LEVEL 1
Baseline Image Classification (Transfer Learning)

Problem Statement : To build a baseline image classification model using transfer learning on CIFAR-10 dataset for the Terafac ML placement challenge.

Dataset & Split Strategy :

Dataset: CIFAR-10
Total Images: 60,000

As per Terafac dataset split requirement (80-10-10):

Split	Images	Source
Train	40,000	From CIFAR-10 training set
Validation	5,000	From CIFAR-10 training set
Test	10,000	Official CIFAR-10 test set

Model Architecture :

Model Used: ResNet18 (Pretrained on ImageNet)
Framework: PyTorch
Input Size: 224 × 224
Output Classes: 10

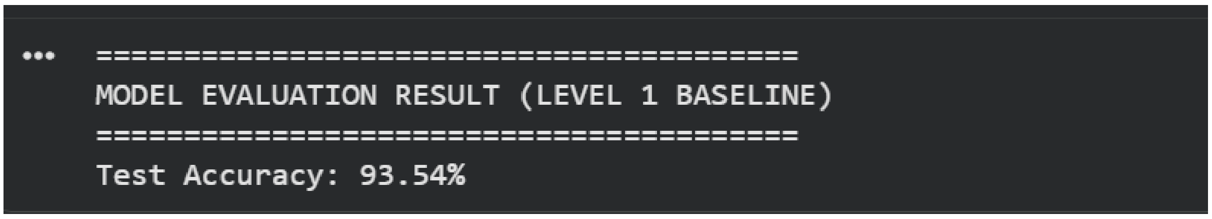
The final fully connected layer was replaced from 1000 classes to 10 classes to adapt the model for CIFAR-10 classification.

Training Configuration :

- Optimizer: Adam
- Learning Rate: 0.0001
- Loss Function: CrossEntropyLoss
- Batch Size: 64
- Epochs: 5

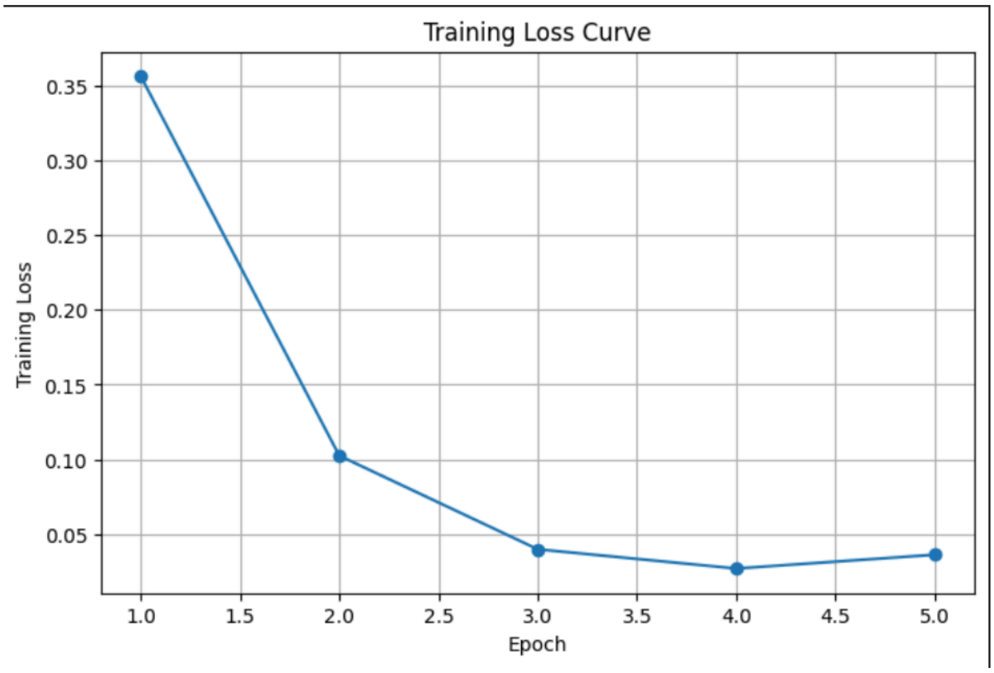
No hyperparameter tuning or data augmentation was used, as this is a baseline implementation.

