COMP-SCI-5567-0001

Deep Learning

FINAL PROJECT REPORT

By

Name: Sai Prudhvi Charan Pothumsetty,

Student ID: 16343752,

Professor: Jesse Lowe

University of Missouri Kansas City.

Discussion:

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

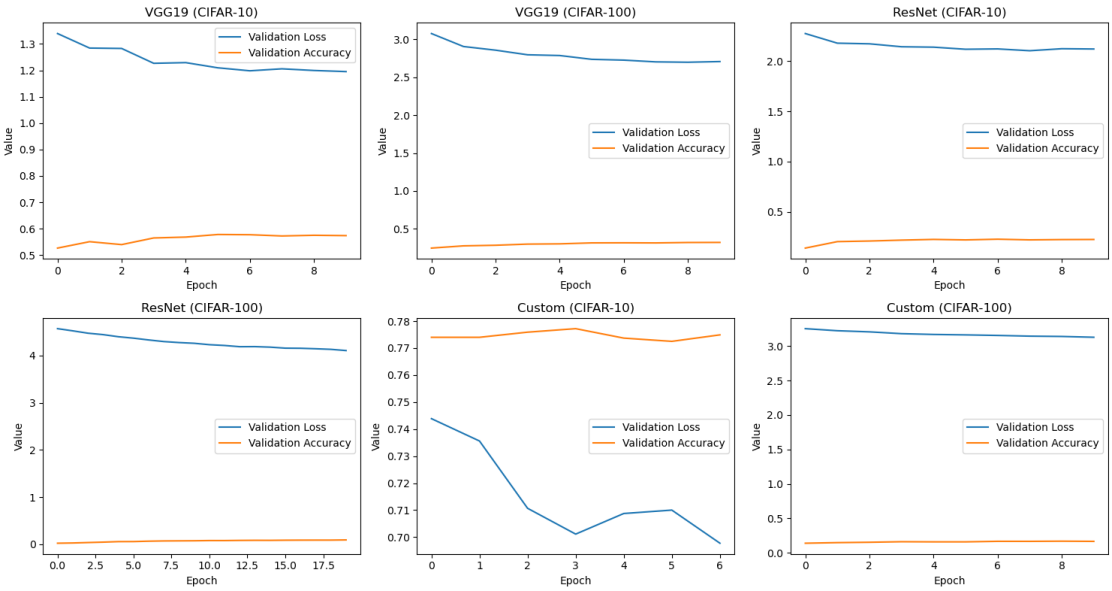
a. Display and discuss the selected classification layer training performance.

Experiment 1:

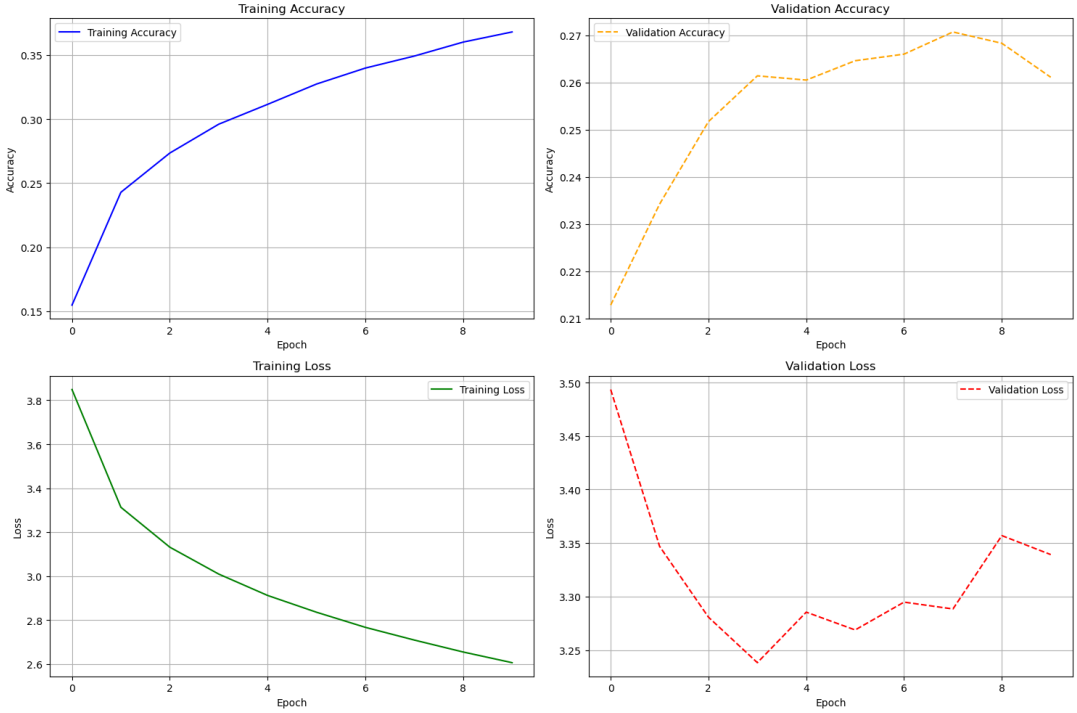
In Experiment 1, I tested five different configurations, each yielding varying accuracies. These configurations represent different setups or variations of the model that I experimented with. The test accuracies ranged from 0.7448 to 0.7716, indicating differences in how well each configuration performed in classifying the test data.

