Intelligence Processing Unit (IPU) Training Labs

Zhenhua He

09/27/2022

ACES
ACCELERATING COMPUTING FOR EMERGING SCIENCES

High Performance Research Computing
DIVISION OF RESEARCH

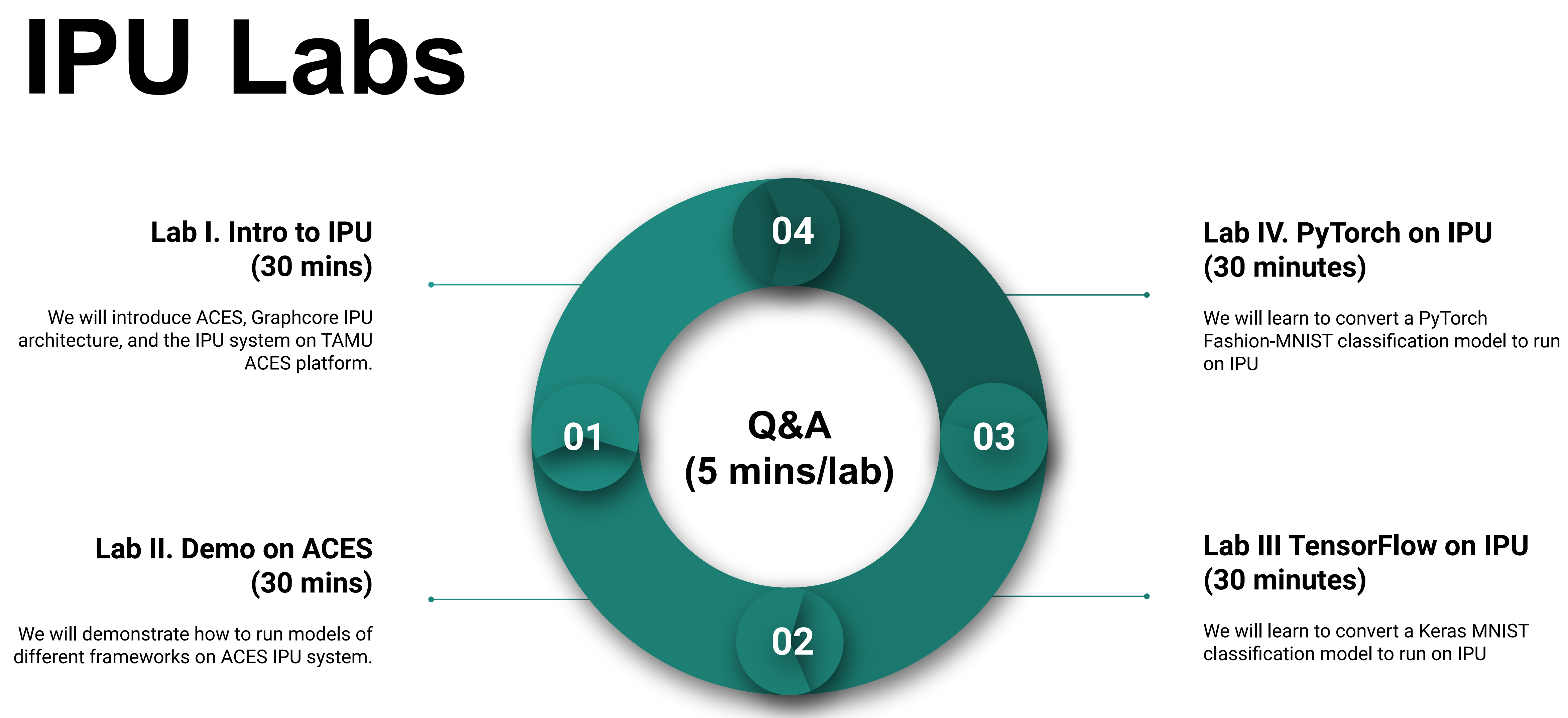


Figure 1. Structure of the IPU Training Laboratories.

Lab I. Intro to Graphcore IPU

ACES
ACCELERATING COMPUTING
FOR EMERGING SCIENCES

ACES - Accelerating Computing for Emerging Sciences

ACES is an innovative advanced computational prototype to be developed by Texas A&M University partnering with TACC and UIUC.

ACES
ACCELERATING COMPUTING FOR EMERGING SCIENCES

HPC wire

Since 1987 - Covering the Fastest Computers in the World and the People Who Run Them

- Home
- Technologies
- Sectors
- COVID-19
- AI/ML/DL
- Exascale
- Specials
- Resource Library
- Podcast
- Events
- Job Bank
- About
- Our Authors
- Solution Channels
- Subscribe

September 23, 2021

As Moore's law slows, HPC developers are increasingly looking for speed gains in specialized code and specialized hardware – but this specialization, in turn, can make testing and deploying code trickier than ever. Now, researchers from Texas A&M University, the University of Illinois at Urbana-Champaign and the University of Texas at Austin have teamed, with NSF funding, to build a $5 million prototype supercomputer ("ACES") with a dynamically configurable smörgåsbord of hardware, aiming to support developers as hardware needs grow ever more diverse.

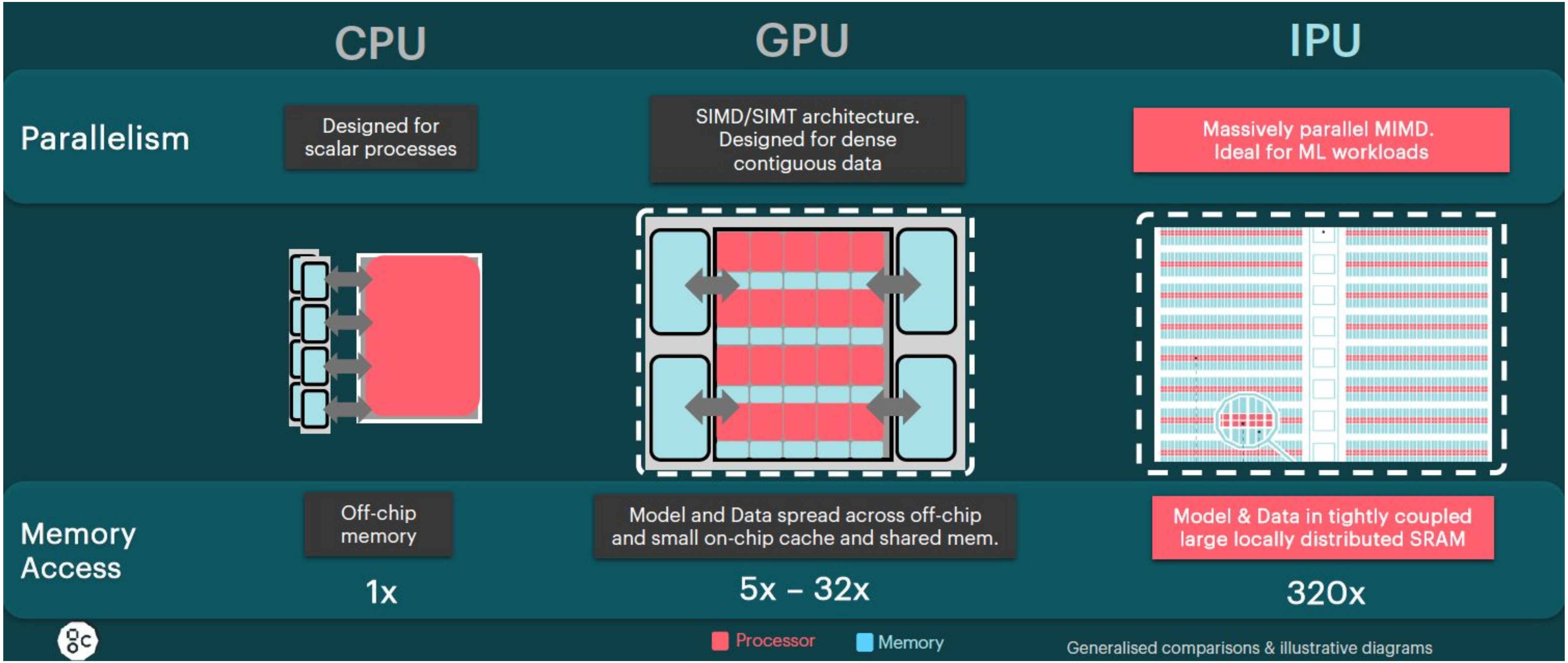
ACES (short for "Accelerating Computing for Emerging Sciences") is presented as an "innovative composable hardware platform." ACES will leverage a PCIe-based composable framework from Liqid to offer access to Intel's high-bandwidth memory Sapphire Rapids processors and more than 20 accelerators: Intel FPGAs; NEC Vector Engines; NextSilicon co-processors; Graphcore IPUs (Intelligence Processing Units); and Intel's forthcoming Ponte Vecchio GPUs. All this hardware will be coupled with Intel Optane memory and DDN Lustre Storage and connected with Mellanox NDR 400Gbps networking.