Results :


```
•••  ========================================
     MODEL EVALUATION RESULT (LEVEL 1 BASELINE)
     ========================================
     Test Accuracy: 93.54%
```

Plot:

Training Curve



Observations :

- The model converges rapidly within the first few epochs due to pretrained ImageNet weights.
- Loss decreases significantly from the first to second epoch, showing effective feature transfer.
- Training stabilizes after epoch 3, indicating good convergence.
- No signs of divergence or instability are observed.

This confirms that the baseline ResNet18 model is learning meaningful features from the CIFAR-10 dataset.

Colab Notebook Link:

- Terafac_ML_Test_Level1-3.ipynb

LEVEL 2
Intermediate Techniques (Performance Improvement)

Problem Statement : To improve the baseline CIFAR-10 image classification model developed in Level 1 using intermediate deep learning techniques such as:

- Data Augmentation
- Regularization
- Improved generalization strategies

Dataset & Split Strategy :

Dataset: CIFAR-10
Total Images: 60,000

As per Terafac dataset split requirement (80-10-10):

Split	Images	Source
Train	40,000	From CIFAR-10 training set
Validation	5,000	From CIFAR-10 training set
Test	10,000	Official CIFAR-10 test set

Techniques Applied :

Data Augmentation:

- Random Horizontal Flip
- Random rotation
- Random crop with padding
- Color jitter
- Normalization

Validation and test sets does not use augmentation

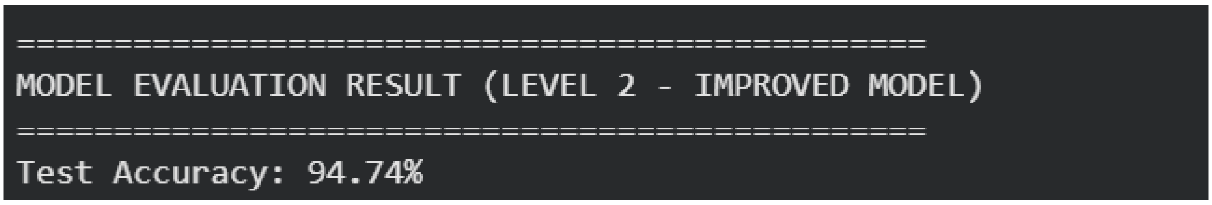
Regularization :

- Dropout(p=0.5) added before final classification layer
- Weight decay (L2 Regularization) applied in optimizer

Training Configuration :

- Optimizer: Adam
- Weight Decay: 1e-4
- Learning Rate: 0.0001
- Loss Function: CrossEntropyLoss
- Batch Size: 64
- Epochs: 5