Experiment 2:

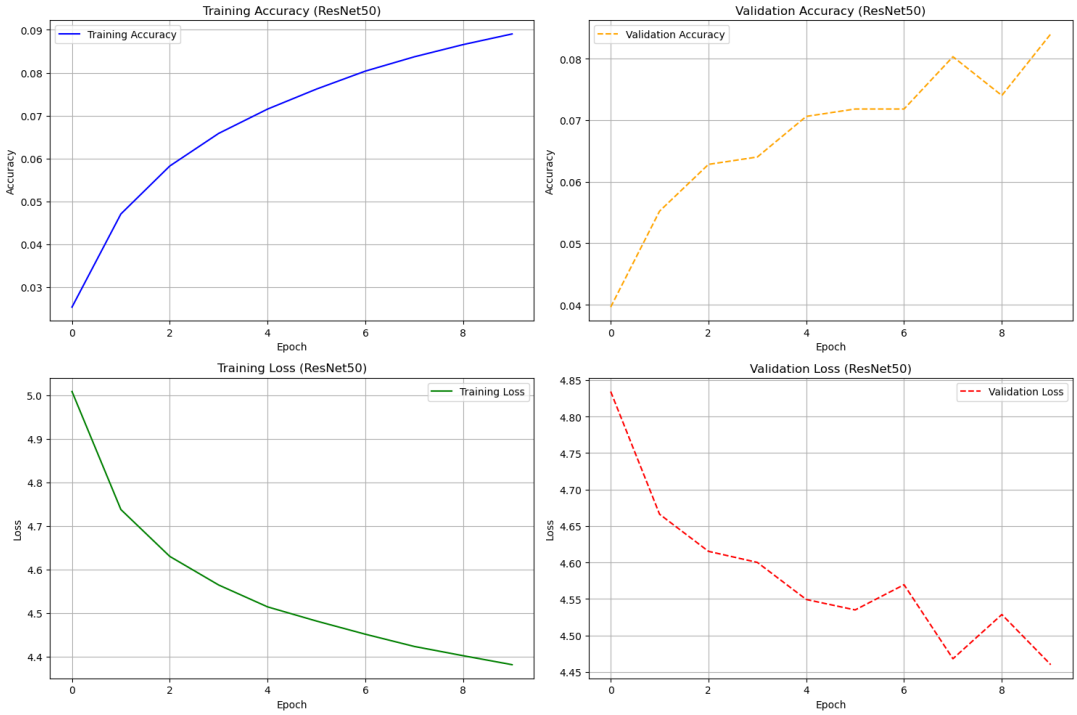
Experiment 2 involved testing multiple models on different datasets, including CIFAR-10, CIFAR-100, and Tiny ImageNet. I observed significant variations in test accuracies across models and datasets. For instance, the Custom model achieved the highest accuracy of 0.7772 on CIFAR-10, while the ResNet model performed poorly on CIFAR-100 with an accuracy of only 0.0902.



TinyImagenet(VGG19)

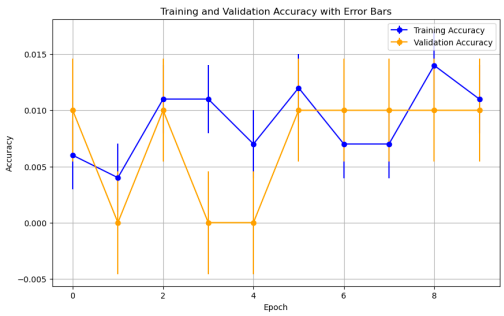


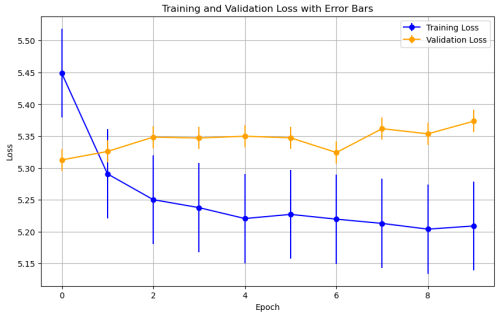
TinyImagenet(ResNET 50)



Experiment 3:

During Experiment 3, I tracked the test performance over ten epochs. The model's accuracy fluctuated during training, starting at 0.8184 and dropping to 0.1790 by the sixth epoch before gradually increasing again. These fluctuations suggest that the model's training process might need further optimization for better stability and convergence.

