Modified ResNet for CIFAR Mini-project 7123

Ruining Li

[1]New York University
rl4895@nyu.edu
the link to the code base is https://github.com/leeeexi/7123-mini-project-resnet-sp24

Abstract

The mini-project requires to design, and train modified residual network architecture with CIFAR-10 image and do the image classification on a custom test dataset like CIFAR-10 image. The architecture should have less then 5 million parameters. To design an efficient and accurate model, some experiments have been conducted. In this paper, there is a brief introduction to the architecture of the model, the data preprocessing techniques and the ex-periments on activation and optimizers to finally train an accurate model.

Introduction

A residual neural network is a type of deep learning model where the weight layers learn residual functions based the layer inputs. It operates similarly to a highway network with gates that are opened through strongly positive bias weights. This characteristic allows deep learning models with numerous layers to train more effectively and achieve higher accuracy as they become deeper. In this project, the goal is to modify the classic residual neural network to design a lighter model with lower than 5 million parameters and then do image classification on a custom test. The basic architecture is ResNet 18 Figure 1 which consists 18 layers, including convolutional layers, pooling layers, fully connected layers and identity shortcuts. Some data augmentation techniques and optimizers are also listed in the project

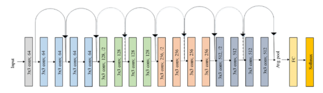


Figure 1: Original model of ResNet-18

Methodology

The dataset consists 60000 32x32 image in which 50000 are for training and 10000 are for testing. Data augmentation is the process of applying a range of transformations to the current dataset to generate additional images. This technique aims to improve the model's resilience to noise and enhance its ability to generalize by introducing variations in the input data. Some data augmentation techniques are defined in this project including: including resized crop, vertical flip, horizontal flip, color jitter, rotation, and Gaussianblur.

Random resized crop

The input image is first resized to a random size within a certain range,which helps the model learn to recognize objects at different scales. After resizing, a random crop of the specified size is taken from the resized image. This random cropping helps the model learn to focus on different parts of the image, enhancing its ability to generalize.By applying Random Resize Crop, the model is exposed to variations in the input data, making it more robust to different image sizes and compositions.

Random vertical flip

For each image during training, a random decision is made whether to perform a vertical flip or not. If the random decision is to flip the image vertically, the image is flipped along the vertical axis, effectively reversing the top-to-bottom orientation of the image.By applying Random Vertical Flip, the model is exposed to variations in the orientation of the input images, which can help improve its robustness and generalization performance.

Color jitter

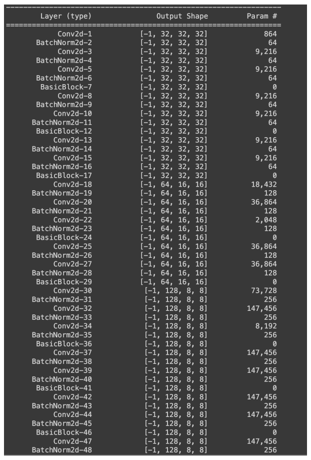
For each image during training, random adjustments are made to its color channels. The magnitude of the color adjustments is controlled by predefined parameters such as the range of possible changes for each color channel.

Cutout

Since the images in the dataset are smaller in size, another technique named cutout is also applied. Cut out is a simple regularization method which involves randomly masking out sections of input images during the training process. In this way, Cutout mimics occluded examples and encourages the model to focus on smaller details when making predictions, rather than solely relying on a few prominent features. In this way, more detailed feature in smaller images can be used