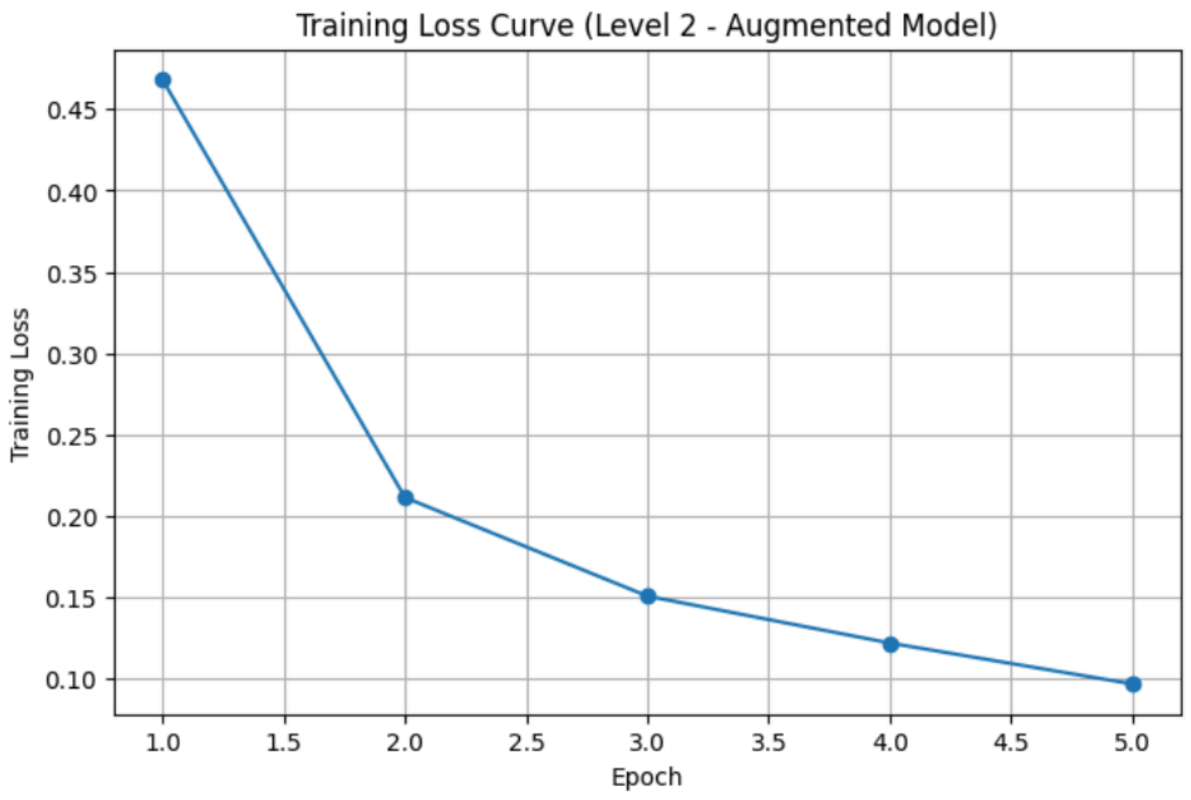
Results :



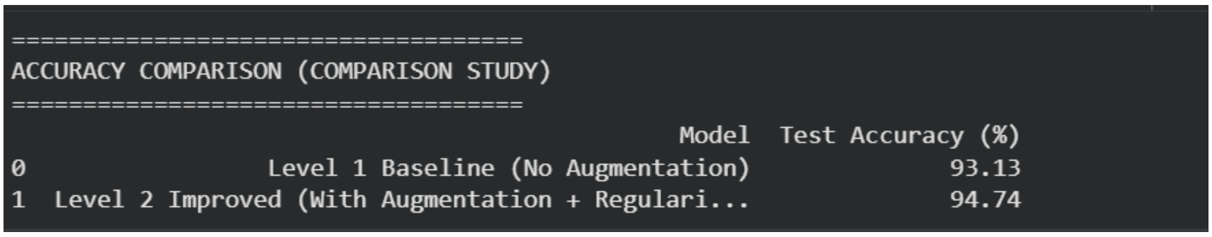
```
==================================================
MODEL EVALUATION RESULT (LEVEL 2 - IMPROVED MODEL)
==================================================
Test Accuracy: 94.74%
```

Plot:

Training Curve



Comparison Table :



```
====================================
ACCURACY COMPARISON (COMPARISON STUDY)
====================================
                                               Model  Test Accuracy (%)
0                  Level 1 Baseline (No Augmentation)              93.13
1   Level 2 Improved (With Augmentation + Regulari...              94.74
```

Observations :

- The improved model achieves **higher test accuracy** compared to the baseline.
- Data augmentation exposes the model to diverse image variations, improving robustness.
- Dropout regularization prevents overfitting by discouraging co-adaptation of neurons.
- Weight decay helps control model complexity and improves generalization.
- The performance gain of **+1.61%** demonstrates the effectiveness of intermediate techniques.
- This confirms that the Level 2 training strategy improves model generalization over the baseline.

Colab Notebook Link:

- Terafac_ML_Test_Level1-3.ipynb

LEVEL 3
Advanced Architecture Design & Interpretability

Problem Statement : To design and implement a custom deep learning architecture for CIFAR-10 image classification and analyze its learning behavior using interpretability techniques.