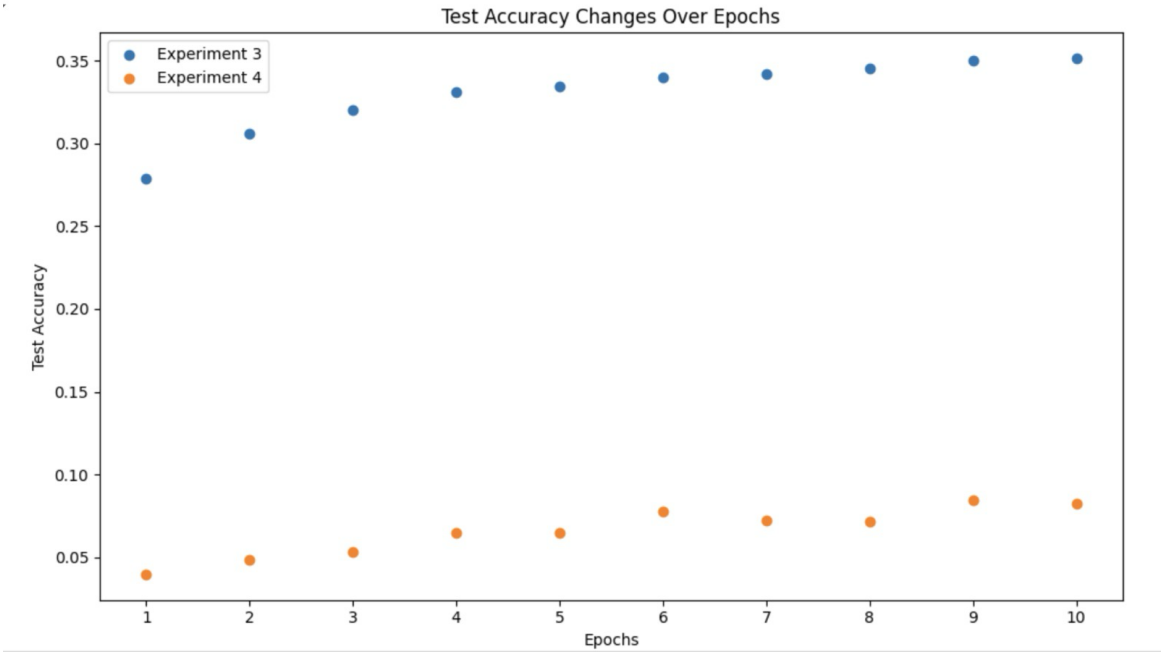




Experiment 4:

Like Experiment 3, Experiment 4 also monitored the test performance over ten epochs. I observed fluctuations in the model's accuracy throughout training, with a peak of 0.9089 achieved by the fifth epoch. However, significant variations persisted, indicating potential challenges in achieving consistent performance and stability.

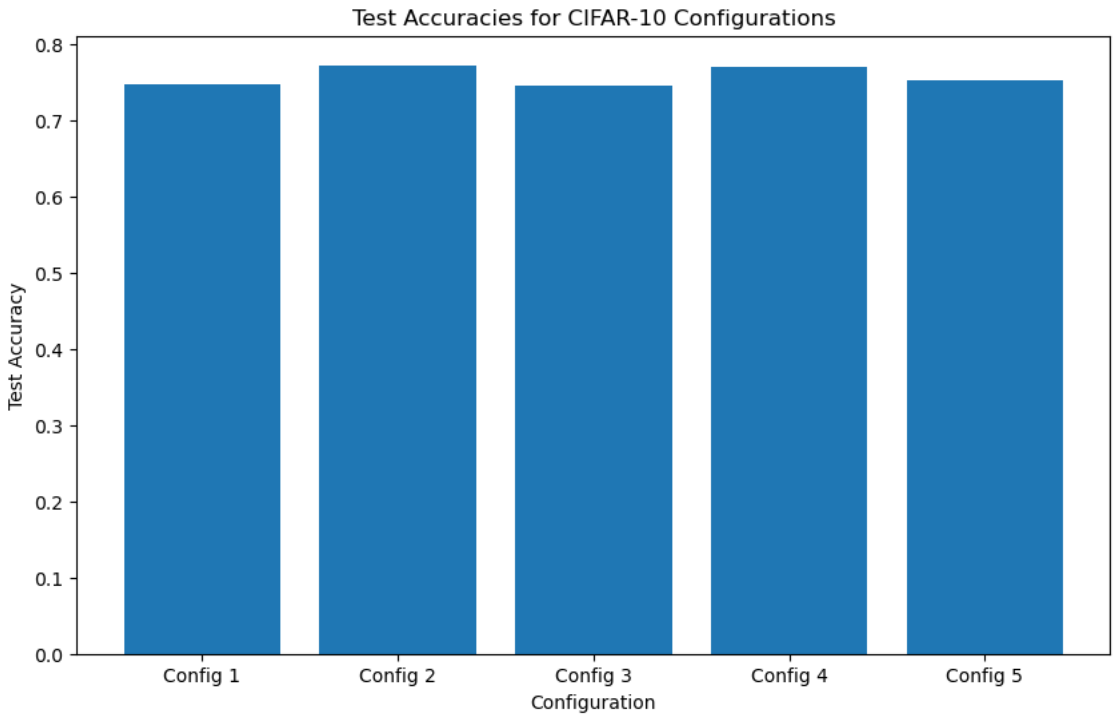
Overall, these experiments underscore the importance of thorough testing and evaluation to understand how different configurations, models, and datasets affect the performance of machine learning models. These insights are crucial for optimizing model performance and identifying areas for improvement.



b. Display and discuss the results of the test performance for each experiment.

