

Introduction to Deep Learning with PyTorch

Jian Tao

jtao@tamu.edu

HPRC Short Course

10/29/2021



TEXAS A&M UNIVERSITY

Visualization



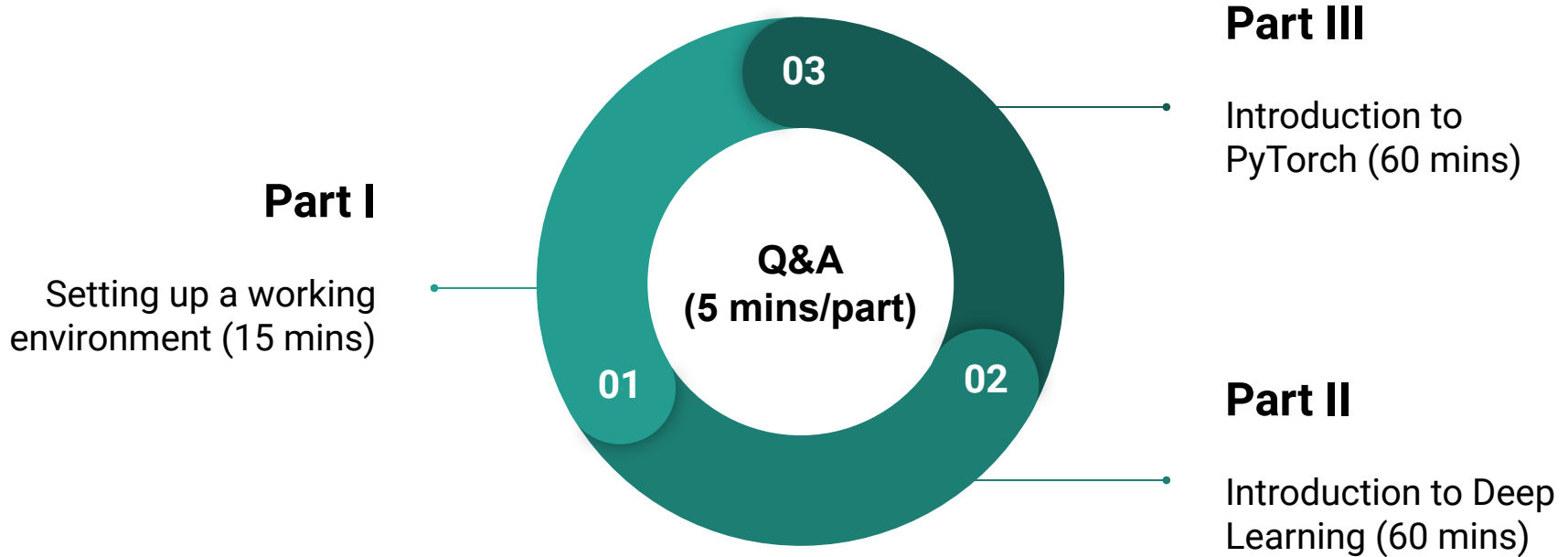
High Performance
Research Computing
DIVISION OF RESEARCH



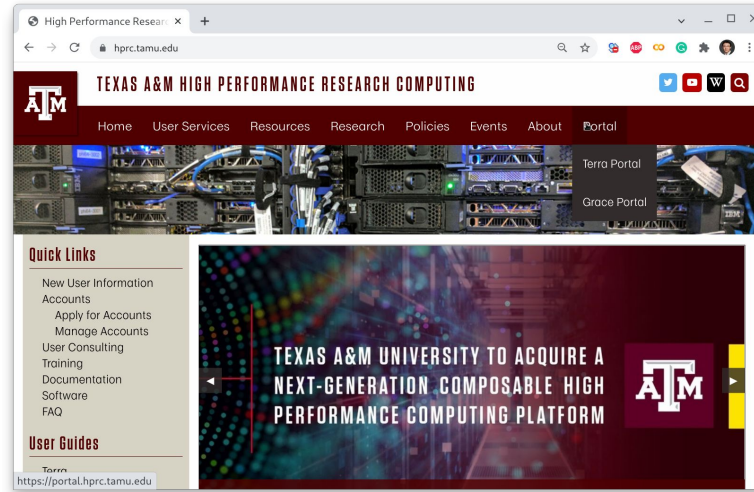
TEXAS A&M

Institute of
Data Science

Introduction to Deep Learning with PyTorch



Part I. Working Environment



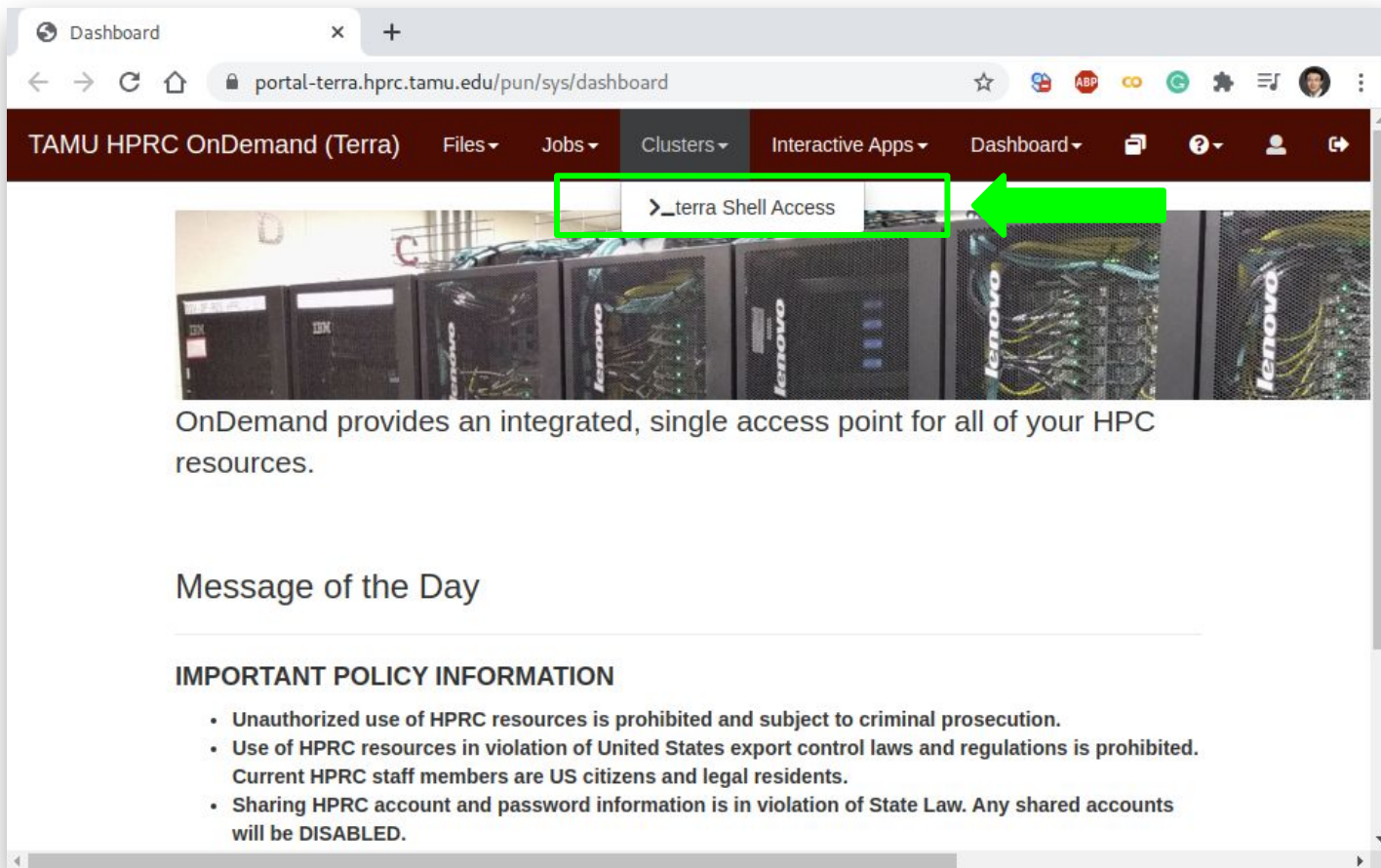
HPRC Portal

*** VPN is required for off-campus users.**

Login HPRC Portal (Terra)

The screenshot shows a web browser window with the URL `hprc.tamu.edu`. The page header includes the Texas A&M University logo and the text "TEXAS A&M HIGH PERFORMANCE RESEARCH COMPUTING". A navigation menu contains the following items: Home, User Services, Resources, Research, Policies, Events, About, and Portal. The "Portal" item is highlighted, and a dropdown menu is visible with two options: "Terra Portal" and "Grace Portal". A green arrow points to the "Terra Portal" link. Below the navigation menu is a "Quick Links" sidebar with the following items: New User Information, Accounts (Apply for Accounts, Manage Accounts), User Consulting, Training, Documentation, Software, and FAQ. Below the sidebar is a "User Guides" section with a link to "Terra". The main content area features a large banner with the text "TEXAS A&M UNIVERSITY TO ACQUIRE A NEXT-GENERATION COMPOSABLE HIGH PERFORMANCE COMPUTING PLATFORM" and the Texas A&M logo. The browser's address bar shows `https://portal.hprc.tamu.edu`.

Terra Shell Access - I



The screenshot shows a web browser window with the URL `portal-terra.hprc.tamu.edu/pun/sys/dashboard`. The navigation bar includes links for 'TAMU HPRC OnDemand (Terra)', 'Files', 'Jobs', 'Clusters', 'Interactive Apps', and 'Dashboard'. A dropdown menu is open under 'Clusters', and the option '>_terra Shell Access' is highlighted with a green box and a green arrow pointing to it. Below the navigation bar is a banner image of server racks with 'lenovo' branding. The main content area contains the text: 'OnDemand provides an integrated, single access point for all of your HPC resources.' Below this is a section titled 'Message of the Day' and a section titled 'IMPORTANT POLICY INFORMATION' with a bulleted list of rules.

Dashboard

portal-terra.hprc.tamu.edu/pun/sys/dashboard

TAMU HPRC OnDemand (Terra) Files Jobs Clusters Interactive Apps Dashboard

>_terra Shell Access

OnDemand provides an integrated, single access point for all of your HPC resources.

Message of the Day

IMPORTANT POLICY INFORMATION

- Unauthorized use of HPRC resources is prohibited and subject to criminal prosecution.
- Use of HPRC resources in violation of United States export control laws and regulations is prohibited. Current HPRC staff members are US citizens and legal residents.
- Sharing HPRC account and password information is in violation of State Law. Any shared accounts will be DISABLED.

