NS&T ML-AI

Big Data Machine Learning Artificial Intelligence

This symposium involves audio/video recording of all presentations, discussions, and comments. The recording for this symposium will be made available to the public. By receiving this notification, your participation consents to recording your interactions with the symposium and public release.

Your audio and video is muted by default

Use the “Chat” feature to ask questions. All questions will be addressed at the end of each presentation (time permitting)

Use the “Chat” feature to let us know if you have technical difficulties

For low-quality connections, switch off video and do not use VPN, if possible

A separate audio PIN will be provided when you sign in for the phone-in audio option

Webinar will begin at 11:00 am MST

Welcome to the

Artificial Intelligence and Machine Learning Symposium 8.0

May 26, 2022

IDAHO NATIONAL LABORATORY

Big Data, Machine Learning, Artificial Intelligence

AI/ML Computational Infrastructure

Agenda – ML/AI Symposium 8.0

May 26, 2022 – 11:00 AM to 1:00 PM MDT

Time	Presentation Subject	Speaker(s)
11:00-11:05	Kickoff for the INL AI/ML 8.0 Symposium	**Ron Boring, INL**
11:05-11:20	Quantum Computing and Machine Learning	**Anand Kiran Shah, Qauntinuum**
11:20-11:35	Using Field Programmable Gate arrays (FPGAs) to accelerate AI/ML Inference Pipelines	**Matt Anderson, INL**
11:35-11:50	Fully on-chip neuromorphic backpropagation	**Andrew Sornborger, LANL**
11:50-12:05	An overview of the GPU hardware and system Conda environments for AI/ML on HPC	**Matt Sgambati, INL**
12:05-12:20	Management of AI/ML Programming Environments	**Brandon Biggs, INL**
12:20-12:35	Jupyter Notebooks - Open OnDemand	**Bradlee Rothwell, INL**
12:35-12:50	HPC Storage	**Shane Grover, INL**
12:50-1:00	A preview on the INL AI/ML Summer 2022 Symposium (S22S)	**Shad Staples, INL**

NS&T ML-AI

Big Data Machine Learning Artificial Intelligence

Welcome

Ronald Boring, PhD, FHFES

Manager, Human Factors and Reliability Department
Idaho National Laboratory

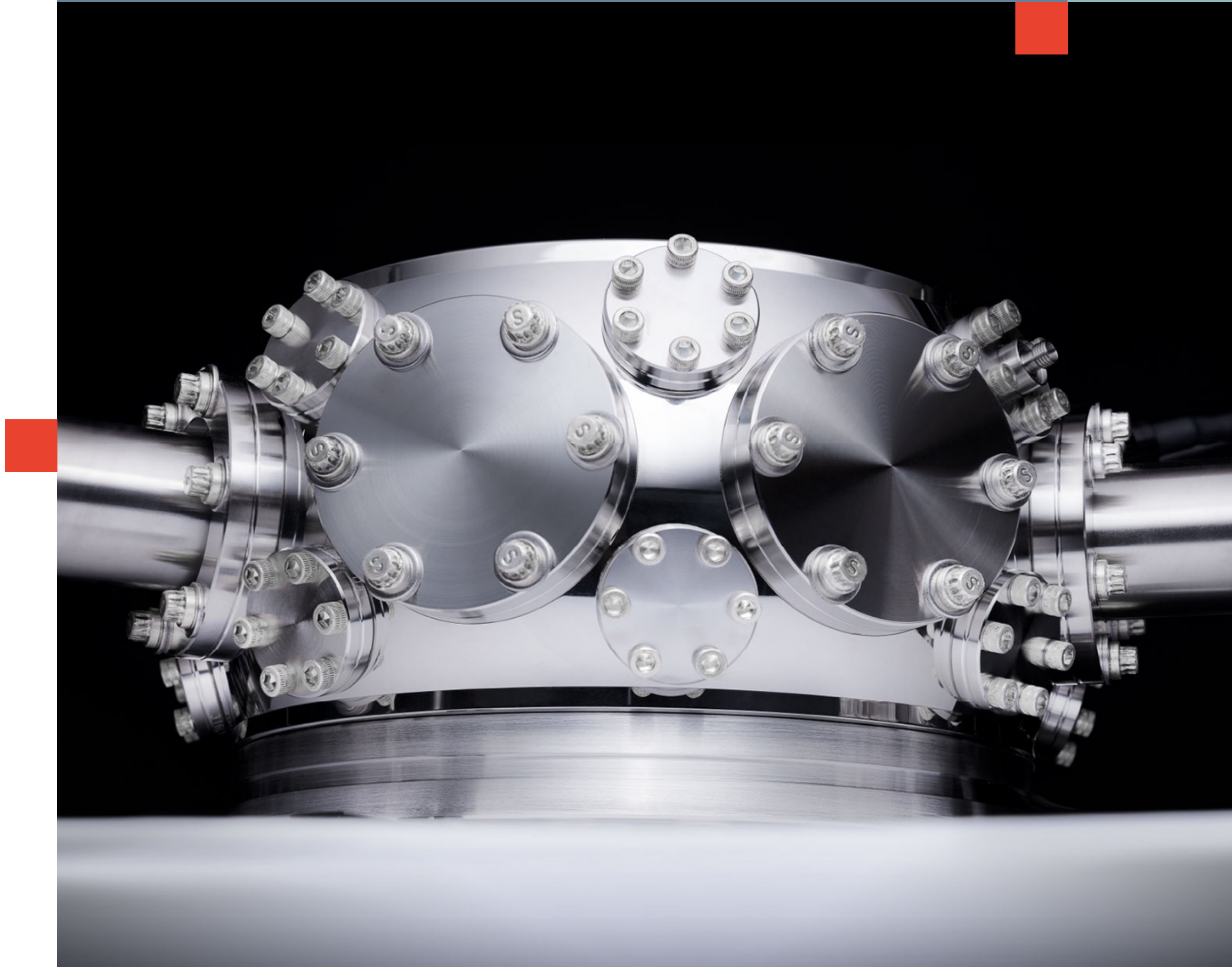
QUANTINUUM

QUANTUM MACHINE LEARNING

PRESENTED BY:

Anand Shah

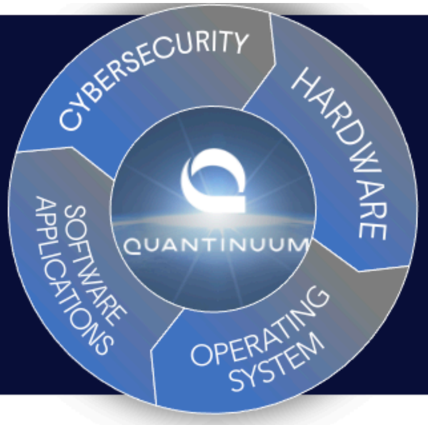
MAY 26, 2022



WHO WE ARE

Cambridge Quantum

Leader in Quantum Computing Software



Honeywell

Quantum Solutions

Leading Quantum Computing Hardware

GLOBAL PRESENCE

Germany, Japan, United Kingdom, United States, adding location in France

400 employees – 300+ Scientists and Engineers

© 2022 by Quantinuum. All rights reserved.

MACHINE LEARNING

DATA

MODELS

TRAINING

QUANTUM MACHINE LEARNING

DATA

MODELS

TRAINING

Using quantum data with classical or quantum ML models for more accurate predictions of quantum systems

Faces a data-loading challenge

QUANTUM MACHINE LEARNING

DATA

MODELS

TRAINING

Using quantum data with classical or quantum ML models for more accurate predictions of quantum systems

Faces a data-loading challenge

Either polynomial speedups based on faster searching of unstructured databases OR exponential speedups for performing faster linear algebra

Requires fault-tolerance

QUANTUM MACHINE LEARNING

DATA

Using quantum data with classical or quantum ML models for more accurate predictions of quantum systems

Faces a data-loading challenge

MODELS

Quantum ML models based on parameterized quantum circuits (PQCs) are more “expressive”

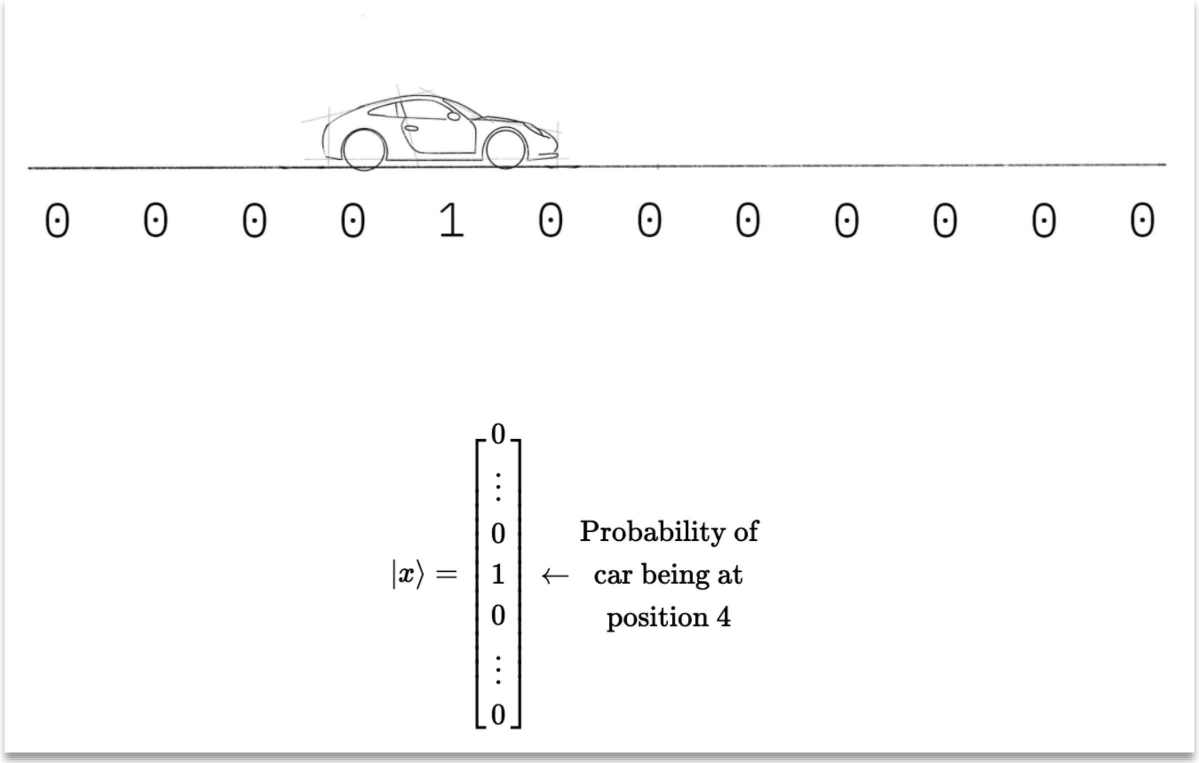
Model and sample from probability distributions that are classically intractable

TRAINING

Either polynomial speedups based on faster searching of unstructured databases OR exponential speedups for performing faster linear algebra

Requires fault-tolerance

QUANTUM 101 – QUBIT STATES



0 0 0 0 1 0 0 0 0 0 0 0

$$|x\rangle = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \begin{array}{c} \text{Probability of} \\ \text{car being at} \\ \text{position 4} \end{array}$$

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

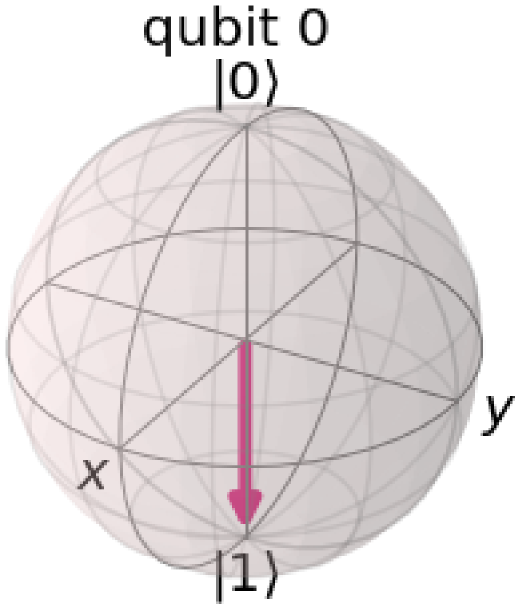
Since |0> and |1> form an orthonormal basis, we can represent any 2D vector with a linear combination of these two states. For example:

$$|q_0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix} \qquad |q_0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$$

QUANTUM 101 – SINGLE QUBIT GATES

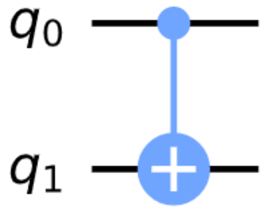
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$



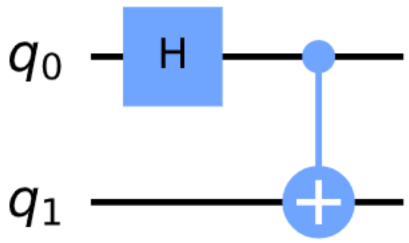
QUANTUM 101 – MULTIPLE QUBITS & ENTANGLEMENT

CNOT Gate



Input (t,c)	Output (t,c)
00	00
01	11
10	10
11	01

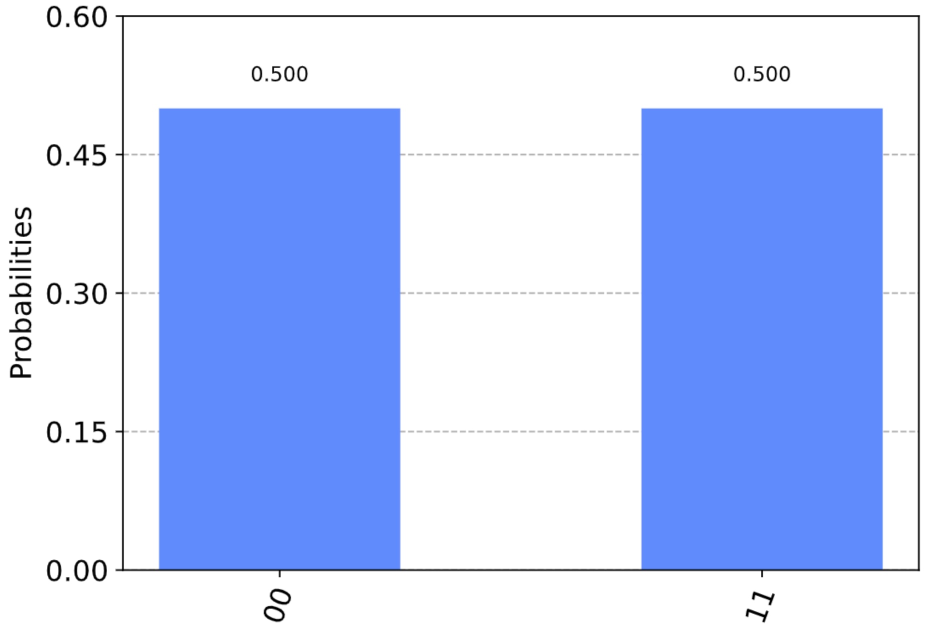
Hadamard + CNOT Gate



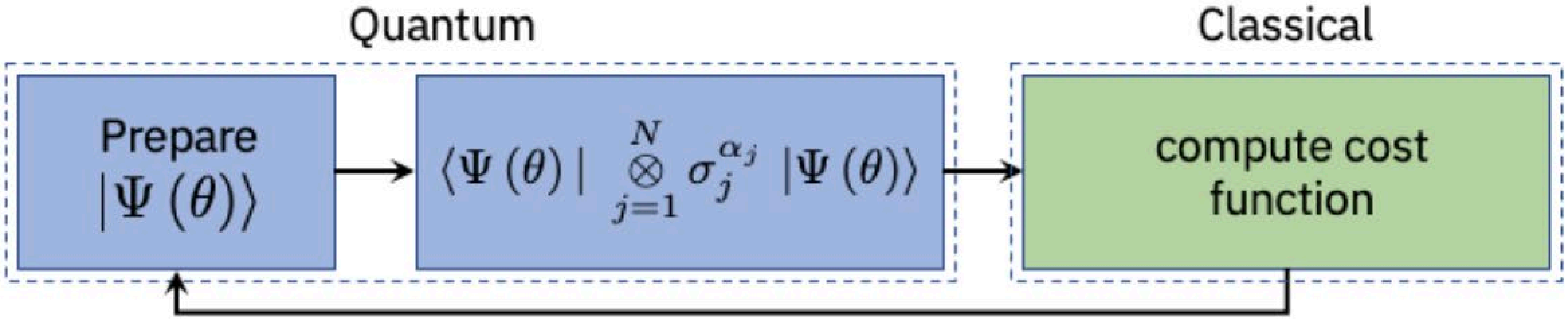
Hadamard: $|0+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$

$$\mathrm{CNOT}|0+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Bell State!



VARIATIONAL ALGORITHMS AS QML MODELS



Quantum

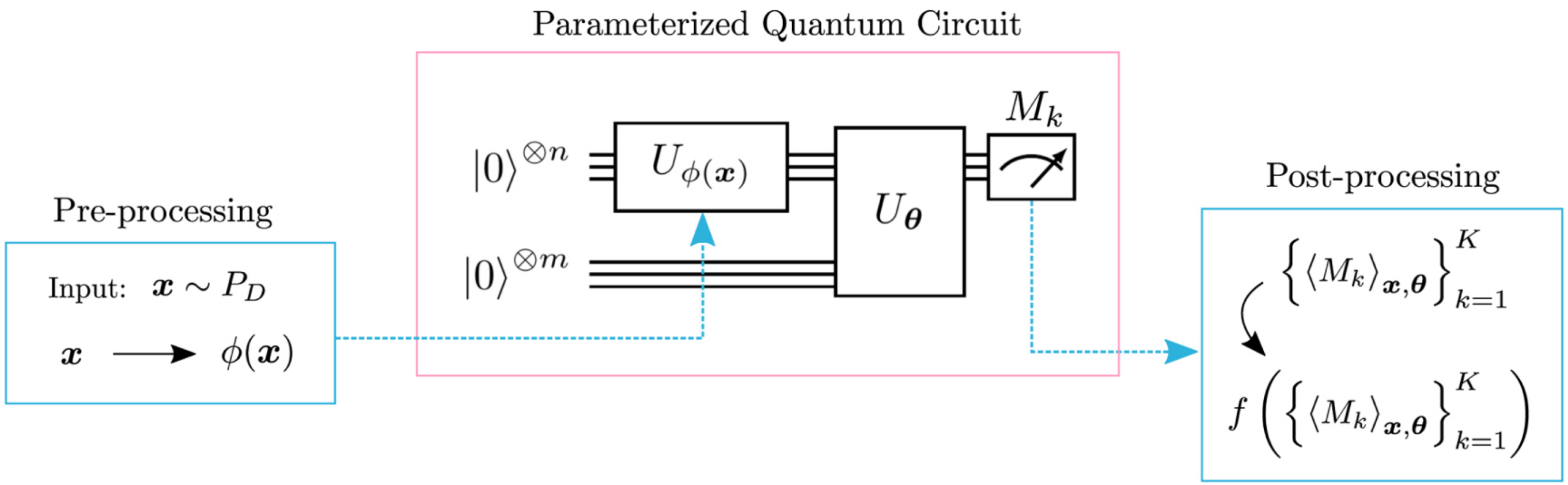
Classical

Prepare $|\Psi(\theta)\rangle$

$\langle\Psi(\theta)|\bigotimes_{j=1}^{N}\sigma_j^{\alpha_j}|\Psi(\theta)\rangle$

compute cost function

VARIATIONAL ALGORITHMS AS QML MODELS

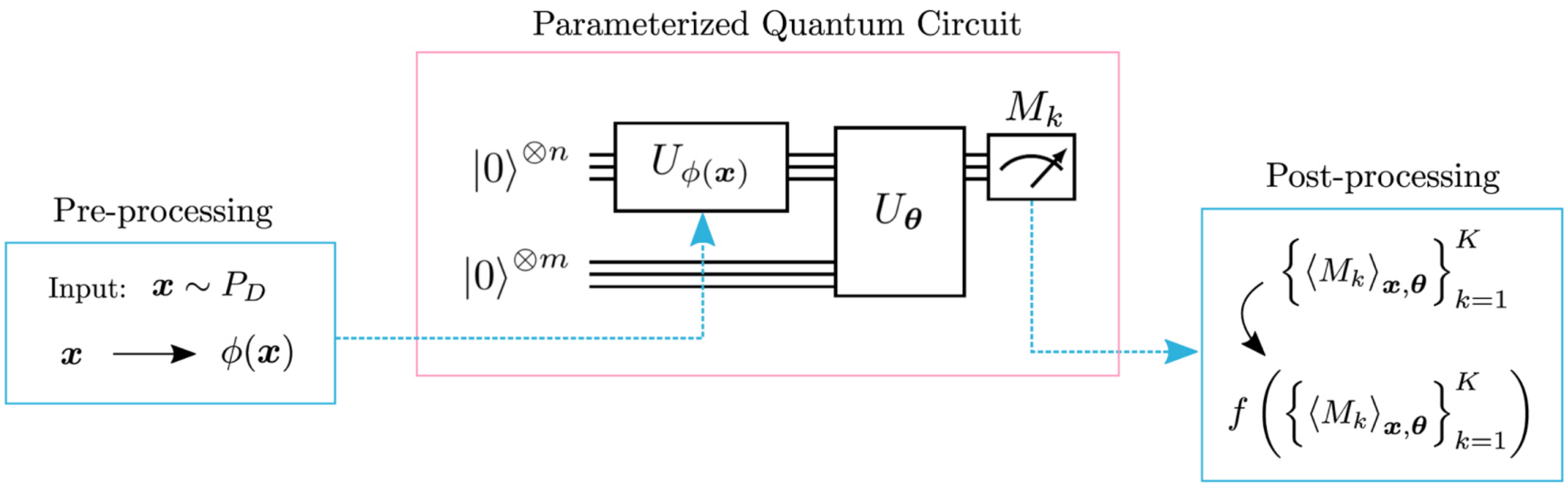

Parameterized Quantum Circuit
Pre-processing
Input: $x \sim P_D$
$x \longrightarrow \phi(x)$
$|0\rangle^{\otimes n}$
$|0\rangle^{\otimes m}$
$U_{\phi(x)}$
U_{θ}
M_k
Post-processing
$\left\{\langle M_k\rangle_{x,\theta}\right\}_{k=1}^{K}$
$f\left(\left\{\langle M_k\rangle_{x,\theta}\right\}_{k=1}^{K}\right)$

On QPU:

Create a quantum state, effectively a probability distribution, using some parameterized rotation gates

Make measurement in some basis on each qubit which returns a bit string (0s and 1s)

VARIATIONAL ALGORITHMS AS QML MODELS



Parameterized Quantum Circuit
$|0\rangle^{\otimes n}$
$U_{\phi(\boldsymbol{x})}$
$|0\rangle^{\otimes m}$
$U_{\boldsymbol{\theta}}$
M_k
Pre-processing
Input: $\boldsymbol{x} \sim P_D$
$\boldsymbol{x} \longrightarrow \phi(\boldsymbol{x})$
Post-processing
$\left\{\langle M_k\rangle_{\boldsymbol{x},\boldsymbol{\theta}}\right\}_{k=1}^{K}$
$f\left(\left\{\langle M_k\rangle_{\boldsymbol{x},\boldsymbol{\theta}}\right\}_{k=1}^{K}\right)$

On QPU:

Create a quantum state, effectively a probability distribution, using some parameterized rotation gates

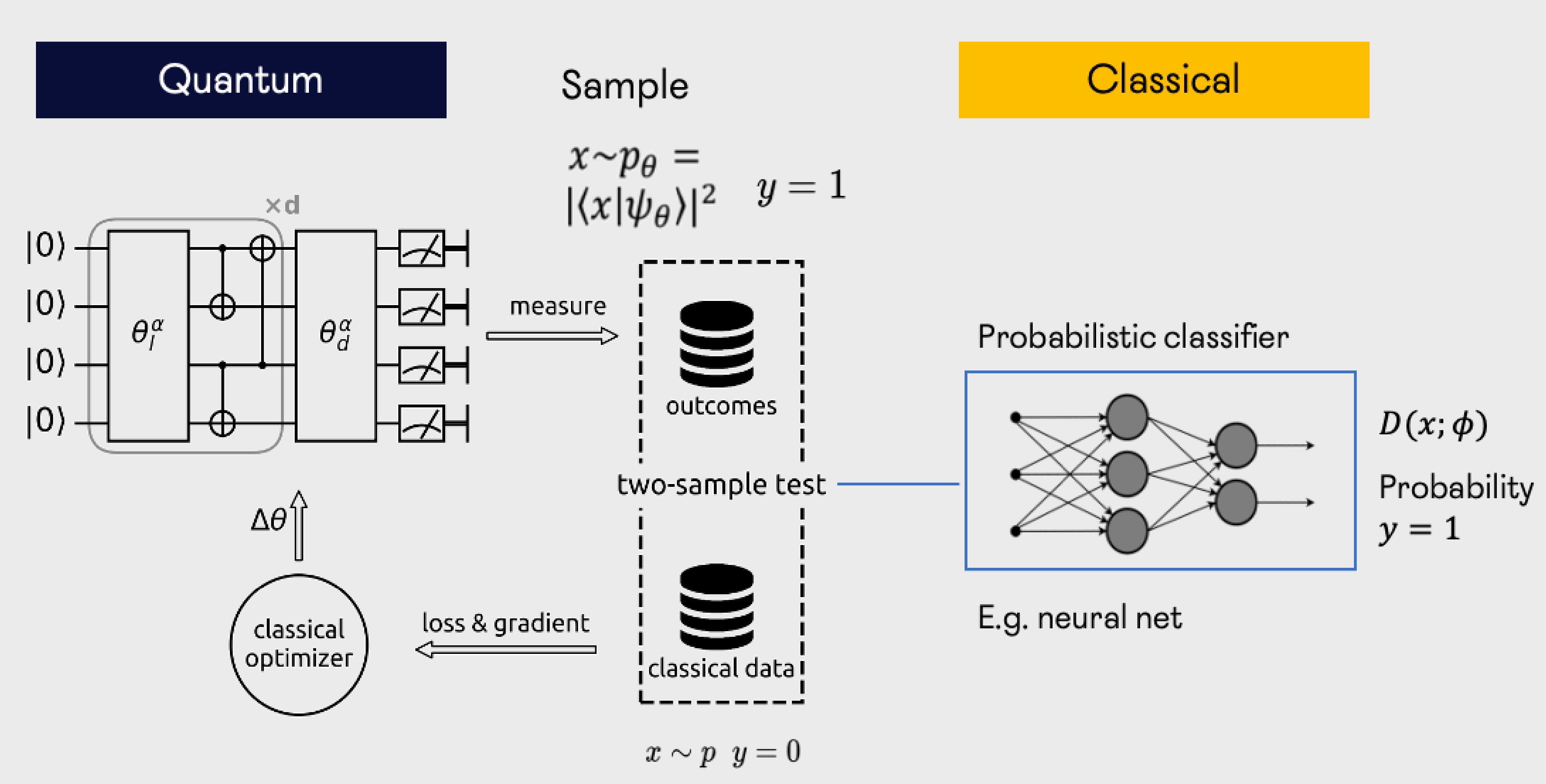
Make measurement in some basis on each qubit which returns a bit string (0s and 1s)

On CPU:

Given a bit string, calculate the energy of the system, i.e., the cost function

Perform optimization procedure if not at minimum and calculate updated parameters

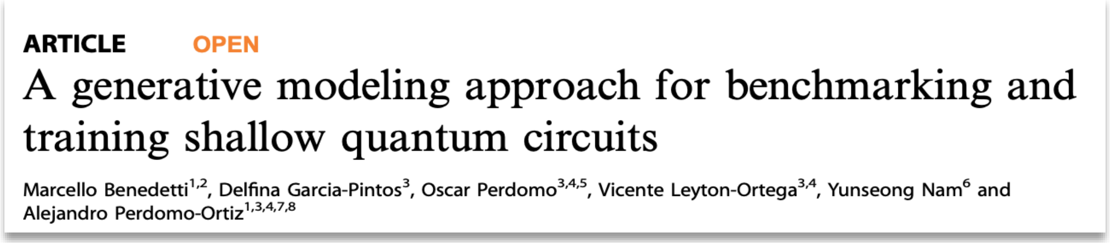
FINDING NEAR-TERM ADVANTAGE



- Generating a complex probability distribution and sampling from it **is classically hard** → quantum advantage

- Useful for unsupervised ML, generative models, Bayesian inference, anomaly detection, etc.

GENERATIVE MODELING AND ANOMALY DETECTION

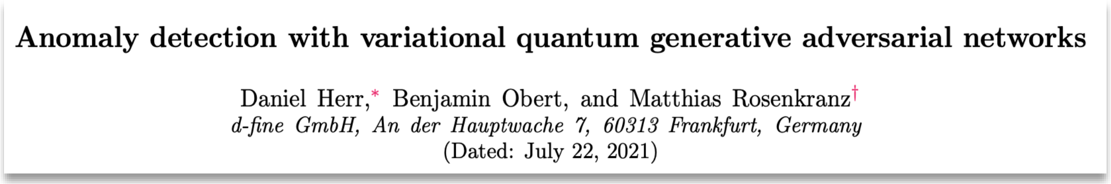


ARTICLE OPEN

A generative modeling approach for benchmarking and training shallow quantum circuits

Marcello Benedetti[1,2], Delfina Garcia-Pintos[3], Oscar Perdomo[3,4,5], Vicente Leyton-Ortega[3,4], Yunseong Nam[6] and Alejandro Perdomo-Ortiz[1,3,4,7,8]

https://doi.org/10.1038/s41534-019-0157-8

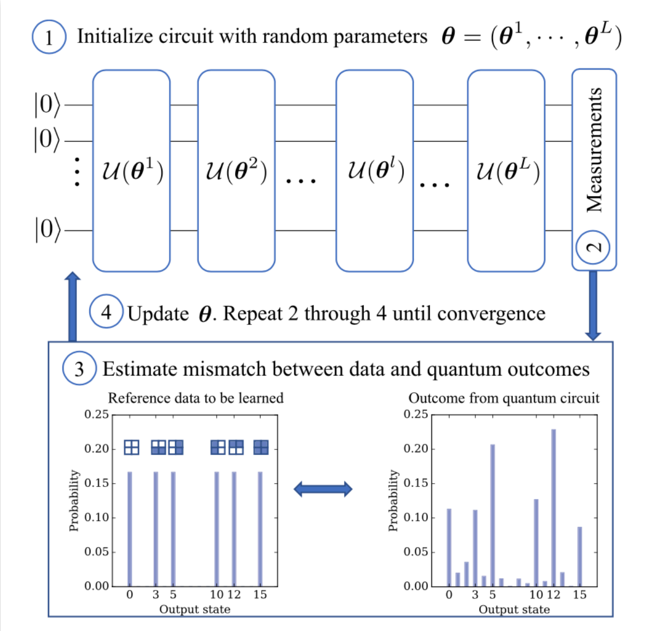


Anomaly detection with variational quantum generative adversarial networks

Daniel Herr,* Benjamin Obert, and Matthias Rosenkranz†
d-fine GmbH, An der Hauptwache 7, 60313 Frankfurt, Germany

(Dated: July 22, 2021)

https://arxiv.org/pdf/2010.10492.pdf

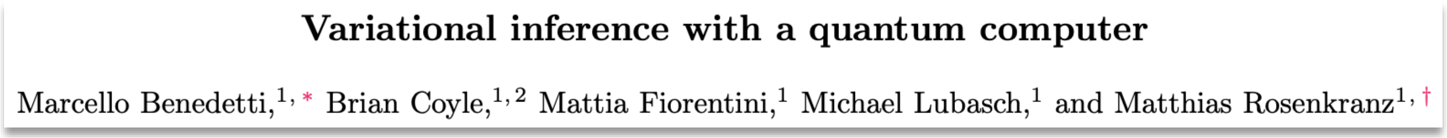

1 Initialize circuit with random parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}^1, \cdots, \boldsymbol{\theta}^L)$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$\mathcal{U}(\boldsymbol{\theta}^1)$
$\mathcal{U}(\boldsymbol{\theta}^2)$
$\mathcal{U}(\boldsymbol{\theta}^l)$
$\mathcal{U}(\boldsymbol{\theta}^L)$
Measurements
2
4 Update $\boldsymbol{\theta}$. Repeat 2 through 4 until convergence
3 Estimate mismatch between data and quantum outcomes
Reference data to be learned
Outcome from quantum circuit
Probability
0.00
0.05
0.10
0.15
0.20
0.25
0
3
5
10
12
15
Output state

- More effectively learn probability distributions to generate accurate synthetic data
- Recognize patterns and detect anomalies effectively leveraging qGANs

PROBABILISTIC REASONING

- **Inference is classically hard, even approximate inference is NP-hard, especially with discrete variables →** quantum advantage

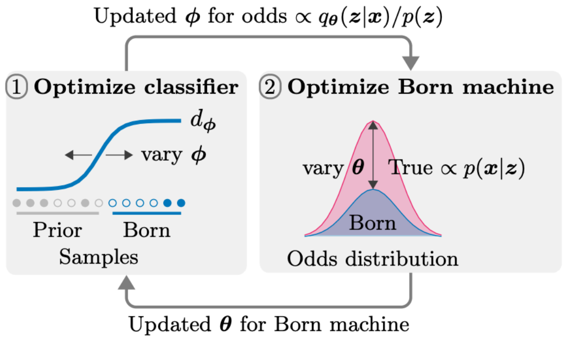
- In March 2021, we published a seminal paper describing two novel quantum algorithms for performing variational inference on quantum computers



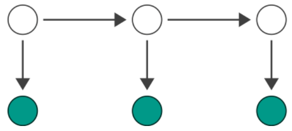
Variational inference with a quantum computer

Marcello Benedetti,[1,*] Brian Coyle,[1,2] Mattia Fiorentini,[1] Michael Lubasch,[1] and Matthias Rosenkranz[1,†]

https://arxiv.org/pdf/2103.06720.pdf



Two novel quantum algorithms enabling near-term quantum computers to reason under uncertainty



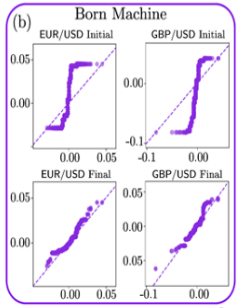
Financial decision system

Hidden: e.g. market regime

Observed: stock market returns

EXAMPLE APPLICATIONS

Correlation Modeling

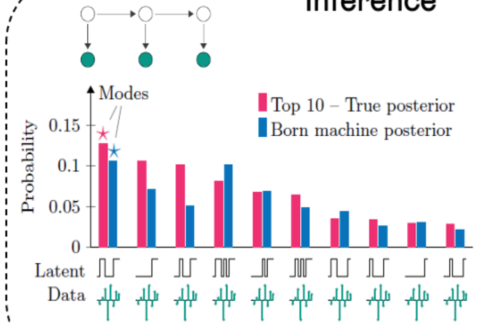


FX spot return modelling

Default correlation

B. Coyle et al., Quantum Sci. Technol. 6, 024013 (2021)

Inference



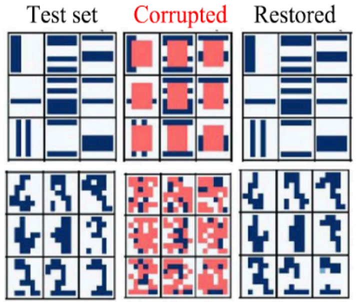
Variational inference

Regime switching

Financial Time Series

M. Benedetti et al., arXiv:2103.06720

Generative Modeling



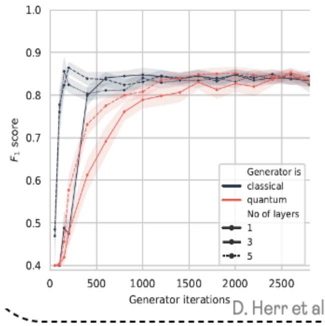
Synthetic data generation

Data recovery – missing time series data

Data augmentation

M. Benedetti et al., Phys. Rev. X 7, 041052 (2017)

Anomaly detection



Fraud detection

Novelty detection

Consumer behaviour

D. Herr et al., Quantum Sci. Technol. 6, 045004 (2021)

Two key concepts for finding advantage:

- Leverage probabilistic nature of quantum computers
- Focus on highly-correlated and complex datasets

QUANTINUUM

ACCELERATING QUANTUM COMPUTING

INL/MIS-22-67261

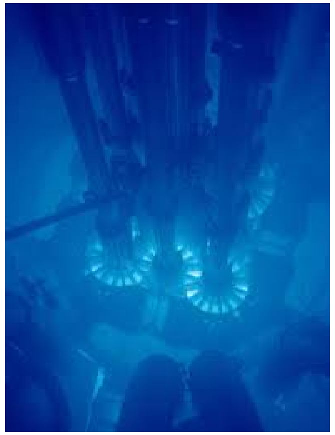
Special Purpose Devices for Inference

Using Field Programmable Gate Arrays (FPGAs) to accelerate AI/ML Inference Pipelines

Matthew Anderson

26 May 2022

When would you need an FPGA for ML inference?