"ACES will enable applications and workflows to dynamically integrate the different accelerators, memory, and in-network computing protocols to glean new insights by rapidly processing large volumes of data," the NSF grant reads, "and provide researchers with a unique platform to produce complex hybrid programming models that effectively supports calculations that were not feasible before."

https://www.hpcwire.com/2021/09/23/three-universities-team-for-nsf-funded-aces-reconfigurable-supercomputer-prototype/

This project is supported by NSF award #2112356

Intelligence Processing Unit (IPU)



Source: Graphcore

BOW IPU PROCESSOR

Deep Trench Capacitor

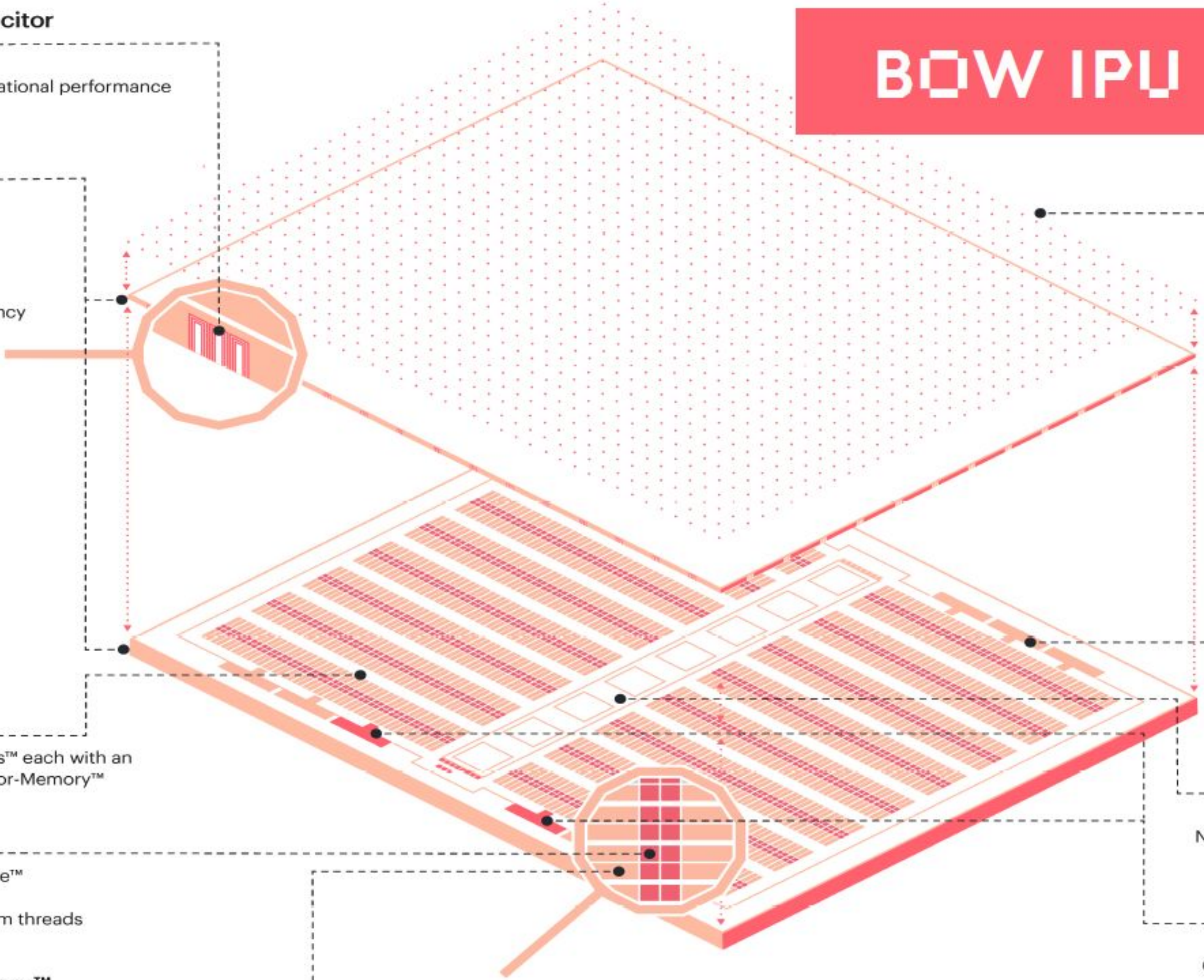
Efficient power delivery
Enables increase in operational performance

