Read before starting the coding:

You must show the question and your answer clearly, for example, write a question title then code if you use a Notebook type, like Jupyter Notebook; If you use softwares like Pycharm, you must organize your files and comment well, points will be deduced if I can not find your answer.

Write a **README** file which is about the environment(like Python version) and steps to run your codes.

Task Description:

Your task is to build a system that can recognize and classify objects in images. You will use a deep learning framework such as TensorFlow or PyTorch to implement your solution.

The **dataset** you will use for this task is the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class.

The classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

1. Data Preprocessing(20):

- **Load** the CIFAR-10 dataset using libraries like TensorFlow or PyTorch.
- **Normalize** the pixel values of the images to the range [0, 1].
- **Resize** the images to a uniform size if necessary (e.g., 32x32 pixels).

- **Augment** the training dataset with techniques such as random rotations, flips, and shifts to increase the diversity of the training data and improve model generalization.

2. Model Architecture(25):

- **Design** a CNN architecture suitable for image classification tasks.
- **Define**
    1) Define the layers (convolutional layers, pooling layers, fully connected layers) and activation functions for your model.
    2) For this step, use a cell or comment to answer the input and output size of your layers, parameter numbers of your layers.
- **Experiment** with different architectures such as VGG, ResNet, or custom architectures by yourself.
- **Tune** hyperparameters such as learning rate, batch size, and optimizer choice.

3. Training(You could merge some steps implement and utilize)(20):

- **Split** the preprocessed dataset into training, validation, and test sets (e.g., 80% for training, 10% for validation, 10% for testing).
- **Train** your CNN model using the training set. Monitor the training process by tracking metrics like loss and accuracy on the validation set.
- **Implement** techniques like batch normalization to stabilize training and dropout to prevent overfitting.
- **Utilize** early stopping to prevent overfitting and save the best-performing model checkpoint.

4. Evaluation(16):

- **Evaluate** the performance of your trained model using appropriate evaluation metrics with accuracy, precision, recall, and F1-score.
- **Visualize** the training/validation loss and accuracy curves to analyze the model's learning progress.
- **Summary(writing or figures)**error analysis to identify common misclassifications and areas for improvement.

5. Testing(9):

- **Test** your trained model on the separate test set to assess its generalization ability.
- **Calculate and report** the final test accuracy to measure the model's performance on unseen data.

6. Writing(10):

- Summary your model performance and your analysis with words 500-1000.

Extra points(10):

- **Deploy** your trained model as a standalone application or a web service using frameworks like Flask or Django.
- **Create** an API endpoint for inference where users can submit images for classification.
- **Provide** clear instructions on how to use your deployed model for inference, including input format and expected output.