

Introduction to Deep Learning with TensorFlow

Jian Tao

jtao@tamu.edu

HPRC Short Course

10/1/2021



TEXAS A&M UNIVERSITY

Visualization



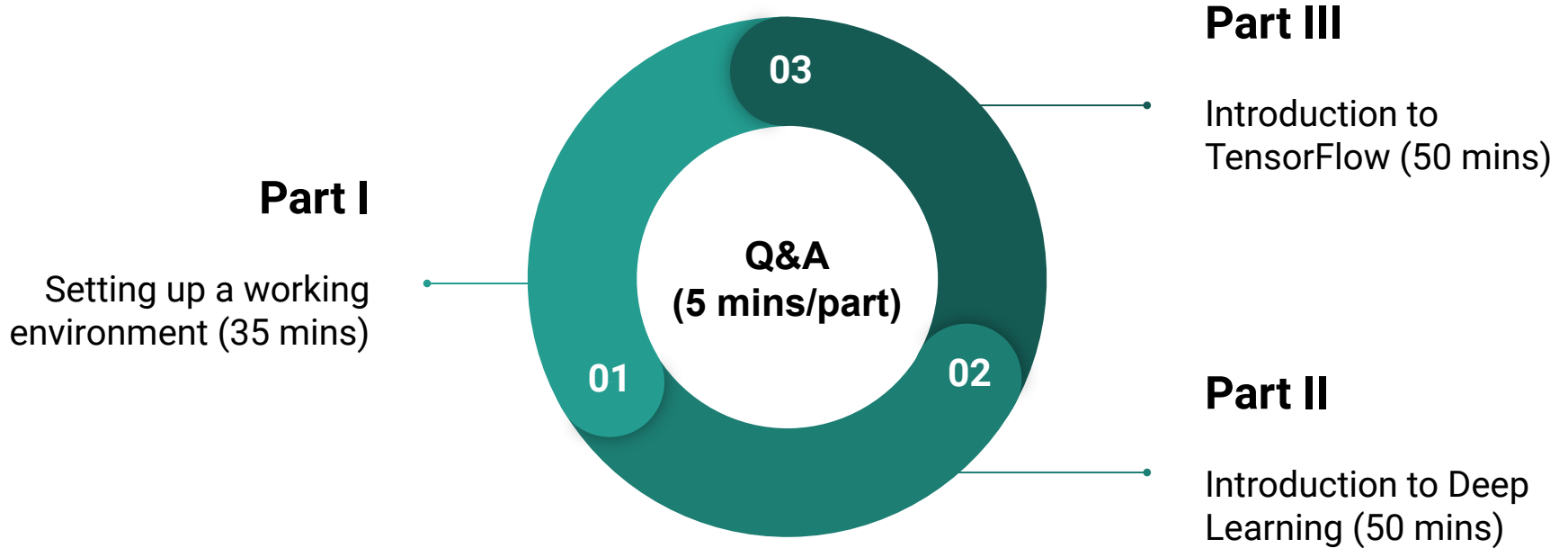
High Performance
Research Computing
DIVISION OF RESEARCH



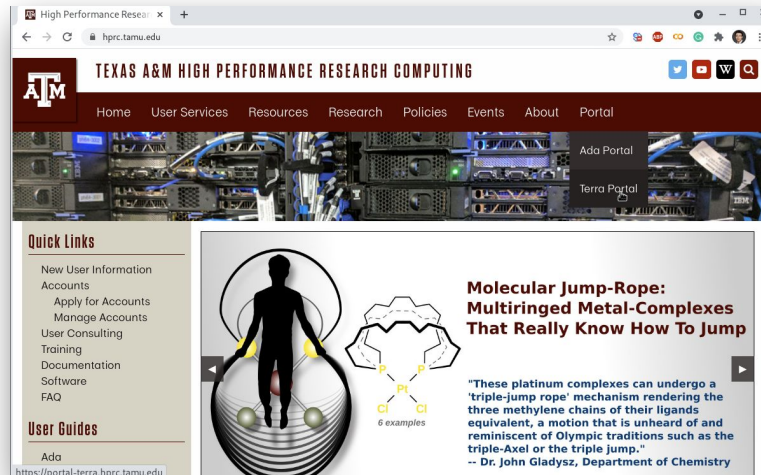
TEXAS A&M

Institute of
Data Science

Introduction to Deep Learning with TensorFlow



Part I. Working Environment

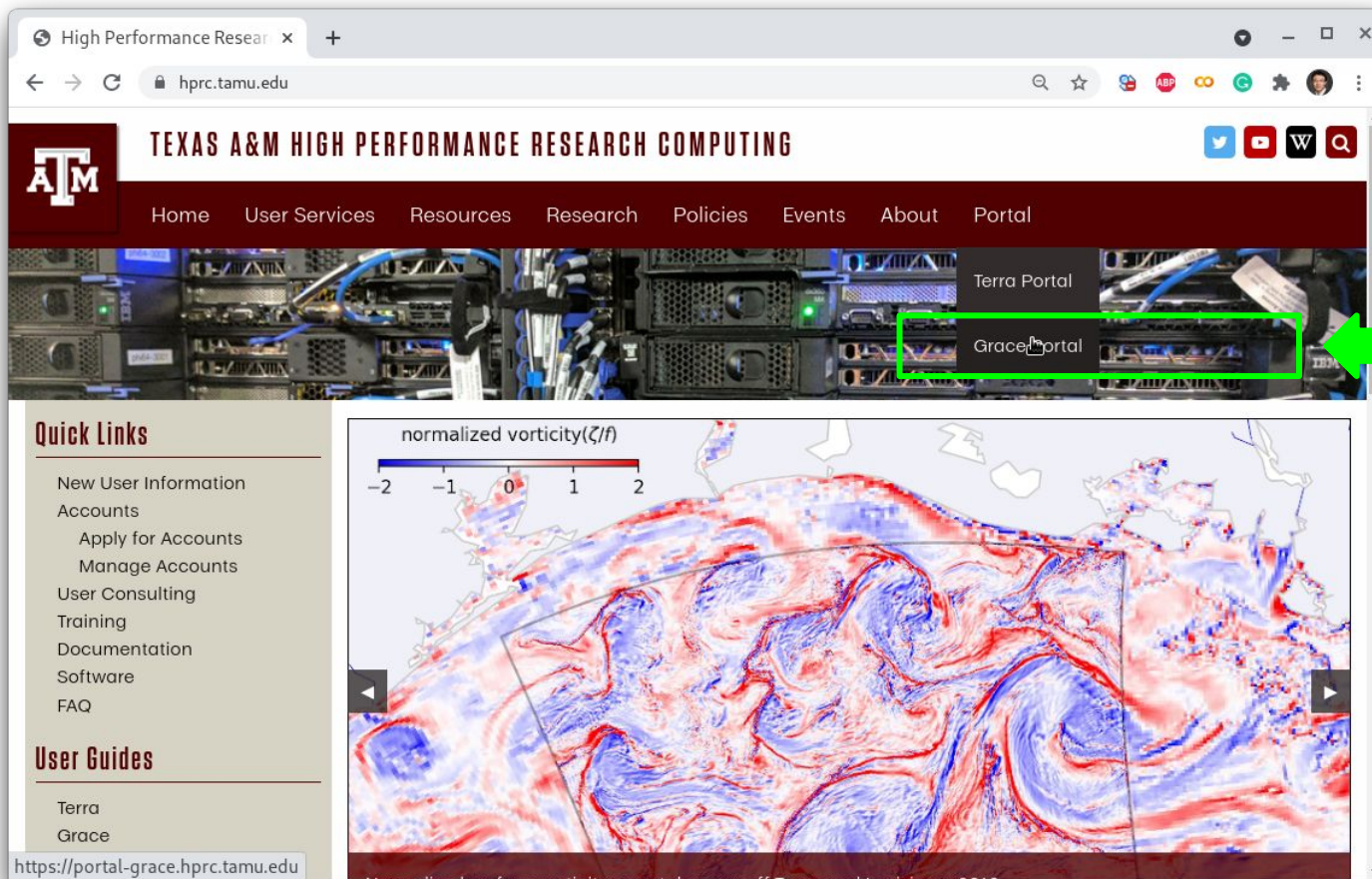


The screenshot shows the HPRC Portal website. The header includes the Texas A&M logo and the text "TEXAS A&M HIGH PERFORMANCE RESEARCH COMPUTING". Below the header is a navigation menu with links for Home, User Services, Resources, Research, Policies, Events, About, and Portal. A banner image shows server racks with "Ada Portal" and "Terra Portal" labels. The main content area features a "Quick Links" sidebar with items like "New User Information", "Accounts", "Apply for Accounts", "Manage Accounts", "User Consulting", "Training", "Documentation", "Software", and "FAQ". The "User Guides" section lists "Ada". The main content area displays a featured article titled "Molecular Jump-Rope: Multiringed Metal-Complexes That Really Know How To Jump" with a chemical structure diagram and a quote from Dr. John Gladysz.

HPRC Portal

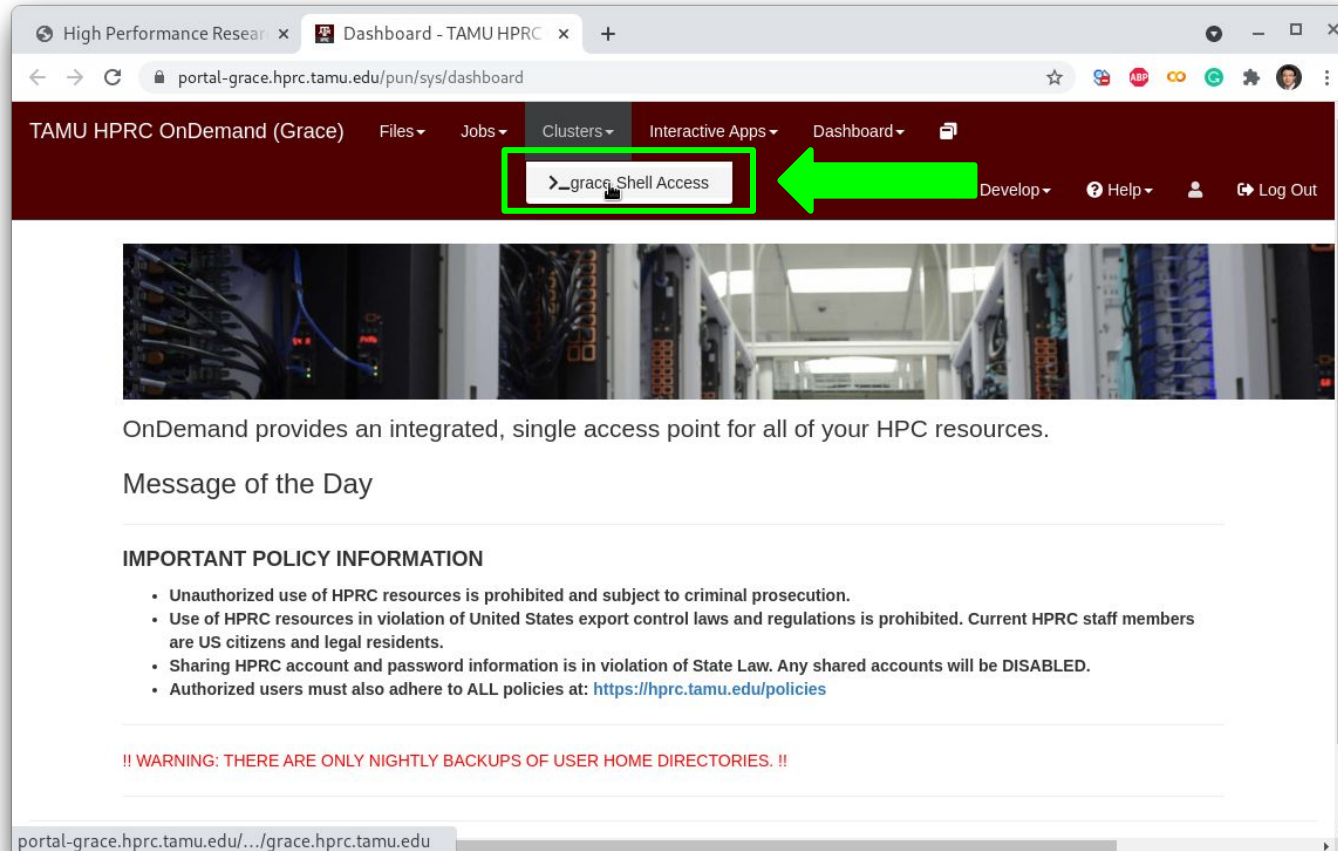
* VPN is required for off-campus users.

Login HPRC Portal (Grace)



The image shows a web browser window displaying the Texas A&M High Performance Research Computing (HPRC) website. The browser's address bar shows the URL `hprc.tamu.edu`. The website header features the Texas A&M logo and the text "TEXAS A&M HIGH PERFORMANCE RESEARCH COMPUTING". A navigation menu includes links for Home, User Services, Resources, Research, Policies, Events, About, and Portal. Below the navigation menu is a banner image of server racks. Overlaid on the right side of the banner is a dark menu with two options: "Terra Portal" and "Grace Portal". The "Grace Portal" option is highlighted with a green rectangular box, and a green arrow points to it from the right. On the left side of the page, there is a "Quick Links" section with a list of links: New User Information, Accounts, Apply for Accounts, Manage Accounts, User Consulting, Training, Documentation, Software, and FAQ. Below this is a "User Guides" section with links for Terra and Grace. At the bottom left, the URL `https://portal-grace.hprc.tamu.edu` is visible. The main content area features a map titled "normalized vorticity(ζ/f)" with a color scale from -2 to 2, showing complex atmospheric patterns over a geographical region.