Wafer-On-Wafer

Advanced silicon 3D stacking technology

Closely coupled power delivery die

Higher operating frequency and enhanced overall performance



Solder Bumps

IPU-Links™

10x IPU-Links,
320GB/s chip to chip bandwidth

IPU-Exchange™

11 TB/s all to all IPU-Exchange™
Non-blocking, any communication pattern

PCIe

PCI Gen4 x16
64 GB/s bidirectional bandwidth to host

IPU-Tiles™

1472 independent IPU-Tiles™ each with an IPU-Core™ and In-Processor-Memory™

IPU-Core™

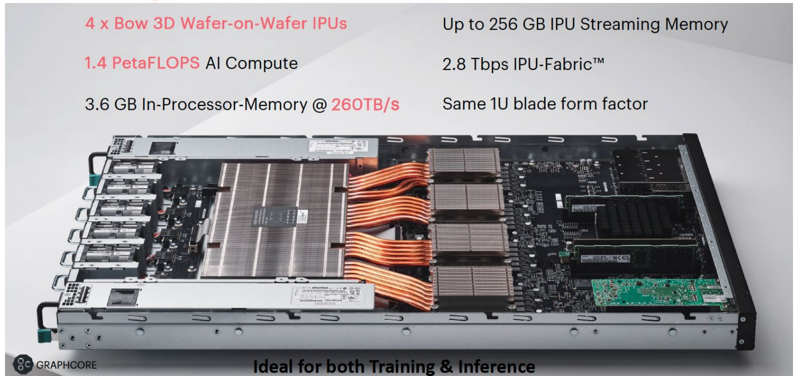
1472 independent IPU-Core™

8832 independent program threads executing in parallel

In-Processor-Memory™

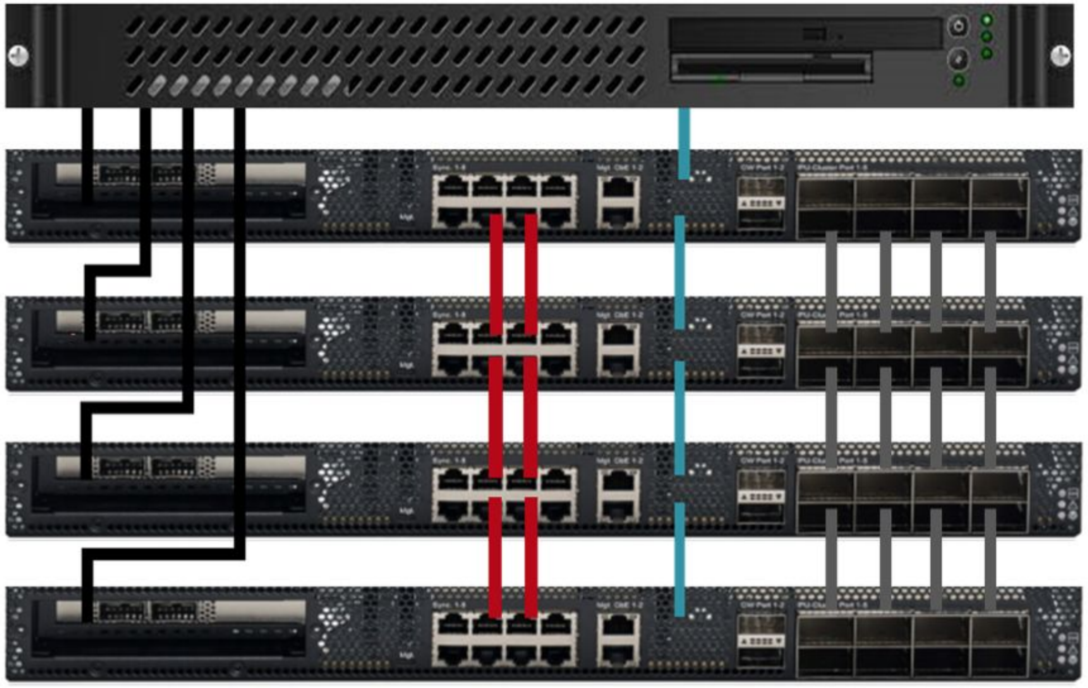
900MB In-Processor-Memory™ per IPU

65.4TB/s memory bandwidth per IPU



Source: Graphcore

BOW IPU-POD16 on ACES



Host-Link 100GE network interface

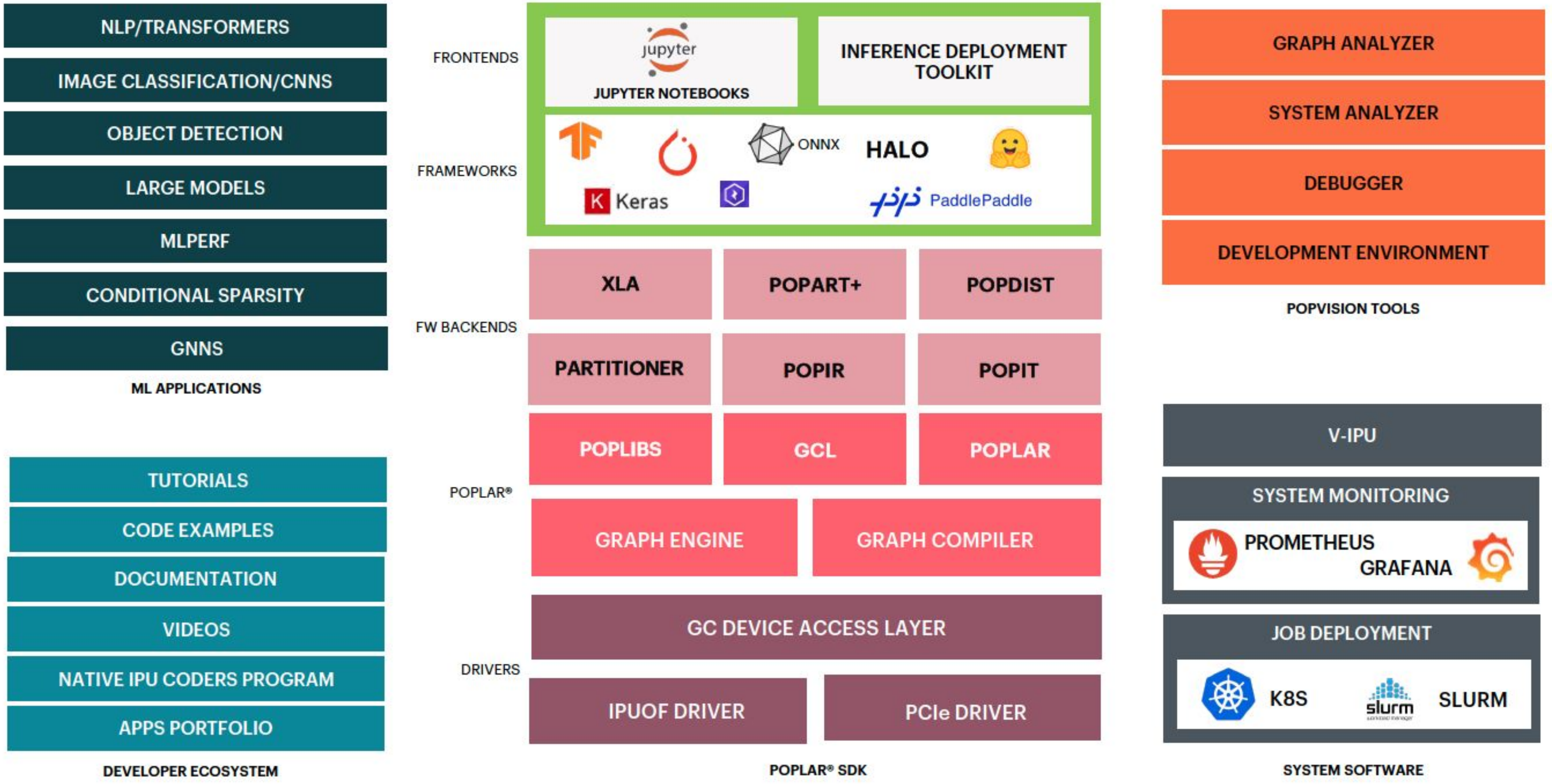
1GbE Management

Sync-Link

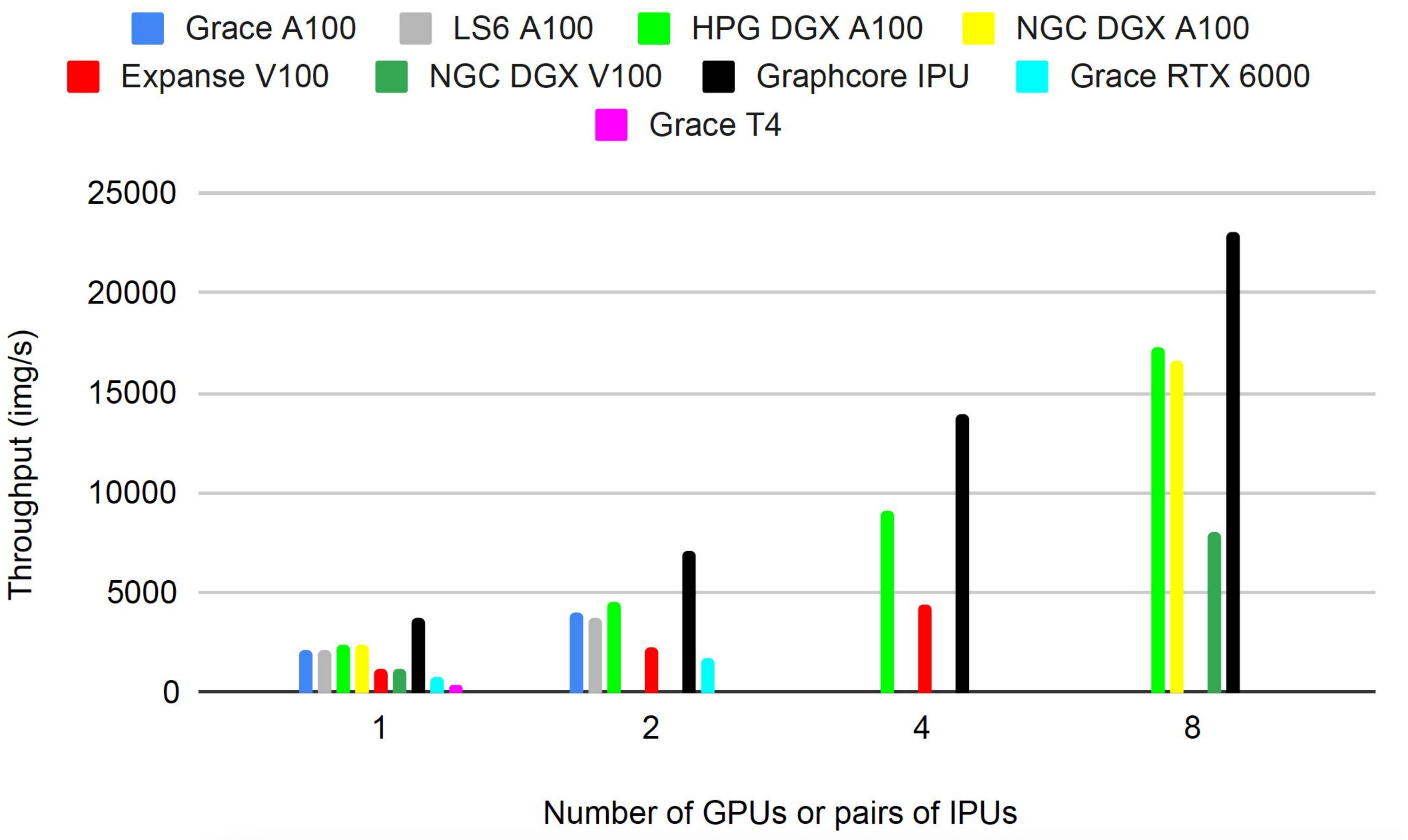
IPU-Link

Source: Graphcore

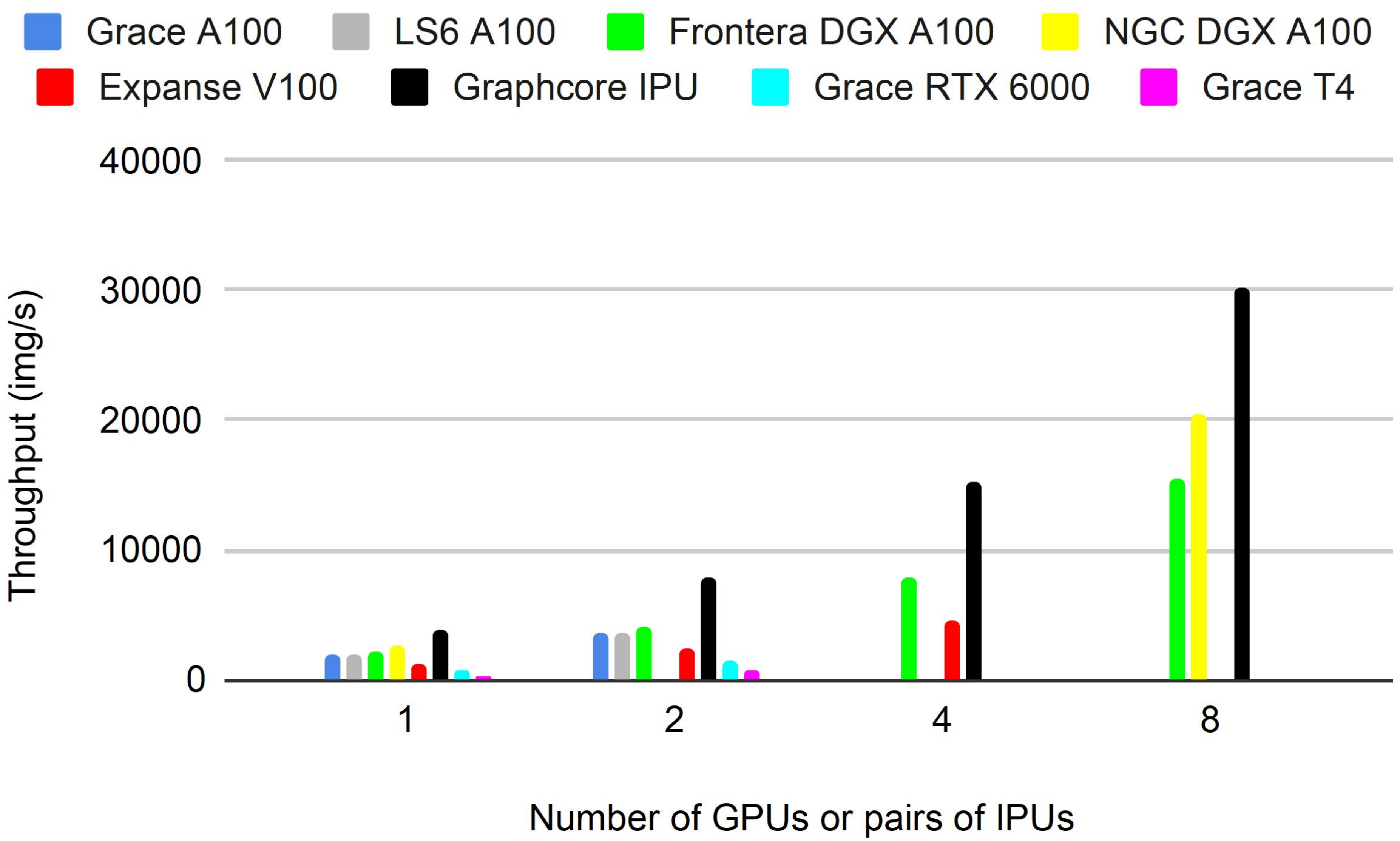
Graphcore Software Stack



Source: Graphcore



- Comparison of GPUs and IPUs on the ResNet 50 model in PyTorch with mixed precision.
- Scaling is similar for GPUs and IPUs, with similar magnitude per-device.



- Comparison of GPUs and IPUs on the ResNet 50 model in TensorFlow-1 with mixed precision.
- Scaling is similar for GPUs and IPUs, with similar magnitude per-device.

Lab II. Demo on ACES

ACES

ACCELERATING COMPUTING FOR EMERGING SCIENCES

ACES

Mission:

- Offer an accelerator testbed for numerical simulations and **AI/ML**
- Provide consulting, technical guidance, and training to researchers
- Collaborate on computational and data-enabled research.

Credit: towardsdatascience.com

Accessing ACES: via SSH

- SSH command is required for accessing FASTER:
  - On campus: `ssh userNetID@faster.hprc.tamu.edu`
  - Off campus:
    - Set up and start VPN (Virtual Private Network): u.tamu.edu/VPnetwork
    - Then: `ssh userNetID@faster.hprc.tamu.edu`
  - *Two-Factor Authentication* enabled for CAS, VPN, SSH
- SSH programs for Windows:
  - MobaXTerm (preferred, includes SSH and X11)
  - PuTTY SSH
  - Windows Subsystem for Linux
- SSH programs for MacOS:
  - Terminal
- https://portal-faster.hprc.tamu.edu/
  - Select the “Clusters” tab and then “_faster Shell Access”
