Homework 1: Backpropagation and Neural Networks

[redacted]
Department of Electronic Engineering
Tsinghua University
[redacted]

1 Part One: Backpropagations

Detailed derivations are given in the appendant file.

2 Part Two: Code Implementation

2.1 MLP

2.1.1 Task 1

Forward and backward algorithms are implemented in "mlp.py". The parameter initialization is based on Kaiming initialization[1]. Set the hyper-parameters as id=1 in Table 1 and run "mlp.py" directly. The result of backpropagation check is shown in Fig. 1, which validates the BP algorithm. The training loss curve is shown in Fig. 2. It can be found that the training is almost converged after 2000 iterations. Top-1 accuracy on training and test set are 93.55% and 91.46%, respectively.

```
Hyper-parameters:
Namespace(hidden_dim=100, lr=0.01, batch_size=32, epochs=30)
Dataset information:
training set size: 10000
test set size: 5000
Gradient check of backward propagation:
Relative error of dw2 4.187937577951869e-12
Relative error of db2 2.453216369394189e-13
Relative error of dw1 1.7169802871146034e-13
Relative error of db1 3.478197481975123e-14
If you implement back propagation correctly, all these relative errors should be less than 1e-5.
```

Figure 1: Gradient check for BP algorithm

2.1.2 Task 2

In this part, different hyper-parameter combinations are tried (from id=2 to 13) and the results are reported in Table. 1. It can be found that id=11 and 13 achieve the highest training and test accuracy (99.7% and 94.24%), respectively.

Some conclusions can be obtained from Table. 1:

1. As the batch size decreases, MLP achieves better performance in both training and test set, since smaller batch size increase the randomness in BP algorithms and it might lead to a better local minimum.
2. As the hidden dimension increases, better accuracy can be achieved when other parameters are fixed. MLP is not overfit when hidden dim equals 100. Thus, we can increases the dimension to 150 for better network expressiveness. However, overfitting takes place when hidden dimensions are increased to 200.

Preprint. Under review.

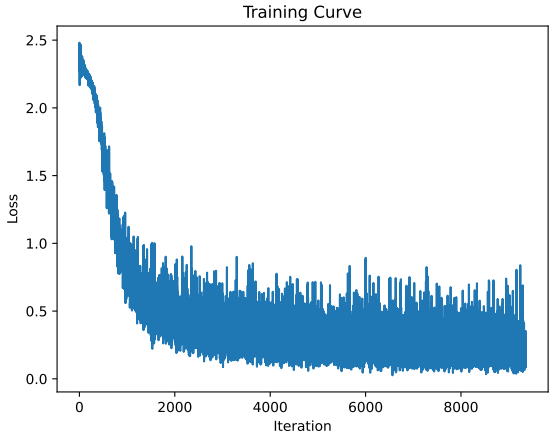


Figure 2: Training loss curve for MLP

Table 1: Hyper-parameters and accuracy in MLP

id	hidden dims	learning rate	batch size	epochs	Top-1 train acc (%)	Top-1 test acc (%)
1	100	0.01	32	30	93.55	91.46
2	100	0.01	16	30	95.86	92.64
3	100	0.01	8	30	98.04	93.62
4	150	0.01	32	30	93.84	92.00
5	150	0.01	16	30	96.03	93.00
6	150	0.01	8	30	98.38	93.92
7	150	0.005	32	30	91.56	90.44
8	150	0.005	16	30	93.46	91.62
9	150	0.005	8	30	95.98	93.12
10	150	0.005	8	50	97.8	93.5
11	150	0.005	8	75	99.23	94.44
12	150	0.005	8	100	99.12	94.2
13	200	0.005	8	100	99.7	94.24

3. As the learning rate decreases, MLP converges slower. Thus, more epochs are needed in the training stage to guarantee convergence.

2.2 CNN

In this part, all experiments share several hyper-parameters and optimizer, as shown in Table. 2.

