Deeper Networks for Image Classification

Yinghao Qin

1. Introduction

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which requires to classify approximately 1 million images into 1000 different classes, stared in 2010 and has become the industrial benchmark of image recognition [1]. Researches on ILSVRC has pushed back the frontier of the field of computer vision and deep learning.

The goal of the project is to perform image classification tasks as well as possible with deeper networks and try to make some improvements based on the original architecture. In order to make models generalize better, I apply data augmentation methods to training images. The models of GoogLeNet [2] and ResNet [3] are implemented, and I have made some changes to them. It is worth noting that all image classification models in my project are trained from scratch. Moreover, all the models implemented are evaluated on the datasets of MNIST and CIFAR-10, and then the results of them are compared and analyzed.

2. Related work

Deep convolutional neural networks have enabled the field of image recognition to advance in unprecedented pace over the past decade. AlexNet which contains 60 million parameters and 650,000 neurons win the ILSVRC contest in 2012 with a top-5 test error rate of 15.3% [4]. To fight overfitting, it employs the regularization methods – dropout (developed at that time) and data augmentation. [5] do a thorough evaluation of deep networks of increasing depth using an architecture with very small (3×3) convolution filters, which presents that the depth of a net can be pushed to 16-19 weight layers with a significant improvement on the prior-art configuration. The best VGGNet model achieves a top-5 test error of 6.8% in ILSVRC-2014.

GoogLeNet (also known as inception v1), a 22 layers deep network using an improved inception module, gains a championship in ILSVRC-2014 [2]. The main contribution of this architecture is improving the utilization of the computing resources of the network. In fact, GoogLeNet only uses around 5 million parameters, which is far less than the number of parameters used in AlexNet and VGGNet, and it has a better performance in terms of the object recognition error rate. After that, Inception-v2 and v3 are proposed in [6], which adopt the idea of VGG network – using two 3×3 convolution instead of 5×5 convolution to reduce the number of parameters – mitigate overfitting. In addition, batch normalization [7] is used to accelerate the network training as well. Inception v4 [8] introduces the idea of ResNet, mainly used to accelerate the training.

A residual learning framework (ResNet) is proposed to ease the training of deeper networks, which is used to handle the degradation problem [3]. The residual net used on ILSVRC-2015 achieves 3.57% error on the ImageNet test set, which results in the 1st place on the classification task. The core idea of ResNet is to map the lower level feature directly to the higher-level using the shortcut structure. The novel idea inspires many interesting models, such as DenseNet [9] and ResNeXt [10].

3. Approach

In this paper, I use GoogLeNet and ResNet for evaluating the effectiveness of these models for image classification on MNIST and CIFAR-10. In addition, I have made some improvements to the original networks and compared the results of these networks in the following experiment section.

3.1 Data Augmentation

Data augmentation is a popular technique used to enhance the training of networks. In addition, it is also a good method to mitigate overfitting.

As we know the original images in MNIST are size of 28×28 (the original datasets are introduced in Experiment section), they are enlarged into 224×224 size images using bilinear interpolation. The former is used to feed the modified GoogLeNet model, the latter is for training original GoogLeNet model. In addition, a crop of random size and position is made. As shown in Figure 1, the numbers in the top row are the original images, the numbers in the middle row are the expanded images with bilinear interpolation, and the bottom row reveals the randomly cropped images [11].

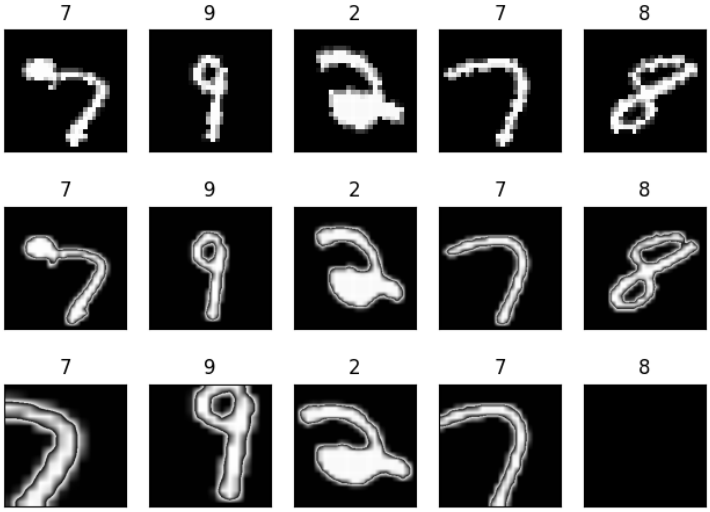

Figure 1. Randomly selected samples of MNIST and their counterparts with different processing

