**Deep Learning Final Project -- SP 24**

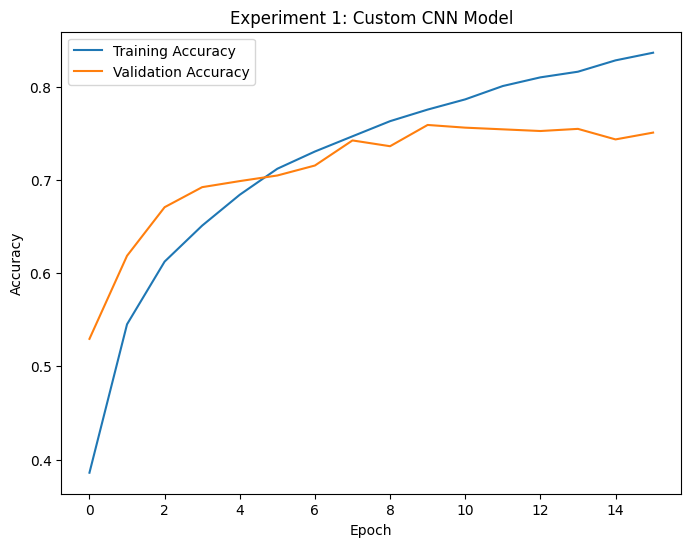
**Introduction**

We explore the utilization of pre-trained feature layers in adjacent tasks for image classification problems. We conduct a series of experiments to explore different approaches, including end-to-end classification, transfer learning, fine-tuning and assessing the potential for memory loss or functional gain. The experiments are performed on the CIFAR-10 and Tiny ImageNet datasets that consist of color images from various classes.

**Experiment Part A: Utilizing Pre-Trained Feature Layers in Adjacent Tasks**

***Experiment 1: End-to-End Classification (Color Images)***

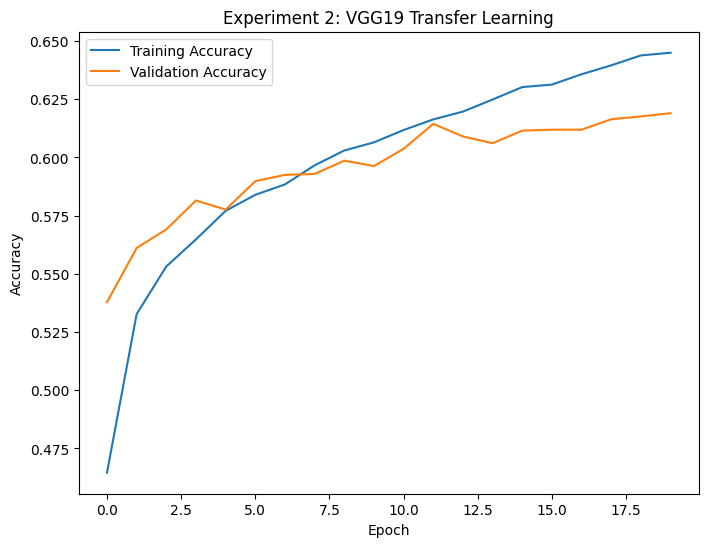
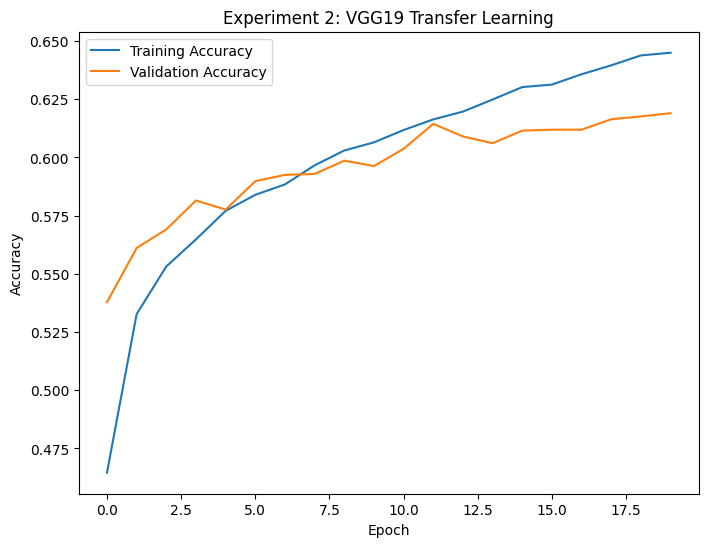
In this experiment a custom Convolutional Neural Network (CNN) model is trained from scratch on the CIFAR-10 dataset for image classification. The model architecture consists of convolutional layers, max-pooling layers, and fully connected layers with dropout regularization. The model is trained using the Adam optimizer and categorical cross-entropy loss. The training performance of the custom CNN model is visualized in the following plot, which shows the training and validation accuracy over epochs:



