Feature Extraction and Visualization from CIFAR-10 using ResNet50

Qiao Qin, 120409875

1. Introduction

The objective of this homework is to analyze the features extracted from the CIFAR-10 dataset using a pre-trained ResNet50 model. The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. ResNet50, a classic convolutional neural network, is renowned for its deep architecture and residual learning framework, making it suitable for extracting rich features from images.

Given that ResNet50 is trained on ImageNet, I hypothesized that the pre-trained model might not fully capture the essence of the CIFAR-10 dataset. Therefore, I also fine-tuned the ResNet50 model to better adapt to the CIFAR-10 data. To gain deeper insights, I extracted features not only from the last average pooling layer but also from an earlier convolutional layer. This approach allowed me to compare the performance and understand the internal workings of the CNN network more comprehensively.

2. Model, Dataset and Features

2.1 Model

For this task, I selected ResNet50, a classic convolutional neural network. ResNet50, or Residual Network with 50 layers, is renowned for its deep architecture and innovative residual learning framework. Introduced by He et al. in 2015, ResNet50 addresses the vanishing gradient problem prevalent in deep networks by introducing residual blocks. Each residual block contains skip connections that allow gradients to flow directly through the network, bypassing one or more layers, which enables the training of very deep networks. This architecture not only facilitates the training of deep networks but also improves accuracy and performance across various image recognition tasks.

2.2 Dataset

The CIFAR-10 dataset was selected for this analysis. It is a well-known dataset in

the computer vision community and consists of 10 different classes including airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each class has 6,000 images, split into 50,000 training images and 10,000 test images.

2.3 Features

Feature extraction is a crucial step in the process of analyzing and understanding the underlying patterns within a dataset. By extracting relevant features from the data, we can reduce its dimensionality, remove redundant information, and focus on the most informative aspects of the dataset. In the context of image data, feature extraction often involves using a pre-trained convolutional neural network (CNN) to identify and extract meaningful representations of the images.

In this homework, to explore the features extracted at different stages of the network, I chose to extract features from two different layers of the ResNet50 model:

- Last Average Pooling Layer: This layer comes just before the final classification layer and provides a high-level summary of the image features. Extracting features from this layer resulted in a numpy array with a shape of 60000x2048, as the CIFAR-10 dataset has 60,000 samples and the output dimension from this layer is 2048.
- Previous Convolutional Layer: I also selected an earlier convolutional layer to understand the features captured at an intermediate stage of the network. The output dimension from this layer is 1024x14x14, where 1024 is the number of channels and 14x14 is the spatial dimension of the feature map. Flattening this output would result in an array of shape 60000x200704, which is too large to store in memory. Therefore, I selected only the first 10 channels, resulting in an array with a shape of 60000x1960, which represents the features from the convolutional layer.

3. TSNE Visualization

t-Distributed Stochastic Neighbor Embedding (t-SNE) was employed to visualize the high-dimensional features extracted from the ResNet50 model. t-SNE reduces the dimensionality of the feature vectors to two dimensions, allowing for visual inspection of the feature distributions. The resulting 2D embeddings were plotted to observe the clustering of images based on their classes.

Below, Figure 1 shows the t-SNE visualization using the pretrained model after the last average pooling layer; Figure 2 shows the t-SNE visualization using the fine-tuned model after the last average pooling layer; Figure 3 shows the t-SNE visualization using the fine-tuned model after the convolutional layer.

These three images reveal expected patterns:

- Figure 1: The features from the last average pooling layer represent high-level representations of the images, resulting in 10 distinct but somewhat noisy clusters corresponding to the 10 classes in the CIFAR-10 dataset.

- Figure 2: With the fine-tuned model, the clusters are more distinct and less noisy, indicating that fine-tuning on CIFAR-10 improves the model's ability to capture the specific features of this dataset.
- Figure 3: Despite fine-tuning, the features from the convolutional layer do not form clear clusters. This is because these features represent low-level or mid-level patterns, such as edges and textures, rather than high-level semantic information. As a result, samples from each class are mixed together.

This analysis highlights the hierarchical nature of feature representations in a CNN, where early layers capture basic visual patterns and deeper layers capture more abstract, high-level features.