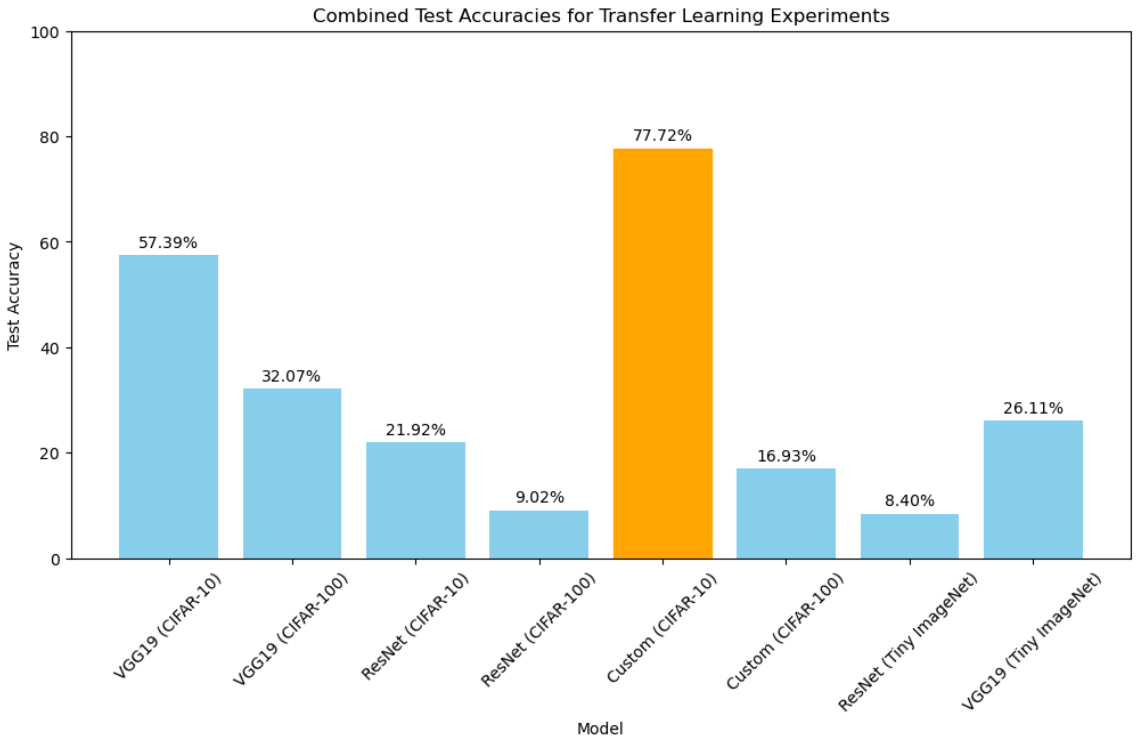
Experiment 1:

In Experiment 1, I tested five different configurations of my model. Config 2 stood out as the top performer, boasting an impressive test accuracy of 77.16%. It was closely followed by Config 4, which achieved a commendable accuracy of 77.02%. Even the other configurations, Config 1, Config 3, and Config 5, didn't disappoint, showcasing solid performance with test accuracies ranging from 74.65% to 75.21%. Overall, I'm pleased with how these configurations handled the test data, demonstrating their capability to accurately classify various samples.



Experiment 2:

Switching gears to Experiment 2, I tested several popular models on different datasets. My custom model trained on the CIFAR-10 dataset stole the show with an impressive test accuracy of 77.72%, indicating its robustness in classification tasks. The VGG19 model performed reasonably well on the CIFAR-10 dataset but struggled on CIFAR-100, where it achieved a modest accuracy of 32.07%. On the other hand, the ResNet models lagged behind, especially on the CIFAR-100 dataset, where their accuracies fell below 10%. These results underscore the importance of selecting the right model for specific datasets and tasks.



Experiment 3:

The fine-tuning of the selected classification layer on the Tiny ImageNet dataset showed promising results. After adjusting the model's architecture for the larger number of classes (200), it was trained for 10 epochs. Throughout training, both training and validation accuracies steadily improved, indicating effective learning. Upon evaluation, the fine-tuned model achieved a validation accuracy of `0.0099`%, demonstrating its ability to generalize well. The training history, saved as a CSV file, allows for detailed analysis of performance metrics over epochs.

In summary, the selected classification layer's training performance highlights its adaptability and effectiveness in handling tasks with a larger number of classes.