Terra Shell Access - II

```
*****
This computer system and the data herein are available only for authorized
purposes by authorized users: use for any other purpose is prohibited and may
result in administrative/disciplinary actions or criminal prosecution against
the user. Usage may be subject to security testing and monitoring to ensure
compliance with the policies of Texas A&M University, Texas A&M University
System, and the State of Texas. There is no expectation of privacy on this
system except as otherwise provided by applicable privacy laws. Users should
refer to Texas A&M University Standard Administrative Procedure 29.01.03.M0.02,
Rules for Responsible Computing, for guidance on the appropriate use of Texas
A&M University information resources.
*****

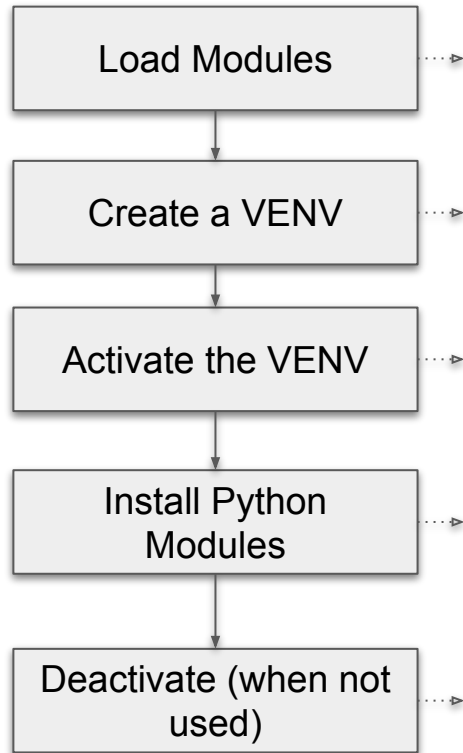
Password:
Duo two-factor login for jtao

Enter a passcode or select one of the following options:

  1. Duo Push to iPhone (iOS)
  2. Duo Push to iPad (iOS)

Passcode or option (1-2): 1
█
```

Python Virtual Environment (VENV)



```
# clean up and load Anaconda
cd $SCRATCH
module purge
module load Python/3.7.4-GCCcore-8.3.0

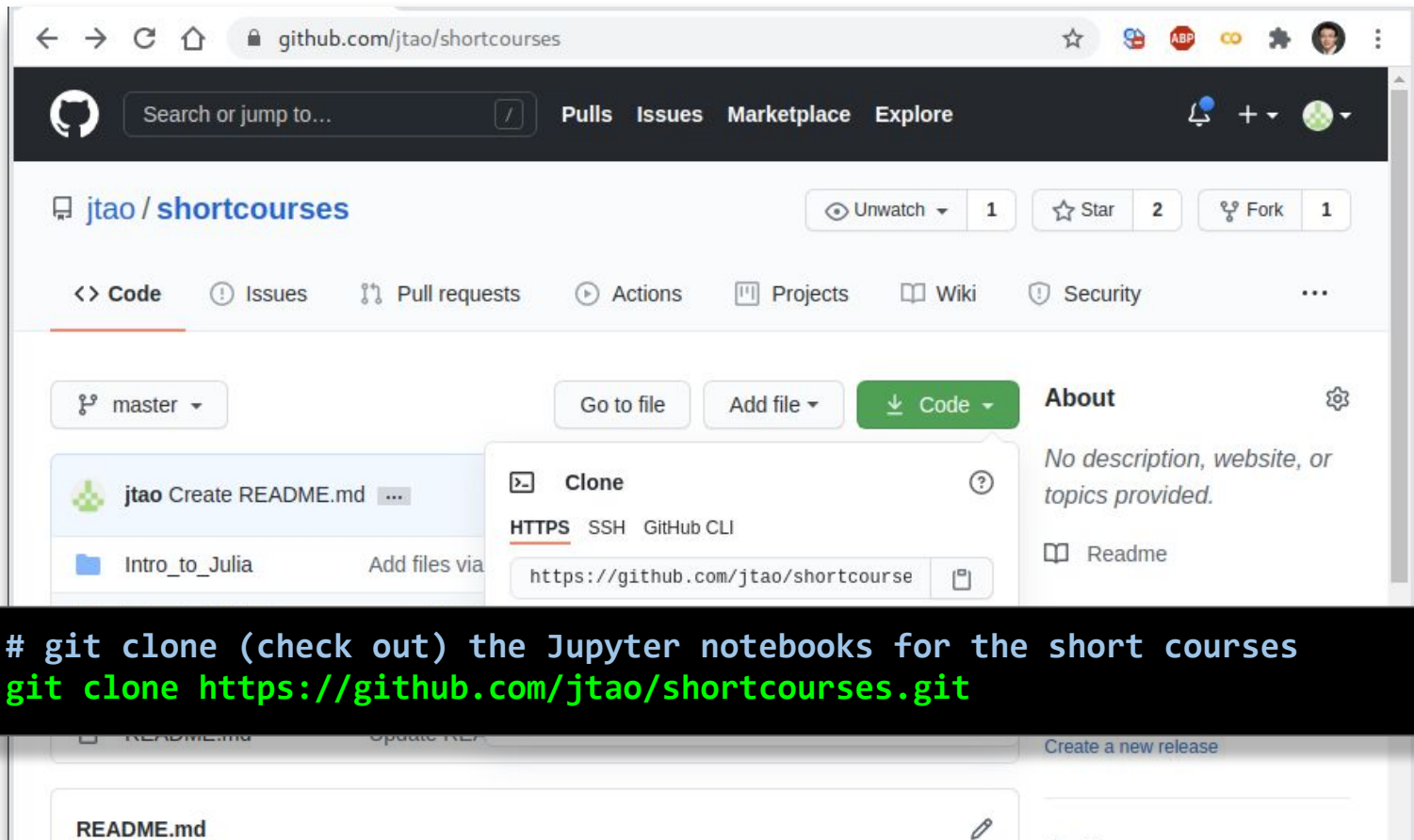
# create a Python virtual environment
python -m venv mylab

# activate the virtual environment
source mylab/bin/activate

# install required package to be used in the portal
pip install --upgrade pip setuptools
pip install jupyterlab torch torchvision tensorboard
pip install pandas scikit-plot tqdm seaborn

# deactivate the virtual environment
# deactivate
```

Check out Exercises



The image shows a screenshot of a GitHub repository page for 'jtao/shortcourses'. The page includes a search bar, navigation links for Pulls, Issues, Marketplace, and Explore, and repository statistics (Unwatch, Star, Fork). A 'Code' button is highlighted, and a 'Clone' dialog is open, showing the repository URL: `https://github.com/jtao/shortcourse`. A terminal overlay at the bottom displays the following commands:

```
# git clone (check out) the Jupyter notebooks for the short courses  
git clone https://github.com/jtao/shortcourses.git
```


Go to JupyterLab Page

The screenshot shows a web browser window with the URL `portal-terra.hprc.tamu.edu/pun/sys/dashboard`. The page features a sidebar menu with the following categories and items:

- GUI
 - ANSYS Workbench
 - Abaqus/CAE
 - LS-PREPOST
 - LS-PREPOST (workshop)
 - MATLAB
 - ParaView
 - VNC
- Imaging
 - Chimera
 - Coot
 - Diffusion Toolkit & TrackVis
 - FSL
 - Fiji
 - ICY
 - ImageJ
 - Vaa3D
 - cisTEM
- Servers
 - Jupyter Notebook
 - JupyterLab** (highlighted with a red box)
 - RStudio R 3.6.1
 - RStudio R 3.6.3
 - Spark-Jupyter Notebook

Other visible text on the dashboard includes:

- Message of the Day
- IMPORTANT POLICY INFORMATION
 - Unauthorized use of HPRC resources is prohibited and subject to...
 - Use of HPRC resources in violation of United States export control...
 - Sharing HPRC account and password information is in violation of...
 - Authorized users must also adhere to ALL policies at: <https://hprc.tamu.edu>
- !! WARNING: THERE ARE ONLY NIGHTLY BACKUPS OF USER HOME DIRECTORIES !!
- powered by **OPEN OnDemand**
- Current HPRC staff members are US citizens and legal residents. All other users will be DISABLED.
- Dashboard version: v1.32.0

The address bar at the bottom shows the URL: `https://portal-terra.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sys/jupyterlab/session_contexts/new`.

Set Virtual Environment

JupyterLab

portal-terra.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sys/jupyterlab/session_contexts/new

TAMU HPRC OnDemand (Terra) Files Jobs Clusters Interactive Apps Dashboard Develop Help Logged in as jtao Log Out

Home / My Interactive Sessions / JupyterLab

Interactive Apps

- BIO
- Beauti
- DIYABC
- FigTree
- IGV
- JBrowse
- Krait

JupyterLab

This app will launch a JupyterLab server on the Terra cluster.

Module

Python/3.7.4-GCCcore-8.3.0

Anaconda/3-x.x.x.x is Python3

Optional Conda Environment to be activated

/scratch/user/jtao/mylab/bin/activate

Enter the name of environment to be activated. Changing this field is optional.

Number of hours

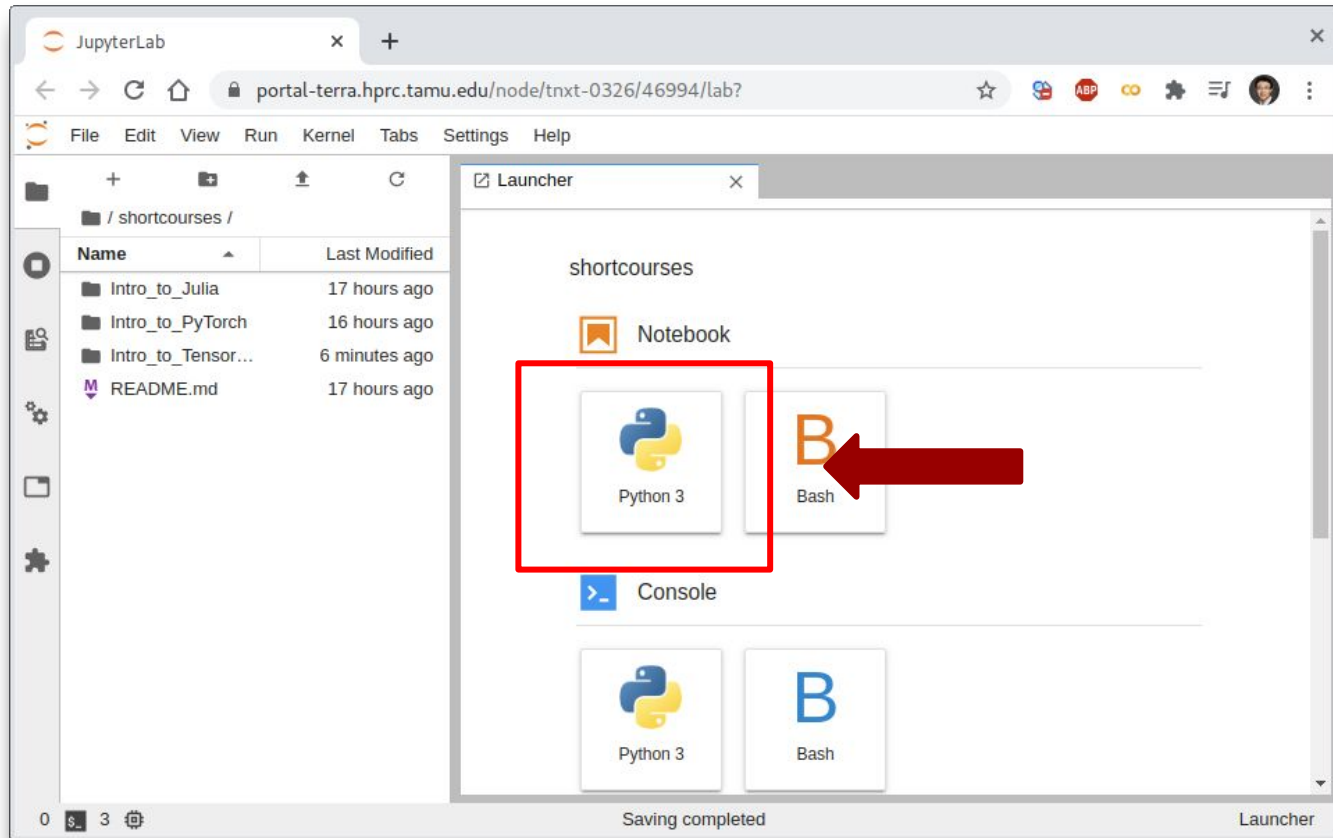
1

```
# enter the full path of the activate command of your virtualenv  
/scratch/user/YOURNETID/mylab/bin/activate
```

Connect to JupyterLab

The screenshot shows a web browser window with the URL `portal-terra.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sessions`. The page title is "My Interactive Sessions". A dark red navigation bar at the top contains the text "TAMU HPRC OnDemand (Terra)" and several menu items: "Files", "Jobs", "Clusters", "Interactive Apps", and "Dashboard". A user profile icon for "jtao" and a "Log Out" button are also present. A green notification box at the top left states "Session was successfully deleted." Below this, a breadcrumb trail shows "Home / My Interactive Sessions". On the left side, there is a sidebar titled "Interactive Apps" with a list of applications: "BIO", "Beauti", "DIYABC", "FigTree", "IGV", "JBrowse", and "Krait". The main content area displays a session for "JupyterLab (6119424)" with a status of "Running" and resource usage of "1 node | 1 core". The session details include: "Host: tnxt-0468", "Created at: 2020-11-12 01:49:27 CST", "Time Remaining: 56 minutes", and "Session ID: 5cbf368d-1a3a-4154-8689-ac13dcd1cdde". A red "Delete" button is located to the right of the session details. At the bottom of the session card, a blue button labeled "Connect to JupyterLab" is highlighted with a red rectangular box, and a large red arrow points to it from the right.