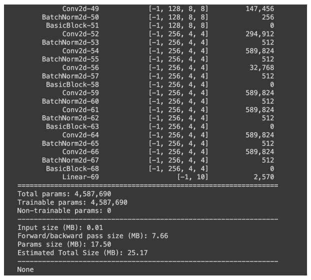
Architecture

The model in Figure 2 and Figure 3 shows the summary of the architecture of the model. In its core part, the be-ginning of the model is an initial convolutional layer, followed by batch normalization. The kernel size is set to 3x3. In order to make the model fix complex pattern of train and test dataset, GELU activation is chosen as the activation layer. With GELU, the model becomes smoother and better at handling complex pattern. With the number of residual blocks as [3,2,4,3], the model performs well and the number of parameters is near 5 mil-lion. As shown in figure 2, the total number of layers in this model is 69. This model is a variant of ResNet 18 architecture with shortcut connections with shortcuts and a global average pooling layer. With this model, the total number of parameters is about 4.5 million and the best test accuracy is 92%.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             864
       BatchNorm2d-2           [-1, 32, 32, 32]              64
            Conv2d-3           [-1, 32, 32, 32]           9,216
       BatchNorm2d-4           [-1, 32, 32, 32]              64
            Conv2d-5           [-1, 32, 32, 32]           9,216
       BatchNorm2d-6           [-1, 32, 32, 32]              64
        BasicBlock-7           [-1, 32, 32, 32]               0
            Conv2d-8           [-1, 32, 32, 32]           9,216
       BatchNorm2d-9           [-1, 32, 32, 32]              64
           Conv2d-10           [-1, 32, 32, 32]           9,216
      BatchNorm2d-11           [-1, 32, 32, 32]              64
       BasicBlock-12           [-1, 32, 32, 32]               0
           Conv2d-13           [-1, 32, 32, 32]           9,216
      BatchNorm2d-14           [-1, 32, 32, 32]              64
           Conv2d-15           [-1, 32, 32, 32]           9,216
      BatchNorm2d-16           [-1, 32, 32, 32]              64
       BasicBlock-17           [-1, 32, 32, 32]               0
           Conv2d-18           [-1, 64, 16, 16]          18,432
      BatchNorm2d-19           [-1, 64, 16, 16]             128
           Conv2d-20           [-1, 64, 16, 16]          36,864
      BatchNorm2d-21           [-1, 64, 16, 16]             128
           Conv2d-22           [-1, 64, 16, 16]           2,048
      BatchNorm2d-23           [-1, 64, 16, 16]             128
       BasicBlock-24           [-1, 64, 16, 16]               0
           Conv2d-25           [-1, 64, 16, 16]          36,864
      BatchNorm2d-26           [-1, 64, 16, 16]             128
           Conv2d-27           [-1, 64, 16, 16]          36,864
      BatchNorm2d-28           [-1, 64, 16, 16]             128
       BasicBlock-29           [-1, 64, 16, 16]               0
           Conv2d-30            [-1, 128, 8, 8]          73,728
      BatchNorm2d-31            [-1, 128, 8, 8]             256
           Conv2d-32            [-1, 128, 8, 8]         147,456
      BatchNorm2d-33            [-1, 128, 8, 8]             256
           Conv2d-34            [-1, 128, 8, 8]           8,192
      BatchNorm2d-35            [-1, 128, 8, 8]             256
       BasicBlock-36            [-1, 128, 8, 8]               0
           Conv2d-37            [-1, 128, 8, 8]         147,456
      BatchNorm2d-38            [-1, 128, 8, 8]             256
           Conv2d-39            [-1, 128, 8, 8]         147,456
      BatchNorm2d-40            [-1, 128, 8, 8]             256
       BasicBlock-41            [-1, 128, 8, 8]               0
           Conv2d-42            [-1, 128, 8, 8]         147,456
      BatchNorm2d-43            [-1, 128, 8, 8]             256
           Conv2d-44            [-1, 128, 8, 8]         147,456
      BatchNorm2d-45            [-1, 128, 8, 8]             256
       BasicBlock-46            [-1, 128, 8, 8]               0
           Conv2d-47            [-1, 128, 8, 8]         147,456
      BatchNorm2d-48            [-1, 128, 8, 8]             256
```

Figure 2: model summary-1



