ELEC5307 Deep Learning Project 2: Fruit Image Classification

*Note: Sub-titles are not captured in Xplore and should not be used

Mengwei Yuan
460101132
myua7783@uni.sydney.edu.au

Zhengyang Zhao
460321110
zzha9698@uni.sydney.edu.au

Rouwei Chen
470132579
rche7843@uni.sydney.edu.au

***Abstract*—Deep neural network approach has shown tremendous potential in various contemporary image classification tasks. As one of the state-of-art network architecture, the 50 layers residual neural network (ResNet50) achieves a high classification accuracy on a range of image sets. However, the pre-trained network is not generalized for every data set, which requires further training for the model to adapt the feature or characteristic of a certain image set before performing the classification. This study conducts an investigation on this network and develops an effective training procedure for a pre-trained ResNet50 model on a given 18-class image set. Several variable-control experiments are set to analyze and compare the effect of different training parameters on model performance. Based on these findings and corresponding background theory, by implementing some data pre-processing techniques and applying an advanced learning rate scheduler with adjusted batch size, we successfully finetune model and achieves a 98% classification accuracy within 20 epochs of training.**

***Index Terms*—Deep Neural Network, ResNet50, Image Classification**

I. Introduction

This aim of this project is to select a proper deep neuron network and fine-tune the parameters until reached the desired accuracy. The training set contains 2625 fruit images in 18 categories; the validation set contains 356 images from the same 18 categories. The number of pictures of each category in the training set varies from 111 to 182, but most of the categories contain 140 to 150 pictures, which is considered reasonable. The given images are all taken from real-life scenarios, so the classifier should be able to handle the disturbing background information, the partial images, the different lighting conditions, etc. Both the training set and validation set are labelled data since it is a supervised training procedure. The details of the data set are discussed in IV.A.(1) Image size and target object location.

To build the desired classifier, we compared the performance of existing pre-trained models between AlexNet, GoogleNet, ResNet with different depth, and finally, the ResNets are selected because of their excellent performance on image classification, although the training speed is slower and the required memory size becomes larger. In the II. previous work A. Different DNNs performance comparison. The detailed structure and analysis of the ResNet are discussed in the III. Backbone network.

The most important parts of the experiment are data pre-processing and parameter tuning, which give the most significant influence on accuracy. The pre-processing procedure is done in the transform part, which includes image crops, resize, colour jitter, greyscale, affine, flip and normalization. We considered the feature of fruit dataset and finally chose random crop, centre crop, colour jitter, vertical/horizontal flips, affine and normalization as our experiment choices, as discussed in IV.A. Training set characteristic. For parameter tuning, we mainly tuned the learning rate, tried different optimizer and schedulers, the theoretical analysis is discussed in the II. previous work part and the detailed experiment.

II. Previous Work

A. Different DNNs Performances Comparison

As image recognition and computer vision is a hot topic in the AI field, and will be widely used in the unmanned industry and auto driving applications, several image processing networks have been developed and the pre-trained models are provided. The most commonly used networks are AlexNet, GoogLeNet, ResNet, VGG, inception. The performance of each network as well as the training speed are shown in Figure 1 and Figure 2 [1]. The network choose often need to consider the compromise between higher performance and faster training speed, However, the resNets show a relative good result on performance and acceptable training speed.

B. Data Pre-processing

The performance of the neuron network is highly depend on the input dataset, so before feed the data into the network, generally the data will be pre-processed by the methods listed in Table I.

Preprocessing method	**Different choices**
Resize	
Crop	*CenterCorp, RandomCrop, FiveCrop*
Flip	*RandomHorizontalFlip, RansomVerticleFlip*
ColorJitter	*Brightness, Contrast, Saturation, Hue*
Affine	*RandomAffine*
Normalization	*Mean, sd*

TABLE I

• Crop: when applying cropping to the images, the input set is enlarged as one image is cropped to multiple pieces.