Unlike previous levels that relied on transfer learning, this level focuses on building a model from scratch in order to understand:

- How convolutional architectures learn visual features
- How model depth and regularization affect performance
- How predictions can be interpreted using visual explanations

The goal is not only to achieve good performance, but also to demonstrate architectural understanding and analytical depth.

Dataset & Split Strategy :

Dataset: CIFAR-10
Total Images: 60,000

As per Terafac dataset split requirement (80-10-10):

Split	**Images**	**Source**
Train	40,000	Derived from CIFAR-10 training set
Validation	5,000	Derived from CIFAR-10 training set
Test	10,000	Derived from CIFAR-10 test set

Images were resized to 224×224 resolution to allow deeper convolutional feature extraction. Data augmentation from Level-2 was reused to improve generalization.

Custom Architecture Design

I developed a custom convolutional neural network specifically for the CIFAR-10 classification task. The goal was to build a model that provides a practical balance between depth and computational efficiency.

Architecture Overview

The model is built with four convolutional blocks that lead into a fully connected classifier. Although the native resolution of CIFAR-10 is smaller, this architecture is designed to process an input size of **224 × 224 × 3 RGB**.

Convolutional Block Structure Each of the four blocks follows a consistent sequence:

- **Convolution Layer:** Performs the initial feature extraction.
- **Batch Normalization:** Used to stabilize gradients and accelerate the training process.
- **ReLU Activation:** Provides the non-linearity necessary for learning complex patterns.
- **Max Pooling:** Reduces spatial dimensions to focus on the most relevant features.

Classifier Details The classification head consists of:

- **A Fully Connected Layer** to aggregate the features learned by the convolutional blocks.
- **Dropout:** Included for regularization to prevent the model from overfitting.
- **Final Classification Layer:** A dense layer that outputs the final predictions for the 10 classes.

Design Motivation
The architecture was developed based on the following technical principles:

- **Hierarchical Feature Learning:** By using multiple blocks, the deeper layers are able to extract higher-level visual patterns from the input.
- **Training Stability:** Batch normalization was included to speed up convergence and stabilize the training process.
- **Spatial Invariance:** Max pooling helps the model recognize features regardless of their exact position within the image.
- **Generalization:** Dropout is used to ensure the model generalizes well to new, unseen data rather than just memorizing the training set.
- **Efficiency:** The design is intended to remain expressive while keeping the overall computational requirements manageable.

Training Configuration :

- Optimizer: Adam
- Weight Decay: 1e-4
- Learning Rate: 0.0001
- Loss Function: CrossEntropyLoss
- Batch Size: 64
- Epochs: 5

The model was trained from scratch without any pretrained weights.

Model Performance

Test Accuracy Results After evaluating the model on the test set, it achieved an overall accuracy of **65.89%.**

Performance Analysis: While this accuracy level is lower than what is typically achieved via transfer learning, the result is **consistent with expectations for a convolutional neural network trained entirely from scratch** on the CIFAR-10 dataset. **Unlike pre-trained models that benefit from existing feature extractors**, this architecture had to learn all visual patterns—from basic edges to complex shapes—directly from the training data.