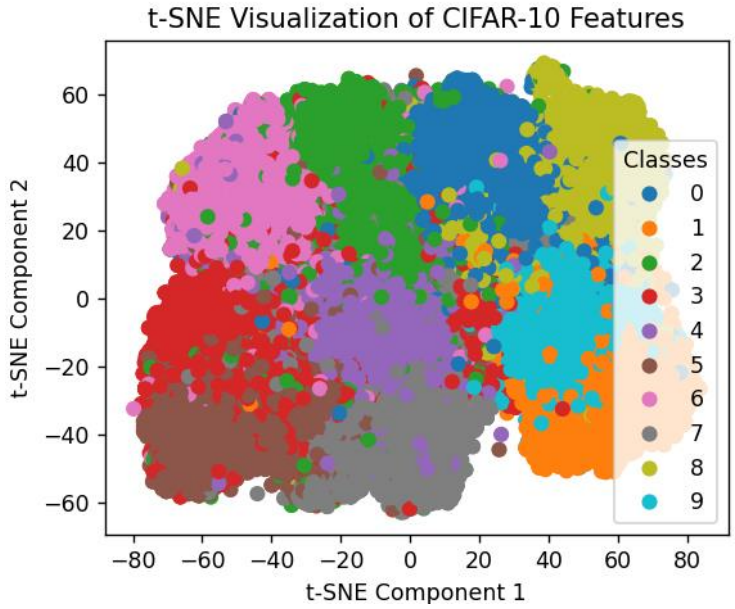


Figure 1 Last Average Pooling Layer from Pretrained Model

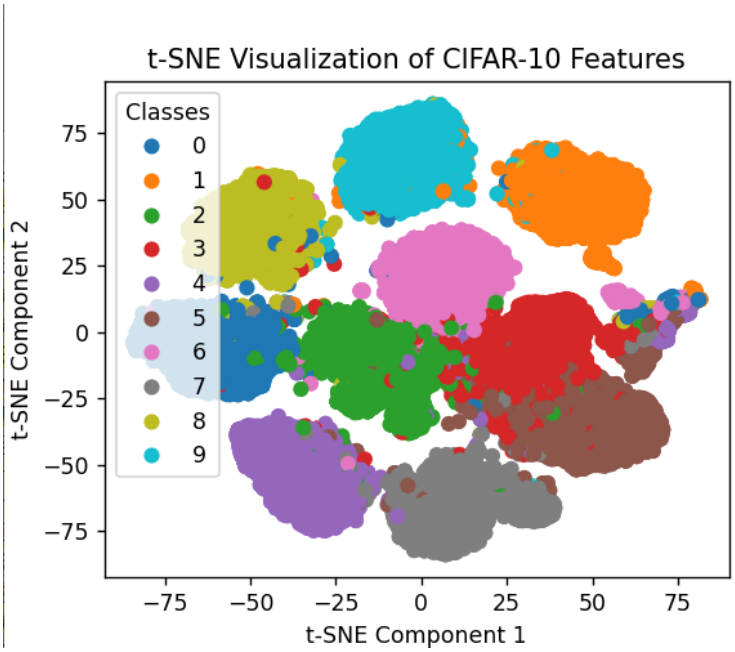


Figure 2 Last Average Pooling Layer from Finetuned Model

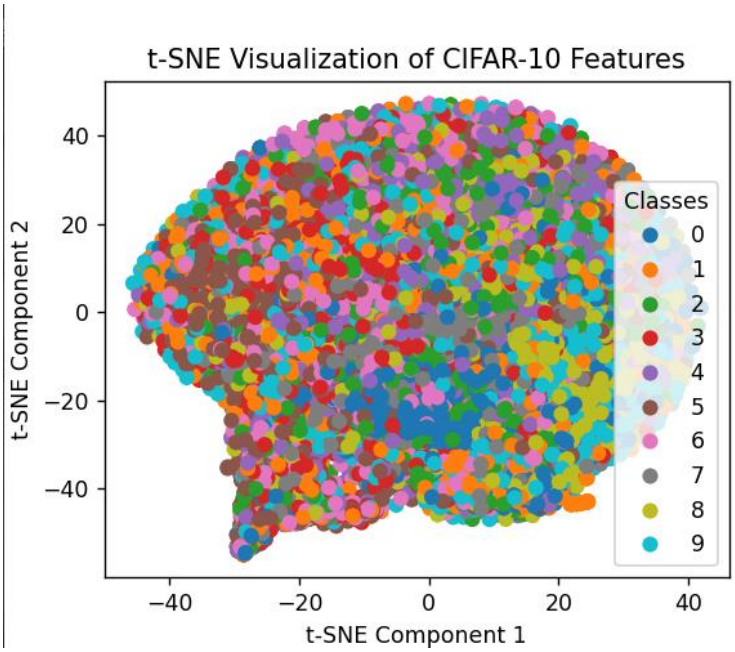


Figure 3 Convolutional Layer from Finetuned Model

4. Intra-Class Variance and Inter-Class Variance

4.1 Intra-Class Variance

Intra-class variance measures the variability of features within each class. It quantifies how much the features of images that belong to the same class differ from each other. A lower intra-class variance indicates that the model captures consistent representations for each class, suggesting good within-class compactness.

To calculate intra-class variance:

- For each class, extract the features of all images belonging to that class.
- Compute the mean feature vector for that class.
- Calculate the variance as the mean squared distance between each feature vector and the class mean feature vector.
