

Lab 8 - 自动机器学习系统练习 - 实验结果

目录说明：

实验结果共 2 个部分，包括手动调参和 NNI 自动调参。

(1) image 目录：实验结果截图。NNI 实验中更详细的实验结果请看**实验报告**或者 resources 文件夹里的运行输出文件。

(2) src 目录：存放代码，共 4 套（手动调参 1 套，NNI 自动调参 3 套）

手动调参 代码每次运行除参数外无其他变化。

NNI 自动调参 则进行了三次（具体不同和其他信息请看实验报告），搜索空间文件和参数设置都有不同。

(3) resources 目录：存放每次实验结束后输出的结果文件（包括日志等），**实验报告**中有完整的总结。

下面是实验报告

实验报告

Lab 8 - 自动机器学习系统练习

一，实验环境

本机：

硬件环境

CPU (vCPU 数目)	:	AMD Ryzen 7 4800H
GPU(型号, 数目)	:	NVIDIA GeForce RTX 2060

软件环境

OS 版本	:	Ubuntu 16.04 LTS
深度学习框架	:	PyTorch 1.5.0
python 包名称及版本	:	Python 3.7.6
CUDA 版本	:	CUDA 10.1, CuDNN 7.6

Bitahub 环境：

硬件环境

CPU (vCPU 数目) : 2 vCPUs
GPU(型号, 数目) : GTX 1080ti *2 or TITAN Xp *2

默认软件环境

OS 版本 : Ubuntu 16.04 LTS
深度学习框架 : PyTorch 1.3
python 包名称及版本 : Python 3.6
CUDA 版本 :
CUDA_PKG_VERSION=10-0=10.0.130-1
CUDA_VERSION=10.0.130
CUDNN_VERSION=7.4.2.24

使用的 nni 镜像软件环境

NNIv1.8 form: [msranni/nni\(docker.com\)](https://msranni/nni/docker.com)
OS 版本 : Ubuntu 16.04 LTS
深度学习框架 : PyTorch 1.5.0
python 包名称及版本 : Python 3.7.6
CUDA 版本 : CUDA 10.1, CuDNN 7.6

二，实验结果

1.记录不同调参方式下，cifar10 程序训练结果的准确率。

原始代码

[src\手动调参代码\hpo\main.py](#)

[src\手动调参代码\hpo\utils.py](#)

(实验提供的原始代码)

手动调参任务：

说明：

由于缺少经验，不了解各个超参数之间的组合关系（比如某种超参数组合可能比单纯的各参数最优的组的表现更优），因此初步想法是采用**控制变量法**，**每次只改变一个超参数**，找到每个超参数的使得单独表现最优的值，最后再将这些单独最优值组合起来，试图找到最好的超参数组合。

默认参数 Default:	原始准确率	
--initial_lr=0.1, --weight_decay=5e-4 --ending_lr=0 --cutout'=0,	0.8488	

--batch_size =128 --epochs =300 --optimizer =sgd --momentum =0.9 --num_workers =2 --model ='resnet18', --grad_clip =0. --log_frequency =20 --seed=42		
只改变一个参数：（其他参数依旧默认）		
改变的参数名称：	最终准确率	
Grad clip:		
grad_clip 5.0	0.8478	
grad_clip 3.0	0.8496	
grad_clip 1.0	0.8556	Best
grad_clip 0.0	0.8488	
grad_clip 2.0	0.85	
Weight decay:		
weight_decay 5e-5	0.853	Best
weight_decay 0.001	0.817	
weight_decay 5e-4	0.8488	
weight_decay 1e-5	0.8488	
weight_decay 1e-6	0.8427	
Learning rate:		
lr 0.0001	0.7381	
lr 0.001	0.7935	
lr 0.01	0.8484	
lr 0.1	0.8488	Best
model:		
densenet121	0.8696	Best
mnasnet1_0	0.5339	
resnext50_32x4d	0.866	
mobilenet_v2	0.7821	
shufflenet_v2_x1	0.7372	
squeezenet1_1	0.1	
vgg16_bn	0.1	
resnet50	0.8567	
vgg16	0.1	

resnet18	0.8488	
<i>optimizer:</i>		
rmsprop	0.1255	
adam	0.4385	
sgd	0.8488	Best
<i>Batch size:</i>		
batch_size 64	0.6843	
batch_size 128	0.8488	Best
batch_size 256	0.7936	
<i>Epochs:</i>		
epochs 600	0.8625	Best
epochs 300	0.8488	
epochs 100	0.8266	
<i>Cutout:</i>		
cutout 16	0.8386	
cutout 8	0.8441	
cutout 0	0.8488	
cutout 4	0.8503	Best
cutout 12	0.8411	
将上述的最优参数组合起来有		
1: 第一组组合:		
--model resnet50 --initial_lr 0.01 --grad_clip 1.0 --optimizer sgd --weight_decay 5e-5 --cutout 4 --epochs 600 --batch_size 128	0.7887	
2: 第二组组合:		
--model densenet121 --initial_lr 0.1 --grad_clip 1.0 --optimizer sgd --weight_decay 5e-5 --cutout 4 --epochs 600 --batch_size 128	0.876600	
3: 第三组组合:		

手动调参所得最好参数组合：		
<pre>--model densenet121 --initial_lr 0.1 --grad_clip 1.0 --optimizer sgd --weight_decay 5e-5 --cutout 0 --epochs 600 --batch_size 128</pre>	0.8818	Best

NNI 自动调参

说明：

由于个人机器性能限制，本次 NNI 自动调参部分进行了三次，一次本地，两次远程。

第一次是在自己的机器上运行（由于性能限制，调小了 epochs 以完成足够多的 trials）
有 WebUI 结果，有输出的结果文件，共 10 个 trials，运行 **3h 18m 44s**，使用一张 **2060**

第二次是在 Bitahub 上使用 nni 镜像正常进行调参实验（感谢助教的帮助）
所有参数正常设置，有输出的结果文件（包含所有结果），但由于在远程进行，无法获取 WebUI（尝试解决，但没有成功），共 20 个 trials，运行 **20h 7m 15s**，使用 **Titanxp*2**
完成证明：