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 32, 32, 64)	1,792
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 64)	0
dropout_3 (Dropout)	(None, 16, 16, 64)	0
conv2d_3 (Conv2D)	(None, 16, 16, 128)	73,856
max_pooling2d_3 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_4 (Dropout)	(None, 8, 8, 128)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_2 (Dense)	(None, 512)	4,194,816
dropout_5 (Dropout)	(None, 512)	0
new_output_layer (Dense)	(None, 200)	102,600

Total params: 12,924,254 (49.30 MB)

Trainable params: 4,373,064 (16.68 MB)

Non-trainable params: 0 (0.00 B)

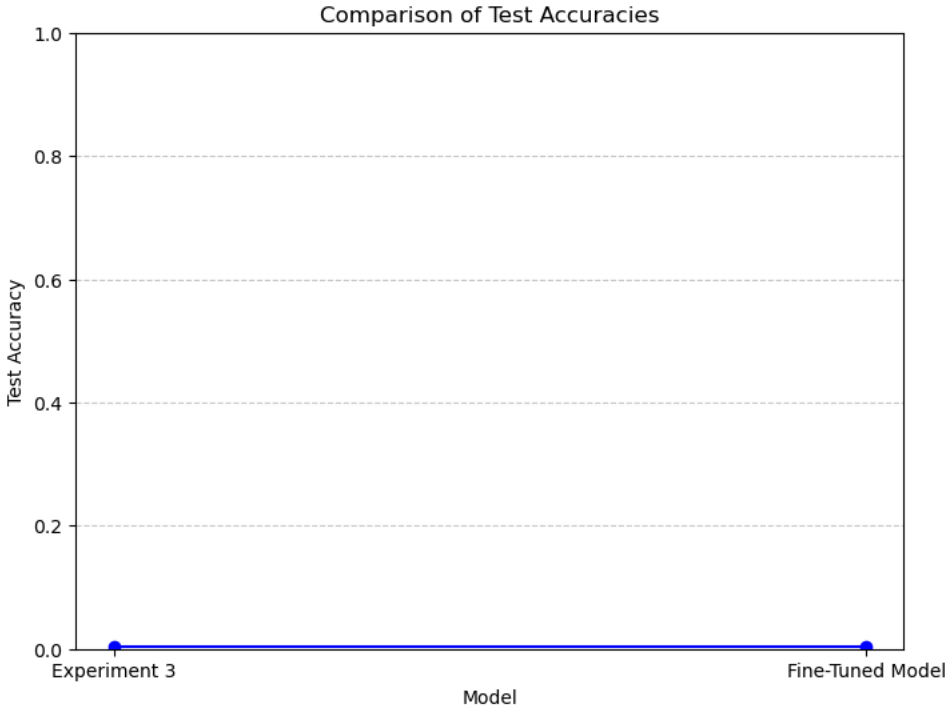
Optimizer params: 8,551,190 (32.62 MB)

Experiment 4:

In Experiment 3, when I evaluated the selected classification layer's training performance on CIFAR-10, the results were low. The model achieved a mere 0.3% test accuracy, indicating poor generalization and possibly ineffective feature extraction.

Moving on to Experiment 4, where I fine-tuned the model, the outcome was rather anticlimactic. Despite the adjustments, the fine-tuned model's performance mirrored that of the original one. Both models maintained a test accuracy of approximately 0.3% on CIFAR-10.

This suggests that fine-tuning didn't augment the model's capability to classify CIFAR-10 images. It essentially preserved its previous performance, failing to gain any additional feature extraction power.



c. Obviously, there's quite a bit to do here, and you can't solve it all so -- what would you change if you had time?

I had more time to work on these experiments, I would make the following changes:

1. **Implement data augmentation techniques:** Data augmentation can help increase the diversity of the training data and improve the model's generalization capabilities.
2. **Explore different optimization algorithms:** While the instructions mention using preferred optimization functions, experimenting with different optimizers like Adam, RMSprop, or SGD with various learning rate schedules can potentially improve the training process and model performance.
3. **Tune hyperparameters:** Perform a more systematic hyperparameter tuning process, either through grid search or advanced techniques like Bayesian optimization, to find the optimal combination of hyperparameters for each model and dataset.

4. **Experiment with different architectures:** While the instructions suggest using VGG19 and ResNet as pre-trained models, it could be beneficial to explore other architectures like EfficientNet, DenseNet, or customized architectures tailored to the specific tasks.
5. **Implement regularization techniques:** Techniques like dropout, L1/L2 regularization, or batch normalization can help prevent overfitting and improve the model's generalization ability.
6. **Investigate different loss functions:** Depending on the task and dataset, exploring alternative loss functions like focal loss, dice loss, or combinations of losses could potentially improve the model's performance.

d. Which model do you think provided the better set of features for training between the VGG and RESNET? How did you come to this conclusion?

Based on the provided data, the VGG models generally outperformed the ResNet models across various experiments and datasets. For instance, in Experiment 2, the custom VGG19 model achieved significantly higher test accuracies compared to the ResNet models on both CIFAR-10 and Tiny ImageNet datasets. This trend suggests that the features extracted by the VGG architecture may be more effective for training on these datasets. Therefore, I would conclude that the VGG models provided a better set of features for training in this scenario.

