Analysis of Nvidia OpenSeq2Seq (v0.5)

Code base at https://github.com/NVIDIA/OpenSeq2Seq

Documentation can be found at https://nvidia.github.io/OpenSeq2Seq/html/index.html

In this study I looked at 4 algorithms from the OpenSeq2Seq package. This is an extremely practical set of applications for studying algorithm performance as they are all based on a single framework, Tensorflow, and are run from a common harness. This creates a standardized "look and feel" making configuration and manipulation of the algorithms a great deal easier. Further as the code base is maintained by Nvidia one can reasonably assume representative performance on Nvidia GPUs will be achieved "out of the box".

The different applications and algorithms appear to have a common style of logging, printing out the parameter sizes, current step, loss and time/step. The logging control parameters sometimes need a bit of editing to keep the logs easy to read. There is a tendency for excessive intermediate evaluation output which makes the logs very large. The applications can be run in FP32 or a mixed mode of fp16 and fp32. The default is not always fp32. Switching to mixed mode can also enable increasing the batch size, which for many of these large applications will increase the throughput. More on mixed mode later.

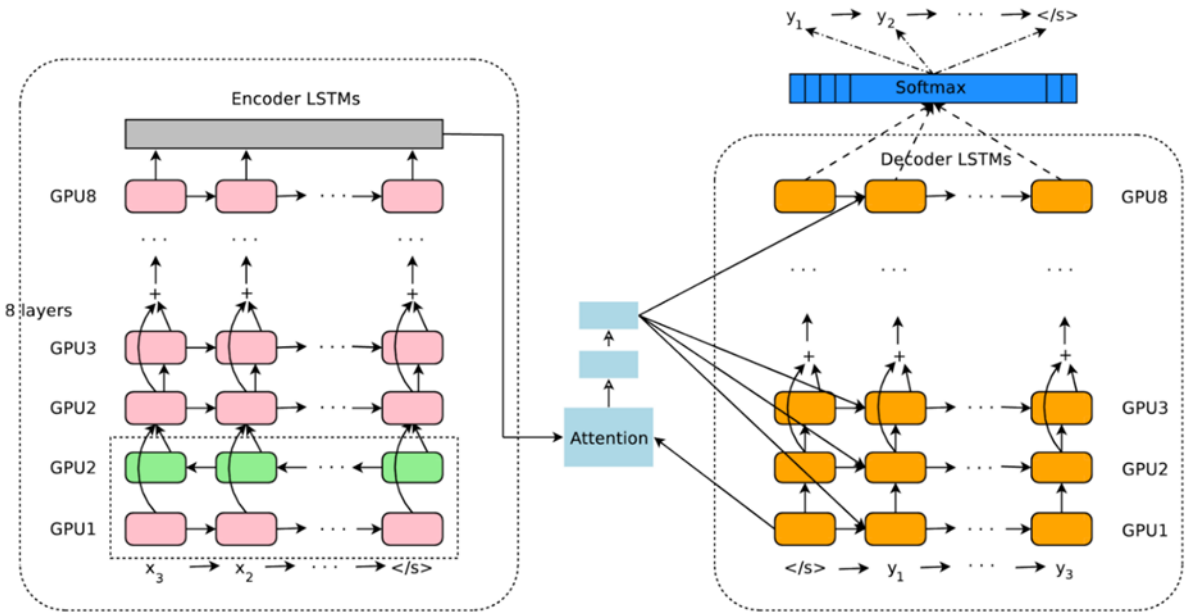
Two were translators based on Google open sourced codes. One for an LSTM based seq2seq translator (see https://arxiv.org/pdf/1609.08144.pdf and https://github.com/tensorflow/nmt) and one the transformer based algorithm (see https://arxiv.org/pdf/1706.03762.pdf and https://github.com/tensorflow/tensor2tensor) The other two were speech to text codes. The first was a tensorflow based encoding of Deepspeech2 (see https://arxiv.org/abs/1512.02595). The second a tensorflow encoding of the Jasper algorithm. In these studies, the librispeech data set was used for training and was not augmented using synthesized data (Tacotron2) as discussed in the OpenSeq2Seq documentation. (https://nvidia.github.io/OpenSeq2Seq/html/speech-recognition/synthetic_dataset.html,

Translation

8 layer GNMT-like algorithm (diagram from https://arxiv.org/pdf/1609.08144.pdf)

Hidden size = embedding size = 1024, gnmt_v2 style attention, forget bias = 1 first encoder bi directional, Adam optimizer, default FP representation fp32, batch size 32/gpu. The batch size was not changed for the mixed mode run on cuda 9.2.