- FASTER has 4 login nodes. Check the bash prompt.

https://hprc.tamu.edu/wiki/HPRC:Access#Access_Using_SSH

Login sessions that are idle for **60** minutes will be closed automatically
Processes run longer than **60** minutes on login nodes will be killed automatically.
Do not use more than 8 cores on the login nodes!
Do not use the sudo command.

Two-Factor Authentication

- Duo NetID two-factor authentication to enhance security (it.tamu.edu/duo/)
  - All web login (howdy, portal.hprc.tamu.edu, Globus) through CAS
  - VPN to TAMU campus (since Oct 1st, 2018)
  - SSH/SFTP to HPRC clusters (since Nov 4th, 2019)

- See instructions in two-factor wiki page (hprc.tamu.edu/wiki/Two_Factor)
- SSH clients work with Duo
  - ssh command from Linux, macOS Terminal, Windows cmd
  - MobaXterm for Windows (click on "Session" icon or via local session: hit "enter" 3 times and wait for "Password:" prompt)
  - Putty for Windows
- SFTP clients work with Duo
  - scp/sftp command from Linux, macOS Terminal, Windows cmd
  - WinSCP for Windows
  - Cyberduck for macOS

- Not all software supports SSH+Duo: SFTP in Matlab

hprc.tamu.edu/wiki/Two_Factor

Example: SSH login with Duo

```
$ ssh userNetID@faster.hprc.tamu.edu
******************************************************
…. warning message (snipped) …….
******************************************************

Password:
Duo two-factor login for userNetID

Enter a passcode or select one of the following options:

 1. Duo Push to XXX-XXX-1234
 2. Phone call to XXX-XXX-1234
 3. SMS passcodes to XXX-XXX-1234

Passcode or option (1-3): 1
Success. Logging you in...
```

Accessing ACES: via ACCESS

- The Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support (ACCESS) is a virtual collaboration funded by the National Science Foundation that facilitates free, customized access to advanced digital resources, consulting, training, and mentorship.

- View the getting started documentation to create an ACCESS account.
  - https://access-ci.atlassian.net/wiki/spaces/ACCESSdocumentation/pages/76743011/Getting+Started

- SSH via Jump Host:
  - `ssh -J fasterusername@faster-jump.hprc.tamu.edu:8822 fasterusername@login.faster.hprc.tamu.edu`

https://access-ci.org/

Hands-On Session 1

- Please try to access ACES via FASTER now.

- What message do you see when you log on?

