Lab 1: VGGNet & ResNet final-report

CHI YEONG HEO[1],
[1]School of Computer Science and Engineering, Pusan National University, Busan 46241 Republic of Korea

1. Introduction

This report presents an analysis of the implementation of VGGNet[3] and ResNet[1], and training on the CIFAR-10 dataset[2].

2. Methodology

2.1. Dataset

The CIFAR-10 dataset consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. For the purposes of this experiment, training and evaluation were conducted using a subset of three classes (plane, car, and bird), resulting in 15,000 training images and 3,000 test images.

To enhance model generalization, data augmentation techniques, including random horizontal flipping and random cropping, were applied to the training images. Furthermore, both the training and test datasets were standardized using normalization based on the dataset's mean and standard deviation.

2.2. Implementation Details

Table 1: VGG-16 Architecture

VGG-16 with (mini-)CIFAR-10 config.		
layer name	output size	
conv1_x	32x32	(3x3, 64) x2
pool1	16x16	maxpool
conv2_x	16x16	(3x3, 128) x2
pool2	8x8	maxpool
conv3_x	8x8	(3x3, 256) x3
pool3	4x4	maxpool
conv4_x	2x2	(3x3, 512) x3
pool4	2x2	maxpool
conv5_x	2x2	(3x3, 512) x3
pool5	1x1	maxpool
	1x1	3-d fc

VGGNet (Table 1): The implementation follows the original VGG-16 (Type-D) architecture, with modifications to accommodate the CIFAR-10 dataset, including adjusting the number of output classes to match the dataset. Batch normalization and the Rectified Linear Unit (ReLU) activation function are applied after each convolutional layer to enhance training stability and mitigate vanishing gradients.

ResNet (Table 2): The ResNet-50 architecture is implemented with bottleneck blocks. The initial convolutional layer (i.e., conv1) is different from the one in the paper[1] and the initial max-pooling layer is removed (because the size of CIFAR-10 images is too small). Instead of max-pooling, strided convolutions were applied for down-sampling in the first residual block of each convn_x layer. Additionally, convolutional layers were introduced into shortcut connections when there was a mismatch between the number of input and output channels or when strided convolutions were applied to align the dimensions of the convolutional stack and the shortcut connection.

Training Setup: Both networks were trained using a consistent setup. The training data was processed in batches of 128, with shuffling applied to randomize the order of the samples. For the test data, a batch size of 100 was used, without shuffling. The default weight initialization provided by PyTorch was applied to both networks. The models were optimized using

Table 2: ResNet-50 Architecture

ResNet-50 with (mini-)CIFAR-10 config.		
layer name	output size	
conv1	32x32	3x3, 64, stride 1
conv2_x	32x32	$\begin{pmatrix} 1\times1,64 \\ 3\times3,64 \\ 1\times1,256 \end{pmatrix}\times3$
conv3_x	16x16	$\begin{pmatrix} 1\times1,128 \\ 3\times3,128 \\ 1\times1,512 \end{pmatrix}\times4$
conv4_x	8x8	$\begin{pmatrix} 1\times1,256 \\ 3\times3,256 \\ 1\times1,1024 \end{pmatrix}\times6$
conv5_x	4x4	$\begin{pmatrix} 1\times1,512 \\ 3\times3,512 \\ 1\times1,2048 \end{pmatrix}\times3$
	1x1	average pool, 3-d fc

stochastic gradient descent (SGD) with a learning rate of 1×10^{-2}, a momentum of 0.9, and a weight decay of 5×10^{-4}. VGGNet and ResNet were trained for 20 and 15 epochs, respectively, without the use of learning rate scheduling.

2.3. Evaluation Metrics

The evaluation of both models is primarily based on classification accuracy on the test set, which measures the percentage of correct predictions. Additionally, model performance is assessed using loss convergence and training accuracy throughout the training process. Computational complexity is evaluated in terms of the number of trainable parameters and inference time.