Table 2: Hyper-parameters and experiment settings CNN

num classes	input size	batch size	learning rate	optimizer
10	224	36	0.001	SGD(momentum=0.9)

Table 3: Experiment settings and accuracy in CNN

id	model	L	k	θ	data aug.	learn stra.	epochs	train acc (%)	test acc(%)
1	ResNet18	\	\	\	\	\	100	86.40	81.39
2	DenseNet	4	12	1	\	\	100	84.50	81.09
3	DenseNet	4	24	1	\	\	100	85.24	82.27
4	DenseNet	4	36	1	\	\	100	87.10	84.00
5	DenseNet	6	12	1	\	\	100	84.10	81.70
6	DenseNet	6	24	1	\	\	100	85.4	82.17
7	DenseNet	6	36	1	\	\	100	86.89	82.09
8	DenseNet	6	12	0.75	\	\	100	84.8	81.79
9	DenseNet	6	24	0.75	\	\	100	87.08	82.98
10	DenseNet	6	32	0.75	\	\	100	88.2	83.67
11	DenseNet	4	36	1	\	\	150	91.46	85.07
12	DenseNet	4	36	1	cyclic shift	\	150	90.0	85.83
13	DenseNet	4	36	1	\	step	150	93.98	86.61
14	DenseNet	4	36	1	cyclic shift	step	150	94.27	87.09

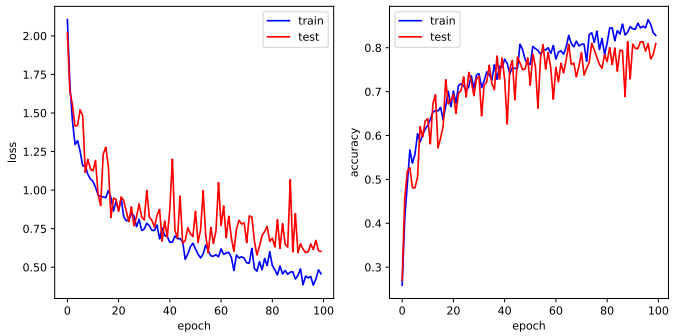


Figure 3: Training and testing curves for ResNet18

2.2.1 Task 1

In this part, start code "main.py" is run directly. The training and testing curves are depicted in Fig. 3. The best test accuracy is 81.39%.

2.2.2 Task 2

In this part, DenseNet is chosen to design model B. We illustrate the structure of densenet in a bottom-up way.

1. **Dense Layer**: The basic unit in densenet is dense layer. As shown in Fig. 4, each dense layer produce k feature-maps, no matter how many input feature maps. There is a bottleneck before the 3×3 convolution layer, which is used to reduce the number of input features (since $k_0 \gg k$). Thus, it can reduce the parameter amount in 3×3 convolution layer and improve computation efficiency.
2. **Dense Block**: Compared with ResNet, Dense Block connects the output of each layer to every other layer in a feed-forward way. A 5-layer dense block is shown in Fig. 5. If the input feature maps of a dense block are k_0, then the output feature maps are $\lfloor \theta(k_0 + kL) \rfloor$, where θ is the compression rate, k is the growth rate and L is the layer number. It significantly strengthens the information propagation and alleviates gradient-vanishing problem [2] since it can reuse the input and output feature maps of any layer.
3. **DenseNet**: As illustrates in Fig. 6, the whole dense net is cascaded by several dense blocks. Notice that there is a convolution layer and a pooling layer linking between different blocks.

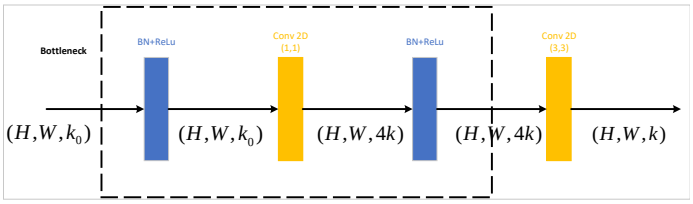


Figure 4: Illustraion of a dense layer

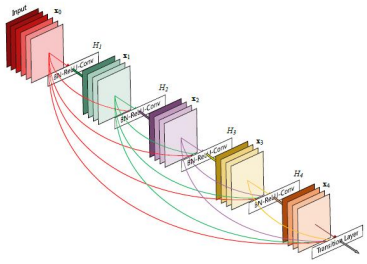


Figure 5: 5-layer dense block

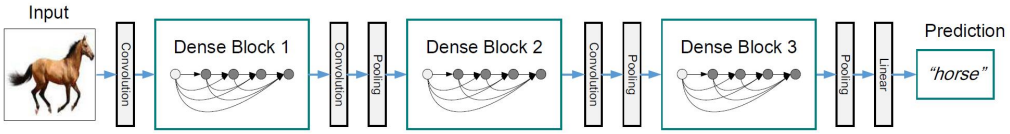


Figure 6: 3-blocks dense net

Here, structure of model B is implemented based on the description in [2]. Some structure hyper-parameters are fixed, specifically,