The final training accuracy achieved by the custom CNN model on the CIFAR-10 dataset is 0.836359977722168 and the test accuracy is 0.755999982357025.

***Experiment 2: Transfer Learning***

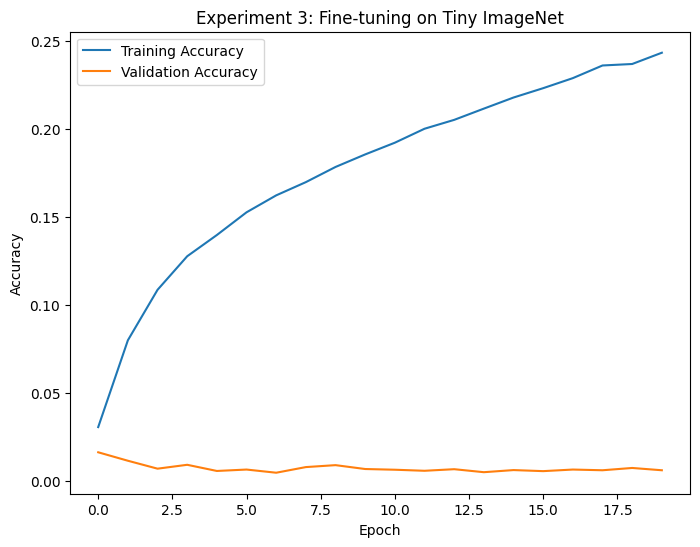
This experiment explores the use of transfer learning by leveraging pre-trained models on the ImageNet dataset. Two pre-trained models, VGG19 and ResNet50, are used. The pre-trained feature extraction layers are frozen, and custom classification layers are added for the CIFAR-10 dataset. The models are then trained on the CIFAR-10 dataset. The training performance of the VGG19 and ResNet50 models is visualized in the following plots:



* The final training accuracy achieved by the VGG19 model on the CIFAR-10 dataset is 0.644959986209869 and the test accuracy is 0.619000017642974.
* The final training accuracy achieved by the ResNet50 model on the CIFAR-10 dataset is 0.274179995059967, while the test accuracy is 0.368400007486343.

***Experiment 3: Fine-tuning***

A custom CNN model is trained from scratch on the Tiny ImageNet dataset, which consists of 200 classes. The model architecture includes convolutional layers, max-pooling layers, fully connected layers, and dropout regularization. The model is trained using the Adam optimizer and categorical cross-entropy loss. The training performance of the custom model on the Tiny ImageNet dataset is visualized in the following plot:



The final training accuracy achieved by the custom model on the Tiny ImageNet dataset is 0.243190005421638, while the test accuracy is 0.00620000017806887.

**The fine-tuned model is saved for later use in Experiment 4.**

***Experiment 4: Memory Loss or Functional Gain***

This experiment explores whether the fine-tuned model from Experiment 3 retains its performance on the original CIFAR-10 dataset. The fine-tuned model is loaded and evaluated on the CIFAR-10 test set to assess potential memory loss or functional gain.

The test accuracy of the fine-tuned model on the CIFAR-10 dataset is 0.00620000017806887.

