**Project Report on ResNet implementation using CIFAR10 dataset**

Contents

1. Objective
2. Neural Networks
3. Introduction to ResNet
4. ResNet9 Architecture
5. Dataset – CIFAR10
6. Hyperparameters
7. Optimizers
8. Epoch
9. Dropout
10. Learning rate
11. Early stopping
12. Implementation of ResNet9 using Pytorch
13. Experiments around hyperparameters
14. Analysis
15. Future work
16. References

**Section 1: Objective**

The main objective of this report is to study ResNet9 architecture and to compare the impact of the different hyperparameters such as epoch, optimizers, learning rate, dropout etc on model’s performance.

**Section 2: Neural Networks**

A neural network is a combination of multiple layers where each layer consists of multiple units, and these layers are mainly categorized into three sections 1. An input layer, 2. Hidden layer(s), and 3. Output layer. A neural network is said to be dense layered NN. When each unit from one layer is connected to every other unit in the next layer then, it is said to be a dense, layered neural network. Basic NN structure:

A diagram of a network

Description automatically generated

Figure 1: Basic structure of Neural network

Another variant of neural network is convolutional Neural network. Convolutional Neural Networks (CNNs) have an input layer, an output layer, numerous hidden layers, and millions of parameters, allowing them to learn complicated objects and patterns. It uses convolution and pooling processes to sub-sample the given input before applying an activation function, where all of them are hidden layers that are partially connected, with the completely connected layer at the end resulting in the output layer. The output shape is like the size of the input image.

Convolutional neural networks are hard to train. As deeper networks begin to converge, a degradation issue emerges and as the depth of the network increases, the accuracy becomes stagnant. At this point, if we further add more layers to the network, accuracy falls steadily. To overcome the issue, ResNets were introduced, which implements the shortcut connections that neither add extra parameter nor computation complexity.

**Section 3: Introduction to Resnet**

Image recognition pertains to the field of computer vision and involves the automatic identification and interpretation of various objects within pictures. Image recognition involves use of Convolutional network, which are hard to train. As deeper networks begin to converge, a degradation issue emerges and as the depth of the network increases, the accuracy becomes stagnant. At this point, if we further add more layers to the network, accuracy falls steadily. To overcome the issue, the concept of skip connections in ResNet was introduced by Kaiming He et al in 2015. These shortcut connections neither add extra parameters nor computation complexity. The implementation of residual network is tested on various datasets such as ImageNet, CIFAR-10 etc and the analysis showed promising results, with the error percentage of 3.75% on ImageNet test dataset, as shown in the paper - <https://arxiv.org/abs/1512.03385>

So, a residual network, or ResNet for short, is an artificial neural network that helps to build deeper neural networks by utilizing skip connections or shortcuts to jump over some layers. Skipping connections helps build deeper network layers without falling into the problem of vanishing gradients.

There are different versions of ResNet, including ResNet-9, ResNet-18, ResNet-34, ResNet-50, and so on. The numbers denote layers, although the architecture is the same.

A diagram of a traffic light

Description automatically generated

Figure 2: Skip connections in the layers

**Section 4: RestNet9 Architecture**

ResNet9 consists of 8 convolutional layers and 1 linear layer. The primary function of convolutional layer is to do feature engineering, extraction and transformation and linear layer is used to predict the final class. In between the convolutional layers, ReLu activation function is used, while for the final linear layer either sigmoid or softmax activation function is used, depending on the final output which could be binary or multi class. The sigmoid function compresses the output values to a range between 0 and 1, serving as a representation of the probability of belonging to one of the two classes. If you have more than two classes, softmax is typically more suitable.

A diagram of a graph

Description automatically generated

Figure 3: ResNet9 architecture

**Section 5: Dataset**

CIFAR stands for Canadian Institute for Advanced Research. The CIFAR10 dataset consists of 60,000 32x32 color images, belonging to 10 classes with 6,000 images per class. The ten classes of image in it are frog, bird, automobile, airplane, cat, dog, truck, horse, ship, deer. There are 50,000 images for training data with 5,000 images per class and 10,000 for testing the model with 1000 images per class.

**Section 6: Hyperparameters**

Hyperparameters are configuration settings with predefined values that are established prior to the initiation of the model training process. Selecting suitable values for these hyperparameters is a crucial aspect of constructing impactful machine learning models The process of finding the best combination of hyperparameters for a given machine learning model is known as hyperparameter tuning, which is typically assessed on a validation set. They assume a pivotal role in both the training and performance of machine learning models.