1. the output feature-maps of first convolution layer (i.e., initial feature) is set as twice of growth rate k.
2. BN size is set as 4 in each dense block.
3. the number of dense block is set as 4, which is determined by the input size of image.
4. The layer number in each dense block is kept as L.

Thus, we have 3 adjustable structure hyper-parameters in model B, namely, layer number L, growrh rate k and compression rate θ. We vary the the 3 hyper-parameters to find the combination with highest test accuracy, as shown from id=2 to 10 in Table .3.

Compare experiment cases id=(3,4) to id=1. It can be found that with the same "depth" (4 blocks), Model B can improve the test accuracy by 0.88% and 2.61%, respectively. Additionally, the final feature-maps of the experiment case id=3 is only 432, which is smaller than 512 in Model A. Thus, it validates the feature propagation ability of densenet.

Meanwhile, some observations about the structure hyper-parameters of Model B can be obtained:

1. When fixing other parameters, better test accuracy can be achieved by increasing k. It is due to the fact that the features dimension before the last classifier is proportional to k. Thus, different classes can be classified easier in such high dimensional space.
2. Compare experiment combinations (2,3,4) and (5,6,7), it can be found that the former combination achieves higher train and test accuracy except for the case $k = 12$. Comparing

the gap between training and test curves for id=(4,7) in Fig. 7 and 8, it can be observed that the generalization capability of case id=7 is weaker than case id=4. Meanwhile, it seems that the network is harder to optimize or converge with larger L, since the training accuracy are even lower.

3. Compare experiment combinations (2,3,4) and (5,6,7), it can be found that higher train accuracy and test accuracy can be achieved. Therefore, with deeper depth L, we can lower the compression rate θ for better optimization and generalization.

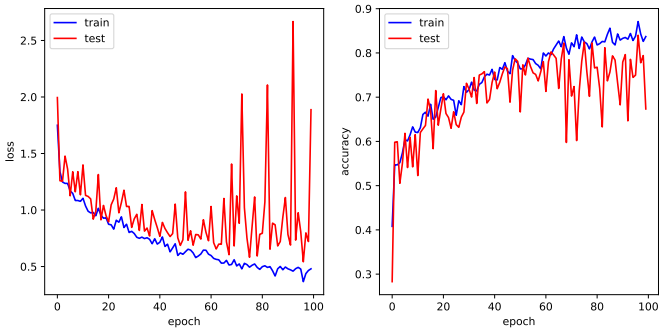


Figure 7: Training and testing curves for Model B $(L, k, \theta) = (4, 36, 1)$

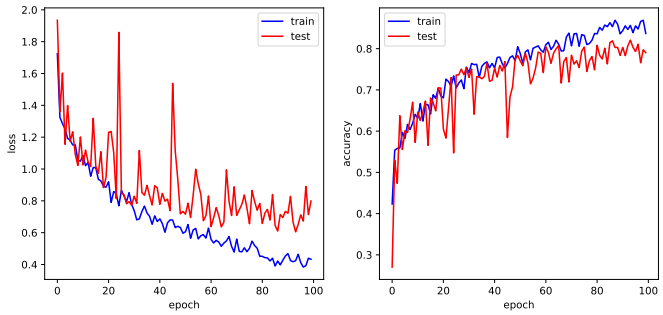
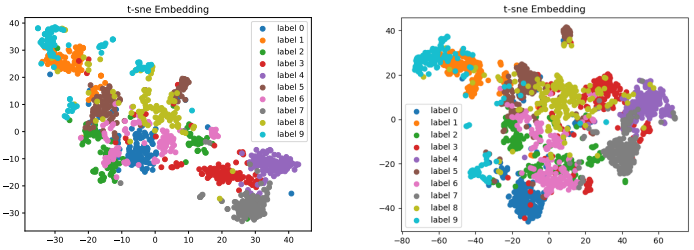


