CIFAR -100 OBJECT RECOGNISTION

NAME – SUJEET KUMAR

ENROLLMENT NO. – 12021002016013

UNDER THE SUPERVISION OF PROF. SWARNENDU GHOSH

Academic Year: 2023-2024

INSTITUTE OF ENGINEERING AND MANAGEMENT

ARCHITECTURE

The classification task required building a convolutional neural network and since there are 100 classes in the dataset the task can be completed with better performance by building a deep neural network to help the model learn better. I started with 5-layer network with 32, 64 and 128 filters in convolutional layers and 256 units in the first fully connected layer. This network had 643,492 trainable parameters and helped me achieve an accuracy of 38.7% on the validation set and 39.5% on the test set. The graphs of loss versus number of epochs and accuracy versus number of epochs for the same has been shown for comparison in the results section. Upon optimization of the model further by increasing the number of layers and filters in convolutional layers and number of units in fully connected layer, I later on finalized a model with a 9-layers having 13,870,484 train- able parameters. The ConvNet architecture of this 9-layer deep neural network has 3 stacks of CONV-RELU layers followed by a POOL layer and then 2 fully connected (FC) RELU layers and a final fully connected output layer.

INPUT → [[CONV → RELU] × 2 → MAX-POOL] × 3 → [FC → RELU] × 2 → FC

The specifics of each layer are as follows:

1. Each CONV layer uses zero-padding to ensure than the dimension of output is same as the dimension of input.

2. ReLU has been used as the activation function in all the hidden layers because of its sparsity and advan- tage to avoid vanishing gradient problem. While training my neural network ReLU showed good convergence performance and was also computationally efficient.

$$f(x) = \max(0, x)$$

The reason for this is that ReLU does not activate all the neurons at the same time due to its nature to choose the maximum value between 0 and x. As a few inputs can have negative value too, so ReLU takes the maximum value as 0 and do not activate them. As my model had millions of parameters, this helped to converge faster.

3. A small kernel size of 3 has been used for all the CONV layers as the dataset has 32 × 32 pixels images and I wanted to extract most of the details form these images. It also lowered down the number of weights in my network and helped in faster training.

4. Max pool of size 2 has been used in the model as max pooling calculates the maximum value in each patch of the feature map. The stride of 2 has been used in the model to move filters 2 pixels at a time and to downsample the image by 2 in both width and height.

5. The number of filters used in the first stack of convolutional layers is 128, the number of filters used in the second stack of convolutional layers is 256 and the number of filters used in the third stack of convolutional layers is 512. Since, CIFAR-100 has 100 classes to

classify each image, thus large number of filters helped in achieving better performance. The process of tuning these filters started from 32 and was gradually increased while monitoring the validation loss. The graphs for the same are shown in the results section to show performance difference in models.

6. The output of the convolution and pooling layer is flattened into a single vector of values and fed to the fully connected layers.

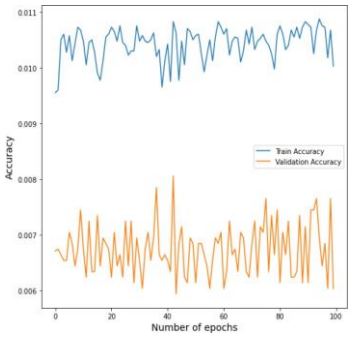
7. Three fully connected layers with 1000 units in the first two layers and 100 units (units equal to the number of classes) in the last (output) layer have been used to make the network dense and learn the non-linear combinations of features obtained from the convolutional layers. This combination of units and layers helped the network classify images with more accuracy (5% more than the network with only two fully connected layers).

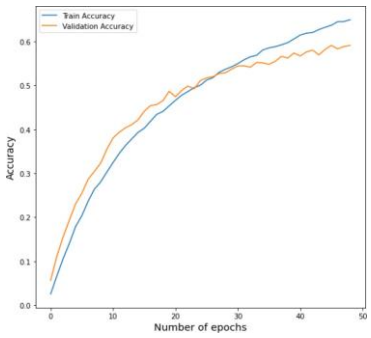
8. The output layer uses Softmax as the activation function to give probabilities as the output by squashing the values in the range [0,1]. This has been done to ensure that the final activations all sum up to 1 and fulfill the constraints of probability distribution.

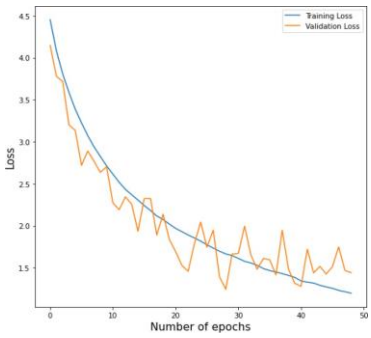
9. Dropout has been used in all the hidden layers with values 0.2 in the input layer and 0.5 in the hidden layers. Upon using the dropout as 0.5 in the input layer as well, the model got low accuracy due to lesser number of parameters and the use of 0.2 in all the layers caused memory issues due to a greater number of parameters to train. This setting of 0.2 and 0.5 worked well and was computationally efficient as well.

Result

The project focuses on building a convolutional neural network to recognize and classify images from the CIFAR-100 dataset, which contains 60,000 colored images categorized into 100 classes and 20 superclasses. It is clear from the graphs that the model is not learning at learning rates 0.01 and 0.001 whereas a lot of learning has been happening at the learning rate of 0.00001. The final optimized model uses the Adam optimizer and learning rate 0.0001, providing better accuracy than the original model with a learning rate of 0.0001.







The model uses multiple layers and filters to improve accuracy. It has a 5-layer neural network with 643,492 parameters and a loss versus number of epochs graph. Early stopping prevents overfitting and the model stops training at the 49th epoch after 20 epochs. The model has an accuracy of 59% on the testing dataset and a loss of 1.47. However, the model's precision and recall are low in most categories, indicating need for further optimization. The model misclassifies images with multiple objects.