#1	+	暂停	2021-05-18 16:41:23	20h 7m 14s	查看 TensorBoard
----	---	----	---------------------	------------	--

第三次是在上两次实验的基础上，修改了搜索空间，去除了一些不会是最优解的参数选择，
以期待得到更精确的最优参数搭配，有输出的结果文件（包含所有结果），但由于在远程进行，无法获取 WebUI，共 20 个 trials，运行 **23h 34m 59s**，使用 **GTX1080ti*4**
完成证明：

#2	+	暂停	2021-05-22 23:27:25	23h 34m 59s	查看 TensorBoard
----	---	----	---------------------	-------------	--

第一次：本地 来源：第一次结果 FcMxgqBE\db\ nni.sqlite

序号	Trials	参数	最终准确度
0	twfDT	<pre>parameter_id: 0, parameters: initial_lr: 0.09034715974869717, weight_decay: 1.1604509481016609e-05, cutout: 16, batch_size: 64, optimizer: sgd, grad_clip: 2.0, momentum: 0.654505661200692, model: resnet18,</pre>	0.8243

1	Svi0h	parameter_id: 1, parameters: initial_lr: 0.018686472460319094, weight_decay: 0.0004940755899789372, cutout: 8, batch_size: 128, optimizer: adam, grad_clip: 5.0, momentum: 0.9146790576258961, model: shufflenet_v2_x1_0,	0.6031
2	D5reV	parameter_id: 2, parameters: initial_lr: 0.017023419589240807, weight_decay: 0.00015143759297568958, cutout: 0, batch_size: 64, optimizer: sgd, grad_clip: 0.0, momentum: 0.6674085677966901, model: densenet121,	0.7804
3	Vefwc	parameter_id: 3, parameters: initial_lr: 0.005297584834581654, weight_decay: 8.527475106759368e-05, cutout: 12, batch_size: 64, optimizer: rmsprop, grad_clip: 4.0, momentum: 0.6097509860471888, model: mobilenet_v2,	0.1775
4	tlAYX	parameter_id: 4, parameters: initial_lr: 0.006649932990738287, weight_decay: 1.693810225083496e-05, cutout: 4, batch_size: 256, optimizer: sgd, grad_clip: 5.0, momentum: 0.8915282530863404, model: shufflenet_v2_x1_0,	0.6837
5	aSeoi	parameter_id: 5, parameters: initial_lr: 0.029785190727620536, weight_decay: 3.7355986605476486e-05, cutout: 4, batch_size: 64, optimizer: rmsprop, grad_clip: 5.0, momentum: 0.508321042970751, model: resnet18,	0.1365
6	U30QA	parameter_id: 6, parameters: initial_lr: 0.0005572218738943095, weight_decay: 7.285670352486437e-06, cutout: 16, batch_size: 256, optimizer: adam, grad_clip: 5.0, momentum: 0.9011437564741125, model: resnet18,	0.7985
7	TGiat	parameter_id: 7, parameters: initial_lr: 0.006602584100839187, weight_decay: 3.799227096016227e-05, cutout: 0, batch_size: 64, optimizer: sgd, grad_clip: 0.0, momentum: 0.573377101743183, model: shufflenet_v2_x1_0,	0.6812
8	Z48ET	parameter_id: 8, parameters: initial_lr: 0.0064165071324468375, weight_decay: 8.343136306290842e-06, cutout: 4, batch_size: 128, optimizer: sgd, grad_clip: 2.0, momentum: 0.8488139804651124, model: shufflenet_v2_x1_0,	0.6239
9	m0eKl	parameter_id: 9, parameters: initial_lr: 0.00023785663001286191, weight_decay: 8.674155263395103e-05, cutout: 0, batch_size: 128, optimizer: sgd, grad_clip: 5.0, momentum: 0.5207161382597885, model: resnet50,	0.1129

第二次：Bitahub nniv1.8 来源：第二次结果 cXbLe89m \db\ nni.sqlite

序号	Trials	参数	最终准确度
----	--------	----	-------

0	CpCtD	parameter_id: 0, parameters: initial_lr: 0.0001002983907142384, weight_decay: 5.560145611250914e-06, cutout: 16, batch_size: 128, optimizer: sgd, grad_clip: 5.0, epochs: 600, momentum: 0.718875687080359, model: resnet50,	0.1922
1	Y7126	parameter_id: 1, parameters: initial_lr: 0.08618160046401947, weight_decay: 1.2588098704003962e-06, cutout: 4, batch_size: 128, optimizer: adam, grad_clip: 2.0, epochs: 100, momentum: 0.821033045817649, model: mobilenet_v2,	0.5422
2	E9uKS	parameter_id: 2, parameters: initial_lr: 0.0006171905523361188, weight_decay: 2.2888638974644543e-05, cutout: 12, batch_size: 64, optimizer: rmsprop, grad_clip: 0.0, epochs: 100, momentum: 0.6052453891969551, model: resnext50_32x4d,	0.838
3	BwMHi	parameter_id: 3, parameters: initial_lr: 0.0010987861378000863, weight_decay: 0.0001099794471295465, cutout: 0, batch_size: 128, optimizer: rmsprop, grad_clip: 3.0, epochs: 100, momentum: 0.9100868110147879, model: mobilenet_v2,	0.7682
4	rfase	parameter_id: 4, parameters: initial_lr: 0.00020755517989244334, weight_decay: 1.8107798219648837e-06, cutout: 0, batch_size: 128, optimizer: adam, grad_clip: 5.0, epochs: 600, momentum: 0.7161170266479149, model: resnet50,	0.8515
5	BoIOs	parameter_id: 5, parameters: initial_lr: 0.085837861930452, weight_decay: 9.849469605233018e-06, cutout: 8, batch_size: 256, optimizer: rmsprop, grad_clip: 3.0, epochs: 100, momentum: 0.5041133150671907, model: densenet121,	0.1
6	FazmJ	parameter_id: 6, parameters: initial_lr: 0.09856953539170227, weight_decay: 8.566945736212825e-05, cutout: 12, batch_size: 64, optimizer: rmsprop, grad_clip: 1.0, epochs: 300, momentum: 0.7478225372389389, model: mobilenet_v2,	0.1
7	jBBAf	parameter_id: 7, parameters: initial_lr: 0.033838026887613205, weight_decay: 5.20258071532106e-06, cutout: 8, batch_size: 256, optimizer: rmsprop, grad_clip: 2.0, epochs: 100, momentum: 0.6090944068875751, model: resnet50,	0.1
8	TYZP1	parameter_id: 8, parameters: initial_lr: 0.0008901599621915286, weight_decay: 0.00032731302681685925, cutout: 8, batch_size: 128, optimizer: sgd, grad_clip: 0.0, epochs: 100, momentum: 0.7779850101918306, model: mobilenet_v2,	0.6313
9	JsrgP	parameter_id: 9, parameters: initial_lr: 0.00812167238974434, weight_decay: 5.500163400743114e-05, cutout: 4, batch_size: 256, optimizer: rmsprop, grad_clip: 0.0, epochs: 300, momentum: 0.9869407753379864, model: densenet121,	0.1431
10	L6Qqe	parameter_id: 10, parameters: initial_lr: 0.002633594836534122, weight_decay: 1.1727350970881157e-06, cutout: 12, batch_size: 256, optimizer: rmsprop, grad_clip: 2.0, epochs: 100, momentum: 0.6273090653385966, model: shufflenet_v2_x1_0,	0.7546