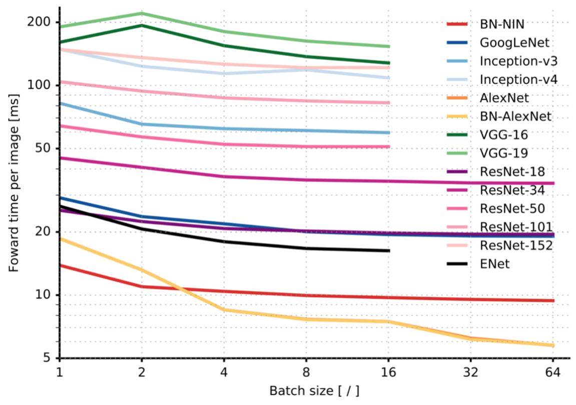


Fig. 1. [1]

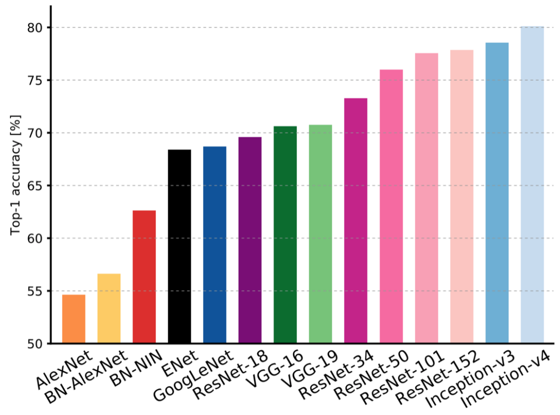


Fig. 2. [1]

The crop method may leave the key features in the dataset and remove the disturbance information from the background, which will increase the accuracy of recognition; however, it may also crop out the objects but leave the useless part to the input set. Generally, before selecting the crop method, it's necessary to check the condition of dataset and choose the crop method accordingly.

• Resize: resize is necessary as most of the neuron networks are designed to take the input with a certain size, for example, the ResNet only take the image of 224*224. However, the resize method may change the shape of the image, which will give a negative effect to the classifier.

• Flip and affine: these two methods are not so commonly used as other methods, as rotate and inverse the images will not make significant improvement to the result.

• Normalization: it normalize the pixel intensity based on the given mean and standard deviations as $image = \frac{image - mean}{sd}$ .The most commonly used normalization parameter is mean = [0.485, 0.456, 0.406], sd = [0.229, 0.224, 0.225].

C. *Weight-related Parameter Tuning for Network Training*

Other parameters that has the most significant effect on the result is the learning rate. The learning rate should be reduced during the training procedure so that it will not jump the optimal point thus result in the vibrating in the result. However, the learning rate cannot be too small, or it may be stuck in the local minimum. There are two functions adjusting the learning rate, i.e. the optimizer and the scheduler. The choice and analysis are listed in Table II.

Optimizer	*RMSprop,Adam,SGD,Adadelta*
Scheduler	*Exponential,Step,multistep,lambda*

TABLE II

1) Optimizer: • Adagrad: it wisely adjust the learning rate to different weights, so that most of the weight can obtained most suitable learning rate, thus result in fast converge.

• RMSprop: it shows the best performance among the commonly used optimizers. It is updated from the Adagrad optimizer.

• SGD: SGD provide a method to set all the parameters manually, which may cause the slow convergence but high performance.

• Adam: it is a very commonly used method for learning rate adjusting as it has fast convergence and relatively good generalization.

2) Scheduler: Schedulers run after the optimizer to give a further adjustment to the learning rate. Exponential scheduler adjust the learning rate as $lr = lr \times \ gamm^2$ while step drops the learning rate by half in each 10 epochs[2]. The performance comparison of all the optimizers and schedulers are shown in Figure 3:

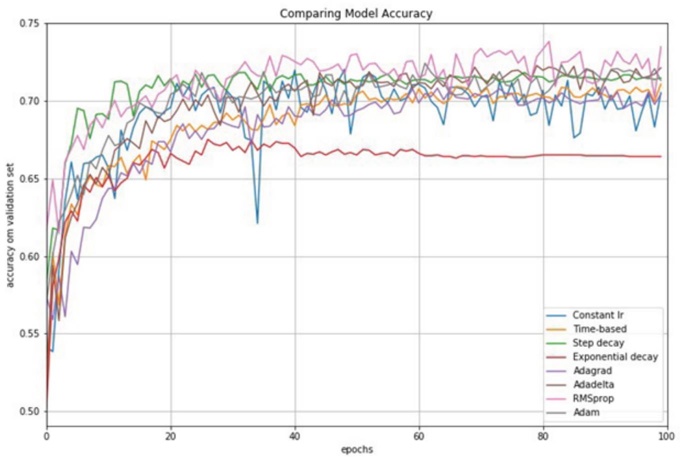


Fig. 3.

