Computer Vision Assignment6 Report

Siyuan Yin 2100017768

1 Introduction

In this assignment, I implemented VGG16, ResNet34, ResNet50, ResNeXt50_34x4d and conducted experiments on CIFAR10 with results in Table 1. All results have a test accuracy above 90% . I did a lot of experiments to augment dataset, avoid overfitting, select optimizers and shedulers, choose better hyperparameters. Implementation details are in Section 2 and Section 3. Finally I found some interesting conclusions which had been verified by some existing papers, See Section 5.

Model	Model Size	Train Loss	Train Acc (Max)	Test Acc (Max)
ResNext	23,0M	**0.12**	**95.838%**	**93.98%**
ResNet50	**23.5M**	0.15	94.938%	93.44%
ResNet34	21.2M	0.15	94.762%	93.01%
VGG	21.0M	0.17	94.554%	92.92%

Table 1: Model Performance Comparison

2 Data Augumentation

My code in file `AugmentCifar.py` is based on Augmentation. The CIFAR10Policy class contains 15 SubPolicy instances, each with predefined probabilities, operations, and magnitude indices. This policy is intended to apply a randomly selected sub-policy from this list to an image. When the CIFAR10Policy is called on an image, it randomly selects one of these sub-policies and applies it to the image. The sub-policy contains:

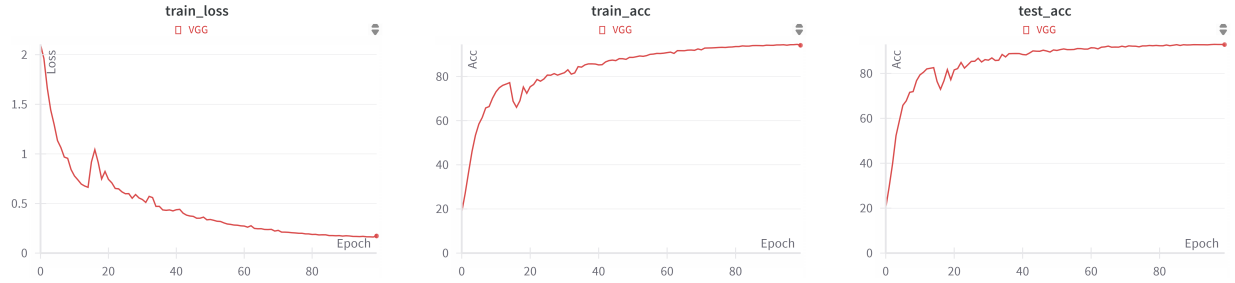
- Rotate: Rotates the image by a certain number of degrees, determined by the magnitude.
- TranslateX: Shifts the image horizontally by a certain number of pixels.
- ShearX: Shears the image along the horizontal axis.
- Color: Alters the balance of colors in the image.
- Contrast: Adjusts the contrast level of the image.
- Brightness,Sharpness, Posterize, Solarize, AutoContrast, Equalize etc.

After data augmentation and 100 epochs, from an expected standpoint, we can obtain a dataset that is 15 times larger.

3 Models Implementation

3.1 VGG16

In VGG, all convolutional layers use 3x3 filters with stride and pad of 1, along with max pooling layers that follow some of the convolutional layers. The small filter size (3x3) with deeper architecture allows the network to have a larger effective receptive field. **Batch normalization** layer is located after convolution and before ReLu activation function. And **Dropout** is located in fully connected layer. And I found BN and dropout are incompatible. That is say, if we put a dropout layer after a BN layer, then the training would fail. Newer network architectures adopt BN instead of Dropout. More details in Section 5.