**Discussion:**

***Compare the results of your experiments for Part A.***

**a. Classification Layer Training Performance:** The classification layer training performance for the different models is displayed in the plots provided in the report. Here are the key observations:

* Experiment 1 (Custom CNN on CIFAR-10): The custom CNN model showed a steady increase in training and validation accuracy over epochs, reaching a final training accuracy of 0.836359977722168.
* Experiment 2 (Transfer Learning on CIFAR-10): The VGG19 model exhibited a similar trend to the custom CNN, with training accuracy increasing over epochs and reaching a final training accuracy of 0.644959986209869. The ResNet50 model had a lower final training accuracy of 0.274179995059967.
* Experiment 3 (Fine-tuning on Tiny ImageNet): The custom model trained on Tiny ImageNet had a lower training accuracy of 0.243190005421638 compared to the CIFAR-10 models, likely due to the increased complexity of the dataset with 200 classes.

**b. Test Performance Comparison:** The test performance of the models is summarized in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment** | **Model** | **Dataset** | **Test Accuracy** |
| 1 | Custom CNN | CIFAR-10 | 0.755999982357025 |
| 2 | VGG19 | CIFAR-10 | 0.619000017642974 |
| 2 | ResNet50 | CIFAR-10 | 0.368400007486343 |
| 3 | Custom Fine-tuned | Tiny ImageNet | 0.00620000017806887 |
| 4 | Custom Fine-tuned | CIFAR-10 | 0.00620000017806887 |

The custom CNN model achieved the highest test accuracy of 0.755999982357025 on the CIFAR-10 dataset, followed by the VGG19 model at 0.619000017642974. The ResNet50 model performed relatively lower, with a test accuracy of 0.368400007486343. The custom fine-tuned model achieved a low test accuracy of 0.00620000017806887 on both the Tiny ImageNet and CIFAR-10 datasets, suggesting significant memory loss or functional gain when evaluated on the CIFAR-10 dataset.

**c. Potential Changes:**

* Adjusting hyperparameters: Experimenting with different learning rates, batch sizes, and optimization algorithms.
* Modifying architectures: Exploring alternative CNN architectures or adding more layers.
* Data augmentation: Applying data augmentation techniques to increase the diversity of the training data.
* Transfer learning from different pre-trained models: Experimenting with other pre-trained models like EfficientNet or MobileNet.
* Fine-tuning strategies: Exploring different fine-tuning strategies, such as unfreezing and fine-tuning specific layers.

**d. Comparison of VGG and ResNet Features:**

* The VGG19 model provided better features for training on the CIFAR-10 dataset compared to the ResNet50 model. This conclusion can be drawn from the higher test accuracy achieved by the VGG19 model (0.619000017642974) compared to the ResNet50 model (0.368400007486343).
* The performance of the VGG19 model could be attributed to its deeper architecture and the effectiveness of its learned features in capturing the relevant visual patterns in the CIFAR-10 dataset.

**Part B – Segmentation and Style Transfer**

***Experiment 1: Localization [ROI]***

YOLO (You Only Look Once) is a real-time object detection algorithm that divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell. It enables fast and accurate localization of objects in images.

Values:

Image Object Type X\_ min Y\_ min X\_ max Y\_ max Probability

image\_1.jpg car 125 143 149 162 0.72

image\_1.jpg truck 141 139 157 154 0.31

***Experiment 2 SegFormer Segmentation***

SegFormer is a transformer-based semantic segmentation model that achieves state-of-the-art performance on various benchmarks. It leverages a hierarchical structure and a lightweight decoder to efficiently capture long-range dependencies and produce high-resolution segmentation maps.

