CIFAR-10 Image Classification in Pytorch using LeNet and ResNet Architectures

Rushiil Bhatnagar, *CSE- Machine Learning and Artificial Intelligence*

Abstract

This is a project report for the assignment provided by IDfy, Mumbai. It was required that there be CNN model(s) built using Pytorch framework for classifying the CIFAR-10 Images with the help of architectures like LeNet, ResNet or VGG.
In this study we have used the LeNet architecture (designed from scratch, and not called from pytorch) and the ResNet architecture, for which we have used pytorch functionalities and trained our model on it.

In this study I have experimented with concepts such as Pruning (as required by the assignment) as well as hyperparameter tuning to get better results.

Finally we have achieved satisfactory accuracy with the used architectures even though there is a scope of improvement in results if given a better choice of models in the industry.

1. INTRODUCTION

CIFAR10 Dataset is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research. It consists of 60,000 32x32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. The dataset is divided into 50,000 training set data points and 10,000 test set data points.
Focus and Purpose: The aim of this study is to successfully classify the images into their respective classes with the help of pytorch framework using various CNN Architectures (ResNet and LeNet for the purpose of this report).
A foreword: This aims to exhibit my skills and utmost interest in the field of Machine Learning, AI and Deep Learning, This also aims to deepen my understanding of the concepts of ANNs and CNNs along with Computer Vision fundamentals.
Literature Review and Previous Works: In literature, there are numerous journals, research papers and code resources present on various platforms which aim to implement the CNN Architectures that we have used in our study– LeNet and ResNet (specifically, ResNet18 variant from pytorch).
Moreover, there are numerous illustrative and literature implementations on a larger dataset variant of CIFAR 10, i.e. CIFAR–100 present in the academic world.
These architectures and the dataset have comprehensively been studied and vigorously used to train numerous models.

2. PROBLEM FORMULATION

I. We aim to define and train the models based on– i.) LeNet and ii.) ResNet architectures that are really popular in implementation besides also being simple.

II. After the model training it's required to be pruned in order to make it light weight without affecting the accuracy. For this an L1-unstructured pruning is proposed and implemented.

3. METHODOLOGY

The data set was pulled from an online resource (kaggle) and can also be available through pytorch datasets utility. It's already divided into training and testing sets (50,000 and 10,000 images respectively) so we just have to split the training set into Train and Validation sets for the purpose of our model training. According to the guidelines the dataset was split into training and validation sets as 40,000 and 10,000 images respectively.

Design and Analysis:

i. **LeNet architecture:** We have used the basic LeNet-5 as shown in the figure(1.) which is input with 3 channel (coloured images) of the dataset.

ii. **ResNet architecture:** We have used the pre-defined (and custom trained, i.e. *pretrained=False*) ResNet-18 architecture with the pytorch framework utilities.

iii. **L1-regularization Pruning technique:** Pruning is a data compression technique in machine learning and search algorithms that reduces the size of our dense network. We have implemented a specific type of pruning method– *l1 unstructured* pruning, using pytorch for this study.

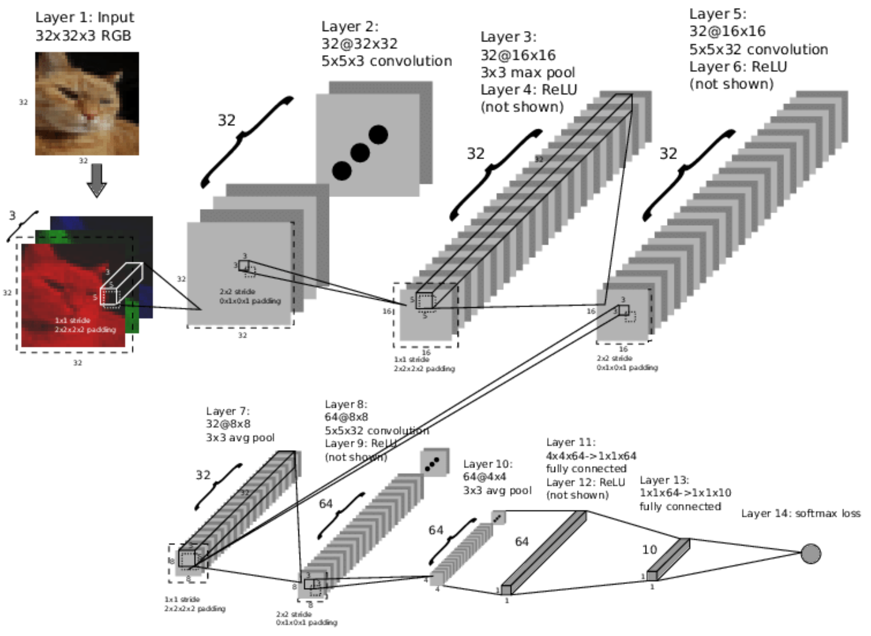

Figure (1). **LeNet** Architecture for Image classification that is used.