Different from the processing of the dataset MNIST, I take a consideration about the processing of test data in CIFAR-10 as well. As shown in Figure 2, firstly enlarge the size of original images to 224×224 (the first row), the second row shows the results after applying random crop, and there is a 50% probability that the images will horizontally flip after random crop (the third row). Besides, it is noteworthy that I make a process on the test data as well (which is only used in the ResNet experiment), containing expanding the 32×32 image to 256×256 and then make a center crop of the size 224×224.

Figure 2. Randomly selected samples of CIFAR-10 and their counterparts with different processing

3.2 Model Architecture

(I). Modified GoogLeNet

On the basis of the original architecture, I have modified it from two aspects. First, design a modified architecture fit the size of input data. Second, batch normalization is applied in order to accelerate the training process and reduce the risk of overfitting.

Modify the model structure more compact to fit the smaller size input data. The original architecture was designed for the ImageNet dataset (the image size is 224×224 pixel), whereas 28×28 pixel is the size of images in MNIST. Generally, there are two ways to deal with the problem - using interpolation to enlarge the image or designing the structure of the model directly to satisfy the input image of smaller size. The detailed design is shown in Table 1 in Appendix, referencing to [2][7]. Furthermore, according to the theory of [2], the ratio of 3×3 and 5×5 convolutions in the inception are designed to increase as moving onto higher layers. This is because as features of higher abstraction are captured by higher layers, their spatial concentration is expected to decrease.

In addition, a batch normalization layer is added between a convolutional layer and its ReLU activation function. Batch normalization recommended by [7] can reduce internal covariate shift phenomenon which slows down the training owing to requiring lower learning rate and careful parameter initialization. Besides, this method regularizes the model, which is a good option to cope with overfitting. Although dropout is a typical approach to reduce overfitting [12], in a batch-normalized network it can be either removed or reduced in strength.

(II). Modified ResNet

In this part, I replace all rectified activation units in the original ResNet with Parametric Rectified Linear Unit (PReLU), which is proposed by [13]. Compared with the traditional rectified unit, PReLU improves model fitting nearly without extra computational cost and little overfitting risk. As shown in Figure 3, for PReLU, the coefficient of the negative part is adaptively learned, which is the most different from ReLU.

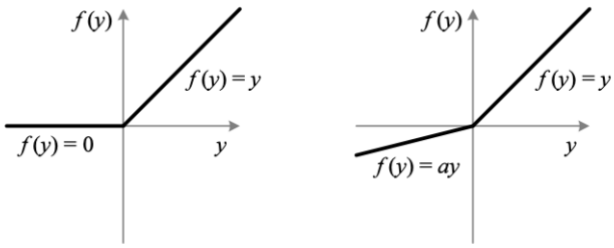

Figure 3. ReLU vs. PReLU [13]

4. Experiment

I train all the models from sratch on Quadro RTX 6000/GeForce RTX 2070, and only train 30 epoches per model. All the models are constructed using Pytorch. The detailed setting for these models is described in the following. All the models use a batch size of 32. For GoogLeNet model, Cross Entropy loss function and Adam optimizer (learning rate is set as 0.0003) are used, whereas the modified version uses SGD optimizer with learning rate of 0.01 and momentum of 0.9. For both ResNet model and its variant, Cross Entropy loss and Adam optimizer (learning rate is set as 0.0001) are used.

4.1 Datasets

In this project, the MNIST and CIFAR-10 datasets are used. The MNIST database [14] of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. The CIFAR-10 dataset

contains 60,000 32×32 colour images in 10 different classes, with 6,000 images per class. Some randomly selected samples are shown in Figure 1 and Figure 2.

4.2 Testing Results

The experimental result is described as follows: First, explore whether the data augmentation works; Second, whether the modified models performs better than the base ones in terms of specific aspects; Third, compare the performance of GoogLeNet and ResNet under the same setting.