- Average these variances across all classes to get the overall intra-class variance.

4.2 Inter-Class Variance

Inter-class variance measures the variability of features between different classes. It quantifies how distinct the features of images from different classes are. A higher inter-class variance indicates that the model effectively separates different classes in the feature space.

To calculate inter-class variance:

- For each class, compute the mean feature vector.
- Calculate the variance as the squared distance between the mean feature

vectors of each pair of classes.

- Average these variances across all pairs of classes to get the overall inter-class variance.

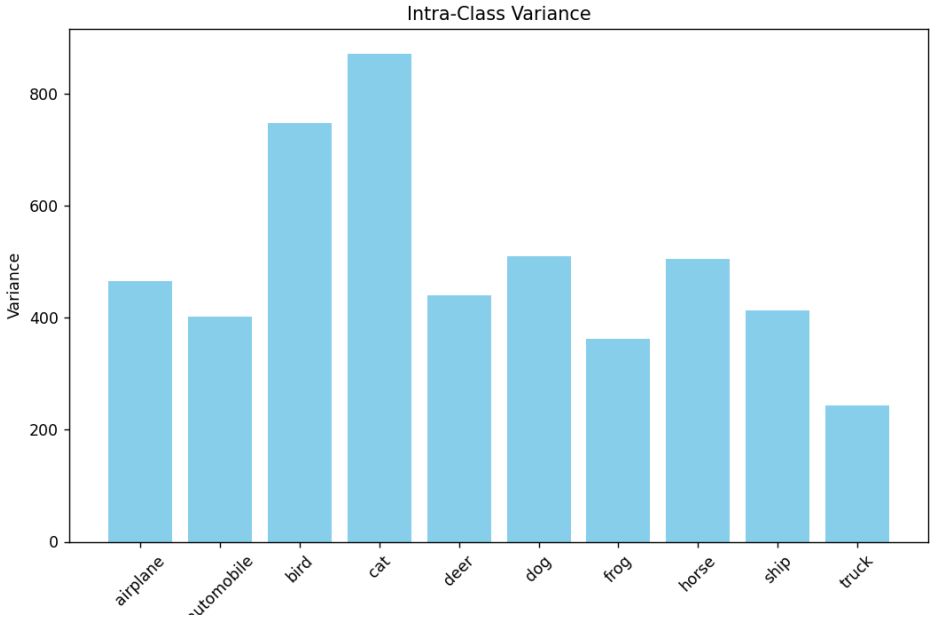
4.3 Results and Visualization

A. Pretrained model, last average pooling layer

Intra-class Variances:

[466.2, 401.9, 747.6, 872.3, 439.9, 510.0, 362.8, 504.5, 412.7, 243.8]

