

W251 - Homework #9

Travis Metz

NB: When I first spooled up two VMs, WDC07 let me set up only one with 2xV100. So I set up the other in LON04. But once I started training the model it became apparent that this slowed down the network speed substantially (was only getting 20mbps of throughput between the two VMs). So I tore down the WDC07 one and built another one in LON04 and re-started process. But by this point I was also sorting out that I had substantially overrun on my IBMCloud credits (which is subject of separate email chain with Brad and Darragh) so I ended up stopping the training at 20k steps, not 50k steps as recommended. So my results below reflect (unless otherwise noted) the results through 20k steps. And because I was anxious to tear down the machines to stop the spending, I did not carefully inspect the checkpoint files before deleting machines.

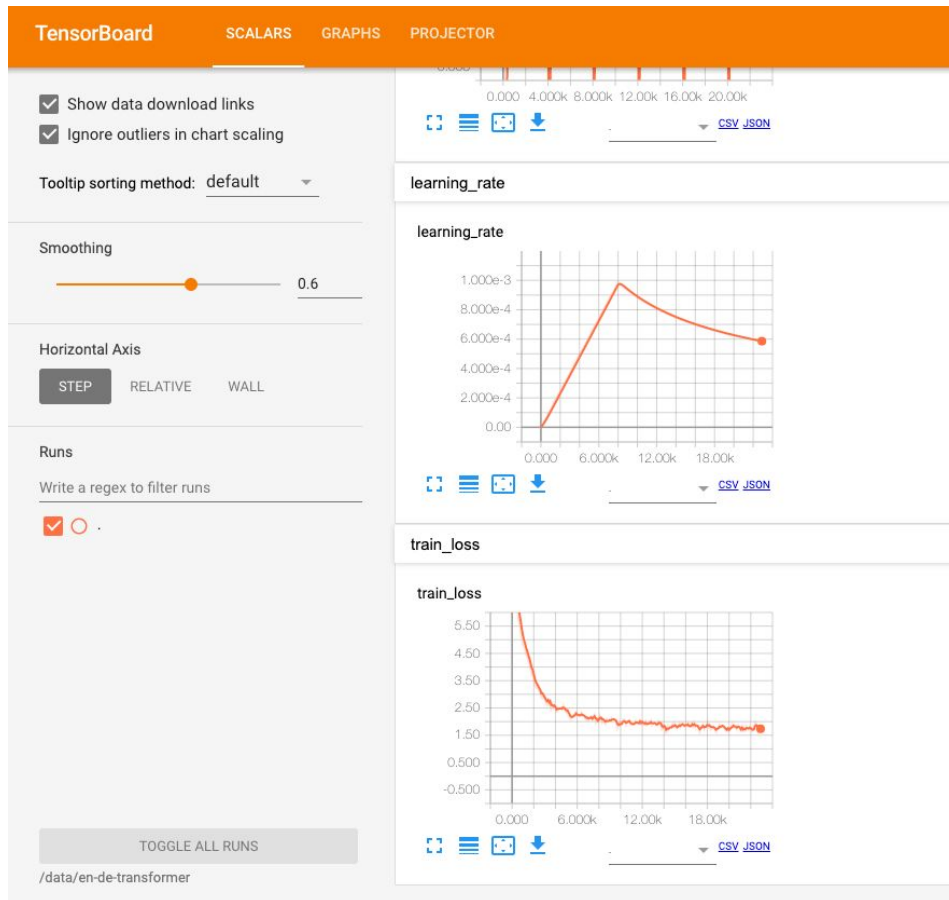
How long does it take to complete the training run? (hint: this session is on distributed training, so it *will* take a while)

I got to 20,000 steps in just under nine hours before I shut down for reasons described above. This is approximately 1.6 seconds per step.

Do you think your model is fully trained? How can you tell?

At 20,000 steps, I think it is not fully trained. Training loss continues to decline and the BLEU evaluation score is increasing.

Eval BLEU score	
Step	Value
0	0.95020002
0	0.94910002
4001	0.2269
8002	0.28580001
12003	0.31990001
16004	0.33250001
20005	0.3452



Were you overfitting?

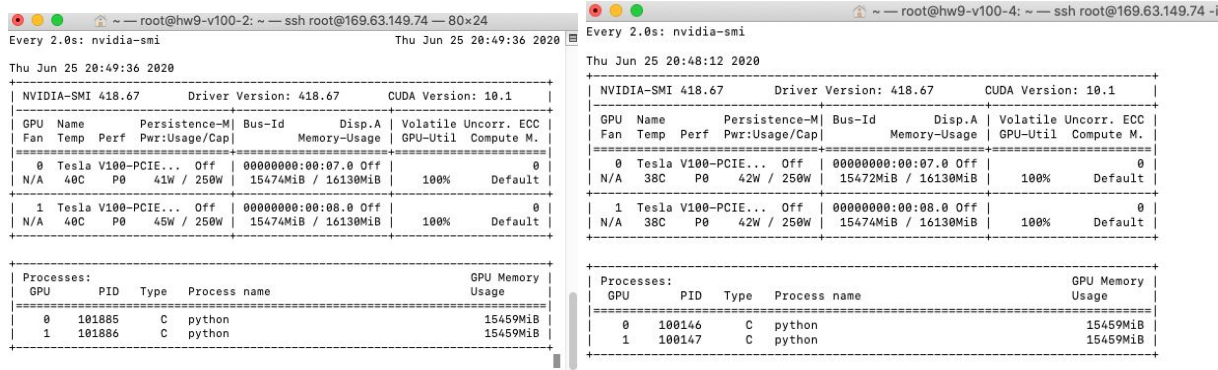
No, the eval loss was continuing to decrease and the eval BLUE score was continuing to increase.