```
           Conv2d-49            [-1, 128, 8, 8]         147,456
      BatchNorm2d-50            [-1, 128, 8, 8]             256
       BasicBlock-51            [-1, 128, 8, 8]               0
           Conv2d-52            [-1, 256, 4, 4]         294,912
      BatchNorm2d-53            [-1, 256, 4, 4]             512
           Conv2d-54            [-1, 256, 4, 4]         589,824
      BatchNorm2d-55            [-1, 256, 4, 4]             512
           Conv2d-56            [-1, 256, 4, 4]          32,768
      BatchNorm2d-57            [-1, 256, 4, 4]             512
       BasicBlock-58            [-1, 256, 4, 4]               0
           Conv2d-59            [-1, 256, 4, 4]         589,824
      BatchNorm2d-60            [-1, 256, 4, 4]             512
           Conv2d-61            [-1, 256, 4, 4]         589,824
      BatchNorm2d-62            [-1, 256, 4, 4]             512
       BasicBlock-63            [-1, 256, 4, 4]               0
           Conv2d-64            [-1, 256, 4, 4]         589,824
      BatchNorm2d-65            [-1, 256, 4, 4]             512
           Conv2d-66            [-1, 256, 4, 4]         589,824
      BatchNorm2d-67            [-1, 256, 4, 4]             512
       BasicBlock-68            [-1, 256, 4, 4]               0
           Linear-69                   [-1, 10]           2,570
================================================================
Total params: 4,587,690
Trainable params: 4,587,690
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 7.66
Params size (MB): 17.50
Estimated Total Size (MB): 25.17
----------------------------------------------------------------
None
```

Figure 3: model summary-2

Result

The model is trained with 50000 CIFAR images. The combinations of different activation and opti-mizer are also examined. The activation method includes Relu and GELU, and the optimizer in-cludes Adam and SGD. The combination that produces best result is GELU and Adam with learning rate of 0.01. The highest test accuracy is 92.36%. Figure 4,5,6,7 are the result from the different experiments.

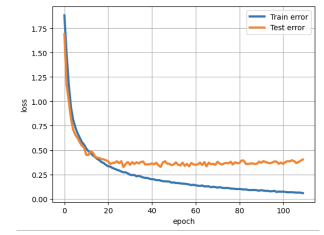


Figure 4: Result of SGD and Relu

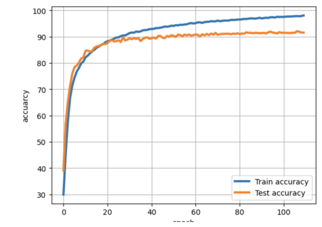


Figure 5: Result of SGD and Relu

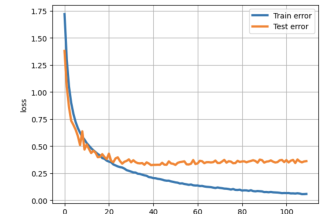


Figure 6: Result of Adam and GELU

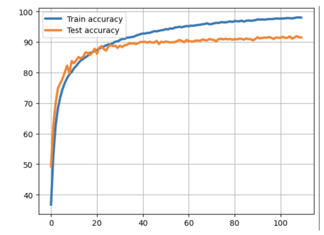


Figure 7: Result of Adam and GELU

References

Hassan, E.; Shams, M.; Hikal, N.; and Elmougy, S. 2022. The effect of choosing optimizer algorithms to improve computer vision tasks: a

Takahashi, R.; Matsubara, T.; and Uehara, K. 2020. Data Augmentation Using Random Image Cropping and Patching for Deep CNNs. IEEE Transactions on Circuits and Systems for Video Technology, 30(9): 2917–2931.

DeVries,T.;Taylor,G.2017.Improved Regularization of Convolutional Neural Networks with Cutout

Ramen,F.;Khan,M.;Iqbal,S.2019. A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks