**Unbiutu Tips: To switch to root, sudo -s (**[**https://stackoverflow.com/questions/24313562/root-login-ubuntu-vm-on-azure**](https://stackoverflow.com/questions/24313562/root-login-ubuntu-vm-on-azure)**)**

*Tips:*

1. *DLVM’s tensorflow is installed in Anaconda so inside x-term, tensorflow cmd will not be found. So use “source activate tensorflow” to activate the tensorflow in order to run the python cmd*
2. *If see internal service error, like 500 in the workspace, goto Azure portal’s wtamExperinment and “****resync the storage key****”*

**DSVM/DLVM Jupyter tips: (**By default, Jupyter Notebook only accepts connections from localhost (eg, from the same computer that its running on). By modifying the NotebookApp.allow\_origin option from the default ' ' to '\*', you allow Jupyter to be accessed externally.)

* **Use X-term to logon**, than cd /anaconda/bin
* jupyter notebook --generate-config
* edit /home/wtam/.jupyter/jupyter\_notebook\_config.py
  + c.NotebookApp.allow\_origin = '\*' #allow all origins
  + c.NotebookApp.ip = '0.0.0.0' # listen on all IP
* add the VM with port 8888 inbound open
* **Run the Jupyter on the Xterm’s /home/wtam/Desktop icon**
* **Open browser** and replace the token
  + [http://dsvmmwr2ii6hetoh4.australiaeast.cloudapp.azure.com:8888/**?**token=3274744c41d16983a4c277e3c5370f27fd66ac5ec959f0cf](http://dsvmmwr2ii6hetoh4.australiaeast.cloudapp.azure.com:8888/?token=3274744c41d16983a4c277e3c5370f27fd66ac5ec959f0cf)

OR

**http://dsvmmwr2ii6hetoh4.australiaeast.cloudapp.azure.com:8888/**

* + Remove the token by adding this into the /home/wtam/.jupyter/jupyter\_notebook\_config.py
    - c.NotebookApp.token = ''
    - c.NotebookApp.password\_required = True
    - ## Hashed password to use for web authentication.
    - #
    - # To generate, type in a python/IPython shell:
    - #
    - # from notebook.auth import passwd; passwd()
    - #
    - # The string should be of the form type:salt:hashed-password.
    - c.NotebookApp.password = 'sha1:bede9bd803cc:b26da7f343e276cd2ba6f9b530bed2ec1cffb681'

**Tensor flow with AMLWorkBench** <https://github.com/Azure/MachineLearningSamples-tf>

[AZMLWorkBench, use the cmd prompt to run the juypter >**az ml notebook start**] Watchout, prj can’t bigger than 25M. so no big dataset folder!!

1. Create a Machine Learning Experimentation, Machine Learning Model Management

<https://docs.microsoft.com/en-gb/azure/machine-learning/preview/quickstart-installation>

* create a ML Experimentation with Model management from the Azure portal (acctname = wtamExperiment)
  + using williamWorkspace,
  + wtamexperimentstorage ,
  + wtamExperimentModelMgmt
  + in the MLWorkBenchRG (S1 plan has 100model and 10 deployment)

1. Install the MLWorkBench either on Windows or MacOS then create a project under C:\Github-MyWork\TensorflowLearning folder

1st Project : MNIST using <https://github.com/Azure/MachineLearningSamples-tf>

* + Open the command-line window by clicking on \*\*File\*\* --> \*\*Open Command Prompt\*\*, then run `tf\_mnist.py` in local Python environment installed by Azure ML Workbench by typing in the following command.
    - # first install tensorflow library using pip, you only needed to do this once.

(pip install tensorflow or tensorflow\_gpu)

Then I see this Error: Could not find 'cudart64\_80.dll'when I click the run in local environment, so I Download and install CUDA 8.0 from this URL: <https://developer.nvidia.com/cuda-80-ga2-download-archive> Make sure choose win10

Then also download the cudnn [cuDNN v6.0 Library for Windows 10](https://developer.nvidia.com/compute/machine-learning/cudnn/secure/v6/prod/8.0_20170307/cudnn-8.0-windows10-x64-v6.0-zip) that work for cuda8.0 Then copy the cuda/bin/cudnn64\_6.dll to C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\bin directory and also the include and lib folder respectively(Note: use the Workbench’s CMD prompt to check the path with echo %PATH%)

* + Try running the tf\_gpu\_test.py but failed with this error
    - tensorflow.python.framework.errors\_impl.InvalidArgumentError: Cannot assign a device for operation 'MatMul': Operation was explicitly assigned to /device:GPU:0 but available devices are [ /job:localhost/replica:0/task:0/device:CPU:0 ]. Make sure the device specification refers to a valid device.
    - [[Node: MatMul = MatMul[T=DT\_FLOAT, transpose\_a=false, transpose\_b=false, \_device="/device:GPU:0"](a, b)]]
    - **So I need to change the tf.session to allow** allow\_soft\_placement [**https://github.com/InFoCusp/tf\_cnnvis/issues/14**](https://github.com/InFoCusp/tf_cnnvis/issues/14)
  + Try running the tf\_mnist.py in docker environment and receive the following error
    - Step 1/14 : FROM microsoft/mmlspark:plus-0.9.9
    - plus-0.9.9: Pulling from microsoft/mmlspark
    - image operating system "linux" cannot be used on this platform

<https://github.com/Azure/mmlspark/pull/195>

So I change the baseDockerImage to v0.10 from 0.9.9 in the docker.compute. Then I got this error “Azure ML Workbench requires the mmlspark plus variants”

[Note: <https://hub.docker.com/r/microsoft/mmlspark/tags/> ]

**So let try run the docker in remote VM such as DSVM/DLVM that has docker env installed already**

[**https://docs.microsoft.com/en-us/azure/machine-learning/preview/tutorial-classifying-iris-part-2**](https://docs.microsoft.com/en-us/azure/machine-learning/preview/tutorial-classifying-iris-part-2)

**Pre-step: Make sure the DSVMDemo-Keras VM is running 1st**

1. **creates an myvm compute target**

**az ml computetarget attach remotedocker --name** DSVMDemo-Keras **--address** dsvmdemo-keras.westus2.cloudapp.azure.com **--username wtam --password Wdam1234!!!!**

**OR**

**az ml computetarget attach remotedocker --name** DSVMDemo --address dsvmdemo.westus2.cloudapp.azure.com **--username wtam --password Wdam1234!!!!**

**OR**

**az ml computetarget attach remotedocker --name** DLVMDemo --a dsvmmwr2ii6hetoh4.australiaeast.cloudapp.azure.com **--username wtam --password Wdam1234!!!!**

1. **az ml experiment prepare -c DSVMDemo-Keras**

**OR**

**az ml experiment prepare -c DLVMDemo**

1. **Edit the DLVMDemo.runconfig : Framework : Python**
2. **REM executes iris\_sklearn.py in a remote Docker container**

**az ml experiment submit -c** DSVMDemo-Keras **iris\_sklearn.py**

**or**

1. **az ml computetarget attach remotedocker --name DLVMDemo --address dsvmmwr2ii6hetoh4.australiaeast.cloudapp.azure.com --username wtam --password Wdam1234!!!!**
2. **az ml experiment prepare -c DLVMDemo [Note: I need to get the docker env setup everytime I update the .Compute and/or runconfig files ]**
3. **Edit the DS.runconfig : Framework : Python**
4. **REM executes iris\_sklearn.py in a remote Docker container**

**az ml experiment submit -c** DLVMDemo **Tensorflow-tutorial.py**

**Then I see this error:**

**Failed to load the native TensorFlow runtime.**

**See** [**https://www.tensorflow.org/install/install\_sources#common\_installation\_problems**](https://www.tensorflow.org/install/install_sources#common_installation_problems)

**(The problem is mainly didn’t use the NvidiaD okcer image!!), doing below is except you don’t run in docker**

* + - * **export LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:/usr/local/cuda-cudnn-5.1/lib64"**

**Then I hit into this issue** [**https://github.com/tensorflow/tensorflow/issues/4078**](https://github.com/tensorflow/tensorflow/issues/4078)