Create a Jupyter Notebook

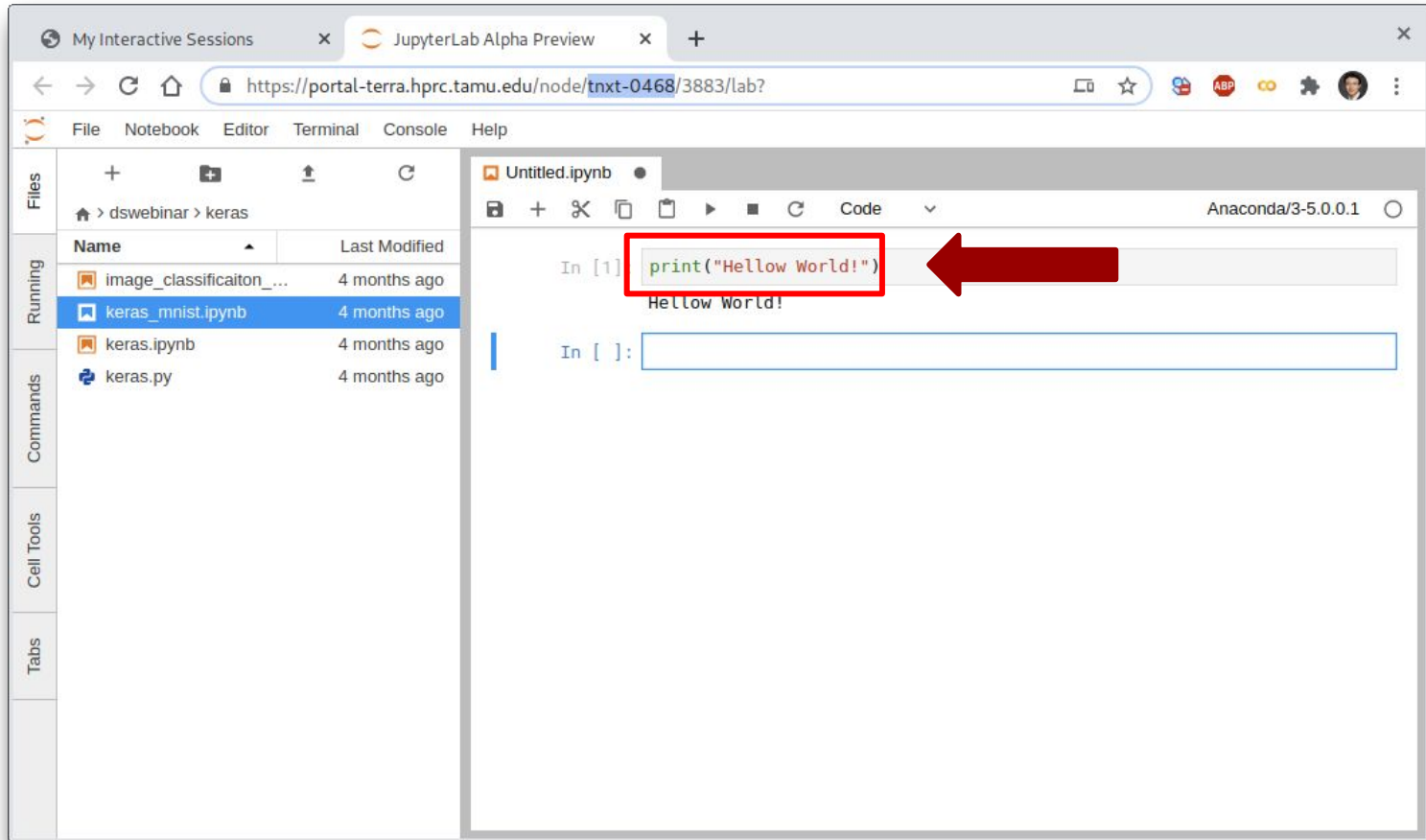


The screenshot displays the JupyterLab web interface. The browser address bar shows the URL `portal-terra.hprc.tamu.edu/node/txn-0326/46994/lab?`. The interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Tabs, Settings, and Help. On the left, a file browser shows the current directory `/ shortcourses /` with a table of files:

Name	Last Modified
Intro_to_Julia	17 hours ago
Intro_to_PyTorch	16 hours ago
Intro_to_Tensor...	6 minutes ago
README.md	17 hours ago

The main area is the Launcher, which shows a grid of environment options. A red box highlights the **Python 3** and **Bash** options. A red arrow points to the **Bash** option. Below the Launcher, a status bar indicates "Saving completed" and "Launcher".

Test JupyterLab



The screenshot displays a web browser window with two tabs: "My Interactive Sessions" and "JupyterLab Alpha Preview". The address bar shows the URL <https://portal-terra.hprc.tamu.edu/node/tnxt-0468/3883/lab?>. The JupyterLab interface includes a top menu bar with "File", "Notebook", "Editor", "Terminal", "Console", and "Help". On the left, a sidebar contains sections for "Files", "Running", "Commands", "Cell Tools", and "Tabs". The "Files" section shows a directory structure: "dswebinar > keras", with files like "image_classificaion_...", "keras_mnist.ipynb" (highlighted), "keras.ipynb", and "keras.py". The "Running" section shows a table of active notebooks:

Name	Last Modified
image_classificaion_...	4 months ago
keras_mnist.ipynb	4 months ago
keras.ipynb	4 months ago
keras.py	4 months ago

The main workspace shows a notebook titled "Untitled.ipynb" with a toolbar and "Anaconda/3-5.0.0.1" environment. The first code cell, labeled "In [1]", contains the code `print("Hello World!")`, which is highlighted with a red box. A large red arrow points to the right side of this cell. Below the code, the output "Hello World!" is displayed. The second code cell, labeled "In []:", is currently empty.

Part II. Deep Learning

Deep Learning

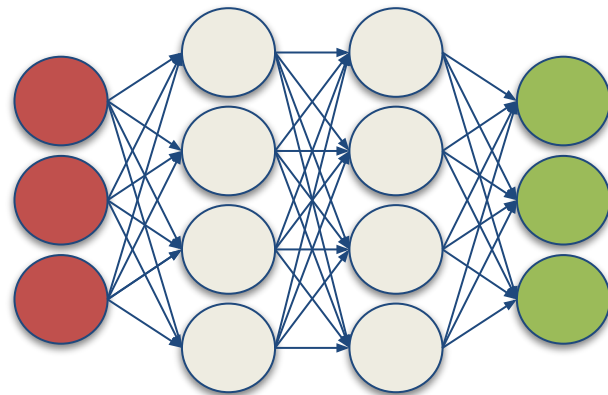
by Ian Goodfellow, Yoshua Bengio, and Aaron Courville

<http://www.deeplearningbook.org/>

Animation of Neutron Networks

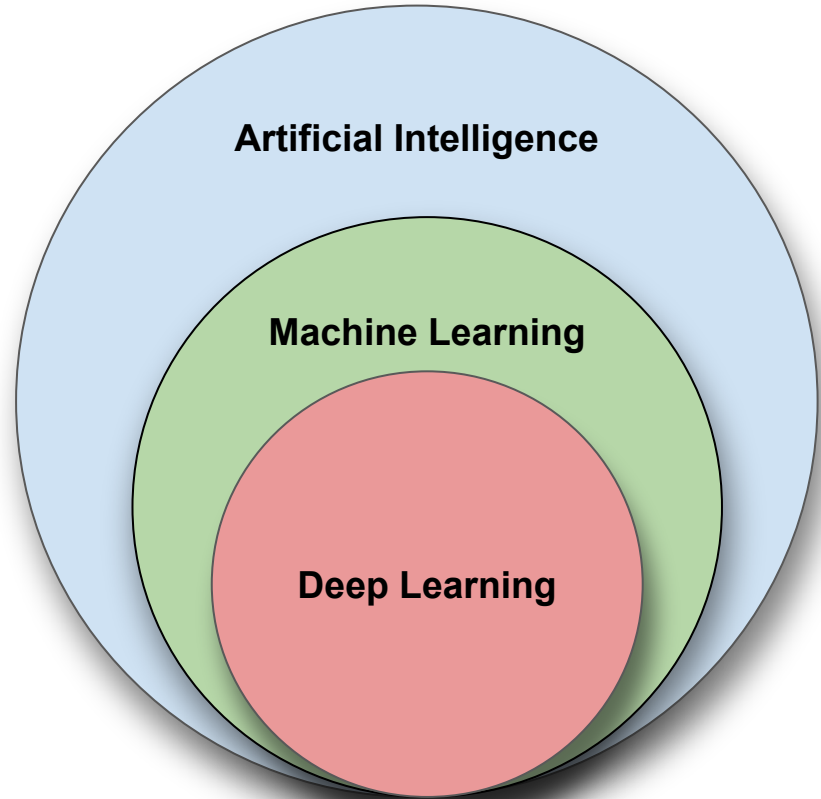
by Grant Sanderson

<https://www.3blue1brown.com/>



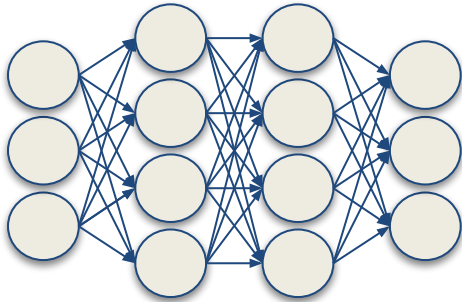
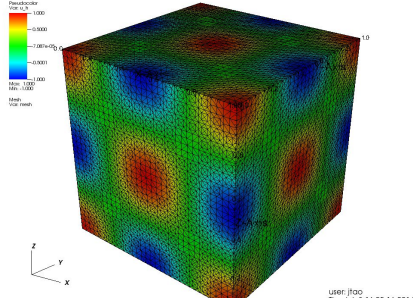
Relationship of AI, ML and DL

- **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**.

