Singularity for Machine Learning Applications – Analysis of Performance Impact

HPEC 2019

Bruce R. Jordan Jr., David Barrett, David Burke, Patrick Jardin, Amelia Littrell, Paul Monticciolo, Michael Newey, Jean Piou, Kara Warner

This material is based upon work supported by the USD NON-LINE under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the USD NON-LINE.

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Agenda

- Background
- Workload Descriptions
- Experimental Setup
- Results
- Conclusions

Background

- Reproducible results and mobility of compute is a common problem for software
- In deep learning applications libraries and dependencies are often:
  - Rapidly developed
  - Tightly coupled
  - Mutual exclusive with other libraries
- Containers seek to address these problems
  - All libraries and dependencies are maintained *with* software
- High Performance Computing (HPC) has unique security requirements
  - The security posture of Docker often prevents its installation – containers run as root

Singularity vs. Docker

- Singularity designed with HPC in mind
- Containers run as the *user* not as root
  – This is different from Docker where the containers run as root
  – No possibility of privilege escalation from the container
- Singularity can execute containers built by Docker
  – Singularity can also build containers

- Singularity deployed at[1]:
  – Texas Advanced Computing Center
  – GSI Helmholtz Center for Heavy Ion Research
  – Oak Ridge Leadership Computing Facility
  – Purdue University
  – National Institutes of Health HPC
  – UFIT Research Computing at the University of Florida
  – San Diego Supercomputing Center
  – Lawrence Berkeley National Laboratory
  – University of Chicago
  – McGill HPC Centre/Calcul Québec
  – Barcelona Supercomputing Center
  – Sandia National Lab
  – Argonne National Lab

Singularity provides the capabilities of containerized software without the security risks of Docker

1. https://sylabs.io/guides/3.3/user-guide/installation.html#singularity-on-a-shared-resource

Agenda

- Background
- Workload Descriptions
- Experimental Setup
- Results
- Conclusions

Workload Description

- **For deep learning there are two primary tasks:**
  - **Model Training – Determining the appropriate weights for the neural network**
  - **Model Inference – Making predictions based on given inputs**
- **For these experiments two Neural Networks are tested**
  - **Large Neural Network (LNN)**
  - **Small Neural Network (SNN)**
- **Training is tested with GPU Acceleration**
  - **Due to computational load training requires GPU**
- **Inference is tested:**
  - **Running on CPU Only**
  - **Running with GPU Acceleration**

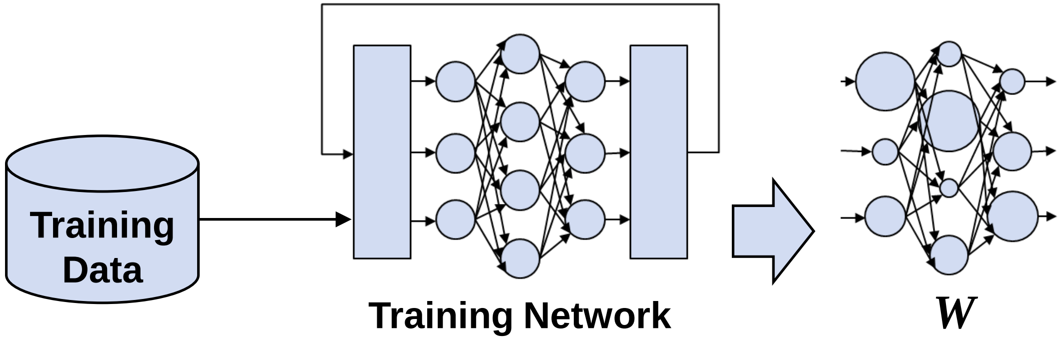
Parameter	LNN	SNN
Total Layers	28	10
3x3 Convolutional Layers	8	3
Dense Layers	4	2
Dropout Layers	6	2

Machine Learning Function Overview

Training

The generation of the model where features are learned from the data

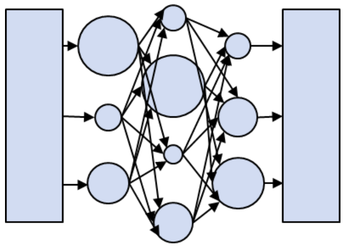
- **Performed infrequently**
- **Non-Realtime**
- **High Computational Load**