Total parameters 275 million



Transformer

Diagram from https://arxiv.org/pdf/1706.03762.pdf

Big model uses batch size 64/gpu by default, 6 layers, default FP32

Total parameters: 210 million

I have not investigated whether the batch size can be changed for mixed mode, just changed the data type and ran with everything else the same

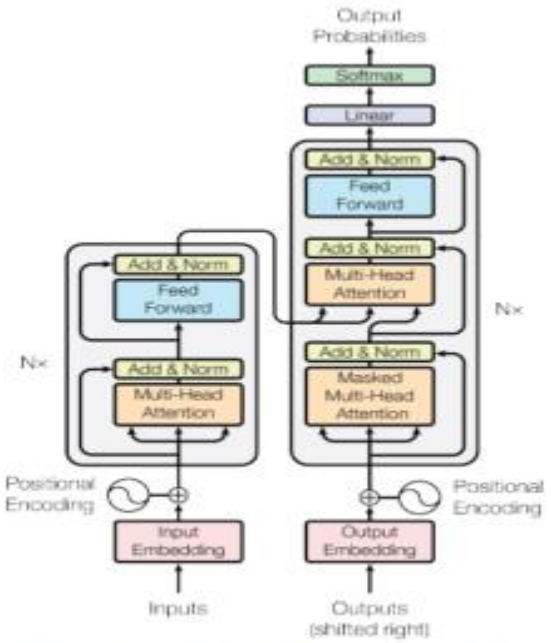


Figure 1: The Transformer - model architecture.

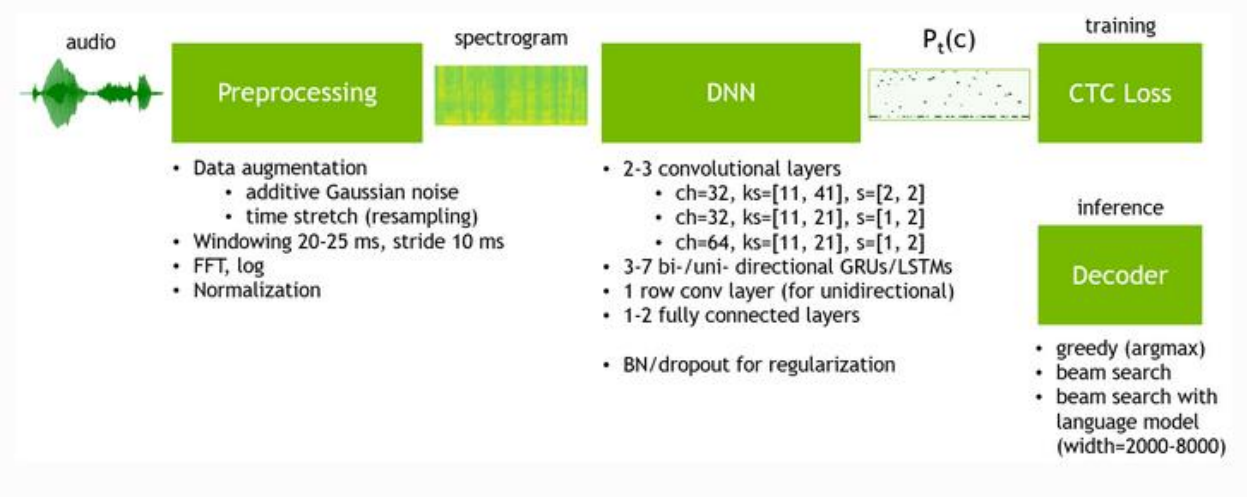
Nvidia supports several models for translation and achieved the following accuracies for English to German translation.

Model description	SacreBLEU	Config file	Checkpoint
Transformer	27.52	transformer-big.py	link
Transformer	26.4	transformer-base.py	link
ConvS2S	25.0	en-de-convs2s-8-gpu.py	link
GNMT	23.0	en-de-gnmt-like-4GPUs.py	TBD

