

# Benchmarks

- [Laptop ram upgrade](#)
- [nvidia apex](#)
- [CFG.jl](#)
- [ROCm pytorch](#)
- [Linpack results](#)

# Laptop ram upgrade

## before upgrade:

- unigine heaven: - 9.4 fps - 236 score - min 5.4 - max 21.2 - 1920 x 1080
- universe sandbox 37

## after upgrade

- unigine heaven
  - 11.1 fps
  - 281 score
  - min 7.0
  - max 25.2
  - 1920 x 1080
- universe sandbox 39

# nvidia apex

Scarecrow 1123 wrote a trainer for allennlp that uses nvidia's apex package to enable mixed precision training.

The full gist is available [here](#).

This is a copy of the trainer provided..

I find that my models are more often successful if I specify "O1" instead of "O2" for amp. This uses only a set of whitelisted operations in half precision mode.

[This trainer](#) has the change already made.

To use this during training include a snippet like this in your training json config.

```
{
  // ....
  "trainer": {
    "type": "fp16-trainer",
    "mixed_precision": true,
    // other options
  }
  // ....
}
```

and make sure the trainer is in a directory that you are including using `--include-package`.

For a bert model I was training, it ran out of VRAM on a single GTX 1070 without apex configured. However with apex configured the model was only using 4.5GB. There was no discernable penalty with regard to the number of epochs required though I haven't investigated a ton.

# CFG.jl

grammar size: rules: 52 lexicon: 66

parsing execution times on a single thread

size	time
300 sents	1.172 seconds
3,000 sents	2.988 seconds
30,000 sents	22.06 seconds
300,000 sents	208.03 seconds

parsing execution times on two threads

size	time
300 sents	1.000 seconds
3,000 sents	2.216 seconds
30,000 sents	13.874 seconds
300,000 sents	127.376 seconds

# ROCm pytorch

Used this tutorial to install pytorch for rocm, however I checked out release 1.5.

<https://github.com/ROCmSoftwarePlatform/pytorch/wiki/Building-PyTorch-for-ROCm> Allennlp was version 0.9.

## GRU

## BERT

This used bert-base with a batch size of 8.

## Vega FE notes

The vega frontier edition results were obtained from a rented gpueater instance.

A batch size of 16 was also tried for the vega frontier edition to see if it would fit in vram and strangely the time per epoch dropped (01:12) with the larger batch size). This was also with thermal throttling as the vega fe was hitting 87 C and the clocks were down to 1.2 Ghz from 1.6 Ghz. The fans were limited to 40% under load on gpueater.com. It would be interesting to see what the performance is like with better thermals.



GRU  
BERT  
pospos-  
base  
tag  
tagger  
(1- (2-  
hid  
regres

GTX 1070	1:26:00.24.3		
Tesla M40	1:32:00.04.3		
RTX 3090	0:20:20.2.6		
RX580	1:14:00.98.5		
Vega Frontier	1:29:34.05.1		
Vega Frontier (90% fans)	1:09:02.33.0		
Vega frontier (rocm 4.0)	1:07:52.42.9		
i7- 7800x	00:18:23		
i9- 7900x (defective?)	00:19:23		
i9- 7900x	00:16:20		
i9- 7980xe	00:15:18		
e5- 2680v3	00:20:34		

using rocm apex gave no discernable performance improvement (with use\_apex = true) However, it did reduce memory consumption by ~1GB for a batch of 16.

The RTX 3090 was tested with cuda 11, all other nvidia gpus were using cuda 10.2 (the RTX 3090 is not supported in this earlier version of cuda).

# Linpack results

Instance	Processor	RAM size	RAM speed	GFLOPS	Notes
					No
e5-2690	48	133GB	3avx2		
	40000	1152n1			
	x4				this cpu
e5-2680v3	16	2133GB	3size.		limited by memory
	30000	1273.2			
	x4				Not the peak performance
7835	32	3200Ghz	3200Ghz		4.1
	50000	1535x-			
	GB				
	x4.				512 clock
7940	32	2933No			
	40000	570.5			
	GBx2				overclock
7940	32	2200No			
	40000	660.7			
	GBx4.				overclock
7940	32	200Rented			
	128	1275n			
	GB				
	x4				vast.ai