Running ML in a radiation environment

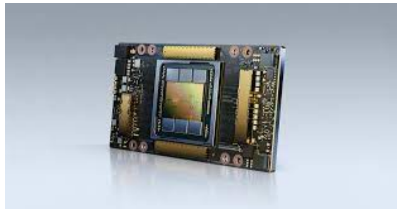
Operating in a power-constrained environment

Running ML at the edge

Running ML where functional safety certifications are needed

Running ML where data requirements need inference 10x faster than GPU

Performance Metrics on Inference Hardware Available at INL HPC



Model	NVIDIA V100	NVIDIA A100	ZCU104	VCK190
3 Resnet block	12k FPS	14k FPS	8k FPS	21k FPS
4 layer CNN	14k FPS	18k FPS	8k FPS	21k FPS
Autoencoder	14k FPS	21k FPS	8k FPS	24k FPS

Model to FPGA implementation:

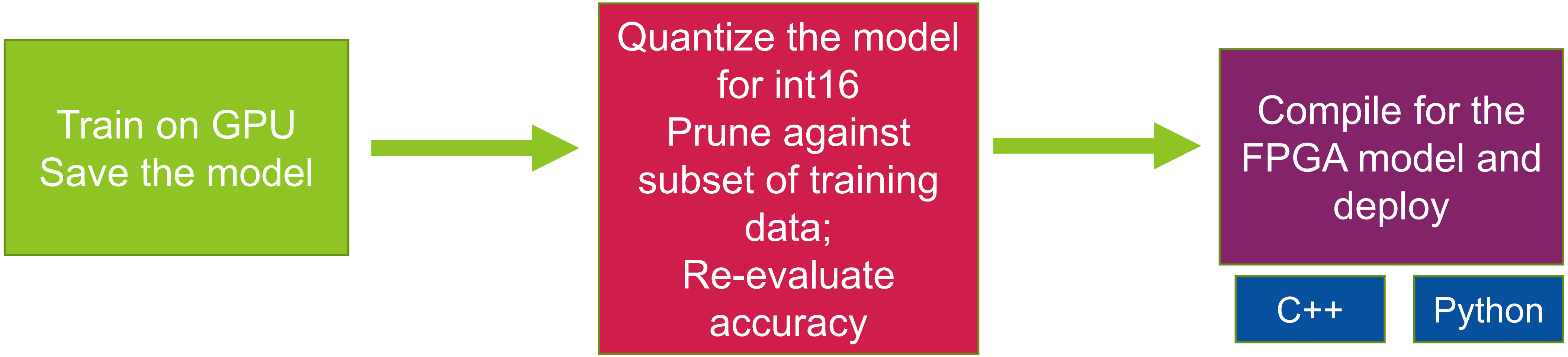
Supported frameworks:

PyTorch
Tensorflow 1, 2
Neptune
Caffe

Accuracy loss in moving to FPGA: ~2%

Petalinux images ready for stand-alone deployment

Power requirement: ~6 Watts



Train on GPU
Save the model

Quantize the model for int16
Prune against subset of training data;
Re-evaluate accuracy

Compile for the FPGA model and deploy

C++

Python

Questions?

Los Alamos
NATIONAL LABORATORY

Fully On-Chip Neuromorphic Backpropagation

Alpha Renner, Forrest Sheldon, Anatoly Zlotnik, Louis Tao, Andrew Sornborger

INRC Seminar, June 16, 2021

LDRD LABORATORY DIRECTED RESEARCH & DEVELOPMENT

U.S. DEPARTMENT OF ENERGY | Office of Science

NNSA National Nuclear Security Administration

University of Zurich UZH

ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

LA-UR-22-24625

Background – Backpropagation Algorithm

Backprop is used as a function approximator for reinforcement learning

Superhuman performance at Atari

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." *Nature* 518, no. 7540 (2015): 529-533.

Superhuman performance at Go

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529, no. 7587 (2016): 484.

Grandmaster performance at Star Craft II

Vinyals, Oriol, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." *Nature* 575, no. 7782 (2019): 350-354.

Results Preview: MNIST

Validation – 96%

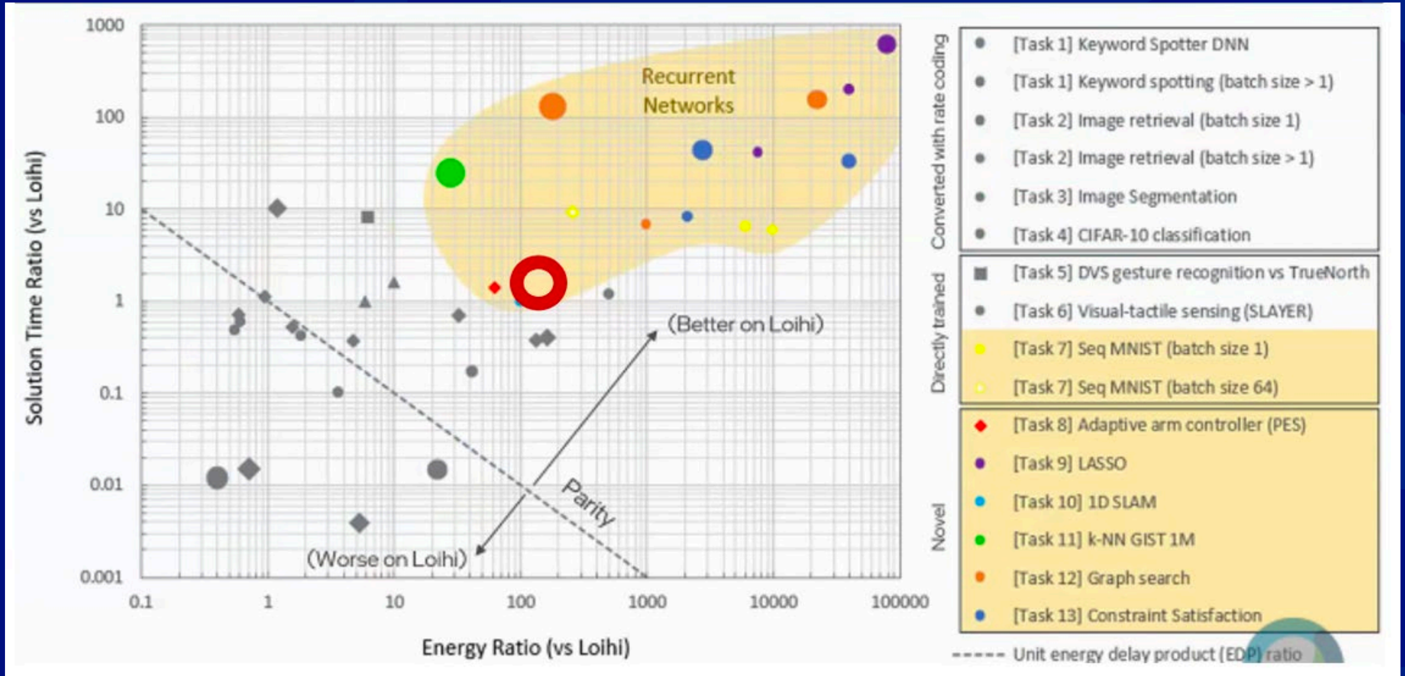
14 Loihi timesteps per training sample

Inference after 3 timesteps

676 FPS, 1.48 ms/sample

0.592 mJ/sample

Energy-delay product = 0.9µJs



Toolkit of Neuronal and Circuit Mechanisms for Spiking Backprop

Neuronal and network mechanisms for implementing backprop:

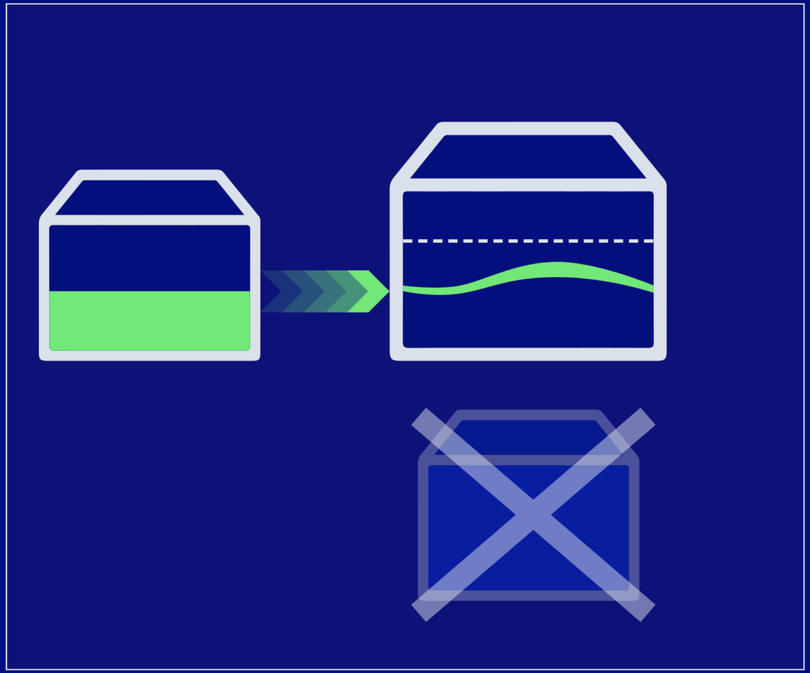
- *Synfire-gated synfire chain(s)*
- *Short-term memories*
- *Push-me pull-you pairs for encoding real numbers and probabilities*
- *Gating of thresholded activity*
- *Gating of derivative of activity via SGSC*
- *Implementation of Hadamard product via pulse-gating*
- *Simultaneous gating of graded information to pre- and post-synaptic neuronal populations for Hebbian synaptic update (turning learning on and off via pulse-gated control)*

Sornborger, Andrew, Louis Tao, Jordan Snyder, and Anatoly Zlotnik. "A Pulse-gated, Neural Implementation of the Backpropagation Algorithm." In *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*, pp. 1-9. 2019.

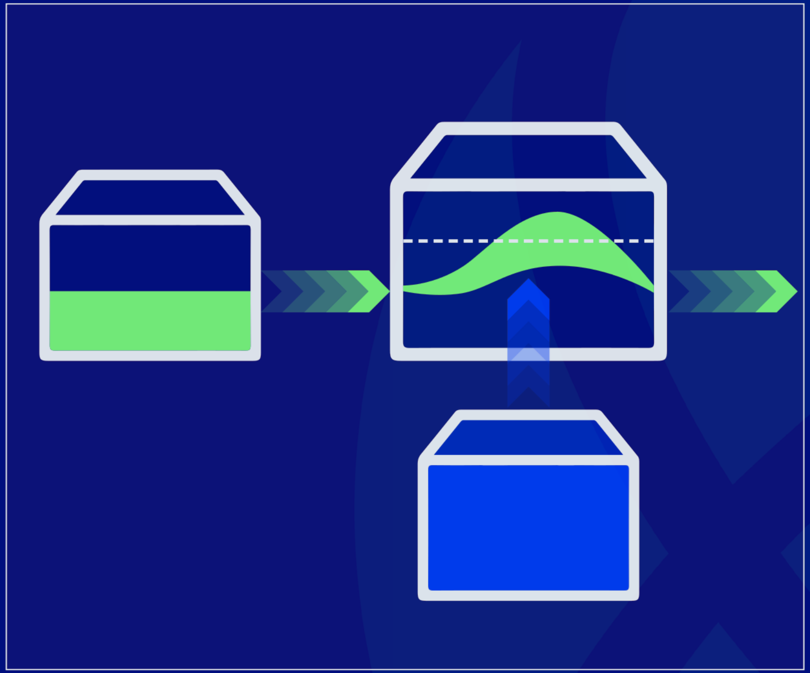
Alpha Renner, Forrest Sheldon, Louis Tao, Anatoly Zlotnik, Andrew Sornborger. "A Pulse-gated, Spiking Neural Implementation of the Backpropagation Algorithm." *Proceedings of the 78h Annual Neuro-inspired Computational Elements Workshop*, pp. 1-9. 2020.

Renner, Alpha, Forrest Sheldon, Anatoly Zlotnik, Louis Tao, and Andrew Sornborger. "The backpropagation algorithm implemented on spiking neuromorphic hardware." *arXiv preprint arXiv:2106.07030* (2021).

Synfire Gated Synfire Chains – Implementing Communication Through Coherence



Synfire-chain neuron (blue with X) inactive and hence fails to potentiate information flow



Synfire-chain neuron (blue) active and hence potentiates information flow

Sornborger, Andrew T., Zhuo Wang, and Louis Tao. "A mechanism for graded, dynamically routable current propagation in pulse-gated synfire chains and implications for information coding." *Journal of computational neuroscience* 39, no. 2 (2015): 181-195.

Wang, Zhuo, Andrew T. Sornborger, and Louis Tao. "Graded, dynamically routable information processing with synfire-gated synfire chains." *PLoS computational biology* 12, no. 6 (2016): e1004979.

Issues in Implementing On-Chip Spiking Backprop

Weight transport problem: For correct credit assignment, feedback weights must be the same as feedforward weights.

Backwards computation problem: Forward and error backpropagation passes implement different computations.

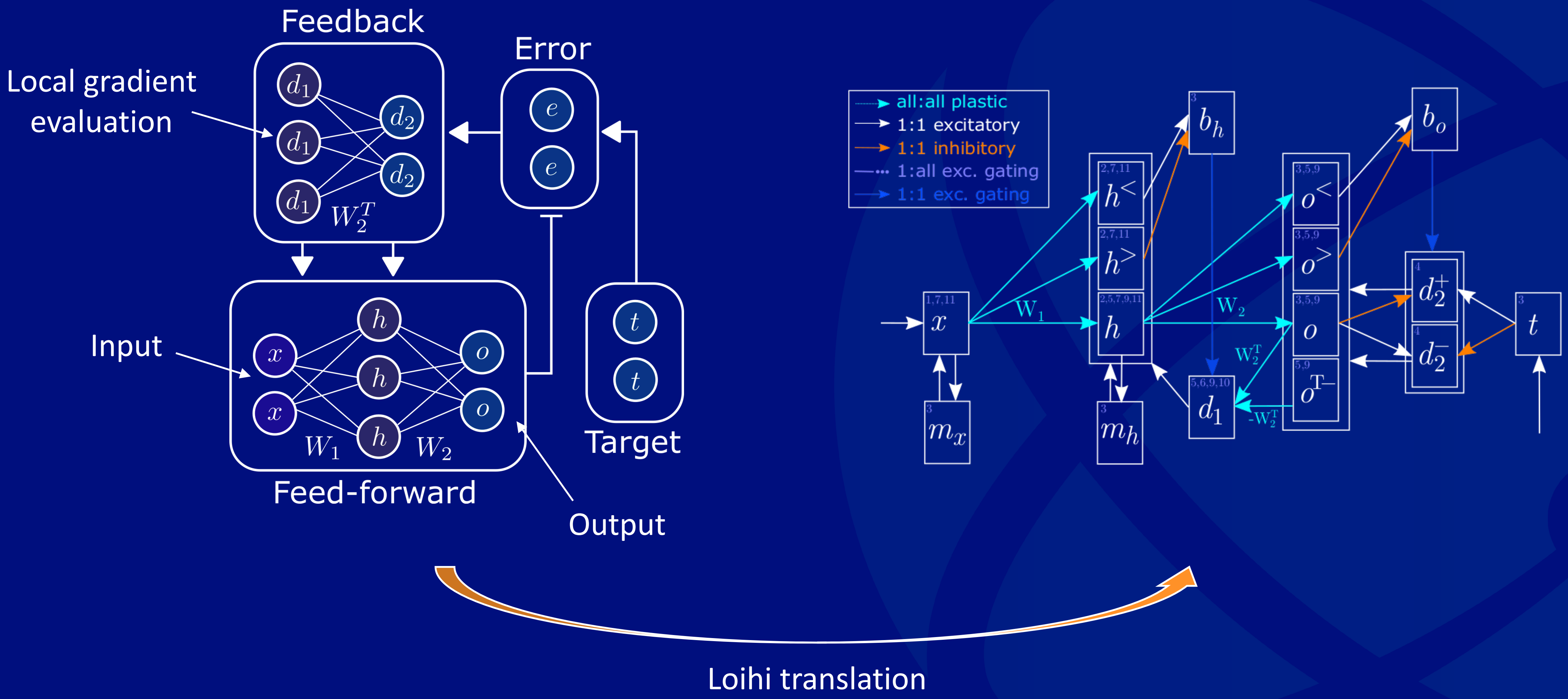
Differentiability problem: Spikes are non-differentiable.

Hardware constraints problem: Constraints on plasticity mechanisms. On some hardware, no plasticity is offered at all; in some cases only specific STDP rules are allowed; and, in almost all cases, it is necessary that information is local, i.e. information is only shared between neurons that are synaptically connected. This is also important for scalability. Furthermore, sufficient weight precision is needed for training.

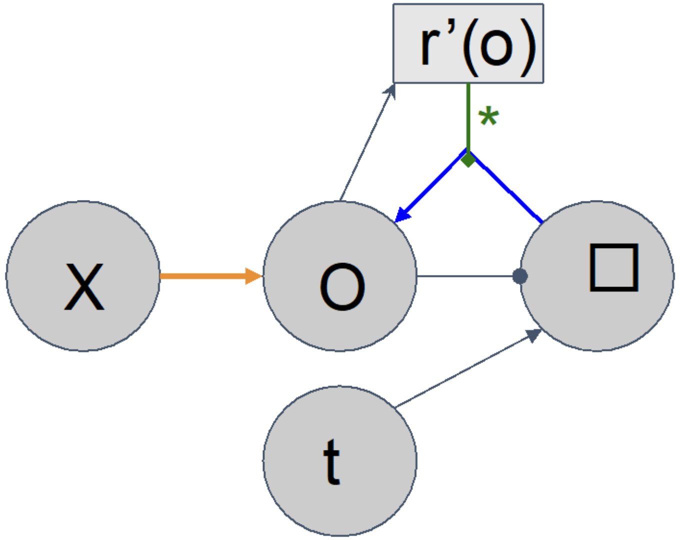
Renner, Alpha, Forrest Sheldon, Anatoly Zlotnik, Louis Tao, and Andrew Sornborger. "The backpropagation algorithm implemented on spiking neuromorphic hardware." *arXiv preprint arXiv:2106.07030* (2021).

Our Circuit Structure

Alpha Renner, Forrest Sheldon, Anatoly Zlotnik, Louis Tao, Andrew Sornborger. ”The Backpropagation Algorithm Implemented on Spiking Neuromorphic Hardware." arXiv:2106.07030 [cs.NE].