Below are some examples of hyperparameters in machine learning:

1. **Optimizers**

In the context of machine learning and neural network training, an optimizer is an algorithm or method that is used to adjust the parameters of a model to minimize the error or loss function. Below are the commonly used optimizers:

**Stochastic Gradient Decent (SGD):** This is the most popular and widely used optimizer. It can have variations such as Mini-batch SGD, Batch SGD, and Stochastic SGD. Stochastic Gradient Descent (SGD) is an optimization algorithm that approximates the gradient of the error for the current model state by utilizing examples from the training dataset. Subsequently, it updates the model weights through the application of the backpropagation algorithm, commonly known as backpropagation.

**Adaptive Moment Estimation (Adam):** The Adam is considered more efficient and requires less memory when working with a large amount of data and parameters. It requires less memory and is efficient.

**Root Mean Square Propagation (RProp):** The RProp is widely used in multi-layered feed-forward networks.

1. **Epoch**

In an epoch, the model traverses the entire dataset once. The model parameters are updated in each epoch until they reach optimal values. Specifying too many epochs, the model will commit overfitting. It will learn the training data too well but predict poorly for a new dataset. To determine the optimal number of epochs, we use the loss and accuracy metrics. When a model is trained with more epochs, the loss will decrease, and the accuracy will increase. After a certain number of epochs, the loss will stop decreasing but increase, and the accuracy decreases. It indicates the model training should end with that epoch. The function EarlyStopping() can be used to monitor and find the optimal number of epochs, thereby reducing processing time.

1. **Dropout**

In machine learning, “dropout” refers to dropping out neuron units in a neural network layer at random during training. Dropping units means these units are not considered during a particular forward or backward pass. A dropout is a regularization technique that **prevents overfitting.**

A diagram of a network

Description automatically generated

Figure 4: Dropout depiction with respect to figure 1

1. **Learning rate**

The learning rate, as a hyperparameter, regulates the extent to which the model should be adjusted based on the calculated error during each update of the model weights in optimization. Selecting an appropriate learning rate poses a challenge, as opting for a value too small may lead to an extended training duration with a risk of stagnation. Conversely, a value too large may induce rapid learning of suboptimal weights or an unstable training process.

A graph of a function

Description automatically generated

Figure 5: Learning rate optimization

1. **Early stopping**

Early stopping is a regularization technique used in the training of machine learning models, particularly in the context of iterative optimization algorithms such as gradient descent. The idea behind early stopping is to monitor the performance of a model on a validation set during training and stop the training process once the performance stops improving or starts to degrade. This is done to prevent overfitting and to obtain a model that generalizes well to unseen data.

**Section 7: Implementation of ResNet9 using Pytorch**

In this section, implementation of ResNet9 architecture using Pytorch is explained. It will cover the basic structure of Pytorch, followed by steps involved in ResNet9 implementation.

The basic structure of any algorithm using Pytorch is:

# importing the necessary libraries

import torch

import torch.nn as nn

# define your model using nn.module

class YourModel(nn.Module):

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

super(YourModel, self).\_\_init\_\_()

self.layer1 = nn.Linear(input\_size, hidden\_size)

self.relu = nn.ReLU()

self.layer2 = nn.Linear(hidden\_size, output\_size)

def forward(self, x):

x = self.layer1(x)

x = self.relu(x)

x = self.layer2(x)

return x

# Instantiate the model

input\_size = integer1

hidden\_size = integer2

output\_size = integer3

model = YourModel(input\_size, hidden\_size, output\_size)

# define loss function and optimizer

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

# define dataloader and training part

num\_epochs = integer4

for epoch in range(num\_epochs):

for inputs, labels in dataloader:

optimizer.zero\_grad()

outputs = model(inputs)

loss = criterion(outputs, labels)

loss.backward()

optimizer.step()

#finally save the model

torch.save(model.state\_dict(), 'your\_model.pth')

Next part will cover ResNet9 implementation on CIFAR10 dataset.

1. Load the dataset and split it to train and test dataset
2. Data Normalization:

A computer screen shot of a code

Description automatically generated

In the code above, the data is being normalized to bring the input data to a standard scale, which helps improve the training stability and convergence of neural network. The code iterates through batches of data from a Data Loader. For each batch, it calculates the mean of the batch along dimensions 0 (channels), 2 (height), and 3 (width), separately for each channel. These means are accumulated in a variable channel\_sum. It also calculates the squared mean of the batch along the same dimensions and accumulates them. Once it iterates through all batches, it computes the overall mean and standard deviation across them.

1. Transform the dataset and convert it to tensor

The next step in to transform the images and convert our image into a tensor. This is the basic requirement of pytorch that our inputs must be tensors only. During the loading of images from the training dataset, we will implement randomly selected transformations. Specifically, we will apply a 4-pixel padding to each image, followed by a random crop of dimensions 32 x 32 pixels. Additionally, we will horizontally flip the image with a probability of 50%.