Speech to Text

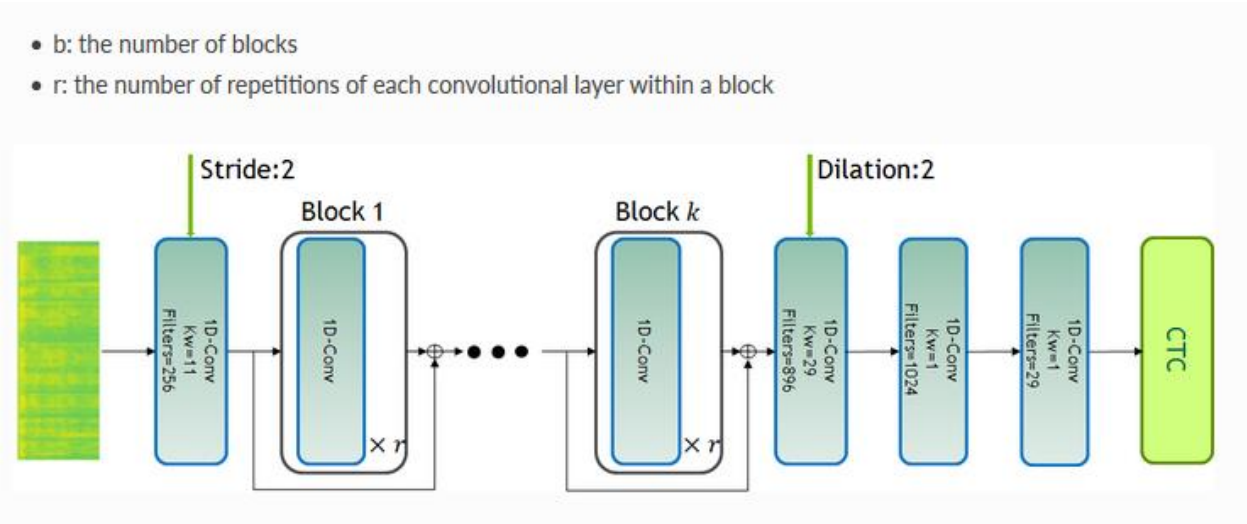
These algorithms all use Connectionist Temporal Classification loss (CTC loss https://www.cs.toronto.edu/~graves/icml_2006.pdf).

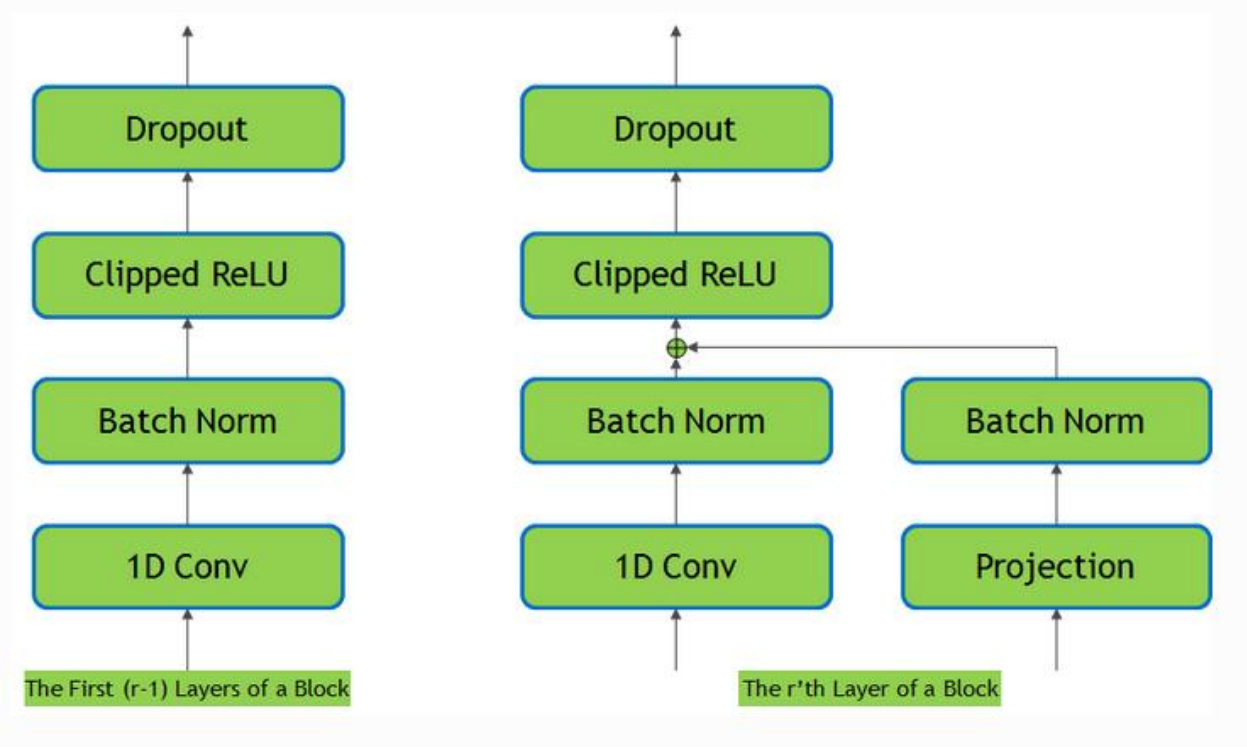
Deepspeech2



The deepspeech2 model fails when attempting to print out all the array sizes (https://github.com/NVIDIA/OpenSeq2Seq/issues/316) but runs to completion after complaining about unknown array size.