Demos

Tutorials can be found on the TAMU HPRC Wiki

https://hprc.tamu.edu/wiki/ACES#Graphcore_IPUs

and covers

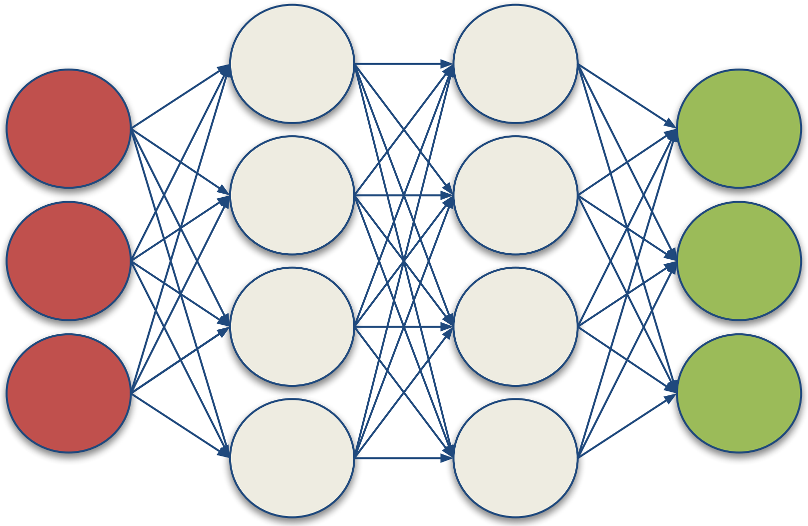
- PyTorch (PopTorch)
- TensorFlow 1
- TensorFlow 2

Lab III. TensorFlow on IPU

Deep Learning
by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
http://www.deeplearningbook.org/

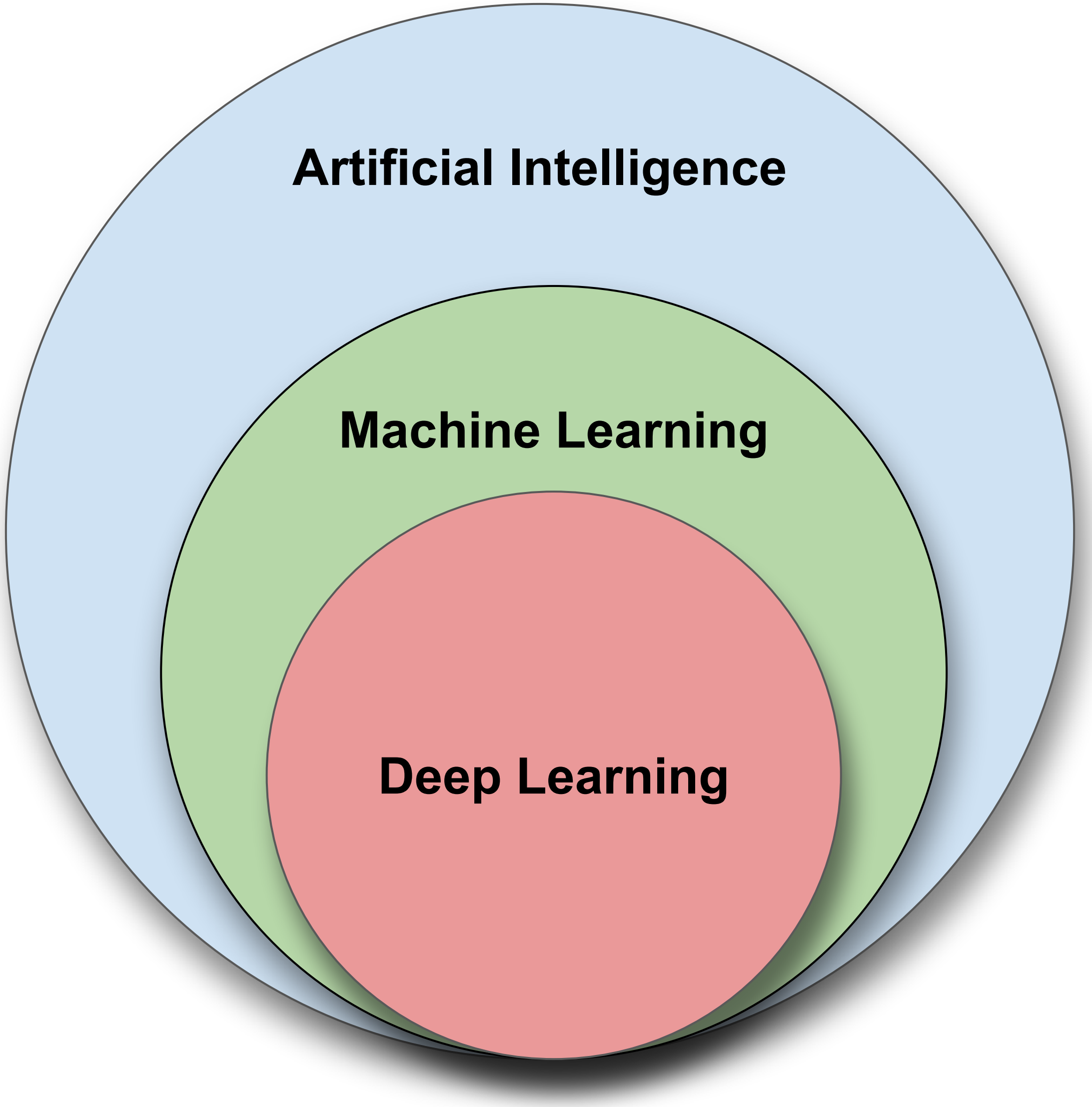
Animation of Neutron Networks
by Grant Sanderson
https://www.3blue1brown.com/

Visualization of CNN
by Adam Harley
https://adamharley.com/nn_vis/cnn/3d.html



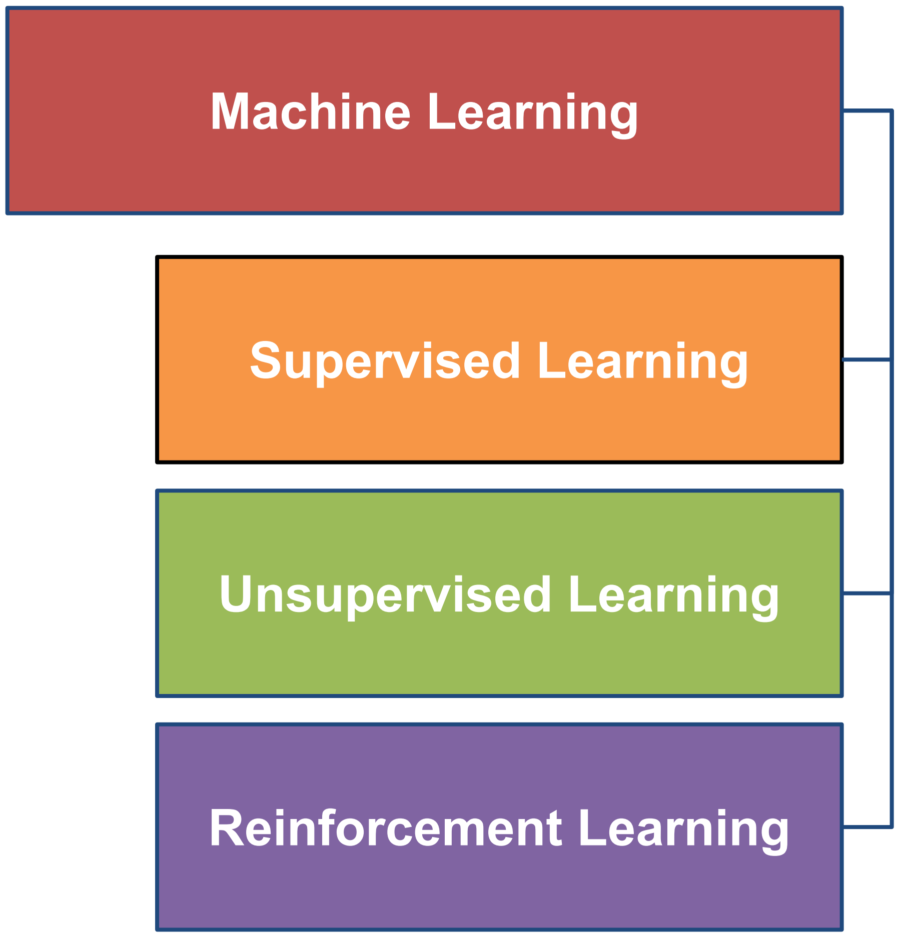
Relationship of AI, ML, and DL

- **Artificial Intelligence (AI)** is anything about man-made intelligence exhibited by machines.
- **Machine Learning (ML)** is an approach to achieve **AI**.
- **Deep Learning (DL)** is one technique to implement **ML**.