1.$ sudo apt install nvidia-361-dev  
2.$ sudo find /usr/ -name 'libcuda.so.1' (then you will know path of libcuda.so.1)  
3. just copy the libcuda.so.1 to /usr/local/cuda/lib/

sudo cp /usr/lib/i3860linux-gnu/libcuda.so.1 /usr/local/cuda/lib  
actually the *libcuda.so.1* you find is a link file, see:  
[image](https://cloud.githubusercontent.com/assets/8605990/19565023/13815d58-9718-11e6-8179-74433464ee0a.png)

**I backup the original lib(is a file) the /usr/local/cuda-8.0/lib to lib.old**

**Then I copy the sudo /usr/lib/i386-linux-gnu/libcuda.so.1 to /usr/local/cuda-8.0/lib**

**(** **https://github.com/tensorflow/tensorflow/issues/4078)**

**But yet got it work ?? even I also make sure to export LD\_LIBRARY\_PATH=”$LD\_LIBRARY\_PATH:/usr/local/cuda-8.0/lib” and ALL NO USE!!!**

**So follow the AMLWorkBench VM Setup procedure below and it works!!**

**Note**: <https://github.com/Azure/mmlspark> provide a number of deep learning and tools (E.g. pre-process image from OpenCV, Train DNN-based image models on N-Series GPU VM on Azure)for [Apache Spark](https://github.com/apache/spark) including seamless integration of Spark Machine Learning pipelines with [Microsoft Cognitive Toolkit (CNTK)](https://github.com/Microsoft/CNTK) and [OpenCV](http://www.opencv.org/), enabling you to quickly create powerful, highly-scalable predictive and analytical models for large image and text datasets.

import mmlspark

# Initialize CNTKModel and define input and output columns

cntkModel = mmlspark.CNTKModel() \

.setInputCol("images").setOutputCol("output") \

.setModelLocation(modelFile)

# Train on dataset with internal spark pipeline

scoredImages = cntkModel.transform(imagesWithLabels)

additional env setup : <https://docs.microsoft.com/en-us/azure/machine-learning/preview/experimentation-service-configuration>

compling.hss.ntu.edu.sg/hkcancor

1. **So let try docker with another example**, make sure the **.compute** specify **nvidiaDocker: true**

# [Azure ML Workbench Kubernetes Deployment](https://stackoverflow.com/questions/46963846/azure-ml-workbench-kubernetes-deployment-failed)

1. Set CNTK backed for Keras
   1. On PC AMLWorkBench(alone):
      1. Edit the .keras/keras.json
   2. On DLVM(alone):
      1. Edit the .keras/keras.json
   3. AML workbench deploy to DLVM (as container), add below code

import cntk

import os

os.environ['KERAS\_BACKEND'] = 'cntk'

import keras as k

1. Batch AI <https://azure.microsoft.com/en-us/documentation/learning-paths/batch/>

Example :

<https://www.microsoft.com/developerblog/2017/10/24/bird-detection-with-azure-ml-workbench/>

#### Setup AMLWorkBench VM Compute(DLVMDemo) with Tensorflow, docker

* **Setup the compute and run config file:**

**az ml computetarget attach remotedocker --name DLVMDemo --address** dsvmmwr2ii6hetoh4.australiaeast.cloudapp.azure.com **--username wtam –password Wdam1234!!!!**

* **C:\Github-MyWork\TensorflowLearning\tensorflow-tutorial>az ml experiment prepare -c DLVMDemo (till you see below result)**

**RunId: tensorflow-tutorial\_1519202519426**

**Executing user inputs .....**

**===========================**

**Hello!**

**Your environment is now ready.**

**Execution Details**

**=================**

**RunId: tensorflow-tutorial\_1519202519426**

* **Setup the VM Env**

**Edit the .Compute and .runconfig files**

* + 1. **.Compute:**

**baseDockerImage: microsoft/mmlspark:plus-gpu-0.9.9** [**https://hub.docker.com/r/microsoft/mmlspark/tags/**](https://hub.docker.com/r/microsoft/mmlspark/tags/) **or** plus-gpu-0.10.9

**nvidiaDocker: true**

* + 1. **.runconfig: (change PySpark to Python)**

**Framework: Python**

PrepareEnvironment: true

**az ml experiment prepare -c DLVMDemo**

* **az ml experiment submit -c** DLVMDemo **Tensorflow-tutorial.py**
* Not sure why lately I need to do the prepare again after the .compute and .runtime changes
* Note: Or turn the **PrepareEnvironment: true [But its take long time to run as its need to prepare]**

**Note: Since the DSVM/DLVM has small disk size and if not adding new disk when create. You need to delete the previous docker created session**

# Stop docker service

**wtam@DLVMDemo**:/$ sudo service docker stop

# Unmount overlay file system

**wtam@DLVMDemo**:/$ sudo umount /data/docker/overlay2

# Remove docker directory

**wtam@DLVMDemo**:/$ sudo rm -rf /data/docker

# Restart service

**wtam@DLVMDemo**:/$ sudo service docker start

**Re-prepare the ENV: az ml experiment prepare -c DLVMDemo**

**wtam@DLVMDemo**:/data/docker/overlay2$ **sudo rm -r filename**

Then you need to fix the failed to mount copy source

<https://www.manuel-bauer.net/blog/fix-docker-error-failed-to-mount-copy-source>

# Distributed Tensorflow

Reference: <https://www.datacamp.com/community/tutorials/tensorflow-tutorial>

1. Open the Azure ML WorkBench’s tensorflow-tutorial project
   1. I’m using the Python Notebook to simulate 1x ParameterServer and 2 x Workers with Step0, Step1 and Step2. Run all 3 notebooks

ClusterSpec: 'ps' : ['localhost:2222'],'worker' : ['localhost:2223','localhost:2224']

* + 1. Job name, ps is the ParameterServer, task\_index 0 is worker 1 and task\_index 1 is worker 2
    2. A and b is local variable

1. Since Distributed Tensorflow is using gRPC based on HTTP2 for massage passing thru. Its not as efficient as using MPI([Open MPI](https://www.open-mpi.org/), message passing interface) and later on Badiu’s Ring-AllReduce approach to resolve the linear latency form purely MPI approach (each node need to compute its trunk of data and send back to the Chief node, when many nodes network transmission latency became obvious) Here’re some article that talk about

* Badiu fork of tensorflow mpi with its Ring-AllReduce <http://research.baidu.com/bringing-hpc-techniques-deep-learning/> The benefit is each node only need to send it trunk of result to its next node as a Ring structure to reduce to waiting all node transmission that especially in a Ethernet network
* Here’s another article talk about other faster network approach such as InfiniBand/RDMA (remote direct memory access) <https://planspace.org/20170423-tensorflow_as_automatic_mpi/> and also Nvidia has adopted all these and improvement with its Nvidia Colletive Communicatin Library <https://developer.nvidia.com/nccl>

1. **Uber’s team also further move the Badiu’s tenforflow to its project Horovod that make its easy to run from a single GPU to multiple node/GPU** <https://groups.google.com/a/tensorflow.org/forum/#!topic/discuss/YAtcAb9X5ms>
2. Now lets look at the **Horovod Github** <https://github.com/uber/horovod>
   1. Download: <https://www.open-mpi.org/software/ompi/v3.0/>
   2. Install open MPI <https://www.open-mpi.org/faq/?category=building#easy-build>
      1. Use bash to unzip the tar: gunzip -c openmpi-3.0.0.tar.gz | tar xf -
      2. **cd** openmpi-3.0.0
      3. .**/**configure --prefix=**/**usr**/local**
      4. touch aclocal.m4 configure Makefile.am Makefile.in [need this otherwise, make failed with 'aclocal-1.15' is missing on your system.
      5. **make** all **install**
   3. **To use Horovod**, make the following additions to your program:

import horovod.tensorflow as hvd

# Initialize Horovod

hvd.init()

# Pin GPU to be used to process local rank (one GPU per process)

config = tf.ConfigProto()

config.gpu\_options.visible\_device\_list = str(hvd.local\_rank())

# Build model...

loss = ...