Table 2. The Best Top-1 Test Accuracy of Model on MNIST

	Base	Modified	Base + Aug	Mod. + Aug
GoogLeNet	99.6%	99.5%	99.5%	
ResNet	99.6%	99.6%	99.5%	99.6%

Evaluate these models on MNIST. According to Table 2, all of the models achieve nearly the best accuracy of 99.5% in test set. From Figure 4, only the train accuracy using data augmentation is far lower than other accuracies by roughly 10%, which is because using randomly crop make the original digits lose much useful information (e.g. the number 8 in Fig. 1 is totally pointless).

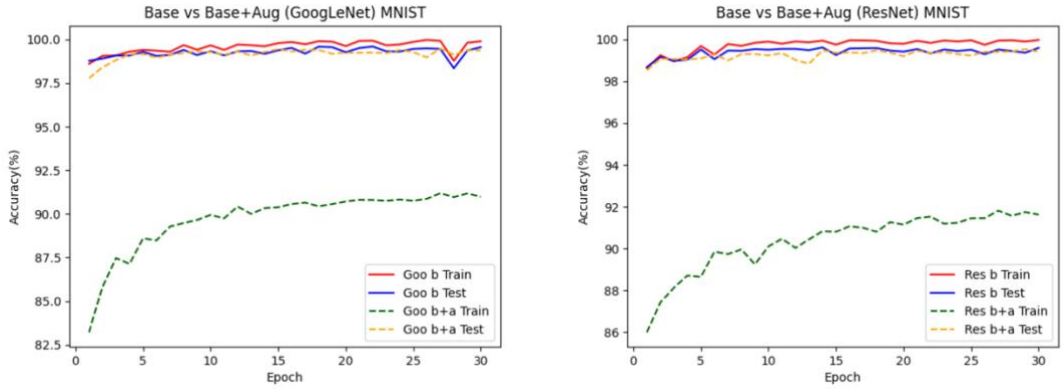

Figure 4. Accuracy comparison between model with and without data augmentation

Moving onto Figure 5, for modified GoogLeNet, there is a slight rise on both of the train and test accuracy (the green dotted line is much higher than the red solid line throughout the training process and the orange dotted line is slightly higher than the blue solid line). Besides, it is noteworthy that the total training time modified GoogLeNet spent is 39.95 minutes which is around 3/4 less than that time the original GoogLeNet spent (108.03 minutes). The detailed running screen shot is placed in Figure 12, 13 in Appendix. The modified ResNet does not gain obvious improvement on MNIST. Finally, from Figure 6, it is clear that ResNet performs better than GoogLeNet does, since ResNet is earlier to achieve a relatively higher accuracy (the green and orange dotted line is higher than red and blue solid line over the first part of the training).

