Homework 1

PENG Qiheng

Student ID 225040065

October 31, 2025

1. Environment and Reproducing code

   Environment:

Table 1: Environment

RTX 4090 24GB
CUDA Version: 12.4
python=3.11.11
jupyter==1.1.1
torch==2.6.0
torchvision==0.21.0
matplotlib==3.10.3
tqdm==4.67.1

   Reproducing code:

Table 2: Preferance of Example

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
7.28M	128	5s/epoch	0.227	87.17%

2. **Methods of Accuracy Improvement:**

(a) Residual mechanism:

Replace some convolutional layers with Residual Blocks.

Table 3: Preferance of Residual Mechanism

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
6.10M	128	5s/epoch	0.321	87.43%

We can see that the number of parameters is reduced, but the accuracy is slightly improved.

(b) Deeper & Wider Network:

Increase the number of residual blocks and the number of channels in each layer.

Table 4: Preferance of Deeper & Wider Network

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
11.03M	128	6s/epoch	0.051	91.09%

We can see that the accuracy is significantly improved.

(c) Optimizer Change:

Change the optimizer from Adam to SGD or AdamW. (AdamW performs better in my experiments.)

Table 5: Preferance of AdamW Optimizer

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
11.03M	128	6s/epoch	0.008	92.57%

We can see that the accuracy is imporved slightly, but the loss is

reduced significantly. So I think the model may be overfitting. I need to take some data augmentation methods to solve this.

(d) Data Augmentation:

Add ColorJitter to the data augmentation methods of baseline.

Table 6: Preferance of Data Augmentation

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
11.03M	128	6s/epoch	0.010	92.61%

We can see that the accuracy is remained almost the same, but the loss increased slightly. That means the model is less likely to overfit.

(e) Attention Mechanism:

Replace the residual blocks with attention blocks (framework like ViT)

Table 7: Preferance of Attention Mechanism

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
8.97M	128	10s/epoch	0.015	90.85%

We can see that the accuracy decreased. This may be because the attention mechanism is suitable for NLP tasks but not for image classification tasks. Another reason may be that the attention mechanism requires more data to train the model, since cifar10 is a small dataset, the model may not be fully trained.

(f) Structure Improvement

Base on the above experiments, I choose to use residual mecha-

nism, and try several experiments to modify the structure of my model to find the best structure. Finally, I choose a 17-layer residual network.

Table 8: Preferance of Final Structure

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
53.24M	128	10s/epoch	0.004	93.24%

3. **Classification on Tiny-imagenet Dataset**

   (a) Reuse the Model: Reuse the best model structure from cifar10 experiments, and modify the input size of the first layer to adapt to the 64×64 images of tiny-imagenet dataset.

Table 9: Preferance of Model in Cifar-10

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
46.87M	64	16s/epoch	0.1004	38.82%

   We can see that the accuracy is 38.82%, but the loss is extrimely low (0.1004). This may be because the model is overfitting. So we can try strongly data augmentation and regularization methods to solve this problem.

   (b) More Data Augmentation: Add strongly RandomAffine and ColorJi.

   And to accelarate the training speed, I modify the batch size to 1000 and number of workers to 8 in dataloader.

Table 10: Preferance of Model in Cifar-10

Param	**Epoch**	**Speed**	**Loss**	**Test Acc**
50.54M	128	25s/epoch	2.1935	46.40%

4. **Thoughts**

   (a) Residual mechanism can accelerate the reducing of training loss, and make the model easier to train.

   (b) A deeper and wider network can improve the performance of the model, but it may also lead to overfitting.

   (c) For solving overfitting, the best method in my experiments is data augmentation and adding dropout layers.

   (d) Changing the optimizer have little effect on the accuracy in my experiments.

   (e) Attention mechanism may not be suitable for image classification tasks, or it may require more data to train the model.

   (f) Attention layer has less parameters but slower training speed than residual layer.

   (g) Modify the batch size and number of workers in dataloader suitable can affect the training speed significantly. (It depends on the numbers of cpu cores and gpu memory size.)

   (h) Printing the training loss graph can help to find the suitable learning rate and epoch number.

5. **Difficulties**

   (a) The tiny-imagenet dataset has some **gray images**, which throw errors when putting them into dataloader. (Gray images have

only 1 channel, while others are 3 channels input, which cause dimension mismatch error in dataloader.)

– Solution: I write a function to convert gray images to RGB images before loading them into dataloader.

(b) The model training is too slow when training on tiny-imagenet dataset (64×64).

– Solution: I rewrite a new dataset class (myDataset.py), putting the transform operations in the new dataset class instead of huggingface's default dataset class, which can accelarate the data loading speed significantly.