III. Backbone Network

The network is based on the trained ResNet50. We perform the modifications on its output layer, and training procedure as well.

A. *Network Structure*

Initially the 18-layer ResNet was used to train the model, however, after comparing the performance and time cost, we found that the 50-layer ResNet has a better efficiency for our dataset. The construction of a ResNet block consists of 3 layers, each contains a convolutional layer with 3×3 filters. A shortcut connection is inserted based on that. The input image of the net holds the size of 224×224 and the output of the trained ResNet50 is 1000. However, the fruit classifier has 18 collections, we changed the last fully connected layer, which is the output layer, into a linear function with 2048 inputs and 18 outputs. The additional training set added to the network retrained the weight of this layer.

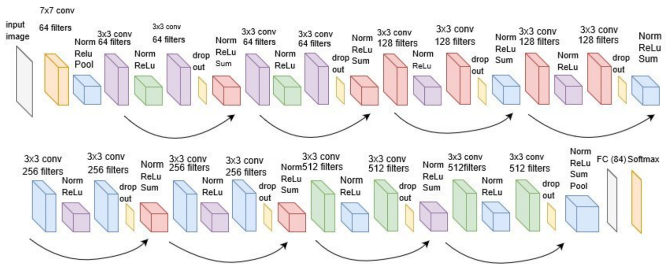
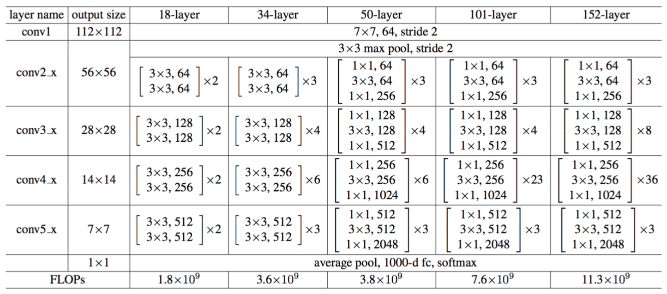


Fig. 4. [3]



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 5. [3]

B. Training Procedure

There are some pre-processing methods applied to our training set at the beginning, which include cropping and flipping, for the purpose of dataset augmentation. Color modification on brightness, contrast and saturation taken on the training set also contributes to the performance of the model.

For the time efficiency of the training process, we set the number of epochs to be 40 to ensure a proper speed of converge progress, as we want to get high performance model in a reasonable period time.

The model is optimized by SGD with the momentum of 0.9. Implement of stochastic gradient descent accelerate the converge and simplifies calculation. And by implementing the comparative experiments, we can get the best classification accuracy at when the batch size is 16 and the learning rate is initialized 0.0003 with a cosine annealing scheduler.

IV. Experiments and Analysis

Since the model has been pre-trained, the core architecture of the model is reserved. The experiment mainly focused on the data pre-processing and training procedure refinement. There are three stages in the experiment. We firstly investigate the characteristics of the training set and the data distribution, which is essential for the choice of data pre-processing techniques. Then the parameters of learning rate and optimizer were adjusted to fit the optimization processes. After these parameters were set, in stage three, the batch size was fine-tuned together with the corresponding adjustment of the learning rate.

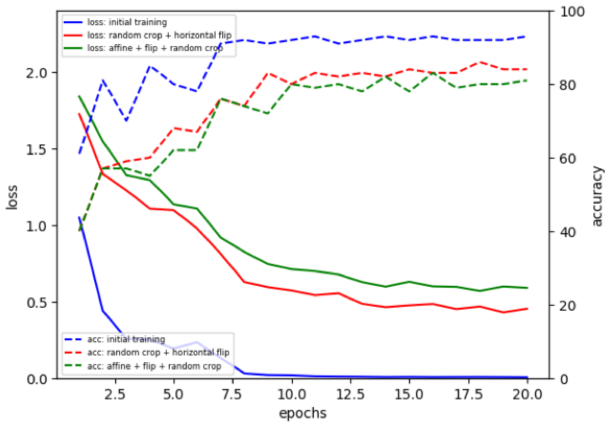


Fig. 6.

A. *Training set characteristic*

1) Image size and target object location: The training set contains 18 classes that represent the images of different types of fruit. Most of the image is in a size larger or equal to 600*800 with a rectangle shape. However, the pre-trained model we used (ResNet50) only allows an input size of 224*224, which implies that the input image for this training set should be either cropped or resized. Therefore, the initial or baseline setting is tested by only resizing the image at data pre-processing stage. The setting for rest training parameters are illustrated in Table III. The initial training achieves a 93% accuracy on the validation set (Figure 6)which indicates that the resizing operation successfully includes the features which are well learned by the network.

Training Setting	**Name**	**Parameters**
Optimizer	*Adam*	*lr = 0.0005* *decay = 0*
Learning Rate Scheduler	*StepLR*	*Step size = 7* *gamma = 0.1*
Batch size		*64*
Loss Function	*Cross-Entropy Loss*	
Epoch		*20*

TABLE III

In order to further improve the classification accuracy and make the model has the ability to classify an image based on a local feature or small piece of information, the random crop data pre-processing operation is implemented. On the other hand, the random crop could also substantially increase the training set diversity and better avoid the overfitting problem. The horizontal flipping is also included in this experiment together with the random cropping for the training set diversification. Before making these modifications, an investigation on the location of objects was conducted on the training set. Figure 7 shows the randomly sampling the images from the training set, in which most of the target object takes more than 1/3 of the entire image and located in the center area. This feature indicates that the target object is likely to be contained in the cropped image after random crop operation.

Fig. 7.

However, the performance of the classification accuracy decreased (86% accuracy) after implementing the random cropping and horizontal flipping. From figure 6, after 20 epochs of training, both accuracy and loss are worse than before. Theoretically, this training curves will converge slower than the previous experiment because the training set is enlarged by data augmentation, it requires more training time and epochs to reach the optimal and leads to a better accuracy and stronger classification ability. Considering that the data augmentations are not diverse enough, another two image transformation methods were then added which are affine and vertical flipping but with an even worse performance (81% accuracy) shown in Figure 6.

This decreased performance with the combination of different transformation reminds us that some training set data augmentation may not suitable for the validation set (in the previous experiment, same transformations were used for both training set and validation set). In terms of the data augmentation techniques with a random probability such as random cropping, it may lead to the instability of validation image features so that hard to achieve high accuracy. The transformation for the validation set, therefore, is changed to the center cropping with a size of 224 after resizing the image to 256 on the shorter edge and keeping the original aspect ratio. This operation is also based on the feature that in the training set, the objects were contained in the centre area.

2) Color channels of the images: We also studied the color information of the training set. As shown in Figure 8, some images that are similar in their color distribution. When the network learned the color distribution as the feature of a certain class or put more weight on the color feature when making the classification decision, the image is likely to be mis-classified.To force the network focusing on the learning of image shape and other critical features, a color augmentation method is added which randomly adjusts the brightness, contrast and saturation of an image by scaling these values by a factor sampled from a distribution with the range from 0.6 to 1.4. As mentioned before, since this transformation include random probability, it only applied on the training set so that it maintains the feature stability of the validation set.

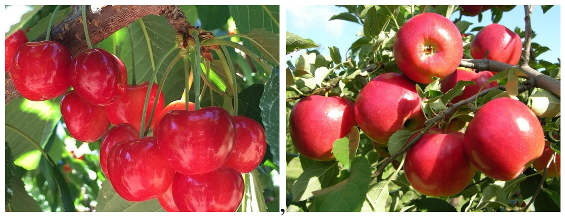


Fig. 8.