Inference

The usage of the trained model to predict the classes of the inputs

- **Performed frequently**
- **Realtime**
- **Moderate Computational Load**



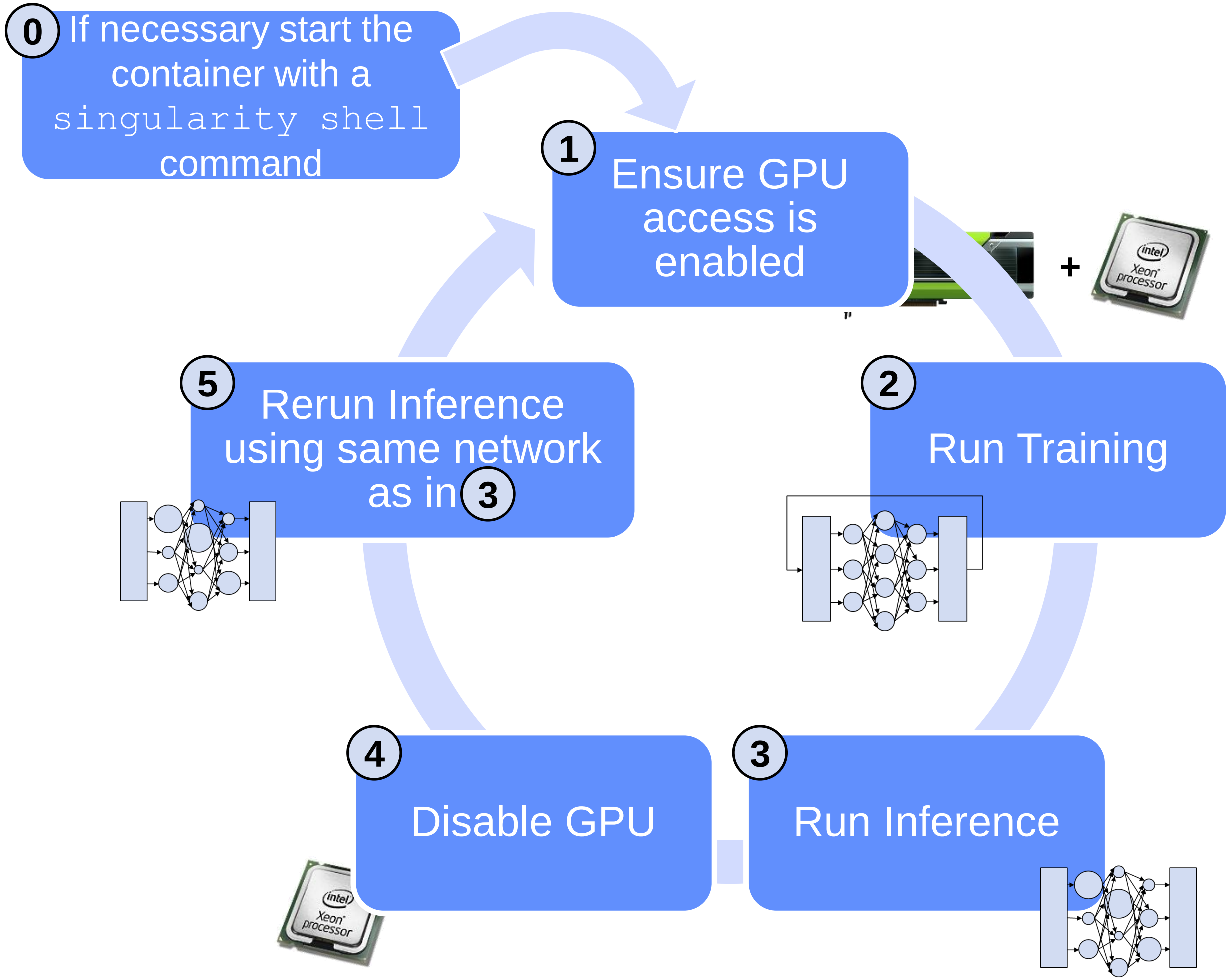
Inference Network with W

Agenda

- Background
- Workload Descriptions
- Experimental Setup
- Results
- Conclusions