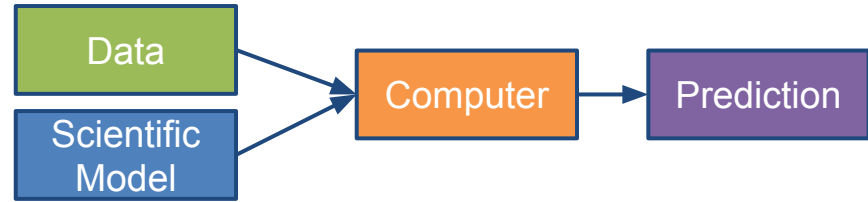


Machine Learning

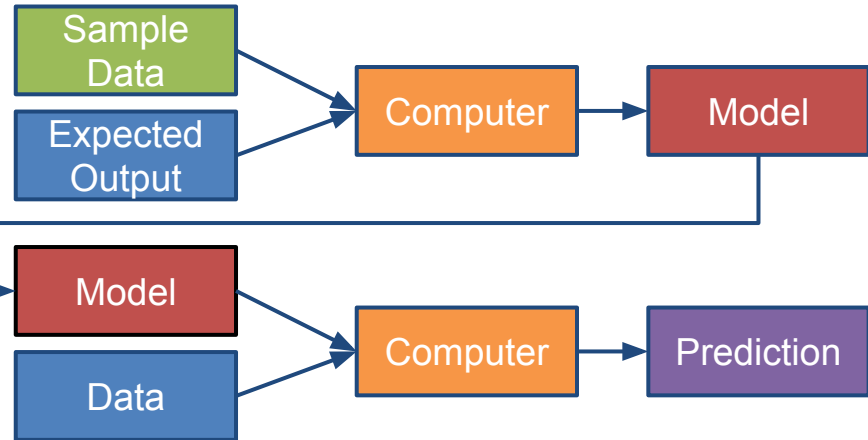
DB: simplest.vtk



Traditional Modeling

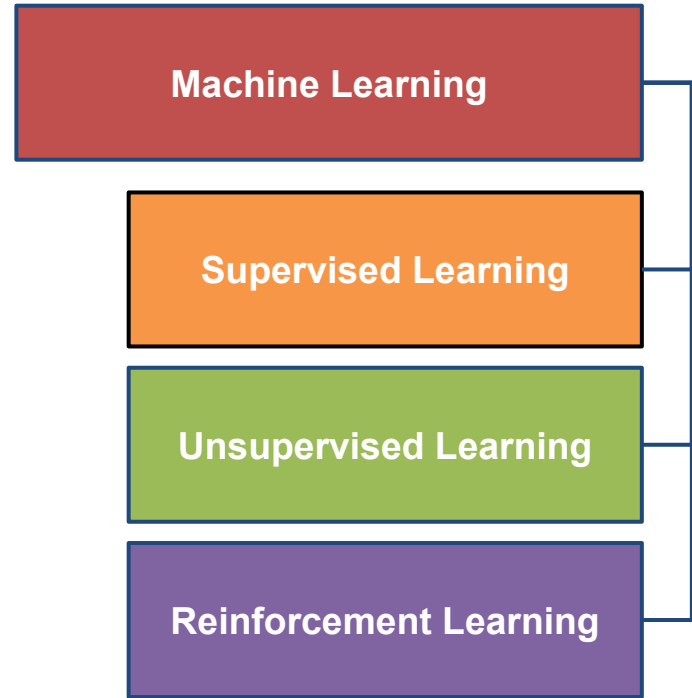


Machine Learning (Supervised Learning)



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 self-driving cars.



Why Deep Learning?

- Limitations of traditional machine learning algorithms
 - not good at handling high dimensional data.
 - difficult to do feature extraction and object recognition.
- Advantages of deep learning
 - DL is computationally expensive, but it is capable of handling high dimensional data.
 - feature extraction is done automatically.

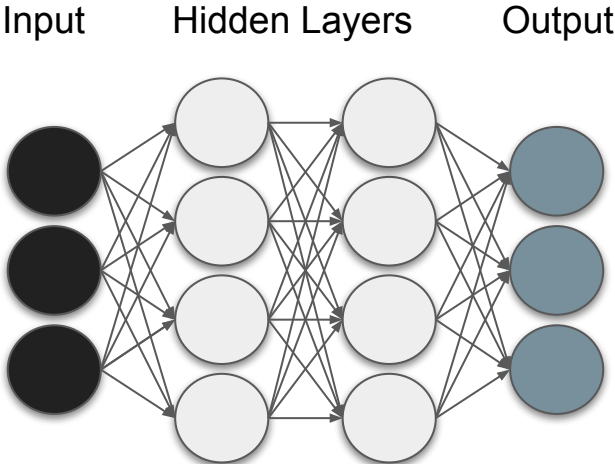
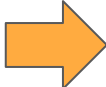
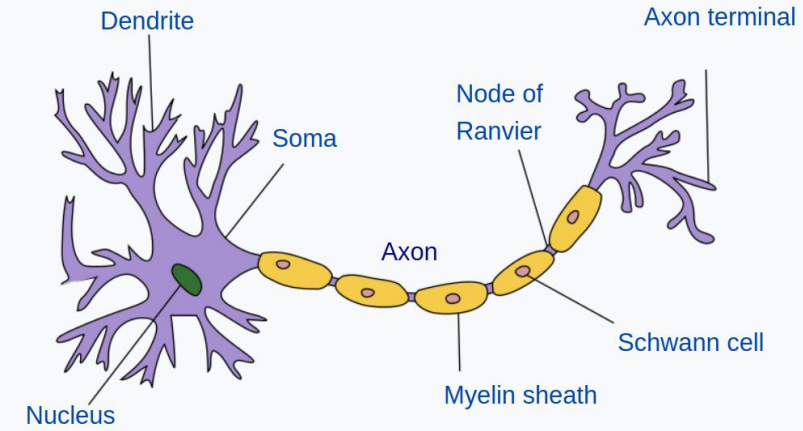
What is Deep Learning?

Deep learning is a class of machine learning algorithms that:

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

(Source: Wikipedia)

Artificial Neural Network



(Image Credit: Wikipedia)

Inputs and Outputs



```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 71 26
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 44 48 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 55 43 30 03 49 13 36 65
52 70 95 23 04 60 11 42 60 21 68 56 01 32 36 71 37 02 36 91
22 31 16 71 51 63 03 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 31 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 36 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 32 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
23 46 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 45 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 93 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 24 41 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 21 67 48
```

What the computer sees

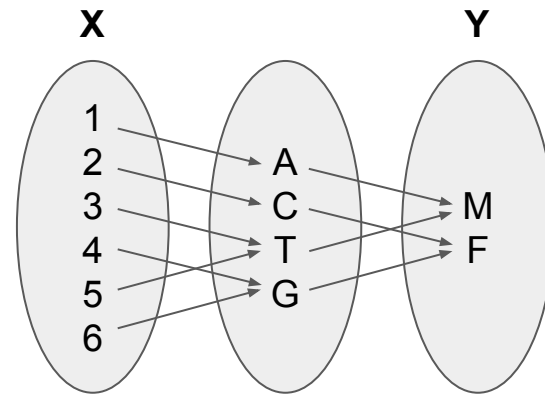
image classification → 82% cat
15% dog
2% hat
1% mug

Image from the [Stanford CS231 Course](#)

256 X 256
Matrix

DL model

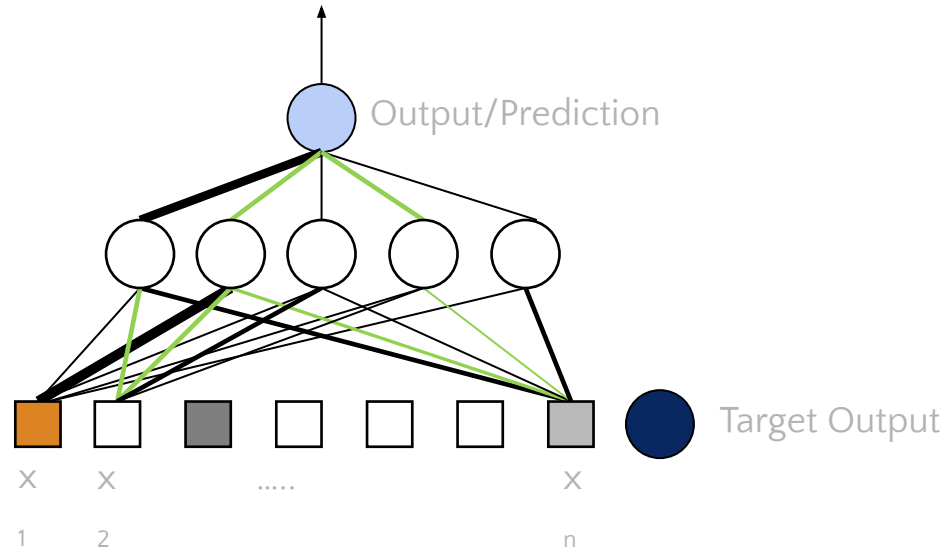
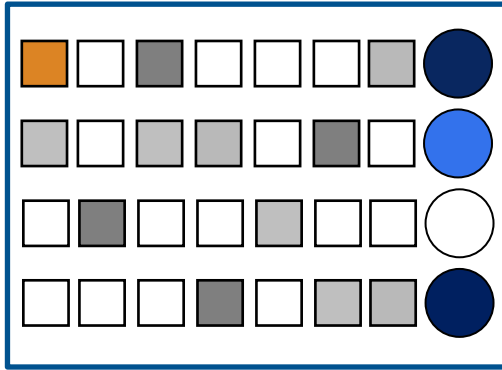
4-Element Vector



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

Learning Principle

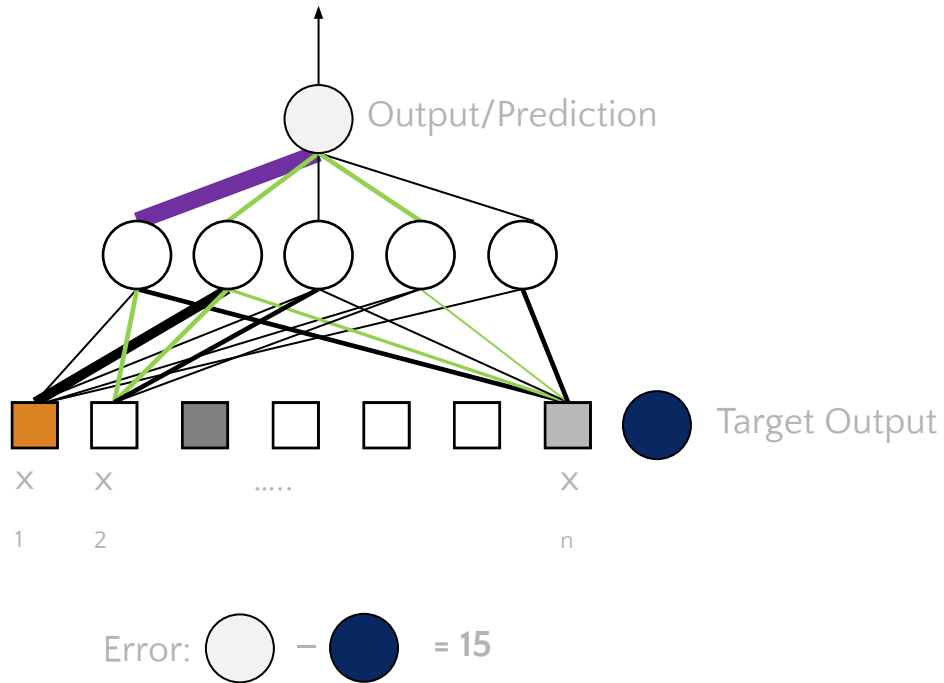
Dataset



Error: $\text{Light Blue Circle} - \text{Dark Blue Circle} = 5$

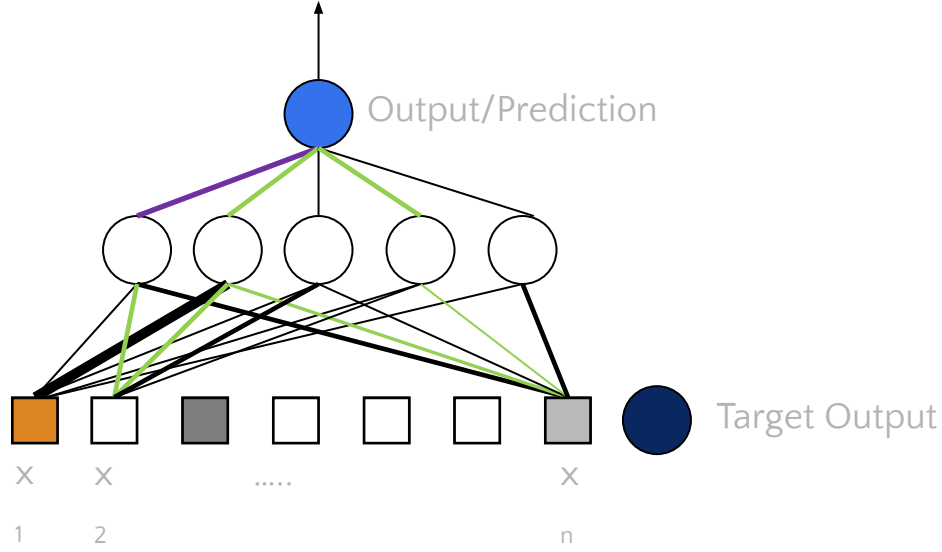
(Image Credit: NVIDIA Deep Learning Institute)