Eval Loss	
Step	Value
0	9.42892742
0	9.51872635
4001	2.43809485
8002	2.03338408
12003	1.84889245
16004	1.785671
20005	1.73930073

Were your GPUs fully utilized?

Screenshot below shows usage of four GPUs when I was in constrained network state (described below). They all show 100% utilization, though their power usage seems low. They

were also 100% utilized when the network throughput was increased by ~10x.



Every 2.0s: nvidia-smi									
Thu Jun 25 20:49:36 2020									
NVIDIA-SMI 418.67		Driver Version: 418.67		CUDA Version: 10.1					
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M.		
0	Tesla V100-PCIE...	Off	00000000:00:07:0	Off			0		
N/A	40C	P0	41W / 250W	15474MiB / 16130MiB	100%	Default			
1	Tesla V100-PCIE...	Off	00000000:00:08:0	Off			0		
N/A	40C	P0	45W / 250W	15474MiB / 16130MiB	100%	Default			

Processes:					
GPU	PID	Type	Process name	GPU Memory Usage	
0	101885	C	python	15459MiB	
1	101886	C	python	15459MiB	

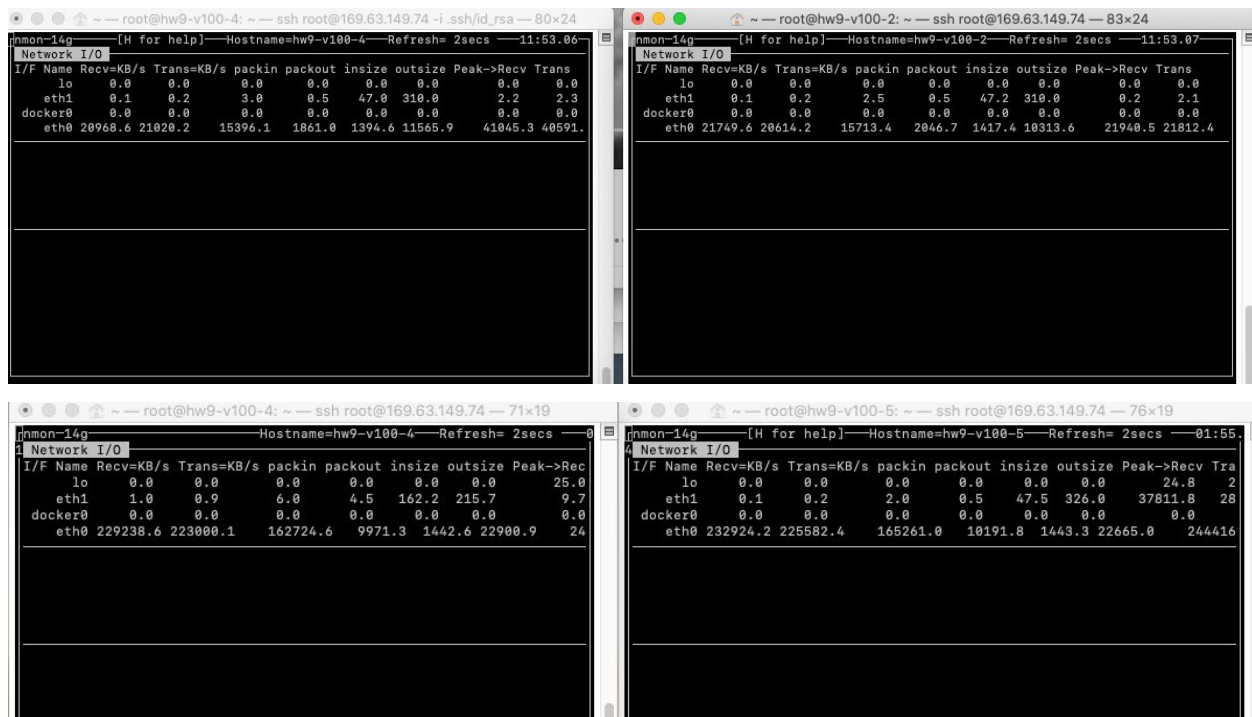
Every 2.0s: nvidia-smi									
Thu Jun 25 20:48:12 2020									
NVIDIA-SMI 418.67		Driver Version: 418.67		CUDA Version: 10.1					
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M.		
0	Tesla V100-PCIE...	Off	00000000:00:07:0	Off			0		
N/A	38C	P0	42W / 250W	15472MiB / 16130MiB	100%	Default			
1	Tesla V100-PCIE...	Off	00000000:00:08:0	Off			0		
N/A	38C	P0	42W / 250W	15474MiB / 16130MiB	100%	Default			

Processes:					
GPU	PID	Type	Process name	GPU Memory Usage	
0	100146	C	python	15459MiB	
1	100147	C	python	15459MiB	

Did you monitor network traffic (hint: apt install nmon) ? Was network the bottleneck?