11	PHXi6	parameter_id: 11, parameters: initial_lr: 0.004503450239871497, weight_decay: 3.7341597002271735e-06, cutout: 8, batch_size: 64, optimizer: sgd, grad_clip: 0.0, epochs: 600, momentum: 0.6502610193794631, model: resnet18,	0.817
12	CkJVa	parameter_id: 12, parameters: initial_lr: 0.06252930245918477, weight_decay: 1.6751187259974524e-05, cutout: 8, batch_size: 128, optimizer: rmsprop, grad_clip: 4.0, epochs: 100, momentum: 0.7436164616360401, model: mobilenet_v2,	0.1
13	p9Ftn	parameter_id: 13, parameters: initial_lr: 0.00019117375743175972, weight_decay: 7.0799498921561614e-06, cutout: 0, batch_size: 256, optimizer: adam, grad_clip: 3.0, epochs: 100, momentum: 0.5556479079908643, model: mobilenet_v2,	0.6876
14	hByem	parameter_id: 14, parameters: initial_lr: 0.00014856192255898555, weight_decay: 0.0003136369788285683, cutout: 0, batch_size: 64, optimizer: rmsprop, grad_clip: 4.0, epochs: 100, momentum: 0.601554696818033, model: resnet50,	0.8227
15	LMgpS	parameter_id: 15, parameters: initial_lr: 0.0008554928829711074, weight_decay: 0.0002607488013760008, cutout: 4, batch_size: 64, optimizer: sgd, grad_clip: 1.0, epochs: 300, momentum: 0.5205258267981119, model: mobilenet_v2,	0.3963
16	NumrX	parameter_id: 16, parameters: initial_lr: 0.0452915927889782, weight_decay: 1.2442873698417629e-05, cutout: 8, batch_size: 64, optimizer: rmsprop, grad_clip: 5.0, epochs: 600, momentum: 0.5839976264110929, model: resnet18,	0.2376
17	z4pwh	parameter_id: 17, parameters: initial_lr: 0.002234914405267082, weight_decay: 0.00013551618702152096, cutout: 0, batch_size: 256, optimizer: adam, grad_clip: 1.0, epochs: 100, momentum: 0.5586143041156879, model: densenet121,	0.8289
18	Z6TkC	parameter_id: 18, parameters: initial_lr: 0.008347621928389719, weight_decay: 0.00016355554100964592, cutout: 12, batch_size: 128, optimizer: rmsprop, grad_clip: 0.0, epochs: 100, momentum: 0.6425540926156985, model: resnext50_32x4d,	0.4554
19	Ekzgs	parameter_id: 19, parameters: initial_lr: 0.003312271871547241, weight_decay: 4.857584617984858e-05, cutout: 12, batch_size: 128, optimizer: rmsprop, grad_clip: 3.0, epochs: 100, momentum: 0.8797858336026911, model: densenet121,	0.7283

第三次: Bitahub nniv1.8 来源: 第三次结果: xq4PrepE \db\ nni.sqlite

序号	Trials	参数	最终准确度
----	--------	----	-------

0	BTTdV	parameter_id: 0, parameters: initial_lr: 0.0043874179464854016, weight_decay: 2.316421572609614e-06, cutout: 4, grad_clip: 1.0, epochs: 600, momentum: 0.7757109839847657, model: resnext50_32x4d	0.707
1	ekmPc	parameter_id: 1, parameters: initial_lr: 0.0017240759362312748, weight_decay: 3.2999895177310656e-05, cutout: 8, grad_clip: 1.0, epochs: 600, momentum: 0.9688827210843811, model: resnet50	0.7676
2	nW8Io	parameter_id: 2, parameters: initial_lr: 0.024407391227057477, weight_decay: 4.195046185445729e-05, cutout: 0, grad_clip: 2.0, epochs: 600, momentum: 0.8733833227714977, model: resnet50	FAIL
3	KVdGM	parameter_id: 3, parameters: initial_lr: 0.0011580264913597065, weight_decay: 1.4462238056601881e-05, cutout: 4, grad_clip: 2.0, epochs: 600, momentum: 0.8911391674092791, model: densenet121	0.7577
4	lreRN	parameter_id: 4, parameters: initial_lr: 0.002145541294464762, weight_decay: 4.769626499485306e-06, cutout: 0, grad_clip: 1.0, epochs: 600, momentum: 0.7229055510485315, model: densenet121	0.7163
5	qvju6	parameter_id: 5, parameters: initial_lr: 0.04081070811838679, weight_decay: 3.9688789643106386e-05, cutout: 4, grad_clip: 1.0, epochs: 300, momentum: 0.9421096185491687, model: resnet50	0.8758
6	F8R7t	parameter_id: 6, parameters: initial_lr: 0.0017204645077830132, weight_decay: 2.2004538492340218e-06, cutout: 0, grad_clip: 3.0, epochs: 600, momentum: 0.9988992517536266, model: resnet50	0.8501
7	PASdq	parameter_id: 7, parameters: initial_lr: 0.0014634017418157688, weight_decay: 4.108382319503372e-06, cutout: 0, grad_clip: 3.0, epochs: 300, momentum: 0.7056559818130397, model: resnext50_32x4d	0.4633
8	genEZ	parameter_id: 8, parameters: initial_lr: 0.03968074084652299, weight_decay: 2.0690510376426946e-05, cutout: 0, grad_clip: 2.0, epochs: 600, momentum: 0.9993126137936816, model: densenet121	0.2758
9	pXKyQ	parameter_id: 9, parameters: initial_lr: 0.02023408227429747, weight_decay: 1.9627493317376494e-06, cutout: 8, grad_clip: 3.0, epochs: 300, momentum: 0.7804427326776218, model: resnet50	0.8342
10	KPZGU	parameter_id: 10, parameters: initial_lr: 0.06025095002806948, weight_decay: 1.7025336910552377e-06, cutout: 0, grad_clip: 2.0, epochs: 300, momentum: 0.7382930760985371, model: densenet121	0.8481
11	hSx8U	parameter_id: 11, parameters: initial_lr: 0.005288783764022968, weight_decay: 1.9638230436028526e-05, cutout: 8, grad_clip: 2.0, epochs: 300, momentum: 0.7768932866375755, model: densenet121	0.7786
12	spCyy	parameter_id: 12, parameters: initial_lr: 0.02214824885549789, weight_decay: 2.686127516726607e-05, cutout: 0, grad_clip: 3.0, epochs: 600, momentum: 0.9995998821512169, model: resnet50	0.0987