1. Create dataloaders

Data loaders returns us the image in batches with the given transformation and directory. The batch size is set to 400. Shuffling is set true so that images are reshuffled at every epoch, batch size specifies the number of samples that are loaded per batch, num\_workers specify the number of sub processors used while loading the data, pin\_memory is set to true when you want your dataset to be pushed on GPU while training. So, we don’t need shuffling in validation dataloader as we are just validating the model. Also, the batch size is doubled for validation as no training is involved here.

1. Defining accuracy functions for training and validation step

A screenshot of a computer code

Description automatically generated

1. Defining the ResNet function

A screenshot of a computer program

Description automatically generated

1. **Setting up hyperparameters**

Learning Rate: For learning rate, **“One Cycle Learning Rate Policy”** method is used, which involves starting with a low learning rate, gradually increasing it batch-by-batch to a high learning rate for about 30% of epochs, then gradually decreasing it to a very low value for the remaining epochs.



The One Cycle Learning Rate Policy is a methodology employed in the training of neural networks, involving the specific adjustment of the learning rate throughout the training process. This strategy was introduced by Leslie N. Smith in a paper titled "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates."

The fundamental concept of the One Cycle Policy involves initiating the training with a relatively modest learning rate, elevating it abruptly to a higher value, and subsequently tapering it gradually. This cycle generally encompasses two phases: the increasing phase, where the learning rate is heightened to a maximum value, and the decreasing phase, during which it is progressively diminished. The primary advantages of adopting the One Cycle Policy comprise accelerated convergence, enhanced generalization, and the capacity to train models using higher learning rates while mitigating the risk of divergence.

A computer code with black text

Description automatically generated

**Section 8: Experiments around hyperparameters**

Below is the summary of different experiments carried out on ResNet9 -

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Hyperparameters** | **Model 1** | **Model 2** | **Model 3** | **Model 4** | **Model 5** | **Model 6** | **Model 7** |
| **Epoch** | 10 | 10 | 10 | 30 | 25 | 25 | 25 |
| **Optimizer** | SGD | Adam | RMSprop | Adam | Adam | Adam | Adam |
| **Learning Rate** | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.001 |
| **Dropout** | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.4 | 0.2 |
| **Observation** | Model accuracy is 84.8% | Model accuracy increased to 92% | Model accuracy decreased to 78% | Model accuracy is 86% | Model accuracy increased to 92.6% | Model accuracy decreased to 89% | Model accuracy remains comparable to 92.4% |

**Section 9: Analysis**

With respect to the change in optimizer from SGD or RMSprop to Adam, we observe that Adam optimizer gives better accuracy than SGD and RMSprop. Accuracy levels were observed at various epoch levels also, starting from 10 epochs. It was observed that increasing the epoch value leads to better accuracy. For example, epoch value was changed from 10 to 15 to 20 to 25, but further increasing the epoch value to 30 leads to decrease in accuracy. Another hyperparameter that was observed was dropout. Increasing dropout from 20% to 40%, decreases the accuracy. For learning rate, increasing the learning rate from 0.01 to 0.001, gives comparable accuracy value.

**Section 10: Future work**

Future work could involve working on further fine tuning the hyper parameters, usage of early stopping, batch size and reducing the training time. Furthermore, it could also be exploring other ResNets models with a greater number of layers.

**Section 11: References**

ResNet: <https://tensorspace.org/html/playground/resnet50.html>

<https://medium.com/@ketanmk26/classifying-cifar10-images-using-resnets-regularization-and-data-augmentation-in-pytorch-explained-34ce828502c7>

[Transfer Learning for Image Classification — (6) Build and Fine-tune the Transfer Learning Model | by Chris Kuo/Dr. Dataman | Medium](https://dataman-ai.medium.com/transfer-learning-for-image-classification-6-build-the-transfer-learning-model-67d87999af4a)

<https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/#:~:text=Deep%20learning%20neural%20networks%20are,the%20model%20weights%20are%20updated>.

<https://www.jeremyjordan.me/nn-learning-rate/>

<https://paperswithcode.com/method/early-stopping#:~:text=Early%20Stopping%20is%20a%20regularization,improves%20on%20a%20validation%20set>.

[ResNet. Residual Neural network on CIFAR10 | by Arun Purakkatt | Analytics Vidhya | Medium](https://medium.com/analytics-vidhya/resnet-10f4ef1b9d4c)

<https://arxiv.org/abs/1512.03385>

<https://www.analyticsvidhya.com/blog/2022/08/introduction-to-neural-network-in-deep-learning/>

<https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/>