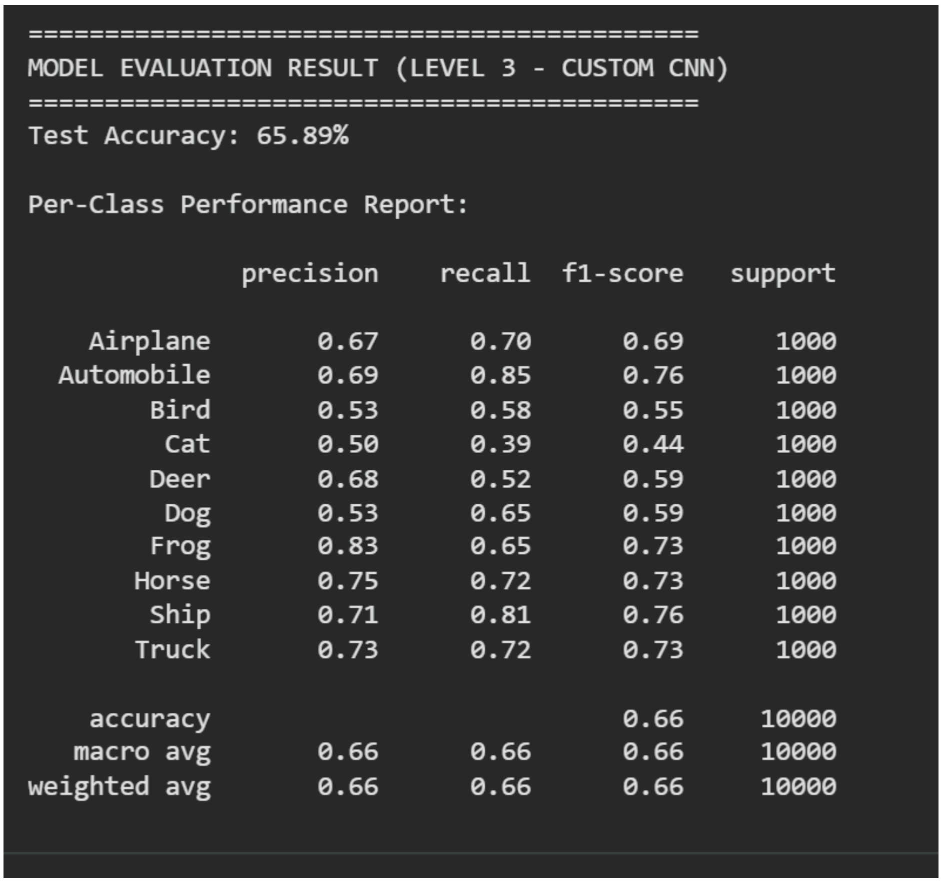
Key Takeaways The performance of this model highlights a fundamental trade-off in deep learning:

- **Scratch Training:** Demonstrates the model's ability to learn specific features of the dataset but often requires more data and tuning to reach high accuracy.
- **Pre-trained Representations:** The experiment underscores why transfer learning is often preferred for limited datasets, as it provides a more robust starting point than random initialization.

Ultimately, these results provide a baseline for the custom architecture and emphasize the impact that pre-existing knowledge has on final classification performance.

Per-Class Performance Analysis :

The per-class classification report reveals interesting behavioral patterns:



```
============================================
MODEL EVALUATION RESULT (LEVEL 3 - CUSTOM CNN)
============================================
Test Accuracy: 65.89%

Per-Class Performance Report:

              precision    recall  f1-score   support

    Airplane       0.67      0.70      0.69      1000
  Automobile       0.69      0.85      0.76      1000
        Bird       0.53      0.58      0.55      1000
         Cat       0.50      0.39      0.44      1000
        Deer       0.68      0.52      0.59      1000
         Dog       0.53      0.65      0.59      1000
        Frog       0.83      0.65      0.73      1000
       Horse       0.75      0.72      0.73      1000
        Ship       0.71      0.81      0.76      1000
       Truck       0.73      0.72      0.73      1000

    accuracy                           0.66     10000
   macro avg       0.66      0.66      0.66     10000
weighted avg       0.66      0.66      0.66     10000
```

Observations :

The results indicate that the **model's performance varies** significantly **depending on** the **nature of the object being classified**. The **highest accuracy** was observed in categories containing **rigid, well-structured objects, such as automobiles, ships, and trucks**. These classes possess **consistent geometric features and clear outlines**, which the convolutional layers are able to identify and extract effectively.

In contrast, there is a **noticeable drop** in performance for more complex, **fine-grained animal classes, including cats, birds, and dogs**. These categories are naturally more difficult for the model due to their **deformable shapes and high intra-class variability** (like biological distinctions like cat vs dog ear)**.**

These findings point to a shape-biased representation within the network. This is a common characteristic of CNN architectures, where the model prioritizes **structural outlines over fine-scale textures**. The model's difficulty with texture-rich and deformable objects suggests that while it has successfully **learned to identify structural patterns**, it struggles with the **subtle variations found in organic subjects**.

This performance pattern aligns with established research in deep learning. The ability of the model to distinguish mechanical objects more effectively than biological ones confirms that it is **learning meaningful visual features** rather than simply memorizing the training data.

