First Submission due on Sunday, March 30th at 11:59 pm
Answer all problems following the instructions. You must use the Latex template for your solutions. Please compile the Latex template and submit the PDF report to Canvas.

Instruction Guidelines:

Please use the Latex template for your solutions. Presentation is very important (and worth 10% of the grade). For every problem, you will write up your solution, assign yourself a numerical grade **with reasons/justification for the grade**, and acknowledge any web sources, classmate help that you used in this assignment. Note: ChatGPT usage is allowed for problems, but you are responsible for ensuring the validity of its solutions and justifications. Also note that writing of the report can also use genAI tools, but your reasons/justifications/explanations in the report should be backed up with evidence (figures, numerical calculations, etc.) wherever possible.

Problem 1. *(Image Classification with PyTorch)* (20 points)
This problem is designed to familiarize you with PyTorch, the main software package we will be using in this class for training deep learning models.

1. **PyTorch 60-Minute Blitz**

   Go through the PyTorch 60-Minute Blitz tutorial from `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html` and complete the sections on **Tensors**, **A Gentle Introduction to torch.autograd**, and **Neural Networks**. These sections will introduce you to the basic concepts of PyTorch, which are crucial for understanding the tasks below.

2. **Investigating Bias in Image Classification** (10 points)

   Try the experiment of dataset bias featured in `https://arxiv.org/pdf/1911.11834.pdf` Create two datasets: (1) Take five of the ten classes, and convert all the images in those classes to grayscale, and then stack the same gray image 3 times into the RGB channels. Leave the other five classes the same. (2) Convert all data to grayscale, stack the same gray image 3 times into the RGB channels. Report the accuracy of the previous neural network on Datasets (1) and (2). Which one performs better? Try one of the strategies mentioned in the paper to mitigate bias (doesn't have to be the paper's method, it can be one of the baselines). Discuss what your strategy was and how you implemented it.

3. **PCA Comparison** (10 points)

   Compare your classification network versus a machine learning baseline of **principle component analysis (PCA)**, following approaches in the literature, for example:`https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf`, `https://ieeexplore.ieee.org/document/8038237`, `https://www.mdpi.com/2073-8994/11/1/1`. Describe your PCA method in detail (with equations) and report the accuracy of a PCA approach to your own neural network. It would be helpful to show your principal components for classes, or any other interpretable information that you get from PCA.

Problem 2. *Vision Transformer (Segmentation Task)*(30 points)

In this problem, we will explore the task of image segmentation using a pretrained model and build a small application around it. You will apply segmentation to a selfie image, simulate a blurred background effect, and use monocular depth estimation to create realistic lens blur.

1. Explore the Hugging Face Segmentation Models Library for image segmentation. Choose any model of your choice. Take a picture of a foreground object against a complex background, and demonstrate that the chosen model can successfully segment the foreground from the background. Display the input image and the output mask, where the background should be completely black, and only the foreground object should appear in white. (5 points)

2. Add a Gaussian blur with $\sigma = 15$ to the background of the segmented image, leaving the foreground sharp. This simulates the background blur effect often seen in video conferencing tools like Zoom. Display the input image and the output image with blurred background side by side. (5 points)

3. Explore the Hugging Face Monocular Depth Estimation Models Library and choose any depth estimation model. Run the same input image (512×512) through the model and display the input image alongside the depth map generated by the model. (5 points)

4. Normalize the depth map values to a range to represent the relative distance of objects from the camera. Using this normalized depth map, apply a variable Gaussian blur to the image, where the blur intensity is proportional to the depth of the objects. Specifically, objects that are farther away from the camera (having higher depth values) should appear more blurred, while closer objects (with lower depth values) should remain sharper. This simulates a realistic lens blur effect, similar to what is observed with a camera's depth of field. Display the original input image alongside the output image with the applied depth-based lens blur for comparison. (5 points)

5. Create a Google Colab notebook that contains all the resources necessary to run those models and generate results. Ensure that the images are linked, and all dependencies are included, so the results can be reproduced by simply running the notebook. Share the link. (5 points)

6. Develop a Hugging Face Space app that showcases the Gaussian blur and lens blur effects based on user input. Share the app link and a screenshot of the final output. (5 points)

Problem 3. *(Object Detection at Night)* (40 points)

1. First collect a series of your own videos (at least 3-5 videos of 2-3 seconds in length) of objects moving at night (cars are a good example). Annotate the data with bounding boxes for ground truth (the tool CVAT is helpful here: `https://www.cvat.ai/`). Visualize this data with the ground truth bounding boxes and tell us why you picked the videos you did.(10 points)

2. Take an existing object detection model such as Faster R-CNN, YOLOv8, or SSD and evaluate its performance in terms of IoU and mAP for your test videos. (10 points)

3. Now, try to improve your object detection model to work better at night. You can do any approach you want, as long as you clearly describe your approach to try and improve the metrics for object detection on your collection of test videos. (20 points)

Grading and Report

Grading breakdown is as follows:

- Problem 1: 20 points
- Problem 2: 30 points
- Problem 3: 40 points
- Presentation (are answers clearly defined and easy to find, is code snippets and figures utilized well for the report): 10 points