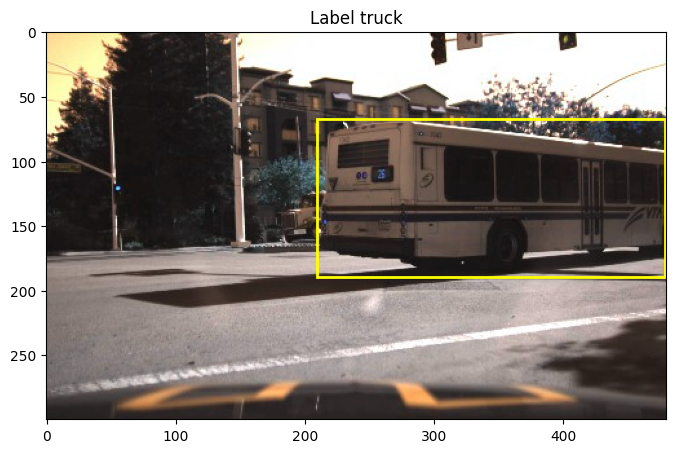
Values: IoU [0.9783214257044233, 0.964239124119974, 0.9908357640865111, ...]

***Experiment 3: Generative Models Working with Style Transfer***

Neural style transfer is a technique that uses deep neural networks to combine the content of one image with the style of another image, creating a new image that preserves the content while adopting the artistic style. It leverages the feature representations learned by convolutional neural networks to capture and transfer style information.

***Compare the results of your experiments for Part B.***

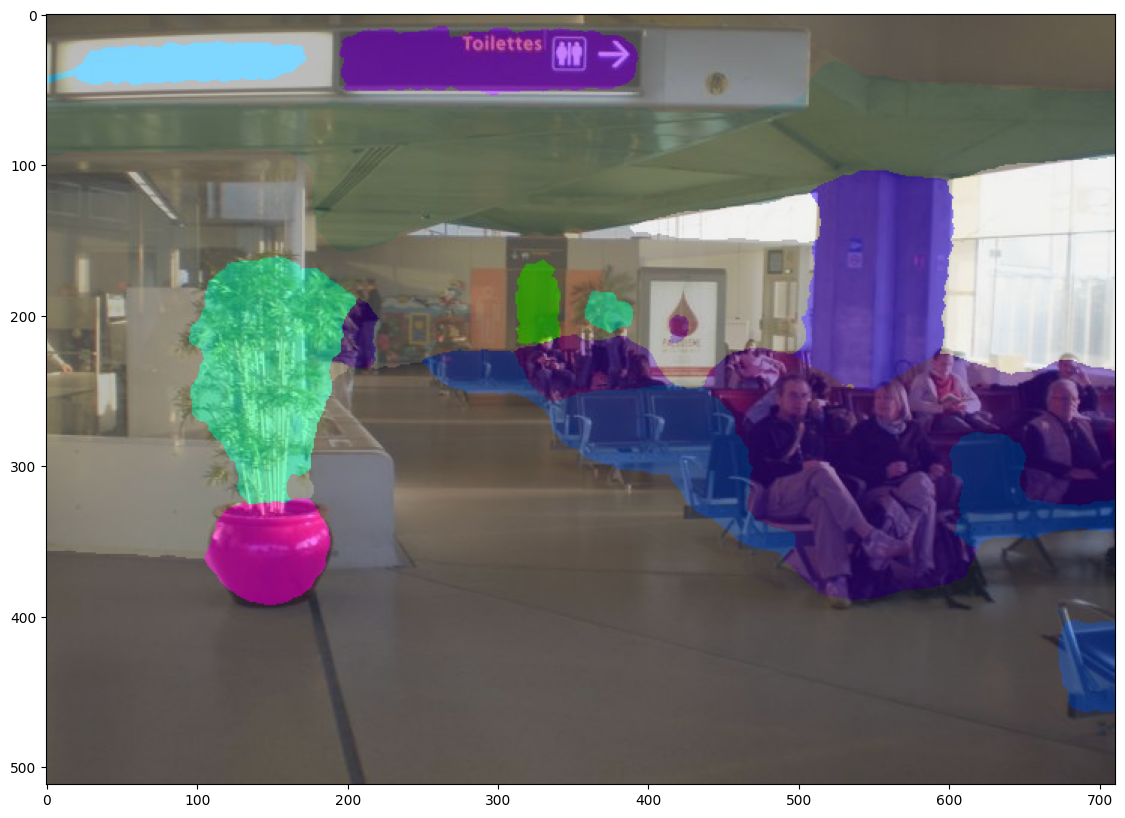
A. The YOLOv8 model is able to detect and localize objects in the given images from the self-driving car dataset. It identifies objects such as cars, trucks, pedestrians, bicycles, and traffic lights, and provides their bounding box coordinates and confidence scores. The model seems to perform reasonably well on these types of images, although the confidence scores vary for different detections.