Learning Principle



(Image Credit: NVIDIA Deep Learning Institute)

Learning Principle



Error:  -  = 2.5

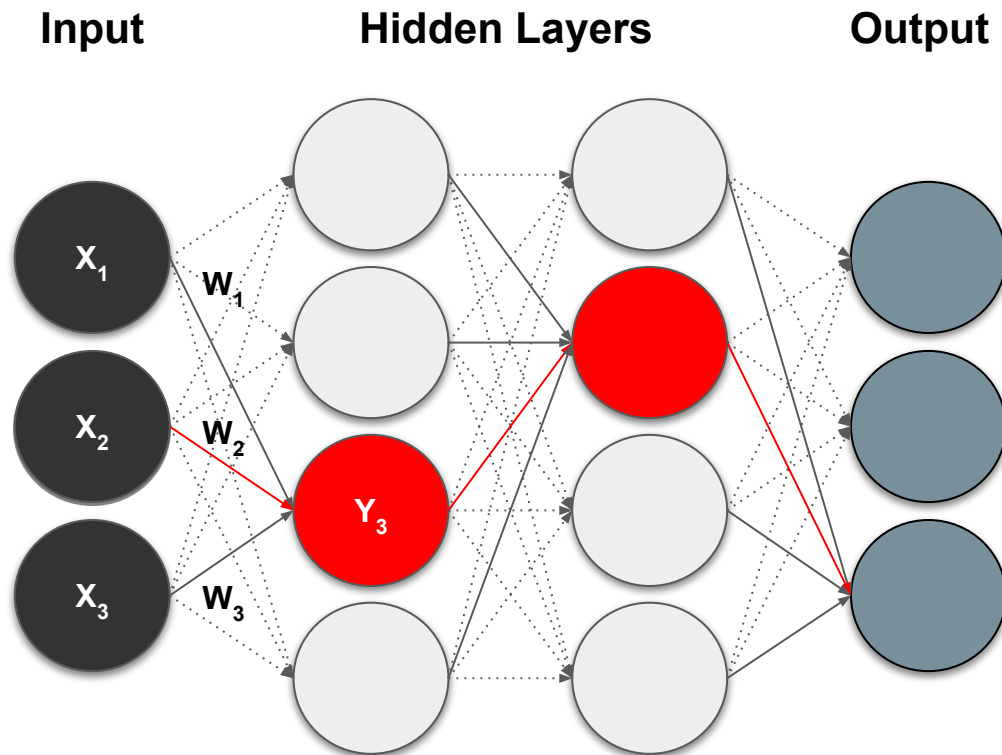
(Image Credit: NVIDIA Deep Learning Institute)

Supervised Deep Learning with Neural Networks

From one layer to the next

$$Y_j = f\left(\sum_i W_i X_i + b_i\right)$$

f is the activation function,
 W_i is the weight, and b_i is
the bias.



Training - Minimizing the Loss

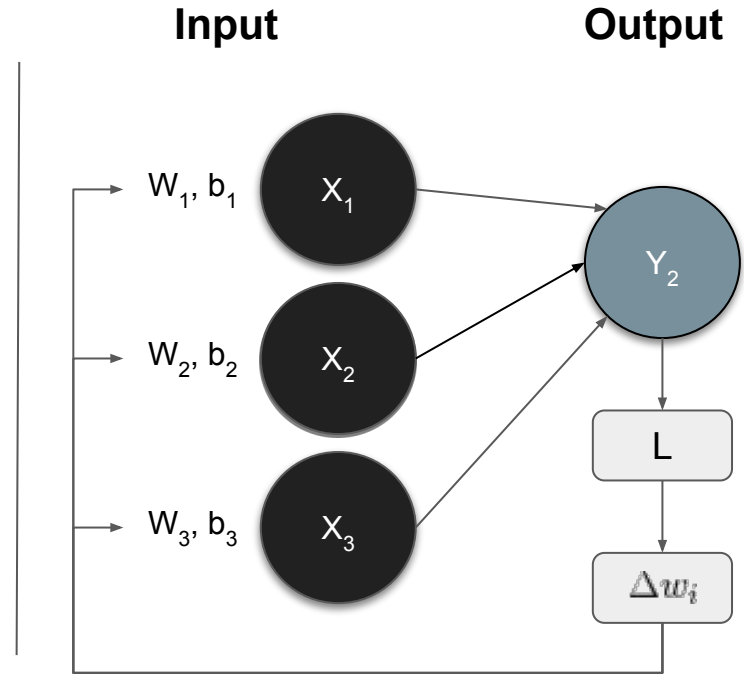
The loss function with regard to weights and biases can be defined as

$$L(\mathbf{w}, \mathbf{b}) = \frac{1}{2} \sum_i (\mathbf{Y}(\mathbf{X}, \mathbf{w}, \mathbf{b}) - \mathbf{Y}'(\mathbf{X}, \mathbf{w}, \mathbf{b}))^2$$

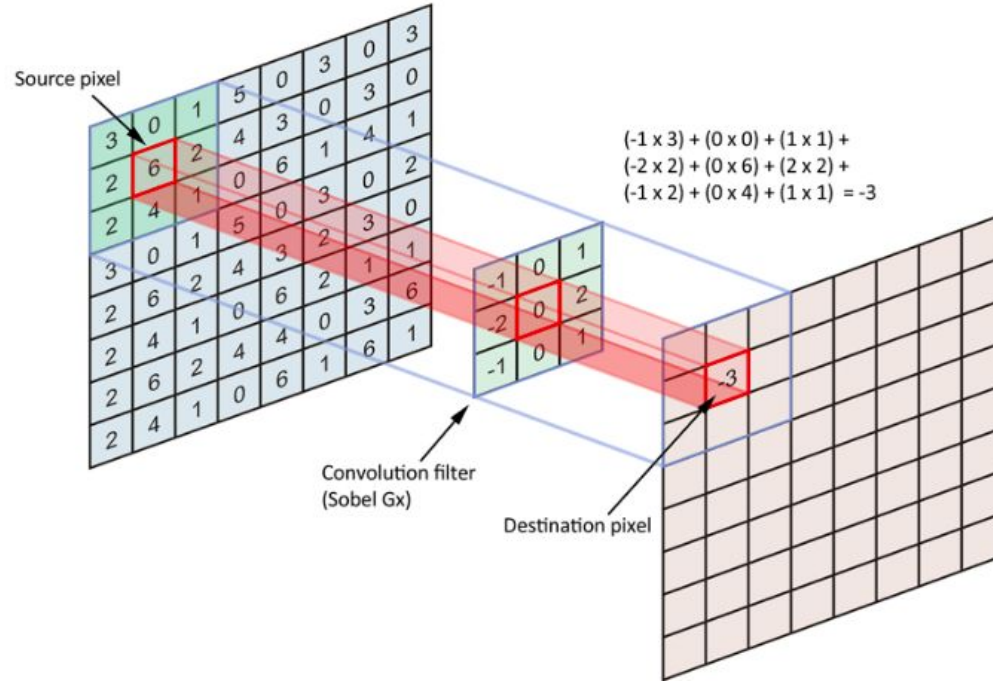
The weight update is computed by moving a step to the opposite direction of the cost gradient.

$$\Delta w_i = -\alpha \frac{\partial L}{\partial w_i}$$

Iterate until L stops decreasing.



Convolution in 2D



(Image Credit: [Applied Deep Learning | Arden Dertat](#))

Convolution Kernel

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

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

(Image Credit: [Applied Deep Learning | Arden Dertat](#))

Convolution on Image



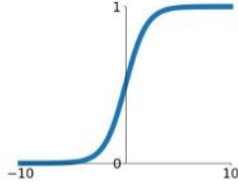
Input

Image Credit: [Deep Learning Methods for Vision | CVPR 2012 Tutorial](#)

Activation Functions

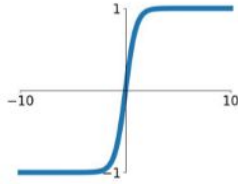
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



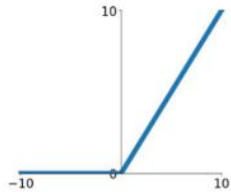
tanh

$$\tanh(x)$$



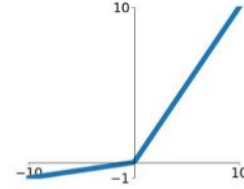
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

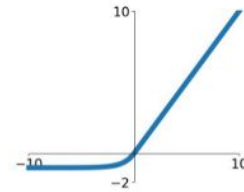
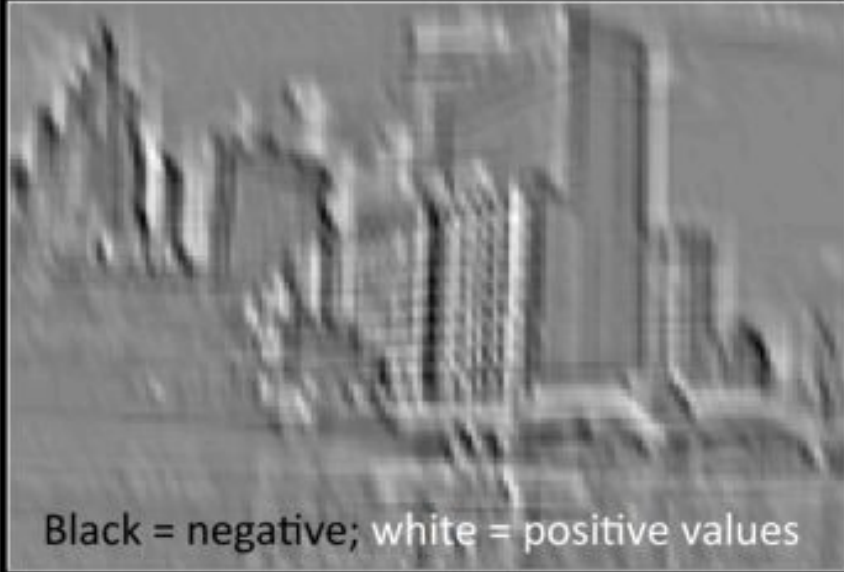


Image Credit: towardsdatascience.com

Introducing Non Linearity (ReLU)

Input Feature Map



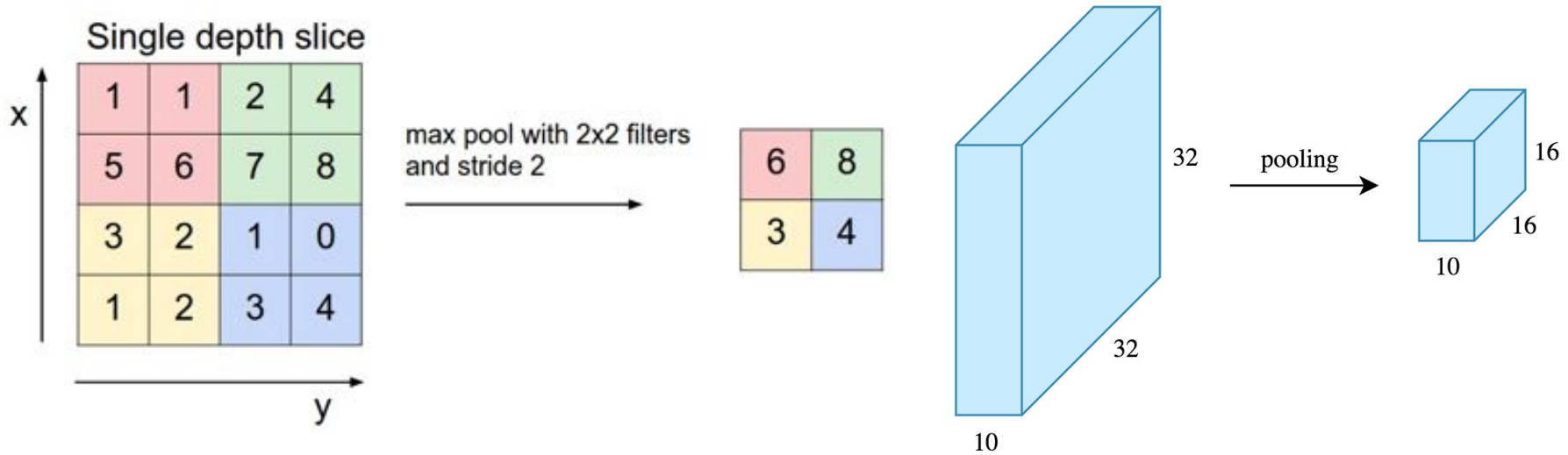
ReLU
➔

Rectified Feature Map



Image Credit: [Deep Learning Methods for Vision | CVPR 2012 Tutorial](#)