Experimental Steps

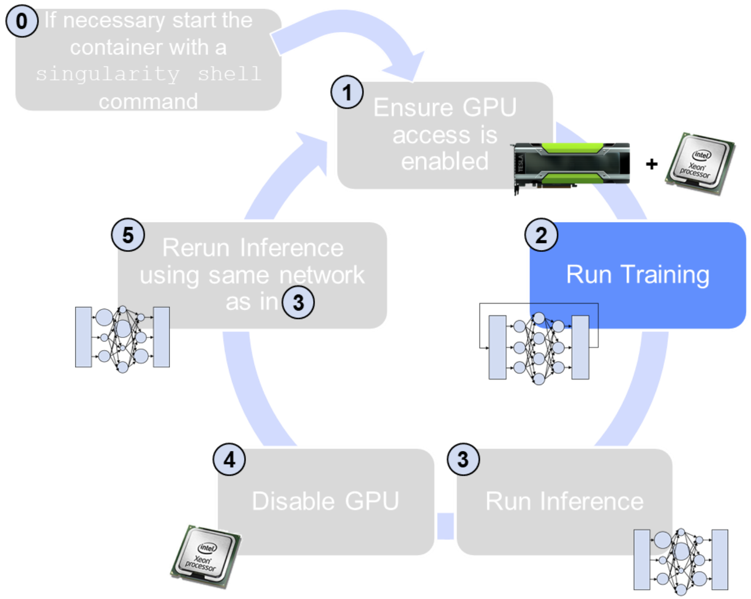
- **Executed on the MIT LLSC[1] which is representative of other HPC Centers**
  - **Using the SLURM scheduler exclusive access to a node was used for all experiments**
- **The hardware used included**
  - **Intel Xeon-E5 Processors**
  - **NVIDIA K80 GPUs**
- **Steps 1-6 automated with bash script**
- **Each of the three workloads was run 100 times both within a container and natively**



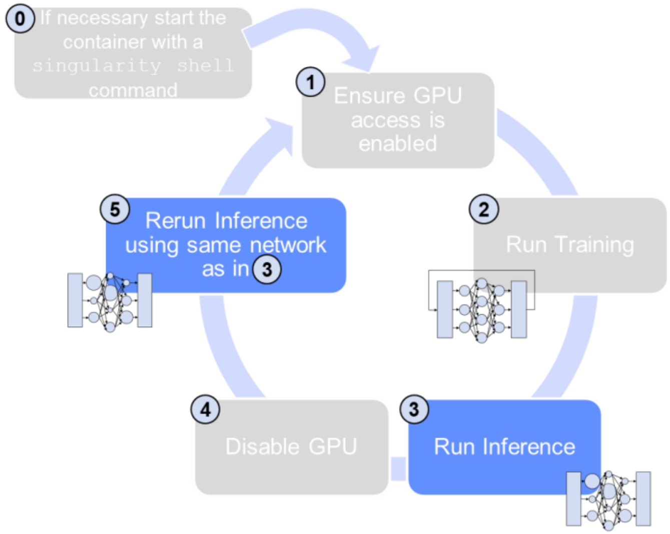
1. Lincoln Laboratory Supercomputing Center

Training Details

- **GPUs used to accelerate the training time**
  - **Training has extremely high computational load – CPU only training would take too long**
- **1526 Images used for training**
  - **256x256 pixels**
  - **2 Channels of grayscale data per image**
  - **5 image classes within the data set**
- **SNN – Used 50 Epochs with a batch size of 301 images**
- **LNN – Used 300 Epochs with a batch size of 301 images**



Inference Details



- **The network that was trained (either natively or within a container) is used**
- **145 images are presented and categorized into one of five output classes**
- **By default the version of Tensorflow will use GPU Accelerated functions**
  – **For CPU Only inference the environment variable `CUDA_VISIBLE_DEVICES` was set to the empty string**
  – **When using CPU Only Tensorflow falls back to non-GPU Accelerated functions**

Agenda

- Background
- Workload Descriptions
- Experimental Setup
- Results
- Conclusions