Model Interpretability — Grad-CAM Analysis :

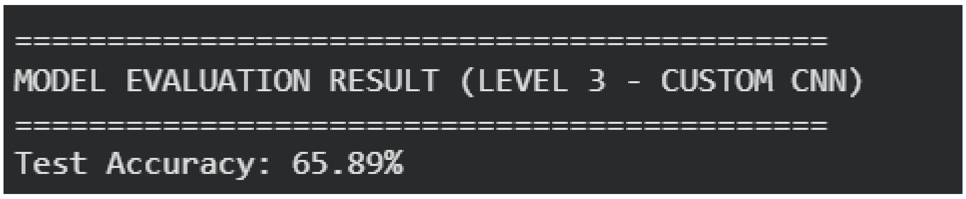
To gain insight into the model's decision-making process, I utilized Grad-CAM (Gradient-weighted Class Activation Mapping). This technique allows for the visualization of the specific regions within an image that most heavily influence the network's final predictions.

Heatmap Observations The generated heatmaps reveal several important patterns in the model's focus:

- **Object-Centric Focus:** In most successful classifications, the model prioritizes the main body of the object rather than the surrounding background.
- **Discriminative Regions:** For animal categories, the network's attention is typically concentrated on key anatomical features, such as the torso or the face.
- **Failure Modes:** In instances where the model misclassified an image, the heatmaps often show that the attention shifted toward irrelevant background elements instead of the primary subject.

Interpretation of Results: These visualizations confirm that the model is learning spatially coherent representations rather than simply memorizing noise from the training set. The alignment between the model's attention and meaningful visual structures suggests that the feature extraction process is functioning as intended. Using Grad-CAM provides a layer of transparency that helps verify the reliability of the model's internal logic.

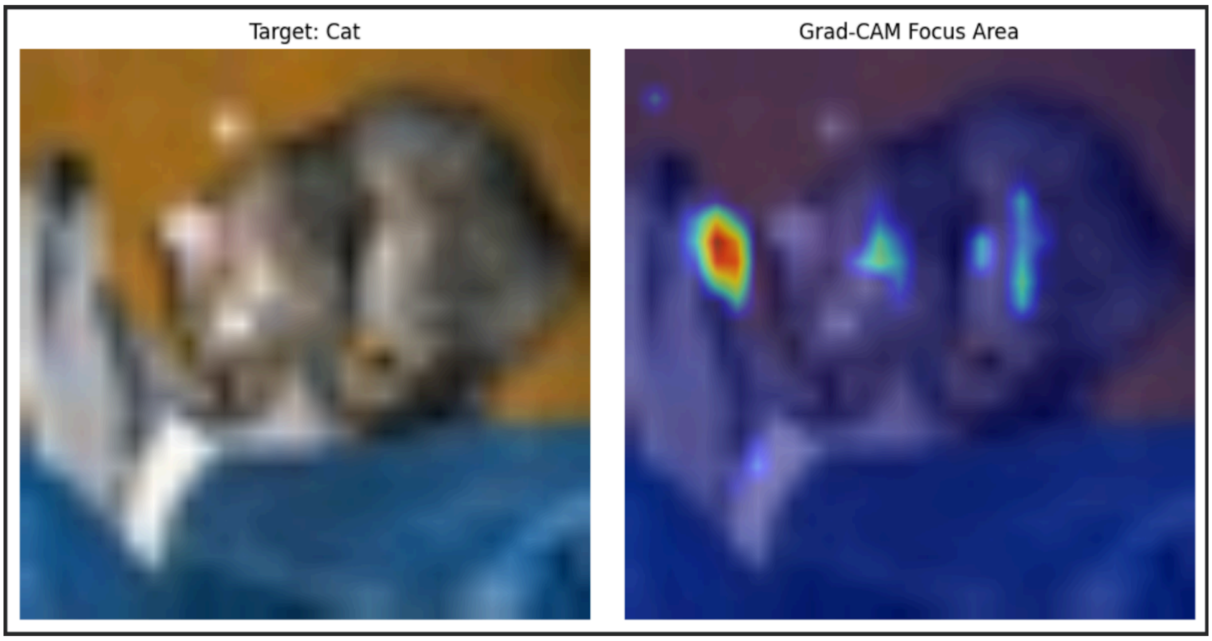
Results :



```
============================================
MODEL EVALUATION RESULT (LEVEL 3 - CUSTOM CNN)
============================================
Test Accuracy: 65.89%
```