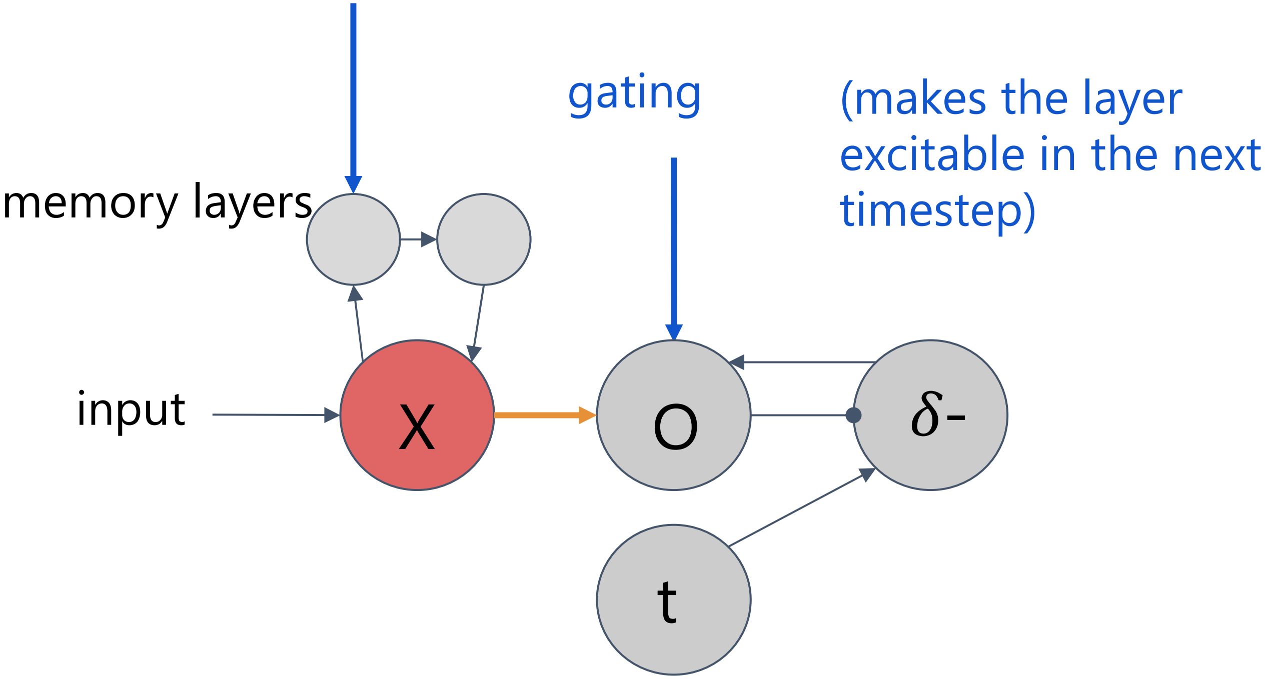
Backpropagation Algorithm



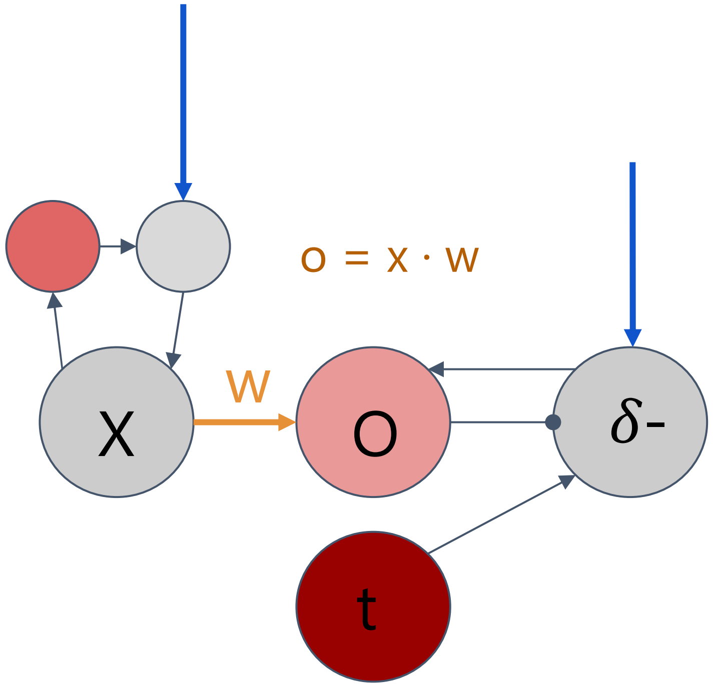
Update for a single neuron:

$$\Delta w_0^{ij} = \underbrace{\delta_a^i}_{\text{“error”}} \cdot \underbrace{x^j}_{\text{Input}} \cdot \underbrace{r'}_{\text{Derivative of activation function}}(z^i)$$

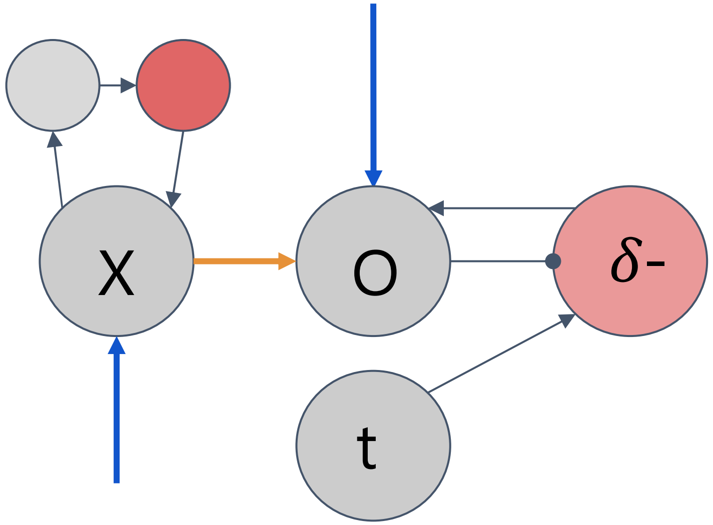
The learning mechanism in detail



The learning mechanism in detail

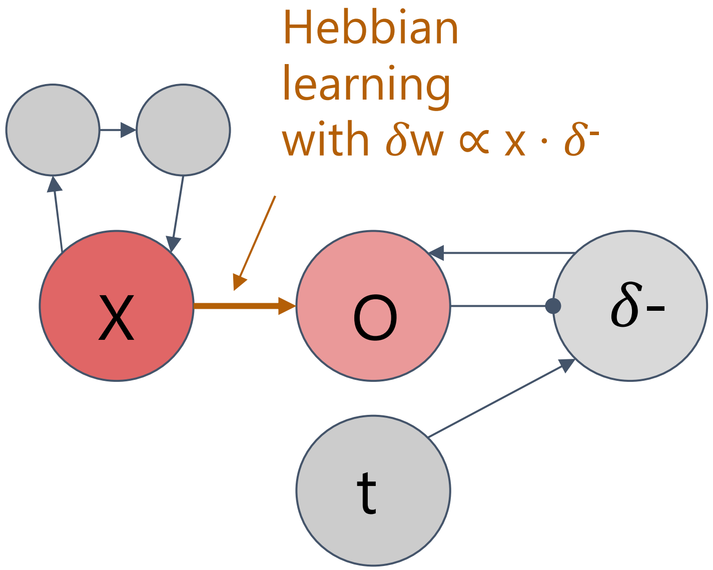


The learning mechanism in detail



δ^{-} = t - o

The learning mechanism in detail

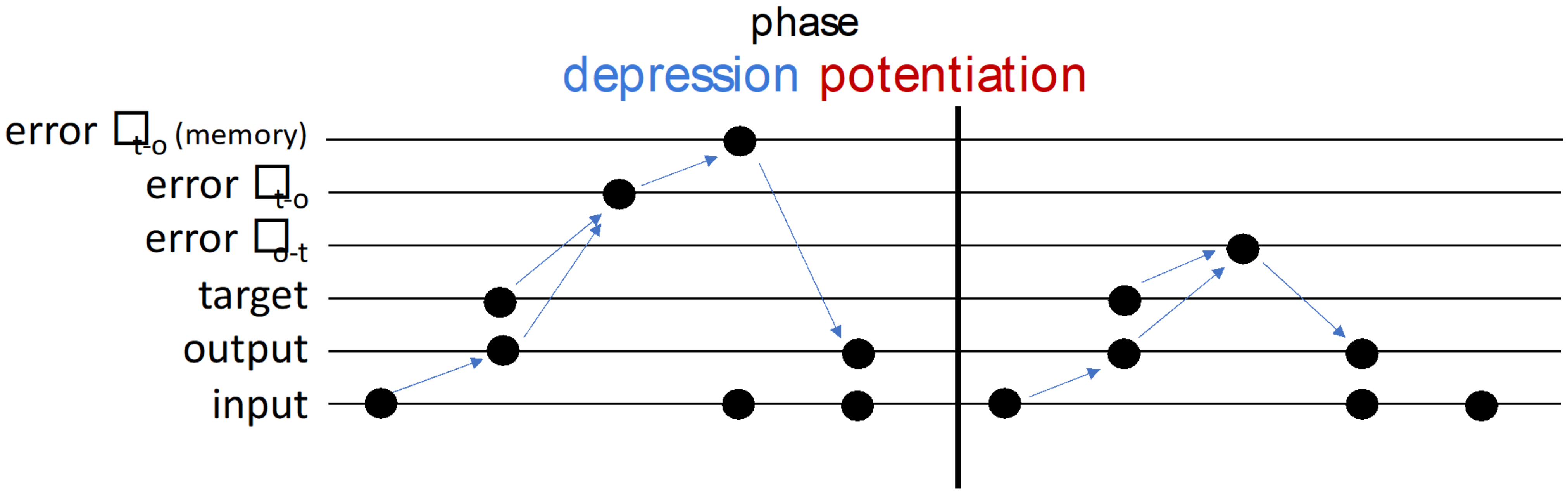


Note:

This is just a simplified visualization, the actual δw is:

$\delta w \propto \delta \cdot x \cdot r'(o)$

The learning mechanism in detail



phase

depression potentiation

error □$_{t-o}$ (memory)

error □$_{t-o}$

error □$_{o-t}$

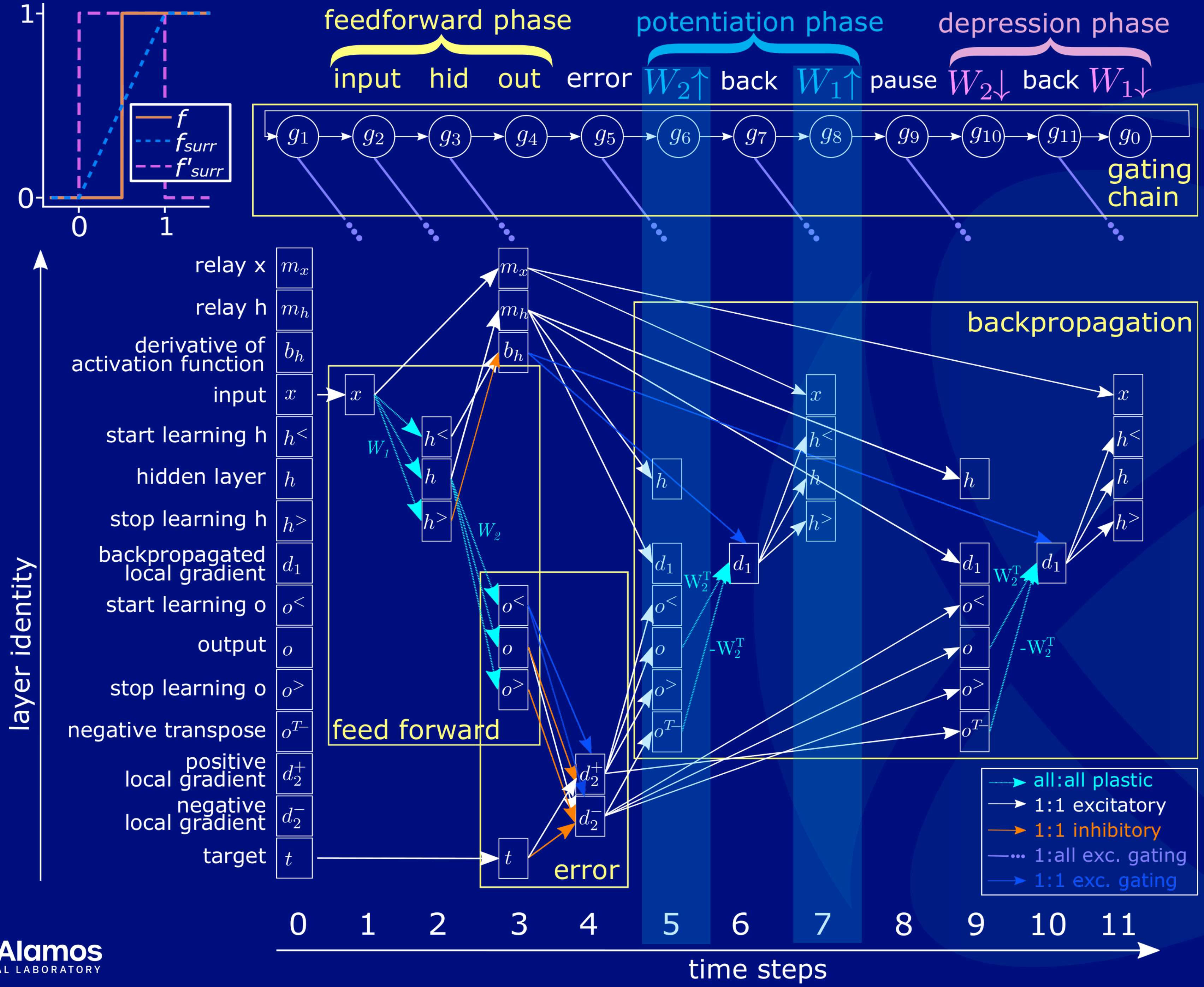
target

output

input

Target > Output
→ Hebbian learning in
potentiation phase

Target < Output
→ Hebbian learning in
depression phase



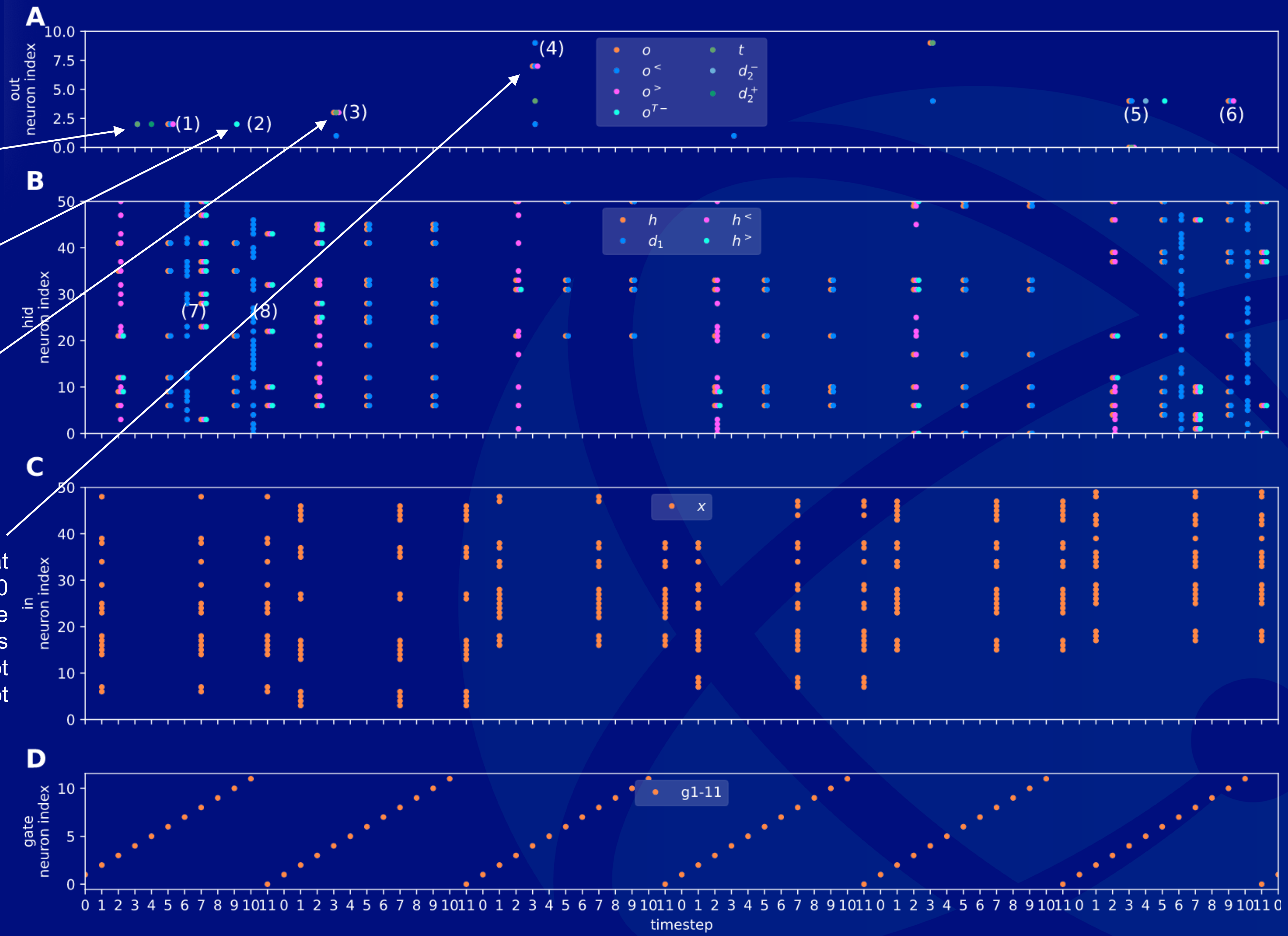
Alpha Renner, Forrest Sheldon, Anatoly Zlotnik, Louis Tao, Andrew Sornborger. ”The Backpropagation Algorithm Implemented on Spiking Neuromorphic Hardware." arXiv:2106.07030 [cs.NE].

error (target but no output spike) leads to potentiation of the W2 synaptic weight and the positive transpose

The same error leads to depression of the negative transpose via activity of d1

no error because o and t fire at the same location, i.e. there is no update in this iteration

there is an error (t fires at index 4, but o at index 7), but the local gradient is 0 because it is gated ‘off’ at index 7 because the derivative of the activation function is 0, i.e. both o< and o> fire. Also, it is not gated ‘on’ at index 4, because o< does not fire



Alpha Renner, Forrest Sheldon, Anatoly Zlotnik, Louis Tao, Andrew Sornborger. ”The Backpropagation Algorithm Implemented on Spiking Neuromorphic Hardware." arXiv:2106.07030 [cs.NE].