Jasper models are denoted as Jasper bxr where b and r represent:





The 10X3 model (smaller of the 2 examples) I ran has 200.5 million parameters

The 10X5 model has 322 million parameters

The package has more algorithms. Nvidia achieved the following accuracies:

Model description	WER, %	Config file	Checkpoint
DeepSpeech2	6.71	ds2_large_mp	link
Wave2Letter+	6.67	w2l_plus_large_mp	link
Jasper 10x3	5.10	jasper_10x3_8gpus_mp	link
Jasper 10x5 syn	4.32	jasper_10x5_8gpus_mp	link
Jasper 10x5 dense res syn	4.15	jasper_10x5_8gpus_dr_mp	link

Mixed mode: (*arXiv preprint arXiv:1710.03740*, 2017)

To set this mode one merely sets a flag in the configuration python script (examples in ~/OpenSeq2Seq/example_configs/speech2text for the DS2 and jasper examples) adding

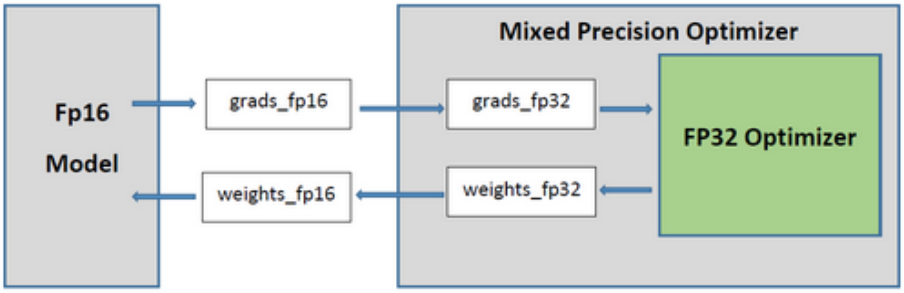
diff ds2_medium_4gpus_orig.py ds2_medium_4gpus.py

> "dtype" : "mixed",

And moving the saved checkpoint directory (so a new one can be created) and rerunning the exact same invocation, but changing the target log file for stdout and stderr

python3.6 run.py --config_file=example_configs/speech2text/ds2_medium_4gpus.py --mode=train_eval --use_horovod=False > med_4gpu_ds2_mixed.log 2>&1

The mechanism maintains and updates an fp32 copy of the weights but uses fp16 for forward and back propagation. This was implemented with a wrapper around the standard Tensorflow optimizers so everything that is not explicitly required to be fp32 (operations on CPU for example) is done in fp16.



"Mixed precision" optimizer wrapper around any TensorFlow optimizer.

The performance improvement associated with setting the mixed option depends on the algorithm quite strongly. In extreme cases one can gain performance from multiple effects achieving performance gains well above 2. In other cases, the gain can be minimal. In the case of DS2 a bug report was submitted and an upgrade to the source made by Nvidia. In addition, it was strongly recommended to upgrade cudnn to 7.4.2 and build TF with Horovod. This is now underway.

The experiments were performance using a machine with 4 V100 GPUs. Tensorflow R1.12 built from source per the instructions in https://nvidia.github.io/OpenSeq2Seq/html/installation.html. In some cases, the combination of cuda 9.2, cudnn 7.1.4 and nccl 2.1.3 was used. In others the Nvidia libraries were upgraded to cuda 10, cudnn 7.3.1 and nccl 2.3.5. Horovod was not included in these builds.

Below are a table of results on a system with 4 16GB V100s, with all 4 in use.

workload	fp32 time/step	mixed time/step	ratio	build
ds2_med	2.95	2.81	1.049822064	cuda9.2
GNMT 8 layer	1.15	0.917	1.254089422	cuda9.2
workload	fp32 time/step	mixed time/step	ratio	build
ds2_med	2.04	2.05	0.995121951	cuda10
jasper 10x3	2.33	1.12	2.080357143	cuda10
jasper 10x5				cuda10
GNMT 8 layer	1.15	0.963	2.388369678	cuda10
large transformer	0.954	0.469	2.034115139	cuda10
GNMT mixed run at double batch size of fp32				