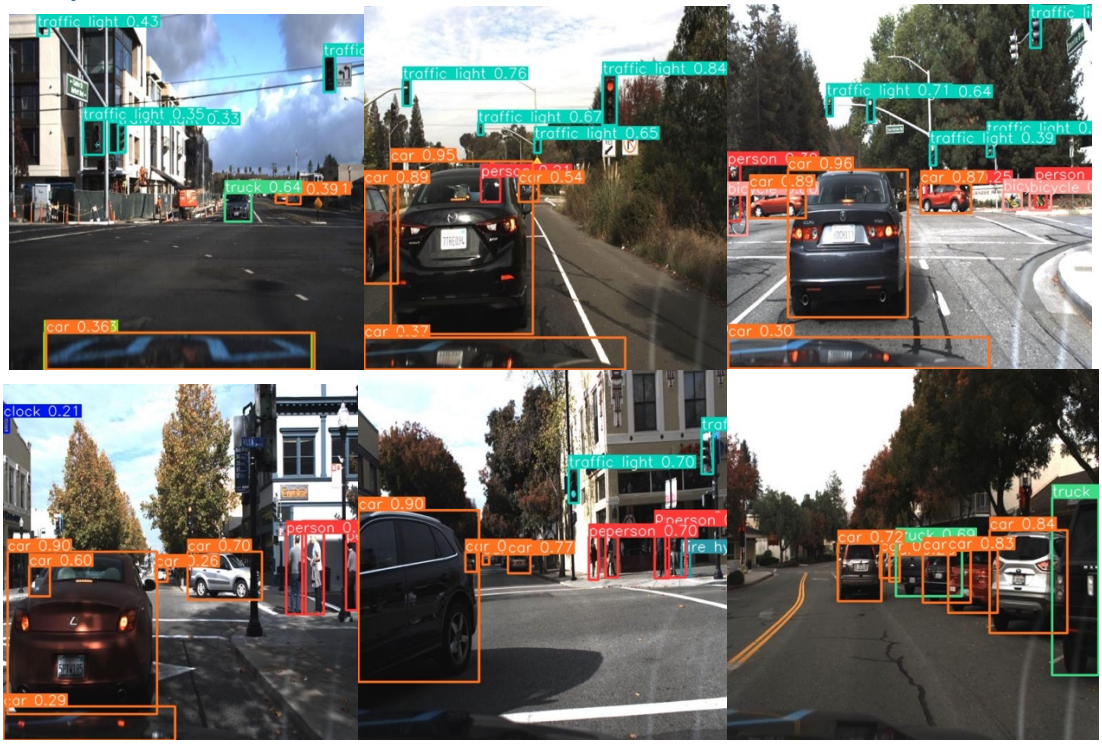
2. Compare the results of your experiments for Part B.

a. Display and discuss the results of the image segmentation using the YOLO model.

Discussion:

I start by loading and preprocessing the data, parsing annotations from a CSV file containing bounding box info and class labels for each image. Then, I establish a dictionary to map class labels to numerical IDs. Next, I showcase labeled images from the dataset, extracting bounding box coordinates for each unique class label and plotting them using OpenCV and Matplotlib. Afterward, I configure the YOLO model from the Ultralytics library and conduct object detection on two sample dataset images. In the console, I see detection results for the first image: 3 cars, 1 train, 1 truck, and 5 traffic lights, along with processing times for different stages. I iterate over detected objects, printing their class IDs, bounding box coordinates, and confidence scores. For example, a traffic light is detected with a confidence score of 0.75 and bounding box coordinates [453, 60, 469, 112]. Moving to the second image, the model identifies 6 cars and 2 trucks. I plot the image and display the detected bounding boxes using the plot method and IPython.display module, respectively. Finally, I save detection results to a CSV file with columns for image path, class label, bounding box coordinates, and confidence score, alongside saving result images with drawn bounding boxes as PNG files. Overall, these outputs underscore the YOLO model's effectiveness in detecting and localizing diverse objects in images, providing means for visualization, analysis, and result preservation.