Results Preview: MNIST

Validation – 96%

14 Loihi timesteps per training sample

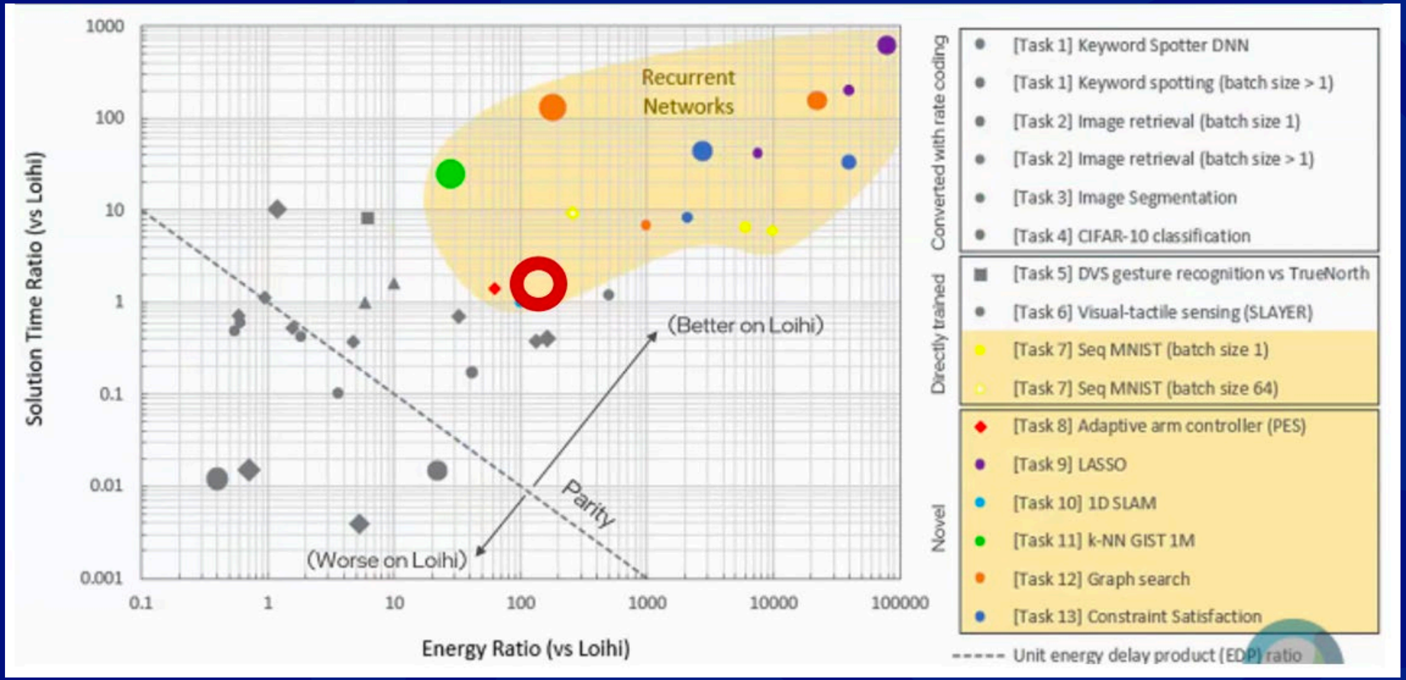
Inference after 3 timesteps

676 FPS, 1.48 ms/sample

0.592 mJ/sample

Energy-delay product = 0.9µJs

Roughly 2 orders-of-magnitude less power used relative to GPU

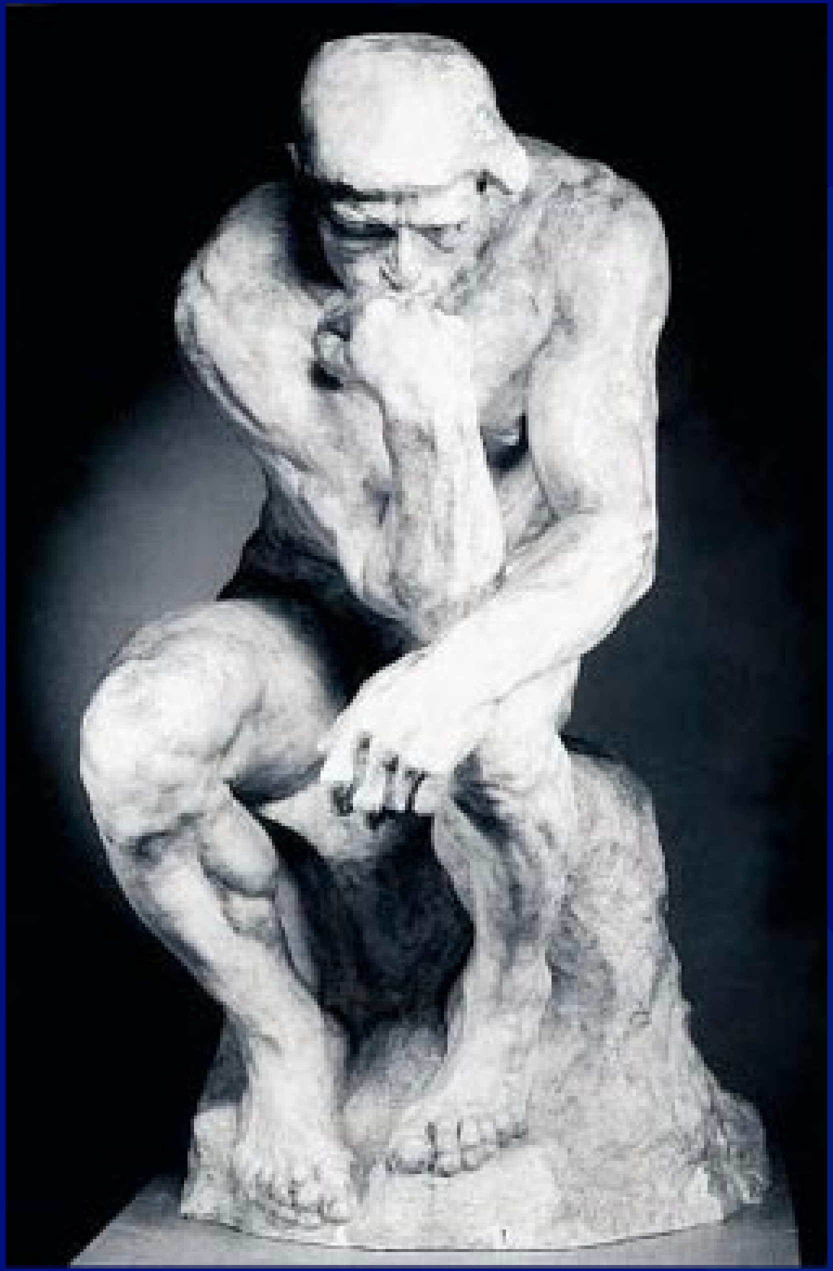


Los Alamos
NATIONAL LABORATORY

Summary

- We have implemented the first backpropagation algorithm that is fully on-chip with no computer in the loop or help from the on-chip x86 microprocessors.
- Compared to an equivalent implementation on a GPU, there is no loss in accuracy, but there are about two orders of magnitude power savings in the case of small batch sizes which are more realistic for edge computing settings.
- The network model we propose offers significant opportunities as a building block that can, e.g. be integrated into larger SNN architectures that could profit from a trainable on-chip machine learning module.

Questions?



May 9, 2022

Matthew Sgambati
HPC System Administrator

An overview of the GPU hardware and System Conda Environments for AI/ML on HPC

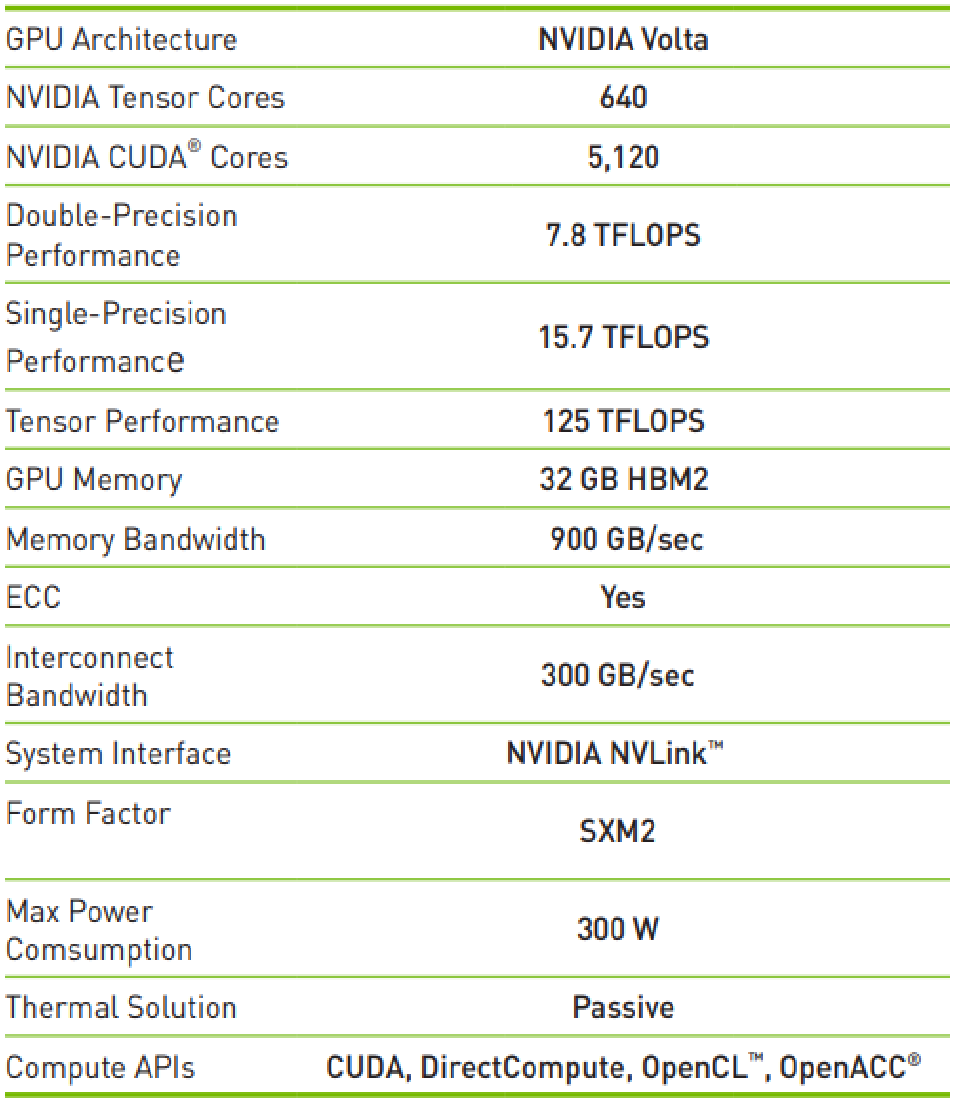
Idaho National Laboratory

Overview

- Sawtooth
  - 108 NVIDIA V100 SMX2s
  - 100Gb/s NVIDIA Mellanox EDR InfiniBand
- Hoodoo
  - 44 NVIDIA A100 SMX4s
  - 200Gb/s NVIDIA Mellanox HDR InfiniBand

Sawtooth

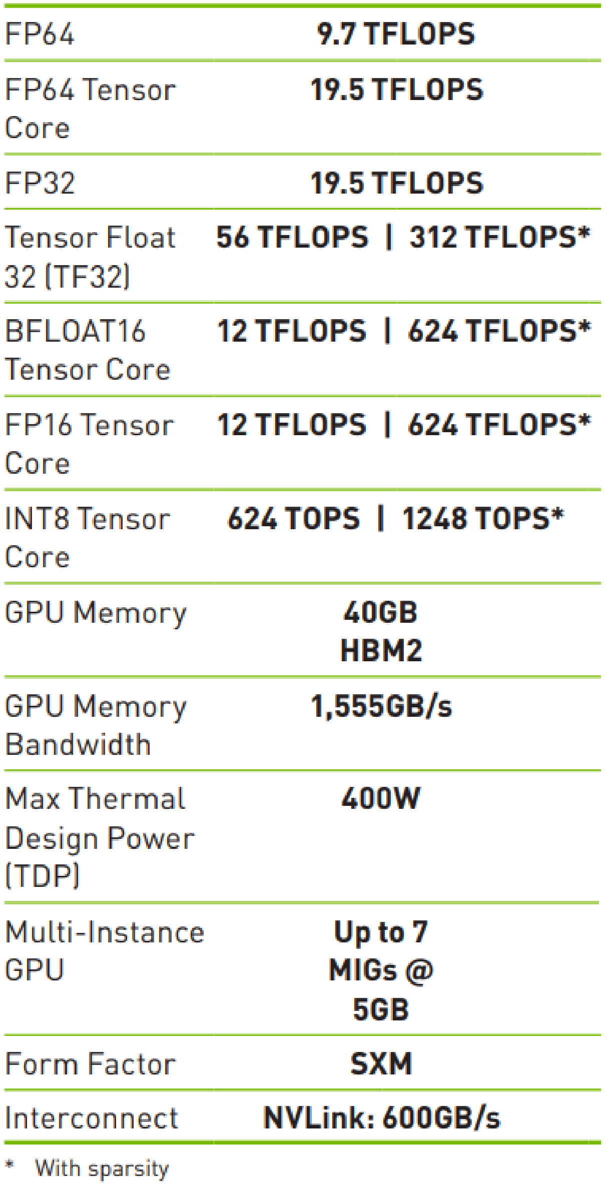
- V100 SXM2



GPU Architecture	NVIDIA Volta
NVIDIA Tensor Cores	640
NVIDIA CUDA® Cores	5,120
Double-Precision Performance	7.8 TFLOPS
Single-Precision Performance	15.7 TFLOPS
Tensor Performance	125 TFLOPS
GPU Memory	32 GB HBM2
Memory Bandwidth	900 GB/sec
ECC	Yes
Interconnect Bandwidth	300 GB/sec
System Interface	NVIDIA NVLink™
Form Factor	SXM2
Max Power Comsumption	300 W
Thermal Solution	Passive
Compute APIs	CUDA, DirectCompute, OpenCL™, OpenACC®

Hoodoo

- A100 SXM4

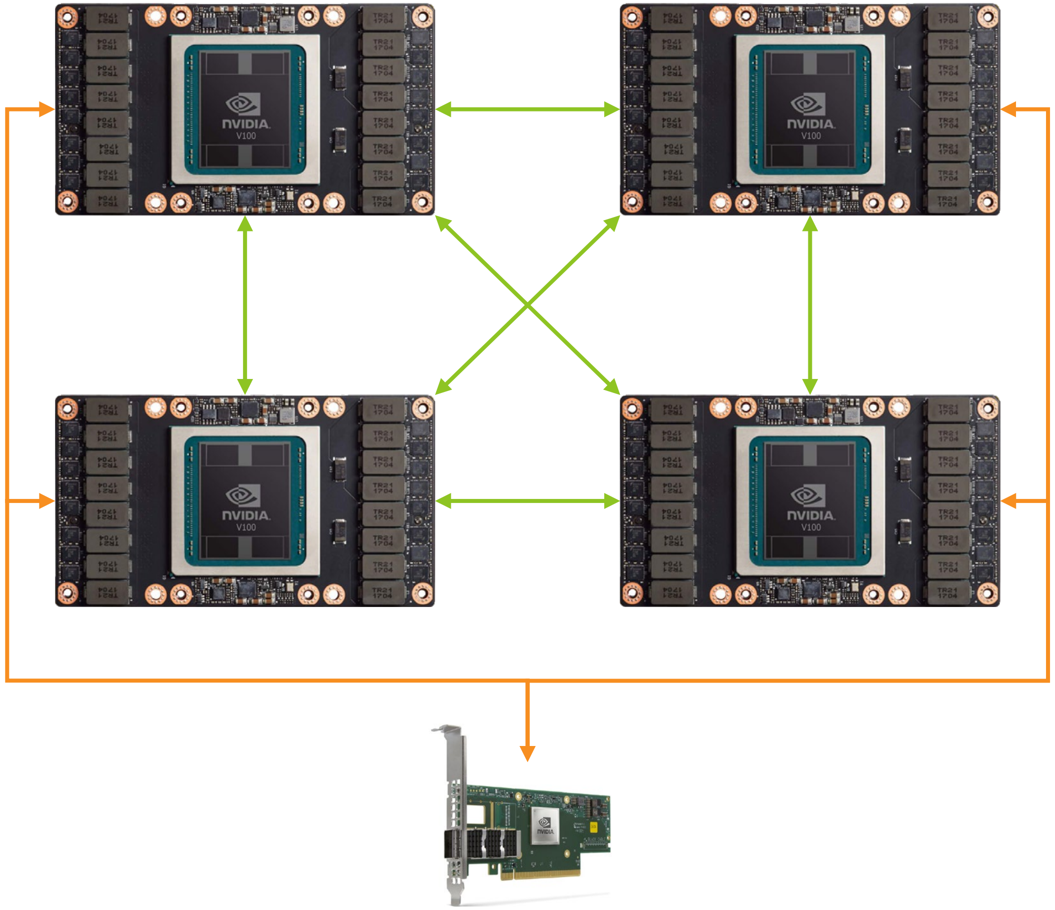


FP64	9.7 TFLOPS
FP64 Tensor Core	19.5 TFLOPS
FP32	19.5 TFLOPS
Tensor Float 32 (TF32)	56 TFLOPS | 312 TFLOPS*
BFLOAT16 Tensor Core	12 TFLOPS | 624 TFLOPS*
FP16 Tensor Core	12 TFLOPS | 624 TFLOPS*
INT8 Tensor Core	624 TOPS | 1248 TOPS*
GPU Memory	40GB HBM2
GPU Memory Bandwidth	1,555GB/s
Max Thermal Design Power (TDP)	400W
Multi-Instance GPU	Up to 7 MIGs @ 5GB
Form Factor	SXM
Interconnect	NVLink: 600GB/s

* With sparsity

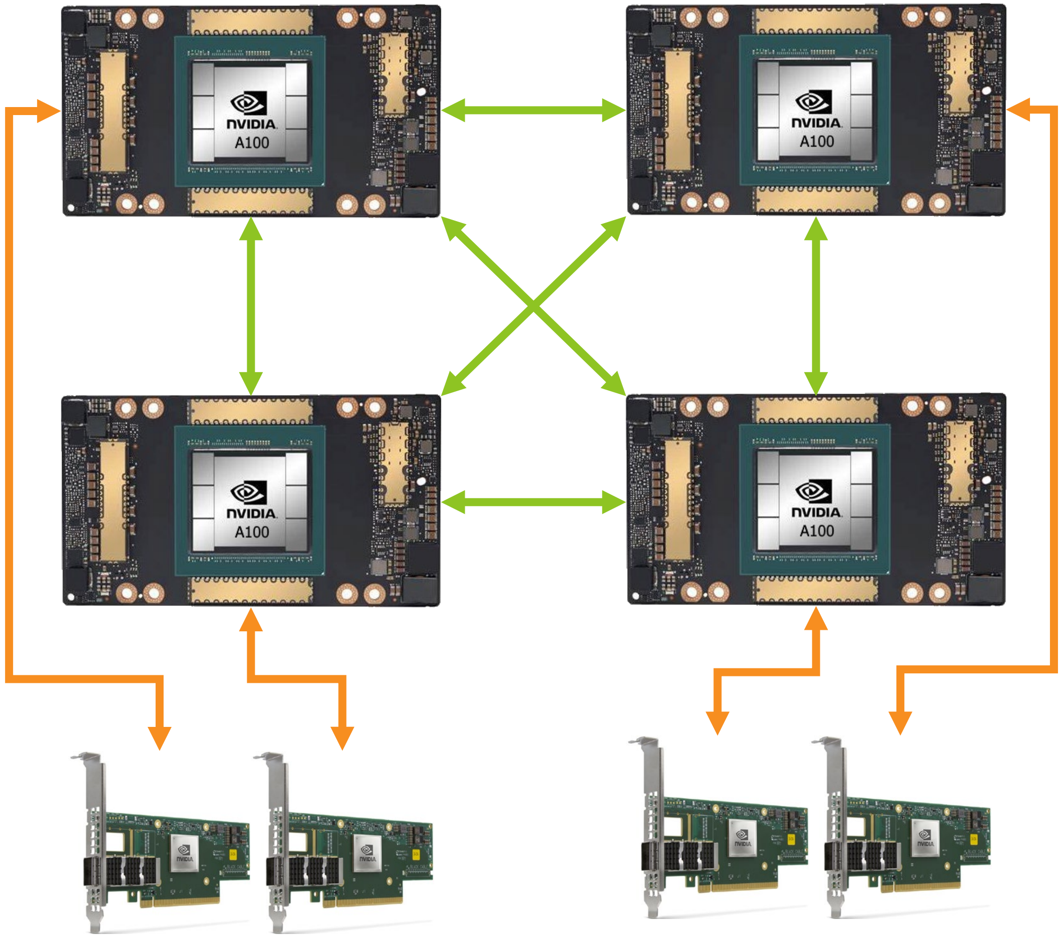
Sawtooth

- Connection structure
  - NVLink – 300GB/sec
  - IB EDR –100Gb/sec



Hoodoo

- Connection structure
  - NVLink – 600GB/sec
  - IB HDR –200Gb/sec



System Conda Environments Sawtooth

- OpenAI Gym
- Python 3
- Python 3.7 Boltz TraP2
- Python 3.7 Pytorch 1.4
- Python 3.7 Rapids 0.13
- Python 3.7 Tensorflow 1.15
- Python 3.7 Tensorflow 2.1 GPU
- Python 3.7 Tensorflow 2.1 Horovod
- Python 3.7 Tensorflow 2.4 gpu
- Python 3.8 Rapids 22.04
- R 3.6.1
- Tensorflow 2.5
- pymatgen
- Pytorch-1.8.1

System Conda Environments Hoodoo

- Python 3
- Fastai PyTorch CUDA 11.2
- PyTorch 1.11.0 Horovod Cuda 11.4
- Pytorch 1.7.1 Horovod Cuda 11.1
- pytorch 1.8.1
- Tensorflow 2.4 Horovod Cuda 11.1
- Tensorflow 2.4 Horovod Cuda 11.2
- tensorflow-2.8

Questions?