opt = tf.train.AdagradOptimizer(0.01 \* hvd.size())

# Add Horovod Distributed Optimizer

opt = hvd.DistributedOptimizer(opt)

# Add hook to broadcast variables from rank 0 to all other processes during

# initialization.

hooks = [hvd.BroadcastGlobalVariablesHook(0)]

# Make training operation

train\_op = opt.minimize(loss)

# Save checkpoints only on worker 0 to prevent other workers from corrupting them.

checkpoint\_dir = '/tmp/train\_logs' if hvd.rank() == 0 else None

# The MonitoredTrainingSession takes care of session initialization,

# restoring from a checkpoint, saving to a checkpoint, and closing when done

# or an error occurs.

with tf.train.MonitoredTrainingSession(checkpoint\_dir=checkpoint\_dir,

config=config,

hooks=hooks) as mon\_sess:

while not mon\_sess.should\_stop():

# Perform synchronous training.

mon\_sess.run(train\_op)

* 1. **Running Horovod**

1. To run on a machine with 4 GPUs: (note: horovod use one GPU per process and no Tensorflow Cluster (PS, Worker define)

$ mpirun -np 4 \

-H localhost:4 \

-bind-to none -map-by slot \

-x NCCL\_DEBUG=INFO -x LD\_LIBRARY\_PATH \

python train.py

1. To run on 4 machines with 4 GPUs each:

$ mpirun -np 16 \

-H server1:4,server2:4,server3:4,server4:4 \

-bind-to none -map-by slot \

-x NCCL\_DEBUG=INFO -x LD\_LIBRARY\_PATH \

python train.py

Horovod supports Keras and regular TensorFlow in similar ways.

Example: <https://github.com/uber/horovod/blob/master/examples/keras_mnist.py>

Let try to run on the localhost to test it. (Perquisite, pip install open mpi and horovod) and since I installed it on Bash so my mpirun will run on Bash

1. Goto root@MSHK5377301C:/mnt/c/Github-MyWork/TensorfloeLearning/tensorflow-tutorial
2. And to mpirun my sample,”Horovod\_StockPredicition-keras.py”
   1. I need to conda install the pandas-datareader: conda install -c anaconda pandas-datareader
   2. pip install fix\_yahoo\_finance
   3. pip install keras
   4. pip install tensorflow & tensorflow-gpu
3. mpirun -np 1 -x NCCL\_DEBUG=INFO python Horovod\_StockPrediction-Keras.py

But I got the Error : ImportError: libcublas.so.9.0: cannot open shared object file: No such file or directory. Don’t wanna bother the fix with the symbolic liunk, path etc. <https://github.com/tensorflow/tensorflow/issues/15604>

[Since I don’t want to manually configure a VM network (not sure even Batch AI DSVM/DLVM has the InfiniBand/RDMA support?) And installing Open MPI taking hours!! So I will try to use Batch AI to see if I can save these steps

1. **Using BatchAI preview (not BatchService! And current provision is using Az CLI) :** <https://docs.microsoft.com/en-us/azure/batch-ai/quickstart-python>
   1. **Az CloudShell**
      1. az provider register -n Microsoft.BatchAI
      2. az provider register -n Microsoft.Batch
      3. <https://github.com/Azure/BatchAI> (<https://github.com/Azure/BatchAI/blob/master/recipes/Horovod/Horovod/cli-instructions.md> )

**Step1: Create ResourceGroup and Storage acct:**

Create a RG :

william@Azure:~$ az group create -n batchaidemo -l eastasia

Create a Storage acct:

william@Azure:~$ az storage account create -n batchaidemostorage --sku Standard\_LRS -l eastasia -g batchaidemo

OR

<https://docs.microsoft.com/en-us/rest/api/batchai/workspaces/create>

Note: current location **use eastus**, eastus2 or westus2

<https://azure.microsoft.com/en-us/global-infrastructure/services/>

**Step2: Deploy the code to Storage**

Create an Azure File Share with horovod\_samples folder and upload tensorflow\_xxxx.py(Horovod\_StockPrediction-Keras.py) into it: I use AzureStorageExplorer as below cli got compilant

william@Azure:~$ az storage share create --name batchaidemofolder --account-name batchaidemostorage

william@Azure:~$ az storage directory create --share-name batchaidemofolder --name horovod\_demo

william@Azure:~$ az storage file upload --share-name batchaidemofolder --source Horovod\_StockPrediction-Keras.py --path horovod\_demo

**Step3: Create the Cluster**

For this recipe we will use a GPU cluster with two nodes (min node = 2 max node = 3) of Standard\_NC6 size (one GPU) with Ubuntu DSVM (UbuntuDSVM) image and Azure File share batchaidemofolder mounted at $AZ\_BATCHAI\_MOUNT\_ROOT/external

az batchai cluster create -l eastus -g batchaidemo --storage-account-name batchaidemostorage -n nc6 -i UbuntuDSVM -s Standard\_NC6 --min 3 --max 4 --afs-name batchaidemofolder --afs-mount-path external -u "wtam" -p "Wdam1234!!!!"

When creating the cluster using the Azure portal, Make sure choos the Custom VM, Microsoft-ad which is the DSVM. If choosing default most of the script below will run will different error or missing liabray etc!!!!!!!!

(Az Cli somehow has issue in accepting the username and pwd, so I need to manually type it instead of copy and paste?? – just remove the empty space at the end on line, after pwd should work!)

**So here’s the workaround !!!**

az batchai cluster create -l eastus -g batchaidemo --storage-account-name batchaidemostorage -n nc6 -i UbuntuDSVM -s Standard\_NC6 --min 2 --max 3 --afs-name batchaidemofolder --afs-mount-path external

[you will see error of asking the username and pwd, repeat the above command them manually type in the username and pwd!!]

1. **Step3: Run a job (**<https://github.com/Azure/BatchAI> )

<https://github.com/uber/horovod/blob/master/docs/running.md>

**Note**: Currently Batch AI has no native support for Horovod framework, but it's easy to run it using Batch AI custom toolkit. Horovod framework [and IntelMPI](https://github.com/Azure/BatchAI/blob/master/recipes/Horovod/Horovod-Infiniband-Benchmark/job.json) (FYI [*https://intothewave.wordpress.com/2014/11/06/which-mpi-implementation/*](https://intothewave.wordpress.com/2014/11/06/which-mpi-implementation/) *Nvidia is more on Open MPI*) will be installed in the container using job preparation command line

The examples below are for Open MPI. Check your MPI documentation for arguments to the mpirun command on your system.

Typically one GPU will be allocated per process, so if a server has 4 GPUs, you would run 4 processes. In Open MPI, the number of processes is specified with the -np flag.

Starting with the Open MPI 3, it's important to add the -bind-to none and -map-by slot arguments. -bind-to none specifies Open MPI to not bind a training process to a single CPU core (which would hurt performance). -map-by slot allows you to have a mixture of different NUMA configurations because the default behavior is to bind to the socket.

With the -x option you can specify (-x NCCL\_DEBUG=INFO) or copy (-x LD\_LIBRARY\_PATH) an environment variable to all the workers.

Note: the LD\_LIBRARY\_PATH shown in DLVM (echo $ LD\_LIBRARY\_PATH and open MPI is installed on DLVM not sure DSVM, so I can run mpirun )

Step1: see Nc6 cluster and 2**->3** nodes provisioned, goto job

Step2: Add a job

JobName: horovod\_stockpredictiion-job#1

NodeCount: 2->**3**

Cluster: nc6\_3nodes

Tool Type: Custom (as I’m going to run Horovod)

Command line: (run 2 process as 2 GPU, each GPU per process)

mpirun -np 3 -H tvm-1783593343\_1-20180406t041438z:1, tvm-1783593343\_2-20180406t041438z:1 -bind-to none -map-by slot -x NCCL\_DEBUG=INFO -x LD\_LIBRARY\_PATH python Horovod\_StockPrediction-Keras.py

Input directory: path is input

Output directory: path is output

However, when using BatchAI to run the Horovod job, the above won’t work, as I passed the incorrect input path which storing the python script and also missing the env preparation which are the pip dependency installation like fix\_yahoo\_finance, keras etc.