Grace Shell Access - I



The screenshot shows a web browser window with the URL `portal-grace.hprc.tamu.edu/pun/sys/dashboard`. The page title is "TAMU HPRC OnDemand (Grace)". The navigation menu includes "Files", "Jobs", "Clusters", "Interactive Apps", and "Dashboard". A dropdown menu under "Clusters" is open, showing a button labeled ">_gracc Shell Access" which is highlighted with a green box and a green arrow pointing to it. Other items in the dropdown include "Develop", "Help", and "Log Out".

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

Message of the Day

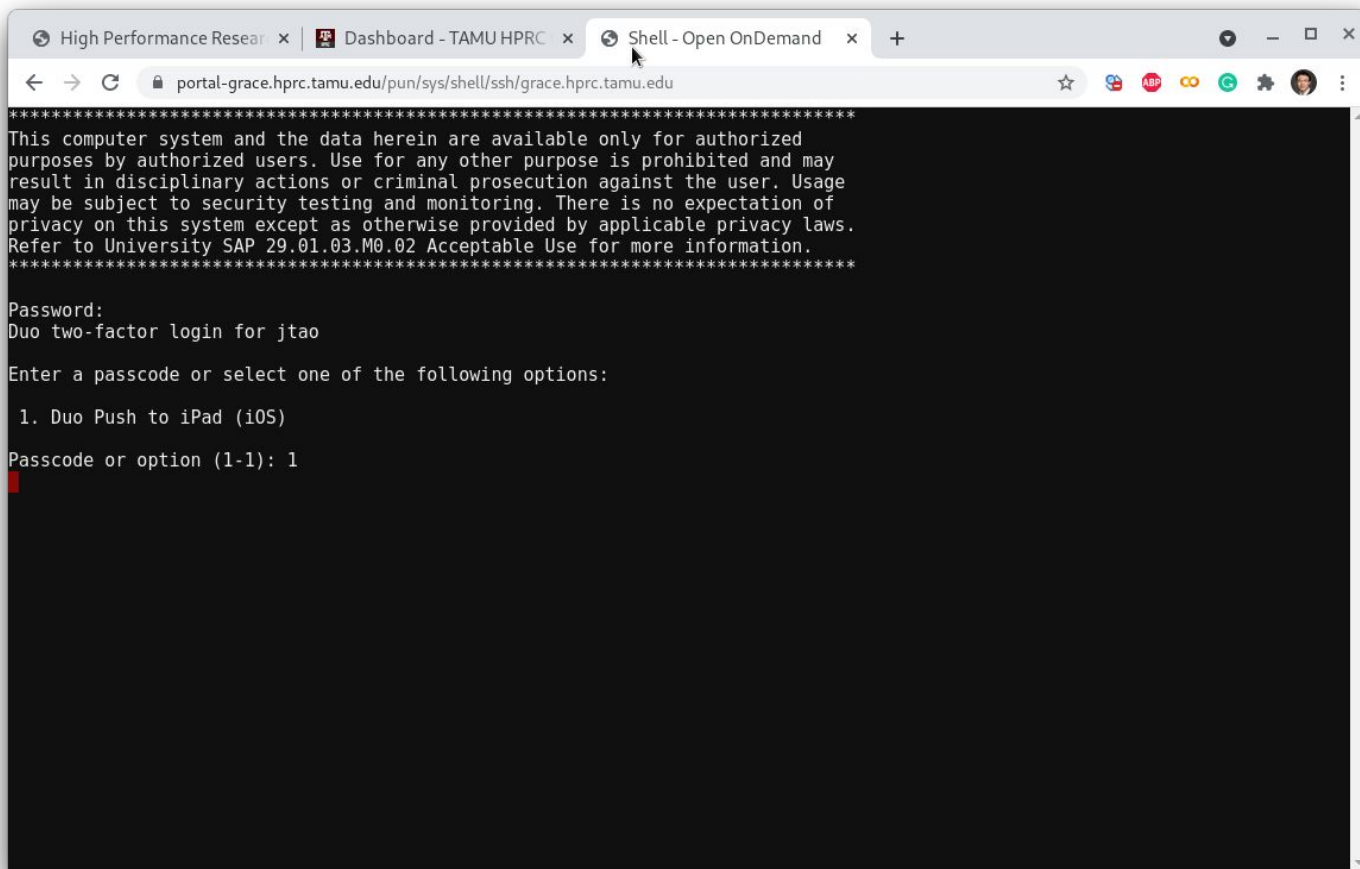
IMPORTANT POLICY INFORMATION

- 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.
- Authorized users must also adhere to ALL policies at: <https://hprc.tamu.edu/policies>

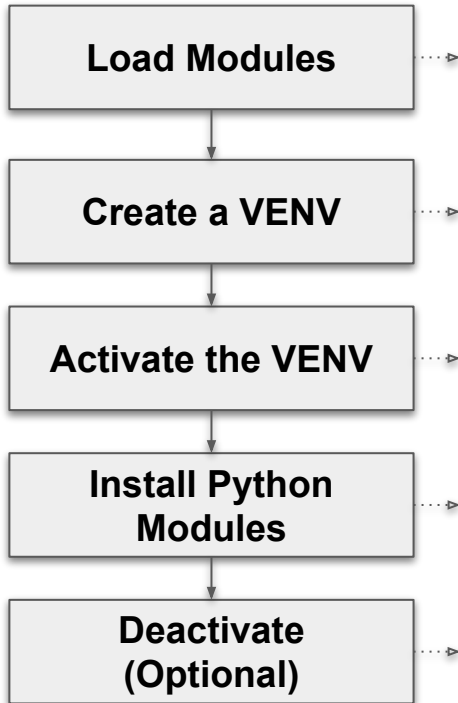
!! WARNING: THERE ARE ONLY NIGHTLY BACKUPS OF USER HOME DIRECTORIES. !!

portal-grace.hprc.tamu.edu/.../grace.hprc.tamu.edu

Grace Shell Access - II



Python Virtual Environment - CPU



```
# clean up and load Anaconda
cd $SCRATCH
module purge
module load GCCcore/9.3.0 GCC/9.3.0 Python/3.8.2

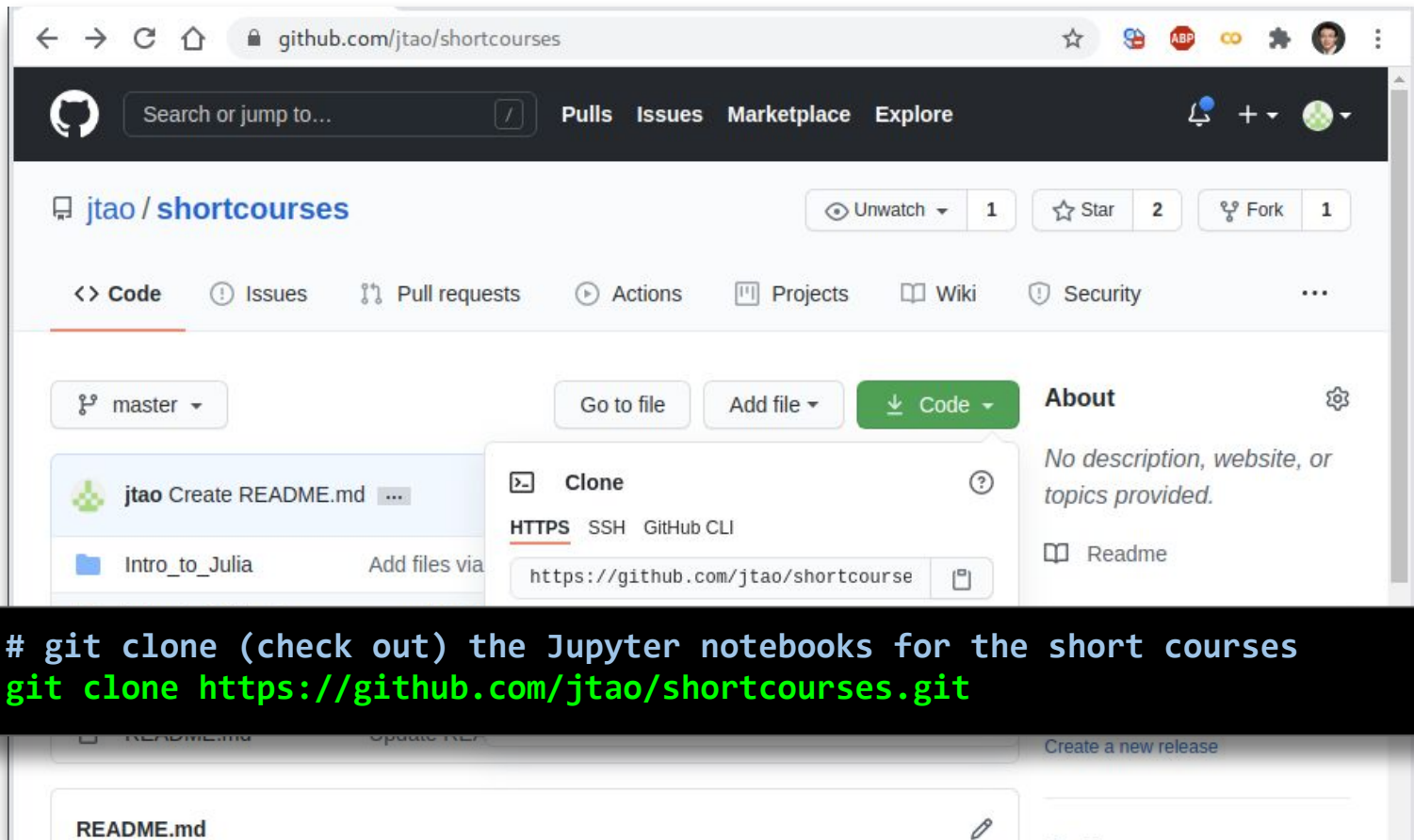
# create a Python virtual environment
python -m venv mylab

# activate the virtual environment
source mylab/bin/activate

# install required package to be used in the portal
pip install -U pip setuptools
pip install jupyterlab tensorflow sklearn matplotlib

# deactivate the virtual environment
# source 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-grace.hprc.tamu.edu/pun/sys/dashboard/`. The page title is "Dashboard - TAMU HPRC". The navigation bar includes "TAMU HPRC OnDemand (Grace)", "Files", "Jobs", "Clusters", "Interactive Apps", "Dashboard", "Develop", "Help", and "Logged in as jtiao". The "Interactive Apps" menu is open, showing categories: BIO (Beauti, CRISPR-Local, Gap5, IGV, Mauve, Structure), GUI (ANSYS Workbench, Abaqus/CAE (testing), MATLAB, ParaView, VNC), Servers (Jupyter Notebook, JupyterLab, RStudio, Spark-Jupyter Notebook), and TESTING (Jupyter Notebook (TESTING), JupyterLab (TESTING)). A red box highlights "JupyterLab (TESTING)", with a red arrow pointing to it from the right. The main content area includes a "Message of the Day" section, "IMPORTANT POLICY INFORMATION" with a list of rules, and a warning: "!! WARNING: THERE ARE ONLY NIGHTLY BACKUPS OF USER HOME". The footer shows "powered by OPEN OnDemand" and "OnDemand version: v1.8.20".

Set Virtual Environment

JupyterLab (TESTING) - TA x +

portal-grace.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sys/jupyterlab_update/session_contexts/new

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

Home / My Interactive Sessions / JupyterLab (TESTING)

Interactive Apps

- BIO
- Beauti
- CRISPR-Local
- Gap5
- IGV
- Mauve
- Structure
- GUI
- ANSYS Workbench

JupyterLab (TESTING)

This app will launch a [JupyterLab](#) server on the [Grace](#) cluster.

Module

Python/3.8.2

Anaconda/3-x.x.x.x and Anaconda3 use Python3

Optional Environment to be activated

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

Enter the name of the environment to be activated. (Optional)

Enter `/sw/hprc/sw/Anaconda3/5.3.0/envs/jupyterlmod` to enable loading lmod modules in a Conda environment.

```
# enter your virtualenv
```

```
/scratch/user/YOUR_NETID/mylab/bin/activate
```

VNC

Your optional Conda environment must have been previously built with one of the Anaconda or Python modules listed in the Module option above. See

Connect to JupyterLab

The screenshot shows a web browser window with the URL `portal-grace.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sessions`. The page title is "TAMU HPRC OnDemand (Grace)". The navigation bar includes "Files", "Jobs", "Clusters", "Interactive Apps", "Dashboard", "Develop", "Help", and "Log Out".

A green notification bar at the top states: "Session was successfully created." Below this, the breadcrumb "Home / My Interactive Sessions" is visible.

On the left, a sidebar titled "Interactive Apps" lists various applications: BIO, Beauti, CRISPR-Local, Gap5, IGV, Mauve, Structure, GUI, and ANSYS Workbench.

The main content area displays a JupyterLab session (ID: 1156840) with the following details:

- Status: 1 node | 1 core | Running
- Host: >_g015
- Created at: 2021-09-29 02:57:52 CDT
- Time Remaining: 38 minutes
- Session ID: 085b0465-a203-46d4-99ae-a11257dab270

A red box highlights the "Connect to JupyterLab" button, with a red arrow pointing to it from the right.

Create a Jupyter Notebook

The screenshot displays the JupyterLab web interface in a browser. The browser's address bar shows the URL `portal-grace.hprc.tamu.edu/node/g015/26259/lab`. The JupyterLab interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Tabs, Settings, and Help. On the left, there is a file browser sidebar with a search bar and a table of files. The main area is titled "Launcher" and shows the current directory `shortcourses/Intro_to_TensorFlow`. It lists three options: "Notebook", "Console", and "Other". The "Notebook" option, which features a Python logo and the text "Python 3 (ipykernel)", is highlighted with a red rectangular box. A large red arrow points from the right towards this box. The status bar at the bottom indicates "Simple" mode, "Saving completed", and the "Launcher" view.

Name	Last Modified
images	2 hours ago
Introductio...	2 hours ago
keras_mni...	3 minutes ago
keras.ipynb	2 hours ago
README....	2 hours ago

Test JupyterLab

The screenshot displays a JupyterLab environment in a web browser. The browser's address bar shows the URL: `portal-grace.hprc.tamu.edu/node/g015/26259/lab/tree/shortcourses/Intro_to_TensorFlow/Untitled.ipynb`. The JupyterLab interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Tabs, Settings, and Help. On the left, a file browser sidebar shows a directory structure with files like `images`, `Introduccio...`, `keras_mni...`, `keras.ipynb`, `README...`, and `Untitled.ip...`. The main workspace contains a code cell for `Untitled.ipynb` with the following content:

```
[1]: print("Hello World!")  
Hello World!
```

The code `print("Hello World!")` is enclosed in a red rectangular box, and a large red arrow points from the right towards this box. Below the code cell is an empty input field. The status bar at the bottom indicates the current mode is 'Simple', the kernel is 'Python 3 (ipykernel)', and the file is 'Untitled.ipynb'.

Part II. Introduction to Deep Learning

Deep Learning

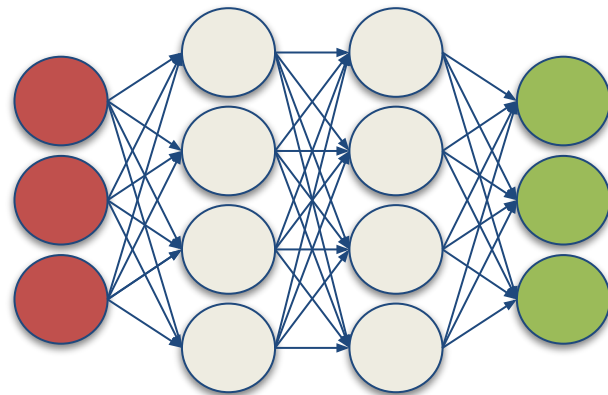
by Ian Goodfellow, Yoshua Bengio, and Aaron Courville

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

Animation of Neutron Networks

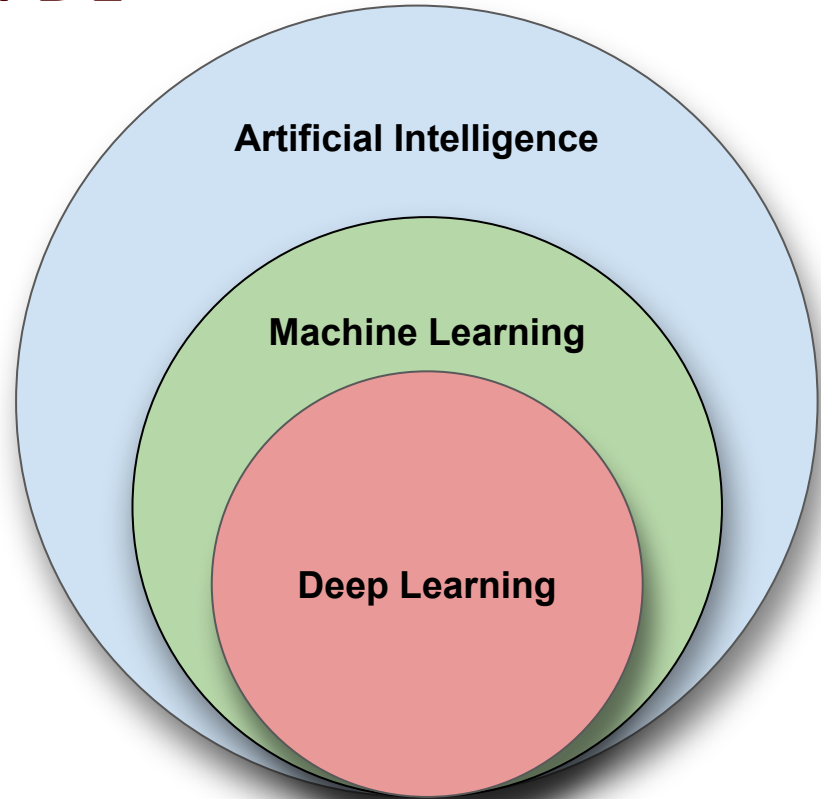
by Grant Sanderson

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



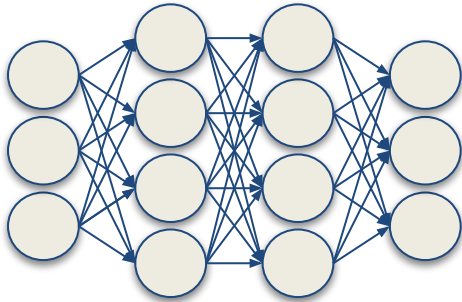
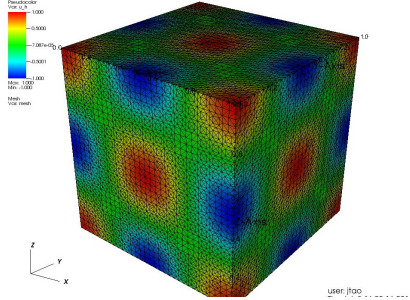
Relationship of AI, ML, and DL

- **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**.

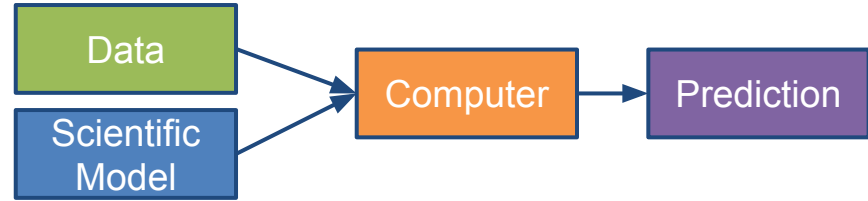


Machine Learning

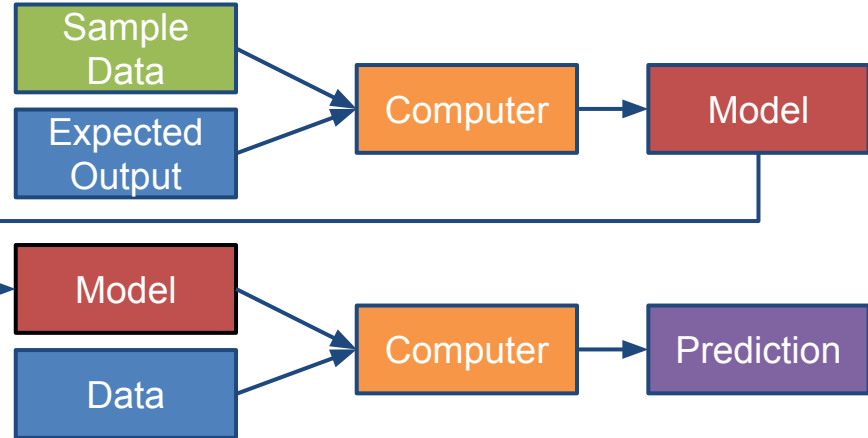
DB: simplest.vtk



Traditional Modeling

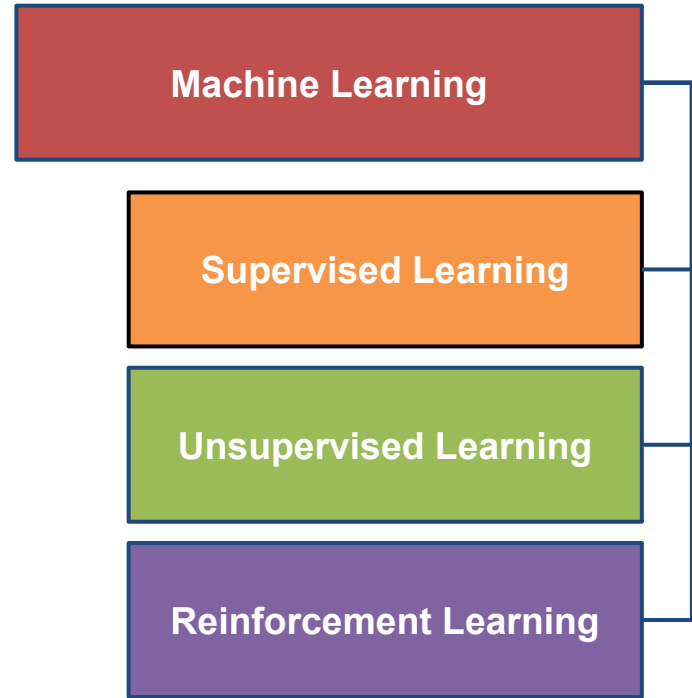


Machine Learning (Supervised Learning)



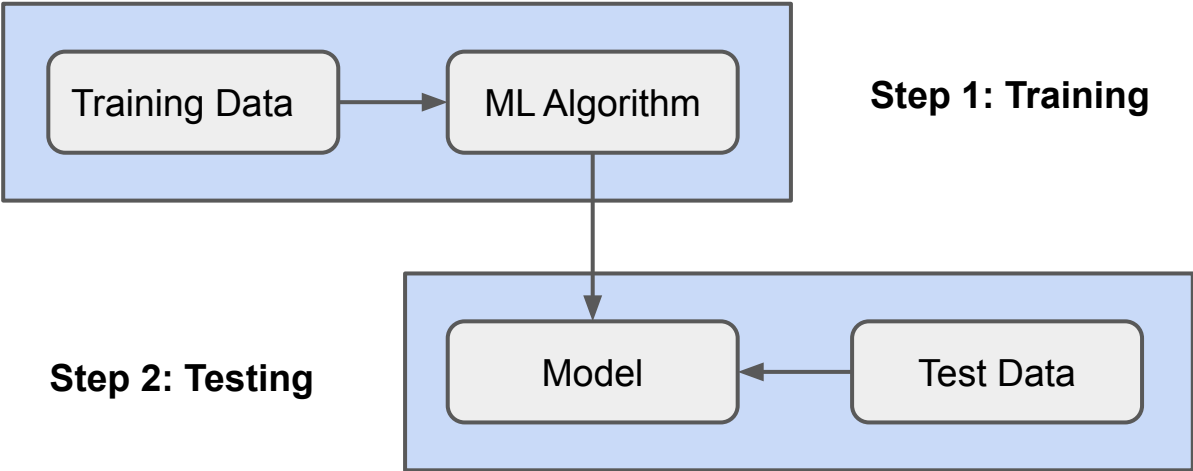
Types of ML Algorithms

- **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.



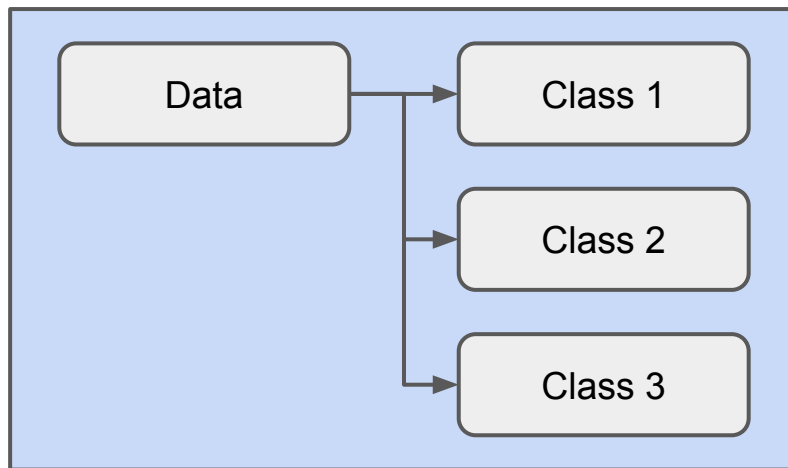
Supervised Learning

When both input variables - X and output variables - Y are known, one can approximate the mapping function from X to Y.



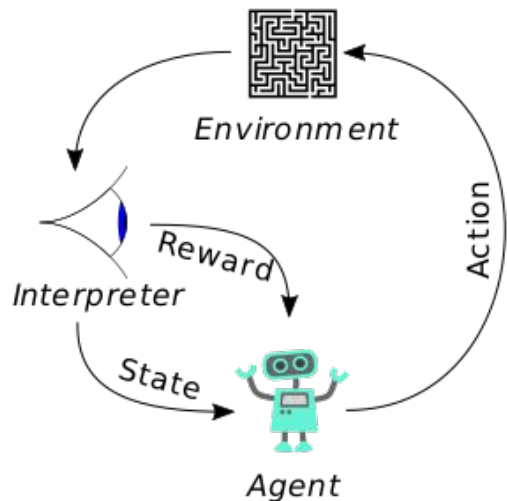
Unsupervised Learning

When only input variables - X are known and the training data is neither classified nor labeled. It is usually used for clustering problems.

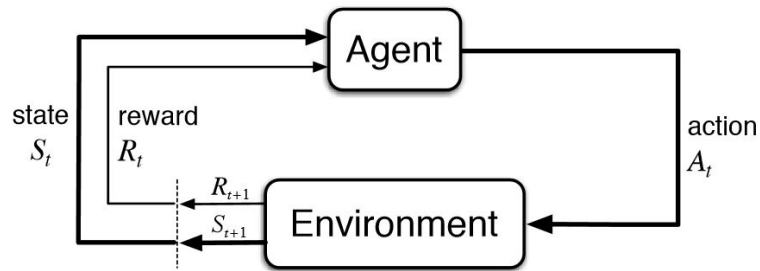


Reinforcement Learning

