1 Impact of batch size, number of work, and dataloader

Table 1: Using iNaturalist2018 dataset, standard Resnet50, running for 2 epochs, using multi-epoch dataloader

num_workers	num_gpus	batch_size per GPU	time_per_epoch	time prepare(s)	total time(s)
10	12	64	420	66	905
20	12	64	300	119	802
40	12	64	350	248	966

Table 2: Using iNaturalist2018 dataset, standard Resnet50, running for 2 epochs, using the standard dataloader

num_workers	num_gpus	batch_size per GPU	time_per_epoch	time prepare(s)	total time(s)
10	12	64	650	5	1338
20	12	64	650	5	1263
40	12	64	650	4	1308

Table 3: Using CIFAR10 dataset, standard Resnet32, running for 200 epochs, using the multiepoch dataloader

num_workers	num_gpus	batch size per GPU	total batch size	time prepare	total time	runtime	best acc
10	12	64	768	58	544	486	91.92
15	12	64	768	84	564	480	91.44
20	12	64	768	110	592	482	91.65
40	12	64	768	214	703	489	91.83
10	12	32	384	57	993	936	92.11
10	12	128	1536	57	322	265	89.78
15	6	128	768	91	518	427	91.77
15	3	256	768	91	509	418	91.31

Results suggest that

- Tables 1 and 2 show that multi dataloader require much less computational time than the regular dataloader. Using less worker saves time for multi-epoch dataloader.
- Number of workers should not be too large or too small, maybe 15 is fine.
- Batch size has an impact on the performance and the runtime. The smaller batch size, the longer runtime is required.
- When GPU memory is enough regarding the target batch size, the fewer GPUs to use the better, probably due to that fact that the communication between GPUs for gradients takes time.