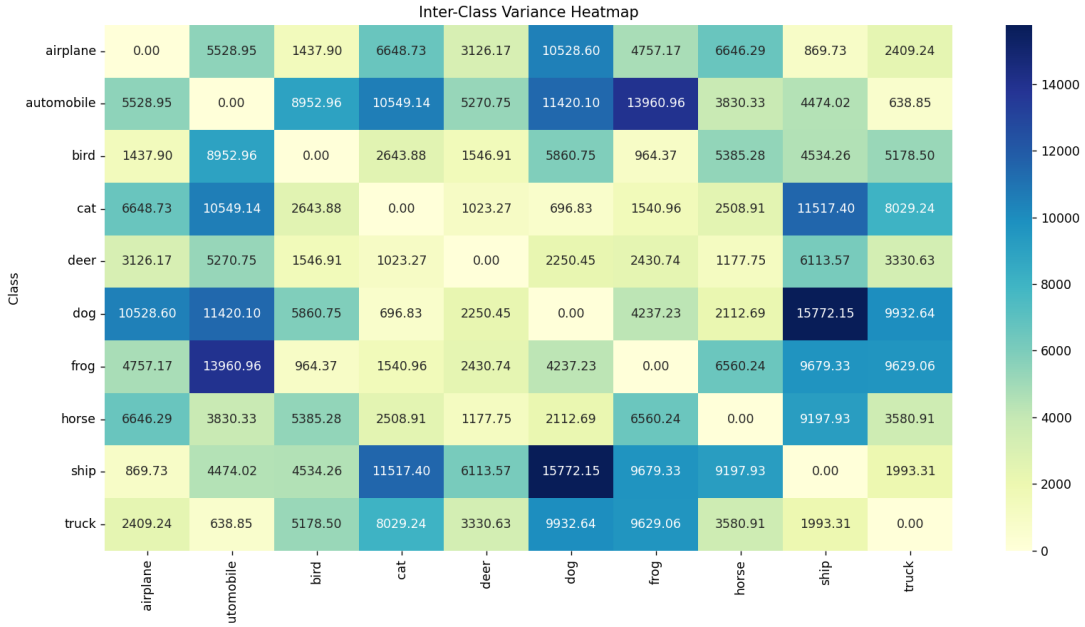
Mean Intra-class Variance: 496.21



Inter-class Variances:

[5528.9, 1437.8, 6648.7, 3126.1, 10528.6, 4757.1, 6646.2, 869.7, 2409.2, 8952.9, 10549.1, 5270.7, 11420.0, 13960.9, 3830.3, 4474.0, 638.8, 2643.8, 1546.9, 5860.7, 964.3, 5385.2, 4534.2, 5178.4, 1023.2, 696.8, 1540.9, 2508.9, 11517.4, 8029.2, 2250.4, 2430.7, 1177.7, 6113.5, 3330.6, 4237.2, 2112.6, 15772.1, 9932.6, 6560.2, 9679.3, 9629.0, 9197.9, 3580.9, 1993.3]

Mean Inter-class Variance: 5343.97

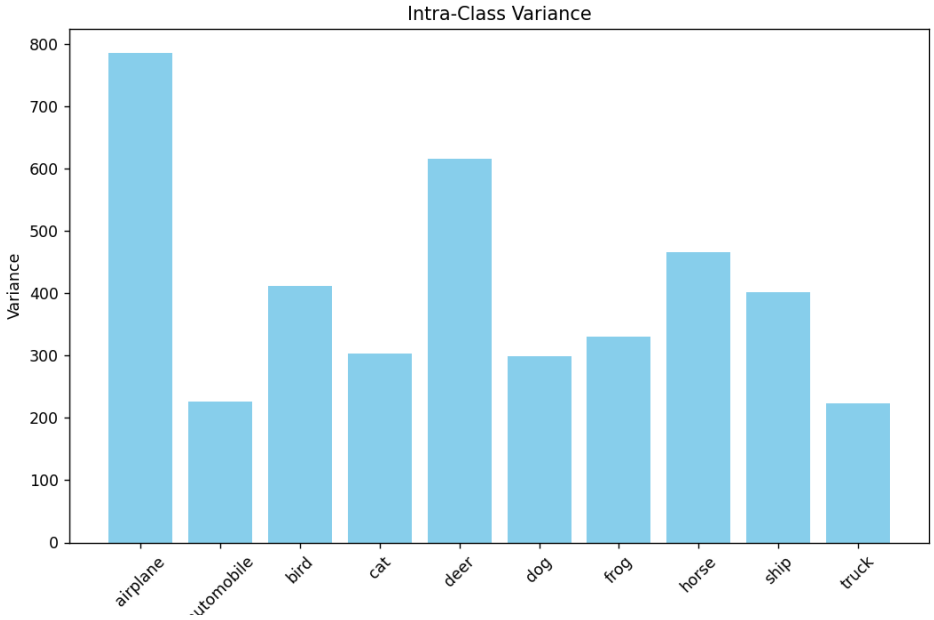


B. Fine-tuned model, last average pooling layer

Intra-class Variances:

[785.6, 226.7, 412.4, 303.8, 616.0, 299.8, 330.4, 466.3, 401.9, 223.4]

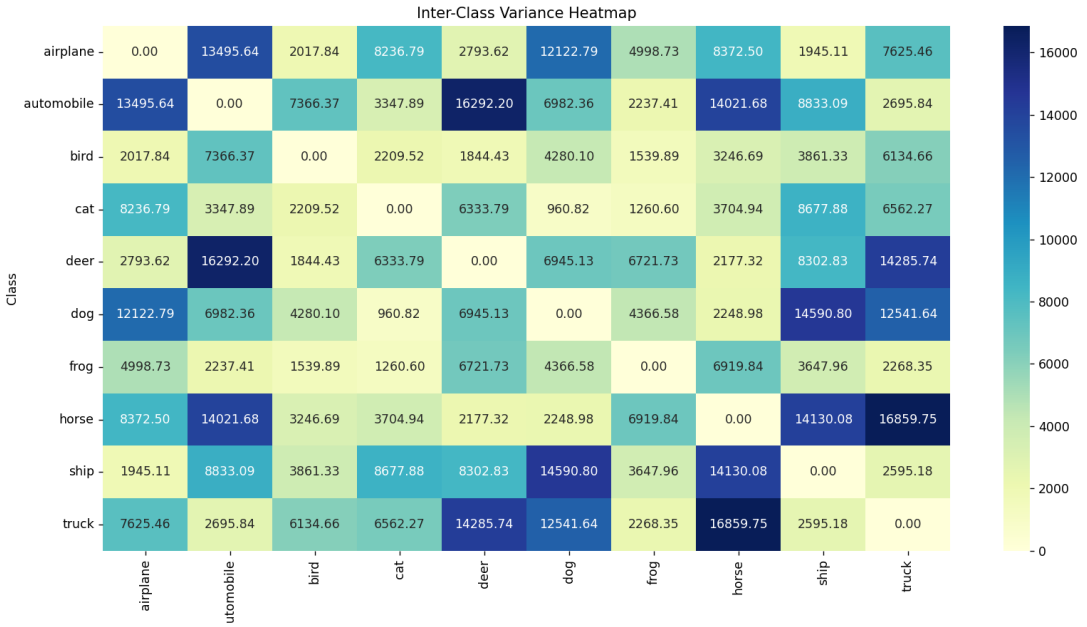
Mean Intra-class Variance: 406.68



Inter-class Variances:

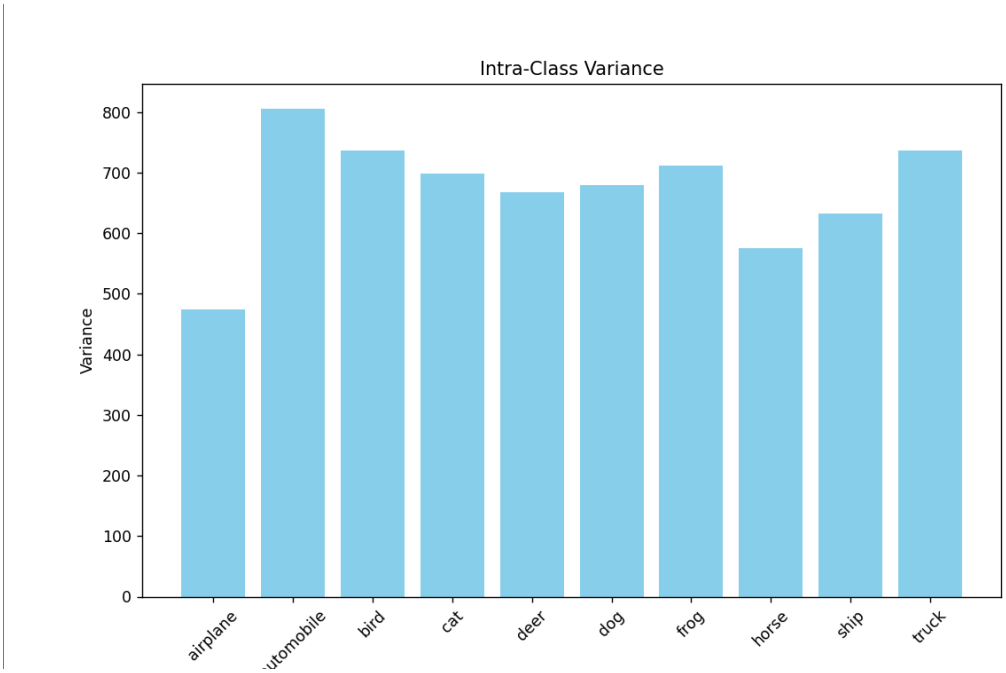
[13495.6, 2017.8, 8236.7, 2793.6, 12122.7, 4998.7, 8372.4, 1945.1, 7625.4, 7366.3, 3347.8, 16292.1, 6982.3, 2237.4, 14021.6, 8833.0, 2695.8, 2209.5, 1844.4, 4280.0, 1539.8, 3246.6, 3861.3, 6134.6, 6333.7, 960.8, 1260.6, 3704.9, 8677.8, 6562.2, 6945.1,

6721.7, 2177.3, 8302.8, 14285.7, 4366.5, 2248.9, 14590.7, 12541.6, 6919.8, 3647.9, 2268.3, 14130.0, 16859.7, 2595.1]
Mean Inter-class Variance: 6502.31



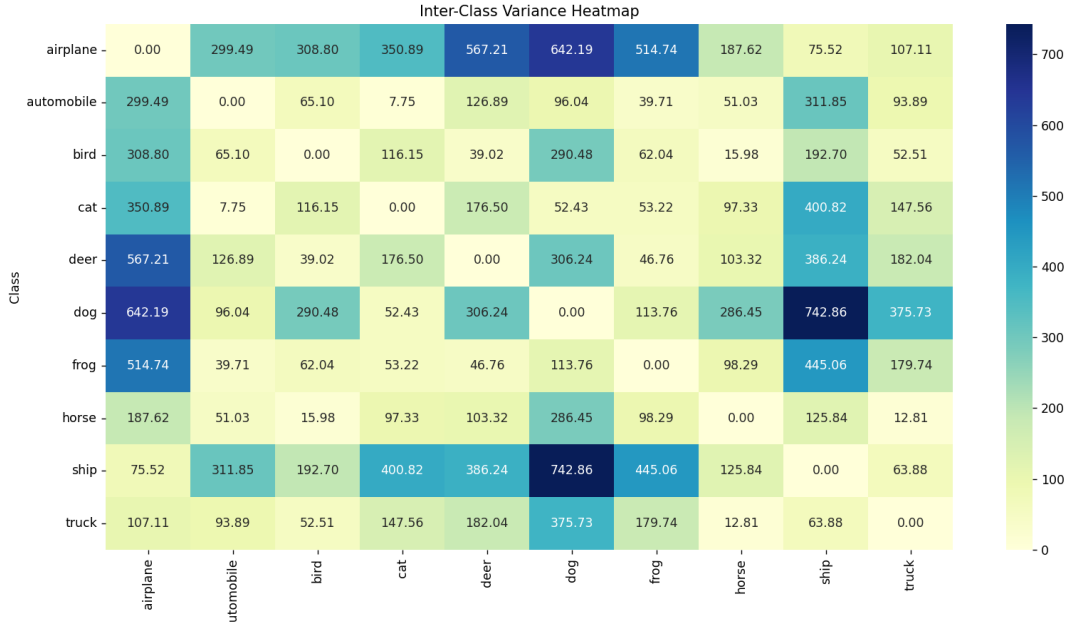
C. Fine-tuned model, convolutional layer