When the input variables are only available via interacting with the environment, reinforcement learning can be used to train an "agent".



(Image Credit: Wikipedia.org)



(Image Credit: deeplearning4j.org)

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.

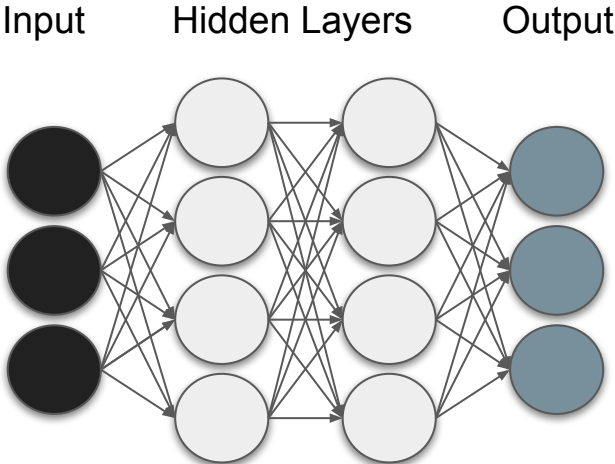
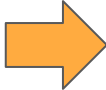
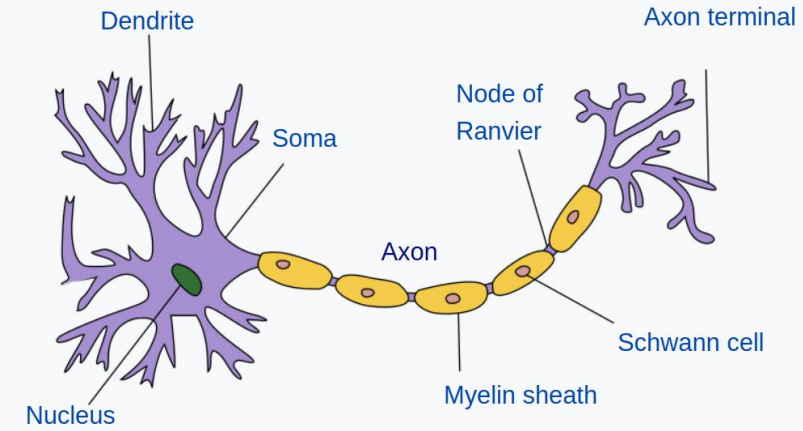
What is Deep Learning?

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.

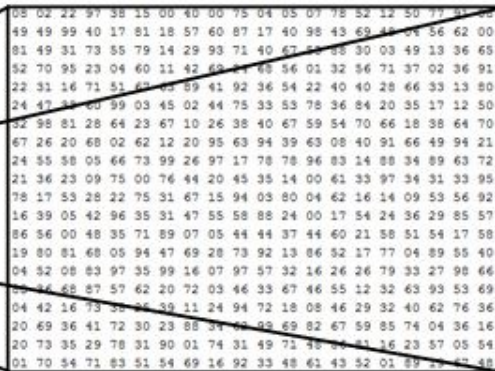
(Source: Wikipedia)

Artificial Neural Network



(Image Credit: Wikipedia)

Inputs and Outputs



What the computer sees

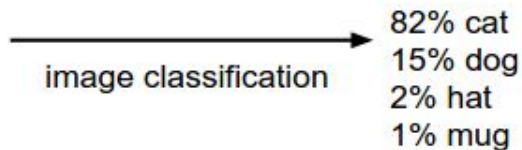
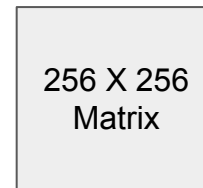
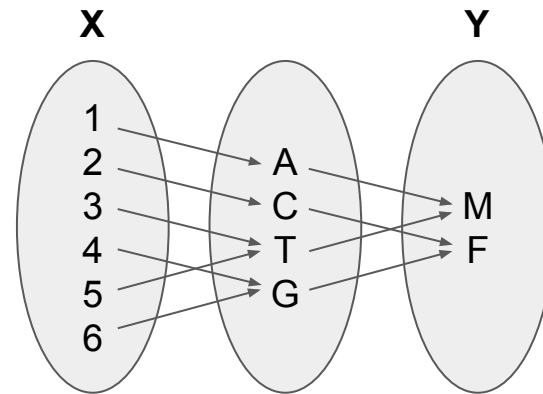


Image from the [Stanford CS231 Course](#)



DL model

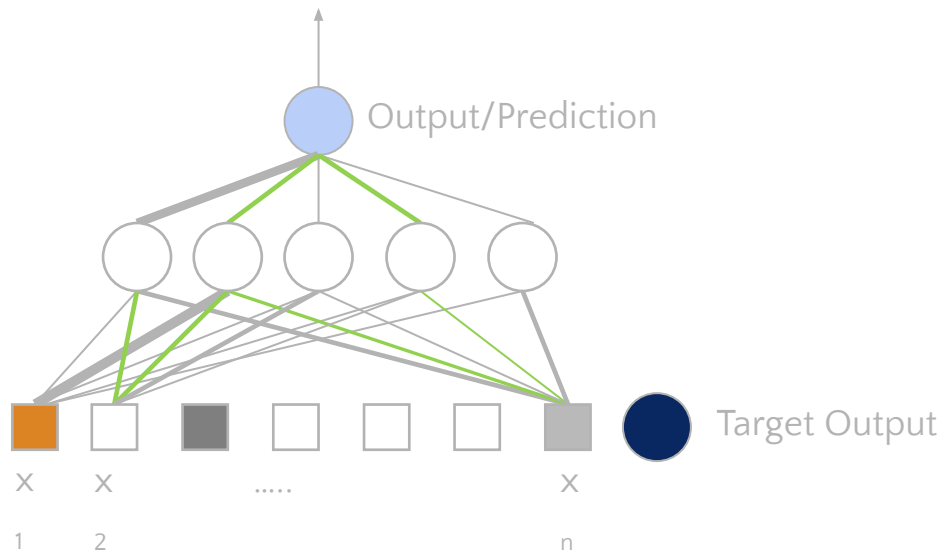
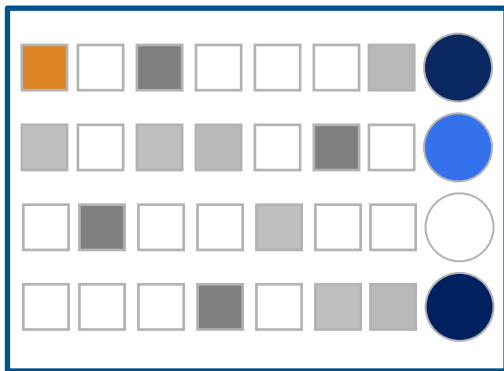
4-Element Vector



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

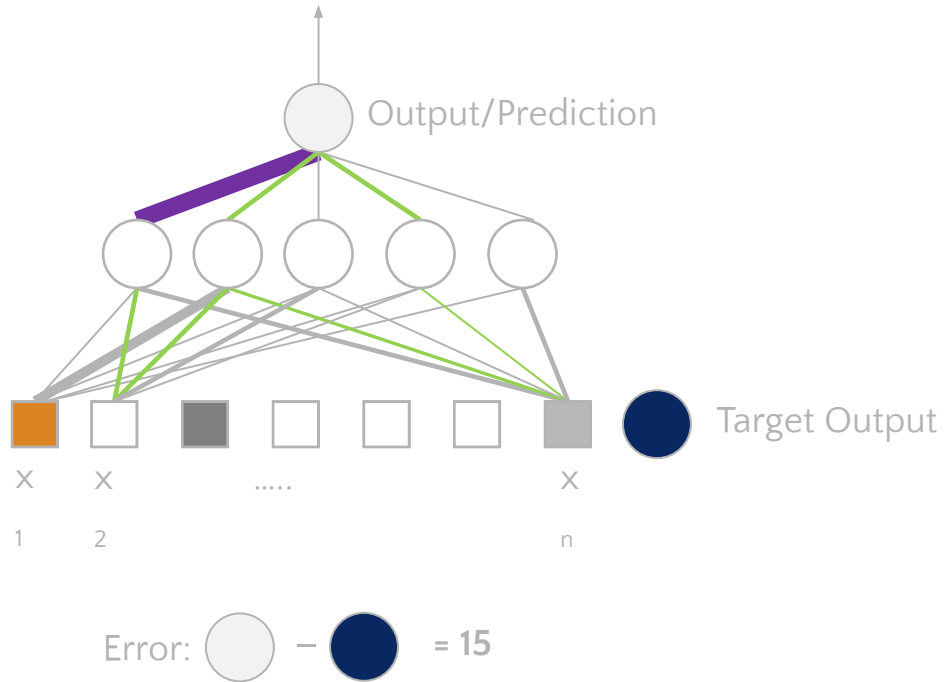
Learning Principle - I

Dataset

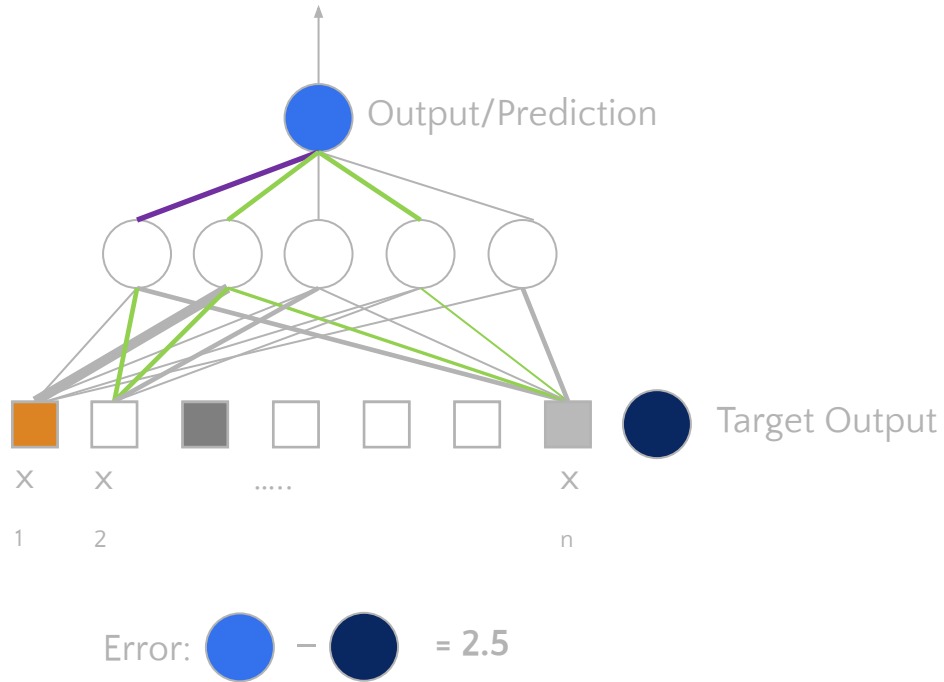


Error:  -  = 5

Learning Principle - II



Learning Principle - III



Credit: nvidia.com

Deep Neural Network as a Universal Approximator

Universal Approximation Theorem

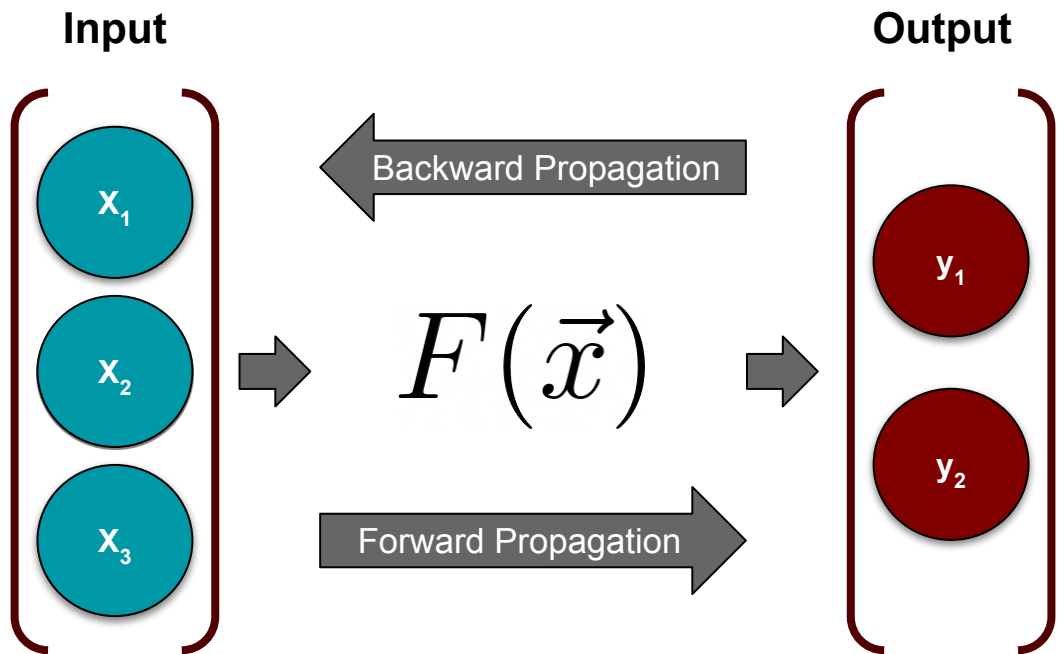
(Cybenko, 1989)

Universal approximation theorems imply that neural networks can represent a wide variety of **functions**.

Pinkus Theorem

(Pinkus, 1999)

Pinkus theorems imply that neural networks can represent **directives of a function** simultaneously.



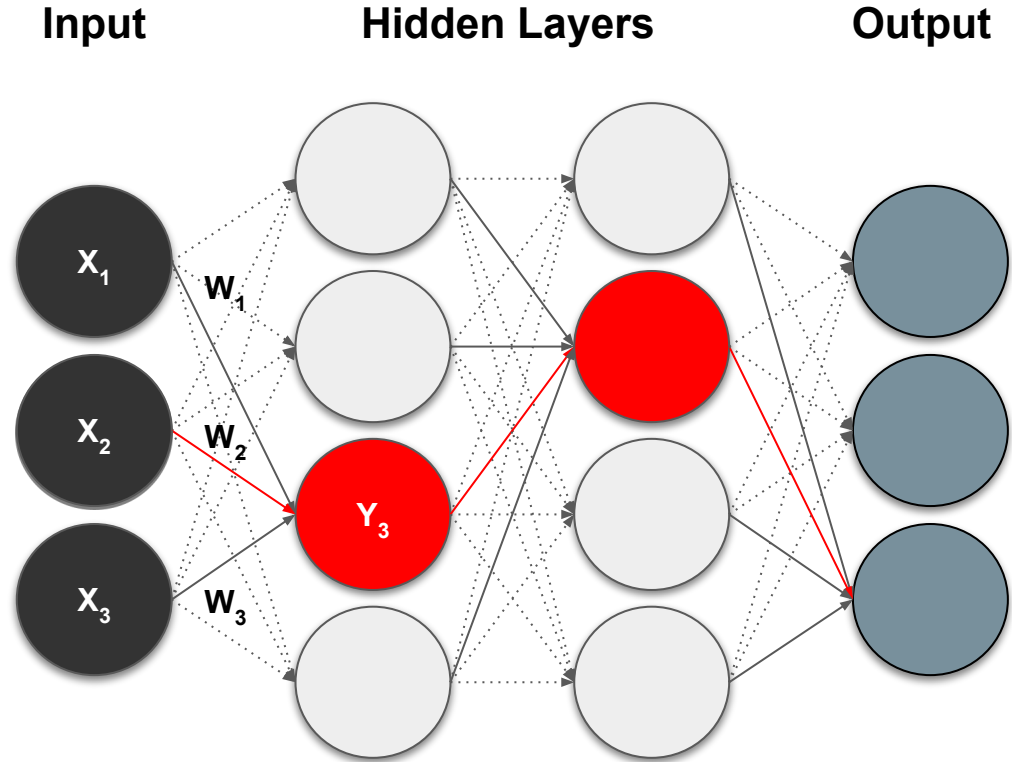
- **Training:** given **input** and **output**, find best-fit F
- **Inference:** given **input** and F , predict **output**

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

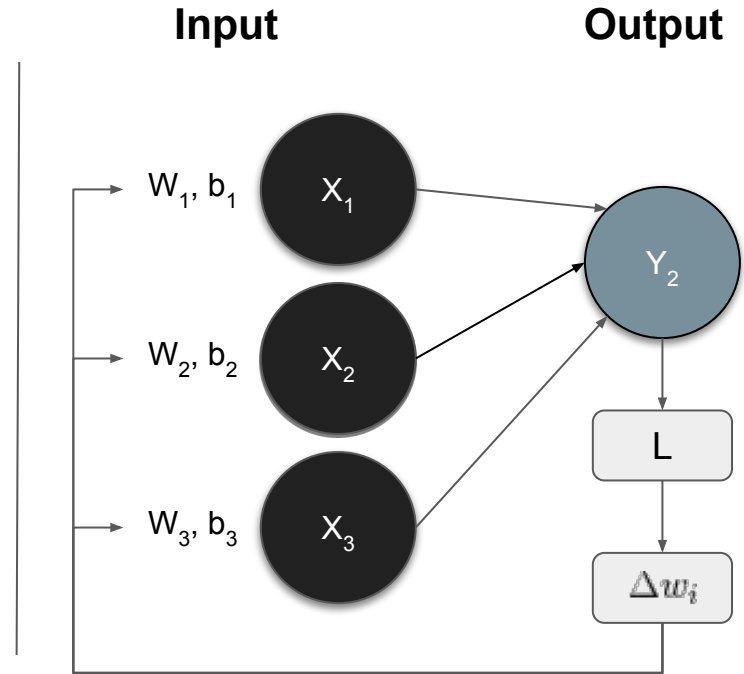
The loss function with regard to weights and biases can be defined as

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

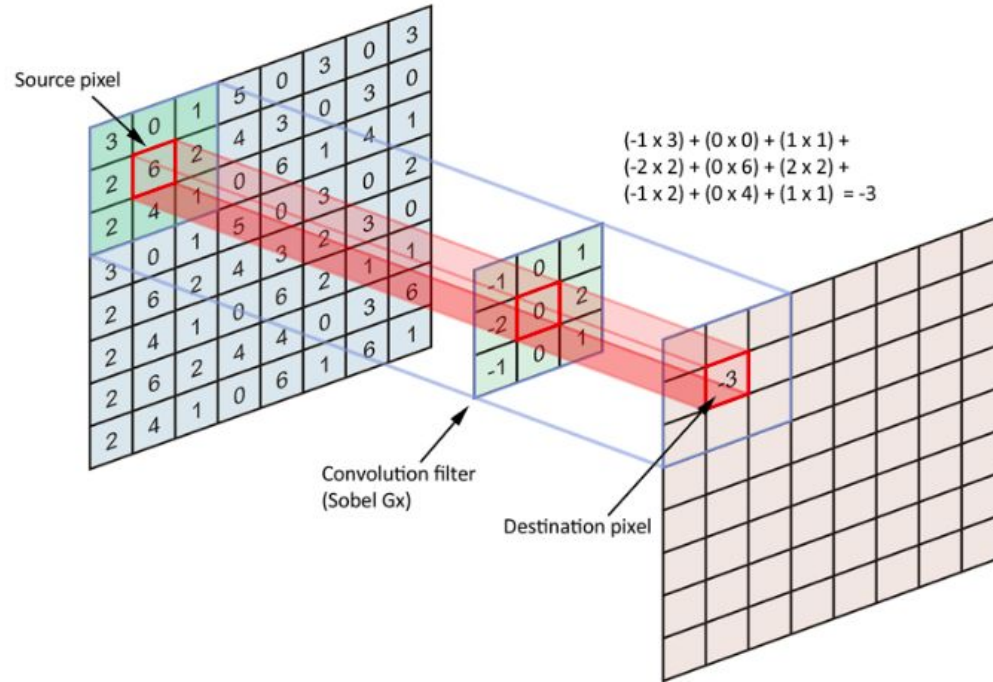
The weight update is computed by moving a step to the opposite direction of the cost gradient.

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

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



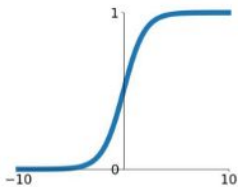
Input

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

Activation Functions

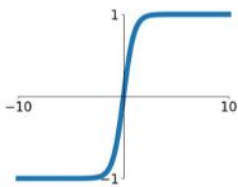
Sigmoid

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



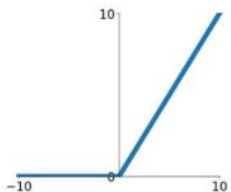
tanh

$$\tanh(x)$$



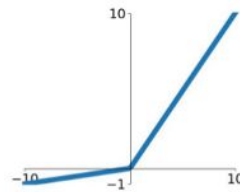
ReLU

$$\max(0, x)$$



Leaky ReLU

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



Maxout

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

ELU

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

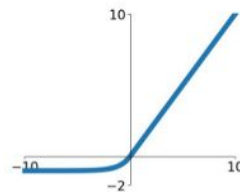


Image Credit: towardsdatascience.com

Introducing Non Linearity (ReLU)

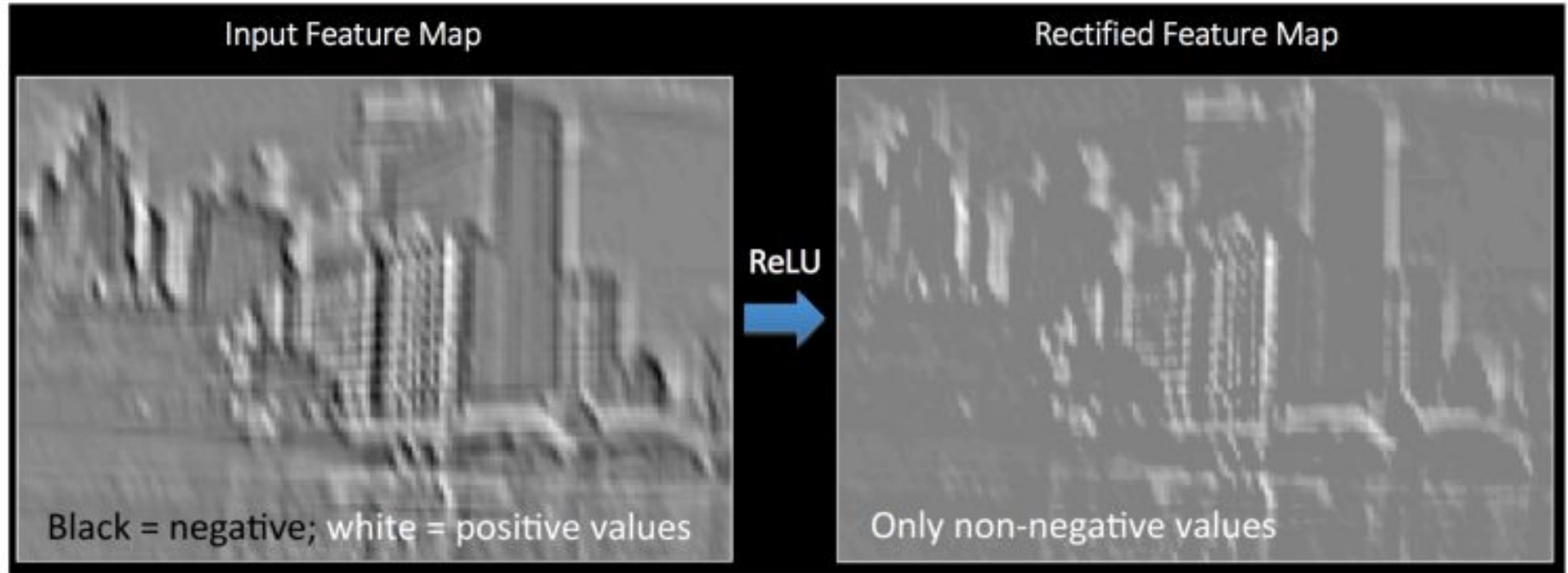
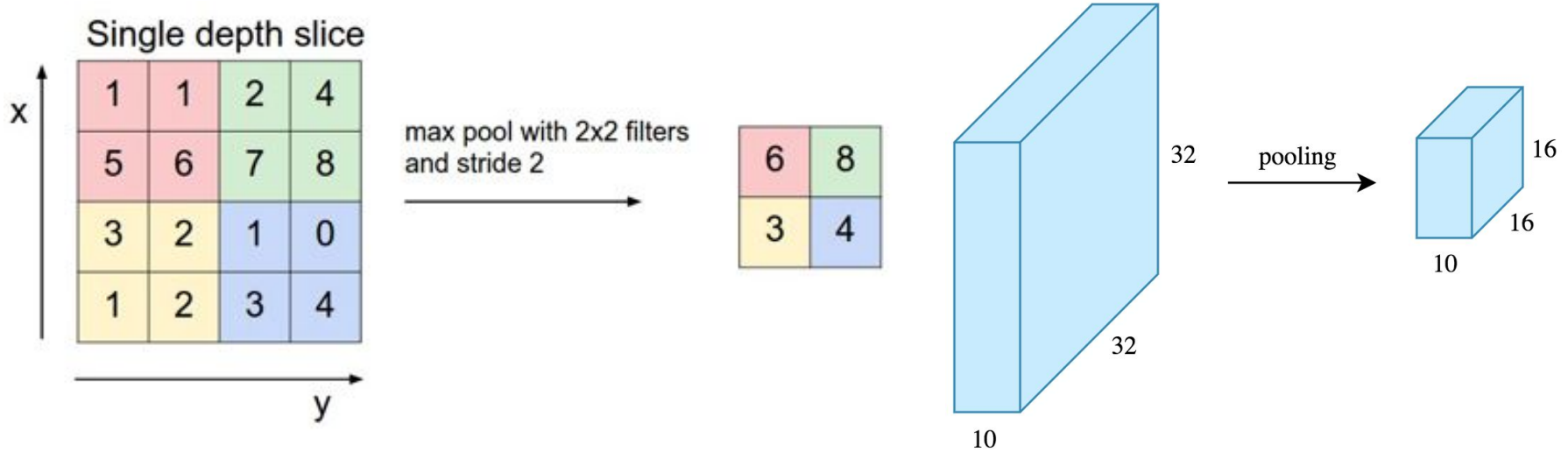


Image Credit: [Deep Learning Methods for Vision | CVPR 2012 Tutorial](#)

Max Pooling



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

Pooling - Max-Pooling and Sum-Pooling

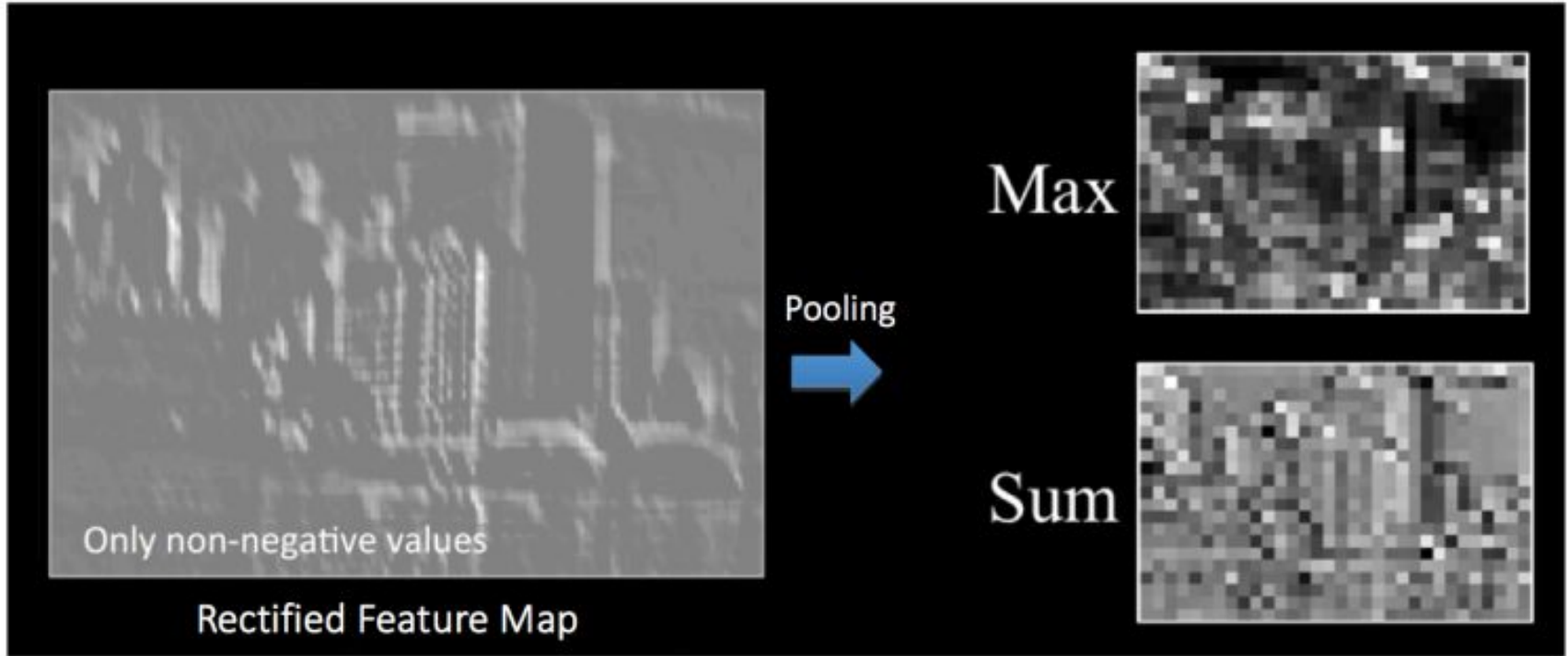
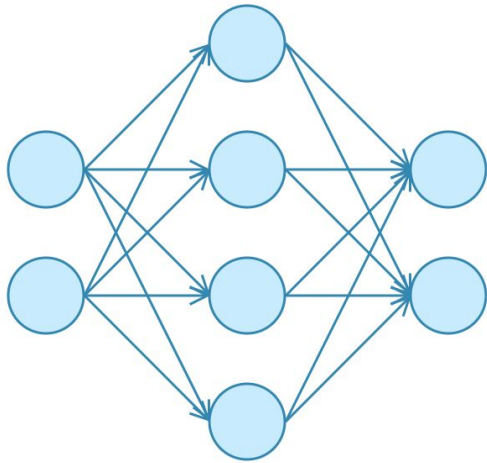


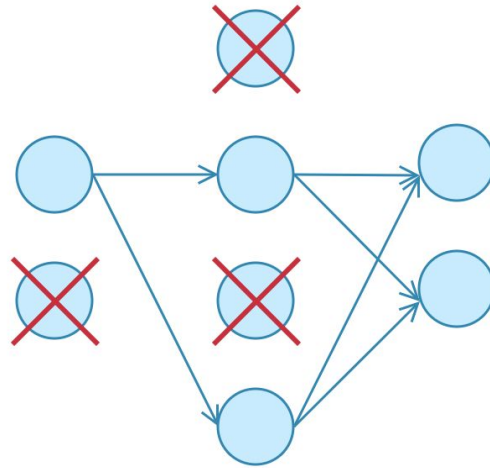
Image Credit: [Deep Learning Methods for Vision | CVPR 2012 Tutorial](#)

CNN Implementation - Drop Out

Dropout is used to prevent overfitting. A neuron is temporarily “dropped” or disabled with probability P during training.



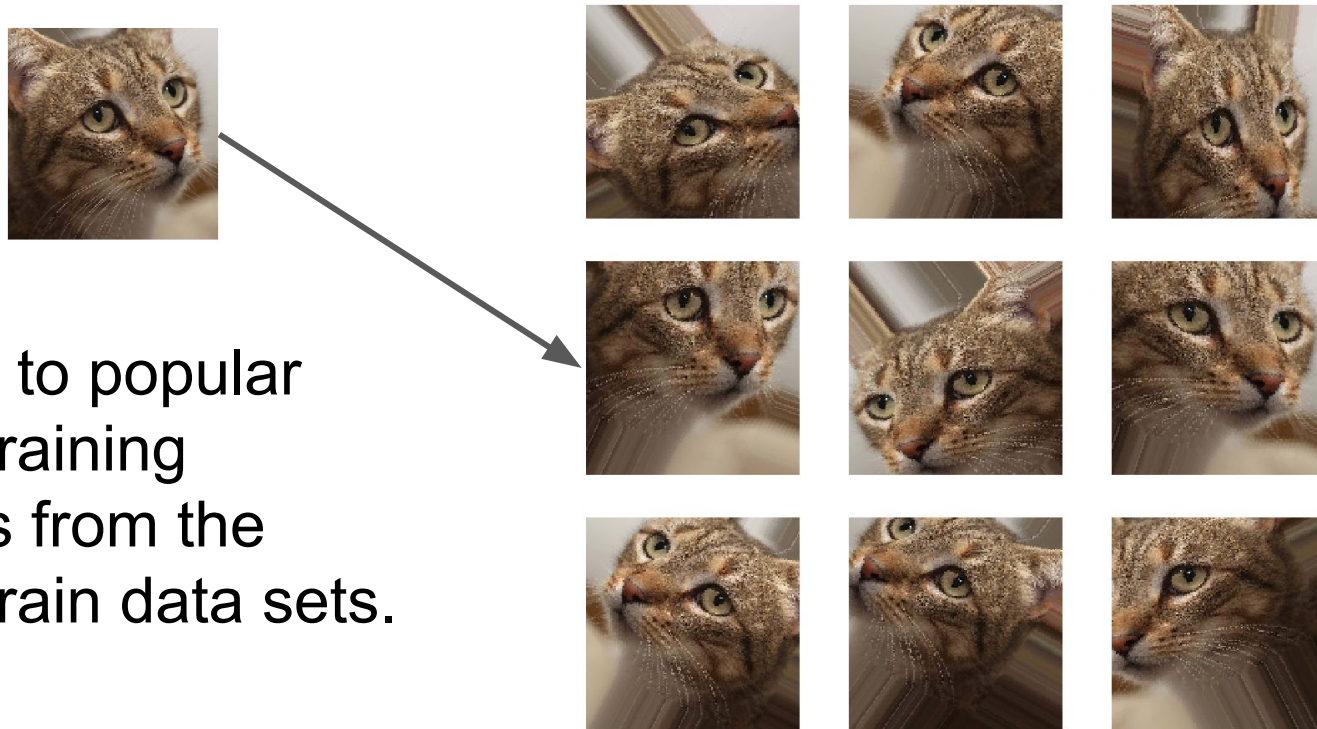
No Dropout



With Dropout

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