Resource Utilization

Time

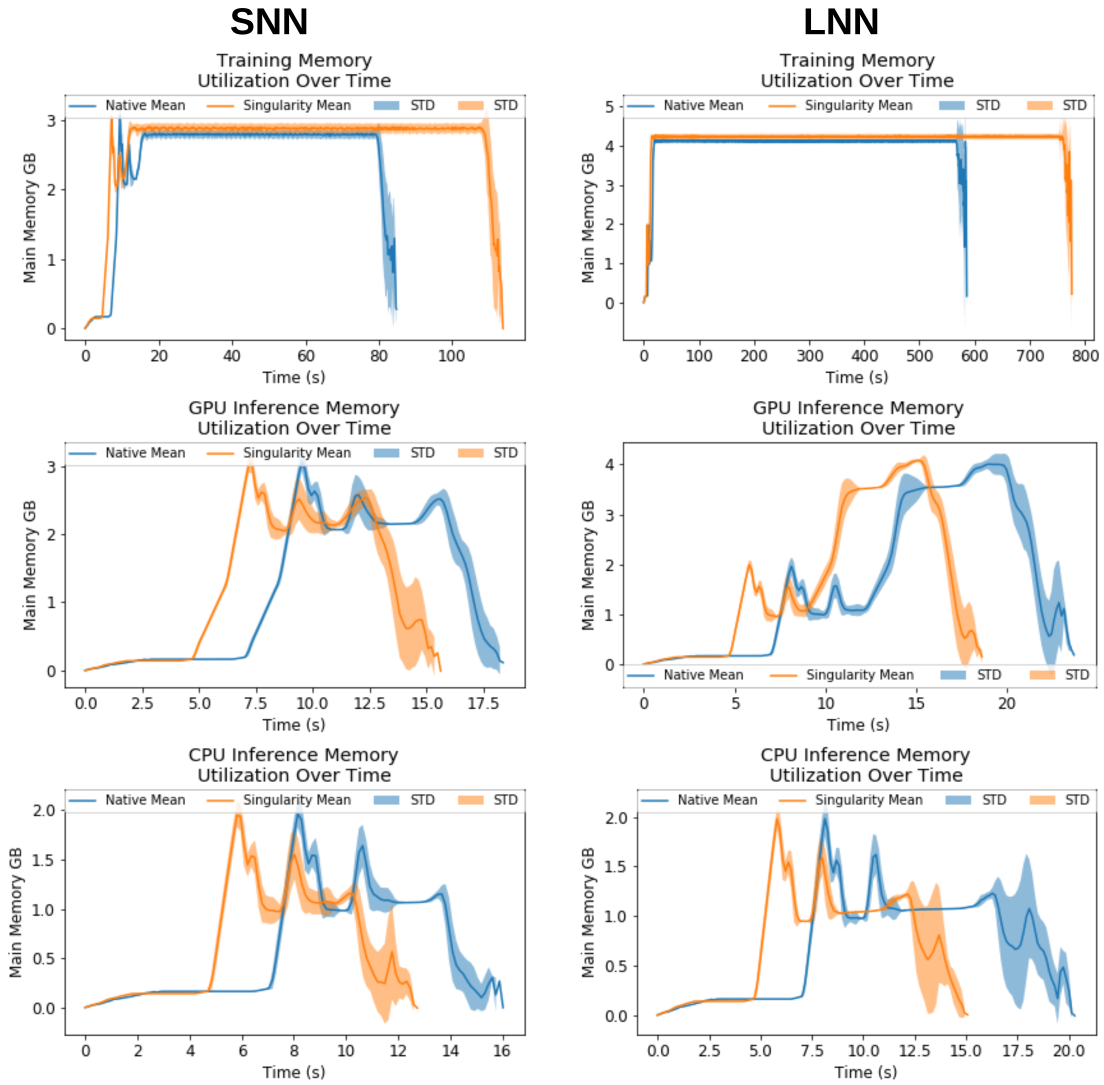
	Mean	Min	Max	STD	*Overhead*
SNN Statistics					
Native Training	82.94	80.70	86.03	0.98	
Sing. Training	110.34	108.44	112.95	0.85	***+33.0%***
Native Inf. (GPU)	17.60	17.03	18.46	0.29	
Sing. Inf. (GPU)	14.16	13.64	15.55	0.38	***-24.4%***
Native Inf. (CPU)	15.07	14.61	16.16	0.26	
Sing. Inf. (CPU)	11.40	10.98	12.70	0.36	***-19.5%***
LNN Statistics					
Native Training	579.93	569.39	590.17	5.19	
Sing. Training	760.52	753.06	771.57	3.43	***+31.1%***
Native Inf. (GPU)	22.42	21.08	23.92	0.45	
Sing. Inf. (GPU)	17.65	17.10	18.73	0.33	***-25.6%***
Native Inf. (CPU)	18.32	17.34	20.45	0.88	
Sing. Inf. (CPU)	13.62	12.87	15.03	0.64	***-21.2%***

