Report and Documentation

Sourav Saini

We are performing image classification on the well known CIFAR-10 dataset. The dataset contains 10 classes namely - ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'].   
The dataset contains 60000 of such (32, 32, 3) RGB images. Out of which we use 40000 (training), 10000 (validation) and 10000 (testing). We train the WideResNet22 model for the task.

Data Preprocessing and Feature Engineering

As mentioned we split the dataset into ( 40K-training, 10K-validation, 10K-test). We also perform Data Augmentation:

1. **RandomCrop()**: Randomly crops the image to 32x32 pixels, with padding of 4 pixels. Padding is done using reflection.
2. **RandomHorizontalFlip()**: Randomly flips the image horizontally.

We employ feature engineering by using normalization to convert the pixel ranges in between [0,1] for stable model training. We use :

1. **ToTensor()**: Converts the image into a tensor (which is needed for PyTorch models).
2. **Normalize(\*stats)**: Normalizes the image using the mean and standard deviation values provided in stats

We use stats = ((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)), these are the mean and standard deviation calculated for all the three channels over the entire dataset. We visualized some images for performing EDA.

Model Selection and Optimization Approach

We use the WideResNet22, some of the advantages it provides are:

1. It has a larger width ( more number of channels ) as compared to the standard ResNet which helps in learning of complex patterns and fine grained details.
2. The fewer layers ( depth ) it has helps in faster training, without compromising on performance
3. As is the advantage of using residual connections, we do not run into the problem of vanishing gradients.

We used Adam for optimization and Cross Entropy loss for our training. A learning rate of 0.01 is kept for balanced training. We use Early stopping to prevent overfitting and also use MultStepLR for controlled decay of learning rate.

Deployment strategy and API usage guide

We build a basic application using FastAPI, and run it using Uvicorn (ASGI server to run the app). We employ lightweight authentication for API access using HTTP Basic Authentication. Then we create an endpoint /predict that takes an image as input and outputs the predicted class. We also deploy a basic frontend using Streamlit.