1 3	EkoU3	parameter_id: 13, parameters: initial_lr: 0.008922116245685043, weight_decay: 8.54323610374235e-06, cutout: 0, grad_clip: 3.0, epochs: 300, momentum: 0.9135934360743698, model: resnet50	0.8418
1 4	OESAw	parameter_id: 14, parameters: initial_lr: 0.0070174467834221935, weight_decay: 2.058839793315472e-05, cutout: 4, grad_clip: 2.0, epochs: 600, momentum: 0.8731062335626152, model: densenet121	0.8018
1 5	oR6WS	parameter_id: 15, parameters: initial_lr: 0.043961361281396376, weight_decay: 3.139148632743806e-05, cutout: 4, grad_clip: 1.0, epochs: 300, momentum: 0.9024145405492363, model: resnet50	0.854
1 6	h3JFn	parameter_id: 16, parameters: initial_lr: 0.006727697366343249, weight_decay: 4.60721154787219e-05, cutout: 4, grad_clip: 3.0, epochs: 300, momentum: 0.8634664195228346, model: resnet50	0.7856
1 7	fwmAs	parameter_id: 17, parameters: initial_lr: 0.0063585598741283085, weight_decay: 1.00782642666135e-05, cutout: 0, grad_clip: 1.0, epochs: 600, momentum: 0.9175541941266154, model: resnet50	0.7752
1 8	HK4Zb	parameter_id: 18, parameters: initial_lr: 0.001344077276133078, weight_decay: 5.477064253721725e-06, cutout: 0, grad_clip: 2.0, epochs: 300, momentum: 0.9752560464967974, model: resnet50	0.7619
1 9	y2YxC	parameter_id: 19, parameters: initial_lr: 0.0035349877596874304, weight_decay: 1.9801277130266243e-06, cutout: 4, grad_clip: 3.0, epochs: 300, momentum: 0.8025949680248826, model: densenet121	0.7671

NNI 最终所得最好参数组合：

parameters: initial_lr: 0.04081070811838679 weight_decay: 3.9688789643106386e-05 cutout: 4 grad_clip: 1.0 epochs: 300 momentum: 0.9421096185491687 model: resnet50	0.8758
---	--------

结果总结：

可以看到，本次实验里，NNI 的最终最优结果 (**0.8758**) 比人工调参 (**0.8818**) 的结果稍低，但相信如果继续实验，运行更多的 trials 一定能很快找到更好的解。而与此同时，为达到上述的效果，若使用手动调参则需进行大量的操作，包括不断启动实验进程，记录参数，思考下一组参数搭配等等。而

若使用 NNI 自动调参只需修改两次搜索空间即可。NNI 在辅助人工调参的高效性在这里体现的非常充分。

2.提交使用 NNI 自动调参方式，对 main.py、search_space.json、config.yml 改动的代码文件或截图。

对 main.py 的修改：

(1) 说明： import nni

<pre>import numpy as np import nni import torch</pre>	29 30 31	<pre>import numpy as np import torch import torch.nn as nn</pre>
---	----------	--

(2) 说明： 每次训练完一个 epoch 向 nni 报告当前结果； 向 nni 报告最终结果

<pre>for epoch in range(1, args.epochs + 1): train(model, train_loader, criterion, optimizer, scheduler, args) top1, _ = test(model, test_loader, criterion, args, epoch, device) # report intermediate result nni.report_intermediate_result(top1) logger.debug('test accuracy %g', top1) logger.debug('Pipe send intermediate result done.') logger.info("Final accuracy is: %.6f", top1) nni.report_final_result(top1) logger.debug('Final result is %g', top1) logger.debug('Send final result done.')</pre>	146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161	<pre>for epoch in range(1, args.epochs + 1): train(model, train_loader, criterion, optimizer, scheduler, args) top1, _ = test(model, test_loader, criterion, args, epoch, device) logger.info("Final accuracy is: %.6f", top1) if __name__ == '__main__': available_models = ['resnet18', 'resnet50', 'vgg16', 'shufflenet_v2_x1_0', 'mobilenet_v2_x1_0'] parser = argparse.ArgumentParser(description='PyTorch ImageNet Training') parser.add_argument('--initial_lr', default=0.1, type=float) parser.add_argument('--weight_decay', default=5e-4, type=float) parser.add_argument('--ending_lr', default=0, type=float) parser.add_argument('--cutout', default=0, type=int) parser.add_argument('--batch_size', default=128, type=int)</pre>
---	---	---