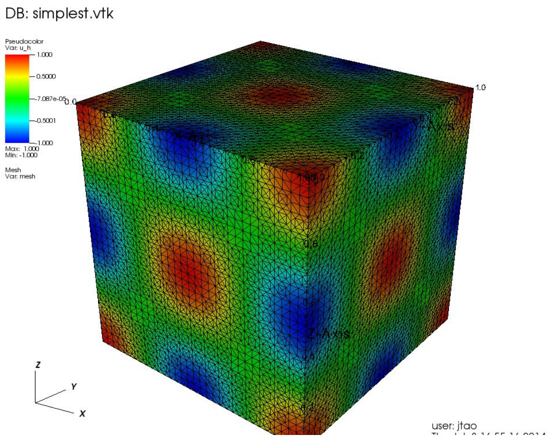


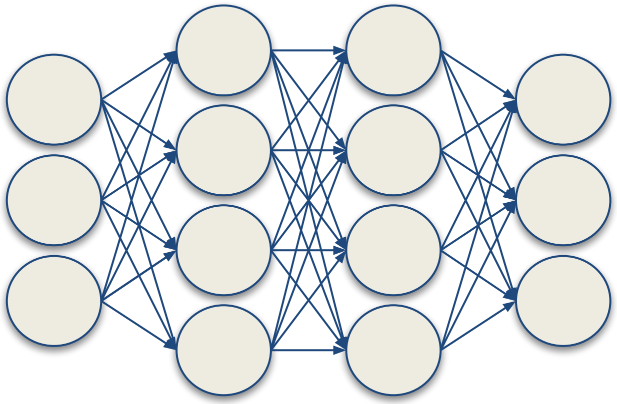
Types of ML Algorithms

- **Supervised Learning**
  - trained with labeled data; including regression and classification problems
- **Unsupervised Learning**
  - trained with unlabeled data; clustering and association rule learning problems.
- **Reinforcement Learning**
  - no training data; stochastic Markov decision process; robotics and business strategy planning.

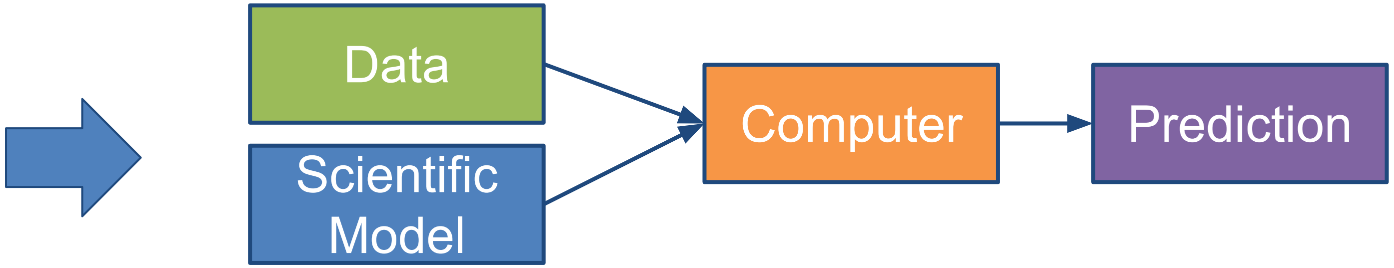


Machine Learning

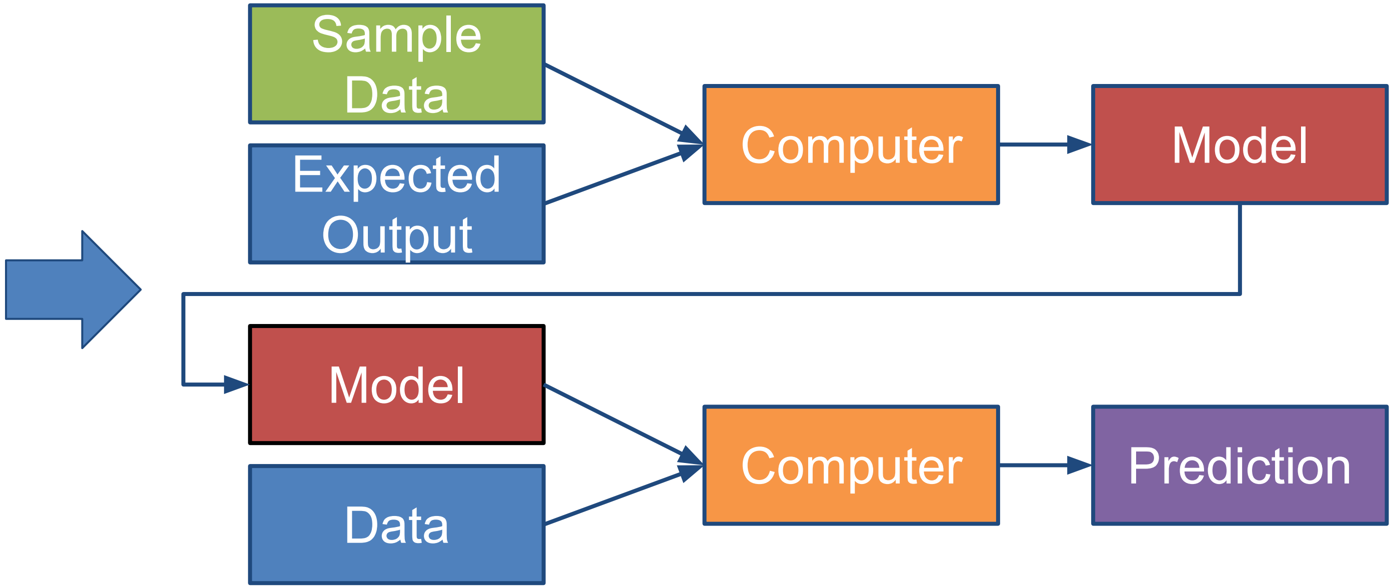




Traditional Modeling



Machine Learning (Supervised Learning)



