate convergence.

The **optimizer** was changed from **Adam** to **AdamW**. The AdamW optimizer incorporates weight decay, which provides an additional form of regularization and helps to create a more generalizable model.

I also modified the learning rate scheduler to a **StepLR** scheduler, which reduces the learning rate every 10 epochs. This allows the model to settle into local minima more gradually and helps prevent the optimizer from overshooting.

4.3 Results and Performance Evaluation

Through careful monitoring of training and validation losses, as well as Top-1 and Top-5 accuracies across all epochs, the second version of the model demonstrated more stable performance compared to the first iteration. Overfitting was delayed, and the model showed consistent improvement through **epoch 17**. Early stopping was implemented, triggered when the validation loss showed no further improvement, ensuring that the model did not overfit. By the time training was halted, the model exhibited a significant increase in generalization capabilities, providing a strong balance between training efficiency and robustness. The test accuracy of this second version **was 47.73%,** which represented a marked improvement in performance. This iteration showed not only better generalization but also enhanced stability and reliability, making it a more effective model overall.

5. ResNet Family Experiments – ResNet-50 and ResNet-18

After experimenting with MobileNetV2, I extended the analysis to the **ResNet family**, specifically with **ResNet-50** and **ResNet-18**, both pre-trained on ImageNet. These architectures are known for their **residual connections**, which help mitigate the vanishing gradient problem and allow for deeper networks. The goal was to compare the performance of these deeper models with the MobileNetV2 baseline and analyze their effectiveness in my classification task.

5.1 ResNet-50 Overview

ResNet-50 is a **deeper model** with 50 layers, providing more capacity for complex feature extraction. I employed a similar approach as with MobileNetV2, starting with freezing all layers except the **last block (layer4)** and the **fully connected (fc) layer**, which were fine-tuned during training.

The **classifier head** was modified to consist of a **fully connected layer** followed by **batch normalization**, **LeakyReLU activation**, and **dropout** for regularization. I used **AdamW** as the optimizer and tracked training and validation metrics using both **Top-1** and **Top-5 accuracy**.

5.2 ResNet-18 Overview

ResNet-18 is a **shallower version** of ResNet, consisting of 18 layers. While having fewer parameters, it still offers solid performance and faster training due to its smaller size. Similar to ResNet-50, I froze all layers except for the last block and the fully connected layer. I used a similar architecture for the classifier head, but training was typically faster compared to ResNet-50.

5.3 Performance Comparison

Both ResNet-50 and ResNet-18 performed well, but ResNet-50 consistently outperformed ResNet-18 in terms of validation accuracy and Top-1/Top-5 metrics, likely due to its deeper architecture. However, ResNet-18 trained faster, which could be advantageous in time-sensitive or resource-limited scenarios. Below are the summarized results for both:

- **ResNet-50**:
  - **Best Validation Accuracy:** 44.06%
  - **Top-1 Accuracy:** 43.52%
  - **Top-5 Accuracy:** 62.19%
- **ResNet-18**:
  - **Best Validation Accuracy:** 43.89%
  - **Top-1 Accuracy:** 42.61%

- **Top-5 Accuracy:** 60.42%

6. Conclusion and Insights

The experimentation with **MobileNetV2** and **ResNet family models (ResNet-50 and ResNet-18)** provided valuable insights into model selection, fine-tuning strategies, and the balance between model complexity and efficiency. MobileNetV2 demonstrated strong baseline performance with efficiency in resource-limited environments, while ResNet-50 outperformed ResNet-18 in terms of accuracy, offering better feature extraction capacity at the cost of longer training times.

Key Takeaways:

- **MobileNetV2** was optimal for efficiency, offering good performance in less resource-intensive settings.
- **ResNet-50** outperformed ResNet-18 in terms of accuracy, showcasing the benefits of deeper architectures with residual connections.
- The **early stopping** technique was critical in preventing overfitting across both model families, ensuring that training halted at the optimal point.