**We need to follow this job procedure** <https://github.com/Azure/BatchAI/blob/master/recipes/Horovod/Horovod/cli-instructions.md> and create a **horovod job.json** (the one I created is like is in my Bash: */mnt/c/Github-MyWork/TensorflowLearning/tensorflow-tutorial/ Horovod\_StockPrediction-Keras\_BatchAIconfig.json*) <https://raw.githubusercontent.com/Azure/BatchAI/master/recipes/Horovod/Horovod/job.json> (Horovod\_StockPrediction-Keras\_BatchAIconfig.json then upload to the *batchaidemoStorage’s* ***batchaidemofolder/horovod\_demo*** where the Horovod\_StockPrediction-keras.py reside **using the Azure Storage Explore**)

* + - * **Show the AzureStorage Explorer: batchaidemostorage/FileShares/batchaidemofolder**
        + *See the horovode\_demo has the StockPrediction script, and the conifg.json (for cli submit) is no use as I submit the job using Az portal*

================================================================

Try

JobName: horovod\_stockprediction-job#1

"stdOutErrPathPrefix": "$AZ\_BATCHAI\_MOUNT\_ROOT/external",

Choose custom (for running Horovod framework and add below command line)

**Command line: [Custom]**

**[**[**Use OpenMPI**](https://www.open-mpi.org/faq/?category=tcp)**:** use TCP for MPI communications, but want to restrict it from certain networks, use the btl\_tcp\_if\_include or btl\_tcp\_if\_exclude MCA parameters]

mpirun -mca btl\_tcp\_if\_exclude docker0,lo --allow-run-as-root --hostfile $AZ\_BATCHAI\_MPI\_HOST\_FILE python $AZ\_BATCHAI\_INPUT\_SCRIPTS/Horovod\_StockPrediction-Keras.py

**OR [**[**Use Intel MPI**](https://software.intel.com/en-us/mpi-developer-guide-linux-selecting-fabrics) **( MPSS: Open Fabrics Enterprise Distribution)]**

source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh; mpirun -n 3 -ppn 1 -hosts $AZ\_BATCH\_HOST\_LIST -env I\_MPI\_DEBUG=6 -env I\_MPI\_FALLBACK\_DEVICE 0 -env I\_MPI\_FABRICS=dapl -env I\_MPI\_DAPL\_PROVIDER=ofa-v2-ib0 -env I\_MPI\_DYNAMIC\_CONNECTION=0 python $AZ\_BATCHAI\_INPUT\_SCRIPTS/Horovod\_StockPrediction-Keras.py

[**Note:** when running the job, I encounter Intel MPI environment issue, ([ofed-mic service is not running](http://scc.ustc.edu.cn/zlsc/tc4600/intel/2016.0.109/mpi/User_Guide/Environment_Problems.htm) -  (Open Fabrics Enterprise Distribution, Intel InfiniBand, Manycore Platform Software Stack (MPSS) ) **REFER to JOB9 ERROR LOG**

*[0] MPI startup(): dapl fabric is not available and fallback fabric is not enabled*

*terminate called after throwing an instance of 'std::system\_error'*

*what(): Resource deadlock avoided*

*[0] DAPL startup(): trying to open DAPL provider from I\_MPI\_DAPL\_PROVIDER:* ***ofa-v2-ib0***

*9053997dd5a84486bdd933890e68d594000000:CMA:1139:46c81700: 64 us(64 us): open\_hca: getaddr\_netdev ERROR:No such device. Is ib0 configured?]*

[***http://scc.ustc.edu.cn/zlsc/tc4600/intel/2016.0.109/mpi/User\_Guide/Environment\_Problems.htm***](http://scc.ustc.edu.cn/zlsc/tc4600/intel/2016.0.109/mpi/User_Guide/Environment_Problems.htm)

***Choose custom Note: Don’t choose custom MPI as its has conflicted with Intel MPI, see the*** [horovod\_stockprediction-job2](https://portal.azure.com/) log

***Node count: 3***

***So I try to use the TCP instead: (***[***https://github.com/Azure/BatchAI/blob/master/recipes/Horovod/Horovod-Infiniband-Benchmark/cli-instructions.md***](https://github.com/Azure/BatchAI/blob/master/recipes/Horovod/Horovod-Infiniband-Benchmark/cli-instructions.md) ***)***

source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh; mpirun -n 3 -ppn 1 -hosts $AZ\_BATCH\_HOST\_LIST -**env I\_MPI\_DEBUG=6 -env I\_MPI\_FABRICS=tcp** python $AZ\_BATCHAI\_INPUT\_SCRIPTS/Horovod\_StockPrediction-Keras.py

***For MarkSix***

source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh; mpirun -n 3 -ppn 1 -hosts $AZ\_BATCH\_HOST\_LIST -**env I\_MPI\_DEBUG=6 -env I\_MPI\_FABRICS=tcp** python $AZ\_BATCHAI\_INPUT\_SCRIPTS/Horovod\_MarkSix.py

***Process count: 3***

***File Share:***

*Storage:* ***batchaidemostorageastus***

*Folder:* ***batchaidemofolder***

*Relative mount path:* ***external***

"**inputDirectories**": [{

"**id**": "**SCRIPTS**",

"**path**": **"$AZ\_BATCHAI\_MOUNT\_ROOT/external**/**horovod\_demo**"

}

],

"**outputDirectories**": [

{

"**id**": "**MODEL**",

"pathPrefix": "$AZ\_BATCHAI\_MOUNT\_ROOT/external",

"pathSuffix": "**Models**"

}

]

(Note: Since I didn’t save my trained model so the Model folder under the job is empty even after running the job)

Also set the env prepare command line at the bottom:

apt update; apt install mpi-default-dev mpi-default-bin -y; pip install horovod fix\_yahoo\_finance keras pandas-datareader;

**OR**

bash $AZ\_BATCHAI\_INPUT\_SCRIPTS/Horovod\_StockPrediction\_jobPreparation.sh

???????

**OR *[Use Intel MPI as the Horovod\_StockPrediction\_jobPreparation.sh shell script can’t execute?*]**

apt-get update -y; apt-get install -y -q -o Dpkg::Options::="--force-confold" --no-install-recommends cpio libdapl2 libmlx4-1 libsm6 libxext6 wget git; cd /tmp; **wget 'http://registrationcenter-download.intel.com/akdlm/irc\_nas/tec/11595/l\_mpi\_2017.3.196.tgz'; tar zxvf l\_mpi\_2017.3.196.tgz; sed -i -e 's/^ACCEPT\_EULA=decline/ACCEPT\_EULA=accept/g' /tmp/l\_mpi\_2017.3.196/silent.cfg; sed -i -e 's|^#ACTIVATION\_LICENSE\_FILE=|ACTIVATION\_LICENSE\_FILE=/tmp/l\_mpi\_2017.3.196/USE\_SERVER.lic|g' /tmp/l\_mpi\_2017.3.196/silent.cfg; sed -i -e 's/^ACTIVATION\_TYPE=exist\_lic/ACTIVATION\_TYPE=license\_server/g' /tmp/l\_mpi\_2017.3.196/silent.cfg; cd /tmp/l\_mpi\_2017.3.196; ./install.sh -s silent.cfg; cd ..; rm -rf l\_mpi\_2017.3.196\*; echo "source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh" >> ~/.bashrc; source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh;** **apt-get install libffi-dev**;pip install horovod==0.12.1 fix\_yahoo\_finance==0.0.21 keras==2.1.5 pandas-datareader;