Nvidia updated the ds2 code and recommended using newer libcudnn and installing horovod.
A new build was made using the same cuda10.0 package and the list shown below
cuda-repo-ubuntu1604-10-0-local-10.0.145-410.48_1.0-1_amd64.deb
libcudnn7_7.4.2.24-1+cuda10.0_amd64.deb
libcudnn7-dev_7.4.2.24-1+cuda10.0_amd64.deb
libcudnn7-doc_7.4.2.24-1+cuda10.0_amd64.deb
nccl-repo-ubuntu1604-2.3.5-ga-cuda10.0_1-1_amd64.deb
nv-tensorrt-repo-ubuntu1604-cuda10.0-trt5.0.0.10-rc-20180906_1-1_amd64.deb
and a new download of OpenSeq2Seq which had received updates since this work started.
Ubuntu 16.04.5 was the OS. Python3.6 was installed from source into /usr/src following
https://tecadmin.net/install-python-3-6-ubuntu-linuxmint/
TensorRT was not used in the Tensorflow build which used version r1.12, set with a git checkout.
To install Horovod the machine was first scrubbed of all MPI components and then openmpi.4.0.0.tar.gz was built and installed and .bashrc environment variables updated. Tensorflow was built from source following the instructions in OpenSeq2Seq documentation. The exception was that the pip install of the wheel used a target directory using the -t option. MPI was not integrated into Tensorflow during the invocation of configure. Installing mpi4py first required scrubbing old versions with references to the removed mpi components. The install with openmpi.4.0 resulted in an encounter with https://bitbucket.org/mpi4py/mpi4py/issues/115/cannot-build-against-openmpi-400. The workaround therein "sudo pip3.6 install https://bitbucket.org/mpi4py/mpi4py/get/maint.zip" solved the mpi4py installation. Installing horovod when one uses a directory for the tensorflow wheel and the PYTHONPATH environment variable requires one installs horovod as follows:
sudo PYTHONPATH=/home/levinth/tf_r12_10_741_235_py36 HOROVOD_GPU_ALLREDUCE=NCCL pip3.6 install --no-cache-dir horovod
as the Tensorflow wheel had been installed into =/home/levinth/tf_r12_10_741_235_py36
At this point a new shell only required setting PYTHONPATH.
The invocation of a OpenSeq2Seq application changes if using horovod is desired. An invocation like:

```
python3.6 -u run.py --config_file=example_configs/speech2text/ds2_medium_4gpus_mixed_bs64.py \
--mode=train --use_horovod=False > ds2_mixed_cuda10_74_bs64.log 2>&1
```

becomes

```
mpiexec --allow-run-as-root -np 4 \
python3.6 -u run.py --config_file=example_configs/speech2text/ds2_medium_4gpus_mixed_bs64.py \
--mode=train > ds2_mixed_cuda10_74_bs64_hor_mpiexec.log 2>&1
```

Instructions for the installation of cuda10, OpenSeq2Seq and horovod can be found at
https://github.com/David-Levinthal/machine-learning

The results of the runs taken with the latest libraries and horovod are shown below. I t certainly appears that Horovod improves performance even on a single machine. The lack of performance improvement on DS2 when going from fp32 to fp16/mixed is quite confusing.
In order to keep the run time down, runs of the 10x3 jasper code on the latest libraries with horovod were cut off at 40 epochs rather than the default 400.
Future work will include moving to TF r1.13.

Horovod, cudnn7.4.2, cuda10, 4V100						
	batch size	FP	time	loss	horovod/mpi	mixed/fp32 throughput
ds2_med	64	fp32	crashed		yes	
ds2_med	32	fp32	1.67	17.1	yes	
ds2_med	64	mixed	4.27	20	no	
ds2_med	32	mixed	1.56	16	yes	1.070512821
ds2_med	64	mixed	3.2	17.1	yes	1.04375
						mixed/fp32 throughput
GNMT 8 layer	64	mixed	0.932	0.7	yes	1.394849785
GNMT 8 layer	128	mixed	1.06	~0.8	yes	2.452830189
GNMT 8 layer	64	fp32	1.3	0.75	yes	took 2 tries to get it to run
GNMT 8 layer	128	fp32	OOM		yes	
transformer-big	256	mixed	OOM		yes	
transformer-big	128	mixed	0.59	1.5	yes	2.922033898
transformer-big	128	fp32	OOM		yes	
transformer-big	64	fp32	0.862		yes	
only 40 epochs, not 400						
jasper 10x3	32	mixed	1.24			2.290322581
jasper 10x3	64	mixed	2.02	~9	yes	2.811881188
jasper 10x3	64	fp32	OOM		yes	
jasper 10x3	32		2.84	~9	yes	

Since these results were collected newer libraries have become available.
New results will be added soon.
The installation instructions for the new packages are a bit different and have been posted at:
https://github.com/David-Levinthal/machine-learning/blob/master/nvidia_installation_cuda10.1.txt