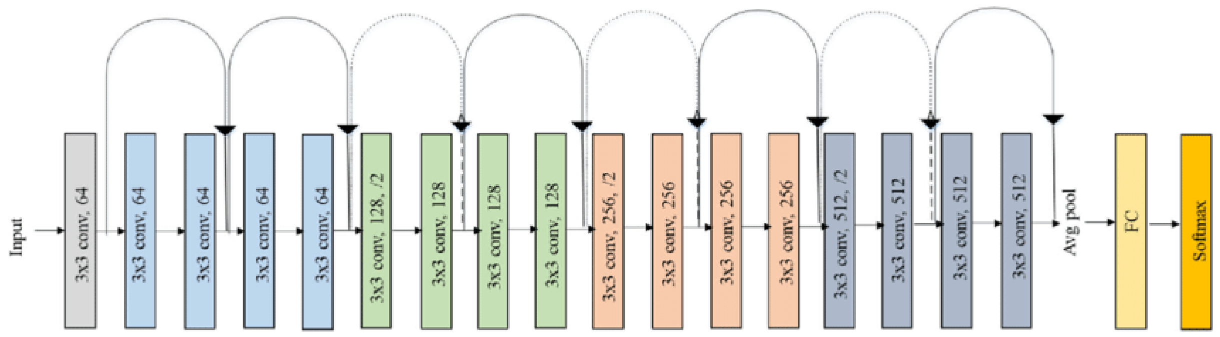

Figure (2). **ResNet** Architecture for Image classification that is used.

Implementation: A Convolutional Neural Network works on ANN principles while using the convolution operation techniques on images for better processing and results. For this study we have trained our model(s) using both techniques at a time and then serialized them for the purpose of future use (*refer to further sections for more on this.*).

The code can be found at this link for reference of implementation (this is a GitHub repository).

Refer to *table(1.)* for the results that were recorded during the training process. Following are some of the important constants to keep in mind:–

epochs= 10
pruning_ratio= 0.5 (50%)
train_size= 40,000 val_size= 10,000
learning_rate= 0.0153 (randomly chosen), momentum= 0.6 [for LeNet]. Pruning method= *L1_unstructured global prune*.

EXPERIMENTAL RESULTS

Batch_Size	**Model Architecture**	**Validation Accuracy** (%)	**Test Accuracy** (%)	**Pruned?** False/True 🔴/🟢
64	ResNet	73.34	72.67	🔴
64	ResNet	84.19	83.93	🟢(50%)
64	LeNet	60.83	61.21	🔴
64	LeNet	61.64	60.88	🟢(50%)
32	ResNet	76.75	76.72	🔴
32	ResNet	86.50	86.53	🟢(50%)
32	LeNet	62.44	62.76	🔴

32	LeNet	61.15	60.77	🟢(50%)
16	ResNet	79.95	80.38	🔴
16	ResNet	84.88	85.04	🟢(50%)
16	LeNet	59.61	60.51	🔴
16	LeNet	52.47	54.09	🟢(50%)
8	ResNet	84.98	84.83	🔴
8	ResNet	84.64	84.37	🟢(50%)
8	LeNet	59.33	59.18	🔴
8	LeNet	57.62	57.09	🟢(50%)

Table(1.) *This table captures all the observations from this machine learning training experiment with pruned and not-pruned ResNet and LeNet Models.*

CONCLUSION/LEARNINGS

This was an intellectually challenging opportunity where I refreshed the topics on Machine Learning, Deep Learning and Computer Vision. More than that, through this I focused on good programming practices such as neat and clean coding as well as comment descriptions for better aiding the understanding of functionalities and efficiency of human cognition.

This also presented an opportunity to implement a technique such as pruning which I was, until now, only theoretically familiar with. This also made me familiar with the Pytorch framework, with which I have worked for the first time.

I observed a vague trend where l1-unstructured pruning led to an increase in the actual accuracy of the model independent of the batch size. (see *table[1.])* At the risk of doing something wrong in the code, I checked and rechecked the logic/code. Upon finding nothing wrong, I started reading more about it before I found it's possible.

Link to GitHub Repo:--

IDfy_DeepLearning