***For MarkSix:***

*apt-get update -y; apt-get install -y -q -o Dpkg::Options::="--force-confold" --no-install-recommends cpio libdapl2 libmlx4-1 libsm6 libxext6 wget git; cd /tmp; wget 'http://registrationcenter-download.intel.com/akdlm/irc\_nas/tec/11595/l\_mpi\_2017.3.196.tgz'; tar zxvf l\_mpi\_2017.3.196.tgz; sed -i -e 's/^ACCEPT\_EULA=decline/ACCEPT\_EULA=accept/g' /tmp/l\_mpi\_2017.3.196/silent.cfg; sed -i -e 's|^#ACTIVATION\_LICENSE\_FILE=|ACTIVATION\_LICENSE\_FILE=/tmp/l\_mpi\_2017.3.196/USE\_SERVER.lic|g' /tmp/l\_mpi\_2017.3.196/silent.cfg; sed -i -e 's/^ACTIVATION\_TYPE=exist\_lic/ACTIVATION\_TYPE=license\_server/g' /tmp/l\_mpi\_2017.3.196/silent.cfg; cd /tmp/l\_mpi\_2017.3.196; ./install.sh -s silent.cfg; cd ..; rm -rf l\_mpi\_2017.3.196\*; echo "source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh" >> ~/.bashrc; source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh; apt-get install libffi-dev;pip install horovod==0.12.1 keras==2.1.5 pandas requests;*

Note: apt-get install libffi-dev is required in this recent BatchAI update(Jul-2018) and I also specified some of the pkg versions due to some compatibility with tensorflow e.g. keras complainslike unexpected keyword argument ……

*Below is the sh script(****Horovod\_StockPrediction\_jobPreparation.sh under batchaidemostorage/batchaidemofolder/horovod\_demo)*** *for reference and future should rum the prepare env in script instead*

apt-get -y -q -o Dpkg::Options::="--force-confold" --no-install-recommends cpio libdapl2 libmlx4-1 libsm6 libxext6 wget git;

cd /tmp;

wget 'http://registrationcenter-download.intel.com/akdlm/irc\_nas/tec/11595/l\_mpi\_2017.3.196.tgz';

tar zxvf l\_mpi\_2017.3.196.tgz;

sed -i -e 's/^ACCEPT\_EULA=decline/ACCEPT\_EULA=accept/g' /tmp/l\_mpi\_2017.3.196/silent.cfg;

sed -i -e 's|^#ACTIVATION\_LICENSE\_FILE=|ACTIVATION\_LICENSE\_FILE=/tmp/l\_mpi\_2017.3.196/USE\_SERVER.lic|g' /tmp/l\_mpi\_2017.3.196/silent.cfg;

sed -i -e 's/^ACTIVATION\_TYPE=exist\_lic/ACTIVATION\_TYPE=license\_server/g' /tmp/l\_mpi\_2017.3.196/silent.cfg;

cd /tmp/l\_mpi\_2017.3.196;

./install.sh -s silent.cfg;

cd ..;

rm -rf l\_mpi\_2017.3.196\*;

echo "source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh" >> ~/.bashrc;

source /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mpi/intel64/bin/mpivars.sh

pip install horovod;

pip install fix\_yahoo\_finance keras pandas-datareader

Enable the container(docler with below specific image path)

"containerSettings": {

"imageSourceRegistry": {

"image url": "tensorflow/tensorflow:1.1.0-gpu"

**OR**

**tensorflow/tensorflow:latest-gpu**

**OR**

**tensorflow/tensorflow:1.4.0-gpu**

}

*Then Run the job…..Show the BatchAI’s nc6 cluster where 3 nodes show from 3 idle to running once the job submitted successfully (in few sec)*

*Then show the job which it’ll auto update the stderr (1st is installation of the dependency…and show the 3 nodes is running the same task)*

OR try this command line,still working on…..

az batchai job create -l eastus -g batchaidemo -n horovod -r nc6 -c Horovod\_StockPrediction-Keras\_BatchAIconfig.json

az batchai job create -l eastus -g batchaidemo -storage-account-name "batchaidemostorage" -n horovod -r nc6 -c "Horovod\_StockPrediction-Keras\_BatchAIconfig.json"

Demo flow

1. Show the MAchineLEarning Experinment(wtamExperinment) inside the MLWorkBechRG
   1. Talk about the AML Workbench download
   2. Talk about creating the Machine Learning Model Management wtamExperinmentModelMgmt (allow to deploy your model and manage on Azure)

**Learn how to configure Machine Learning Compute for Model Management…………..(do it later on how to register the model, then create a docker image for web service and use the docker image to cerate a service….)**

1. Back to my PC’s AML WorkBench:
   1. Talk about AMLWorkBench how to help the data preparation in data cleansing, preparation, renaming etc like using a xls
   2. Show the StockPricePrediction-keras PythonNotebook (that use CNTK as backend run on local machine)
   3. Option: Show the Step0(Parameter Server), Step1(worker0), step2(worker1) on the concept of Distributed Tensorflow on the localhost using different port (Tensorflow cluster)
2. Now show the StockPricePrediction-keras.py now which I can run as a docker on a remote DeepLearning VM. (Show the aml\_config:
3. Show BatchAI using Horovod framework 0 Tensorflow – RING ALLReduce (2 nodes with GPU) [Observation] Result Ref: horovod\_stockprediction-job10
   1. Once the job create (using Intel MPI – TCP) its take 5 mins to start execute
   2. And not sure why each epoch take almost double (46+ sec) as copare to run on a single DSVM?
   3. After scaling the learning rate for the 1st 5 epoch , the remaining epoch take less time but still 40Sec VS the standalone 18+ sec ofr each epoch. In addition using this Intel MPI(TCP) performance is the same as using the Open MPI
   4. Take almost 20 misn to complete the job and the accuracy is worst with 75.82 RMSE. And the price gab is +/- 75…..Totally not acceptable. I suspect its due to Horovod using learning rate = 1.0 x hvd.size9() from the very beginning lead to worst accuracy……..
   5. Readjusting the learning rate to much lower, default 0.001, each epoch time somehow improve back to normal 17-20 sec range. On par with the single node and seems a slightly faster in 1-2 sec. According to Horovod performance chart, its should has significant improvement when its scale to 100+ nodes. Ref result - **horovod\_stockprediction-job11** and the accuracy improve a lot after lower the learning rate, back to RMSE 1.57 [Totally 14+ mins to complete the job]

**DLTK BatchAI Job**

1. **ToolType**: Tensorflow
2. 2->3 node
3. Stdout/stderr Path Prefix: $AZ\_BATCHAI\_MPUNT\_ROOT/external
4. **Python script file path**: $AZ\_BATCHAI\_MOUNT\_ROOT/external/DLTK\_demo/DLTK\_SexClassification.py
5. **Python Interpreter(Note: default is 2.7)**: **/usr/bin/python3.5**

Note: somehow python3 env is faster and has no issue reading the cluster spec from os.environ?

1. **masterCommandLineArgs** : --job\_name=worker --num\_gpus=1 --ps\_hosts=$AZ\_BATCHAI\_PS\_HOSTS --worker\_hosts=$AZ\_BATCHAI\_WORKER\_HOSTS --task\_index=$AZ\_BATCHAI\_TASK\_INDEX --data\_dir=$AZ\_BATCHAI\_INPUT\_DATASET --output\_dir=$AZ\_BATCHAI\_OUTPUT\_MODEL

Since I use Tensorflow Estimator that built-in with Distributed computing and specified in TF\_TFG env variable, so I use this cmd line args instead. Note BactAI env don’t have task.type so I either do all manually like below [Note: I substitute the job\_name to task.type which either master, worker or ps] **Caution**: I tried using the TF\_CONFIG in BatchAI env and due to task.type is missing and I’ve trouble to assign back from cmd line args. Which eventually can’t kick start the job with ERROR!!!!!!!! Due to missing parameter , but the error message is not helpful!!!!!!!!!!!!!!!!!!!!!!