Intra-class Variances:
[474.7, 806.1, 735.9, 698.1, 668.0, 679.2, 712.2, 574.9, 632.2, 736.2]
Mean Intra-class Variance: 671.81



Inter-class Variances:

[299.4, 308.7, 350.8, 567.2, 642.1, 514.7, 187.6, 75.51720322668552, 107.1, 65.1, 7.7, 126.8, 96.0, 39.7, 51.0, 311.8, 93.8, 116.1, 39.0, 290.4, 62.0, 15.9, 192.7, 52.5, 176.5, 52.4, 53.2, 97.3, 400.8, 147.5, 306.2, 46.7, 103.3, 386.2, 182.0, 113.7, 286.4, 742.8, 375.7, 98.2, 445.0, 179.7, 125.8, 12.8, 63.8]
Mean Inter-class Variance: 200.25



4.4 Conclusion

First, the quantification results and the images are consistent with the t-SNE visualization:

- The intra-class variance (496.21) of the features from the last average pooling layer of the pretrained model is lower than the inter-class variance (5343.97). This indicates that each class is well-clustered, reflecting the model's ability to capture meaningful, high-level representations.
- The intra-class variance (406.68) of the features from the last average pooling layer of the fine-tuned model is even lower than the inter-class variance (6502.31). This demonstrates that the fine-tuned model classifies the samples more accurately, with even tighter within-class clustering and more distinct between-class separation.
- However, the intra-class variance (671.81) of the features from the convolutional layer of the fine-tuned model is larger than the inter-class variance (200.25). This suggests that features at this mid-level are not able to form clear clusters, indicating that the convolutional layer's features capture more low-level information that is less discriminative for class separation.

Second, inter-class variance measures the dissimilarity between feature representations of different classes. It provides insights into how well the features distinguish between classes in the dataset. From the last average pooling layer of the

fine-tuned model, we observe varying inter-class variances that reflect the similarities and differences between classes:

- Cat vs. Dog: The inter-class variance between cat and dog is relatively small (960.82), indicating that these classes share similar feature representations. This is expected as both animals share certain visual features that ResNet-50 has learned to recognize.
- Cat vs. Ship: In contrast, the inter-class variance between cat and ship is much larger (8677.88). This reflects the distinct visual characteristics between these classes. Cats and ships are visually dissimilar, leading to higher feature variance.

These insights highlight how CNN's feature representations capture the visual similarities and differences between classes after fine-tuning. The smaller inter-class variances between visually similar classes suggest that the model has effectively learned to differentiate between these classes based on their distinctive features.

5. Conclusion

In this study, I explored the effectiveness of feature extraction and visualization techniques using a pretrained ResNet-50 model on the CIFAR-10 dataset. Here are the key findings and insights:

- Dataset and Model Selection: CIFAR-10 dataset was chosen due to its diversity and complexity, while ResNet-50, a state-of-the-art convolutional neural network, served as our base model for feature extraction.
- Feature Extraction: Features were extracted from two different layers: the last average pooling layer and a mid-level convolutional layer. The last average pooling layer provided high-level features, capturing abstract representations of the images. The mid-level convolutional layer offered more detailed but less discriminative features compared to the last pooling layer.
- Visualization: t-SNE was employed to visualize the extracted features in a reduced-dimensional space. Features from the last average pooling layer formed distinct clusters for each class, indicating the model's ability to learn meaningful representations.
- Features from the mid-level convolutional layer did not form clear clusters, suggesting lower discriminative power at this layer.
- Quantification of Variances: Intra-class and inter-class variances were quantified to measure clustering quality. Features from the last pooling layer showed lower intra-class variance compared to inter-class variance, indicating effective class separation. Fine-tuning the model further reduced intra-class variance, enhancing class-specific clustering. Features from the mid-level convolutional layer exhibited higher intra-class variance, indicating less distinct class separability at this layer.

