一、MNIST

```
————————第13轮测试开始————————
训练次数为：0，损失值为：0.0008919906103983521
训练次数为：100，损失值为：0.002561557339504361
训练次数为：200，损失值为：0.0055953809060156345
训练次数为：300，损失值为：0.007039319258183241
训练次数为：400，损失值为：0.009181051515042782
训练次数为：500，损失值为：0.006317286752164364
训练次数为：600，损失值为：0.0009342824341729283
训练次数为：700，损失值为：0.028235530480742455
训练次数为：800，损失值为：0.09851711988449097
训练次数为：900，损失值为：0.0015504423063248396
测试准确率为：0.991500
```

2 个卷积+池化，后面搭了 3 个全链接。用 RELU 激活函数，每一个卷积+池化外面都套上。损失直接用的 CrossEntropyLoss 函数。训练直接用的 pytorch 的 SGD，调的 lr=0.14

首次训练，第 13 轮达到最高，99.15%

```
————————第15轮测试开始————————
训练次数为：0，损失值为：0.004489143844693899
训练次数为：100，损失值为：0.000854879675898701
训练次数为：200，损失值为：0.009291884489357471
训练次数为：300，损失值为：0.014872782863676548
训练次数为：400，损失值为：0.0001927046396303922
训练次数为：500，损失值为：0.023905713111162186
训练次数为：600，损失值为：0.011385949328541756
训练次数为：700，损失值为：0.003073827363550663
训练次数为：800，损失值为：0.01851477101445198
训练次数为：900，损失值为：0.004932002164423466
测试准确率为：0.970400
```

后面收敛了，到第 15 轮已经过拟合了，降到 97.04%。

```
————————第1轮测试开始————————
训练次数为：0，损失值为：0.005370953585952 52
训练次数为：100，损失值为：0.012754017487168312
训练次数为：200，损失值为：0.00020568983745761216
训练次数为：300，损失值为：0.00065847177756950 26
训练次数为：400，损失值为：0.031276579946279526
训练次数为：500，损失值为：0.016798172146081924
训练次数为：600，损失值为：0.00195101136341691 02
训练次数为：700，损失值为：0.00176029093563556 67
训练次数为：800，损失值为：0.012027489952743053
训练次数为：900，损失值为：0.046438619494438 17
测试准确率为：0.987400
————————第2轮测试开始————————
训练次数为：0，损失值为：0.0017312048003077507
训练次数为：100，损失值为：0.057120528072118 76
训练次数为：200，损失值为：0.043344825506210 33
训练次数为：300，损失值为：0.015320079401135445
训练次数为：400，损失值为：0.002144864760339 26
训练次数为：500，损失值为：0.023327950388193 13
训练次数为：600，损失值为：0.046073697507381 44
训练次数为：700，损失值为：0.008602881804108 62
训练次数为：800，损失值为：0.00032936391653493047
训练次数为：900，损失值为：0.08345226198434 83
测试准确率为：0.990300
```

后面又训了两轮，勉强到 99。

补充 1：我发现之前的训练每一次都没有保存 optimizer 的状态，后面保存了 optimizer 然后重新训练时加载 optimizer，发现效果好了很多，损失值小了特别多。

```
训练次数为：700，损失值为：0.0007277583354152739
训练次数为：800，损失值为：0.0045978110283613205
训练次数为：900，损失值为：0.0020433371554464102
测试准确率为：0.990000
```

补充 2：我在做 Cifar10 的时候发现好像 lr0.14 有点太高了，别人做的都很低的，看来 MNIST 确实太简单了，这么高的 lr 都能收敛到 99

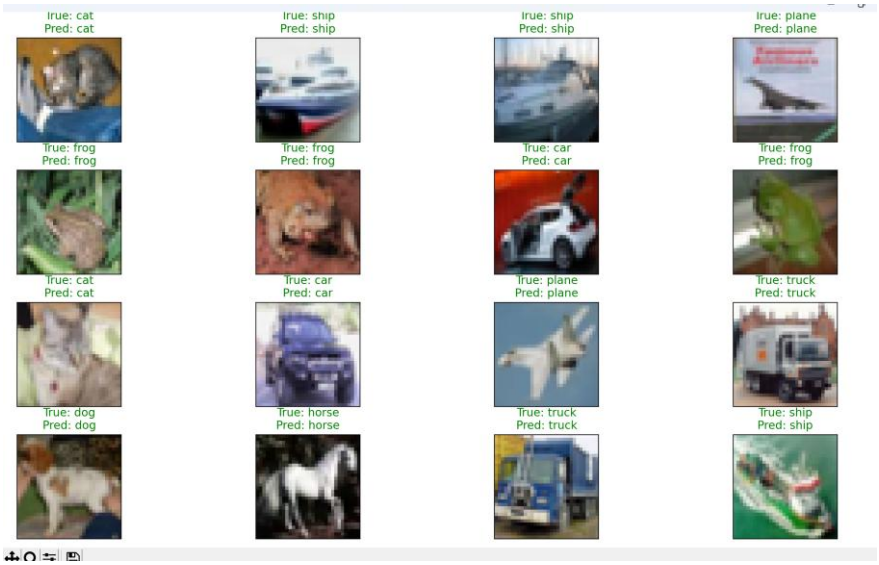
二、Cifar10

经过了漫长的一段时间，我重新捡起来了，之前训到了 80%，然后一直上不去了，放寒假干脆从头开始。寒假回家我有 4060 来训练模型。

先看 acc，94.70%。

```
正在使用设备: cuda
Files already downloaded and verified
Files already downloaded and verified
Files already downloaded and verified
c:\Users\MSI\Desktop\cifar10\test2.py:91: FutureWarning: You are using `torch
t malicious pickle data which will execute arbitrary code during unpickling (
or `weights_only` will be flipped to `True`. This limits the functions that c
owlisted by the user via `torch.serialization.add_safe_globals`. We recommend
Hub for any issues related to this experimental feature.
  model.load_state_dict(torch.load(checkpoint_path, map_location=device))
✅ 成功加载模型权重！
正在计算测试集整体准确率...
🏆 测试集最终准确率 (Accuracy): 94.70%

--- 正在抽取 16 张测试集图片进行“抽查” ---
```



在训练图片中进行随机裁剪、随机反转、随机生成 3*3Cutout 等预处理操作。训练集中 50000 张图片抽 10000 张作验证集（Validation），每一轮用 Validation 算损失值与 acc。

初始训练 Epoch = 68，后面续训了若干次（具体几次记不住了），每一次训 5 轮。Batch_size = 128。

Qptimizer 类型 SGD, 初始 lr = 0.05, 采用动态调节机制, 如果 Loss 连续 2 轮不下降则 lr*0.5。采用保存最优策略, 只有 valid_loss 低于 valid_loss_min, 才会保存模型, 这种操作续训会出问题, 因为续训会使 lr 与 valid_loss_min 全部重置，很可能导致续训后模型表现下降，这归根结底是 lr 偏高造成的。

模型架构如下,

预处理层：Conv（3,32,32）—>（64,32,32）

Layer1：Conv1+bn+relu + Conv2+bn（64,32,32）—>（64,32,32）

Layer2：Conv1+bn+relu + Conv2+bn（64,32,32）—>（128,16,16）

Layer3：Conv1+bn+relu + Conv2+bn（128,16,16）—>（256,8,8）

Layer4：Conv1+bn+relu + Conv2+bn（256,8,8）—>（512,4,4）

池化层：AvgPool（512,4,4）—>（512,1,1）

全链接层：Linear　512—>10