B . Image segmentation using the SegFormer model

* It uses the SegformerImageProcessor from Hugging Face to preprocess the images and segmentation maps, including reducing the number of labels.
* It creates instances of the SemanticSegmentationDataset for the training and validation sets, and wraps them in PyTorch DataLoader objects for efficient batching during training.
* It loads the SegFormer model (SegformerForSemanticSegmentation) and the label mapping (id2label and label2id) required for the ADE20K dataset, which contains 150 classes.
* It defines an optimization loop to train the SegFormer model on the training dataset. During each epoch, it iterates over the batches, computes the loss, performs backpropagation, and updates the model parameters using the AdamW optimizer.





C. The SegFormer model is designed for semantic segmentation that involves understanding and segmenting individual objects and structures within an image. While it captures geometric information implicitly, it is not primarily focused on understanding geometric structures in the same way as models specifically designed for that task.

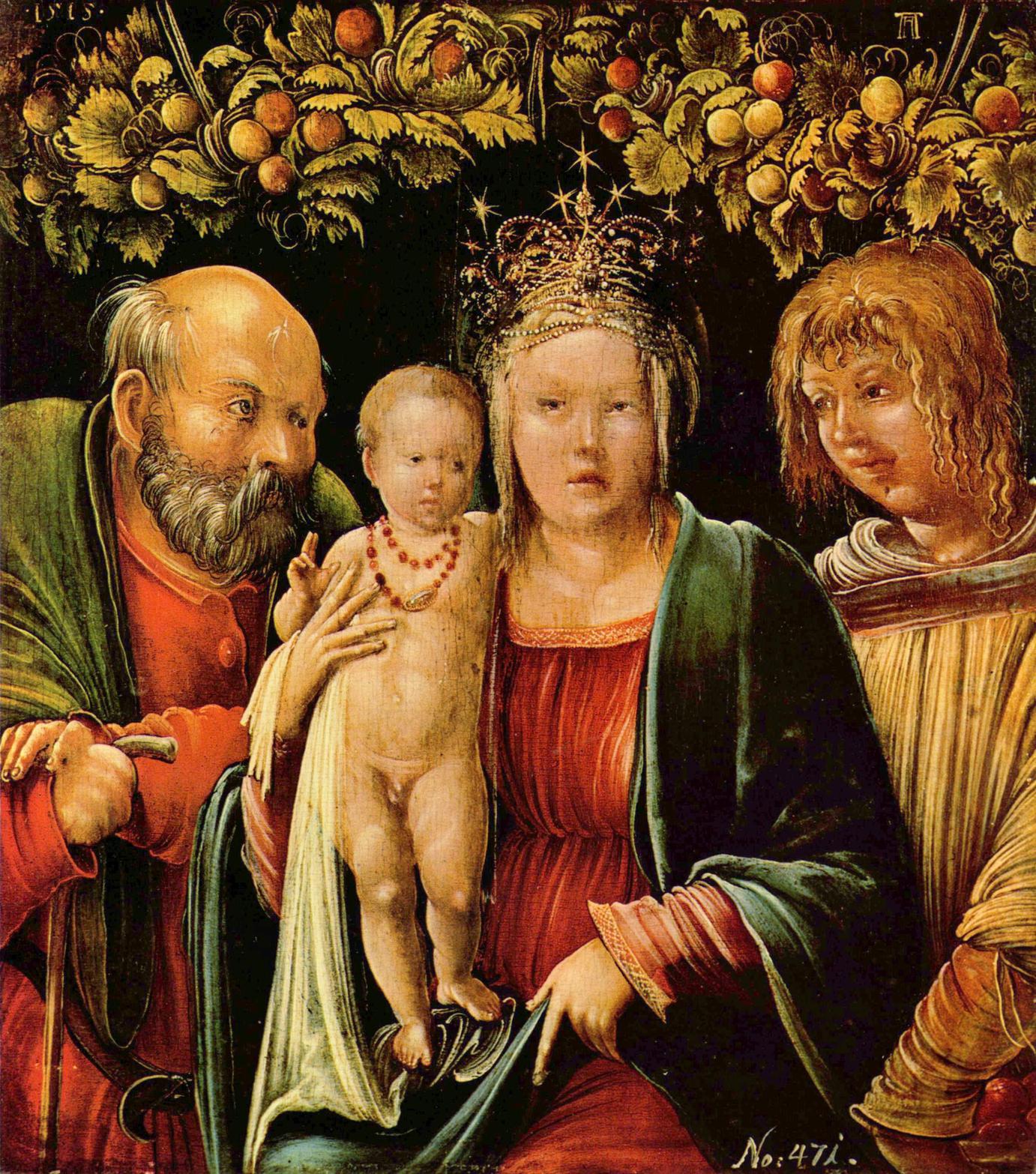
D. Semantic segmentation models like SegFormer should be capable of handling overlapping objects to some extent, as they are trained to segment individual instances of objects. Their performance may degrade in cases of severe occlusion or highly overlapping objects, depending on the complexity of the scene and the training data.