In Conlusion, the pretrained ResNet-50 model effectively captured high-level features from CIFAR-10, enabling meaningful visualizations and quantifications. Fine-tuning the model on CIFAR-10 improved feature discriminability, particularly evident

in the reduced intra-class variance and enhanced clustering. Different layers in the CNN hierarchy contribute differently to feature extraction, with higher layers capturing more abstract and discriminative information compared to lower layers.

This study highlights the importance of feature extraction and visualization techniques in understanding and evaluating deep learning models, showcasing how model architecture and dataset characteristics influence feature representation and classification performance.

6. Appendix

This appendix provides additional information about the code repository and showcases the performance enhancement achieved through fine-tuning the ResNet-50 model on CIFAR-10.

6.1 README

Project Structure:

- main.py: This file contains the main functionality of the project. It handles downloading the dataset, initializing the model (either pretrained or fine-tuned), extracting features using different layers, implementing t-SNE for visualization, calculating variances, and displaying results.
- utils.py: Contains utility functions and classes required by main.py. This includes functions for dataset handling, model initialization, feature extraction, t-SNE computation, variance calculation, visualization, and a Trainer used to fine-tune the model.
- CIFAR10 dataset: The dataset directory containing the CIFAR-10 images.
- model_state_dict.pth: Fine-tuned state dictionary of the ResNet-50 model.

Requirements

This project requires commonly used deep learning libraries including numpy, torch, seaborn, matplotlib, scikit-learn, etc.

Usage

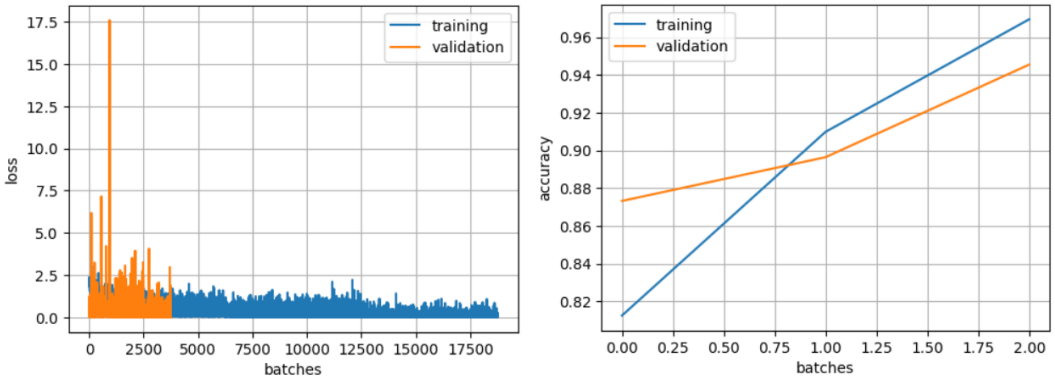
To run the project, use the following commands:

- python main.py pretrained: This uses the pretrained ResNet-50 model to extract features from the last average pooling layer, compute t-SNE visualizations, and quantify variances.
- python main.py finetuned: This first fine-tunes the ResNet-50 model on CIFAR-10 (time-consuming), then uses the fine-tuned model for feature extraction, visualization, and variance quantification.
- python main.py finetuned --load_model: Loads a provided state dictionary of the fine-tuned model and uses it for feature extraction, visualization, and variance quantification, saving time compared to fine-tuning.
- python main.py finetuned --load_model –conv: Loads the fine-tuned model and extracts features from a convolutional layer instead of the last average

pooling layer. Useful for comparing feature representations at different network layers.

6.2 Fine-tuning the model

During the fine-tuning process on CIFAR-10, the performance of the pretrained ResNet-50 model was evaluated. Here are the key results:



The model was trained for 3 epochs, and achieved 96% accuracy on training set and 94% accuracy on the validation set.