In addition to this, we normalize the RGB channel of images by the equation, where the mean for RGB channels are 0.485, 0.456, 0.406 and standard deviation are 0.229, 0.224, 0.225 respectively. These values is calculated based on the image net data set [4]. The normalization is applied on both training set and validation set since this operation does not include any random factor.

$$normalized\ color = \frac{input[channel] - mean[channel]}{std[channel]}$$

The performance after implementing the color augmentation together with the modification on the test set transform is shown in Figure 9. After adjust these data preprocessing parameters, the performance of network which trained based on the initial setting in Table III achieves a high accuracy of 96% on validation set.

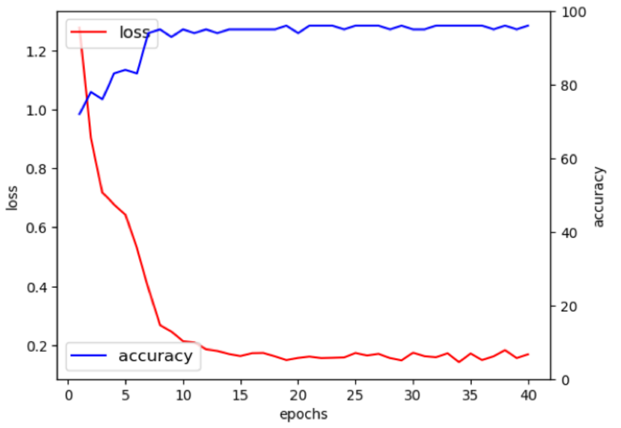


Fig. 9.

B. *Learning rate and Optimizer*

1) Learning rate scheduler: Based on the initial training parameters shown in Table III, the network has already trained with a strong classification ability. Thus, the learning rate of 0.0005 is considered as a reasonable setting for training. However, the training curve indicates that the training converged very fast after 8 epochs of training. Although the step learning rate scheduler reduces the learn step at 7th epochs and boosts the accuracy from 83% to 95%, the step at 14 epochs did not have any affect the learning process as the process is already converged.

In order to refine the learning process and smooth the learning step so that further improve the performance and make the model converges on a pricised global optimal, the cosine annealing learning rate scheduler is introduced. The cosine scheduler modifies the learning rate every epoch based on the equation.

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{T_{cur}}{T_{max}}\pi))$$

This scheduler maintains a high learning rate at the beginning where the model is far from the optimal point. It decreases the learning rate in a linear-like rate (the curve has a slope of 1 at the middle and remain nearly one at a certain period) when the learning process is approaching the optimal. When the learning process is close to the optimal, the learning rate would be almost 0 which allows the model to learning in a very meticulous way that only takes a small step towards the gradient direction and increase the possibility to further close to the true optimal (Figure 10).

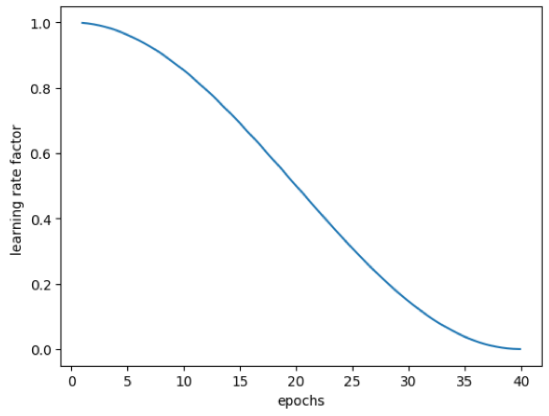


Fig. 10.

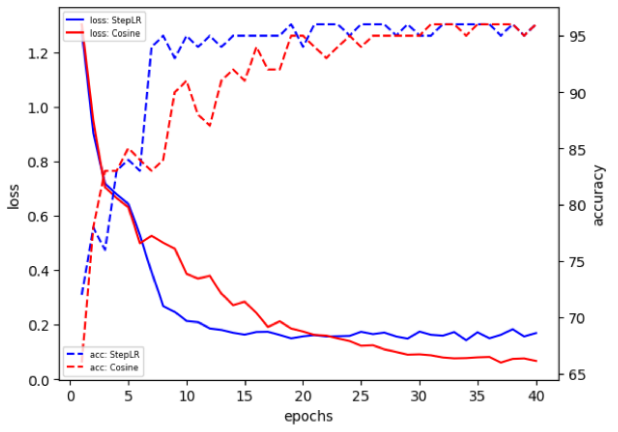


Fig. 11.