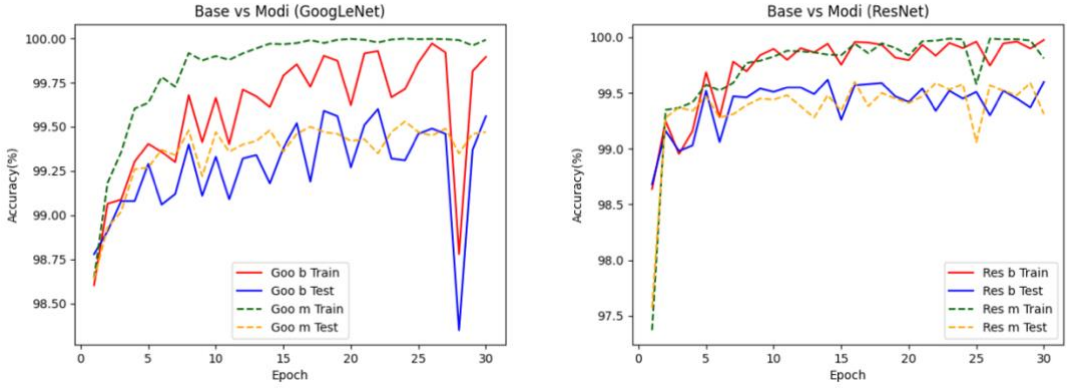

Figure 5. Accuracy comparison between base and modified model

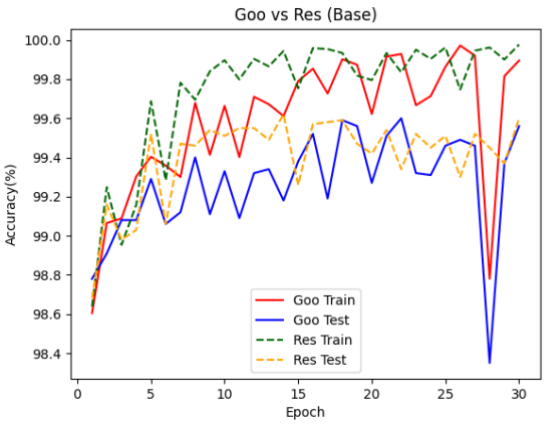

Figure 6. Accuracy comparison between base GoogLeNet and ResNet model

4.3 Further Evaluation

Further evaluation is done on CIFAR-10. According to Table 3, using data augmentation obviously improves the performance of models on CIFAR-10, though the test accuracy of GoogLeNet using augmentation is 0.4 percentage point lower than that without data augmentation (this is because I just train 30 epochs for all models and the concrete reason will be explained in the following part). Moreover, for ResNet the best test accuracy is the highest (92.2%), which means the modified model and data augmentation work. In details, compared with the base ResNet, the accuracy of the modified version goes up by 0.5% and that of the model using augmentation increases by 5.9%.

Table 3. The Best Top-1 Test Accuracy of Model on CIFAR-10

	Base	Modified	Base + Aug	Mod. + Aug
GoogLeNet	84.0%		83.6%	
ResNet	84.9%	85.4%	90.8%	92.2%

From Figure 7, the left part shows GoogLeNet with data augmentation seems perform worse, but the problem is that inadequate training is done (the trend shows that the dotted lines would exceed the solid ones if training more epochs). In the right part, the test accuracy of the model using data augmentation exceeds that without augmentation at about 10th epoch, which also strongly support the correctness of the above point. From Figure 8, there is a little improvement for ResNet using PReLU. Figure 9 proves that ResNet performs slightly better than GoogLeNet do on the top-1 accuracy again.