Idaho National Laboratory

May 2, 2022

Brandon Biggs
INL/MIS-22-67115

Management of AI/ML Programming Environments

Idaho National Laboratory

Roadmap

ML/AI Programming Scenarios

Managing virtual environments with Conda

Conda vs other tools

How INL HPC uses Conda

Scenario - You want to run a model requiring Tensorflow 2.8 and Horovod

- What kind of compute resources do you need?
- If you wanted to set this up locally, you'd have to manage the software stack yourself (Conda, CUDA, RAPIDS, Docker, MPI, etc)
- On INL HPC you can load a module or start a container
- Can also use some existing infrastructure to create your own environment

```
$ module load conda
$ conda create -n “tensorflow_2.8_horovod” --python=3.8
$ conda activate tensorflow_2.8_horovod
```

Scenario - INL HPC is missing a package or framework that I need...

- If you’re looking for a framework or package that we don’t already have let us know by creating a support ticket by emailing hpcsupport@inl.gov
- You can also use the environments that we setup and add your own packages:

```
$ module load conda
$ conda activate “tensorflow_2.8”
$ pip install --user PACKAGE
```

Managing Virtual Environments with Conda

- Package Management
  - Software packages
  - Dependencies (more than just other Python packages!)
- Manages environments
  - Different versions of software
  - Different environment requirements
    - R vs Python Ruby vs Lua vs ...
    - TensorFlow vs Pytorch vs FastAI vs RAPIDS vs ....

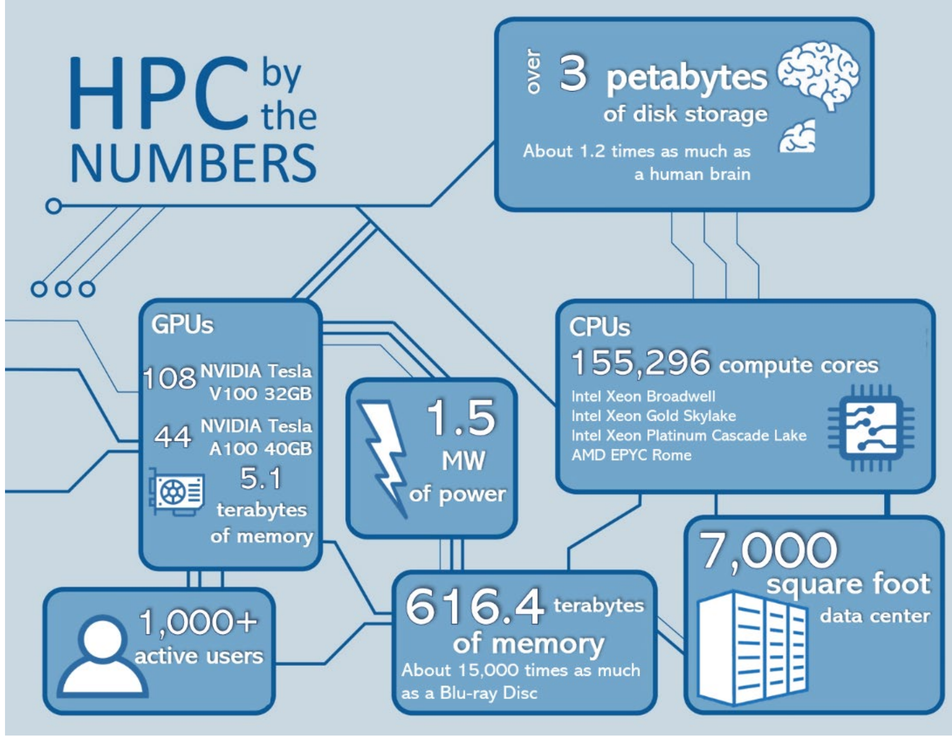
Conda vs Other Tools

- Conda vs pip
  - We use both together
  - Conda has been great for dependency resolution on older operating systems like CentOS7
  - Use Conda to create environment and download some packages, pip for others
  - Some frameworks are dropping support for pip
    - RAPIDS dropped support for pip in 2019[1]
- Conda vs Containers
  - We're in the early stages of using containers for reproducibility
  - Conda isn't great at being reproducible or portable. Basically, start from scratch
  - Users don't need root to build a Conda environment

[1] https://medium.com/rapids-ai/rapids-0-7-release-drops-pip-packages-47fc966e9472

How INL HPC uses Conda

- 33 General environments
  - 11 variants/versions of TensorFlow
  - 6 variants/versions of PyTorch
  - 2 versions of RAPIDS
- Setup as Jupyter Notebook kernels allowing people to change environments with one click
- Allow users to create their own Python environments in their home directory or in a project directory


HPC by the NUMBERS
over 3 petabytes of disk storage
About 1.2 times as much as a human brain
GPUs
108 NVIDIA Tesla V100 32GB
44 NVIDIA Tesla A100 40GB
5.1 terabytes of memory
1.5 MW of power
CPUs
155,296 compute cores
Intel Xeon Broadwell
Intel Xeon Gold Skylake
Intel Xeon Platinum Cascade Lake
AMD EPYC Rome
1,000+ active users
616.4 terabytes of memory
About 15,000 times as much as a Blu-ray Disc
7,000 square foot data center

Jupyter Notebooks – Open OnDemand

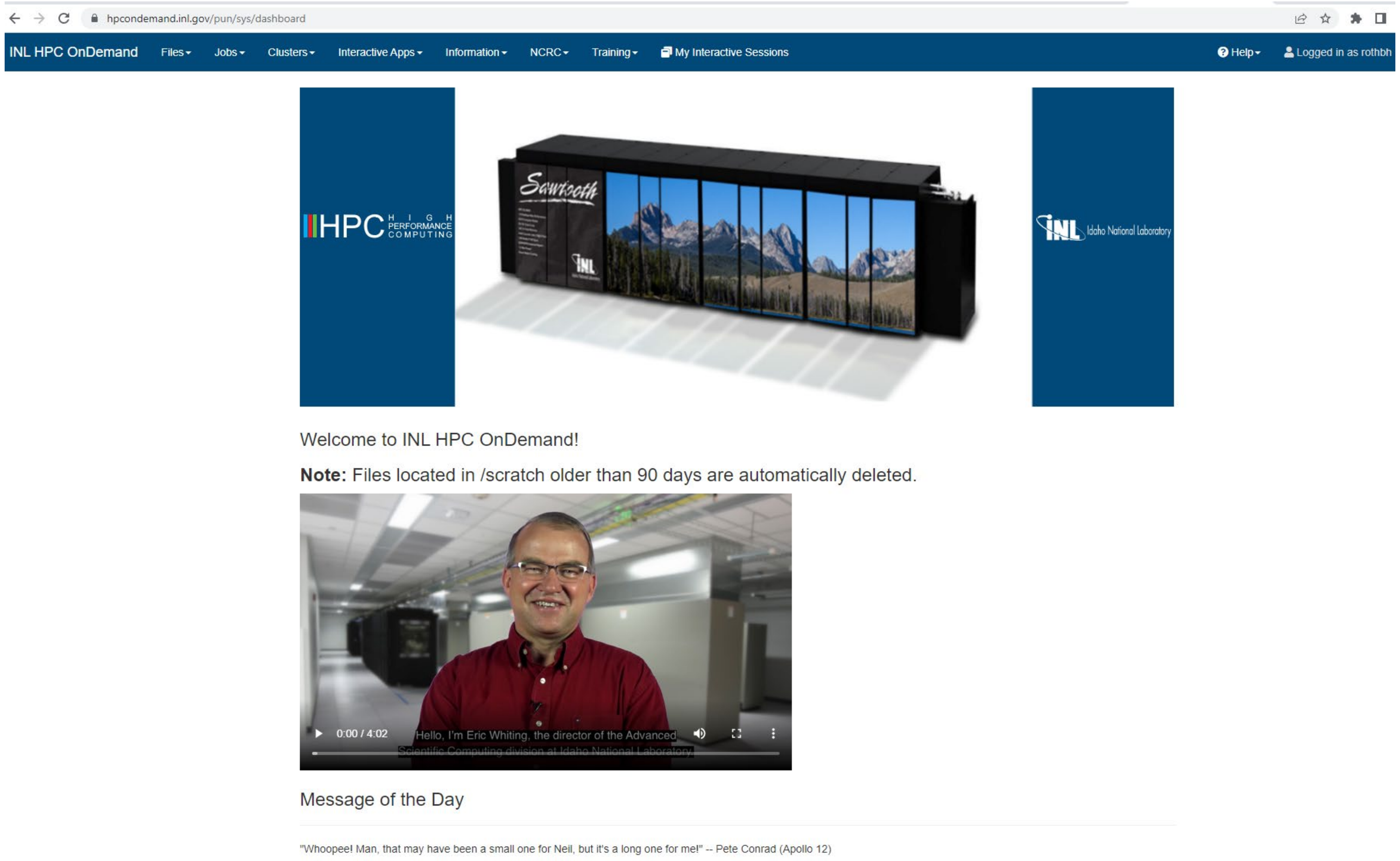
Bradlee Rothwell
High Performance Computing
Idaho National Laboratory

May 26, 2022

www.inl.gov

Idaho National Laboratory

Step 1: Log onto HPC OnDemand


hpcondemand.inl.gov/pun/sys/dashboard
INL HPC OnDemand
Files
Jobs
Clusters
Interactive Apps
Information
NCRC
Training
My Interactive Sessions
Help
Logged in as rothbh
HPC HIGH PERFORMANCE COMPUTING
Sawtooth
INL Idaho National Laboratory
0:00 / 4:02
Hello, I'm Eric Whiting, the director of the Advanced Scientific Computing division at Idaho National Laboratory.

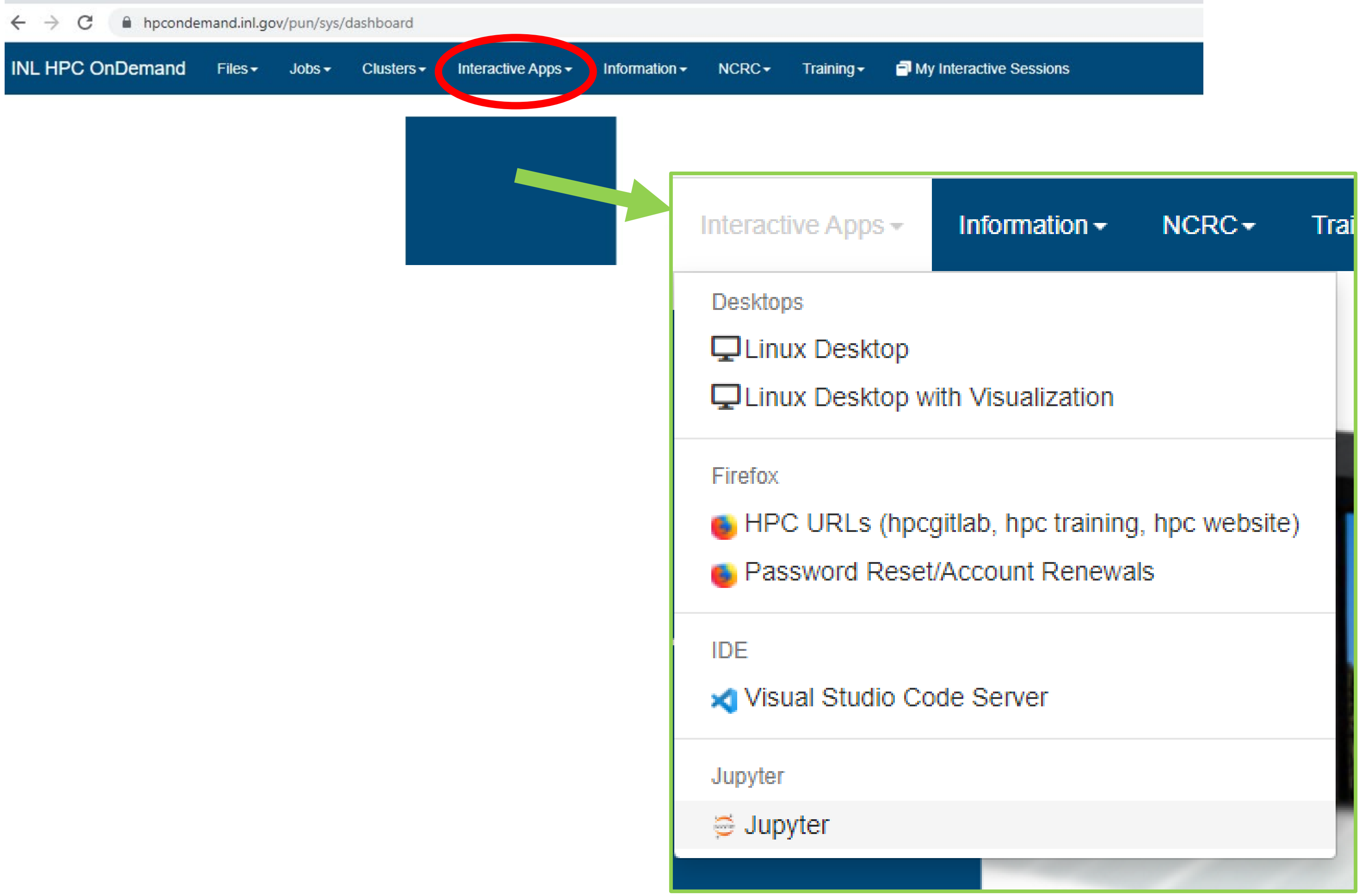
Welcome to INL HPC OnDemand!

Note: Files located in /scratch older than 90 days are automatically deleted.

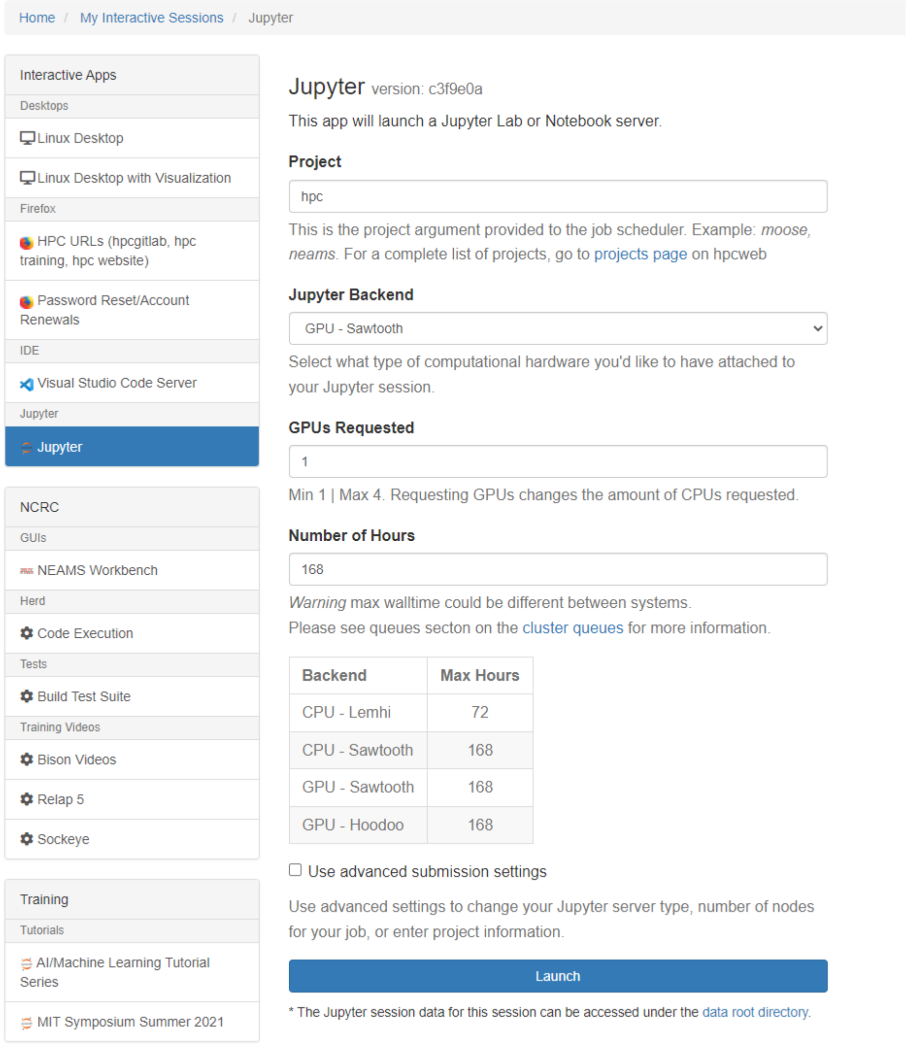
Message of the Day

"Whoopee! Man, that may have been a small one for Neil, but it's a long one for me!" -- Pete Conrad (Apollo 12)

Step 2: Selecting a Jupyter Notebook



Step 3: Launching a Jupyter job



Home / My Interactive Sessions / Jupyter

Interactive Apps

Desktops

- Linux Desktop
- Linux Desktop with Visualization

Firefox

- HPC URLs (hpcgitlab, hpc training, hpc website)
- Password Reset/Account Renewals

IDE

- Visual Studio Code Server

Jupyter

- Jupyter

NCRC

GUIs

- NEAMS Workbench

Herd

- Code Execution

Tests

- Build Test Suite

Training Videos

- Bison Videos
- Relap 5
- Sockeye

Training

Tutorials

- AI/Machine Learning Tutorial Series
- MIT Symposium Summer 2021

Jupyter version: c3f9e0a

This app will launch a Jupyter Lab or Notebook server.