(3) 说明： 每次训练完一个参数组合，准备下一个组合

<pre>args = parser.parse_args() nni.utils.merge_parameter(args, nni.get_next_parameter()) main(args)</pre>	179 180 181 182 183 184	<pre>args = parser.parse_args() main(args)</pre>
--	-------------------------	---

第一次实验： 本地
对 config.yml 的修改：

```

config.yml @ search_space.json
authorName: DaiRui
experimentName: cifar-10-nni
trialConcurrency: 1
maxExecDuration: 10h
maxTrialNum: 10
trainingServicePlatform: local
searchSpacePath: search_space.json
useAnnotation: false
logDir: .
tuner:
  builtinTunerName: TPE
  classArgs:
    #choice: maximize, minimize
    optimize_mode: maximize
trial:
  command: python main.py
  codeDir: .
  gpuNum: 1
localConfig:
  useActiveGpu: true
  maxTrialNumPerGpu: 1

```

对 search_space.json 的修改:

```

{
  "initial_lr": {
    "_type": "loguniform", "_value": [1e-4, 0.1]},
  "weight_decay": {
    "_type": "loguniform", "_value": [1e-6, 1e-3]},
  "dropout": {
    "_type": "choice", "_value": [0, 4, 8, 12, 16]},
  "batch_size": {
    "_type": "choice", "_value": [64, 128, 256]},
  "optimizer": {
    "_type": "choice", "_value": ["adam", "rmsprop", "sgd"]},
  "grad_clip": {
    "_type": "choice", "_value": [0.0, 1.0, 2.0, 3.0, 4.0, 5.0]},
  "momentum": {
    "_type": "uniform", "_value": [0.5, 1]},
  "model": {
    "_type": "choice", "_value": ["resnet18", "resnet50", "densenet121", "shufflenet_v2_x1_0", "mobilenet_v2", "resnext50_32x4d"]}
}

```

第二次实验：远程

对 config.yml 的修改:

```

config.yml
1  authorName: DaiRui
2  experimentName: cifar-10-nni
3  trialConcurrency: 2
4  maxExecDuration: 40h
5  maxTrialNum: 20
6  trainingServicePlatform: local
7  searchSpacePath: search_space.json
8  useAnnotation: false
9  logDir: /output/nni
10 tuner:
11   builtinTunerName: TPE
12   classArgs:
13     #choice: maximize, minimize
14     optimize_mode: maximize
15 trial:
16   command: python main.py
17   codeDir: .
18   gpuNum: 1
19 localConfig:
20   useActiveGpu: true
21   maxTrialNumPerGpu: 1

```

对 search_space.json 的修改:

```

{
  "initial_lr": {
    "_type": "loguniform",
    "_value": [1e-4, 0.1]
  },
  "weight_decay": {
    "_type": "loguniform",
    "_value": [1e-6, 1e-3]
  },
  "cutout": {
    "_type": "choice",
    "_value": [0, 4, 8, 12, 16]
  },
  "batch_size": {
    "_type": "choice",
    "_value": [64, 128, 256]
  },
  "optimizer": {
    "_type": "choice",
    "_value": ["adam", "rmsprop", "sgd"]
  },
  "grad_clip": {
    "_type": "choice",
    "_value": [0.0, 1.0, 2.0, 3.0, 4.0, 5.0]
  },
  "epochs": {
    "_type": "choice",
    "_value": [ 100, 300, 600]
  },
  "momentum": {
    "_type": "uniform",
    "_value": [0.5, 1]
  },
  "model": {
    "_type": "choice",
    "_value": ["resnet18", "resnet50", "densenet121", "shufflenet_v2_x1_0", "mobilenet_v2", "resnext50_32x4d"]
  }
}

```

第三次实验：远程 对 config.yml 的修改：

```

config.yml
1  authorName: DaiRui
2  experimentName: cifar-10-nni
3  trialConcurrency: 4
4  maxExecDuration: 20h
5  maxTrialNum: 20
6  trainingServicePlatform: local
7  searchSpacePath: search_space.json
8  useAnnotation: false
9  logDir: /output/nni
10 tuner:
11   builtinTunerName: TPE
12   classArgs:
13     #choice: maximize, minimize
14     optimize_mode: maximize
15 trial:
16   command: python main.py
17   codeDir: .
18   gpuNum: 1
19 localConfig:
20   useActiveGpu: true
21   maxTrialNumPerGpu: 1