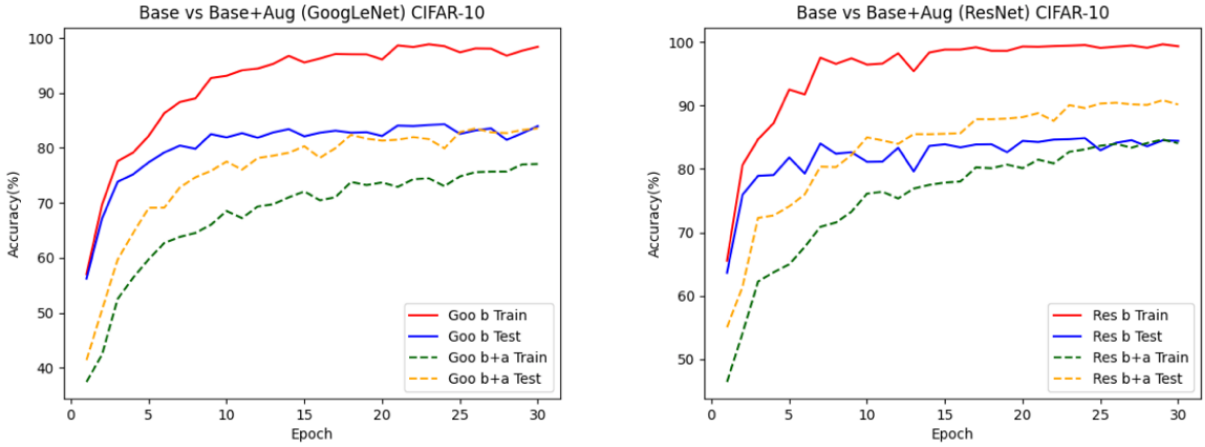

Figure 7. Accuracy comparison between model with and without data augmentation

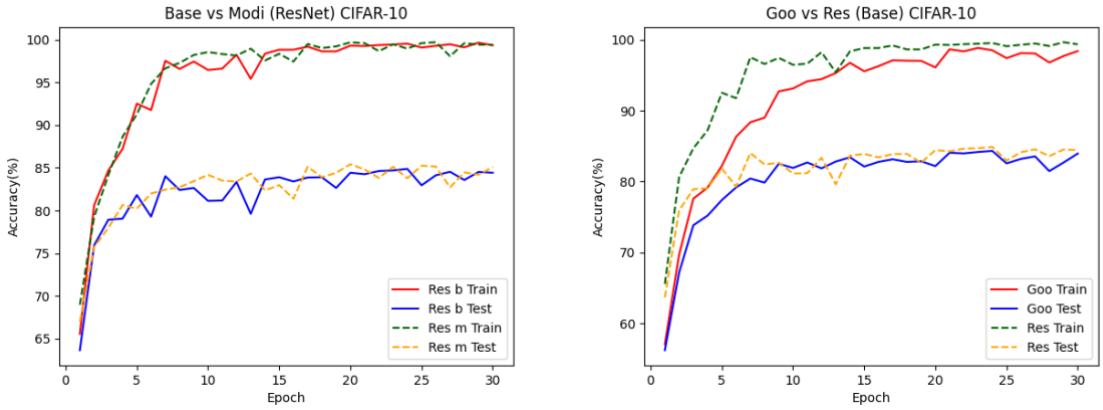

Figure 8 & 9. Accuracy comparison between base and modified model, between GoogLeNet and ResNet

Finally, the running results of the model achieving the best accuracy in Table 3 (the modified ResNet model with data augmentation) are shown in Figure 10 and Figure 11. From Figure 10, it is clear that the training is not sufficient, more epochs should be taken for training.

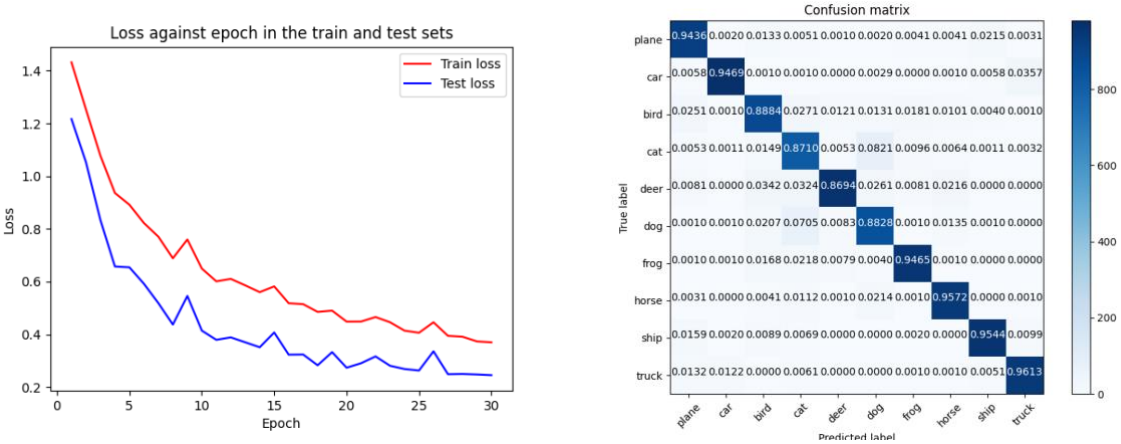

Figure 10 & 11. Loss-epoch plot, Confusion matrix of the modified ResNet with data augmentation

5. Conclusion

The experimental results yield a solid evidence that compact structure and batch normalization help GoogLeNet improve performance in terms of training time and error rate, meanwhile verify that ResNet applying PReLU does ease overfitting. In addition, data augmentation can enhance the training of the deep networks, but there is a risk of missing useful information.

For this coursework, I perform image classification tasks with GoogLeNet and ResNet and evaluate them on MNIST and CIFAR-10. In the meantime, do some modification to the original network architectures to enhance their performance.

Reference:

[1] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC. Imagenet large scale visual recognition challenge. International journal of computer vision. 2015 Dec 1;115(3):211-52.

[2] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 1-9).

[3] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).

[4] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. InAdvances in neural information processing systems 2012 (pp. 1097-1105).

[5] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.

[6] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 2818-2826).

[7] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167. 2015 Feb 11.