Project

hpc

This is the project argument provided to the job scheduler. Example: *moose*, *neams*. For a complete list of projects, go to projects page on hpcweb

Jupyter Backend

GPU - Sawtooth

Select what type of computational hardware you'd like to have attached to your Jupyter session.

GPUs Requested

1

Min 1 | Max 4. Requesting GPUs changes the amount of CPUs requested.

Number of Hours

168

Warning max walltime could be different between systems.
Please see queues secton on the cluster queues for more information.

Backend	Max Hours
CPU - Lemhi	72
CPU - Sawtooth	168
GPU - Sawtooth	168
GPU - Hoodoo	168

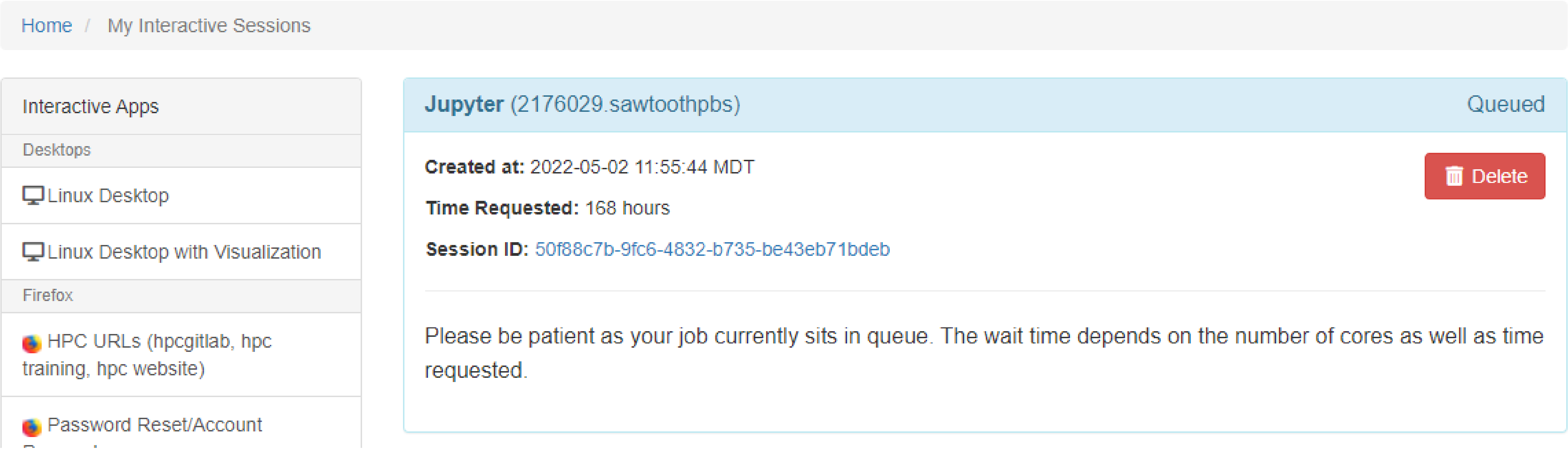
☐ Use advanced submission settings

Use advanced settings to change your Jupyter server type, number of nodes for your job, or enter project information.

* The Jupyter session data for this session can be accessed under the data root directory.

- Project Name
- Cluster – Sawtooth
- CPUs/GPUs Requested
- Number of Hours

Step 3: Launching a Jupyter job



Home / My Interactive Sessions

Interactive Apps
Desktops
Linux Desktop
Linux Desktop with Visualization
Firefox
HPC URLs (hpcgitlab, hpc training, hpc website)
Password Reset/Account
Delete

Jupyter (2176029.sawtoothpbs) Queued

Created at: 2022-05-02 11:55:44 MDT

Time Requested: 168 hours

Session ID: 50f88c7b-9fc6-4832-b735-be43eb71bdeb

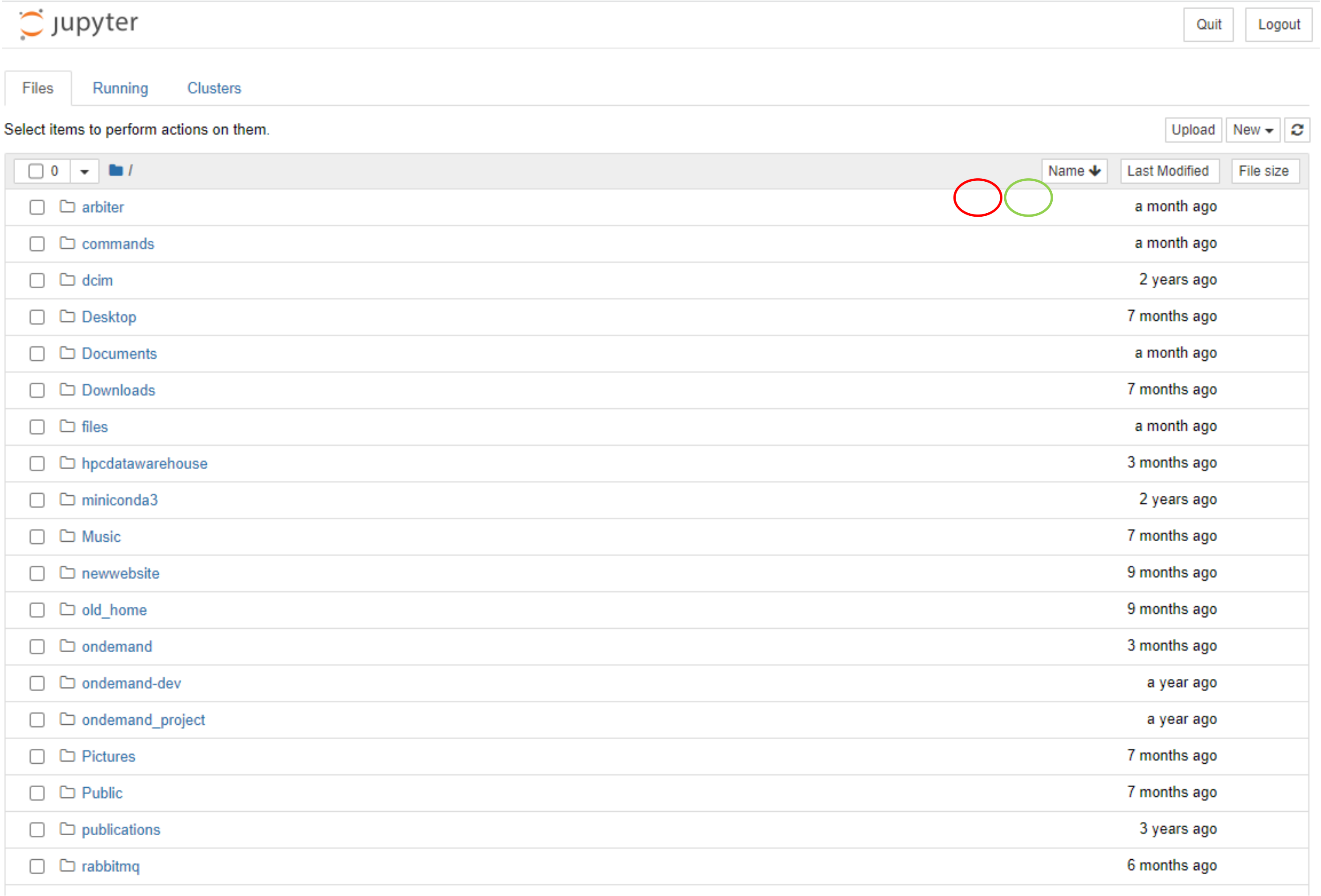
Please be patient as your job currently sits in queue. The wait time depends on the number of cores as well as time requested.

Step 4: Select a Jupyter project

Select a project from home directory

OR

Create a new project



jupyter

Quit | Logout

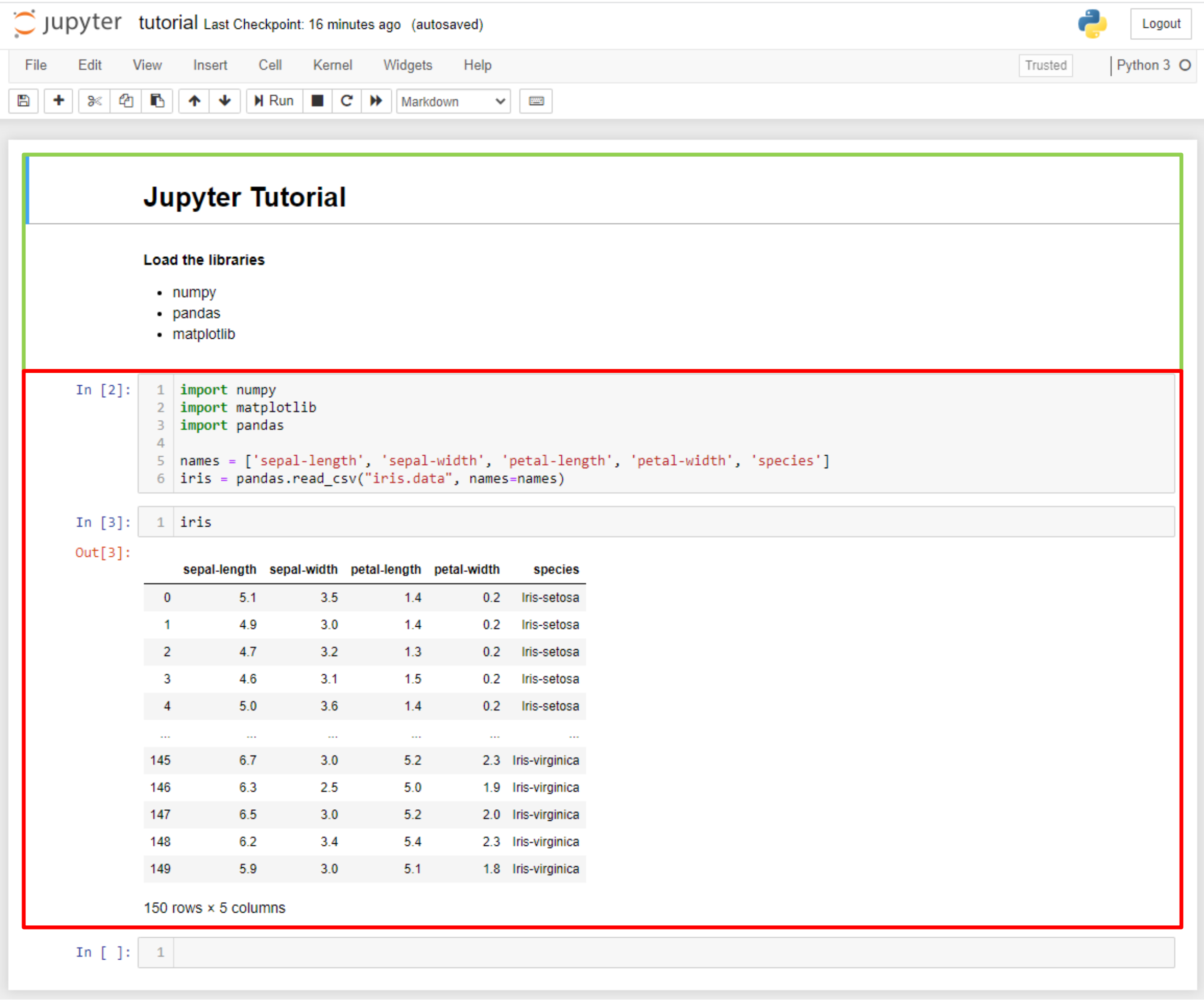
Files | Running | Clusters

Select items to perform actions on them.

Upload | New ▾ | ⟳

□ 0 ▾ /	Name ↓	Last Modified	File size
□ arbiter		a month ago	
□ commands		a month ago	
□ dcim		2 years ago	
□ Desktop		7 months ago	
□ Documents		a month ago	
□ Downloads		7 months ago	
□ files		a month ago	
□ hpcdatawarehouse		3 months ago	
□ miniconda3		2 years ago	
□ Music		7 months ago	
□ newwebsite		9 months ago	
□ old_home		9 months ago	
□ ondemand		3 months ago	
□ ondemand-dev		a year ago	
□ ondemand_project		a year ago	
□ Pictures		7 months ago	
□ Public		7 months ago	
□ publications		3 years ago	
□ rabbitmq		6 months ago	

Running Jupyter Cells



Jupyter Tutorial

Load the libraries

- numpy
- pandas
- matplotlib

In [2]:

```
import numpy
import matplotlib
import pandas

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'species']
iris = pandas.read_csv("iris.data", names=names)
```

In [3]:

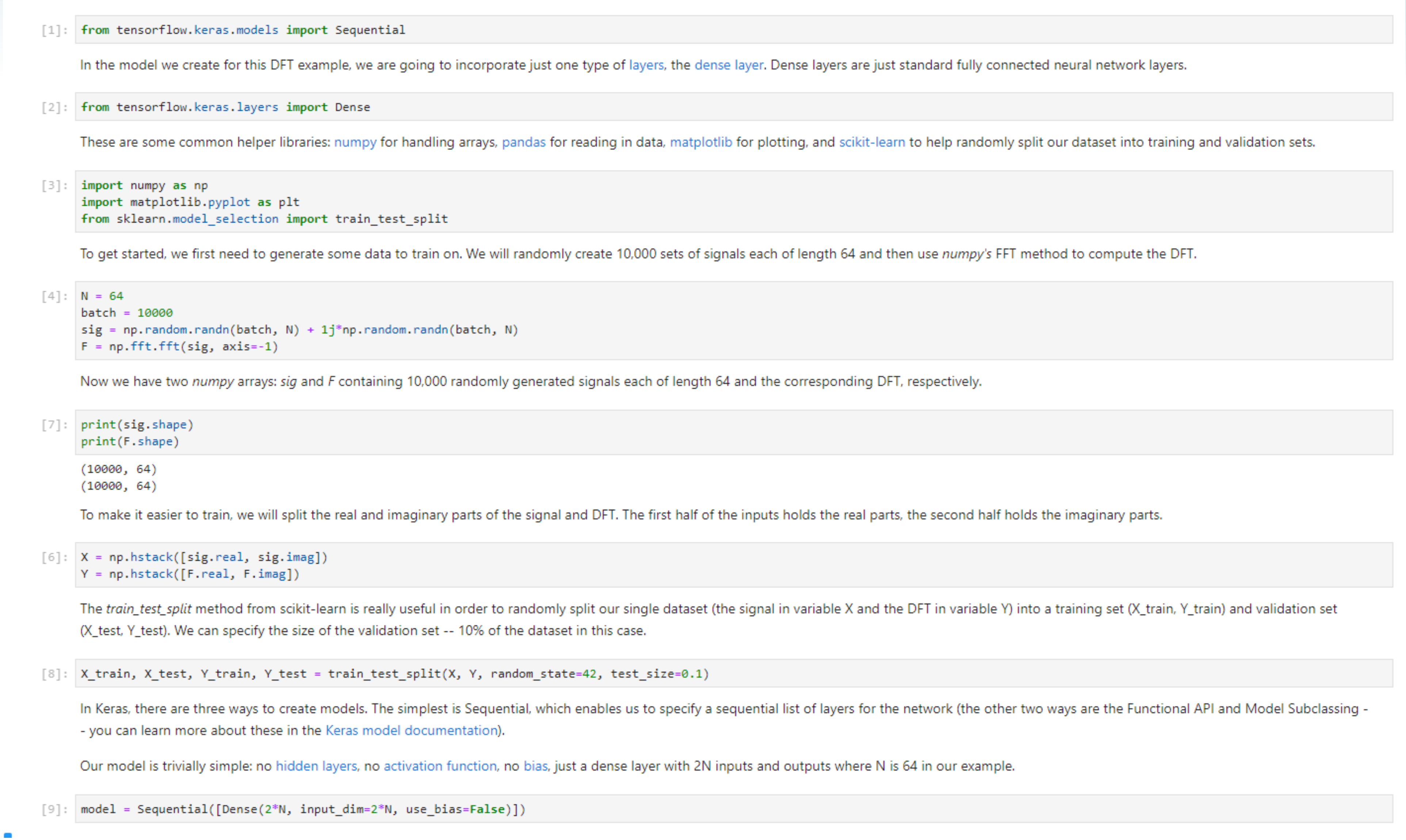
```
iris
```

Out[3]:

	sepal-length	sepal-width	petal-length	petal-width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [ ]:



```
[1]: from tensorflow.keras.models import Sequential
```

In the model we create for this DFT example, we are going to incorporate just one type of layers, the dense layer. Dense layers are just standard fully connected neural network layers.

```
[2]: from tensorflow.keras.layers import Dense
```

These are some common helper libraries: numpy for handling arrays, pandas for reading in data, matplotlib for plotting, and scikit-learn to help randomly split our dataset into training and validation sets.

```
[3]: import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
```

To get started, we first need to generate some data to train on. We will randomly create 10,000 sets of signals each of length 64 and then use *numpy's* FFT method to compute the DFT.

```
[4]: N = 64
     batch = 10000
     sig = np.random.randn(batch, N) + 1j*np.random.randn(batch, N)
     F = np.fft.fft(sig, axis=-1)
```

Now we have two *numpy* arrays: *sig* and *F* containing 10,000 randomly generated signals each of length 64 and the corresponding DFT, respectively.

```
[7]: print(sig.shape)
     print(F.shape)

     (10000, 64)
     (10000, 64)
```

To make it easier to train, we will split the real and imaginary parts of the signal and DFT. The first half of the inputs holds the real parts, the second half holds the imaginary parts.

```
[6]: X = np.hstack([sig.real, sig.imag])
     Y = np.hstack([F.real, F.imag])
```