Figure 8: Training and testing curves for Model B $(L, k, \theta) = (6, 36, 1)$

2.2.3 Task 3

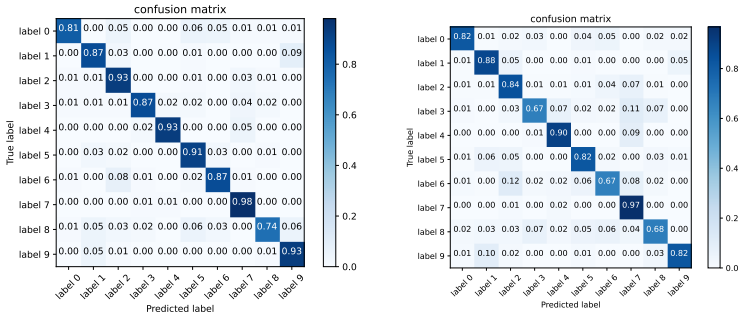
In this part, t-sne embedding are utilized to visualize the the 512-dim features before classifier in Model A. As depicted in Fig. 9, the trained best model A is used to predict training and test data, with 100 and 200 samples in each class, respectively. In Fig. 9, some classes have good embedding performance. These classes (e.g. lable 7) have relatively high accuracy and their samples are gathered in a small region. However, some classes have worse embedding performance. Their samples are spread in the space and are mixed with others (e.g. label 8). Two possibilities for the phenomenon can be summarized:

1. The in-class variances of certain classes are relatively high. Thus, Model A cannot extract good features in these classes even after 100 training epochs. Thus, these classes usually have relatively low accuracy in training dataset, for instance 74% of label 8, as shown in Fig. 10(a).
2. Since training samples is only 20% of test samples, the distribution difference between training and test dataset is relatively high for some classes. As a result, the trained Model A cannot have a have a good generalization performance for certain classes. Therefore, these classes usually have relatively high accuracy degradation in test dataset, for instance 20% of label 3 and 6, as shown in Fig. 10.



(a) train (b) test

Figure 9: t-sne embedding of Model A



(a) train (b) test

Figure 10: confusion matrix of Model A

2.2.4 Task 4

In this part, we fix the structure hyper-parameter of Model B as $(L, k, \theta) = (4, 36, 1)$. Since the network seems not to converge well after 100 epochs, the epochs are extended to 150 here. After several trails of data augmentation (e.g. rotation, Gaussian noise, hist equalization, cutmix [3]) and learning rate strategy (e.g. cosine annealing [4]), I choose **cyclic shift** augmentation and step strategy to improve the performance. The reasons are summarized as follows:

1. **data augmentation**: On the one hand, the original data samples are low-resolution satellite images, Meanwhile, the samples are very sensitive to the change in color. Thus, these pixel based transformation augmentation methods have negative impacts on both training and test dataset. On the other hand, since different classes are not so distinct, mix based augmentation cannot improve accuracy so much. For satellite images, we can use rotation for data augmentation. Since all theses images are taken from the satellite, image rotation is equivalent to take these images at different angles. However, the image rotation will make blank areas near the 4 corners of the image, which leads to information loss and confusion.

   For this task, cyclic shift is proposed for the low resolution satellite image augmentation, as depicted in Fig. 11. The idea is motivated by [5]. The authors augment Dynamic Range Doppler Images (DRAI) in range and Doppler axis to generate the same gesture with different velocity and range, which overcome the lack of wireless dataset effectively.

   Theoretically, cyclic shift only changes the phase shift of image in frequency domain after the same convolution layer. In other words, cyclic shift augmentation helps the network to learn the magnitude feature of an image in frequency domain. For our classification task, different classes are determined by the macroscopic feature of the image, which do not change after cyclic shift augmentation. For instance, the 4 shifted images in Fig. 11 are all classified as "Residential" in human sense. Meanwhile, unlike existing random affine in Pytorch, cyclic shift won't cause blank area, which can make use of all pixels effectively.