FailedStartingContainer - unable to start docker container  
Details: imageName - tensorflow/tensorflow:1.4.0-gpu

--job\_name=master --num\_gpus=1 --ps\_hosts=$AZ\_BATCHAI\_PS\_HOSTS --worker\_hosts=$AZ\_BATCHAI\_WORKER\_HOSTS --task\_index=$AZ\_BATCHAI\_TASK\_INDEX --cuda\_devices=0

--job\_name=chief --num\_gpus=1 --ps\_hosts=$AZ\_BATCHAI\_PS\_HOSTS --worker\_hosts=$AZ\_BATCHAI\_WORKER\_HOSTS --task\_index=$AZ\_BATCHAI\_TASK\_INDEX --cuda\_devices=0

(note: the AZ\_BATCHAI\_TASK\_INDEX always start with 0 for Master, Worker and PS so if I manually reserve one worker to be chief node I need to deduct the worker task.index to start with 0 again. Refer the task table <https://www.tensorflow.org/api_docs/python/tf/estimator/RunConfig> )

**Option: Test on AMLWorkbench:**

*Run on local and pump up the task idex to 1 as it will deduct 1 from my code for run in cluster*

--job\_name=worker worker\_hosts={"127.0.0.1:2221”} --task\_index=1 --cuda\_devices=0

Note: all env variable ate strings

--cuda\_devices=0 --task\_type=master

1. **workerCommandLineArgs**: --job\_name=worker --num\_gpus=1 --ps\_hosts=$AZ\_BATCHAI\_PS\_HOSTS --worker\_hosts=$AZ\_BATCHAI\_WORKER\_HOSTS --task\_index=$AZ\_BATCHAI\_TASK\_INDEX --data\_dir=$AZ\_BATCHAI\_INPUT\_DATASET --output\_dir=$AZ\_BATCHAI\_OUTPUT\_MODEL

--job\_name=worker --num\_gpus=1 --ps\_hosts=$AZ\_BATCHAI\_PS\_HOSTS --worker\_hosts=$AZ\_BATCHAI\_WORKER\_HOSTS --task\_index=$AZ\_BATCHAI\_TASK\_INDEX --cuda\_devices=0

**Note: Due to I reserved one worker node as Chief, I’ve to manually deduct the** =$AZ\_BATCHAI\_TASK\_INDEX by in 1 in my code to not get task\_index out of range error so its start from 0 again

--cuda\_devices=0 --task\_type="worker"

Note: CUDA\_VISIBLE\_DEVICES=1 Only device 1 will be seen

CUDA\_VISIBLE\_DEVICES=0,1 Devices 0 and 1 will be visible

CUDA\_VISIBLE\_DEVICES="0,1" Same as above, quotation marks are optional

CUDA\_VISIBLE\_DEVICES=0,2,3 Devices 0, 2, 3 will be visible; device 1 is masked

CUDA\_VISIBLE\_DEVICES="" No GPU will be visible...

1. **parameterServerCommandLineArgs**: --job\_name=ps --**num\_gpus=0** --ps\_hosts=$AZ\_BATCHAI\_PS\_HOSTS --worker\_hosts=$AZ\_BATCHAI\_WORKER\_HOSTS --task\_index=$AZ\_BATCHAI\_TASK\_INDEX --data\_dir=$AZ\_BATCHAI\_INPUT\_DATASET --output\_dir=$AZ\_BATCHAI\_OUTPUT\_MODEL

--job\_name=ps --num\_gpus=0 --ps\_hosts=$AZ\_BATCHAI\_PS\_HOSTS --worker\_hosts=$AZ\_BATCHAI\_WORKER\_HOSTS --task\_index=$AZ\_BATCHAI\_TASK\_INDEX --cuda\_devices=''

When I try to run 3 nodes to make sure PS job is run 1st, found autoscale seems not work, got deployment error

--cuda\_devices="" --task\_type="ps"

Choose 3 worker as I used worker 0 as chief node count and 1 PS under Tensorflow

**Now need to add the File share reference:**

Azure File share reference

Model: pick OR custom *(old cluster being migrated to migratedus)*

Account name: batchaidemostorageeastus OR batchaidemostorage (*old cluster being migrated to migratedus)*

~~Azure File url:~~ [~~https://batchaidemostorage.file.core.windows.net/batchaidemofolder~~](https://batchaidemostorage.file.core.windows.net/batchaidemofolder)

Relative mount path: external

~~Acct key: YX+Q2l26ZbyfeBsCs59eDN/cVscKdKu7fHh2HlDZnsOnz1jvXqmkujb6W4U7DqgdHJzFeNR3B2z34ysWbSsOoQ==~~ (*old cluster being migrated to migratedus)*

**Advance setting:**

1. **Worker count** = 2->3 *(included the Chief node as worker0),* **Parameter server count** = 1

Stdout/err Path Prefix: $AZ\_BATCHAI\_MOUNT\_ROOT/external

Input dir: **“SCRIPTS”,**

"**path**": "$AZ\_BATCHAI\_MOUNT\_ROOT/external/**DLTK\_demo**"

}

],

"outputDirectories": [

{

"**id**": "**MODEL**",

"pathPrefix": "$AZ\_BATCHAI\_MOUNT\_ROOT/external",

"pathSuffix": "**Models**"

}

]

Environment Variable:

Ref: <https://www.tensorflow.org/api_docs/python/tf/estimator/train_and_evaluate>

**TF\_CONFIG = "{ 'cluster': { 'ps': $AZ\_BATCHAI\_PS\_HOSTS, 'worker': $AZ\_BATCHAI\_WORKER\_HOSTS }, 'task': { 'index': $AZ\_BATCHAI\_TASK\_INDEX, 'type': '' } }"**

1. **Job preparation** : (install azure-nspkg to resolve ModuleNotFoundError: No module named ‘azure-datalake-store’ <https://github.com/Azure/azure-sdk-for-python/issues/1558> ) & I update below to use **pthon3** as Estimator.**RunConfig** keep complaining the job **task.type is not the cluster spec job(Reason: the tasktype name must match the TF\_CONFIG)** and I don’t have this issue on **AMLWorBench running python3.5**, and I found **BatchAI is running python2.7 instead [Note: And somehow the script run much faster in python3.5 env, in python2.7 end its take almost 10+ mins to get to start the script e.g. enter the adls auth, in 3.6 just about 3+ mins!]**

~~apt-get update; apt-get install -y libffi-dev libssl-dev; apt-get install -y --no-install-recommends apt-utils; apt-get install -y python3-pip; pip3 install -I azure-nspkg;~~ **~~pip3 install SimpleITK dltk azure-datalake-store tensorflow-gpu~~ *~~or tensorflow~~***

**Use this cmd line below instead**

apt-get update; apt-get install -y libffi-dev libssl-dev; apt-get install -y --no-install-recommends apt-utils; apt-get install -y python3-pip; pip3 install -I azure-nspkg; pip3 install SimpleITK dltk azure-datalake-store tensorflow-gpu

**OR use below specific datalakestore version due to lately local path download bug**

apt-get update; apt-get install -y libffi-dev libssl-dev; apt-get install -y --no-install-recommends apt-utils; apt-get install -y python3-pip; pip3 install -I azure-nspkg; pip3 install SimpleITK dltk azure-datalake-store==0.0.22 tensorflow-gpu

help Developement to validate the datalake store fix and use tensorflow 1.8.0 instead of 1.10.0 as 1.10.0 seems cause the chief node pending final loss step see DLTK-job2’s worker0 stderr log vs the ok one DLKT-job4(old migrated workspace)

apt-get update; apt-get install -y libffi-dev libssl-dev; apt-get install -y --no-install-recommends apt-utils; apt-get install -y python3-pip; pip3 install -I azure-nspkg; pip3 install SimpleITK dltk azure-datalake-store==0.0.27 tensorflow-gpu==1.8.0

1. **Container**: Enable

**tensorflow/tensorflow:1.4.0-gpu**

**OR**

**tensorflow/tensorflow:1.7.0-gpu**

**OR**

***nvidia/cuda…..drop this***

**USE Nvidia DGX Container** :<https://github.com/Azure/BatchAI/blob/master/documentation/using-nvidia-dgx-container-registry.md>

server\_url: **nvcr.io**

"image": **nvcr.io/nvidia/tensorflow:17.10**

**Note:** [**watch out without this will cause wrong lookup from docker registry instead**](https://github.com/Azure/BatchAI/commit/91f2aabc37206f555df3bb3d14c6ca051b63f140)

"credentials": {

"username": **$oauthtoken**

"passwordSecretReference": **NnN0aGRvbmV2ZXA3cnZlYXFyNnR2NjNsNzU6ZmY2YmUzMzUtMWUwMS00NzdlLTk0Y2MtNWNmNzAzMmRiNjcw**