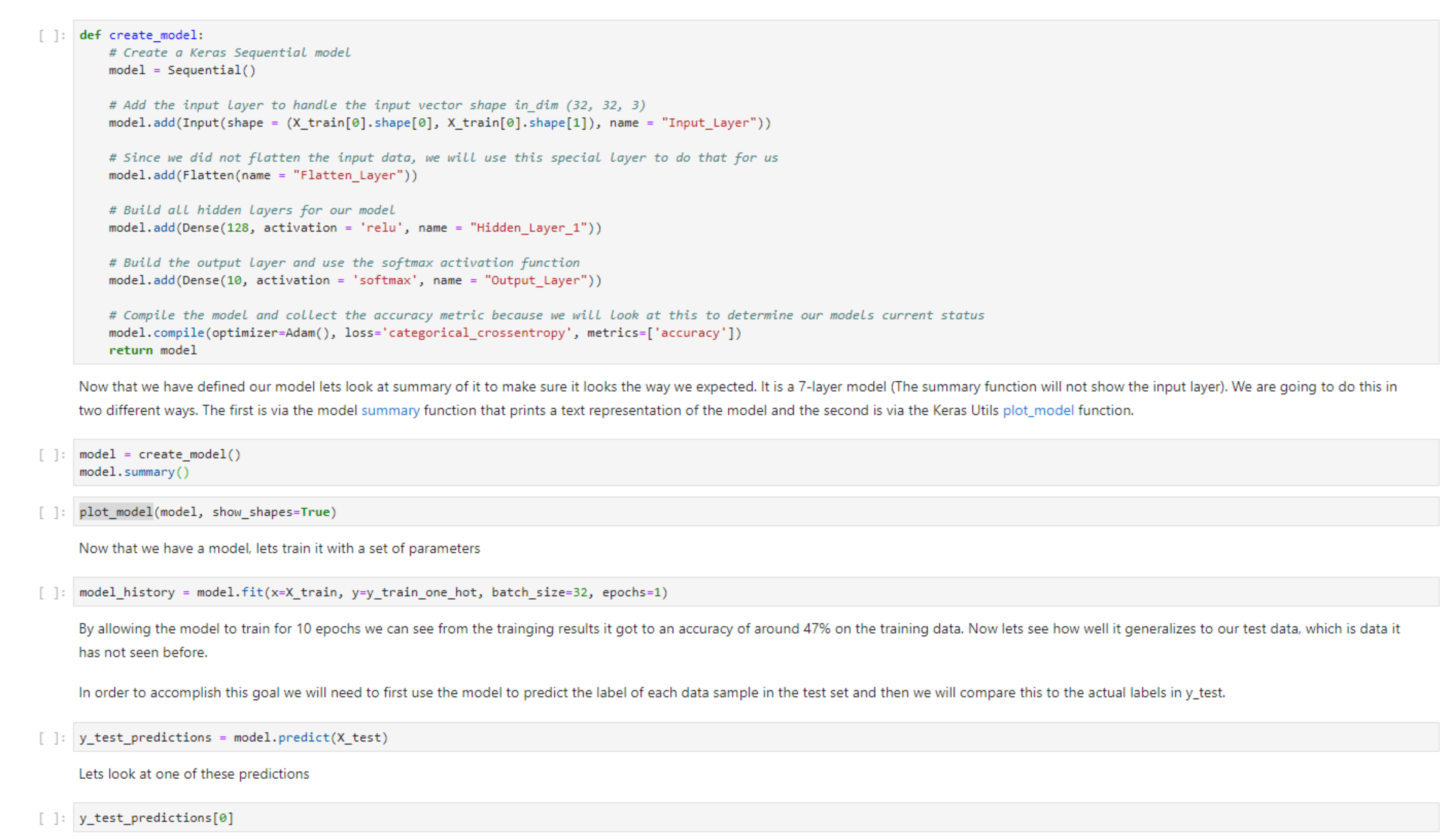
The *train_test_split* method from scikit-learn is really useful in order to randomly split our single dataset (the signal in variable X and the DFT in variable Y) into a training set (X_train, Y_train) and validation set (X_test, Y_test). We can specify the size of the validation set -- 10% of the dataset in this case.

```
[8]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42, test_size=0.1)
```

In Keras, there are three ways to create models. The simplest is Sequential, which enables us to specify a sequential list of layers for the network (the other two ways are the Functional API and Model Subclassing -- you can learn more about these in the Keras model documentation).

Our model is trivially simple: no hidden layers, no activation function, no bias, just a dense layer with 2N inputs and outputs where N is 64 in our example.

```
[9]: model = Sequential([Dense(2*N, input_dim=2*N, use_bias=False)])
```



```
[ ]: def create_model:
         # Create a Keras Sequential model
         model = Sequential()

         # Add the input layer to handle the input vector shape in_dim (32, 32, 3)
         model.add(Input(shape = (X_train[0].shape[0], X_train[0].shape[1]), name = "Input_Layer"))

         # Since we did not flatten the input data, we will use this special layer to do that for us
         model.add(Flatten(name = "Flatten_Layer"))

         # Build all hidden layers for our model
         model.add(Dense(128, activation = 'relu', name = "Hidden_Layer_1"))

         # Build the output layer and use the softmax activation function
         model.add(Dense(10, activation = 'softmax', name = "Output_Layer"))

         # Compile the model and collect the accuracy metric because we will look at this to determine our models current status
         model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
         return model
```

Now that we have defined our model lets look at summary of it to make sure it looks the way we expected. It is a 7-layer model (The summary function will not show the input layer). We are going to do this in two different ways. The first is via the model summary function that prints a text representation of the model and the second is via the Keras Utils plot_model function.

```
[ ]: model = create_model()
     model.summary()
```

```
[ ]: plot_model(model, show_shapes=True)
```

Now that we have a model, lets train it with a set of parameters

```
[ ]: model_history = model.fit(x=X_train, y=y_train_one_hot, batch_size=32, epochs=1)
```

By allowing the model to train for 10 epochs we can see from the trainging results it got to an accuracy of around 47% on the training data. Now lets see how well it generalizes to our test data, which is data it has not seen before.

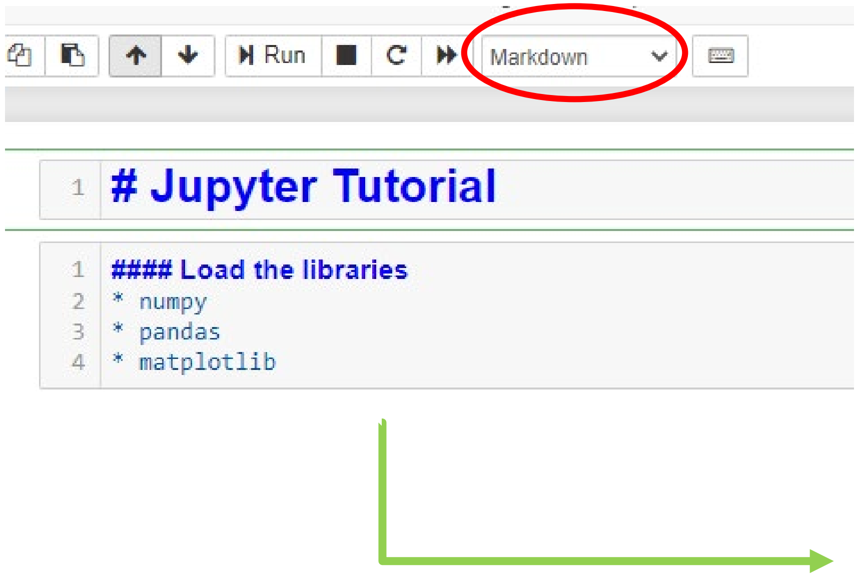
In order to accomplish this goal we will need to first use the model to predict the label of each data sample in the test set and then we will compare this to the actual labels in y_test.

```
[ ]: y_test_predictions = model.predict(X_test)
```

Lets look at one of these predictions

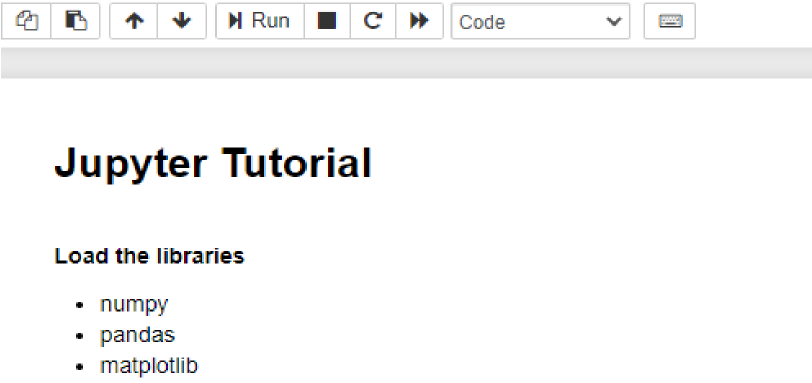
```
[ ]: y_test_predictions[0]
```

Comments



```
# Jupyter Tutorial
```

```
#### Load the libraries
* numpy
* pandas
* matplotlib
```

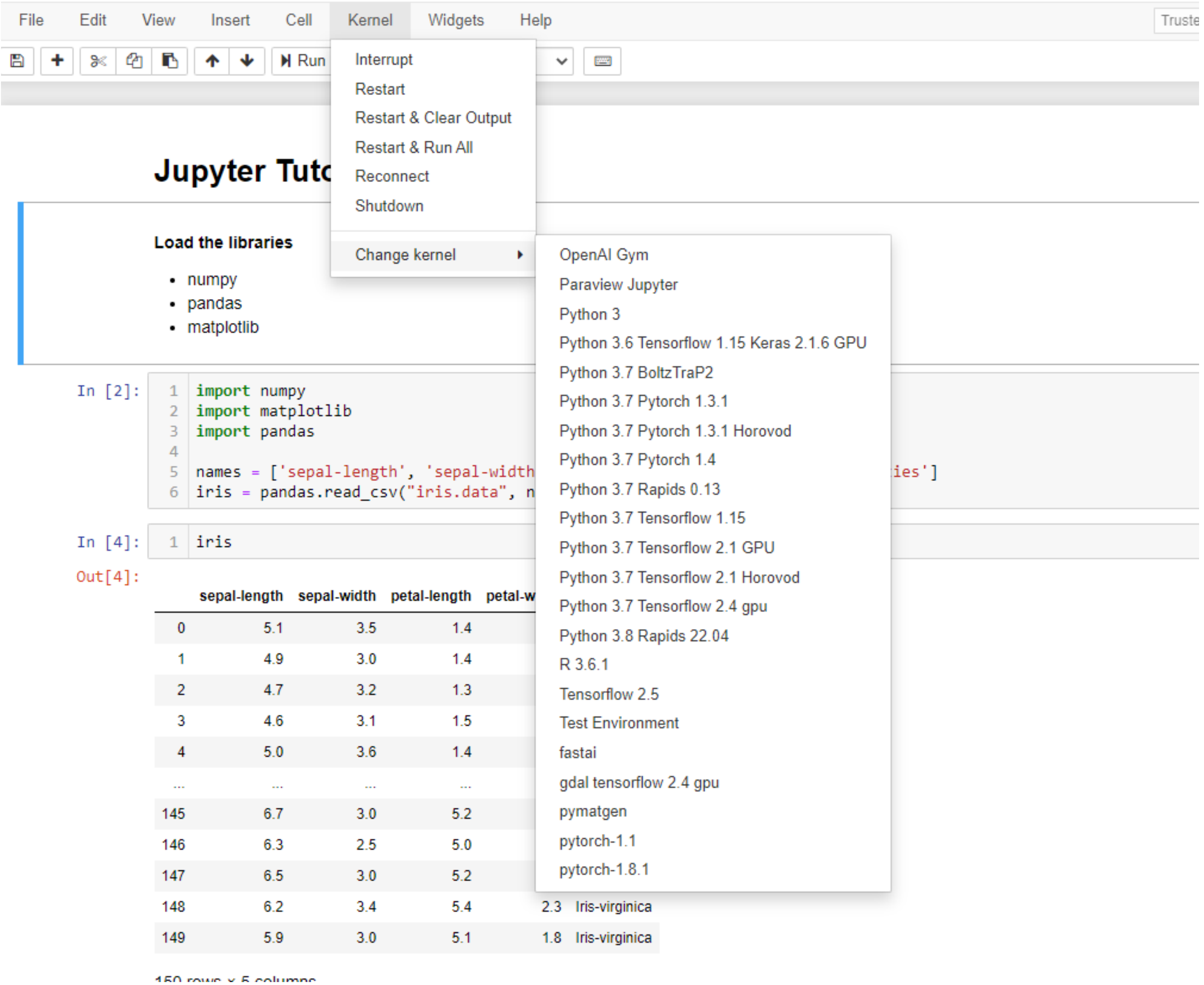


Jupyter Tutorial

Load the libraries

- numpy
- pandas
- matplotlib

Kernels



Jupyter Tuto

Load the libraries

- numpy
- pandas
- matplotlib

In [2]:

```
import numpy
import matplotlib
import pandas

names = ['sepal-length', 'sepal-width                                ies']
iris = pandas.read_csv("iris.data", n
```

In [4]:

```
iris
```

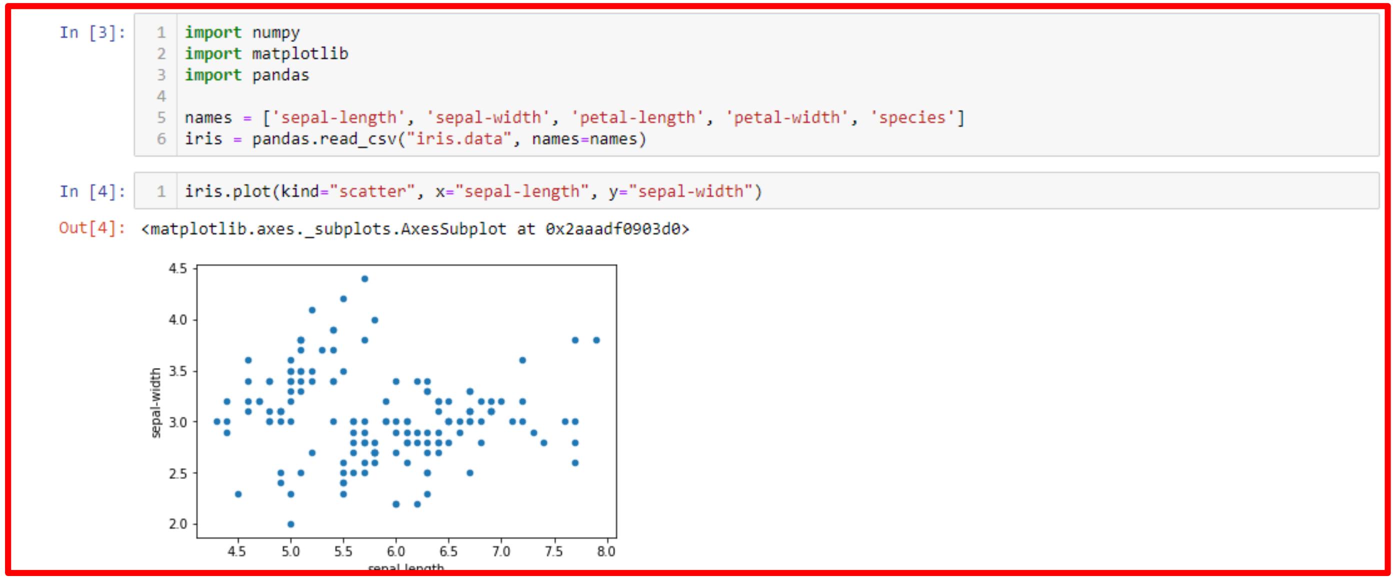
Out[4]:

	sepal-length	sepal-width	petal-length	petal-w	
0	5.1	3.5	1.4		
1	4.9	3.0	1.4		
2	4.7	3.2	1.3		
3	4.6	3.1	1.5		
4	5.0	3.6	1.4		
...	...	...	...		
145	6.7	3.0	5.2		
146	6.3	2.5	5.0		
147	6.5	3.0	5.2		
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Step 5: Saving a Jupyter Project

- Will auto save
- Can also clear cells

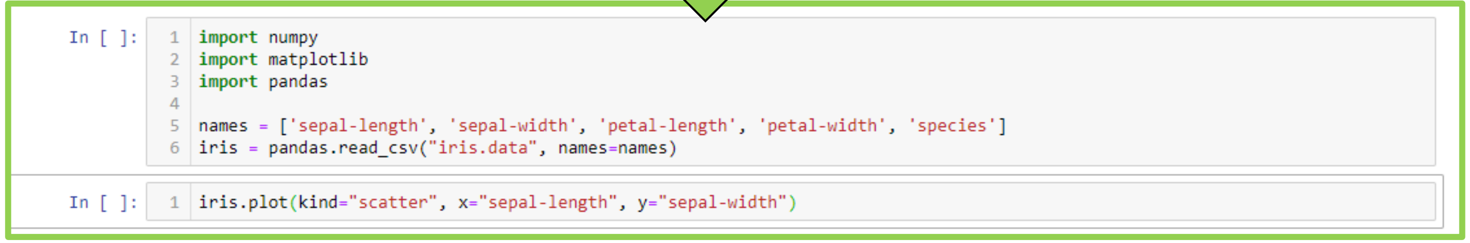


In [3]:
```
import numpy
import matplotlib
import pandas

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'species']
iris = pandas.read_csv("iris.data", names=names)
```

In [4]:
```
iris.plot(kind="scatter", x="sepal-length", y="sepal-width")
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aaadf0903d0>
```



In [ ]:
```
import numpy
import matplotlib
import pandas

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'species']
iris = pandas.read_csv("iris.data", names=names)
```

In [ ]:
```
iris.plot(kind="scatter", x="sepal-length", y="sepal-width")
```

Questions?

- hpcsupport@inl.gov

May 26, 2022

Shane Grover
HPC Storage Administrator

Idaho National Laboratory

HPC Storage

https://hpcweb.hpc.inl.gov/home/storage

Idaho National Laboratory

Home Directories

- DELL/EMC Isilon storage system
- 2.11 PB of storage
- 12 x 40 GbE connections
- Disk Quota Limits for Home Directories
- Backed up for disaster recovery
- Uses snapshots for quick file recovery
- Slowest storage

Scratch

- IBM ESS
- Uses Spectrum Scale/gpfs on Sawtooth
  - Lemhi uses NFS
- 1 PB of storage
- No Disk Quotas
- Files are deleted after 90 days
- Not backed up
- No snapshots
- Fast storage – IO heavy
- Will be updating the system 2022 – More through-put and 2 PB of storage

Ram Disk and local SSD

- Sawtooth
  - /dev/shm – 94 GB
- Lemhi
  - /tmp SSDs – 155 GB
  - /dev/shm – 94 GB
- Hoodoo
  - /local_storage – 1.8 TB
  - /dev/shm – 252 GB
- Space will be limited
- Volatile – The data will be lost if the node goes down

INL

Idaho National Laboratory

INL S22S - AI/ML Competition

- INL will host an artificial intelligence and machine learning competition this summer, the Summer 2022 Symposium (S22S).
- Come show off your skills and submit your results. Prizes will be handed out for the top performers.
- This symposium will consist of six one-hour sessions, which are split into half theory/instruction and half questions and answers. Participants may join for either or both parts of a session.
- We will be reviewing the concepts we taught in last year's S21S symposium and let you use those skills to compete for the top prize.
- This free professional development opportunity is available to only INL staff and interns. The sessions will be held on Wednesdays from 1 to 2 p.m. MT from June 15 to July 27.

June 15	June 22	June 29	July 13	July 20	July 27
Quick Review of S21S, including how to request HPC access and how to use Jupyter Notebooks inside of the HPC enclave.	Review models like Random Forest, Regression, and Support Vector Machines. Go over the warm-up data set.	Review answers from warm-up data set. Introduce the competition data set. Discuss rules and how we plan on scoring the results.	Review Neural Networks, including how to build a simple neural network on the competition data set.	Question and Answer session.	Review results and announce winners.

The competition team is led by Cody Walker, Jacob Farber and Shad Staples.
For more information or to register please contact Shad Staples.

NS&T ML-AI

Big Data Machine Learning Artificial Intelligence

Thank you