2. **learning strategy**: It can be found in Fig. 7 that the test curve is not smooth even after 60 epochs. Thus, the cosine annealing warm restart will make the test curve even more unsmooth and the optimization result cannot be always be guaranteed. Therefore, I choose the step strategy to decay learning rate 10 times after 120 epochs. Obviously, lower learning rate can smooth the training and test curves, which contributes to the convergence and robustness.

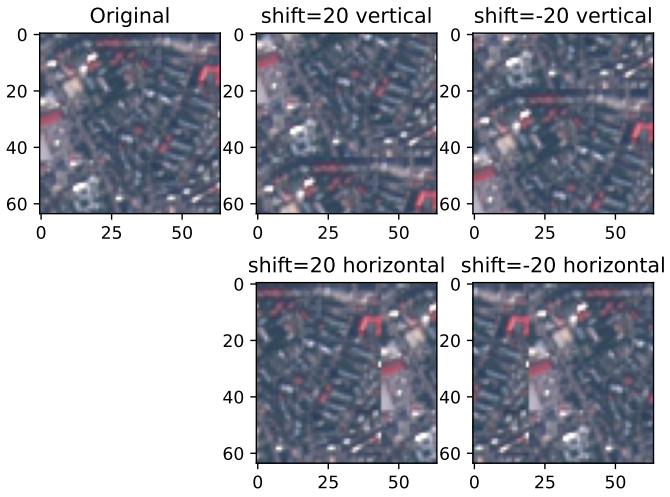
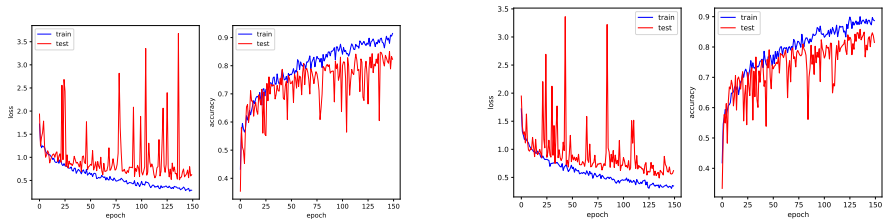
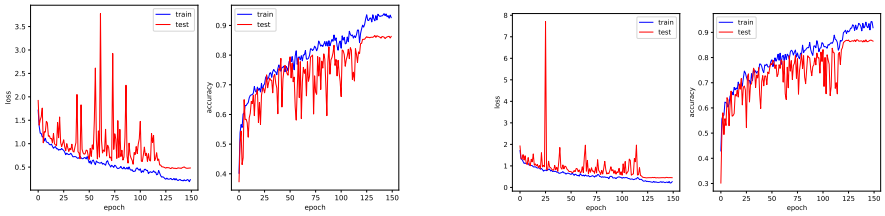


Figure 11: Illustration of cyclic shift



(a) original

(b) cyclic shift



(c) step

(d) cyclic shift+step

Figure 12: training and testing curves

The settings and accuracy of ablation experiments are shown in id=(11,12,13,14) of Table. 3. The train and test curves are plotted from Fig. 12(a) to 12(d). Based on the ablation experiment results, some conclusions can be obtained:

1. Compare experiments id=12 and 11, we can find that the training accuracy is decreased by 1.46% and the test accuracy is increased by 0.76%, which validates that cyclic shift augmentation can overcome the overfitting in the small-scale training dataset and have better generalization capability over test dataset.
2. Compare Fig. 12(c) and Fig. 12(a), we can find that the train and test curves become more smooth after learning rate decay at epoch=120. Meanwhile, step strategy helps the model converge to a better local minimum after 5 epochs. It increases the train and test accuracy by 2.52% and 1.54%, respectively. It validates the effectiveness of step learning rate strategy.
3. As shown in experiment id=14, when both data augmentation and scheduler are utilized, traing and test accuracy are increased by 2.81% and 2.02%, respectively. Meanwhile, comparing experiments id=(14, 12, 13), we can find that both data augmentation and learning strategy are necessary. We can make full use of the model when these two techniques are utilized.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[2] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[3] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[4] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[5] Yadong Li, Dongheng Zhang, Jinbo Chen, Jinwei Wan, Dong Zhang, Yang Hu, Qibin Sun, and Yan Chen. Towards domain-independent and real-time gesture recognition using mmwave signal. *IEEE Transactions on Mobile Computing*, 2022.