```

对 search_space.json 的修改：

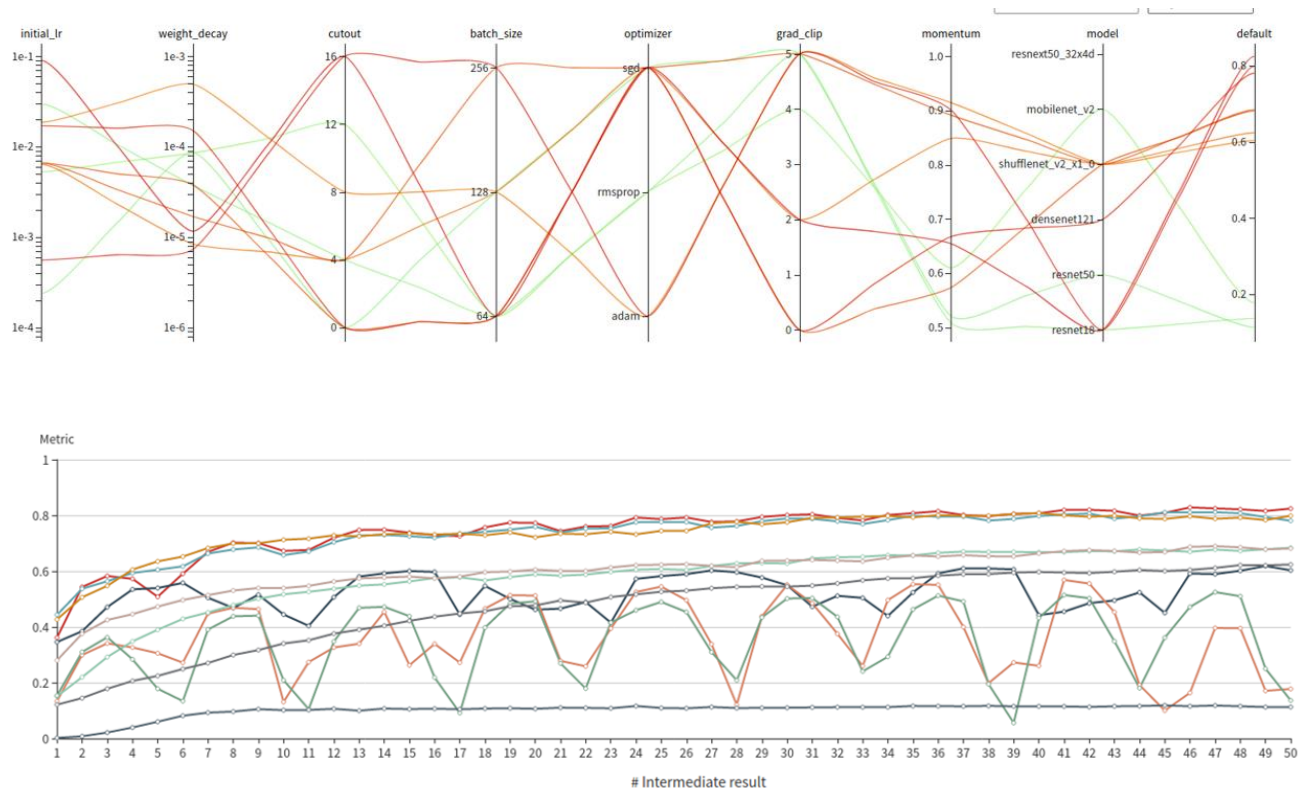
```

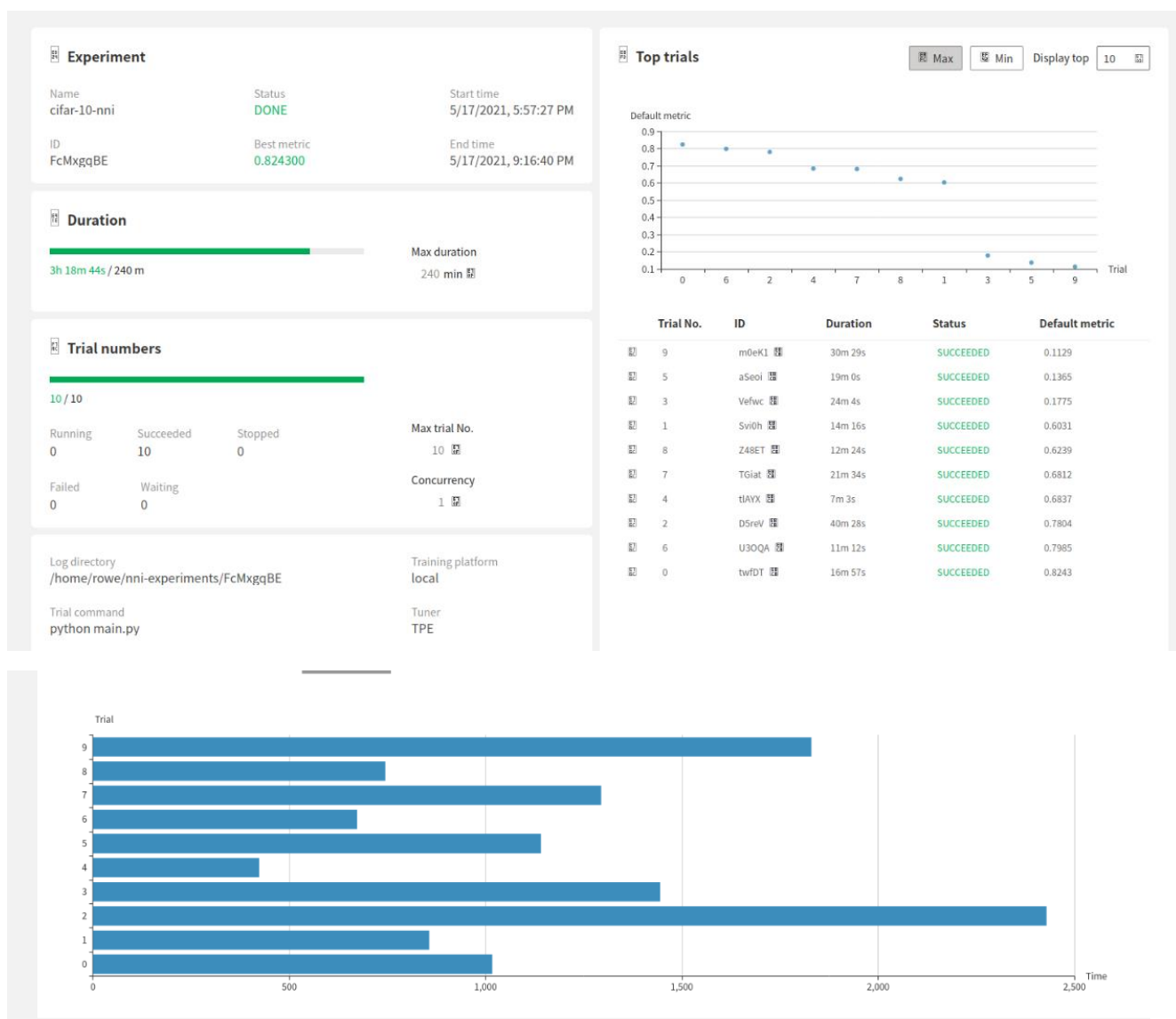
search_space.json
1  {
2    "initial_lr": {
3      "_type": "loguniform",
4      "_value": [1e-3, 0.1]
5    },
6    "weight_decay": {
7      "_type": "loguniform",
8      "_value": [1e-6, 1e-4]
9    },
10   "cutout": {
11     "_type": "choice",
12     "_value": [0, 4, 8]
13   },
14   "grad_clip": {
15     "_type": "choice",
16     "_value": [1.0, 2.0, 3.0]
17   },
18   "epochs": {
19     "_type": "choice",
20     "_value": [300, 600]
21   },
22   "momentum": {
23     "_type": "uniform",
24     "_value": [0.7, 1]
25   },
26   "model": {
27     "_type": "choice",
28     "_value": ["resnet50", "densenet121", "resnext50_32x4d"]
29   }
30 }

