Exploring Convolutional Neural Networks (CNNs)

The objective of this assignment covers three key aspects, aimed at providing you with comprehensive insights into Convolutional Neural Networks (CNNs) and their practical applications:

1. **Training a CNN from Scratch and Hyperparameter:** Tuning Your first task is to construct and train a CNN model from the ground up. This will involve fine-tuning the model's hyperparameters and gaining proficiency in visualizing filters. Through this, you'll grasp the nuances of model architecture and optimization.

2. **Tansfer Learning a Pre-trained Model:** In the context of real-world scenarios, reusing pre-trained models for specific tasks is a common practice. Your second challenge is to finetune an existing pre-trained model, to practice utilizing a pre-trained neural network as a starting point for training a new model to perform different object detection tasks.

3. **Using model from step 2:** Lastly, use the power of an established pre-trained model to embark on an exciting application. By integrating a pre-trained model into your project, you'll explore the potential of advanced applications without starting from scratch.

As you navigate through these tasks, you'll not only solidify your understanding of CNNs but also gain valuable experience in the intricacies of model training, adaptation, and application.

Problem 1. *Design a compact CNN model comprising five convolutional layers, with each layer connected to a ReLU activation function and a subsequent max pooling layer. Your final network should encompass five such blocks, resulting in a dense layer followed by an output layer with 10 neurons, aligning with the 10 available classes. Ensure the input layer's compatibility with the Fasion-MNIST dataset images. The provided code should offer the flexibility to adjust the number of filters, filter sizes, and activation functions within each layer. It should also allow customization of the dense layer's neuron count. Utilize Python, including numpy and pandas, for your implementation.*

Feel free to employ packages from Keras, PyTorch, and TensorFlow, and run your code on a GPU-equipped Jupyter Notebook via Colab. Your task entails the following:

Steps:

1. *Implementation: In your report, document the implementation of this model.*

2. *Performance Report: In your report, provide a detailed report on how well this model classifier performs on the test dataset. Analyze and discuss the results, considering its accuracy and any insights gained from its performance as a baseline.*
3. *What needs to be changed to make this model better?*

Problem 2. *Object Detection Transfer Learning and New Class Addition*

You are given the task of expanding the capabilities of an existing Object Detection model trained on a previous dataset like COCO by incorporating new classes into the detection model. Follow these steps to successfully accomplish the assignment:

Steps:

1. *Acquire Pre-trained Model: Begin by obtaining a pre-trained YOLO model that was trained on the COCO dataset. This will serve as your starting point for transfer learning.*
2. *Select and Prepare New Classes: Choose a new object class that you intend to add to the COCO dataset. These class should be distinct from the existing ones. Collect or create a dataset containing images and annotations for these new classes (this will take some time, try to find good tools to help you with this step, a minimum of 100 images is required).*
3. *Update Model Architecture: Modify the output layer of the pre-trained object detection model - I recommend YOLO model of any version - to accommodate the new classes.*
4. *Transfer Learning: Utilize the pre-trained model as a base and perform transfer learning on your new dataset. Train the modified model using the new dataset while keeping the weights of the pre-trained layers fixed. This helps the model adapt to the new classes without losing its prior knowledge.*
5. *Evaluate Performance: Assess the performance of your fine-tuned model on a validation dataset. Report on accuracy, precision, recall, and other relevant metrics.*
6. *Visualization: Present visual examples of your model's detection results on both the original COCO classes and the newly added classes.*
7. *Comparison and Analysis: Reflect on the effectiveness of the transfer learning approach and the challenges encountered during the process. Discuss how the model performs on the new classes compared to the original COCO classes.*
8. *Extensions (Optional): For an extra challenge, explore strategies to improve the model's performance, such as data augmentation, hyperparameter tuning, or architecture adjustments.*

In your final report, provide code snippets, visualizations, tables, and detailed explanations for each step. Highlight the enhancements made to the COCO dataset and the model's adaptability to new classes through transfer learning.

Problem 3. *Building an Application for New Class Detection using a Trained Model*
Your task is to create an application that showcases the detection capabilities of the newly trained model, focusing specifically on the new class. You have the flexibility to choose between three options: a simple Python application utilizing the webcam, a website, or an Android application utilizing TensorFlow Lite or server-side detection. Follow these steps to successfully complete the assignment:

Steps:

1. *Choose Application Type: Decide whether you want to build a simple Python application using the webcam, a web-based application, or an Android application.*
2. *Integration of Trained Model: Integrate your trained model into the chosen application type. The model should be able to detect and highlight instances of the newly added class accurately.*
3. *User Interface (UI): Design a user-friendly interface for your application. Ensure that the detection results are clearly presented to the user in a visually comprehensible manner.*
4. *Real-time Detection: If applicable to your chosen application type, ensure that the detection process occurs in real-time, allowing users to experience the detection capabilities on the fly.*
5. *Testing and Validation: Thoroughly test the application using a variety of inputs. Verify that the detection of the new class is reliable and consistent. Use both positive and negative examples to ensure the model's accuracy.*
6. *Report and Documentation: Compile a report detailing your application development process. Provide screenshots or images illustrating how the application works. Describe any challenges you encountered and the strategies you used to overcome them.*
7. *Submission: Submit your completed application along with the report describing your implementation process and showcasing the new class detection functionality.*