3. Results

3.1. Training Performance

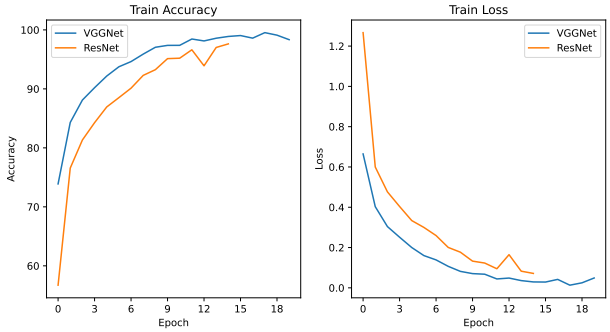

Figure 1: Train accuracy and loss for VGGNet and ResNet across the training epochs.

Both models exhibit convergence trends, indicating that the training configurations were appropriately applied. VGGNet demonstrates higher training accuracy and lower training loss compared to ResNet. However, a direct comparison may not be entirely fair, as the implemented version of ResNet is deeper and more complex than VGGNet, and the number of training epochs is also unequal (20 epochs for VGGNet and 15 epochs for ResNet). Therefore, ResNet may require additional epochs to achieve better convergence.

3.2. Test Set Performance

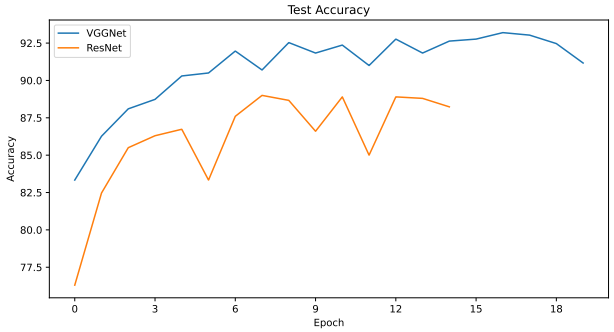

Figure 2: Test accuracy for VGGNet and ResNet across the training epochs.

In alignment with the training performance, VGGNet also achieves higher test accuracy than ResNet over the same period of training.

3.3. Computational Analysis

Model Parameters: VGGNet consists of 14,724,675 parameters, while ResNet contains 23,533,059 parameters.

Inference Time: VGGNet demonstrates significantly faster inference, with an average time of approximately 0.001 seconds per sample, compared to 0.006 seconds for ResNet.

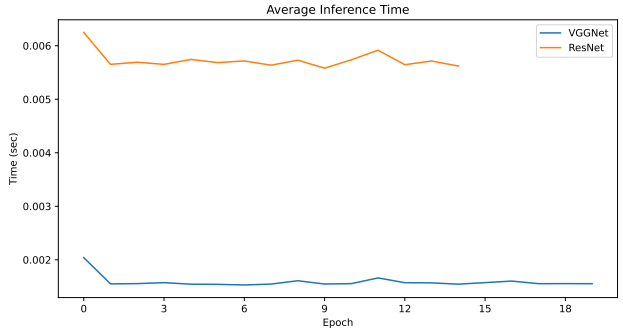

Figure 3: Average inference time for VGGNet and ResNet. Inference time is measured per batch and averaged across the entire test set for each epoch.

Despite VGGNet having fewer parameters and a faster inference time than ResNet in these experiments, this does not imply that VGGNet is more computationally efficient. The implemented models have a differing number of layers (VGGNet has 16 layers, while ResNet has 50 layers). Theoretically, ResNet's residual connections, which use identity mappings, do not introduce additional parameters.

4. Conclusion

VGGNet, with its simpler and more uniform architecture, achieved faster convergence and higher accuracy during both training and testing, as well as a lower inference time due to its fewer parameters. However, these results should be interpreted with caution, as the depth and complexity of the implemented ResNet model were significantly higher, and additional training epochs may be required for ResNet to fully realize its potential.

ResNet, despite requiring more parameters and exhibiting slower convergence initially, offers significant advantages in scalability and deeper architectures. Its residual learning mechanism effectively addresses the degradation problem, which allows for the training of much deeper networks without performance degradation, making it particularly suitable for more complex datasets and tasks beyond the scope of CIFAR-10.

In conclusion, the results indicate that while VGGNet can be more efficient for smaller datasets and tasks requiring quicker training and inference, ResNet might be beneficial for handling deeper architectures and more challenging tasks. To further enhance ResNet's performance, additional training techniques, such as learning rate scheduling, dropout regularization, and more extensive data augmentation, can be applied.

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.