CNN Implementation - Data Augmentation (DA)

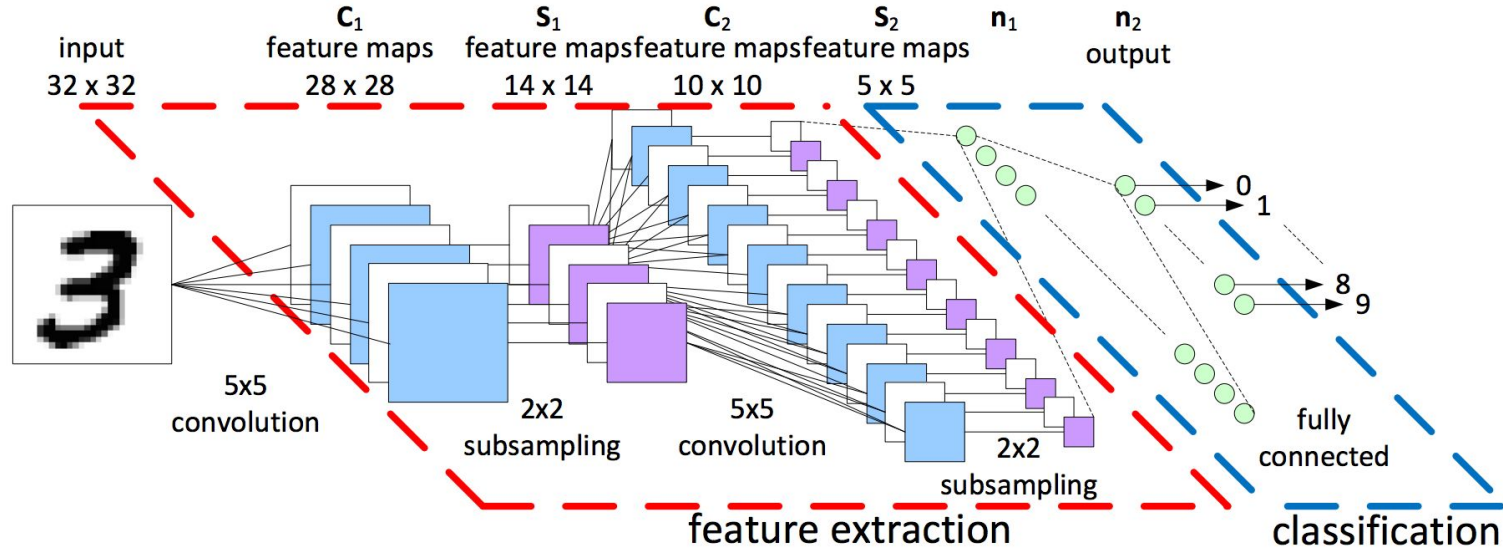


DA helps to popular artificial training instances from the existing train data sets.

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

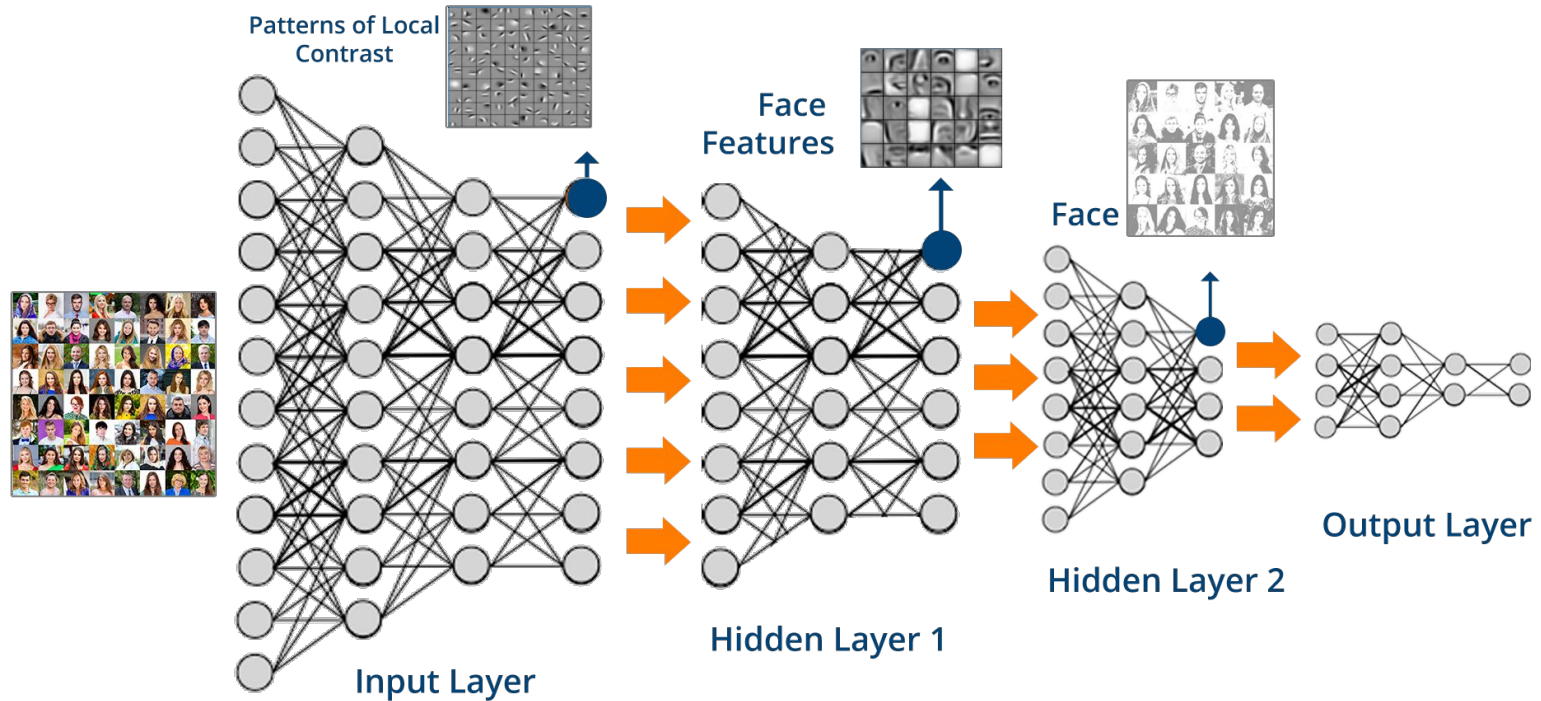
Convolutional Neural Networks

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>)

Deep Learning for Facial Recognition



(Image Credit: www.edureka.co)

Best Practice Guide for Training ML/DL Models

Model Capacity (*what can the model learn?*)

- Overtain on a small data set
- Synthetic data (with known features and properties)

Optimization Issues (*can we make the model learn?*)

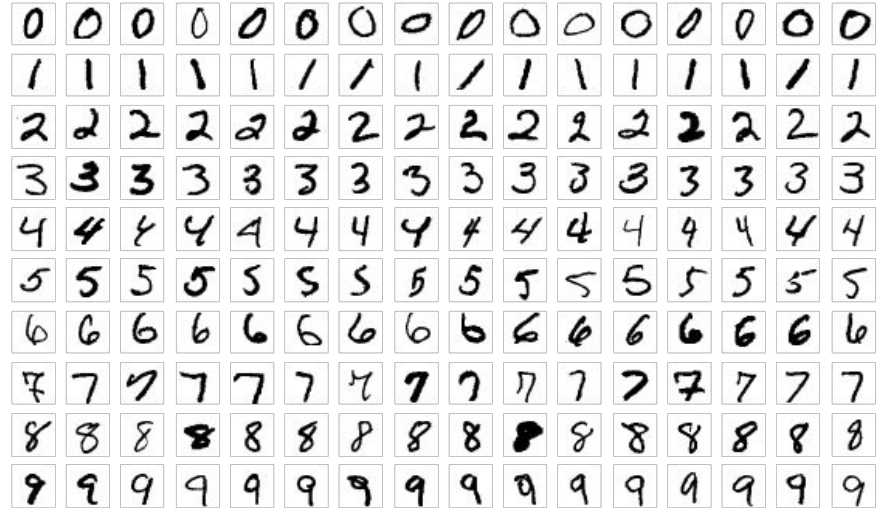
- Look at the learning curves (testing vs training errors)
- Monitor gradient update ratios
- Hand-pick parameters for synthetic data

Other Model "Bugs" (*is the model doing what I want it to do?*)

- Generate samples from your model (if you can)
- Visualize learned representations (e.g., embeddings, nearest neighbors)
- Error analysis (examples where the model is failing, most "confident" errors)
- Simplify the problem/model
- Increase capacity, sweep hyperparameters

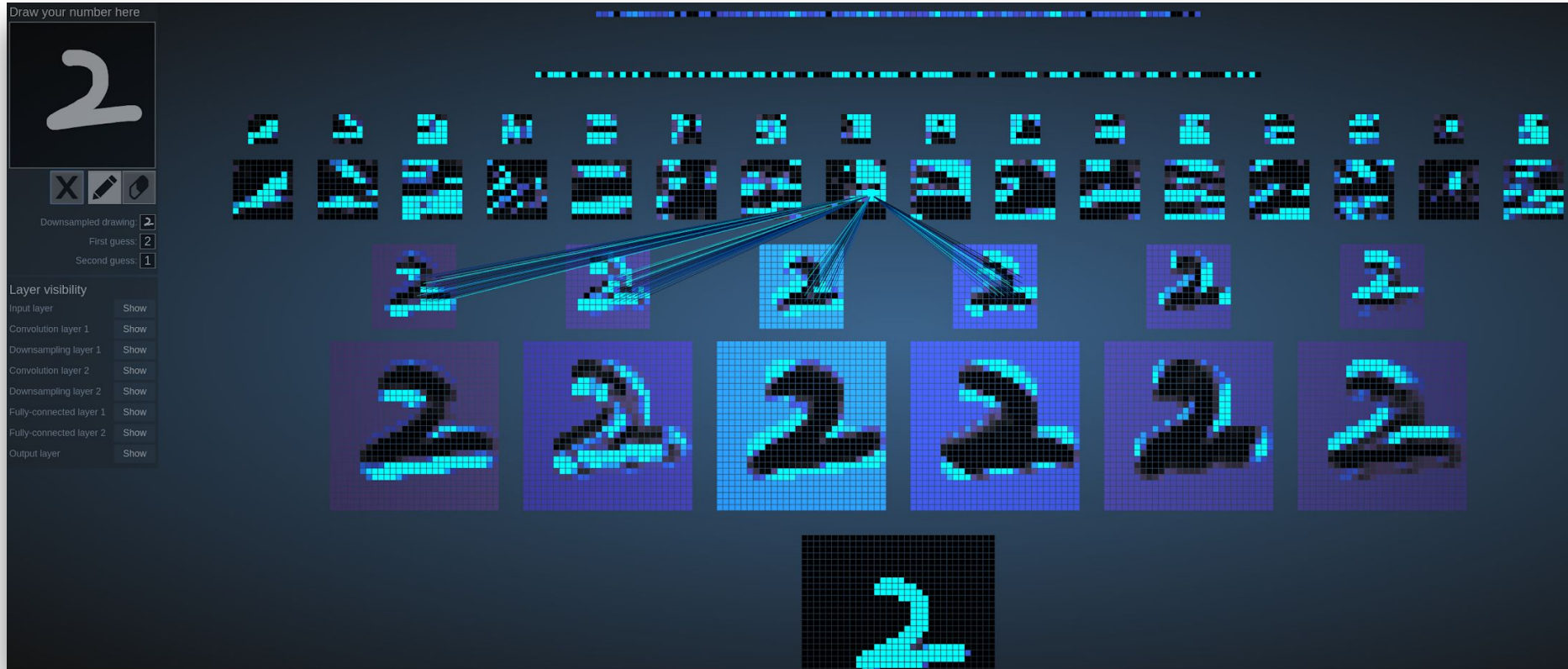
MNIST - Introduction

- **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/>)

Neural Network Playground

DATA

Which dataset do you want to use?



Ratio of training to test data: 50%



Noise: 0



Batch size: 10



REGENERATE

FEATURES

Which properties do you want to feed in?



+ - 2 HIDDEN LAYERS

+ -

4 neurons



This is the output from one neuron. Hover to see it larger.

+ -

2 neurons

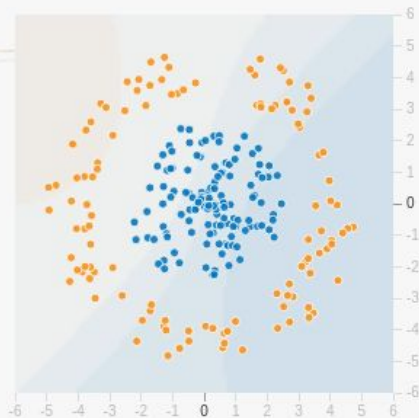


The outputs are mixed with varying weights, shown by the thickness of the lines.