Max Pooling



(Image Credit: [Applied Deep Learning | Arden Dertat](#))

Pooling - Max-Pooling and Sum-Pooling

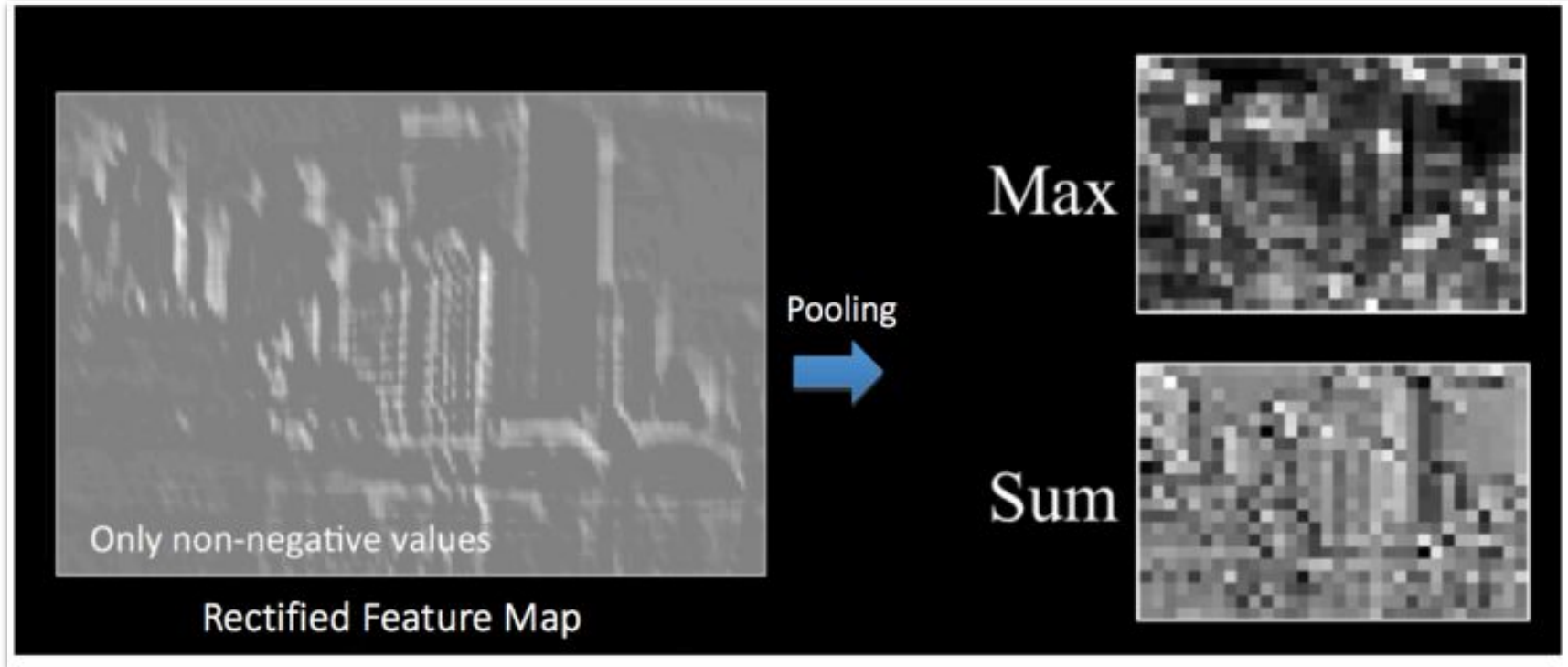
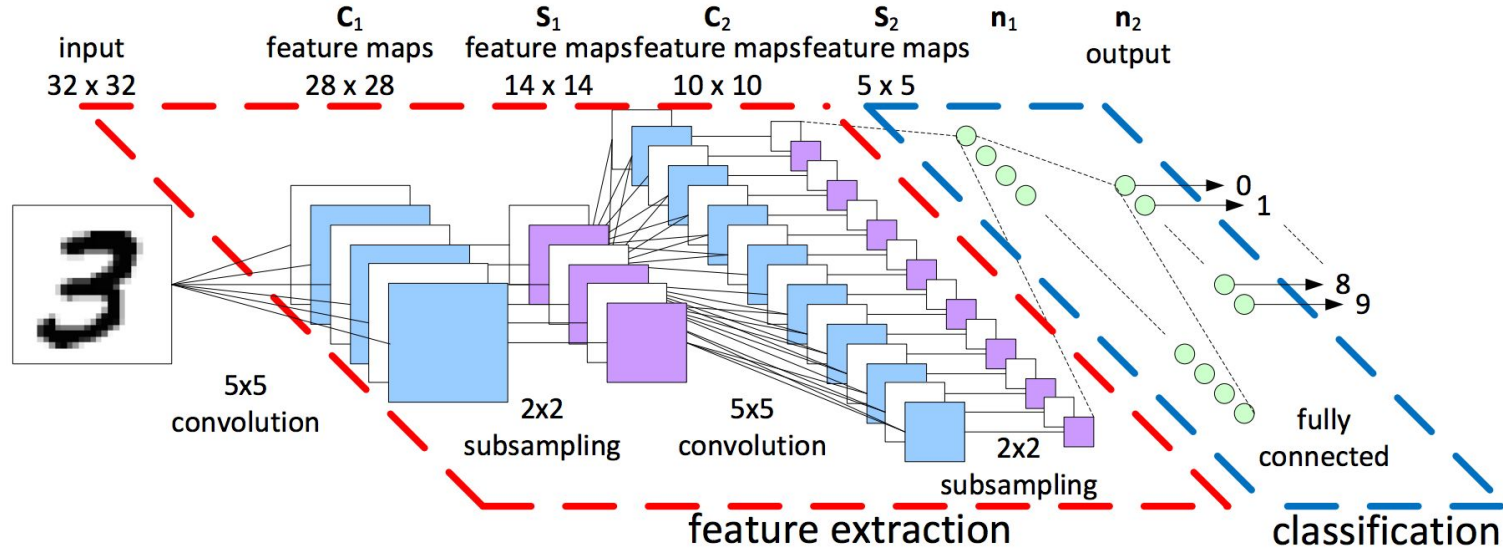


Image Credit: [Deep Learning Methods for Vision | CVPR 2012 Tutorial](#)

Convolutional Neural Networks

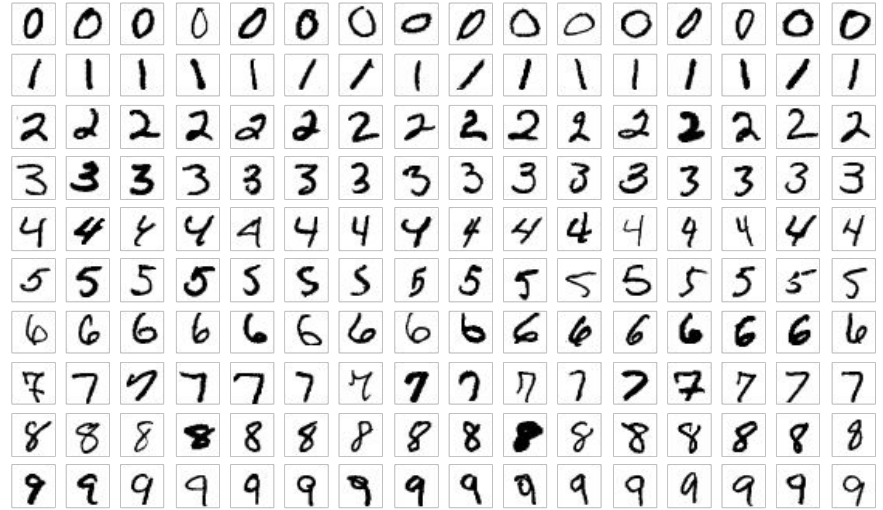
A convolutional neural network (**CNN**, or **ConvNet**) is a class of deep, feed-forward artificial neural networks that explicitly assumes that the inputs are images, which allows us to encode certain properties into the architecture.



LeNet-5 Architecture (Image Credit: <https://becominghuman.ai>)

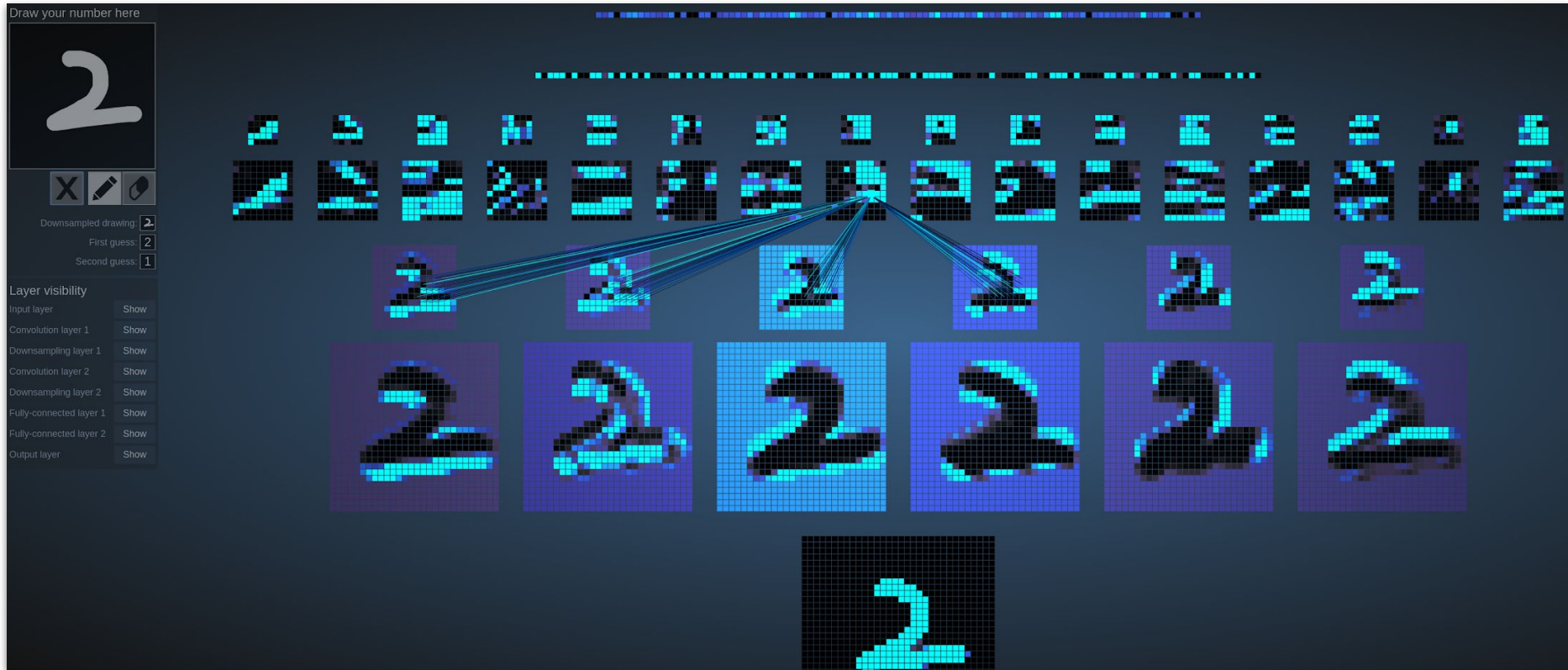
MNIST - Introduction

- **MNIST** (Mixed National Institute of Standards and Technology) is a database for handwritten digits, distributed by Yann Lecun.
- 60,000 examples, and a test set of 10,000 examples.
- 28x28 pixels each.
- Widely used for research and educational purposes.