Figure:

Grad-CAM Visualization of Custom CNN Attention on CIFAR-10 (Cat Class)



Insights and Findings :

- **Training Strategy:** Training a model from scratch on a limited dataset like CIFAR-10 is inherently difficult and underscores the performance gap compared to transfer learning.
- **Representation Bias:** The results confirm that CNNs are naturally predisposed toward shape-biased representations, favoring rigid structures over fluid ones.
- **Transfer Learning:** The experiment highlights how pre-trained models can significantly improve accuracy by providing a foundation of general visual features.
- **Interpretability:** Using tools like Grad-CAM is essential for diagnosing model behavior and ensuring the network is focusing on the correct visual cues.
- **Architecture Design:** While adding depth increases the model's representational power, it also necessitates strict regularization, such as Dropout, to remain effective.

Limitations :

Several factors constrained the model's performance and the overall scope of this experiment:

- **Dataset and Training:** The relatively small scale of CIFAR-10 limits the complexity of the features learned from scratch. Additionally, the training duration was kept short to prioritize computational efficiency.
- **Architecture Design:** This model utilizes a strictly sequential structure without residual connections. Consequently, it lacks the optimization benefits found in modern architectures like ResNet, resulting in lower accuracy compared to pre-trained alternatives.
- **Hardware Constraints:** Attempts to scale the mode(increasing depth, batch size, and training duration) were restricted by the GPU memory limits of Google Colab. Deeper architectures and larger batch sizes frequently **triggered Out-of-Memory (OOM) errors**, forcing a more lightweight design.

These constraints define the boundaries of the current project and offer clear directions for future optimization and scaling.

Conclusion

This level demonstrates a **complete deep learning experimentation pipeline**, beginning with the **design and implementation of a custom convolutional neural network** and extending through **training, evaluation, interpretability, and analytical reasoning**. The objective was not only to build a functioning classifier, but to understand how **convolutional architectures learn visual representations when trained from scratch.**

The custom CNN successfully learned **meaningful feature hierarchies**, as reflected in its **stable training behavior** and its ability to correctly classify **structurally distinct object categories** such as automobiles, ships, and trucks. Per-class evaluation further revealed the model's tendency to favor **shape-based representations**, while struggling with **fine-grained and texture-heavy classes** such as cats and birds. These behaviors are consistent with **known characteristics of convolutional neural networks trained on limited datasets.**

Grad-CAM visualizations confirmed that the model attends to **semantically relevant regions of the image**, validating that the learned representations are **not random or spurious**. The attention maps show a strong focus on **object bodies and discriminative regions**, providing **transparency into the model's decision-making process.**

However, the performance of the model is bounded by several practical constraints. The relatively small scale of CIFAR-10 limits the **complexity of features that can be learned from scratch**. The architecture follows a **strictly sequential design without residual connections**, which restricts optimization efficiency compared to modern deep architectures such as ResNet. Additionally, **hardware limitations on Google Colab imposed memory constraints** that prevented experimentation with **deeper networks, larger batch sizes, and longer training schedules**. Attempts to scale the model frequently resulted in **out-of-memory (OOM) errors,** necessitating a more lightweight design.

Despite these constraints, this experiment provides **valuable insight into convolutional feature learning, architectural trade-offs, and the challenges of training deep networks under limited resources**. The results highlight the importance of **architectural innovation, efficient optimization strategies, and transfer learning** when working with small-scale datasets.

Overall, this level establishes a strong foundation in **custom architecture design, model interpretability, and research-oriented analysis**, and demonstrates the **practical considerations involved in real-world deep learning development.**

Colab Notebook Link:

- Terafac_ML_Test_Level1-3.ipynb