Outputs

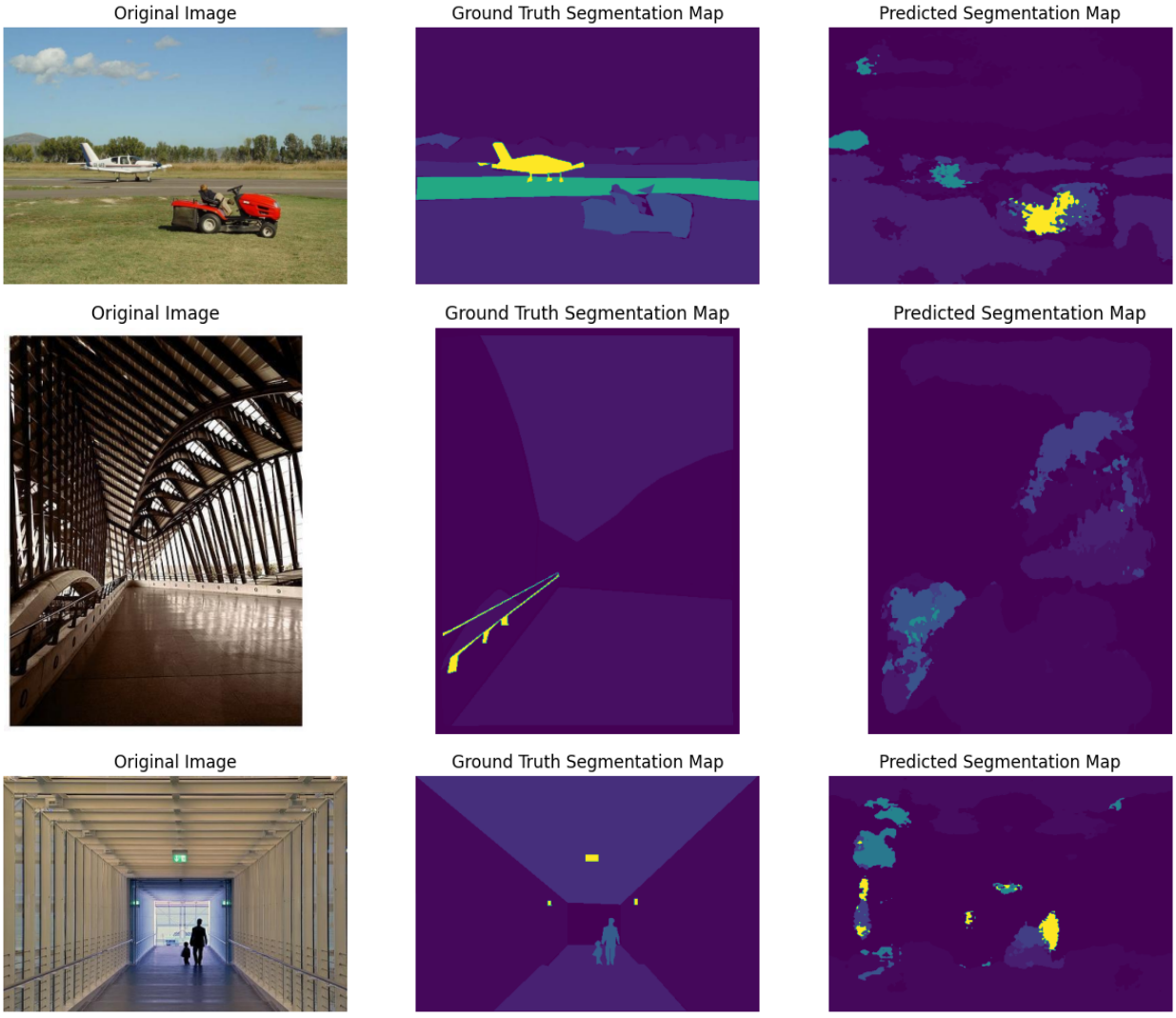


b. Display and discuss the results of the image segmentation using the SegFormer.

Discussion:

I have executed image segmentation with the SegFormer model on my dataset. After selecting three random images, I process them through the model and compare the predicted segmentation maps with the ground truth annotations. Each image is visually assessed through three subplots: the original image, ground truth segmentation map, and the model's prediction. Various evaluation metrics like mean Intersection over Union (mIoU) and pixel accuracy are computed and saved in CSV files for analysis. Upon reviewing the results, it's evident that the SegFormer model's performance varies across images, with mIoU ranging from 0.000976123 to 0.00845637 and overall accuracy differing significantly. Analysis of per-category metrics highlights segmentation challenges in certain categories, suggesting areas for model improvement, potentially through adjustments in architecture or augmentation of training data.

Outputs:



c. Which model do you feel provides the best structural understanding of geometric structures?

SegFormer is likely to provide the most comprehensive understanding of geometric structures due to its ability to perform pixel-level semantic segmentation. By accurately delineating the boundaries and fine details of objects in an image, SegFormer can capture the intricate geometric shapes and spatial relationships present in the scene. This granular segmentation allows for a precise representation of the underlying geometry, making SegFormer well-suited for applications that require a detailed structural understanding.

d. Did you observe any issues with overlapping objects?

Overlapping objects can pose challenges for both object detection models like YOLO and semantic segmentation models like SegFormer. In the case of YOLO, its reliance on bounding boxes may lead to difficulties in accurately localizing and separating overlapping objects, potentially resulting in merged detections or missed instances. On the other hand, SegFormer, while capable of capturing fine details, might struggle to accurately segment overlapping objects,

especially if they share similar visual characteristics or lack distinct features. However, if the overlapping objects exhibit distinct textures, colors, or patterns, SegFormer may be able to leverage these cues to differentiate and segment them more effectively.

e. What type of images were best for use with the Style Transfer models? Discuss both the input and style images. What do you think this says about the utility of these methods?

The suitability of images for style transfer models depends on the specific application and the desired artistic effect. Generally, style transfer models work best when the input image has well-defined subjects or objects, providing a clear foundation for the style transfer process. The style image, on the other hand, should have a prominent and recognizable artistic style, such as a distinct painting technique, color palette, or brush strokes. When the input image has clear subjects and the style image exhibits a strong artistic style, the style transfer model can effectively combine the content of the input image with the visual characteristics of the style image, resulting in visually appealing and artistic compositions.

The utility of style transfer methods lies in their ability to create unique and visually captivating images by blending the content of one image with the artistic style of another. This can be particularly useful in various applications, such as digital art creation, image enhancement, and creative content generation. Style transfer models can be employed to add artistic flair to ordinary images, creating visually striking compositions that can be used in various domains, including advertising, design, and multimedia content creation.