(Image Credit: Wikipedia)

MNIST - CNN Visualization



(Image Credit: <http://scs.ryerson.ca/~aharley/vis/>)

Hands-on Session #1

A Simple Deep Learning Example with PyTorch - First Glance



Part III. Introduction to PyTorch

PyTorch website:

<https://pytorch.org/>

PyTorch Tutorials:

<https://pytorch.org/tutorials/>

Book: Deep Learning with PyTorch

[PDF](#)



A Brief History of PyTorch

PyTorch is an open source machine learning library based on the Torch library, which was first released by Ronan Collobert, Koray Kavukcuoglu, and Clement Farabet in Oct 2002.

- The first official release of PyTorch was by Facebook's AI Research lab (FAIR) in Oct 2016.
- Version 1.0 that integrated both Caffe2 and ONNX was release in May 2018.
- The latest stable release is version 1.10, as of Oct 28, 2021.

Overview of PyTorch

PyTorch is an open-source machine learning library written in Python, C++ and CUDA. PyTorch provides two high-level features:

- Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU)
- Deep neural networks built on a tape-based autodiff system

In a layman's term, PyTorch is a fancy version of NumPy that runs on GPUs and comes with a lot of machine learning functionalities.

TensorFlow, Keras, and PyTorch



TensorFlow is an end-to-end open source **platform** for machine learning. It has a comprehensive, flexible ecosystem to build and deploy ML powered applications.

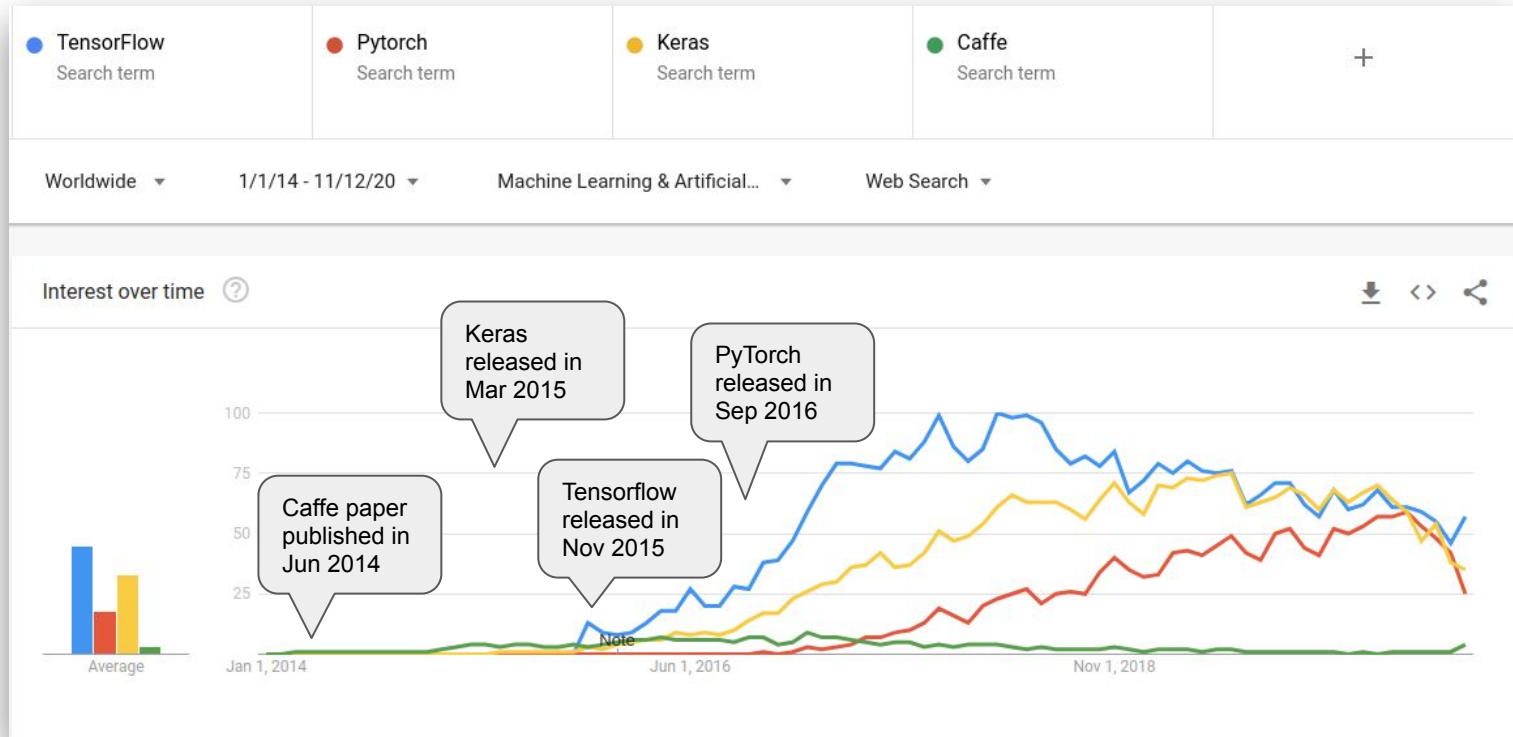


Keras is a high-level neural networks **API**, written in Python and capable of running on top of *TensorFlow*, *CNTK*, or *Theano*. It was developed with a focus on enabling fast experimentation.



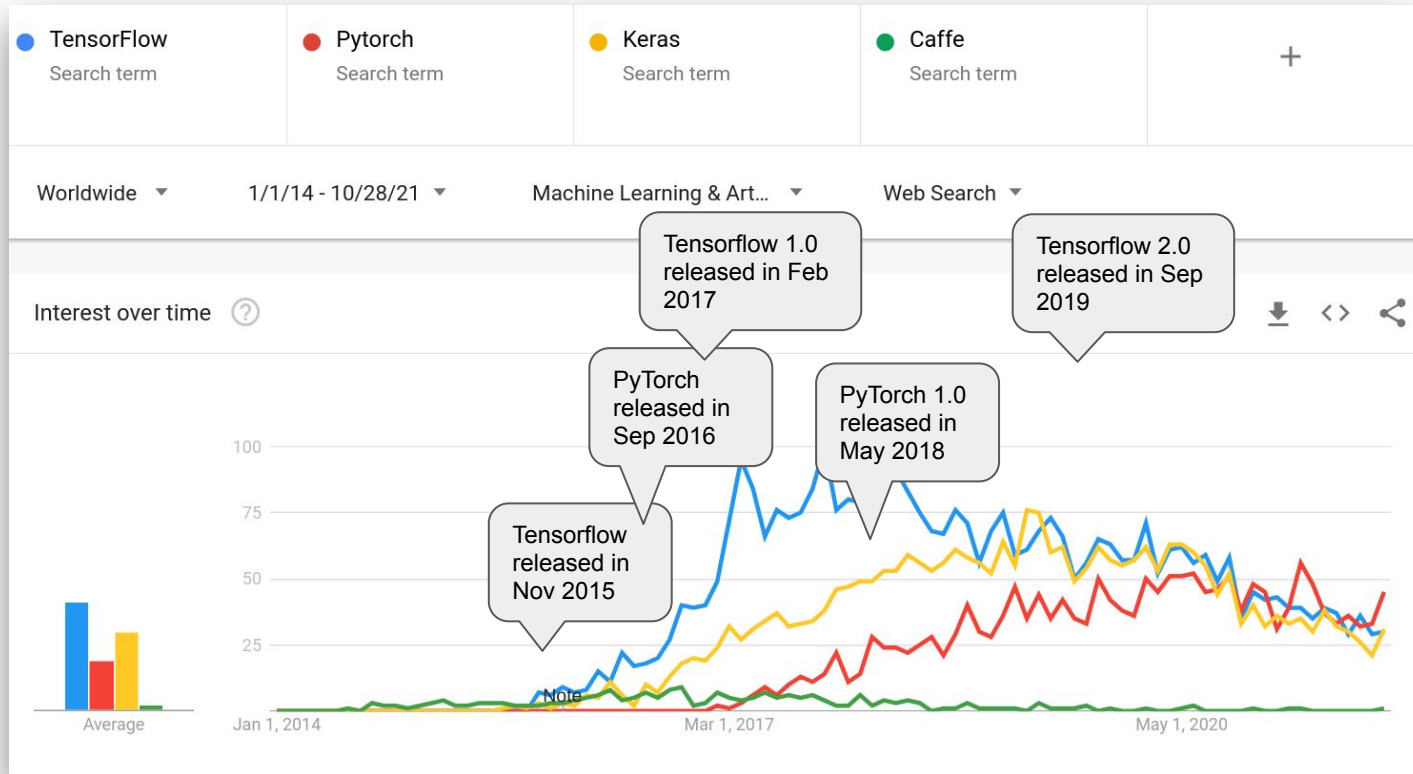
PyTorch is an open source machine learning **framework** that accelerates the path from research prototyping to production deployment.

Google Trends for Popular ML Frameworks



(Image Credit: <https://trends.google.com/>)

Google Trends for Popular ML Frameworks



(Image Credit: <https://trends.google.com/>)

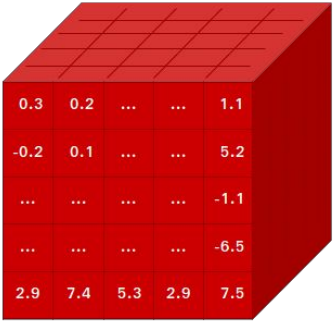
Major Components of PyTorch

Components	Description
torch	a <i>Tensor library like NumPy</i> , with strong GPU support
torch.autograd	a <i>tape-based automatic differentiation library</i> that supports all differentiable Tensor operations in torch
torch.jit	a <i>compilation stack (TorchScript)</i> to create serializable and optimizable models from PyTorch code
torch.nn	a <i>neural networks library</i> deeply integrated with autograd designed for maximum flexibility
torch.multiprocessing	<i>Python multiprocessing</i> , but with magical memory sharing of torch Tensors across processes. Useful for data loading and Hogwild training
torch.utils	<i>DataLoader and other utility functions</i> for convenience

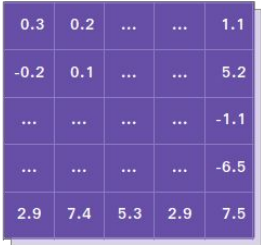
A Powerful Tensor Library - torch

- A PyTorch tensor is an n-dimensional array that can live on either the CPU or GPU. A tensor has a static type, a rank, and a shape.

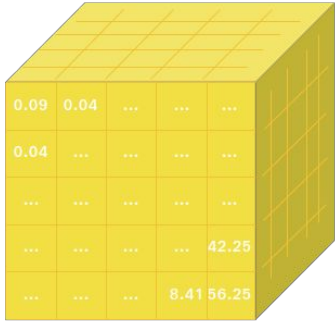
Name	Rank	Tensor
Scalar	0	[5]
Vector	1	[1 2 3]
Matrix	2	[[1 2 3 4], [5 6 7 8]]
Tensor	3	...



