Introduction:

The problem given to develop an image classifier for CIFAR-10 image dataset using deep learning techniques. The CIFAR-10 dataset consists of 60000 32\*32 resolution colour images and the dataset is balanced with 10 classes, each class consists of 6000 images. The datasets consists of 50000 training images and 10000 test images. The dataset is split as follows first randomly 10000 images are selected by randomly sampling 1000 images from each class , then the training set is split into 5 batches of 10000 each. Each batch can consist of non-uniform number of images from every class , however the total number of images in each class across all the batches is uniform. Please note that in our case the number of batches and batch size considering 50000 training images are customized according to the model. The network architecture used here is the deep learning model made up of Convolutional Neural networks , max pooling layers , activation functions Relu after every convolutional and fully connected layer except the last fully connected layer which is fed into softmax. This architecture here is standard for image classification, which essentially goes in the process of extracting features , downsizing them keeping the most dominant or average features , getting non-linearity ,extracting more features and finally feeding them into dense networks which replicates a complex function to map these features to output classes.

Besides the deep learning network developed by us we have also done experiments with ResNet18 , a pretrained deep learning model which is trained on ImageNet dataset. So by training only last few layers and also by training all layers of this model it can serve the purpose of our task. The results with Resnet are better than those of the traditional CNN model built with training accuracy of 75% and test accuracy 74%.

Methodology:

1. CNN Model – This model consists of using convolutional layers and max pooling layers along with fully connected layers. The exact network architecture is as follows. The first layer is convolutional layer of number of input channel 3 representing RGB color and number of output channels 6 with 5\*5 as kernel size. By default padding is done (zeros) and image size is maintained. The output of Convolutional layer is sent to max-polling layer where the image is downsized using a max -pooling kernel of 2\*2. This layer takes the max of pixels of 2\*2 region, also the stride size is 2 which makes it possible to reduce the image by half across both the dimensions. Post this the output is sent to another convolutional layer taking in 6 input channels and 16 output channels with filter size 5\*5. These convolutional filters extract features from the inputs given, considering all dimensions each feature is extracted from a different kernel, the number of kernels represent the number of output features. The convolutional kernels provide sharing of weights as similar type of features are present across entire image , this is more scalable versus providing all the pixel values to mlp. The examples of features could be gradients , lighting etc. However, the kernels are trained and learn to extract the required features. The activation functions used here is RELU, this provides gradients to the layers when the input values are greater than 0, thus mitigating the vanishing gradient problem provided values inputted are greater than 0. Finally, there are fully connected layers after second convolutional layer these fully connected layers takes in the features whose weights and activation functions gives the output replicating the function whose outputs provides classification values. Each of these dense fully connected layers with activation function replicates a non-linear function whose outputs are fed into next dense layer and so on so all together it replicates a nested function. Finally the output of the last layer is fed into softmax which gives probabilities of different classes. Here we are using CIFAR-10 dataset which is transformed with mean and standard deviation as 0.5 across all 3 channel RGB, this is for scaling the inputs for smooth learning and to keep up the gradients in certain range. Then images for training are given to trainloader and in batches of 4 the network is trained across the entire training set for each epoch. The total number of epochs are 20. The training and validation accuracy and loss is monitored across each epoch.

1. **ResNet18 Model –** ResNet18 has 18 layers , the layer has 7\*7 kernel .It has 17 convolutional layers and 18th one is a fully connected layer. It has 4 layers of ConvNets and each layer consists of 2 residual blocks. Each block consists of 2 weight layers and a skip connection connected to the output of second weight layer. The activation function used here is RELU. The output of the final layer is fed into softmax. The convolutional layer size is 3\*3. The skip connections provide a highway for the gradients to pass and also provide additional features. This avoids the issue of vanishing gradients , hence the earlier layers are updated with sufficient gradients. ResNet18 is trained on ImageNet dataset which has more than a million images across 1000 categories. The images have input size of 224\*224. For training the data transformation is similar to that of previous methodology for traditional CNN. However, the last fully connected layer of our customized ReSNet18 model has 10 outputs instead of 1000. Moreover, the network is huge (consisting of 18 layers) it is not viable to train the network from scratch, instead we use pretrained network and train only the last few layers of the network freezing rest of them. The idea is that the initial layers extract base features which are generic across tasks and the later layers are responsible for extracting task specific features. So, for each training pass we unfreeze certain number of layers from the end and iteratively increase the number of features to unfreeze for every training pass until the model performance maximizes. It turns out eventually that all the layers had to be trained to achieve highest performance. However, the pretraining of the ResNet18 provided higher performance on training and test set as the task od CIFAR10 classification is similar to the objective of Image10 to some extent with the difference that number of classes and class types are different but hypothesizing the process of extracting features is similar with some differences corresponding to differences in tasks, which will allow us to use the pre-trained model with some overriding of weights. The training process is similar with training in batches and monitoring the loss and training , test accuracy.

**Results:**

The results for CNN model are as follows. The loss for each epoch verses the number of mini batches trained.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 2000 | 4000 | 6000 | 8000 | 10000 | 12000 |
| Epoch1 | 2.228 | 1.939 | 1.714 | 1.597 | 1.547 | 1.473 |
| Epoch 2 | 1.407 | 1.369 | 1.341 | 1.312 | 1.319 | 1.280 |
| Epoch 3 | 1.200 | 1.224 | 1.228 | 1.193 | 1.170 | 1.155 |
| Epoch 4 | 1.086 | 1.100 | 1.109 | 1.120 | 1.093 | 1.074 |
| Epoch 5 | 1.005 | 1.029 | 0.999 | 1.041 | 1.035 | 1.019 |
| Epoch 6 | 0.946 | 0.955 | 0.961 | 0.967 | 0.978 | 0.983 |
| Epoch 7 | 0.880 | 0.909 | 0.932 | 0.910 | 0.949 | 0.916 |
| Epoch 9 | 0.824 | 0.860 | 0.870 | 0.882 | 0.897 | 0.909 |
| Epoch 10 | 0.801 | 0.837 | 0.825 | 0.838 | 0.862 | 0.862 |

The results of number of epochs versus train and test accuracy

|  |  |  |
| --- | --- | --- |
|  | Training accuracy | Test accuracy |
| Epoch 1 | 47% | 47% |
| Epoch 2 | 56% | 54% |
| Epoch 3 | 61% | 58% |
| Epoch 4 | 61% | 58% |
| Epoch 5 | 67% | 62% |
| Epoch 6 | 70% | 64% |
| Epoch 7 | 69% | 62% |
| Epoch 8 | 73% | 64% |
| Epoch 9 | 73% | 63% |
| Epoch10 | 73% | 62% |

The final test accuracy is 62% which is less compared to ResNet18model