Note 1: Assuming Windows operating system.

Note 2: Steps 7 through 19 are needed only if steps 5 or 6 fail.

1. Store the provided yml file in a desired directory whose path contains no spaces (to ensure that accessing the directory is not made tricky because of space characters within the directory).
2. Install the latest version of Anaconda for Windows.
3. Open Anaconda prompt.
    a. Note 3: Assuming no prior Anaconda installation. If Anaconda has been already installed, some of these steps may need to be adjusted so that the necessary dependencies and installations can be completed successfully.
4. Change directory using the **cd** command to where the yml file is located.
5. Run the following command to create the custom virtual environment using the provided yml file: **conda env create -f cifar10-training-testing-uncertainty.yml**
    a. If successful, proceed to step 6. Otherwise, proceed to step 7.
6. Run the following command to activate the custom virtual environment: **conda activate cifar10-training-testing-uncertainty**
    a. If successful, proceed to step 20. Otherwise, proceed to step 7.
7. Run the following command to create the Keras GPU environment: **conda create -n keras-gpu keras-gpu**
8. If prompted to approve installation of new packages, type ‘y’.
9. Let the packages be installed. This may take some time.
10. Run the following command to activate the Keras GPU environment: **conda activate keras-gpu**
11. Run the following command to install keras: **pip install keras**
12. If prompted to approve installation of new packages, type ‘y’.
13. Let the packages be installed. This may take some time.
14. Run the following command to install matplotlib: **pip install matplotlib**
15. If prompted to approve installation of new packages, type ‘y’.
16. Let the packages be installed. This may take some time.
17. Run the following command to install matplotlib: **pip install scikit-learn**
18. If prompted to approve installation of new packages, type ‘y’.
19. Let the packages be installed. This may take some time.
20. List the CPUs and GPUs available by running the following command: **python -c "import tensorflow as tf; print(tf.config.list_physical_devices())"**
21. If a GPU is not available, execution should fall back on the CPU. The difference between GPU and CPU is significantly perceived only when training; all other execution should complete in a reasonable amount of time using just the CPU. If training with GPU is not possible for some reason, and training with CPU consumes too much time, the provided pretrained model weights file may be passed to the program as an argument to complete execution of the code in a reasonable amount of time while bypassing model training (more in step 20 on how to pass the weights file as an argument).
    a. Note 4: My PC does not have access to a GPU, so I trained the model done using the free GPU in Google Colab. Otherwise, as long as I passed the weights file as an argument to the program, the code was able to execute successfully in a reasonable amount of time

using just the CPU. I did verify that the program is able to start training the model if no weights file is provided.

22. Run the Python script file as follows, with two optional arguments available and explained below: **python cifar10-training-testing-uncertainty.py**
    a. If given, **-swdir**, or **--save_weights_directory**, is the directory where the weights file will be saved. This is best provided only if training from scratch, although the weights file can still be saved even if the weights are simply being loaded from a separate weights file. To avoid potential issues with argument parsing, ensure that the directory given does not contain any spaces.
    b. If given, **-ldwfl**, or **--load_weights_file**, is the weights file that will be loaded directly onto the model architecture and allow the training to be skipped, thus saving time. To avoid potential issues with argument parsing, ensure that the path to the weights file as well as the name of the weights file itself do not contain any spaces.
    c. An example command with both optional arguments given: **python cifar10-training-testing-uncertainty.py -swdir '/directory/' -ldwfl '/directory/weights_file.h5'**
23. Watch for the print statements on the console that provide information on the current state of the program as well as the classification report from testing.
24. Four plots will eventually be shown, with each successive plot shown on a separate figure 20 seconds after the previous one so that some time is allowed for observation of each plot.
25. THE END!