Inputs and Outputs

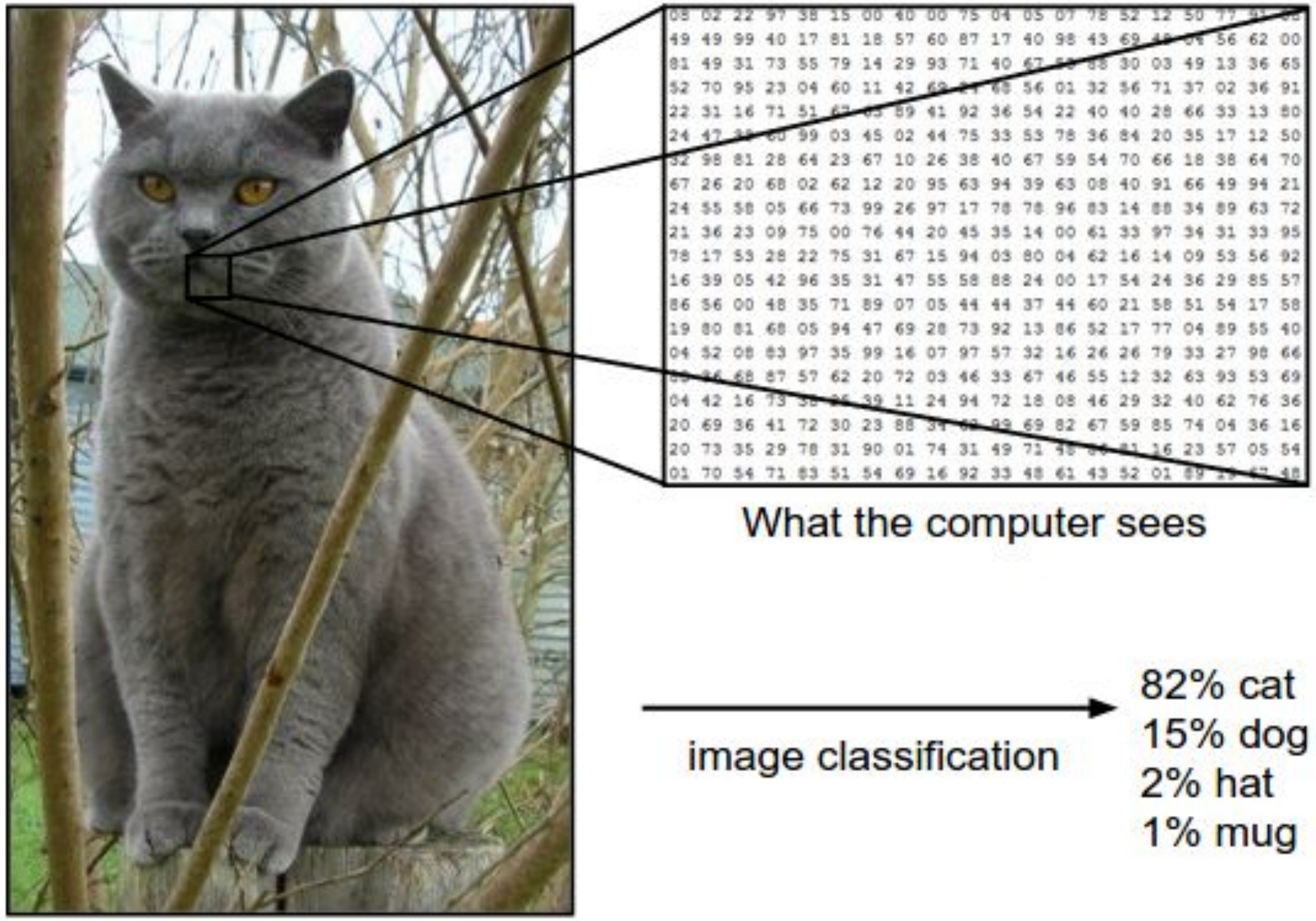
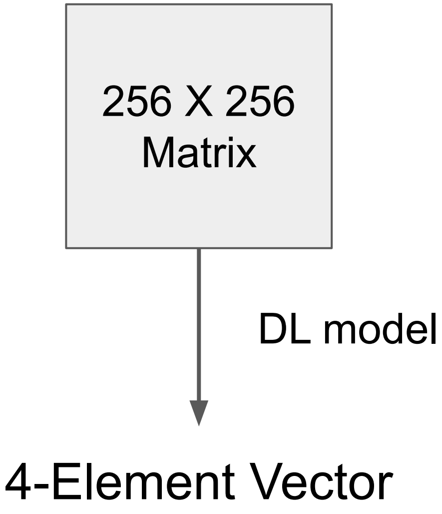
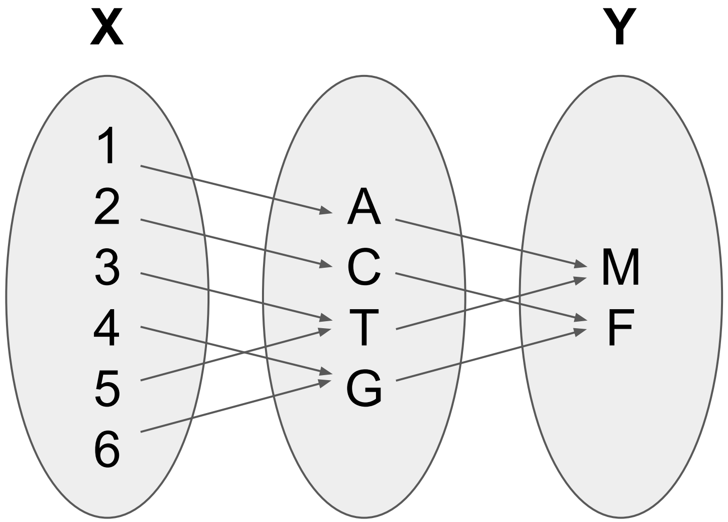


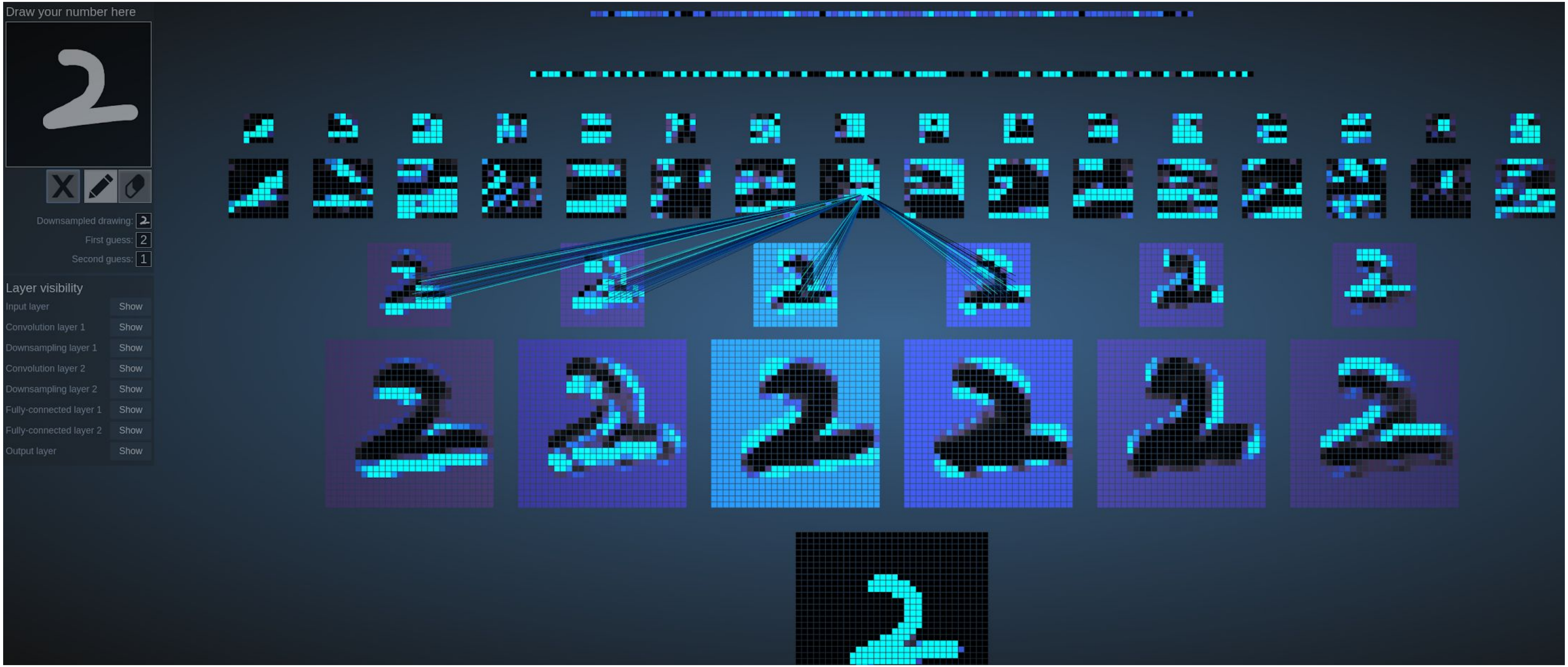
Image from the Stanford CS231 Course





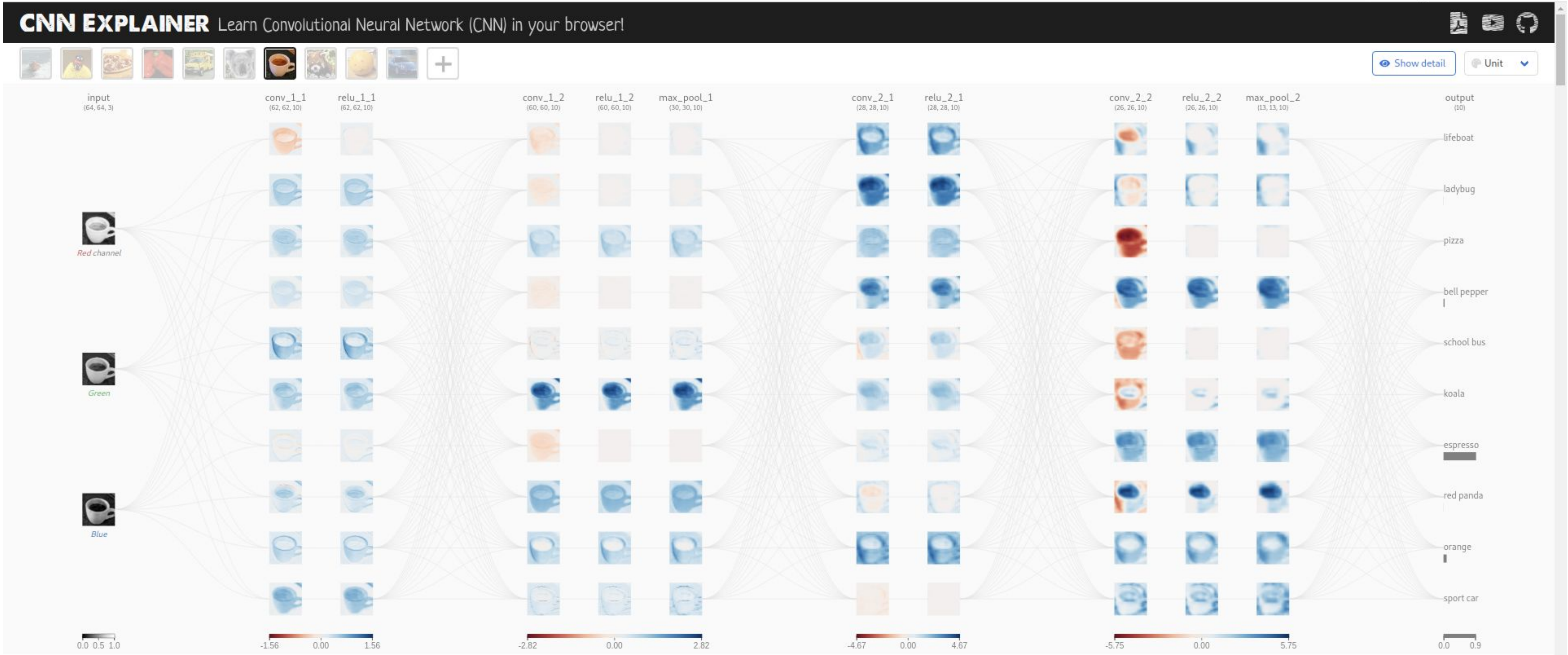
With deep learning, we are searching for a **surjective** (or **onto**) function **f** from a set **X** to a set **Y**.