1. **Need chief node so I manually take the 1st worker reserved as chief node; otherwise you will receive below error from estimator.runConfig**

**valueerror: if "cluster" is set in tf\_config, it must have one "chief" node??**

**For my case I run 3 nodes, 1st node is reserved for PS and Chief node. 2nd and 3rd node are worker node. NoteL I can’t split the PS node to run standalone**

**Ref:** [**https://www.tensorflow.org/api\_docs/python/tf/estimator/RunConfig**](https://www.tensorflow.org/api_docs/python/tf/estimator/RunConfig)

**Ref:** task type must be set to

run\_config\_lib.TaskType

<https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/estimator/run_config.py>

Error : ValueError: worker is not a valid task\_type in the cluster\_spec:

<tensorflow.python.training.server\_lib.ClusterSpec object at 0x7f7d535cd890>

1st the clusterspec object at 0x7f7……. Can be fixed by switching the BatchAI to run python3 env. Refer the above apt-get install and python env path setting

2nd the worker type can be fixed by defined the same name in TF\_CONFIG **cluster** and **task** which I previously use worker\_hosts in the cluster an => change to worker which being used in the **tasktype**

3rd

Estimator example: <https://github.com/tensorflow/models/tree/master/tutorials/image/cifar10_estimator>

<https://cloud.google.com/blog/big-data/2018/02/easy-distributed-training-with-tensorflow-using-tfestimatortrain-and-evaluate-on-cloud-ml-engine>

[**https://gist.github.com/alsrgv/34a32f30292f4e2c1fa29ec0d65dea26**](https://gist.github.com/alsrgv/34a32f30292f4e2c1fa29ec0d65dea26)

**CIFA Estimator ref:** [**https://github.com/tensorflow/models/tree/master/tutorials/image/cifar10\_estimator**](https://github.com/tensorflow/models/tree/master/tutorials/image/cifar10_estimator)

**Another ref:** [**https://k-d-w.org/blog/103/denoising-autoencoder-as-tensorflow-estimator**](https://k-d-w.org/blog/103/denoising-autoencoder-as-tensorflow-estimator)

**For this DLTK Estimator SexClassification example, the original example won’t able to run in distributed environment even adding TF\_CONFIG Run. As its seems stuck at the generator read\_fn. So I rewrite the read\_fn to generate a dictionary of x: np.image.array and y: np.label.array and input\_fn that take the read\_rn and return individual np.arrary which is [‘x’] and [‘y], and use the tensor\_slice with place\_holder to feed\_dict (get the input pipline and not sure if this the same as traditional dict\_feed? ).**

**Watch out if the batch shape and better use the original input image shape without the batch size and add it after with np.expand\_dim; otherwise will see shape not fit or rank issues. Also watch out if a dict return or miss call the output as dict from return, no clear error message to indicate this fault. Hard to troubleshoot, also referencing this** [**https://medium.com/onfido-tech/higher-level-apis-in-tensorflow-67bfb602e6c0**](https://medium.com/onfido-tech/higher-level-apis-in-tensorflow-67bfb602e6c0)

**as I seems also got stuck at the iterartor in the input fn? Or perhaps stuck after estimator reteurn……As I got 4 instance of create session for ps, chief and worker 0,1. Not sure if this is the correct in distributed training. So I check the Estimator doc again and switching to use the estimator train&eval api instead and also using the NFS from blob to ensure all workers can read write in the same model dir**

**……**

**Finally solve its as** Estimator will automatically create and initialize an initializable iterator when you return a tf.data.Dataset from your input\_fn. This enables you to write the following code, without having to worry about initialization or hooks https://stackoverflow.com/questions/48614529/how-to-use-the-iterator-make-initializable-iterator-of-tensorflow-within-a-in

**Estimator, save you lot of code not only the Cluster spec, also help initialize the local, global variable, Iterator for its sessionhook, and with dataset api its save you the the step to feed\_dict!**

**Run the deploy demo: :\Github-MyWork\TensorflowLearning\BatchAI\_DeepLearningToolKit\_Medical\_IMG**

*Note: thought my modified multi-gpu machine training model(ckpt) is stored in the blob( https://batchaidemostorage.file.core.windows.net/batchaidemofolder/DLTK\_NFS\_10kMaxStep\_Success). Better use the DLTK Github come with save models(ckpt) to run the deploy.py(update the paths and enable GPU with cuda=1) for demo*

* + **I update the code to can test any img stored under the C:\Github-MyWork\TensorflowLearning\BatchAI\_DeepLearningToolKit\_Medical\_IMG\DLTK\_IXI\_Dataset\Data\2mm**
  + **Goto > C:\Github-MyWork\TensorflowLearning\BatchAI\_DeepLearningToolKit\_Medical\_IMG>**
  + **Run> python deploy.py --img IXI012**

**Result:**

deploy.py:30: FutureWarning: Method .as\_matrix will be removed in a future version. Use .values instead.

na\_values=[]).as\_matrix()

Loading from IXI\_HH\_DCGAN\_checkpoint\_folder/IXI\_HH\_sex\_classification\1511133313

found IXI012 from DLTK\_IXI\_Dataset/Data/demographic\_HH.csv

Reading : DLTK\_IXI\_Dataset/Data/2mm/IXI012/T1\_2mm.nii.gz

Predicting........

id=IXI012; pred=Male; true=Male; run time=33.88 s;

accuracy=1.0

**Batch AI potential issue on Tensorflow: no way to define Evaluator Spec as its outside the Cluster and current way in manually reserving the Worker 0 as chief node seems has some issue on chief node in writing the event summary which cause the train\_and\_evaluate call not able to return with latest tensorflow 1.10.0 and comparing the DLTKjob-4 in old cluster and DLTK-job 2 ‘s woker0’s stderr.txt? and other puzzle is how to mointoe with Tensorboard and the log is in Blob?**

**~~Maluuba newsqa (~~**[**~~https://github.com/Maluuba/newsqa~~**](https://github.com/Maluuba/newsqa) **~~)~~**

* **~~Use Bash: cd C:\Github-Source\newsqa~~**
  1. **~~Copy the cnn\_stories.tgz & newsqa-data-v1.csv to C:\Github-Source\newsqa\maluuba\newsqa~~**
  2. **~~Can’t use docker, use manual setup~~**
     1. ~~conda create --name newsqa python=2.7 "pandas>=0.19.2"~~
     2. **~~source~~** ~~activate newsqa && pip install --requirement requirements.txt~~

**~~Need to install cython 1~~~~st~~ ~~:~~** ~~conda install -c conda-forge cython~~

* + 1. ~~python maluuba/newsqa/data\_generator.py~~
    2. ~~download~~ [~~stanford-postagger.jar~~](http://www.java2s.com/Code/Jar/s/Downloadstanfordpostaggerjar.htm)~~,~~ [~~slf4j-api.jar~~](https://www.slf4j.org/download.html) ~~to C:\Github-Source\newsqa\maluuba\newsqa~~
    3. ~~python -m unittest discover .~~
    4. ~~javac yet resolve…………~~ 
       1. ~~sudo apt-get install default-jre~~
       2. ~~sudo apt-get update~~
       3. ~~sudo apt-get install default-jdk~~

~~Above not work!~~

~~curl -s "https://get.sdkman.io" | bash~~

~~source "$HOME/.sdkman/bin/sdkman-init.sh"~~

~~sdk install java~~

~~……………………………….~~

**~~Maluuba qgen-workshop(QuestionGen~~** [**~~https://github.com/Maluuba/qgen-workshop~~**](https://github.com/Maluuba/qgen-workshop) **~~)~~**

1. **~~Copy the glove.6B extracted folder, qgen folder (from C:\Github-Source\qgen-workshop) and the notebook.ipynb (rename to QuestionGen) to C:\Github-MyWork\TensorflowLearning\tensorflow-tutorial and use AML Workbench’s notebook to open it~~**
   1. ~~Bash: goto C:\Github-Source\qgen-workshop~~
   2. ~~conda create --name q-gen python=3.5 h5py numpy pandas scipy~~