The training comparison between cosine and step scheduler is shown in Figure 11. The optimization process is smoothed by the cosine optimizer and stabled at the 96% accuracy performance. Then another concern is raised that the adam optimizer has the self-adapted actual learning rate. The combination of cosine scheduler and adam optimizer might decrease the smoothness of the learning rate decay. Therefore, the Stochastic Gradient Descent (SGD) optimizer is then implemented with a momentum of 0.9.

The best performance achieved is increased to 97% by using the SGD optimizer, which concludes the final parameters for learning rate and optimizer. Table IV shows the comparison between different parameters.

Parameters	**Adam + StepLR**	**Adam + Cosine**	**SGD + Cosine**
Accuracy	*96%*	*96%*	*97%*

TABLE IV

C. Batch size

The previous experiment is based on the 64 batch size training which only few steps at each epoch as the number of steps depends on the size of both training set and batch size. We considered that few steps of training may decrease the possibility that the optimization step into the true optimal point. On the other hand, as the training set is shuffled each time, the randomicity of optimization direction exist on each optimization step, a large number of optimization steps will potentially improve the performance since it provides more opportunities for learning process to reach the optimal point. However, too small batch size will lose the global feature of the training set so that increase the randomicity too much. In this situation, the learning process is hard to converge.

To explore the trade-off between them, a set of experiments based on the fine tuned parameters is conducted on 4 different batch sizes which are 64, 32, 16 and 8 (with corresponding learning rate 0.001, 0.0005, 0.0003, 0.00015). The comparison between them is shown in Figure 12. The best result reaches 98% on both 16 and 8 batch size training. The 16 batch size is adopted as the final parameter as it has a more stable performance on the classification accuracy during the training and first reach 98% on 17th epochs.

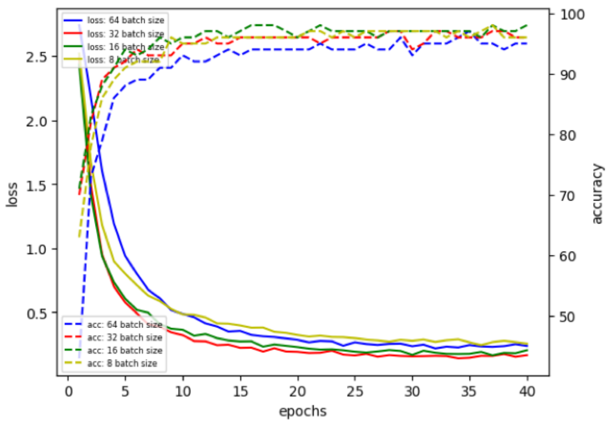


Fig. 12.

V. Conclusion

The best result we got is 98% by using resNet50 with batch size 8 or 16 and trained in 40 epochs. The best combination we found for the learning rate is lr=0.0003 with using the SGD optimizer and cosine schedular. the preprocessing applied to the dataset includes color jitter adjustment, random crop and normalization with the most popular means and standard deviations illustrated in the previous work part. During the parameter procedure, after the input data are properly processed, the result is prompted; after which, the tuning of learning rate as well as the proper choice of optimizer and scheduler also give a better accuracy and less loss; also, a suitable batch size makes the result improve slightly.

VI. Future Work

In this experiment, limited by the time and GPU size, we only explored the parameter tuning for resNets. In the future, we can try to change the layers and parameters in each layer, then find out how each layer contribute to the result. Currently, the training set is well labeled images. However, image labeling is a tedious and time-consuming work, so in the future the desired solution to the real-world problem might be an unsupervised training procedure.

References

[1] A. Canziani, E. Culurciello, A. Paszke, “AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS”arXiv:1605.07678 [cs.CV]

[2] G. Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning. https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1

[3] ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks. https://neurohive.io/en/popular-networks/resnet/

[4] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, M. Li, “Bag of Tricks for Image Classification with Convolutional Neural Networks” arXiv:1812.01187 [cs.CV]