```

3.提交使用 NNI 自动调参方式，Web UI 上的结果截图。

由于远程运行 NNI 没有 WebUI，因此这里只放了第一次实验的 WebUI 结果截图：





更多截图请看[结果/截图](#)文件夹里查看。

4.实验总结

在本次实验中，遇到了许多或大或小的问题，也积累了很多有用的经验和心得，都值得一说。

一，问题：

1. 问题描述：

安装 Ubuntu 18.04 LTS x86_64 过程中，制作完安装盘后，进入安装和体验 linux 时出现花屏死机。

问题原因：

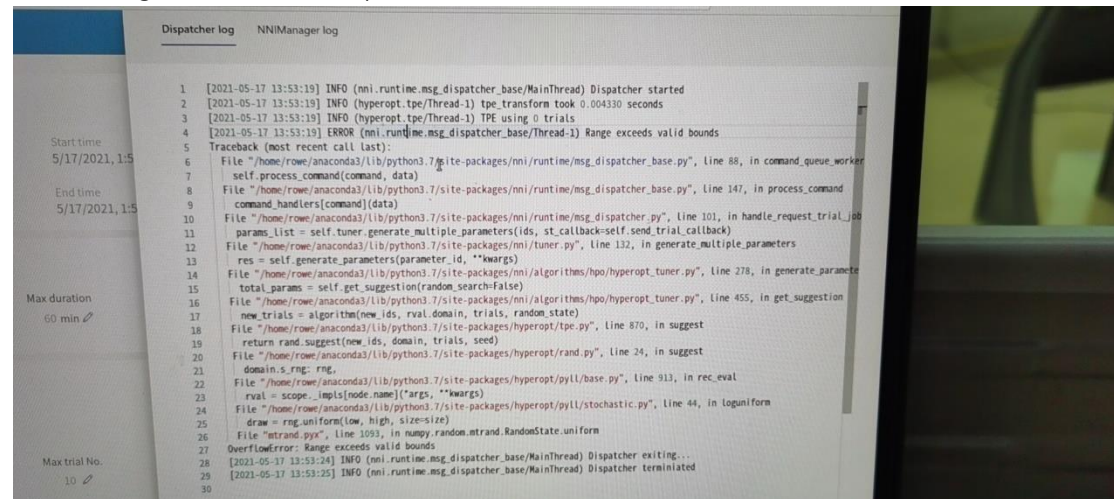
是由于硬盘原因，制作安装盘时使用的是分区而不是一块完整的硬盘，导致引导读取时出现了问题。

解决方法：

<https://blog.csdn.net/legalhighhigh/article/details/81448830>

2. 问题描述：

尝试运行 github 上提供的 hpo-answers，进行初步测试，发生范围溢出报错。



问题原因：

经过排查发现，github 上提供的 hpo-answers 的搜索空间文件编写有问题，其中的

```
"grab_clip": {  
    "_type": "loguniform",  
    "_value": [0.0, 5.0]  
}
```

grab_clip 参数在等于 0 时功能不同于其他取值，其功能等价于 disable，

```
help='gradient clip (use 0 to disable)')
```

因此不能简单用 loguniform 来囊括 0-values。

解决方案：

将 search_space.json 中的 grab_clip 的 type 改为 choice，或者将 0-values 单独出来成为一个选择。

3. 问题描述：

NNI 无法使用 GPU 来运行项目

问题原因：

NNI 系统默认检查空闲的 GPU 并使用，因此有时只要你的 GPU 有哪怕只有 1% 的占用率 NNI 系统都不会使用你的 GPU

解决方案：

在 config.yml 中加入以下部分


```
localConfig:
  useActiveGpu: true
  maxTrialNumPerGpu: 1
```

这些代码的作用是强制使用可用的 GPU，无论是否占用。

4.问题描述:

在 Bitahub 上使用 nni 镜像运行项目任务，任务刚启动就直接成功而结束。

#2	+	成功	2021-05-19 17:56:09	1m 9s
----	---	----	---------------------	-------

问题原因:

在 nni 系统刚启动并准备进行 trials 时，真正的进程还没有开始，但系统却因为暂时无进程而误认为所有任务已经完成，不等待 nni 即将到达的命令就直接结束整个任务。

解决方案:

在运行命令后加上 `&& cat`

启动命令变为:

`mkdir -p /output/nni && nni create --config /code/mine/config.yml && cat`

此命令的作用是让整个任务永远不自动结束，只能用户手动结束，因此不会再让系统错误判断并结束项目。

缺点是就算整个项目完成也无法自动结束，需要用户自行判断并手动结束，否则会浪费算力。

二，经验总结:

1，对 config.yml 和 search_space.json 的改动经验总结

(1) config.yml: 中各主要参数的作用

`trialConcurrency: 1`

同时进行的 trial 数

`maxExecDuration: 10h`

总上限时间，超过这个时间就自动结束。

`maxTrialNum: 10`

总 trial 数，完成后自动结束。

`logDir: .`

输出结果文件的路径

`trial:`

`command: python main.py`

`codeDir: .`

`gpuNum: 1`

gpuNum: 每个 trial 所用 gpu 数量（一般为 1）

`maxTrialNumPerGpu: 1`

每个 gpu 上所跑的 trial 数量（建议为 1）

事实上，在实验中发现，在单个 gpu 上并行的跑多个 trials 并不会有效率提升，总体效率反而会下降。

2，WebUI 使用经验总结

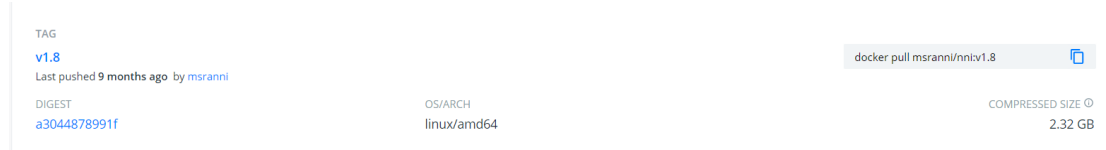
WebUI 的使用非常方便，不仅能随时随地的图形化的查看实验情况，还能临时更改实验的上限时间和 trial 数，甚至即使整个实验完成，只要未彻底停止实验，也能通过 WebUI 追加 trials 以继续运行。还可以详细的看到每一个 trial 的运行情况，结果曲线。