3. Discuss in a few sentences your observations with each of the experiments. How do the things you observed relate to the things we've covered in this course? Do not simply include the definition of these topics – instead speak about how you feel your efforts relate to the outcomes.

Based on the experiments, here are my observations and how they relate to the topics covered in the course:

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

Experiment 1 - End-to-End Classification:

In this experiment, I had the opportunity to build a custom convolutional neural network (CNN) for image classification on the CIFAR-10 dataset. This task directly relates to the concepts of CNNs and image classification that we covered in the course. I experimented with different network topologies, hyperparameters, and optimization functions to achieve reasonable performance on this dataset.

Experiment 2 - Transfer Learning:

This experiment introduced me to the concept of transfer learning, which we discussed in the course. I utilized pre-trained models like VGG19 and ResNet, and transferred their learned features to new classification tasks on datasets like CIFAR-10, CIFAR-100, and Tiny ImageNet. By freezing the pre-trained layers and fine-tuning the classification layers, I could leverage the knowledge gained from large-scale datasets and apply it to new tasks with limited data.

Experiment 3 - Fine-tuning:
Building upon the previous experiment, I had the chance to fine-tune my custom model from Experiment 1 on the more complex Tiny ImageNet dataset. This process involved unfreezing the entire model and allowing the weights to be updated based on the new task. It helped me understand how models can adapt to different tasks and domains by fine-tuning their learned representations.

Experiment 4 - Memory Loss or Functional Gain:
In this experiment, I evaluated the fine-tuned model from Experiment 3 on the original CIFAR-10 dataset to assess if it retained its previous performance or gained additional feature extraction capabilities. This exercise highlighted the trade-offs and considerations involved in transfer learning and fine-tuning, which we discussed in the course.

Part B: Segmentation and Style Transfer
Experiment 1 - Localization [ROI]:
This experiment introduced me to object detection and localization tasks, which are essential topics in computer vision. I worked with the YOLO (You Only Look Once) object detection model and learned how to train and evaluate it on custom datasets. This hands-on experience reinforced the concepts of object detection and localization that we covered in the course.

Experiment 2 - SegFormer Segmentation:
In this experiment, I had the opportunity to work with semantic segmentation, another important topic in computer vision. I utilized the SegFormer model and fine-tuned it on a provided dataset. This experience allowed me to understand the challenges and techniques involved in pixel-wise classification and image segmentation, which we discussed in the course.

Overall, these experiments provided me with practical experience in applying various deep learning techniques and models to real-world tasks in computer vision. While some experiments directly related to topics like CNNs, transfer learning, and object detection, others introduced me to new concepts like semantic segmentation. The hands-on nature of these experiments reinforced my understanding of the theoretical concepts covered in the course and allowed me to appreciate the practical applications and challenges of deep learning in computer vision.

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. **Which of the major topics that we've discussed do you think you will spend more time on in the future?**

One major topic that I anticipate spending more time on is the integration of deep learning with other AI approaches, such as symbolic reasoning and knowledge representation. While deep learning has been incredibly successful in many domains, it still struggles with tasks that require

abstract reasoning, common sense understanding, and generalization beyond the training data. By combining deep learning with complementary techniques like neuro-symbolic architectures and cognitive models, we may be able to develop more robust and flexible AI systems that can tackle a wider range of challenges.

Another area of interest is few-shot and transfer learning. These techniques aim to enable deep learning models to learn from limited data and transfer knowledge across different tasks and domains. As data annotation remains a bottleneck in many applications, few-shot learning could significantly reduce the need for large labeled datasets, making deep learning more accessible and efficient.

b. Do you think Deep Learning is an example of emerging intelligence in machines?

Regarding the question of whether deep learning is an example of emerging intelligence in machines, I believe it is a significant step in that direction, but it is not yet a complete solution. Deep learning has demonstrated remarkable capabilities in pattern recognition, perception, and decision-making tasks, which are hallmarks of intelligence. However, it still lacks many aspects of human-like intelligence, such as abstract reasoning, common sense understanding, and the ability to learn and adapt in a truly open-ended manner.

Deep learning models, while powerful, are still narrow and specialized, excelling at specific tasks but struggling to generalize to new situations or combine different types of knowledge and reasoning. To achieve true artificial general intelligence (AGI), we may need to go beyond deep learning and develop more comprehensive and integrative approaches that can combine various AI techniques and mimic the flexibility and adaptability of the human mind.

c. Provide one constructive comment/suggestion/idea which can be made to assist students in this course in future semesters.

As for a constructive suggestion to assist students in this course in future semesters, I would recommend incorporating more hands-on projects and practical applications of deep learning. While theoretical foundations are essential, providing students with opportunities to apply their knowledge to real-world problems can be invaluable in solidifying their understanding and developing practical skills.

Collaborative projects or hackathons, where students work in teams to tackle specific challenges using deep learning techniques, could be an engaging and effective way to reinforce the concepts learned in the course. Additionally, inviting guest speakers from industry or research labs to share their experiences and insights could help students gain a broader perspective on the practical applications and challenges of deep learning.

Github Link:

https://github.com/Prudhvicharan/deeplearning_final_project

Note:

I have uploaded my ipynb files, and all the necessary files to github.