OUTPUT

Test loss 0.522

Training loss 0.496



Colors shows data, neuron and weight values.



Show test data

Discretize output

(Image Credit: <http://playground.tensorflow.org/>)

Part III. Introduction to TensorFlow

TensorFlow Official Website
<http://www.tensorflow.org>



A Brief History of TensorFlow

TensorFlow is an end-to-end FOSS (free and open source software) library for dataflow, differentiable programming. TensorFlow is one of the most popular program frameworks for building machine learning applications.

- Google Brain built **DistBelief** in 2011 for internal usage.
- TensorFlow 1.0.0 was released on Feb 11, 2017
- TensorFlow 2.0 was released in Jan 2018.
- The latest stable version of TensorFlow is 2.4.1 as of May 2021.

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.

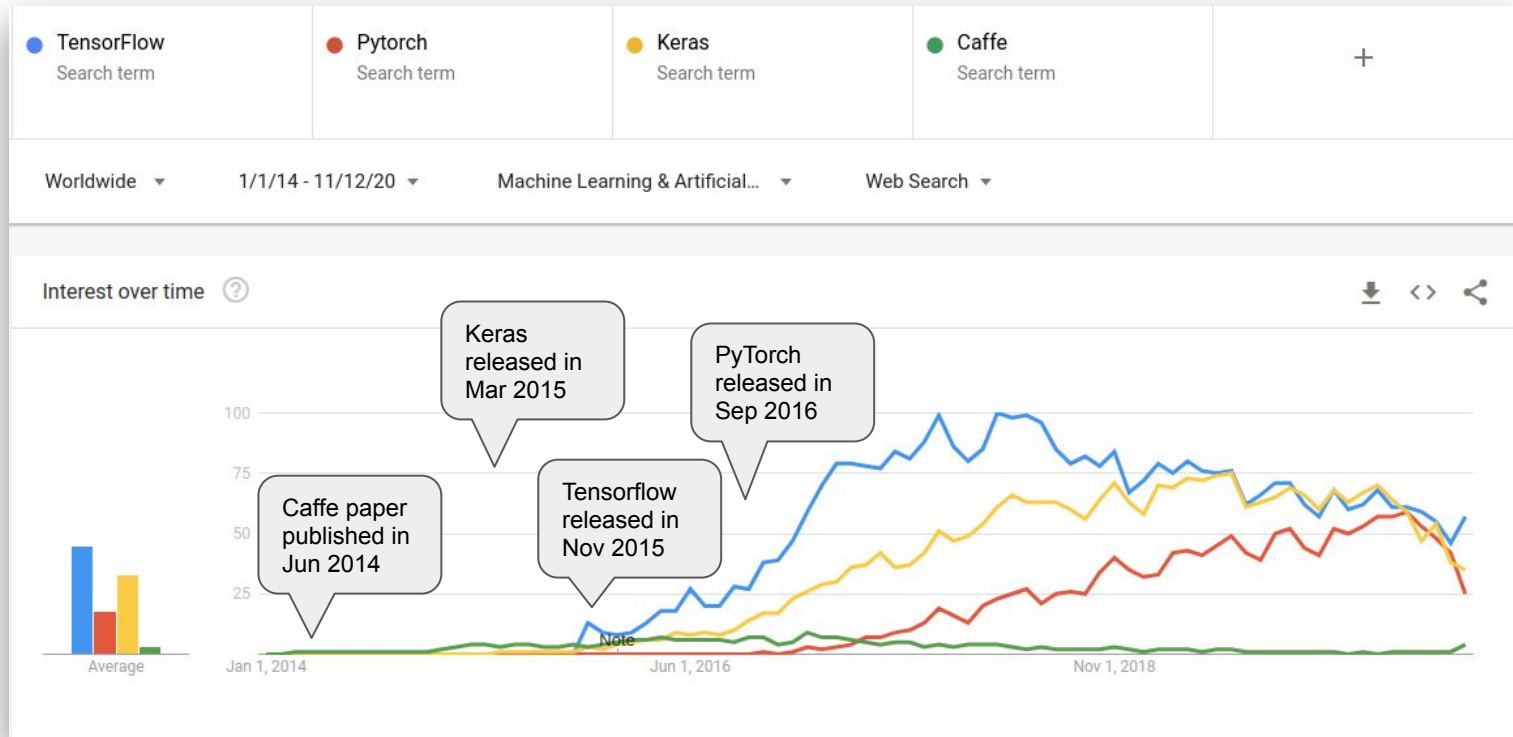


Keras is a high-level neural networks **API**, written in Python and capable of running on top of *TensorFlow*, *CNTK*, or *Theano*. It was developed with a focus on enabling fast experimentation.



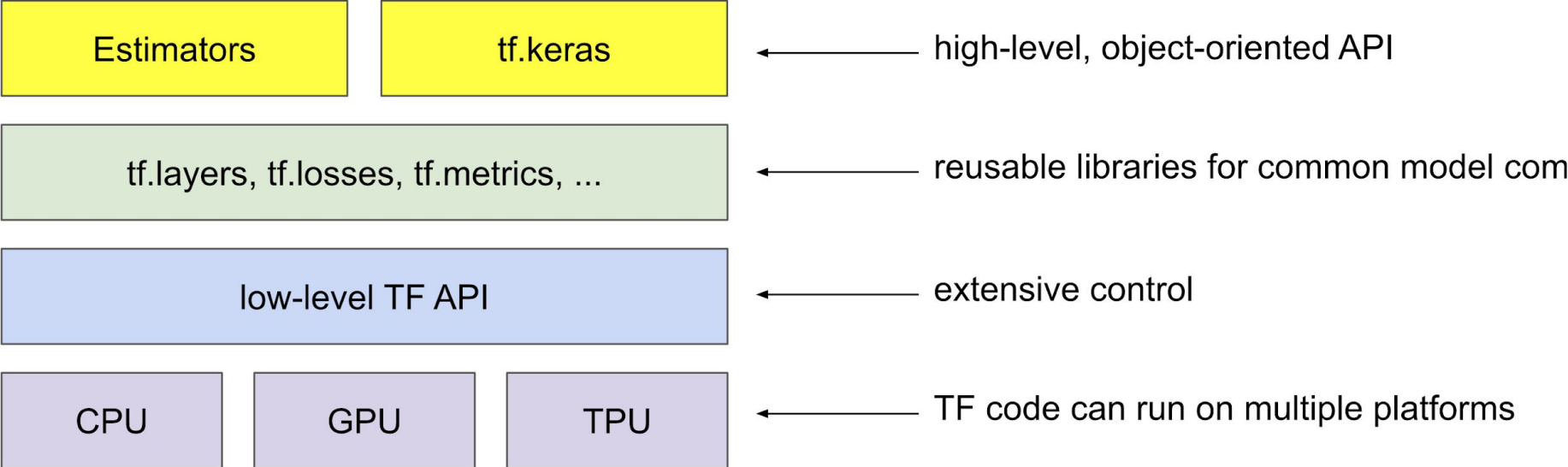
PyTorch is an open source machine learning **framework** that accelerates the path from research prototyping to production deployment.

Google Trends for Popular ML Frameworks



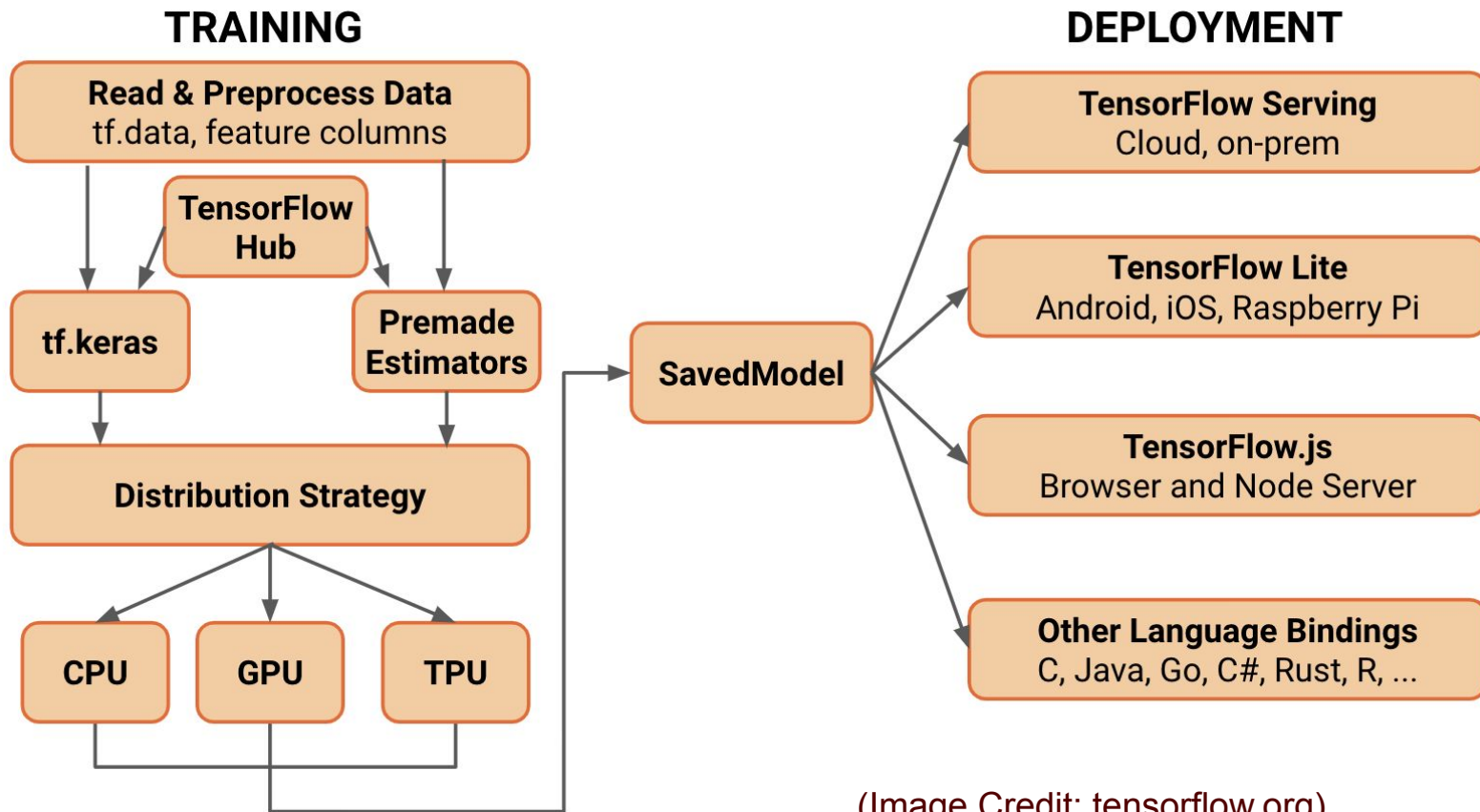
(Image Credit: <https://trends.google.com/>)

TensorFlow 2.0 Toolkits



(Image Credit: tensorflow.org)

Architecture of TF 2.0



(Image Credit: tensorflow.org)

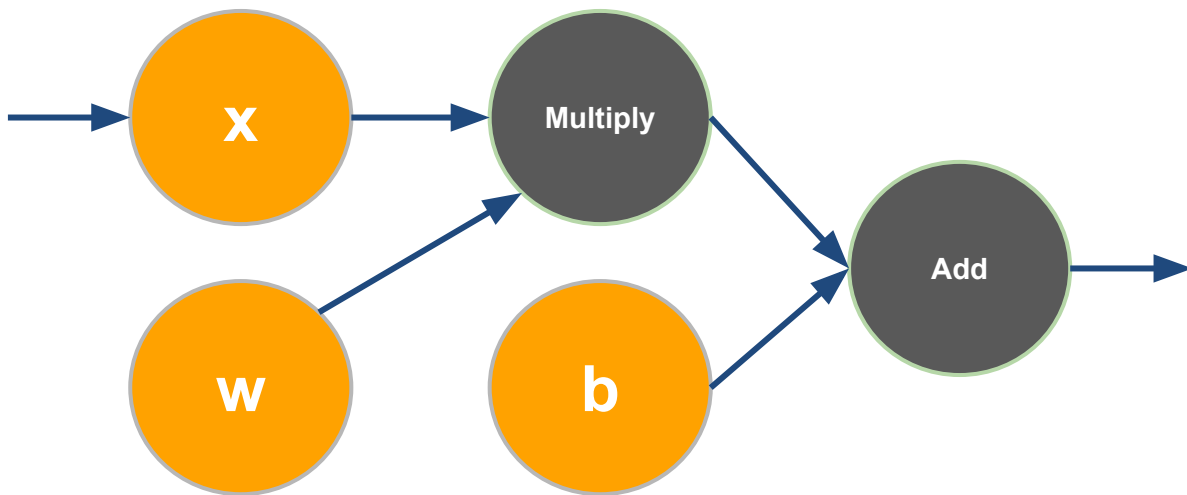
What is a Tensor in TensorFlow?

- **TensorFlow** uses a tensor data structure to represent all data. A TensorFlow tensor as an **n-dimensional array** or list. 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	...

Computational Graph in TF 2.0