[8] Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, inception-resnet and the impact of residual connections on learning. InThirty-first AAAI conference on artificial intelligence 2017 Feb 12.

[9] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 4700-4708).

[10] Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 1492-1500).

[11] Wu J, Zhang Q, Xu G. Tiny ImageNet challenge. Technical report, Stanford University, 2017. Available online at http://cs231n. stanford. edu/reports/2017/pdfs/930. pdf. Last accessed on 14-06-2019; 2017.

[12] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 2014 Jan 1;15(1):1929-58.

[13] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. InProceedings of the IEEE international conference on computer vision 2015 (pp. 1026-1034).

[14] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998 Nov;86(11):2278-324.

Appendix

type	**patch size /stride**	**output size**	**depth**	#1×1	#3× 3 **reduce**	#3×3	#5×5 **reduce**	#5×5	**pool proj**
convolution	5×5/1	28×28×8	1						
convolution	3×3/1	28×28×32	0						
inception(3a)		28×28×64	2	16	24	32	4	8	8
inception(3b)		28×28×120	2	32	32	48	8	24	16
max pool	3×3/2	14×14×120	0						
inception(4a)		14×14×128	2	48	24	52	4	12	16
inception(4b)		14×14×128	2	40	28	56	6	16	16
inception(4c)		14×14×128	2	32	32	64	12	16	16
inception(4d)		14×14×132	2	28	36	72	8	16	16
inception(4e)		14×14×208	2	64	40	80	8	32	32
max pool	3×3/2	7×7×208	0						
inception(5a)		7×7×208	2	64	40	80	8	32	32
inception(5b)		7×7×256	2	96	48	96	12	32	32
avg pool	7×7/1	1×1×256	0						
dropout (40%)		1×1×256	0						
linear		1×1×10	1						
softmax		1×1×10	0						

Table 1: The architecture of modified GoogLeNet for MNIST

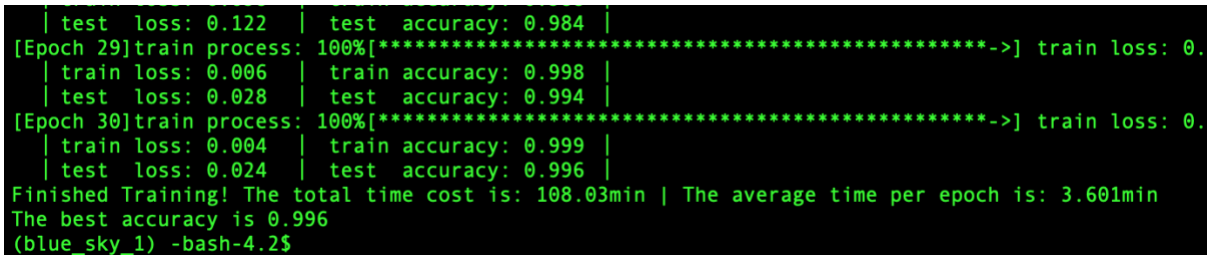

Figure 12. Run-time screenshots: the modified GoogLeNet model

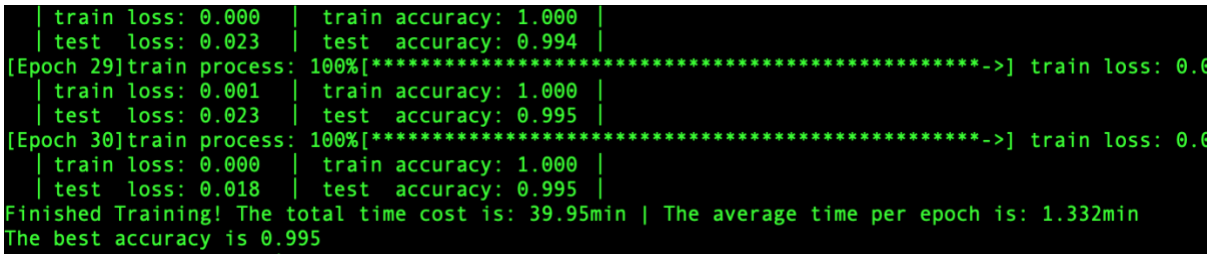

Figure 13. Run-time screenshots: the original GoogLeNet model