- **Time trends are consistent across network sizes**
  - **Training takes longer within a container**
  - **Inference is sped up within a container**
- **Inference speed up is consistent when using CPU only or GPU Accelerated**

Using Singularity containers for inference operations improves runtime lengths

Resource Utilization
Main Memory

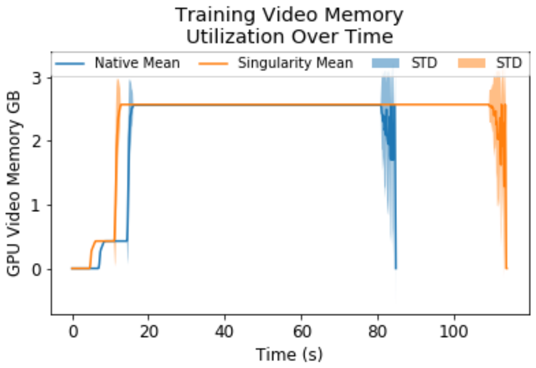
- **Graph Details**
  - **Blue shows Native Utilization**
  - **Orange shows Singularity Utilization**
  - **Dark line is mean for each point**
  - **Shadow is STD for each point**
- **Memory profile shapes are nearly identical except they appear stretched or compressed**
  - **This matches the time utilization**
- **Appears to be slight overhead increase for singularity during training**

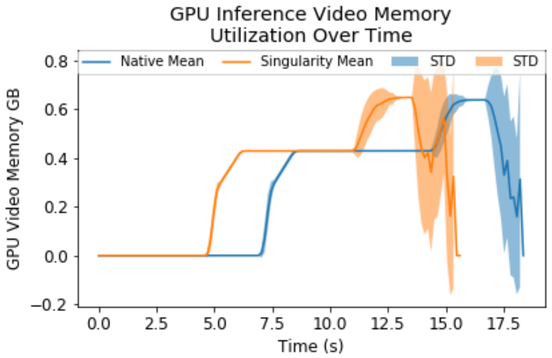


Singularity containers do not appreciably impact memory utilization

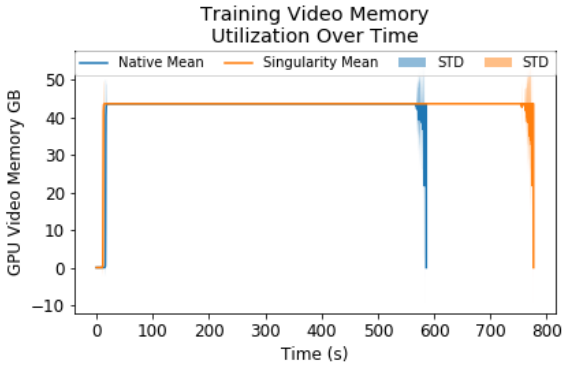
Resource Utilization
GPU Memory

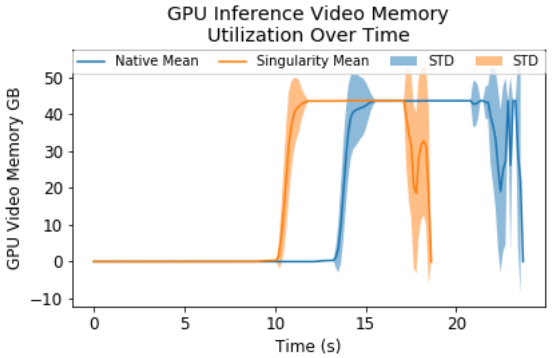
SNN





LNN





- **Graph Details**
  - **Blue shows Native Utilization**
  - **Orange shows Singularity Utilization**
  - **Dark line is mean for each point**
  - **Shadow is STD for each point**
- **As with main memory the shapes are nearly identical except scaled with time**