*



=



(Image Credit: pytorch.org)

Tensors on CPU and GPU - torch

```
x = torch.randn(1)
# check if a CUDA device is available
if torch.cuda.is_available():

    # a CUDA device object
    device = torch.device("cuda")

    # directly create y
    x = x.to(device)
    y = torch.ones_like(x, device=device)

z = x + y
print(z)
print(z.to("cpu", torch.double))
```



Tape-Based AutoGrad - torch.autograd

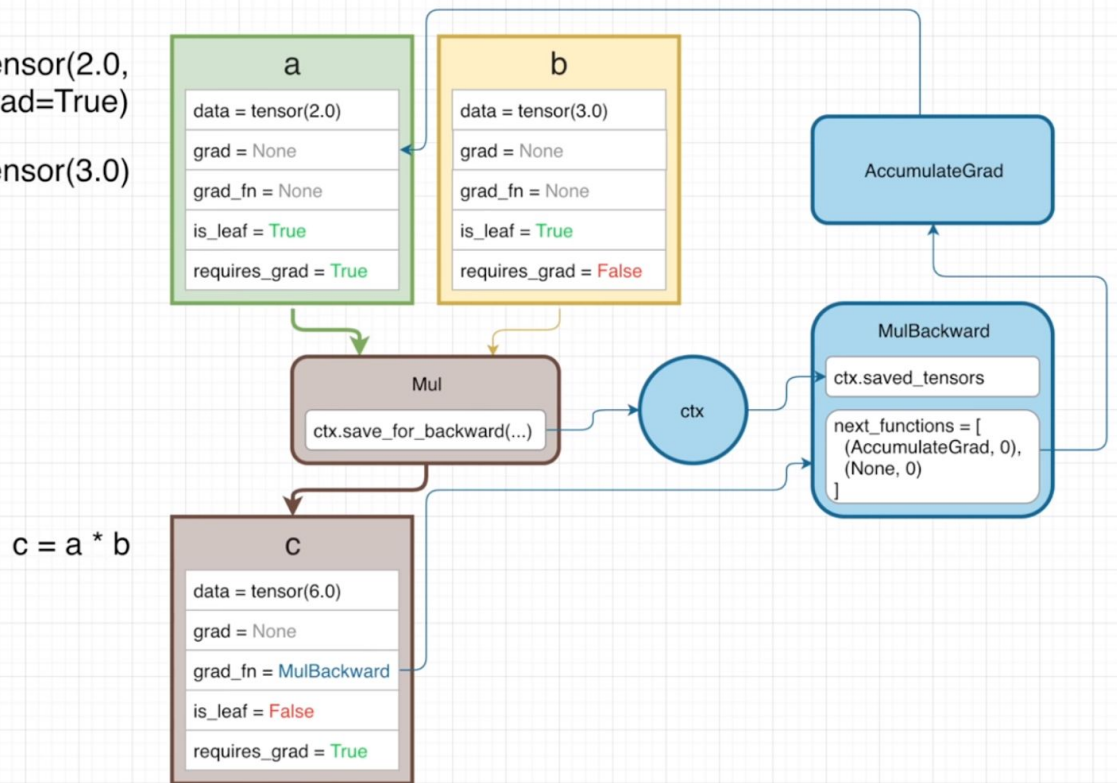
- **torch.autograd** is central to all neural networks in PyTorch.
- The **autograd** package provides automatic differentiation for all operations on Tensors.
- Use "**requires_grad=True**" to keep track of operations on a Tensor.

```
# x = tensor([[1., 1.],  
             [1., 1.]], requires_grad=True)  
x = torch.ones(2, 2, requires_grad=True)  
  
# y = tensor([[3., 3.],  
             [3., 3.]], grad_fn=<AddBackward0>)  
y = x + 2
```

Tape-Based AutoGrad - torch.autograd

```
a = torch.tensor(2.0,  
requires_grad=True)
```

```
b = torch.tensor(3.0)
```



(Image Credit: Elliot Waite: <https://youtu.be/MswxJw-8PvE>)

- PyTorch uses and replays a "**tape recorder**" to build neural networks.
- The official name of the method is called **reverse-mode auto-differentiation**.
- The dependent variable is fixed and the derivative is computed with respect to each sub-expression recursively.
- The method requires extra storage to save intermediate states.

Dynamic Graph with PyTorch

A graph is created on the fly

```
W_h = torch.randn(20, 20, requires_grad=True)
W_x = torch.randn(20, 10, requires_grad=True)
x = torch.randn(1, 10)
prev_h = torch.randn(1, 20)
```



Neural Network - torch.nn

- **torch.nn** depends on **autograd** to define models and differentiate them.
- An **nn.Module** contains layers, and a method **forward(input)** that returns the output.

```
import torch
import torch.nn as nn

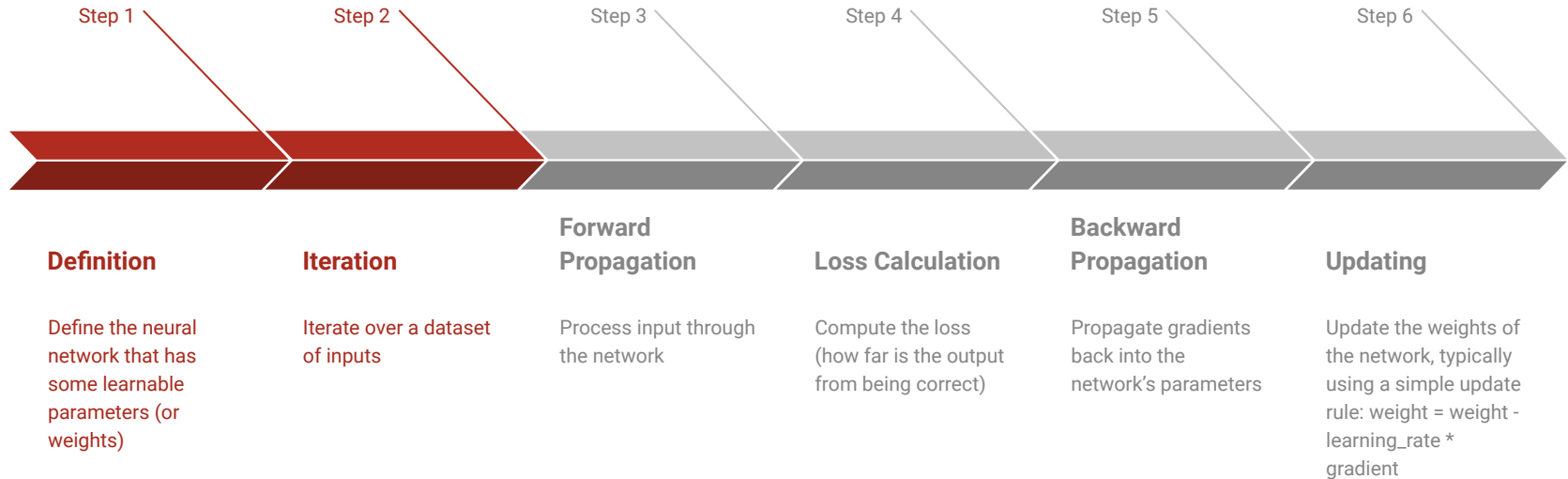
# define a neural network model
class Net(nn.Module):

    def __init__(self, param):
        super(Net, self).__init__()
        self.param = param

    def forward(self, x):
        return x * self.param

net = Net(torch.Tensor([3, 4, 5]))
print(net)
```

Procedure to Train a Neural Network - Given a Data Set



Train a Neural Network - torch.nn

- **Define** the neural network that has some learnable parameters.
- **Iterate** over a dataset of inputs
- **Process** input through the network
- **Compute the loss** (how far is the output from being correct)
- **Propagate gradients back** into the network's parameters
- **Update the weights** of the network.

```
import torch.optim as optim

# Net is a predefined nn model
net = Net(torch.Tensor([3, 4, 5]))
output = net(input)

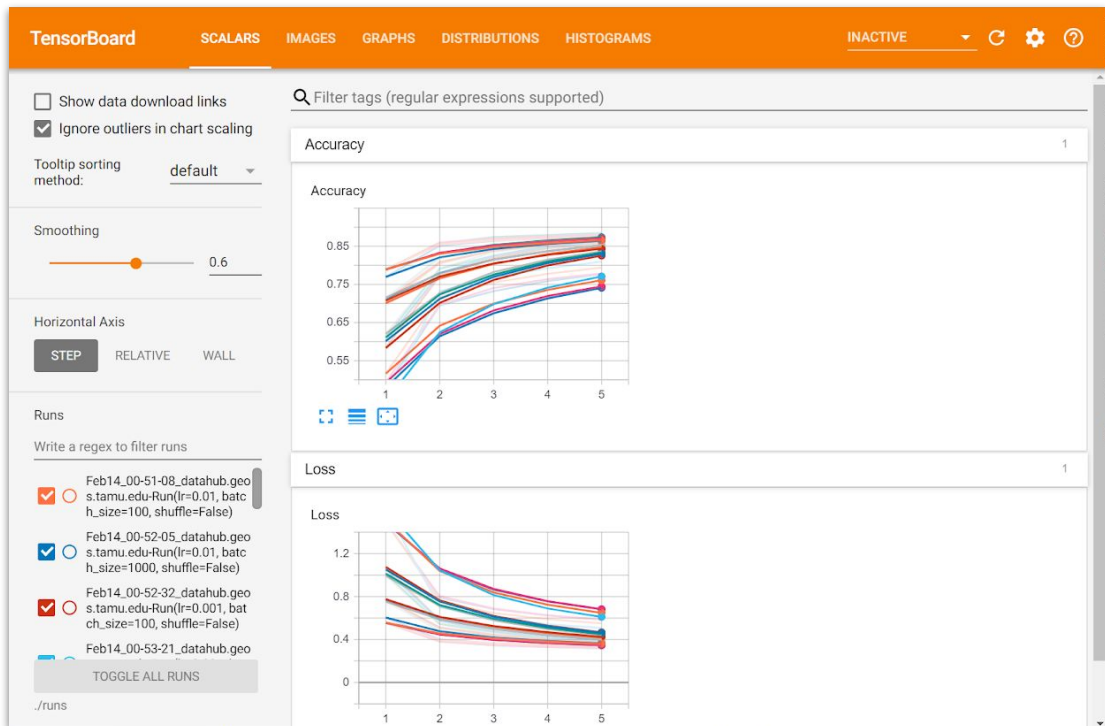
# define a dummy target
target = torch.randn(10)
target = target.view(1, -1)
criterion = nn.MSELoss()
loss = criterion(output, target)

# use one of the update rules such as SGD,
Nesterov-SGD, Adam, RMSProp, etc
optimizer = optim.SGD(net.parameters(),
lr=0.01)

# zero the gradient buffers
optimizer.zero_grad()
loss.backward()
optimizer.step()
```

Monitoring Training with Tensorboard

- TensorBoard is a User Interface (UI) tools designed for TensorFlow.
- More details on TensorBoard can be found at [TensorBoard](https://www.tensorflow.org/tensorboard).
- Once you've installed TensorBoard, these utilities let you log PyTorch models and metrics into a directory for visualization within the TensorBoard UI.



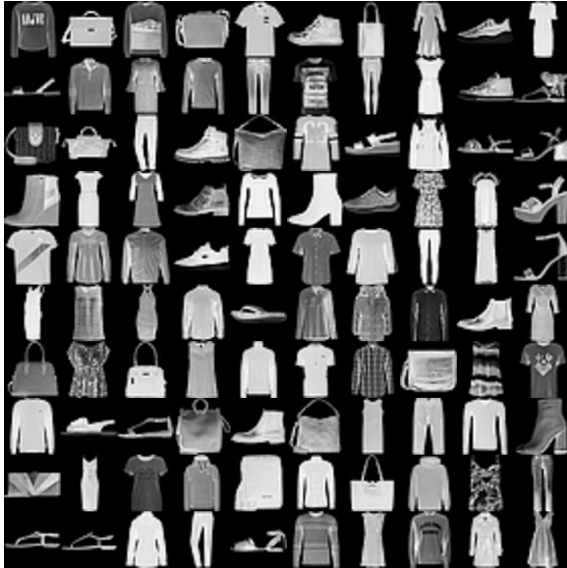
Hands-on Session #2

Getting Started with PyTorch



Hands-on Session #3

Classify Fashion-MNIST with PyTorch



- Fashion-MNIST is a dataset of Zalando's article images
- consisting of a training set of 60,000 examples and a test set of 10,000 examples.
- Each example is a 28x28 grayscale image, associated with a label from 10 classes.