E. Hugging Face models work best when the content image has distinct structural elements (e.g., buildings, landscapes, objects) and the style image has a recognizable and consistent artistic style or texture pattern. As for the utility of style transfer methods, they have several applications in areas such as:

1. Artistic rendering: Style transfer can be used to create artistic renditions of ordinary images, potentially enabling new creative avenues for artists and designers.
2. Image enhancement: Style transfer can be used to enhance the visual appeal of images by applying desired artistic styles or textures.
3. Image editing: Style transfer can be used as a tool for image editing, allowing users to apply artistic styles or textures to specific regions or objects within an image.







**3.** The experiments conducted in this report demonstrate the practical application of various deep learning techniques covered in the course. The custom CNN model (Experiment 1) showcases the implementation of convolutional neural networks for image classification tasks, which is one of the fundamental concepts in deep learning. The transfer learning experiments (Experiment 2) highlight the concept of leveraging pre-trained models and their learned features on a different dataset or task. This approach aligns with the course's emphasis on efficient model development and the importance of transfer learning in practical applications. The fine-tuning experiment (Experiment 3) illustrates the process of adapting a pre-trained model to a new dataset or task by fine-tuning the model's parameters. This technique is essential for achieving better performance on specific tasks while leveraging the knowledge gained from larger datasets. The memory loss or functional gain experiment (Experiment 4) explores an important aspect of model generalization and the potential trade-offs when adapting models to different domains or tasks. These experiments reinforce the practical application of deep learning concepts and techniques, allowing for a deeper understanding of their strengths, limitations, and potential improvements.

**4. Deep Learning has provided us with several amazing advancements over the past few years, take a few minutes to consider what might be next in the field.**

a. Based on the experiments and the course content, I believe I would spend more time exploring transfer learning and fine-tuning techniques. These approaches have proven to be effective in leveraging pre-trained models and adapting them to specific tasks.

b. Deep Learning can be considered an example of emerging intelligence in machines as it allows models to learn complex patterns and representations from data, enabling them to perform tasks that were previously challenging for traditional algorithms.

c. One constructive suggestion for future students in this course would be to incorporate more hands-on projects or case studies that simulate real-world scenarios. This approach could involve working with diverse datasets, exploring different domains, and tackling challenges that may arise in practical applications of deep learning.