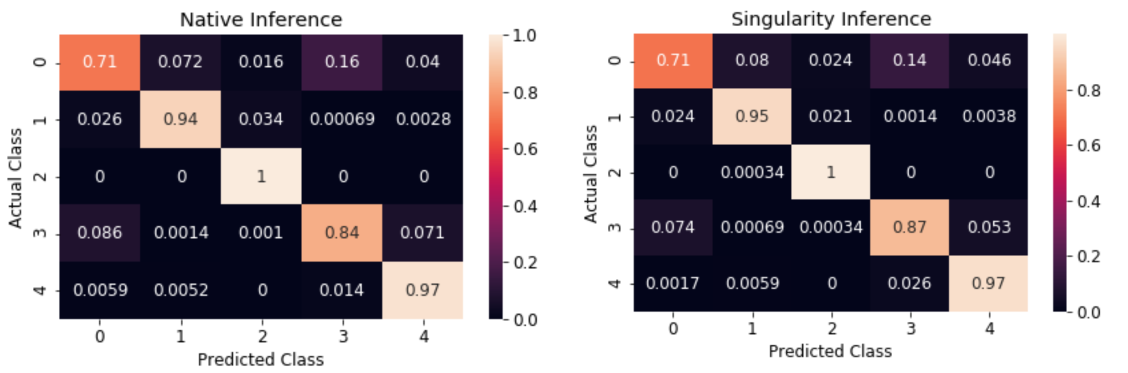
Singularity containers do not appreciably impact video memory utilization

Accuracy Results

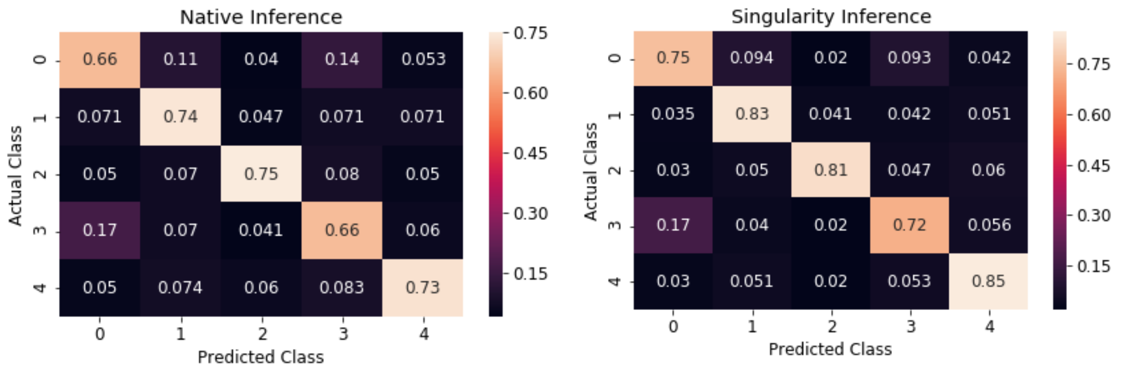
- **Classification Accuracy not impacted by the use of a Singularity Container**
  - **Differences between Native or Containerized are well within a single STD**

	Mean Accuracy	STD	*Difference*
SNN Accuracy			
Native Inference (GPU)	89.3%	5.1%	*-0.3%*
Sing. Inference (GPU)	89.9%	5.1%	*+0.3%*
Native Inference (CPU)	89.3%	5.1%	*-0.3%*
Sing. Inference (CPU)	89.9%	5.1%	*+0.3%*
Combined Mean			*89.9%*
LNN Accuracy			
Native Inference (GPU)	70.9%	33.4%	*-5.5%*
Sing. Inference (GPU)	79.2%	28.3%	*+5.5%*
Native Inference (CPU)	70.9%	33.4%	*-5.5%*
Sing. Inference (CPU)	79.2%	28.3%	*+5.5%*
Combined Mean			*74.8%*

SNN



LNN



Agenda

- Background
- Workload Descriptions
- Experimental Setup
- Results
- Conclusions

Conclusions

- Containers have a number of benefits for development and deployment of software
  - Singularity in particular provides these benefits without the security implications of Docker
- Singularity does not appreciably change the accuracy performance of Deep Learning
  - Differences in performance can be attributed to the stochastic nature of Deep Learning
- Singularity does not appreciably change Memory or GPU Utilization
- Singularity impacts run time lengths differently depending on task
  - For inference tasks Singularity improves run time performance by up to 25%
  - For training tasks Singularity degrades run time performance by up to 33%
  - More research is required to determine underlying reason for difference

Singularity is attractive for deploying containers on HPC or other locations where security prevents the use of Docker