MNIST - CNN Visualization



(Image Credit: https://adamharley.com/nn_vis/cnn/3d.html)

CNN Explainer



(Image Credit: https://poloclub.github.io/cnn-explainer/)

Machine Learning Workflow with Keras

Step 1

Prepare Train Data

The preprocessed data set needs to be shuffled and split into training and testing data.

Step 2

Define Model

A model could be defined with Keras Sequential model for a linear stack of layers or Keras functional API for a complex network.

Step 3

Training Configuration

The configuration of the training process requires the specification of an optimizer, a loss function, and a list of metrics.

Step 4

Train Model

The training begins by calling the fit function. The number of epochs and the batch size need to be set. The measurement metrics need to be evaluated.

1. Import the TensorFlow IPU module

Add the following import statement to the beginning of your script:

```
from tensorflow.python import ipu
```

2. Preparing the dataset

- Make sure the sizes of our datasets are divisible by the batch size

```
def make_divisible(number, divisor):
    return number - number % divisor
```

- Adjust dataset lengths

```
(x_train, y_train), (x_test, y_test) = load_data()
train_data_len = x_train.shape[0]
train_data_len = make_divisible(train_data_len, batch_size)
x_train, y_train = x_train[:train_data_len], y_train[:train_data_len]
test_data_len = x_test.shape[0]
test_data_len = make_divisible(test_data_len, batch_size)
x_test, y_test = x_test[:test_data_len], y_test[:test_data_len]
```

3. Add IPU configuration

To use the IPU, you must create an IPU session configuration:

```
ipu_config = ipu.config.IPUConfig()
ipu_config.auto_select_ipus = 1
ipu_config.configure_ipu_system()
```

A full list of configuration options is available in the API documentation.

4. Specify IPU strategy

```
strategy = ipu.ipu_strategy.IPUStrategy()
```

The `tf.distribute.Strategy` is an API to distribute training and inference across multiple devices. `IPUStrategy` is a subclass which targets a system with one or more IPUs attached.

5. Wrap the model within the IPU strategy scope

- Creating variables and Keras models within the scope of the `IPUStrategy` object will ensure that they are placed on the IPU.
- To do this, we create a `strategy.scope()` context manager and move all the model code inside it.

Hands-on Session 2

- Log into the poplar1 IPU system
    - *ssh poplar1*
    - *cd localdata*
    - *source venv_tf2/bin/activate*
- Clone a copy of IPU-Training GitHub repo
    - *git clone https://github.com/happidence1/IPU-Training.git*
    - *cd IPU-Training/Keras*
- Complete the **#Todo**s in the mnist-ipu-todo.py file.
- Run it in the **venv_tf2** virtual environment.
    - *python mnist-ipu-todo.py*
- After finishing the job, you can deactivate the virtual environment
    - *deactivate*

Lab IV. PyTorch on IPU

PopTorch

- PopTorch is a set of extensions for PyTorch released by Graphcore to enable PyTorch models to run on Graphcore's IPU hardware.

- PopTorch will use PopART to parallelise the model over the given number of IPUs. Additional parallelism can be expressed via a replication factor, which enables you to data-parallelise the model over more IPUs.

Training a model on IPU

- Import the packages

```
import torch
import poptorch
import torchvision
import torch.nn as nn
import matplotlib.pyplot as plt
from tqdm import tqdm
from sklearn.metrics import accuracy_score
```

Load the data

PopTorch offers an extension of `torch.utils.data.DataLoader` class with its `poptorch.DataLoader` class, specialized for the way the underlying PopART framework handles batching of data.

Build the model

```
class ClassificationModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 5, 3)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(5, 12, 5)
        self.norm = nn.GroupNorm(3, 12)
        self.fc1 = nn.Linear(972, 100)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(100, 10)
        self.log_softmax = nn.LogSoftmax(dim=1)
        self.loss = nn.NLLLoss()
```

```
def forward(self, x, labels=None):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.norm(self.relu(self.conv2(x)))
        x = torch.flatten(x, start_dim=1)
        x = self.relu(self.fc1(x))
        x = self.log_softmax(self.fc2(x))
        # The model is responsible for the
calculation of the loss when using an IPU. We do
it this way:
        if self.training:
            return x, self.loss(x, labels)
        return x


model = ClassificationModel()
model.train()
```

Prepare training for IPUs

The compilation and execution on the IPU can be controlled using `poptorch.Options`. These options are used by PopTorch's wrappers such as `poptorch.DataLoader` and `poptorch.trainingModel`.

```
opts = poptorch.Options()
train_dataloader = poptorch.DataLoader(
    opts, train_dataset, batch_size=16, shuffle=True, num_workers=20
)
```

Train the model

```
optimizer = poptorch.optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

poptorch_model = poptorch.trainingModel(model, options=opts,
optimizer=optimizer)

epochs = 30
for epoch in tqdm(range(epochs), desc="epochs"):
    total_loss = 0.0
    for data, labels in tqdm(train_dataloader, desc="batches",
leave=False):
        output, loss = poptorch_model(data, labels)
        total_loss += loss

poptorch_model.detachFromDevice()

torch.save(model.state_dict(), "classifier.pth")
```

Evaluate the model

```
model = model.eval()

poptorch_model_inf = poptorch.inferenceModel(model, options=opts)

test_dataloader = poptorch.DataLoader(opts, test_dataset, batch_size=32,
num_workers=10)

predictions, labels = [], []
for data, label in test_dataloader:
    predictions += poptorch_model_inf(data).data.max(dim=1).indices
    labels += label

poptorch_model_inf.detachFromDevice()

print(f"Eval accuracy: {100 * accuracy_score(labels, predictions):.2f}%")
```

Hands-on Session 3

- Log into the poplar1 IPU system
  - *ssh poplar1*
  - *cd localdata*
  - *source poptorch_test/bin/activate*
- Clone a copy of IPU-Training GitHub repo (if cloned, just *cd PyTorch*)
  - *git clone https://github.com/happidence1/IPU-Training.git*
  - *cd IPU-Training/PyTorch*
- Complete the **#Todo**s in the fashion-mnist-pytorch-ipu-todo.py file.
- Run it in the **poptorch_test** virtual environment.
  - *pip install scikit-learn*
  - *python fashion-mnist-pytorch-ipu-todo.py*
- After finishing the job, you can deactivate the virtual environment
  - *deactivate*

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu
Phone: 979-845-0219

Help us help you. Please include details in your request for support, such as, Cluster (Faster, Grace, Terra, ViDaL), NetID (UserID), Job information (Job id(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.

References

- https://www.graphcore.ai/
- https://github.com/graphcore/tutorials/tree/master/tutorials/tensorflow2/keras
- https://github.com/graphcore/tutorials/tree/master/tutorials/pytorch/basics
- https://hprc.tamu.edu/wiki/Main_Page