```
x = tf.random.normal(shape=(10,10))  
w = tf.Variable(tf.random.normal(shape=(10,5)))  
b = tf.Variable(tf.random.normal(shape=(5,)))  
linear_model = w * x + b
```



A Connected Pipeline for the Flow of Tensors



(Image Credit: Plumber Game by Mobiloids)

TensorFlow Data Types

Basic TensorFlow data types include:

- `int[8|16|32|64]`, `float[16|32|64]`, `double`
- `bool`
- `string`

With `tf.cast()`, the data types of variables could be converted.

Hello World with TensorFlow

```
import tensorflow as tf  
  
v = tf.constant("Hello World!")  
  
tf.print(v)
```

TensorFlow Constants

TensorFlow provides several operations to generate constant tensors.

```
import tensorflow as tf

x = tf.constant(1, tf.int32)
zeros = tf.zeros([2, 3], tf.int32)
ones = tf.ones([2, 3], tf.int32)
y = x *(zeros + ones + ones)

tf.print(y)
```

TensorFlow Variables

TensorFlow variables can represent shared, persistent state manipulated by your program. **Weights** and **biases** are usually stored in variables.

```
import tensorflow as tf
```

```
W = tf.Variable(tf.random.normal([2,2], stddev=0.1),  
name = "W")
```

```
b = tf.Variable(tf.zeros(shape=(2)), name="b")
```

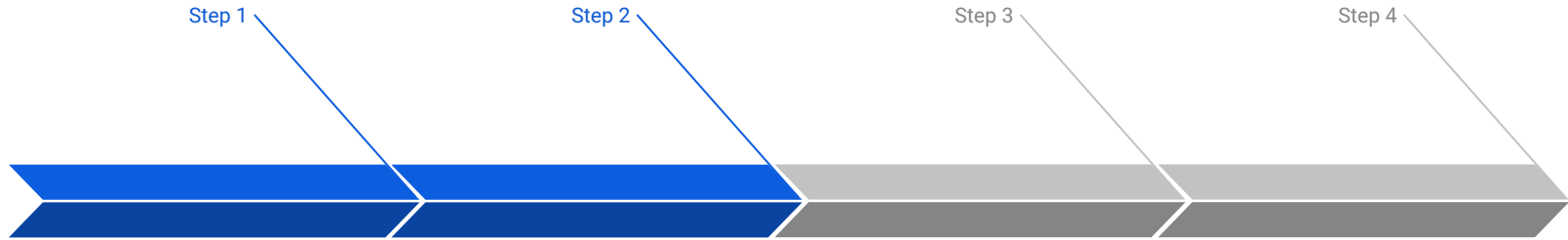

GPU Acceleration

TensorFlow automatically decides if to use the CPU or GPU. One can explicitly pick a device to use. The string ends with **CPU/GPU:<N>** if the tensor is placed on the **N-th CPU/GPU** on the host.

```
# Force execution on CPU
with tf.device("CPU:0"):
    do_something()
```

```
# Force execution on GPU #0/1/2/... if available
if tf.config.experimental.list_physical_devices("GPU"):
    with tf.device("GPU:0"):
        do_something_else()
```

Machine Learning Workflow with tf.keras



Prepare Train Data

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

Define Model

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

Training Configuration

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

Train Model

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

tf.keras Built-in Datasets

- tf.keras provides many popular reference datasets that could be used for demonstrating and testing deep neural network models. To name a few,
 - Boston Housing (regression)
 - CIFAR100 (classification of 100 image labels)
 - MNIST (classification of 10 digits)
 - Fashion-MNIST (classification of 10 fashion categories)
 - Reuters News (multiclass text classification)
- The built-in datasets could be easily read in for training purpose. E.g.,

```
from tensorflow.keras.datasets import boston_housing
(x_train, y_train), (x_test, y_test) = boston_housing.load_data()
```

Prepare Datasets for tf.keras

In order to train a deep neural network model with Keras, the input data sets needs to be **cleaned**, **balanced**, **transformed**, **scaled**, and **split**.

- Balance the classes. Unbalanced classes will interfere with training.
- Transform the categorical variables into one-hot encoded variables.
- Extract the X (variables) and y (targets) values for the training and testing datasets.
- Scale/normalize the variables.
- Shuffle and split the dataset into training and testing datasets

One-hot encoding

Dog	Cat	Horse
1	0	0
0	1	0
0	0	1

Numerical encoding

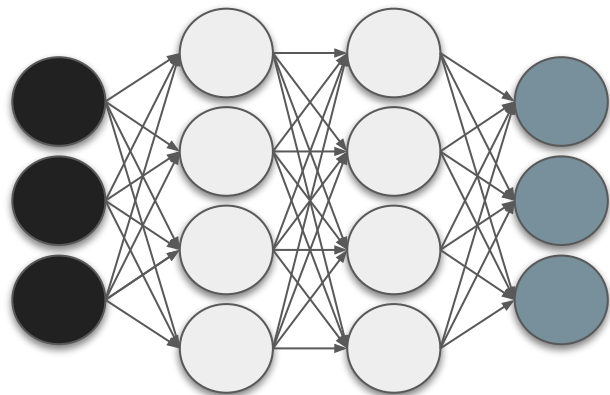
Dog	Cat	Horse
1	2	3

Create a tf.keras Model

- Layers are the fundamental building blocks of **tf.keras** models.
- The **Sequential** model is a linear stack of layers.
- A **Sequential** model can be created with a list of layer instances to the constructor or added with the **.add()** method.
- The input shape/dimension of the first layer need to be set.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,
Activation

model = Sequential([
    Dense(64, activation='relu', input_dim=20),
    Dense(10, activation='softmax')
])
```



Input

Hidden Layers

Output

Compile a tf.keras Model

The **compile** method of a Keras model configures the learning process before the model is trained. The following 3 arguments need to be set (the optimizer and loss function are required).

- An optimizer: **Adam**, **AdaGrad**, **SGD**, **RMSprop**, etc.
- A loss function: **mean_squared_error**, **mean_absolute_error**, **mean_squared_logarithmic_error**, **categorical_crossentropy**, **kullback_leibler_divergence**, etc.
- A list of measurement metrics: **accuracy**, **binary_accuracy**, **categorical_accuracy**, etc.

Train and Evaluate a tf.keras Model

tf.keras is trained on NumPy arrays of input data and labels. The training is done with the

- **fit()** function of the model class. In the fit function, the following two hyperparameters can be set:
 - **number of epochs**
 - **batch size**
- **evaluate()** function returns the loss value & metrics values for the model in test mode.
- **summary()** function prints out the network architecture.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_11 (Dense)	(None, 64)	1344
dense_12 (Dense)	(None, 10)	650

Total params: 1,994

Trainable params: 1,994

Non-trainable params: 0

None

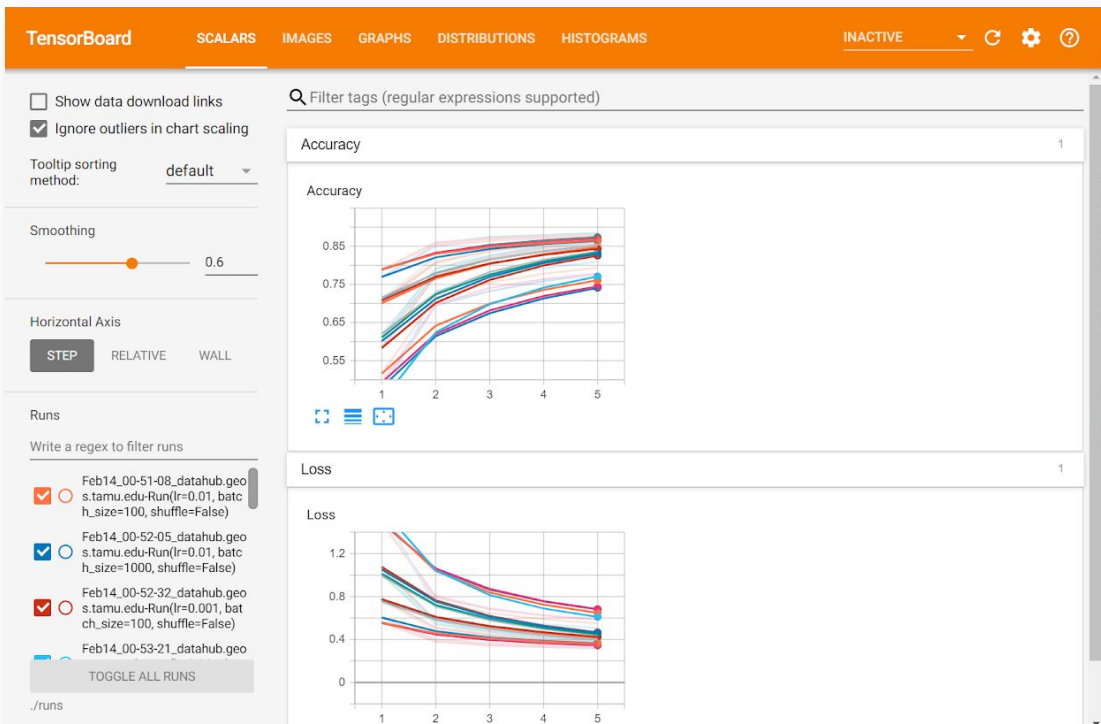
Make Predictions and More

After the model is trained,

- **predict()** function of the model class could be used to generate output predictions for the input samples.
- **get_weights()** function returns a list of all weight tensors in the model, as Numpy arrays.
- **to_json()** returns a representation of the model as a JSON string. Note that the representation does not include the weights, only the architecture.
- **save_weights(filepath)** saves the weights of the model as a HDF5 file.

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 TensorFlow models and metrics into a directory for visualization within the TensorBoard UI.



Hands-on Session #1

Getting Started with TensorFlow



Hands-on Session #2

Classify Handwritten Digits with TensorFlow