**QANet**

**Reference:**

* <https://medium.com/@minsangkim/implementing-question-answering-networks-with-cnns-5ae5f08e312b>
* <https://github.com/NLPLearn/QANet>
* <https://rajpurkar.github.io/SQuAD-explorer/>

**Install Git on DLVMDemo and clone the QANet to the /mnt (extend disk), follow the github procedure and install the depednecy using the conda install ….**

**To run the training or demo**

**cd /mnt/QANet and run source activate tensorflow, then**

* **python config.py --mode train or demo**

**[optional]**

* **tensorboard --logdir=./ then open the browser** [**http://localhost:6006**](http://localhost:6006)

**Demo:** [**http://localhost:8080**](http://localhost:8080) **or** [**http://dsvmmwr2ii6hetoh4.australiaeast.cloudapp.azure.com:8080/**](http://dsvmmwr2ii6hetoh4.australiaeast.cloudapp.azure.com:8080/)

**Try:** [**https://www.msn.com/en-us/news/us/no-state-dislikes-trump’s-job-performance-more-than-mass/ar-BBLWlkG?ocid=spartanntp**](https://www.msn.com/en-us/news/us/no-state-dislikes-trump's-job-performance-more-than-mass/ar-BBLWlkG?ocid=spartanntp)

**Q: Does American like Trump?**

**On DLVM to create a python docker to demo QANet**

**Step 1: Create the Docker Image – qanetdemo-container**

* **cd /mnt/QANet**
* **Create a Dockerfile under /mnt/QANet**

FROM python:3.5

ADD \*.py \*.html /

RUN mkdir /data

ADD data /data

RUN mkdir /datasets

ADD datasets /datasets

RUN mkdir /train

ADD train /train

RUN pip3 install tqdm tensorflow spacy bottle

ENTRYPOINT [“python”, “-u”, “./config.py”] (Note: https://stackoverflow.com/questions/45068455/tensorflow-app-freezes-in-docker-container)

#Expose demo Ports

EXPOSE 8080

* **sudo docker build -t qanetdemo-container .**
* **sudo docker run -p 8080:8080 qanetdemo-container**

**Tensorflow Save Model**

[**https://blog.metaflow.fr/tensorflow-saving-restoring-and-mixing-multiple-models-c4c94d5d7125**](https://blog.metaflow.fr/tensorflow-saving-restoring-and-mixing-multiple-models-c4c94d5d7125)

* The weights filename is as follow: <prefix>-<global\_step>.data-<shard\_index>-of-<number\_of\_shards>.
* **data** files are a lot heavier than the **meta**files which is to be expected as they are containing the weights of our model

Tensorflow GPU container ………………..later

[**How to deploy Machine Learning models with TensorFlow. Part 2 ...**](https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=2ahUKEwjm-Mrt4O7cAhXHxLwKHbNiA68QFjAEegQIBhAB&url=https%3A%2F%2Ftowardsdatascience.com%2Fhow-to-deploy-machine-learning-models-with-tensorflow-part-2-containerize-it-db0ad7ca35a7&usg=AOvVaw05U-7OomU8vzuxLFM34cvO)

[**https://towardsdatascience.com/how-to-deploy-machine-learning-models-with-tensorflow-part-1-make-your-model-ready-for-serving-776a14ec3198**](https://towardsdatascience.com/how-to-deploy-machine-learning-models-with-tensorflow-part-1-make-your-model-ready-for-serving-776a14ec3198)

[**https://towardsdatascience.com/how-to-deploy-machine-learning-models-with-tensorflow-part-2-containerize-it-db0ad7ca35a7**](https://towardsdatascience.com/how-to-deploy-machine-learning-models-with-tensorflow-part-2-containerize-it-db0ad7ca35a7)

[**https://runnable.com/docker/python/dockerize-your-python-application**](https://runnable.com/docker/python/dockerize-your-python-application)

[**https://medium.com/@cswiggz/quick-start-to-tensorflow-in-docker-with-a-gui-39414245251f**](https://medium.com/@cswiggz/quick-start-to-tensorflow-in-docker-with-a-gui-39414245251f)

**Note: if docker build complain device not enough space, try this to free memory (under /data): docker system prune -a [Note!!!!!: in order not to avoid the AMLWorkbench‘s DLVM prepared env(docker), run a job to make the prepared env active so that this cmd won’t delete the prepare env(docker)]**

**Or just remove the image**

**List images: docker images -a**

**Delete the image: docker rmi -f qanetdemo-contiainer**

**Step 2: Deploy the docker to my AKSK8Cluster**

**[logon to DLVM use x-term]**

[**https://kubernetes.io/docs/reference/kubectl/cheatsheet/**](https://kubernetes.io/docs/reference/kubectl/cheatsheet/)

[**https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app**](https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app)

1. Push the docker image to my Azure Container Registry
   1. sudo docker login azcontainerregistrydemo.azurecr.io

username: AzContainerRegistryDemo

password: 8D3IL4MLo0QZpMjMK1=h51jw1ozpN9uN

* 1. sudo docker tag qanetdemo-container azcontainerregistrydemo.azurecr.io/qanetdemo-container
  2. sudo docker push azcontainerregistrydemo.azurecr.io/qanetdemo-container
  3. Use AZ Cloud sell to check “qanetdemo-container” is on the the registry repository:

az acr repository list --name azcontainerregistrydemo --output table

1. [Az Cloud Shell] To configure kubectl to connect to your Kubernetes cluster, AKSK8Cluster

az aks get-credentials -g AKSK8ClusterRG -n AKSK8Cluster

***try***: kubectl get nodes, kubectl cluster-info, kubectl get deployment, kubectl describe deployment, kubectl get services, kubectl describe services/qanetdemo-container

1. Create the yml file (vi qanetdemo-docker-compose.yaml on Az Cloud shell)

apiVersion: apps/v1

kind: Deployment

metadata:

name: qanetdemo-web

spec:

replicas: 2

strategy:

rollingUpdate:

maxSurge: 1

maxUnavailable: 1

minReadySeconds: 5

selector:

matchLabels:

app: qanetdemo-web

template:

metadata:

labels:

app: qanetdemo-web

spec:

selector:

app: qanetdemo-web

containers:

- name: qanetdemo-container

image: azcontainerregistrydemo.azurecr.io/qanetdemo-container

ports:

- containerPort: 8080

resources:

requests:

cpu: 250m

limits:

cpu: 500m

env:

---

apiVersion: v1

kind: Service

metadata:

name: qanetdemo-web

spec:

type: LoadBalancer

ports:

- port: 8080

selector:

app: qanetdemo-web

Then on Az Cloud shell run

kubectl create -f qanetdemo-docker-compose.yaml --validate=false

and you will see

william@Azure:~$ kubectl create -f qanetdemo-docker-compose.yaml --validate=false

deployment.apps/qanetdemo-web created

service/qanetdemo-web created

OR use the cmd based on Kubectl but \*\*not yet work \*\*

kubectl run qanetdemo-web --image=azcontainerregistry.azurecr.ip/qanetdemo-container:latest --port 8080

kubectl expose deployment qanetdemo-web –type=LoadBalancer –port 8080 -target-port 8080

1. **use browser to demo :** [**http://104.43.249.82:8080/**](http://104.43.249.82:8080/)
   1. **kubectl get service , kubectl get pod to show its on AKS**
2. **Alternative, use Azure Web App container using the same qanetdemo-deocker-compose.yaml**

**ERROR - Exception in multi-container config parsing: System.Exception: Invalid config file, must have at least one container in the pod definition..**

**at LWAS.SiteStartInfoRepository.FindWebContainer (LWAS.Kube.PodSpec spec, LWAS.LinuxSiteStartInfo startInfo) [0x0000f] in <029f376c1c6a4bb79892c2f60333c2d8>:0**

**at LWAS.SiteStartInfoRepository.SetupPodSpecForMultiContainerApp (Microsoft.Web.Hosting.StartSiteContext ctx, LWAS.LinuxSiteStartInfo startInfo) [0x0005e] in <029f376c1c6a4bb79892c2f60333c2d8>**