The first screenshot shows network traffic on the two VMs when they were in separate regions. Average in/out was ~20,000 KB/S. The second screenshot shows network traffic when both VMs were in same region (~200,000 KB/S, so order of magnitude faster).

My GPUs were 100% utilized in both the slow and fast network scenarios, but went from processing at 17.5 seconds per step to 1.6 seconds per step, so the network must have been the bottleneck between those two cases. (I can't quite sort out why the GPU utilization remained at 100% in both cases - perhaps has to do with power utilization.)



nmon-14g [H for help] Hostname=hw9-v100-4 Refresh= 2secs 11:53.06													
Network I/O													
I/F	Name	Recv=KB/s	Trans=KB/s	packin	packout	insize	outsized	Peak->Recv	Trans				
lo		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
eth1		0.1	0.2	3.0	0.5	47.0	310.0	2.2	2.3				
docker0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
eth0		20968.6	21020.2	15396.1	1861.0	1394.6	11565.9	41845.3	40591.0				

nmon-14g [H for help] Hostname=hw9-v100-2 Refresh= 2secs 11:53.07													
Network I/O													
I/F	Name	Recv=KB/s	Trans=KB/s	packin	packout	insize	outsized	Peak->Recv	Trans				
lo		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
eth1		0.1	0.2	2.5	0.5	47.2	310.0	0.2	2.1				
docker0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
eth0		21749.6	20614.2	15713.4	2046.7	1417.4	10313.6	21940.5	21812.4				

nmon-14g [H for help] Hostname=hw9-v100-4 Refresh= 2secs 01:55.00													
Network I/O													
I/F	Name	Recv=KB/s	Trans=KB/s	packin	packout	insize	outsized	Peak->Recv	Trans				
lo		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
eth1		1.0	0.9	6.0	4.5	162.2	215.7	9.7	9.7				
docker0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
eth0		229238.6	223000.1	162724.6	9971.3	1442.6	22900.9	24					

nmon-14g [H for help] Hostname=hw9-v100-5 Refresh= 2secs 01:55.01													
Network I/O													
I/F	Name	Recv=KB/s	Trans=KB/s	packin	packout	insize	outsized	Peak->Recv	Trans				
lo		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
eth1		0.1	0.2	2.0	0.5	47.5	326.0	37811.8	28				
docker0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0				
eth0		232924.2	225582.4	165261.0	10191.8	1443.3	22665.0	244416					

Take a look at the plot of the learning rate and then check the config file. Can you explain this setting?

We appear to be using the Transformer's learning rate policy from the [Attention Is All You Need](#) paper. In our case, we used 8000 warmup steps (rather than 4000 in original paper). Thus learning rate increases through step 8000, and decreases thereafter.

```
"lr_policy": transformer_policy,  
"lr_policy_params": {  
    "learning_rate": 2.0,  
    "warmup_steps": 8000,  
    "d_model": d_model,
```

Learning Rate	
Step	Value
7401	0.00091422
7501	0.00092657
7601	0.00093892
7701	0.00095128
7801	0.00096363
7901	0.00097598
8001	0.00098815
8101	0.00098203
8201	0.00097603
8301	0.00097013
8401	0.00096434
8501	0.00095865

$$lr_{rate} = d_{model}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5}) \quad (3)$$

This corresponds to increasing the learning rate linearly for the first *warmup_steps* training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used *warmup_steps* = 4000.

How big was your training set (mb)? How many training lines did it contain?

The training set was 959mb. I believe it had 4,524,868 lines (or sentences).

What are the files that a TF checkpoint is comprised of?

There are three files - meta, data, index. Meta describes the graph. Data contains the parameters. Index contains the info for each tensor.

How big is your resulting model checkpoint (mb)?

I am using a checkpoint from nvidia. It is 741mb in total.

Remember the definition of a "step". How long did an average step take?

1.6 seconds with improved network. I believe each step is training the transformer on one batch (256 sentences).

How does that correlate with the observed network utilization between nodes?

I do not know. The nodes are transferring > 200,000 kb/s. In that same time period we are training on approximately 160 samples. The model parameter file is approximately 700mb, so that would imply that we are passing all the parameters back and forth every three seconds or so, or over 2 steps.

Inference on Jetson

Have been struggling to get model to run on Jetson. The image created by provided Dockerfile did not work out of box as the download LLVM version (7.0.1) is incompatible with the downloaded version of LLVMlite (0.33). Resolved that by downloading LLVM (9.0.1), but now struggling to sort numba/scipy incompatibility upon running of model for inference.