(a) Train Loss (b) Train Acc (c) Test Acc

Figure 1: VGG training curve

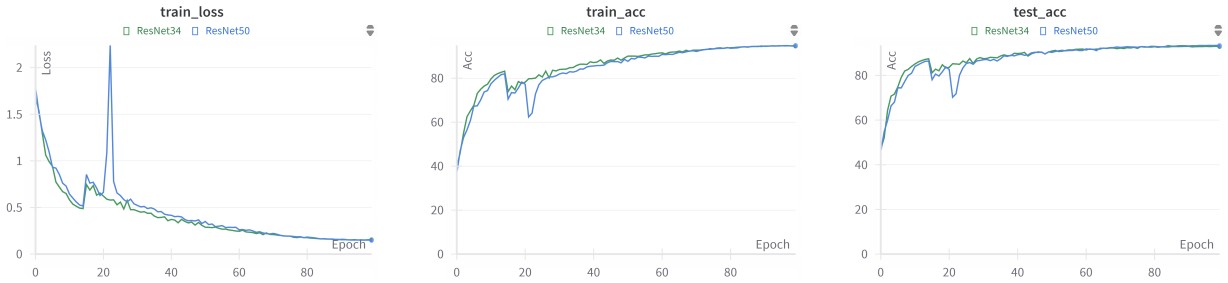
3.2 ResNet

Proposed by Kaiming He et al, ResNets allow for the training of much deeper networks by using skip connections or shortcuts to jump over some layers. The GlobalAvgPool is located after all convolution layers and before fully connected layer.

- **ResNet34** The basic block is the fundamental block used in the construction of ResNet34, a variant of ResNet with 34 layers. Each basic block consists of two layers of 3x3 convolutions, each followed by batch normalization and a ReLU activation.
- **ResNet50** ResNet50 uses a more complex block called the "bottleneck" block designed to reduce the computational complexity. Each bottleneck block has three layers: a 1x1 convolution that reduces the dimensionality, a 3x3 convolution that processes the data, and another 1x1 convolution that increases the dimensionality back to the required level.

3.3 ResNeXt

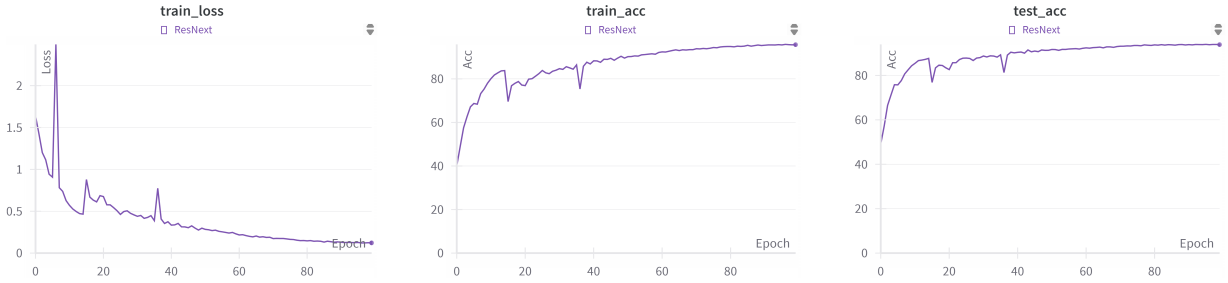
ResNeXt introduces the concept of "cardinality," which refers to the number of independent paths within a block. ResNeXt increases the model's capacity by adding more paths at a low computational cost. We can obtain this block from a ResNet bottleneck by replicating the bottleneck structure



(a) Train Loss (b) Train Acc (c) Test Acc

Figure 2: ResNet34 and ResNet50 training curve

multiple times with the same input. We just need to add an `nn.Conv2d` argument `group` in the 3x3 convolution of bottleneck.



(a) Train Loss (b) Train Acc (c) Test Acc

Figure 3: ResNeXt50_32x4d training curve

4 Specific Configurations

My choice for hyperparameters, optimizer, scheduler and model size are in Table 2.