3，在 Bitahub 上使用 NNI 的使用经验总结

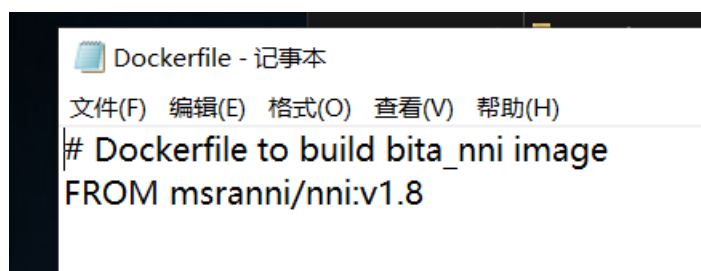
(1) 合适的 NNI 镜像

感谢苏晨林助教提供的 NNI 镜像网址：[msranni/nni\(docker.com\)](https://msranni/nni(docker.com))
这是一个高度集成化的镜像，包含了基本 NNI 实验所需的所有组件和框架。
但本次实验不适合使用最新版本 NNI 的镜像，会出现无法启动的错误
推荐使用：v1.8



(2) 使用 Dockerfile 来构建镜像

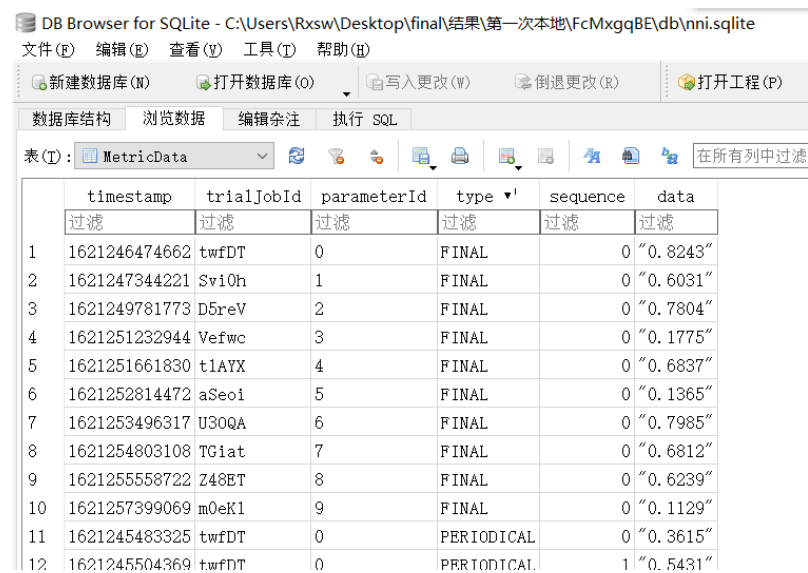
使用 Dockerfile 文件来构建镜像非常简单，一般只需一行代码即可。
编写 Dockerfile 文件也同样十分简单，尤其是在镜像创建者贴心地集成所有必须组件的时候，如本次实验构建 NNI 镜像的 Dockerfile：



然后在 Bitahub 上传构建镜像即可。

4，在 Bitahub 上使用 NNI 后查看结果

使用 *DB Browser for SQLite* 打开结果文件夹中 db\nni.sqlite 文件



The screenshot shows the DB Browser for SQLite application. The title bar indicates the file path: C:\Users\Rxsw\Desktop\final\结果\第一次本地\FcMxgqBE\db\nni.sqlite. The menu bar includes 文件(F), 编辑(E), 查看(V), 工具(T), and 帮助(H). The toolbar has buttons for 新建数据库(N), 打开数据库(O), 写入更改(W), 倒退更改(R), and 打开工程(P). The main window has tabs for 数据库结构, 浏览数据, 编辑杂注, and 执行 SQL. The '浏览数据' tab is active, showing a table named 'MetricData'. The table has 7 columns: timestamp, trialJobId, parameterId, type, sequence, and data. The data is displayed in a list view with 12 rows. The first 11 rows have 'type' 'FINAL' and the last row has 'type' 'PERIODICAL'.

	timestamp	trialJobId	parameterId	type	sequence	data
	过滤	过滤	过滤	过滤	过滤	过滤
1	1621246474662	twfDT	0	FINAL	0	"0.8243"
2	1621247344221	Svi0h	1	FINAL	0	"0.6031"
3	1621249781773	D5reV	2	FINAL	0	"0.7804"
4	1621251232944	Vefwc	3	FINAL	0	"0.1775"
5	1621251661830	t1AYX	4	FINAL	0	"0.6837"
6	1621252814472	aSeoi	5	FINAL	0	"0.1365"
7	1621253496317	U30QA	6	FINAL	0	"0.7985"
8	1621254803108	TGiat	7	FINAL	0	"0.6812"
9	1621255558722	Z48ET	8	FINAL	0	"0.6239"
10	1621257399069	m0eK1	9	FINAL	0	"0.1129"
11	1621245483325	twfDT	0	PERIODICAL	0	"0.3615"
12	1621245504369	twfDT	0	PERIODICAL	1	"0.5431"

就可以看到整个实验过程中所有的数据，包括准确度（包括过程中的），参数设置等

5，使用 NNI 的最终总结：

人工调参一直是一件很困难的事，由于如今的深度学习的深层理论架构仍不够完善，人们对每个超参数之间的联系关系仍一知半解，且没有一个系统的方法去得到理想的参数，因此至今人工调参更多依靠的是个人的经验。

而 NNI 则提供了一个减轻人工调参的负担的方法：只需要编写好搜索空间，设置好运行时间和运行进程数量，然后放那里一直运行就可以了。而且 NNI 在帮你自动调参的同时，可以通过 WebUI 把可视化的工作一起给做了。

通过本次实验，我对 NNI 有了一些了解，且初步学会了使用 NNI 进行自动机器学习，NNI 中还有更多的功能等我去探索。