Term	Value/ Model Size
seed	3407[1]
epoch	100
learning rate	1e-3
batch size	200
Optimizer	AdamW
Sheduler	CosineAnnealing-WarmRestarts

Table 2: The specific config

[1]The reason to choosing such a strange seed is `torch.manual.seed(3407)` is all you need

5 Comparison and Discussion

From Table 1 and Figure 4, we can conclude that ResNext50_32x4d has the best performance on CIFAR10 among the four models though its model size is not largest. Models in the ResNet series tend to converge a bit faster than VGG. The sudden drop in accuracy that occurred at the 15th epoch is due to the scheduler settings; I set the first 15 epochs as a warmup because we did not employ any special initialization methods.

(a) Train Loss (b) Train Acc (c) Test Acc

Figure 4: Model Performance Comparison

I discover some interesting phenomenon and discuss them here:

- **The incompatibility between BN and Dropout**: Because the assignment required the implementation of BN (Batch Normalization) and Dropout in VGG, I initially made the mistake of using them together in the convolutional layers. Specifically, I added a layer of Dropout right after each BN layer. Subsequently, I noticed that the training loss was not decreasing and the test accuracy was only around 20%.

  The issue of incompatibility mainly arises during training. BatchNorm computes the statistics (mean and variance) of the current mini-batch, and these statistics can become noisy and less representative if Dropout is applied before BatchNorm. This is because Dropout randomly drops units, altering the distribution that BatchNorm is trying to normalize.If Dropout is applied after BatchNorm, the scaling factor applied by BatchNorm could be distorted when the activations are randomly dropped, leading to bad learning.

  This can be verified by the paper Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift. In fact, some researchers proposed some methods to combine the two layers: Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks.

- **Test Acc higher than Train Acc**: In Figure 5, we found that for the first 60 epochs, the Test accuracy was higher than the Train accuracy. This seems confusing. My explanation is that the use of extensive data augmentation policies has significantly improved the model's generalization ability, allowing it to easily adapt to the test set. Data augmentation can effectively avoid overfitting.

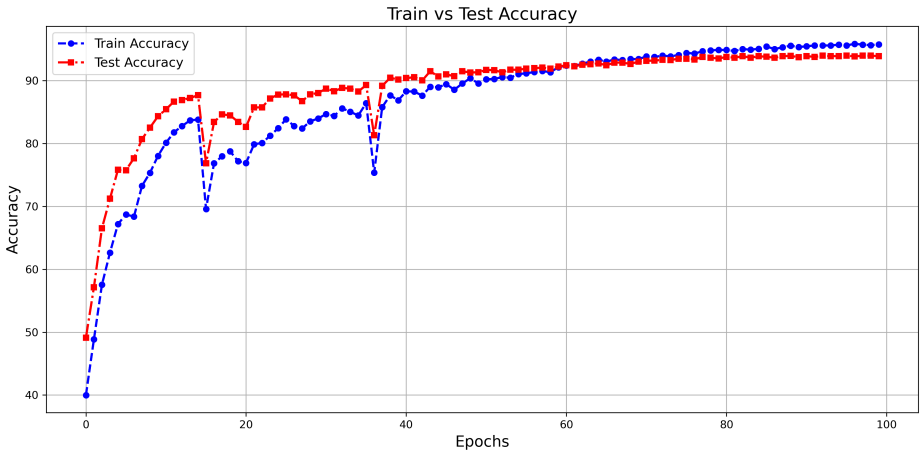


Figure 5: Test Acc vs Train Acc for ResNeXt

- **Smaller kernel size is better:** The standard ResNet uses a 7x7 kernel in the first convolutional layer which leads to a test accuracy about 87%. In my expriments, I replace the 7x7 kernel with 3x3 kernel and get a evident improvment. Smaller kernel size(3x3) has better fitting ability. Smaller kernels may be more effective in processing local features. Since deep learning models abstract local features layer by layer to understand the whole, such a design can better capture image details and local patterns. This may be only suitable for specific dataset.

6 Hand in

The `.zip` file contains:

- `main.py`: code for Train and Test the models.
- `models.py`: code for VGG, ResNet, ResNeXt models.
- `AugmentCifar.py`: code for augmentation policies.
- `report.pdf`: this file.