MACHINE LEARNING WITH PYTHON

Jacalyn Huband
Computational Research Support Specialist

E: jmh5ad@virginia.edu

Alois D'Uston
Computational Support Temp

E: ald6fd@virginia.edu

Gladys Andino
Senior Computational Scientist

E: gka6a@virginia.edu

9 June 2022

University of Virginia | Research Computing

Topics

- Overview of Machine Learning
- Decision Trees
- Random Forest

  >> Break <<

- Overview of Neural Networks
- Tensorflow/Keras
- PyTorch

  >> Break <<

- Overview of Parallelizing Deep Learning

What is machine learning?

A branch of artificial intelligence where computers learn from data, and adapt the computational models to enhance performance.

A method of analysis that allows computers to reveal information within data.

What is machine learning?

The “learning” is not the type of learning that you and I do.

It is a systematic approach to finding an appropriate data transformation from inputs to output.

Why machine learning?

- Computers can sort through data faster than humans can.
- Computers can identify patterns quickly and use these patterns for predictions or classifications.
- Machine learning can handle noisy data – it doesn't find a perfect answer, but rather a "really good" answer.

Problems that ML can solve

- Regression techniques
    - Determines a mathematical model for the relationship among features or attributes so that an outcome can be predicted.
    - Results can be any value within a possible range (e.g., what will the average Earth temperature be in 2050?)
- Classification problem
    - Identifies a combination of attributes that best fits a class or category so that an object can be classified.
    - Results can be from a list of known possibilities (e.g., is the tumor benign or malignant?)

Problems that ML can solve

- Regression techniques
  - Determines a mathematical model for the relationship among features or attributes so that an outcome can be predicted.
  - Results can be any value within a possible range (e.g., what will the average Earth temperature be in 2050?)

- Classification problem
  - Identifies a combination of attributes that best fits a class or category so that an object can be classified.
  - Results can be from a list of known possibilities (e.g., is the tumor benign or malignant?)

Types of Machine Learning. . .

- Supervised Learning:
    - A data set exists where the samples can be categorized into two or more classifications.
    - The computer uses the data set to learn how to predict the classification of an unknown sample.
    - Examples include Decision Trees and Deep Learning
- Unsupervised Learning:
    - The collected data has no known classification or pattern.
    - The computer must identify the groups or hidden structures within the data.
    - Examples include Dendograms, K-means clustering, Self-organizing Maps
- Reinforcement Learning:
    - Computer learns from positive or negative feedback
    - Example includes Swarm intelligence

Types of Machine Learning. . .

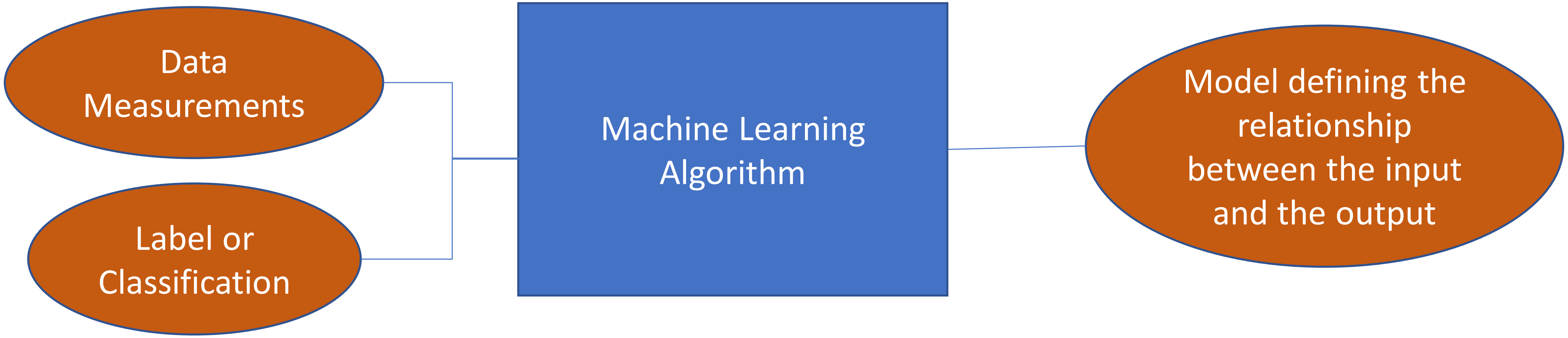
- Supervised Learning:
  - A data set exists where the samples can be categorized into two or more classifications.
  - The computer uses the data set to learn how to predict the classification of an unknown sample.
  - Examples include Decision Trees and Deep Learning
- Unsupervised Learning:
  - The collected data has no know classification or pattern.
  - The computer must identify the groups or hidden structures within the data.
  - Examples include Dendograms, K-means clustering, Self-organizing Maps
- Reinforcement Learning:
  - Computer learns from positive or negative feedback
  - Example includes Swarm intelligence

Data for Machine Learning

- For many Machine Learning algorithms, the data are expected to be in a table format, where:
  - each row represents an object, and
  - each column has the measurements for a specific attribute or feature of the object
- For supervised learning, the classifications of the objects must be known.
- The data with known classifications are divided into a training set and a testing set.
- The data are used to develop a model.
  - The training data are submitted to an algorithm that will fit a model to the data.
  - The test data are submitted to the model to produce predicted classifications and determine the accuracy of the model.
- Finally, the model can be used to predict classifications for "unknown" data.

Ideas behind Machine Learning

- The algorithm determines the best mathematical model for the code
- However, you still need to provide a “framework” for the algorithm.
- The framework provides the algorithm with tools for performing the learning



DECISION TREES

Decision Tree: Overview

- A classification algorithm within supervised learning.

> Motivating Question:
> Given a set of data, can we determine which attributes should be tested first to predict a category or outcome (i.e., which attributes lead to "high information gain") ?

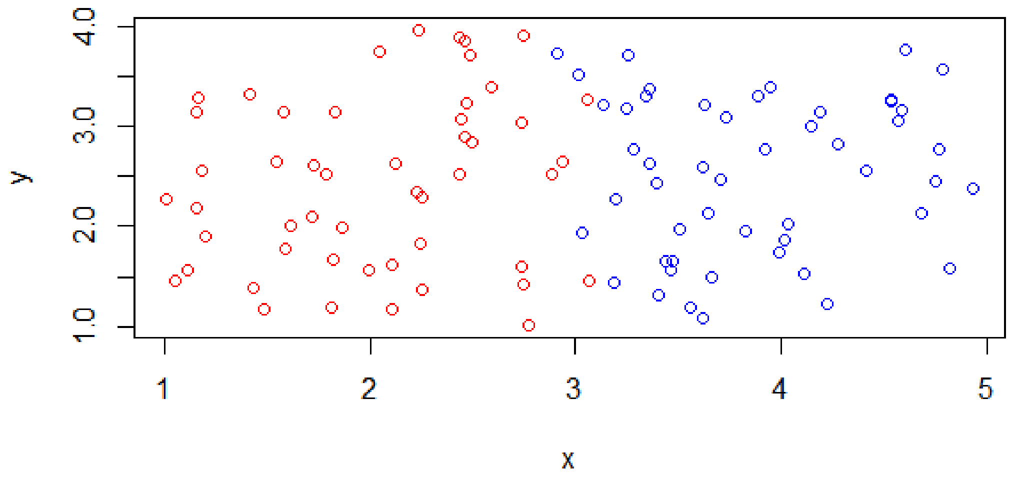
- The algorithm determines a set of questions or tests that will guide it toward a classification of an observation.
- It organizes a series of attribute tests into a tree-structure to help determine classification of the unlabeled data.

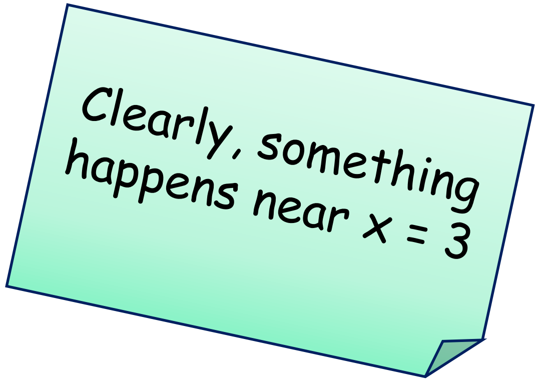
Decision Trees: How do they work?

Suppose we have

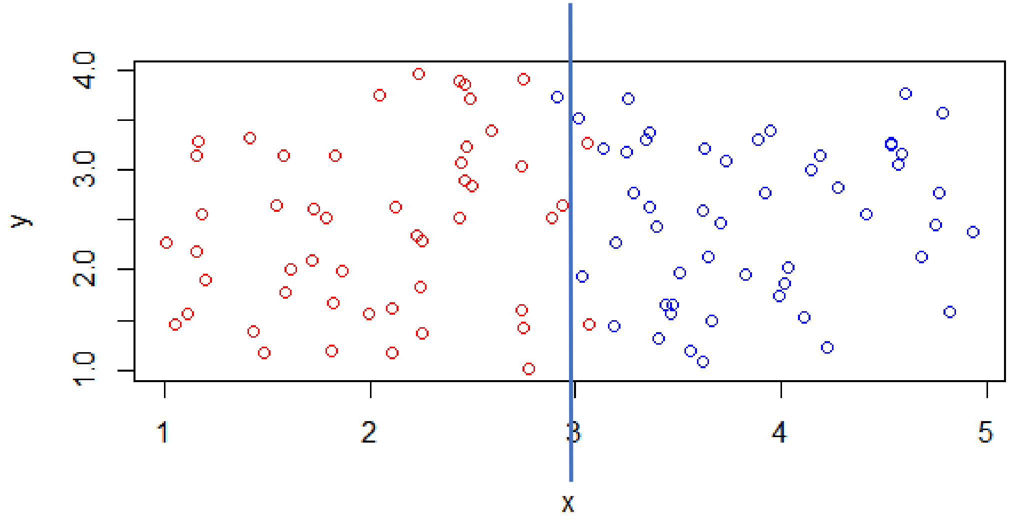
- a group of people, each one with a tumor, and
- two measurements (x, y) for each tumor.

Plotting the data, and coloring the points red for malignant tumors and blue for benign tumors, we might see a plot as follows:



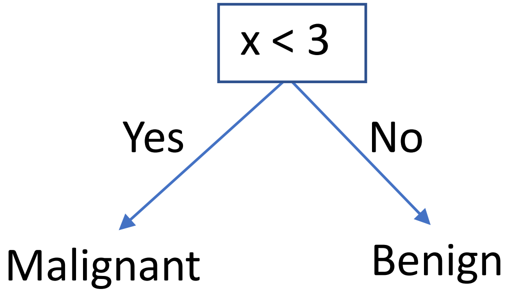


Decision Trees: How do they work?



With very few errors, we can use x=3 as our “decision” to categorize the tumor as malignant vs. benign

Resulting decision tree:

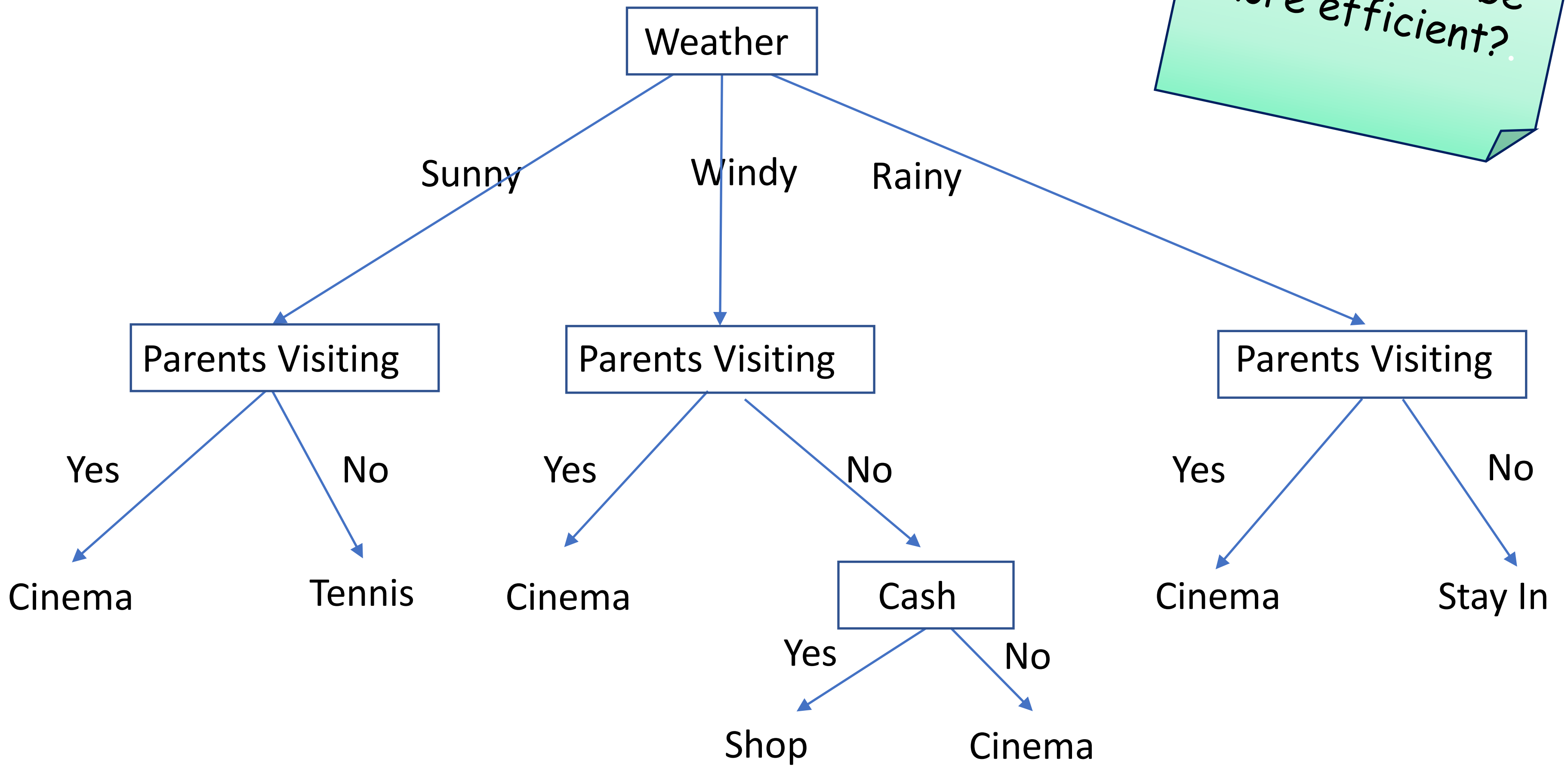


Unfortunately, it is not always this easy, especially if we have much more complex data.
More layers of questions can be added with more attributes.

Example: What should you do this weekend?

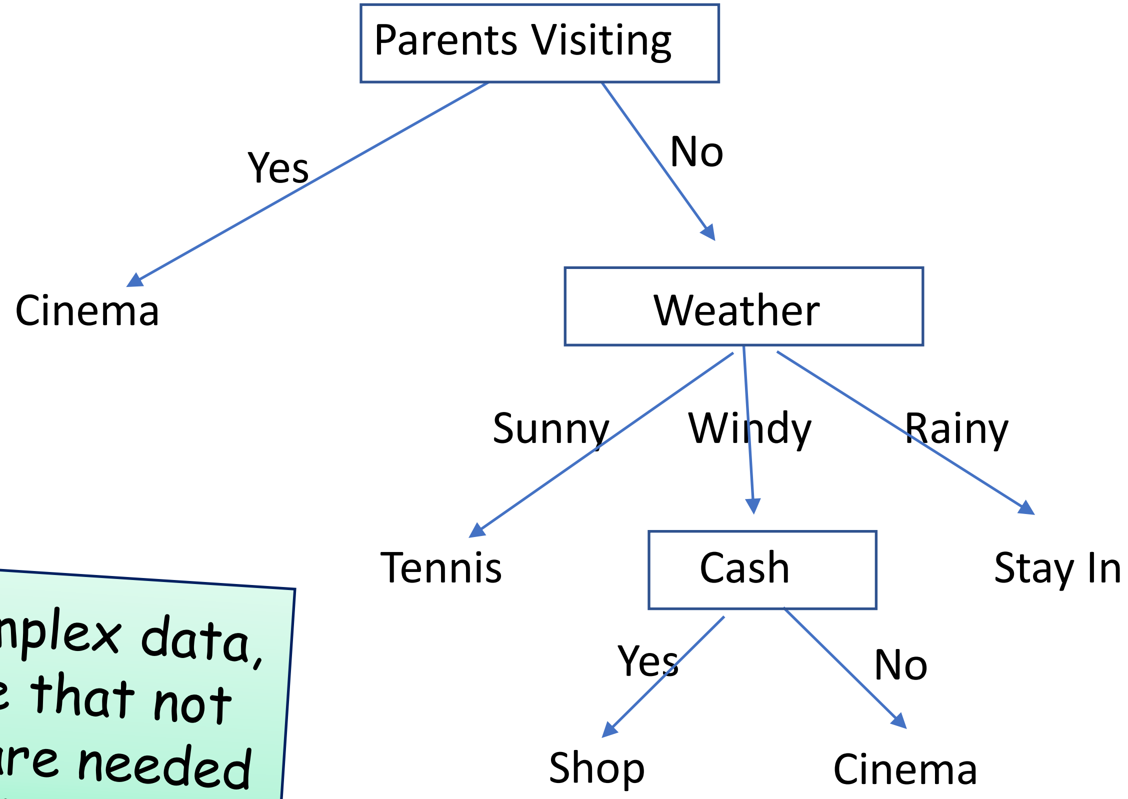
Weather	Parents Visiting	Have extra cash	Weekend Activity
Sunny	Yes	Yes	Cinema
Sunny	No	Yes	Tennis
Windy	Yes	Yes	Cinema
Rainy	Yes	No	Cinema
Rainy	No	Yes	Stay In
Rainy	Yes	No	Cinema
Windy	No	No	Cinema
Windy	No	Yes	Shopping
Windy	Yes	Yes	Cinema
Sunny	No	Yes	Tennis

What to do this weekend?



Can the tree be more efficient?

What to do this weekend?



Also with complex data, it is possible that not all features are needed in the Decision Tree.

Decision Tree Algorithms

- There are many existing Decision Tree algorithms.
- If written correctly, the algorithm will determine the best question/test for the tree.

How do we know how accurate our decision tree is?

Decision Tree Evaluation

- A confusion matrix is often used to show how well the model matched the actual classifications.
  - The matrix is not confusing – it simply illustrates how “confused” the model is!
- It is generated based on test data.

		Predicted Classification			
		Cinema	Shop	Stay In	Tennis
Actual Classification	Cinema	95	20	2	3
	Shop	3	7	1	1
	Stay In	2	0	5	0
	Tennis	36	8	4	73

CODING A DECISION TREE

The Data

For our first example, we will be using a set of measurements taken on various red wines.

The data set is from

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

The data are located at

https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

There are 12 measurements, taken on 1599 different red wines.

Attribute Summary

Data columns (total 12 columns):

fixed acidity	1599 non-null float64
volatile acidity	1599 non-null float64
citric acid	1599 non-null float64
residual sugar	1599 non-null float64
chlorides	1599 non-null float64
free sulfur dioxide	1599 non-null float64
total sulfur dioxide	1599 non-null float64
density	1599 non-null float64
pH	1599 non-null float64
sulphates	1599 non-null float64
alcohol	1599 non-null float64
quality	1599 non-null int64

dtypes: float64(11), int64(1)
memory usage: 150.0 KB

Question: Can we predict the quality of the wine from the attributes?

Coding Decision Trees: General Steps

1. Load the decision tree packages
2. Read in the data
3. Identify the target feature
4. Divide the data into a training set and a test set.
5. Fit the decision tree model
6. Apply the model to the test data
7. Display the confusion matrix

1. Load Decision Tree Package

Python

```
from sklearn import tree
```

2. Read in the data

Python

```
import pandas as pd
data_url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"

wine = pd.read_csv(data_url, delimiter=';')

print(wine.info())
```

3. Identify the target feature

Python

```
#Split the quality column out of the data
wine_target = wine['quality']
wine_data = wine.drop('quality', axis=1)
```

For the functions that we will be using, the target values (e.g., quality) must be a separate object.

4. Divide the Data

Python

```
from sklearn import model_selection
test_size = 0.30
seed = 7
train_data, test_data, train_target, test_target =
model_selection.train_test_split(wine_data,
        wine_target, test_size=test_size,
        random_state=seed)
```

5. Fit the Decision Tree Model

Python

```
model = tree.DecisionTreeClassifier()
model = model.fit(train_data, train_target)
```

6. Apply the Model to the Test Data

Python

```
prediction = model.predict(test_data)
```

7. Display Confusion Matrix

Python

```
row_name ="Quality"
cm = pd.crosstab(test_target, prediction,
     rownames=[row_name], colnames=[''])
print(' '*(len(row_name)+3),"Predicted ", row_name)
print(cm)
```

Activity: Decision Tree Program

- Make sure that you can run the decisionTree code:

Python
`01_Decision_Tree.ipynb`

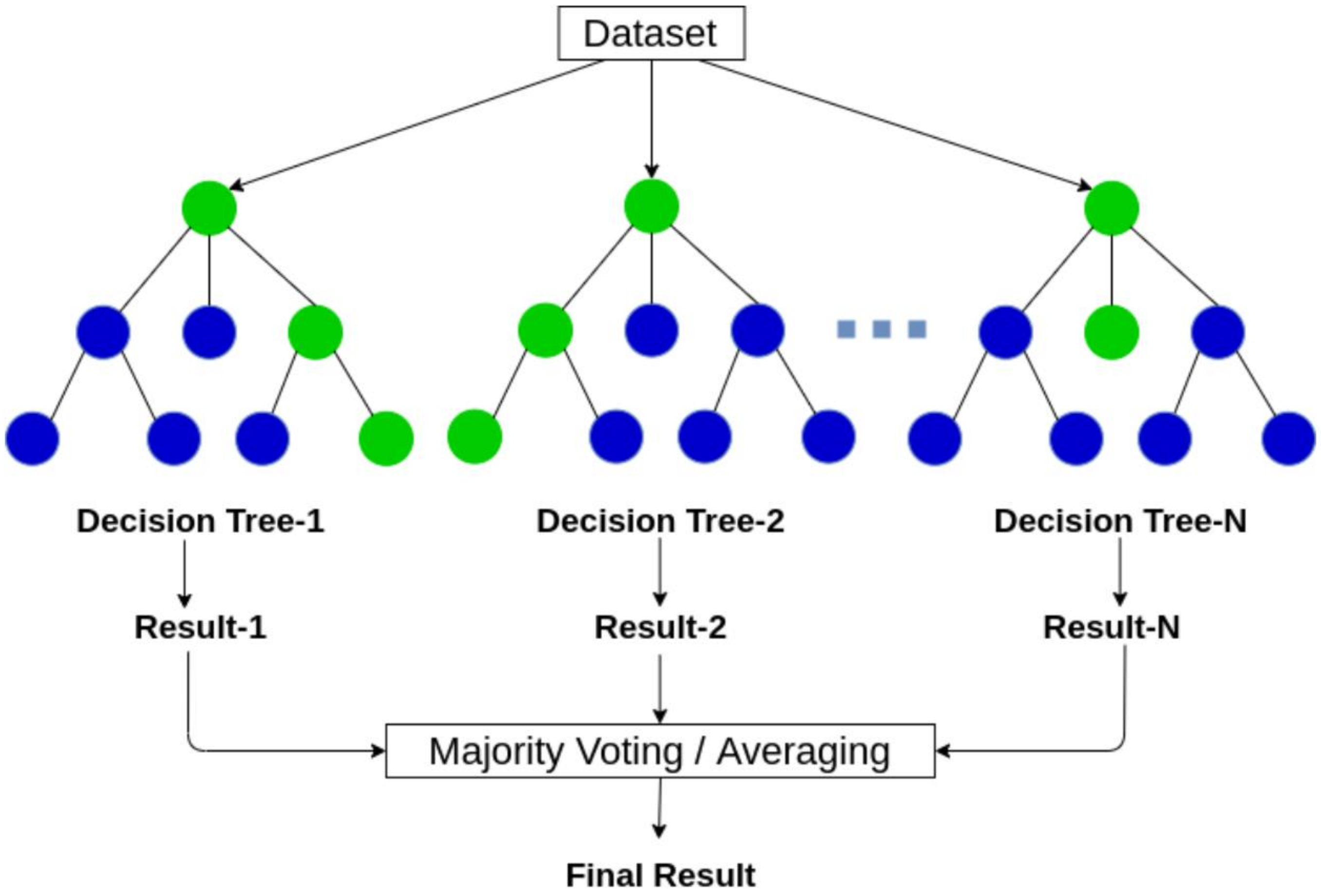
RANDOM FOREST

Random Forest: Overview

- A classification algorithm within supervised learning.

- An Ensemble Technique –
  - These techniques combine a group of “weaker” learning techniques to build a stronger technique.
  - Random Forest combines the results of multiple decision trees to create a more robust result.

Random Forest: How does it work?



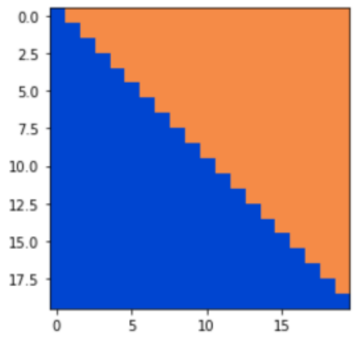
Different decision tree algorithms can produce different results.

The random forest aggregates the decisions from the trees to determine an overall solution.

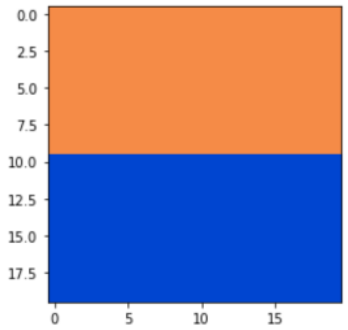
Image borrowed from https://ai-pool.com/a/s/random-forests-understanding

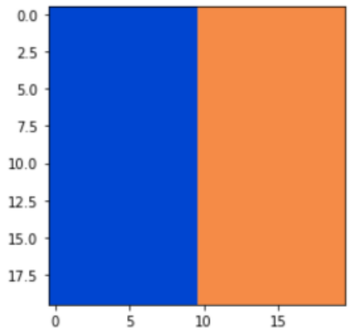
Random Forest: How does it work?

- Suppose that the data fall into one of two categories (blue or orange) depending on two values, x and y, as shown in this figure:



- A decision tree could choose a relationship between x, y, and the categories that matches one of the following figures:

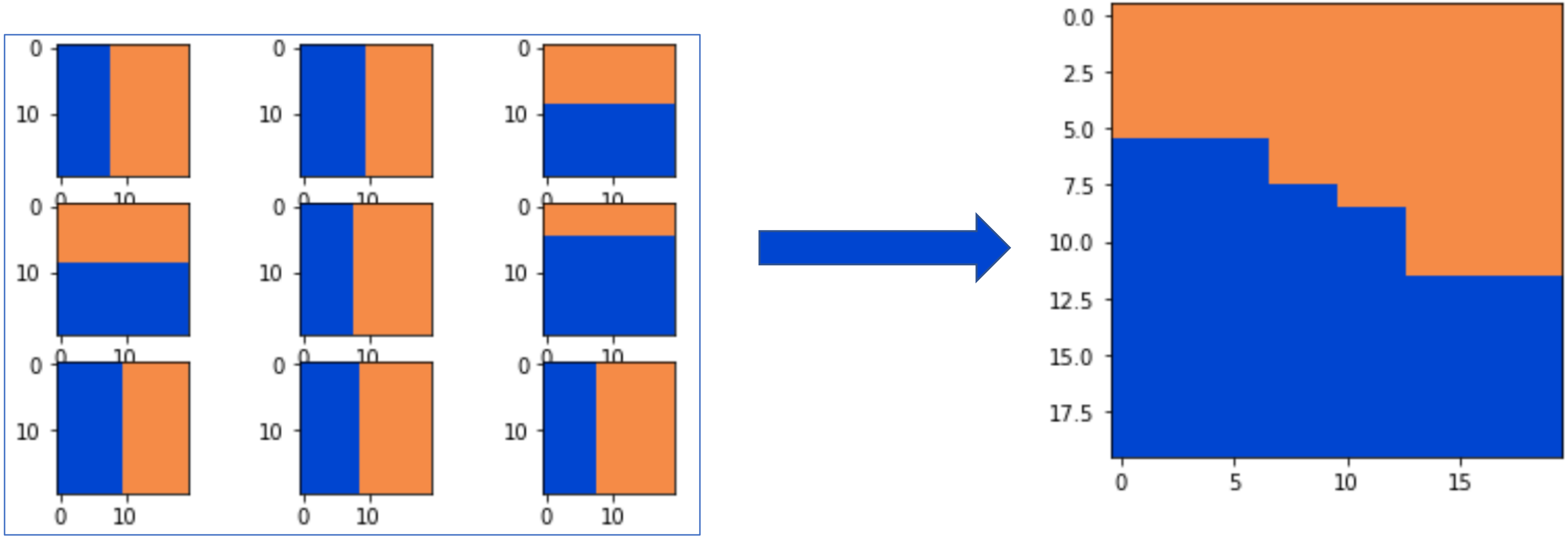




University of Virginia | Research Computing

Random Forest: How does it work?

- By combining the many, many outcomes, the random forest can approach the desired mapping.



Random Forest: How does it work?

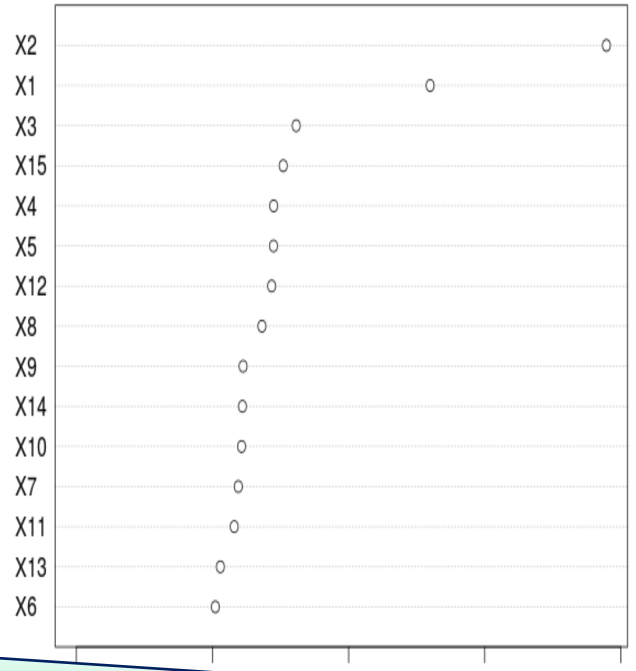
Random Forests can use different techniques for selecting features for computing each decision value.

This can lead to the choice of different features.

Random Forest: Feature Importance

- We would like to know the "importance" of the features (e.g., which features are the most important for making decisions).

- Different algorithms use various metrics to determine the importance of the features.



The value of the measurements are not as important as the order of the features.

CODING A RANDOM FOREST

The Data

For the Random Forest example, we will reuse the winequality_red data set.

Coding Random Forest: General Steps

1. Load the random forest packages
2. Read in the data
3. Identify the target feature
4. Divide the data into a training set and a test set.
   a. Choose the sample size
   b. Randomly select rows
   c. Separate the data
5. Fit the random forest model
6. Apply the model to the test data
7. Display the feature importance

1. Load Random Forest Package

Python

```
from sklearn.ensemble import RandomForestClassifier
```

Steps 2 - 4

Python

```
import pandas as pd
. . .
train_data, test_data, train_target, test_target =
model_selection.train_test_split(wine_data,
        wine_target, test_size=test_size,
        random_state=seed)
```

Repeat steps 2 – 4 from the Decision Tree example.

5. Fit the Random Forest Model

Python

```
model = RandomForestClassifier()
model.fit(train_data, train_target)
```

6. Apply the Model to the Test Data

Python

```
forest_results = model.predict(test_data)
```

7. Compute Feature Importance

Python

```
importances = model.feature_importances_
```

8. List Feature Importance

Python

```
import numpy as np

indices = np.argsort(importances)[::-1]
print("Feature ranking:")
col_names = list(train_data.columns.values)
for f in range(len(indices)):
    feature = col_names[indices[f]]
    space = ' '*(20 - len(feature))
    print("%d.\t %s %s (%f)" % \
   (f + 1, feature, space,
importances[indices[f]]))
```

Activity: Random Forest Program

- Make sure that you can run the Random Forest code:

Python
02_Random_Forest.ipynb

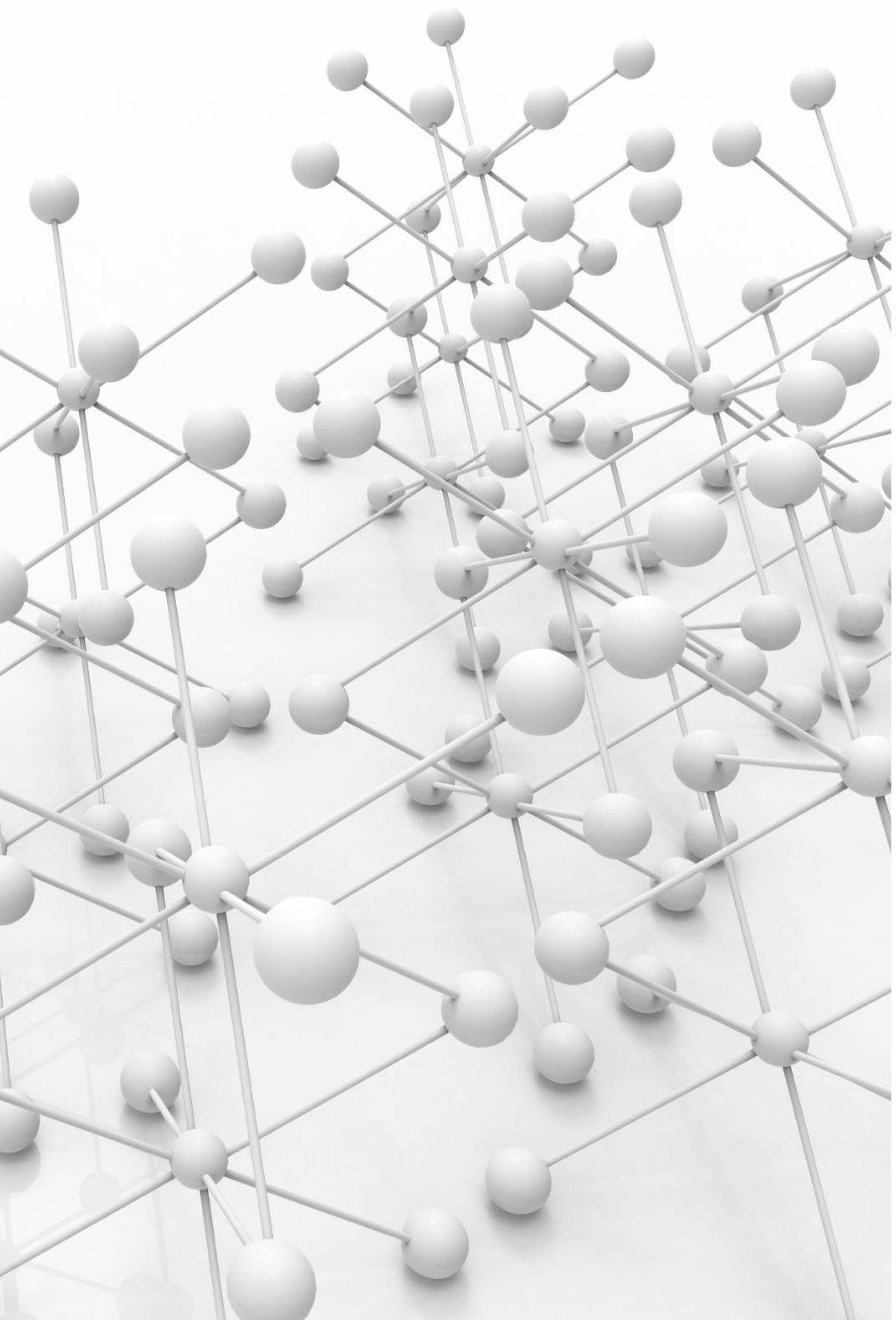
Break

We will return in 15 minutes.

NEURAL NETWORKS

Neural Network

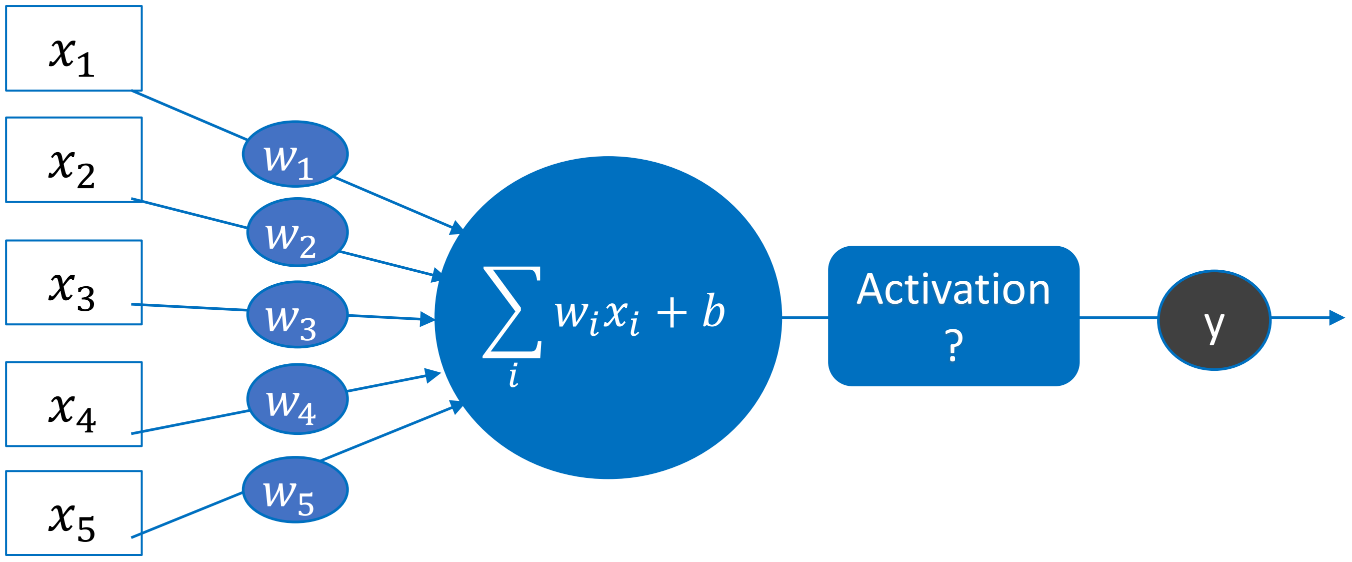
A computational model used in machine learning which is based on the biology of the human brain.



Nodes of a Neural Network

- The building blocks of a neural network are neurons, also known as nodes.
- Within each node is a very basic algebraic formula that transforms the data

Simulation of a Neuron



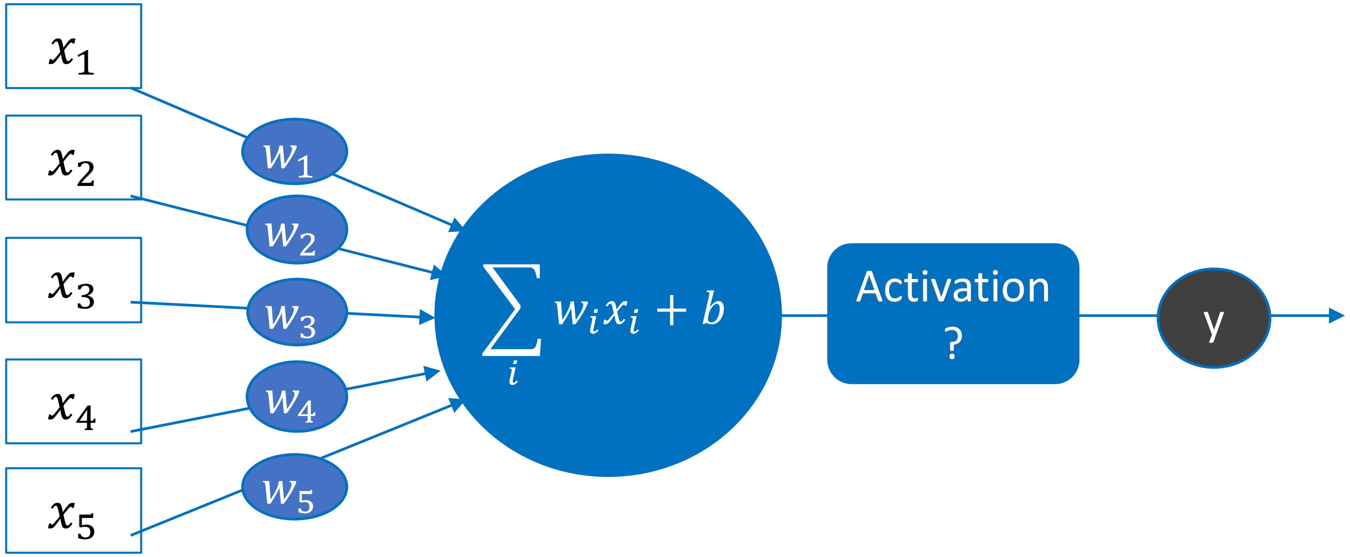
The "incoming signals" would be values from a data set.

A simple computation (like a weighted sum) is performed by the "nucleus".

Then, an "activation" function is used to determine if the output is "on" or "off".

University of Virginia | Research Computing

Simulation of a Neuron



The weights, w_i, and the bias, b, are not known at first. Random guesses are chosen.

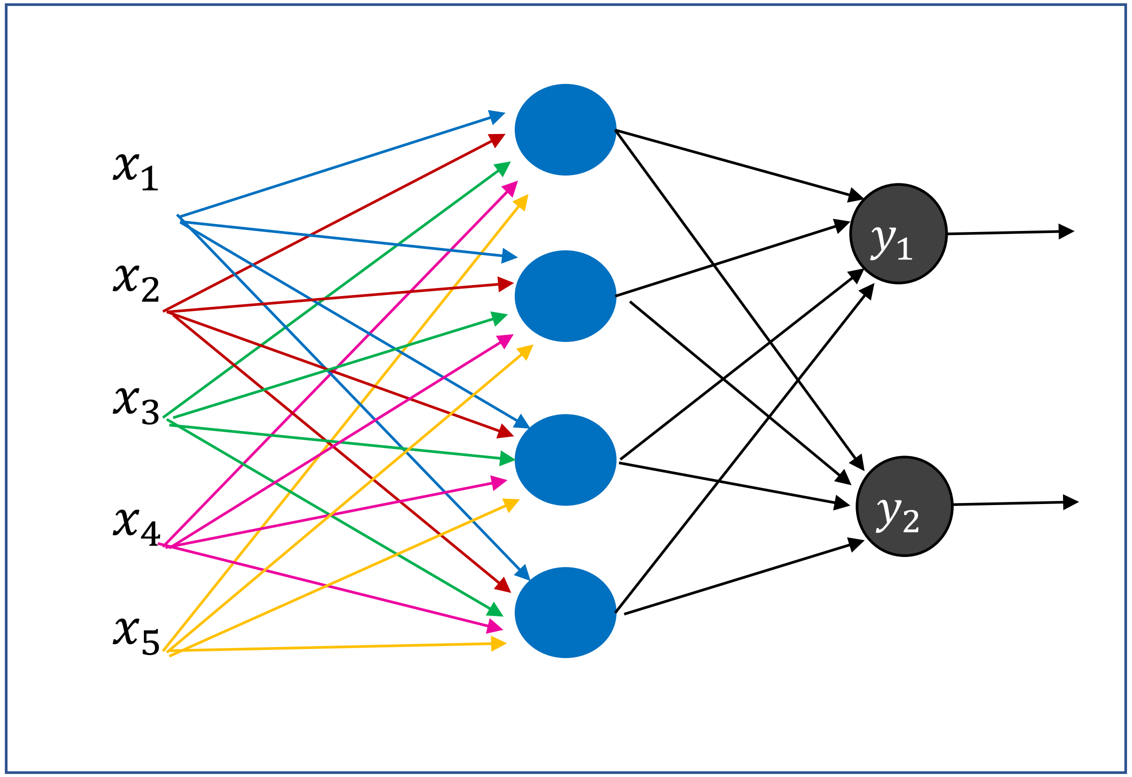
During training, the "best" set of weights are determined that will generate a value close to y for the collection of inputs x_i.

Network of Nodes

A single node does not provide much information (often times, a 0/1 value).

But, creating a network or layer of nodes will provide more information.

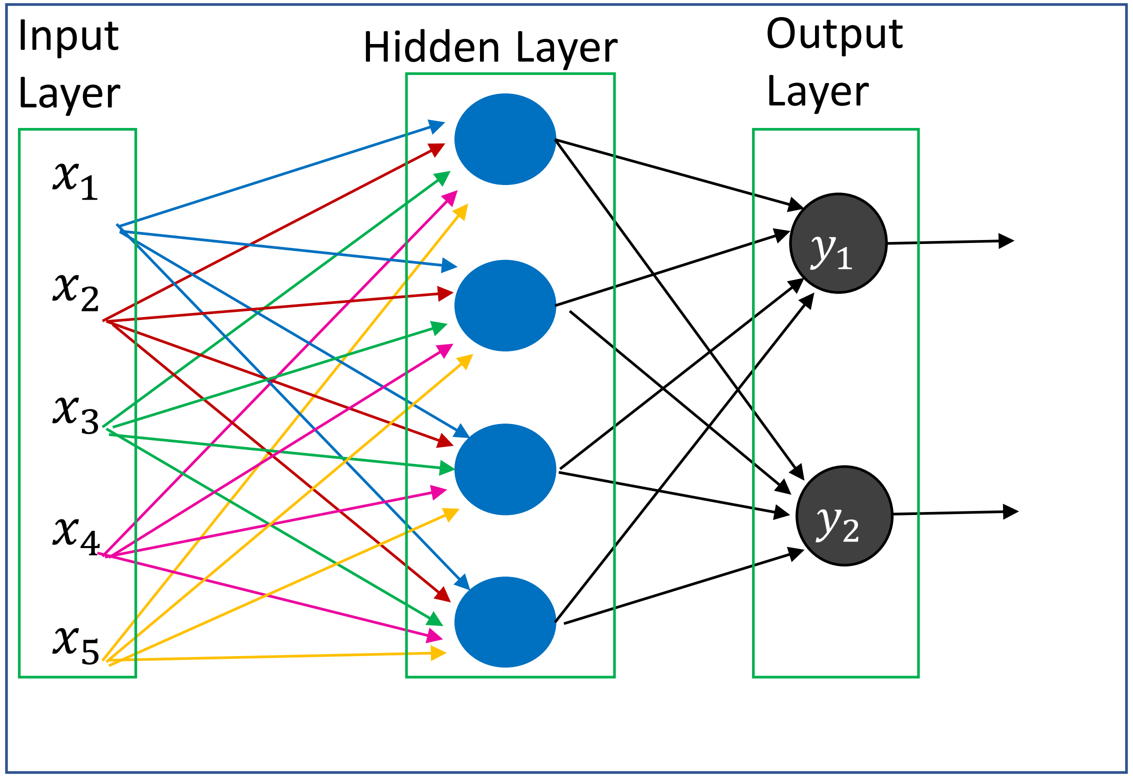
A Network of Neurons



Different computations with different weights can be performed to produce different outputs.

This is called a feedforward network – all values progress from the input to the output.

The Layers of a Network



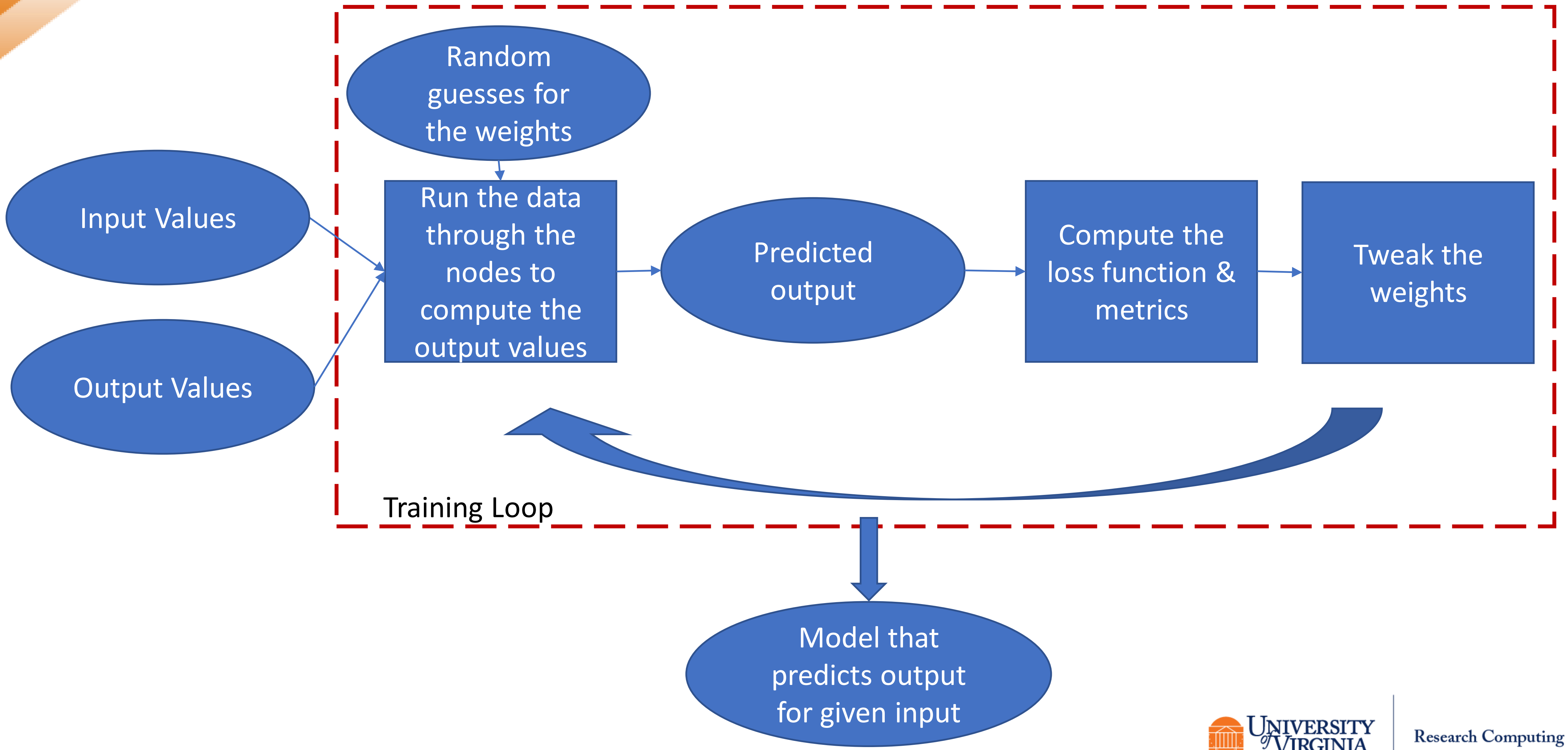
A neural network has a single hidden layer

A network with two or more hidden layers is called a “deep neural network”.

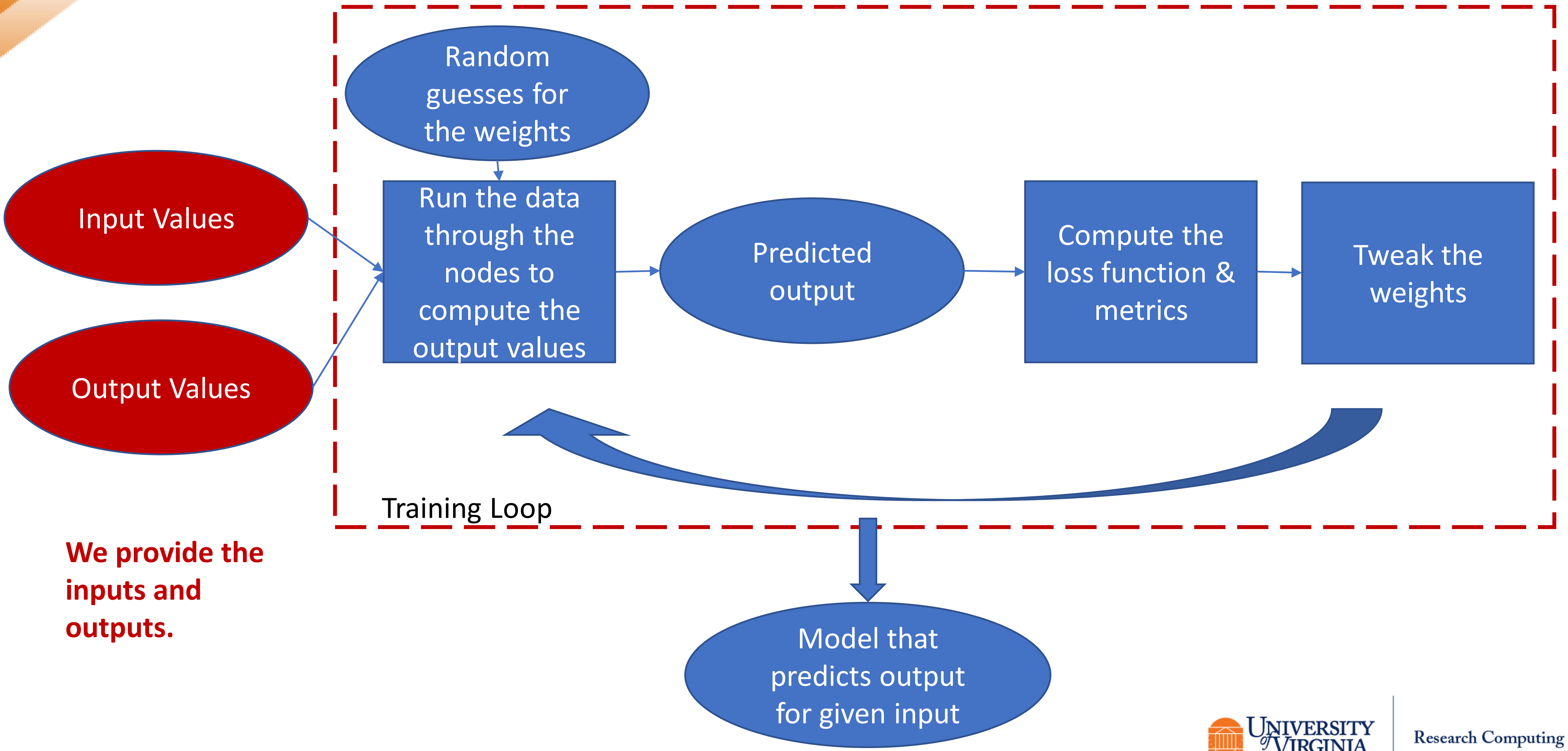
How does the machine learn?

- The output values or "predicted" values of the network can be compared with the expected results/categories/labels.
- Start with a random guess for the weights and biases.
- Another function, called a "loss" or "cost" function can be used to determine the overall error of the model.
- That error can be used to work backwards through the network and tweak the weights/biases.
- This step is called *backward propagation*.

Overview of the Learning Process

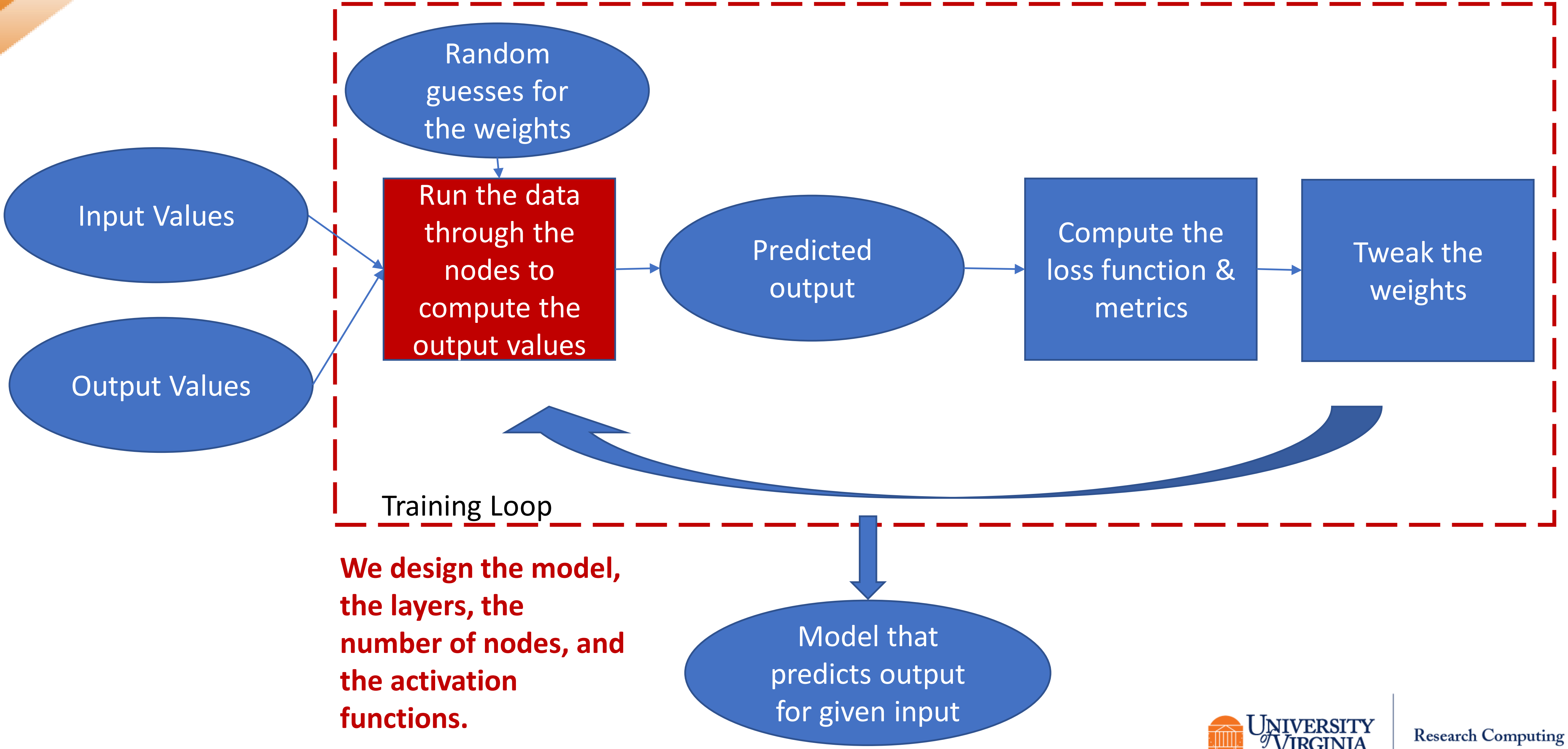


Overview of Learning Process



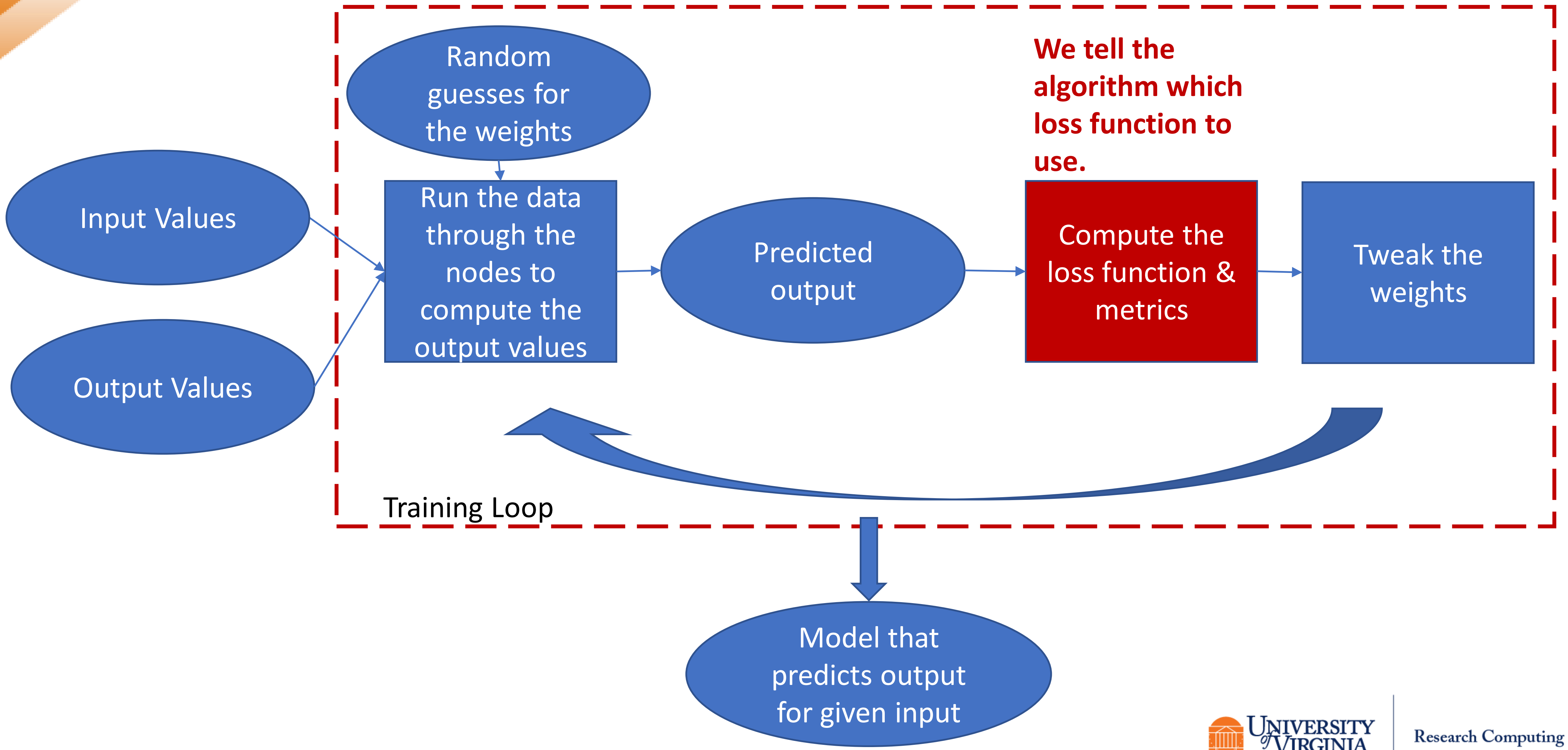
We provide the inputs and outputs.

Overview of Learning Process

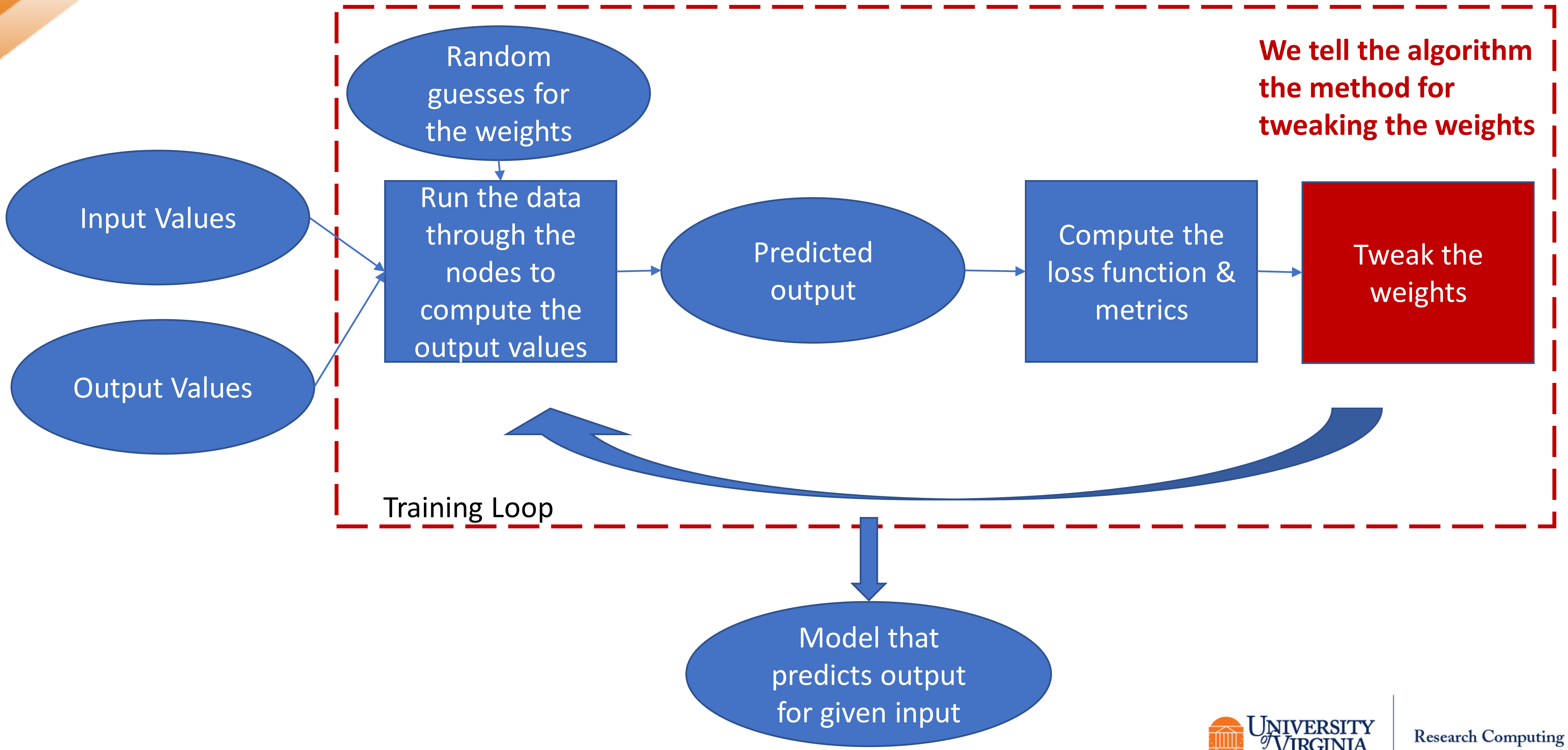


We design the model, the layers, the number of nodes, and the activation functions.

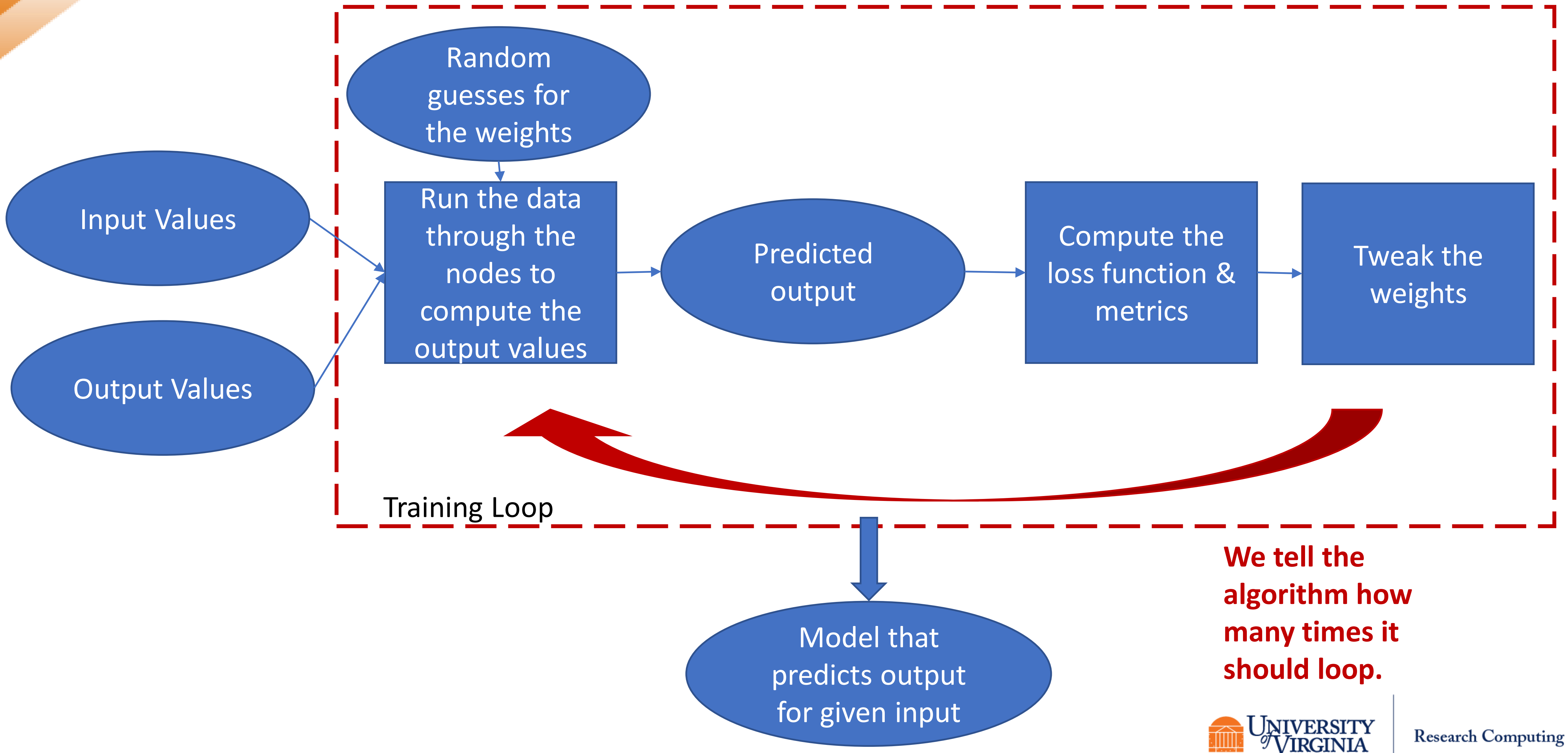
Overview of Learning Process



Overview of Learning Process



Overview of Learning Process



We tell the algorithm how many times it should loop.

Activation Function

- A function that will be determine if a node should “fire”.
- Examples include relu, sigmoid, and softmax.
- A complete list is available at https://keras.io/api/layers/activations/ .

Loss Function

- A function that will be optimized to improve the performance of the model.

- Examples include BinaryCrossEntropy and CategoricalCrossEntropy.

- A complete list is available at https://keras.io/api/losses/ .

Metrics

- A formula for measuring the accuracy of the model.
- Examples include Accuracy and MeanSquaredError.
- A complete list is available at https://keras.io/api/metrics.

Optimizer functions

- The function for tweaking the weights.
- Examples include SGD, Adam, and RMSprop.
- A complete list is available at https://keras.io/api/optimizers/ .

Epochs & Batch Size

- Epochs: Number of loops – how many times the forward/backward process should be performed.

- Batch Size: Within an epoch, the training data are divided into small batches and sent through the network. All training data are processed in an epoch.

NEURAL NETWORK EXAMPLE

Coding a Neural Network

Example: Breast Cancer Data

- The Cancer data set originally was captured at UCI Repository (https://archive.ics.uci.edu/ml/datasets.html)
- Look at the data, so that you understand what is in each file.

Filename	Brief Description
cancer_data.csv	The table of measurements cancer_DESCR.csv – an overview of the data
cancer_feature_names.csv	The names of the columns in cancer_data.csv
cancer_target.csv	The classification (0 or 1) of each row in cancer_data.csv
cancer_target_names.csv	The names of the classifications (malignant or benign)

Coding a Neural Network: General Steps

1. Load the neural network packages
2. Read in the data
3. Divide the data into a training set and a test set.
4. Preprocess the data
5. Design the Network Model
6. Train the model
7. Apply the model to the test data
8. Display the results

1. Load Neural Networks Package

Python

```
from tensorflow import keras
from tensorflow.keras import layers
```

2. Read in the Data

Python

```
import numpy as np

data_file = 'Data/cancer_data.csv'
target_file = 'Data/cancer_target.csv’

cancer_data=np.loadtxt(data_file,dtype=float,
delimiter=',')
cancer_target=np.loadtxt(target_file,
dtype=float, delimiter=',')
```

3. Divide Data

Python

```
from sklearn import model_selection

test_size = 0.30
seed = 7

train_data, test_data, train_target, test_target =
model_selection.train_test_split(cancer_data,
        cancer_target, test_size=test_size,
        random_state=seed)
```

4. Preprocess the Data

Python

```
# Convert the classes to 'one-hot' vector
from keras.utils import to_categorical

y_train = to_categorical(train_target,
num_classes=2)
y_test = to_categorical(test_target,
num_classes=2)
```

5. Design the Network Model

Python

```
def define_model():
    model = keras.Sequential([
        layers.Dense(30, activation="relu"),
        layers.Dense(2, activation="softmax")
    ])
    model.compile(optimizer="rmsprop",
                  loss="binary_crossentropy",
                  metrics=["accuracy"]
    )
    return(model)

model = define_model()
```

6. Train the Model

Python

```
num_epochs = 10
batch_size = 32

model.fit(train_data, y_train,
epochs=num_epochs, batch_size=batch_size)
```

7. Apply the Model to the Test Data

Python

```
predictions =
np.argmax(model.predict(test_data), axis=-1)
```

8. Display the Results

Python

```
score = model.evaluate(test_data, y_test)
print('\nAccuracy:  %.3f' % score[1])

from sklearn.metrics import confusion_matrix
print(confusion_matrix(test_target, predictions))
```

Activity: Neural Network Program

- Make sure that you can run the Neural Network codes:

Python
`03_Neural_Network.ipynb`

TENSOR FLOW

What is TensorFlow?

An example of deep learning; a neural network that has many layers.

A software library, developed by the Google Brain Team

Deep Learning Neural Network

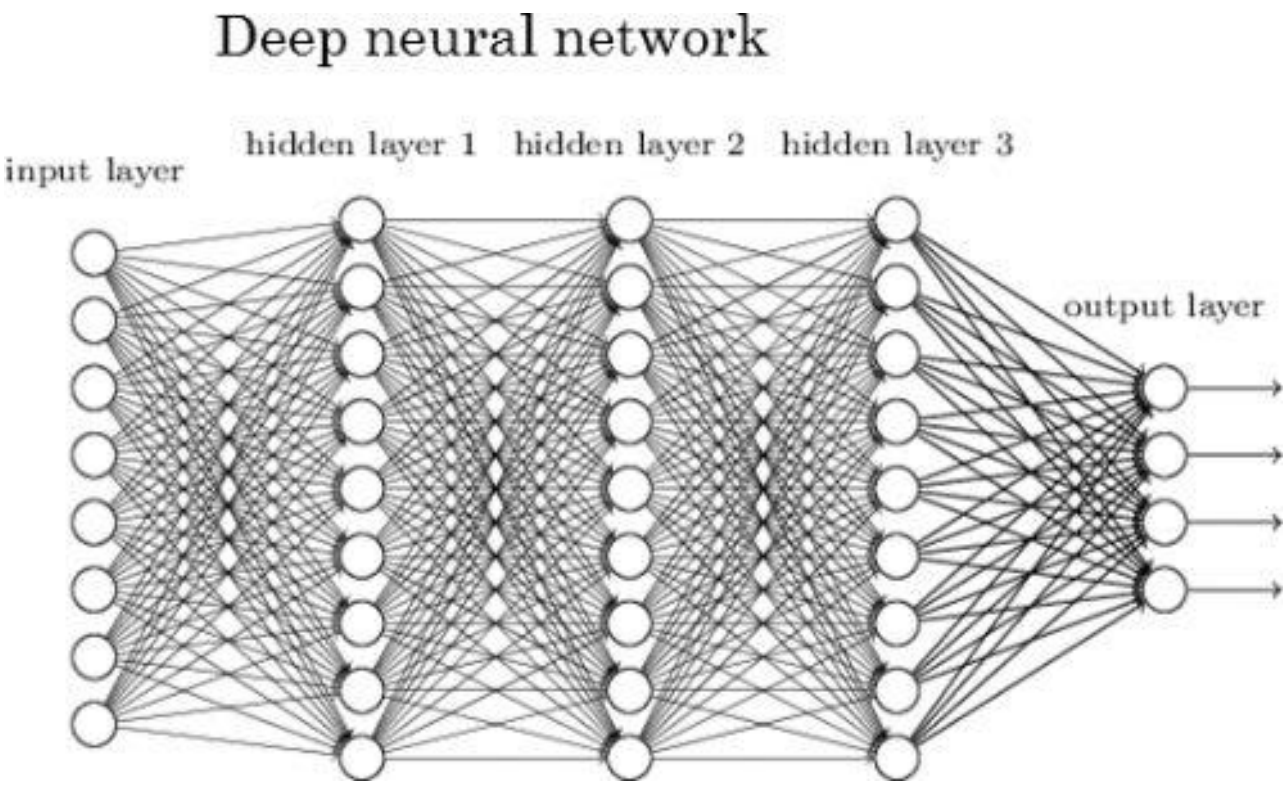
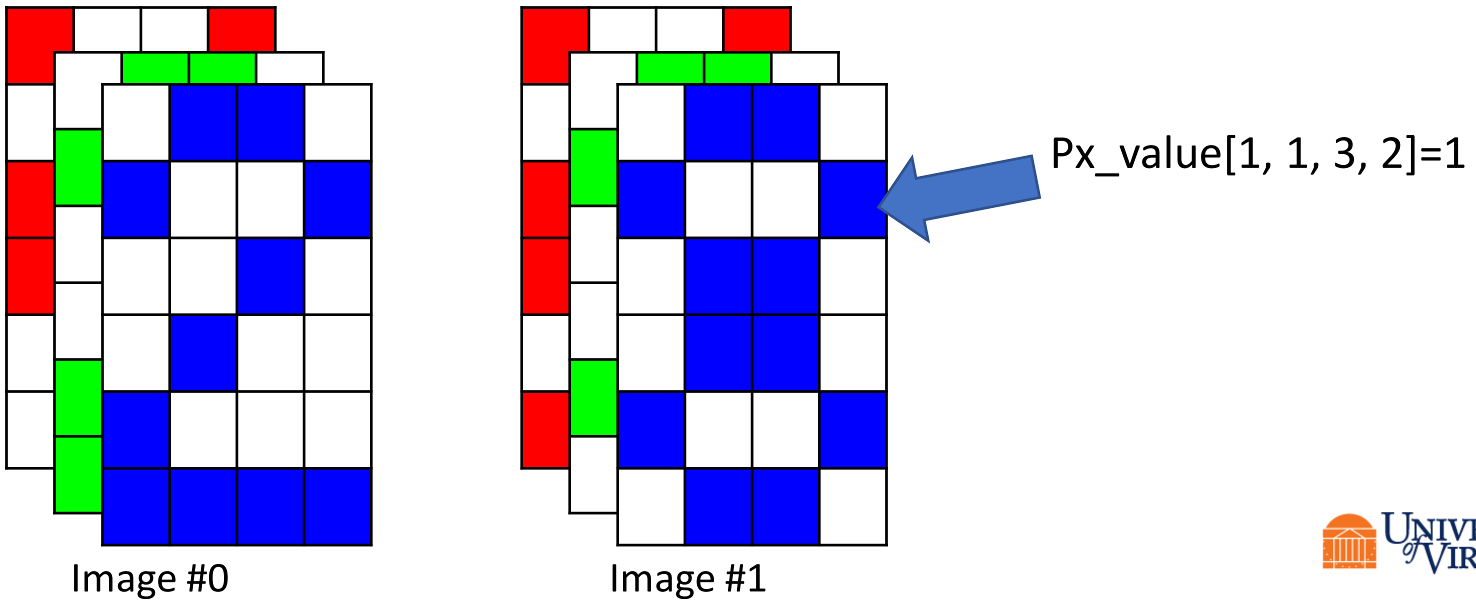


Image borrowed from:
http://www.kdnuggets.com/2017/05/deep-learning-big-deal.html

Terminology: Tensors

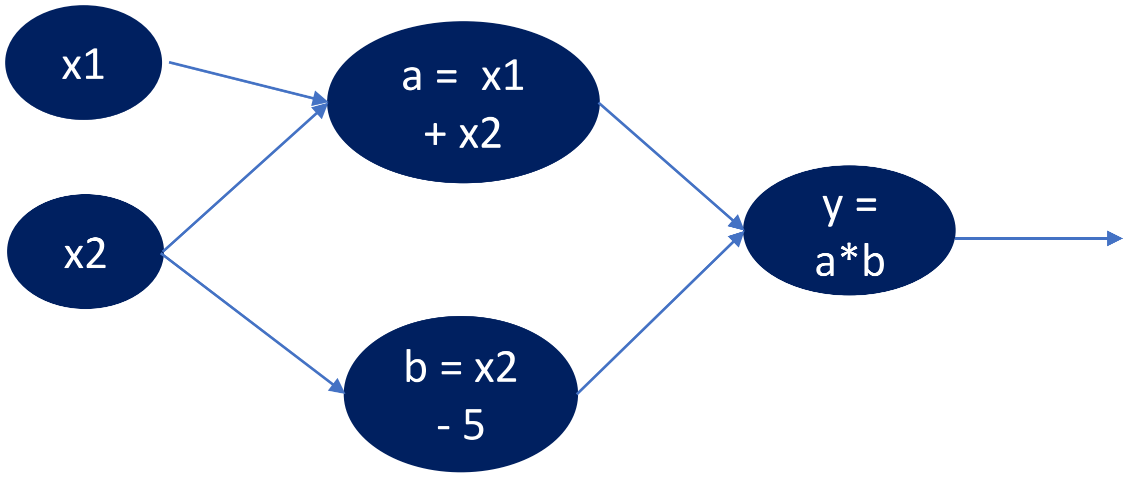
Tensor: A multi-dimensional array

Example: A sequence of images can be represented as a 4-D array: [image_num, row, col, color_channel]



Terminology: Computational Graphs

- Computational graphs help to break down computations.
  - For example, the graph for y=(x1+x2)*(x2 - 5) is



The beauty of computational graphs is that they show where computations can be done in parallel.

The need for GPUs

- With deep learning models, you can have hundreds of thousands of computational graphs.
- A GPU has the ability to perform a thousand or more of the computational graphs simultaneously. This will speed up your program significantly.

- Note: Most algorithms can run without GPUs, but they will be slower.

CODING A TENSOR FLOW

Coding Tensor Flow: General Steps

1. Load the neural network packages
2. Read in the data
3. Divide the data into a training set and a test set.
4. Preprocess the data
5. Design the Network Model
6. Train the model
7. Apply the model to the test data
8. Display the results

1. Load Keras Packages

Python

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.utils import to_categorical
```

2. Read in the Data

Python

```
import numpy as np
data_file = 'Data/cancer_data.csv'
target_file = 'Data/cancer_target.csv'
cancer_data=np.loadtxt(data_file,dtype=float,
delimiter=',')
cancer_target=np.loadtxt(target_file,
dtype=float, delimiter=',')
```

3. Split the Data

Python

```
from sklearn import model_selection
test_size = 0.30
seed = 7
train_data, test_data, train_target,
test_target =
model_selection.train_test_split(canc
er_data,cancer_target,
test_size=test_size,
random_state=seed)
```

4. Pre-process the Data

Python

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# Fit only to the training data
scaler.fit(train_data)

# Now apply the transformations to the data:
x_train = scaler.transform(train_data)
x_test = scaler.transform(test_data)

# Convert the classes to 'one-hot' vector
y_train = to_categorical(train_target,
num_classes=2)
y_test = to_categorical(test_target, num_classes=2)
```

5. Define the Model

Python

```
def define_model():
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense, Dropout
    from tensorflow.keras.optimizers import SGD
    model = keras.Sequential([
       layers.Dense(30, activation="relu"),
       layers.Dropout(0.5),
       layers.Dense(60, activation="relu"),
       layers.Dropout(0.5),
       layers.Dense(2, activation="softmax")
    ])
    model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"]
    )
    return(model)

model = define_model()
```

7. Fit the Model

Python

```
b_size = int(.8*x_train.shape[0])
num_epochs = 20

model.fit(x_train, y_train,
epochs=num_epochs, batch_size=b_size)
```

8. Apply the Model to Test Data

Python

```
predictions =
np.argmax(model.predict(x_test), axis=-1)
```

9. Evaluate the Results

Python

```
score = model.evaluate(x_test, y_test, 
batch_size=b_size)

print('\nAccuracy:  %.3f' % score[1])
from sklearn.metrics import confusion_matrix
print(confusion_matrix(test_target, 
predictions))
```

Activity: TensorFlow Program

- Make sure that you can run the TensorFlow code:

Python
`04_TensorFlow.ipynb`

PYTORCH

What is PyTorch?

Another interface for example of TensorFlow.

A software library, developed by Facebook and maintained by Mega AI.

Overview of PyTorch

- Because PyTorch uses Tensorflow as the underlying code, many of the required functions (e.g., activation, loss, optimizer) will be the same.

Activation Function

- A function that will be determine if a node should “fire”.
- Examples include nn.ReLU, nn.Sigmoid, and nn.Softmax.
- A complete list is available at
  https://pytorch.org/docs/stable/nn.html#non-linear-activations-weighted-sum-nonlinearity
  and
  https://pytorch.org/docs/stable/nn.html#non-linear-activations-other
-

Loss Function

- A function that will be optimized to improve the performance of the model.

- Examples include nn.BCELoss (Binary CrossEntropy) and nn.CrossEntropyLoss.

- A complete list is available at https://pytorch.org/docs/stable/nn.html#loss-functions

Optimizer functions

- The function for tweaking the weights.
- Examples include SGD, Adam, and RMSprop.
- A complete list is available at https://pytorch.org/docs/stable/optim.html?highlight=optimizer#torch.optim.Optimizer

CODING A PYTORCH EXAMPLE

Coding Tensor Flow: General Steps

1. Import the torch package
2. Read in the data
3. Preprocess the data

    3a. Scale the data

    3b. Split the data

    3c. Convert data to tensors

    3d. Load the tensors

4. Design the Network Model
5. Define the Learning Process
6. Train the model
7. Apply the model to the test data
8. Display the results

1. Import torch Package

Python

```
import torch

if torch.cuda.is_available():
    device_type = "cuda:" +
                  str(torch.cuda.current_device())

else:
    device_type = "cpu"

device = torch.device(device_type)
```

2. Read in the Data

Python

```
import numpy as np

data_file = 'Data/cancer_data.csv'
target_file = 'Data/cancer_target.csv'
x=np.loadtxt(data_file,dtype=float,delimiter=',')
y=np.loadtxt(target_file, dtype=float,
delimiter=',')

print("shape of x: {}\nshape of y:
{}".format(x.shape,y.shape))
```

3a. Scale the data

Python

```
#feature scaling
from sklearn.preprocessing import
StandardScaler
sc = StandardScaler()
x = sc.fit_transform(x)
```

3b. Split the Data

Python

```
from sklearn import model_selection
test_size = 0.30
seed = 7

train_data, test_data, train_target,
test_target =
model_selection.train_test_split(x,
y, test_size=test_size,
random_state=seed)
```

3c. Convert data to tensors

Python

```
#defining dataset class
from torch.utils.data import Dataset

class dataset(Dataset):
  def __init__(self,x,y):
    self.x = torch.tensor(x,dtype=torch.float32)
    self.y = torch.tensor(y,dtype=torch.float32)
    self.length = self.x.shape[0]

  def __getitem__(self,idx):
    return self.x[idx],self.y[idx]
  def __len__(self):
    return self.length

trainset = dataset(train_data,train_target)
```

3d. Load the tensors

Python

```
#DataLoader
from torch.utils.data import DataLoader

trainloader =
DataLoader(trainset,batch_size=64,shuffle=False)
```

4. Design the Network Model

Python

```
from torch import nn

class Net(nn.Module):
  def __init__(self,input_shape):
    super(Net,self).__init__()
    self.fc1 = nn.Linear(input_shape,32)
    self.fc2 = nn.Linear(32,64)
    self.fc3 = nn.Linear(64,1)

  def forward(self,x):
    x = torch.relu(self.fc1(x))
    x = torch.relu(self.fc2(x))
    x = torch.sigmoid(self.fc3(x))
    return x

model = Net(input_shape=x.shape[1])
```

5. Define the Learning Process

Python

```
learning_rate = 0.01
epochs = 700

optimizer =
torch.optim.SGD(model.parameters(),
lr=learning_rate)

loss_fn = nn.BCELoss()
```

6. Fit the Model

Python

```
losses = []
accur = []
for i in range(epochs):
  for j,(x_train,y_train) in enumerate(trainloader):

    #calculate output
    output = model(x_train)

    #calculate loss
    loss = loss_fn(output,y_train.reshape(-1,1))

    #accuracy
    predicted = model(torch.tensor(x,dtype=torch.float32))
    acc = (predicted.reshape(-1).detach().numpy().round() == y).mean()
    #backprop
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

  if i%50 == 0:
    losses.append(loss)
    accur.append(acc)
    print("epoch {}\tloss : {}\t accuracy : {}".format(i,loss,acc))
```

7. Apply the Model to Test Data

Python

```
testset = dataset(test_data,test_target)

trainloader =
DataLoader(testset,batch_size=64,shuffle=False)

predicted =
model(torch.tensor(test_data,dtype=torch.float32))
```

8. Evaluate the Results

Python

```
acc = (predicted.reshape(-1).detach().numpy().round()
== test_target).mean()

print('\nAccuracy:  %.3f' % acc)

from sklearn.metrics import confusion_matrix
predicted = predicted.reshape(-
1).detach().numpy().round()

print(confusion_matrix(test_target, predicted))
```

Activity: PyTorch Program

- Make sure that you can run the PyTorch code:

Python
`06_PyTorch.ipynb`

Break

- We will return in 15 minutes.

MULTI-GPU EXAMPLE

Activity: Multi-GPU Program

- Before running the next notebook, you will need to create a new JupyterLab session and request 2 GPUs rather than 1.

Python
`07_Tensorflow_Parallel.ipynb`

NEED MORE HELP?

<u>Office Hours via Zoom</u>

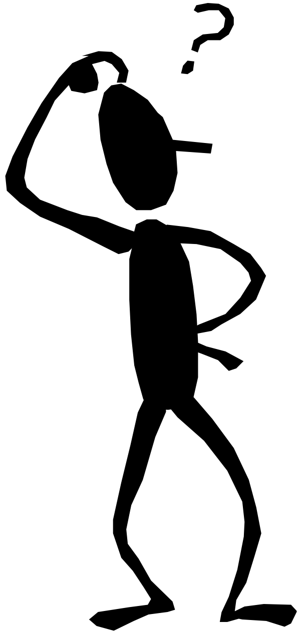
Tuesdays: 3 pm - 5 pm
Thursdays: 10 am - noon

Zoom Links are available at
https://www.rc.virginia.edu/support/#office-hours

Website:

https://rc.virginia.edu

QUESTIONS?



SUPPLEMENTAL MATERIAL

Convolutional Neural Networks

CONVOLUTIONAL NEURAL NETWORKS

What are Convolutional Neural Networks?

Originally, convolutional neural networks (CNNs) were a technique for analyzing images.

CNNs apply multiple neural networks to subsets of a whole image in order to identify parts of the image.

Applications have expanded to include analysis of text, video, and audio.

The Idea behind CNN

Recall the old joke about the blind-folded scientists trying to identify an elephant.

A CNN works in a similar way. It breaks an image down into smaller parts and tests whether these parts match known parts.

It also needs to check if specific parts are within certain proximities.

For example, the tusks are near the trunk and not near the tail.

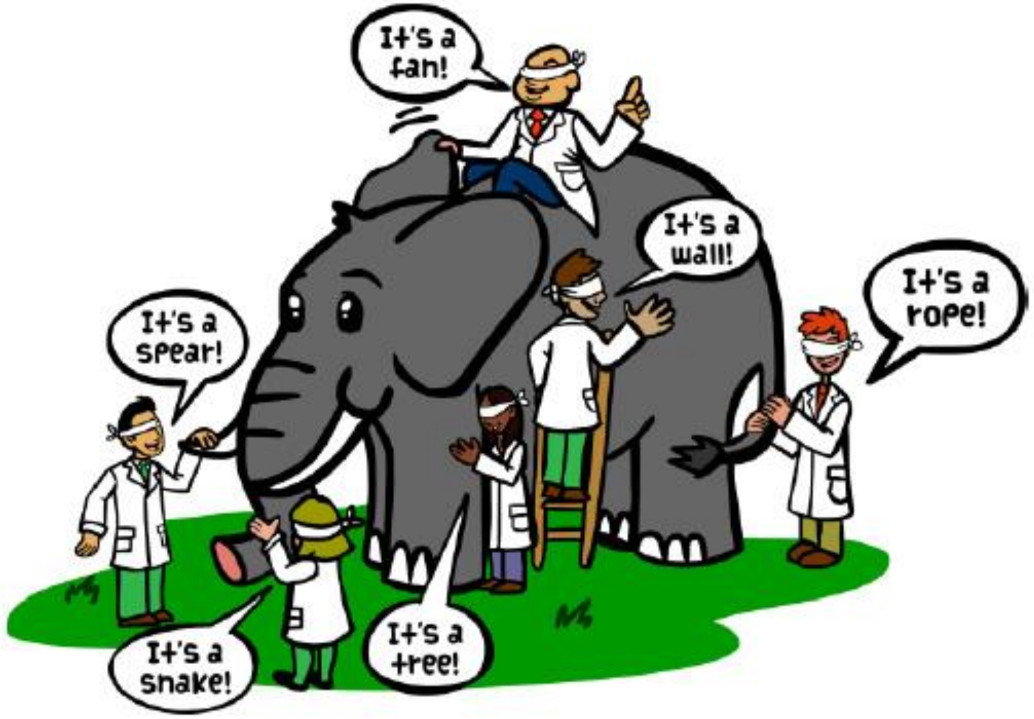
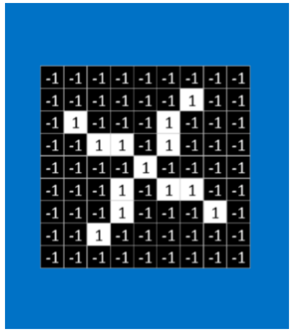
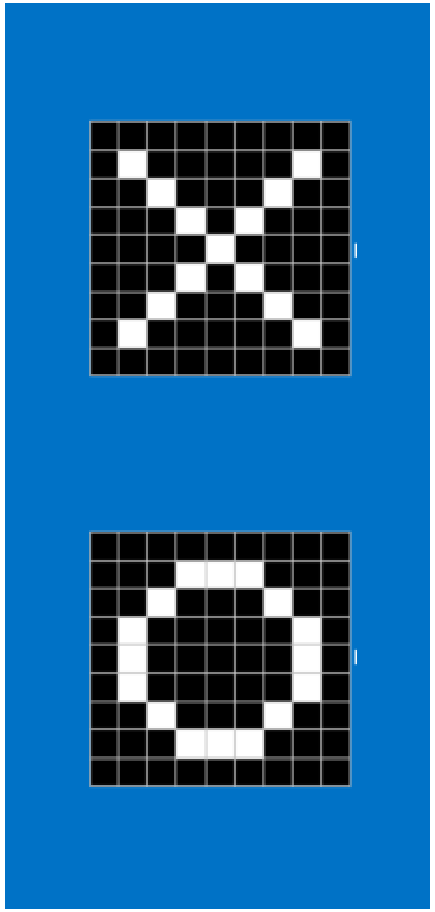


Image borrowed from https://tekrighter.wordpress.com/2014/03/13/metabolomics-elephants-and-blind-men/

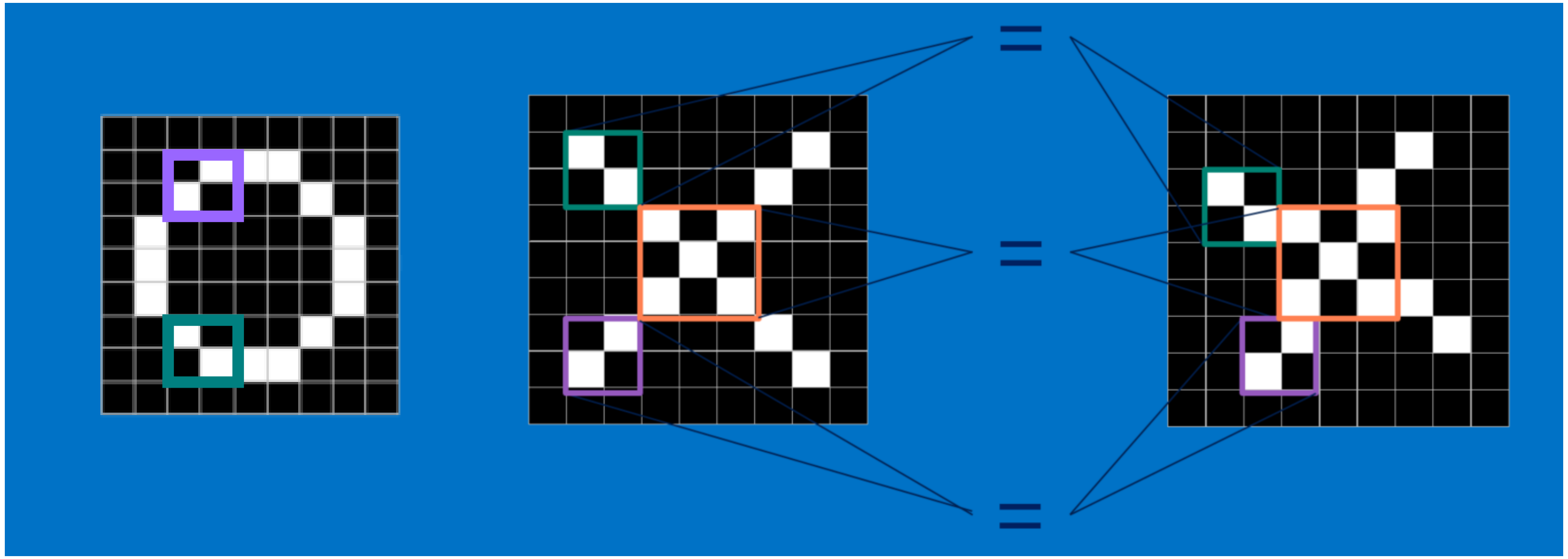
Is the image on the left most like an X or an O?





Images borrowed from
http://brohrer.github.io/how_convolutional_neural_networks_work.html

What features are in common?

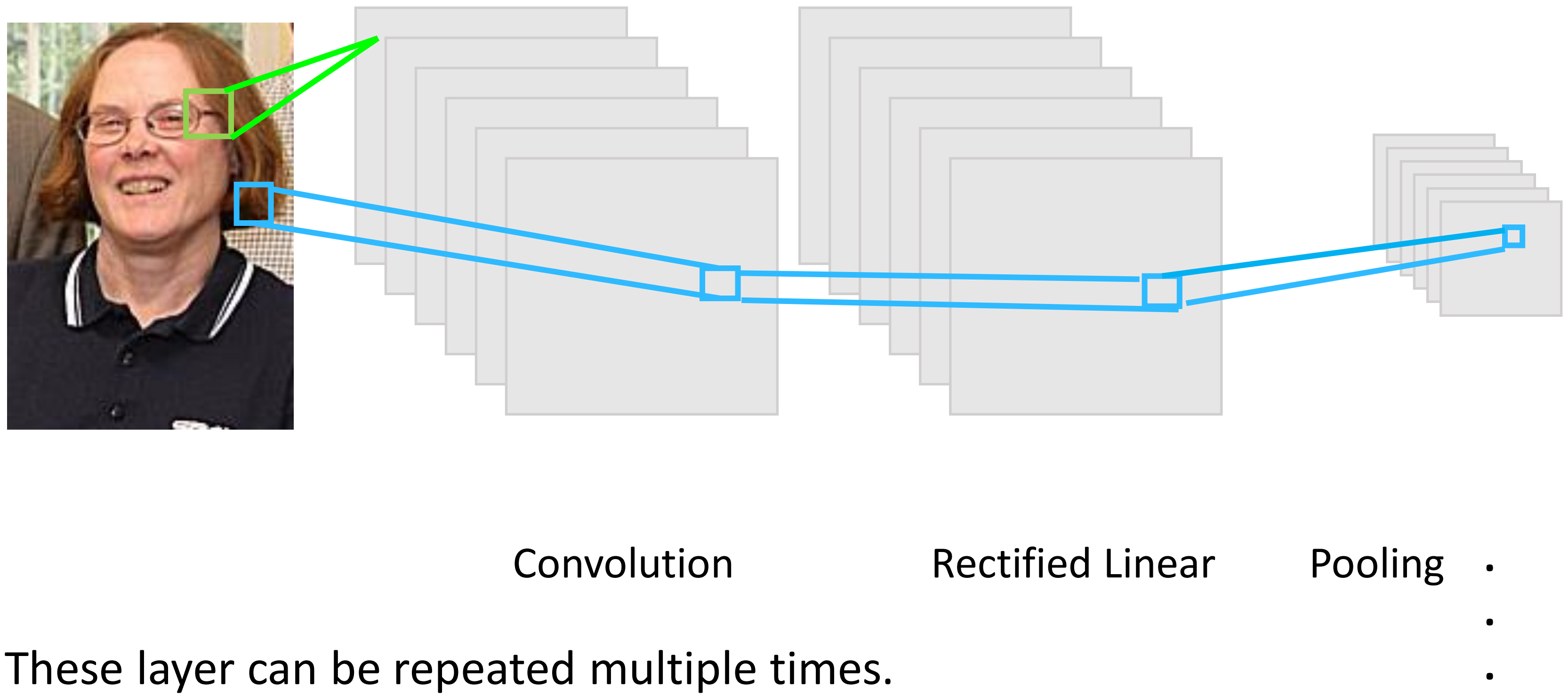


Building Blocks of CNN

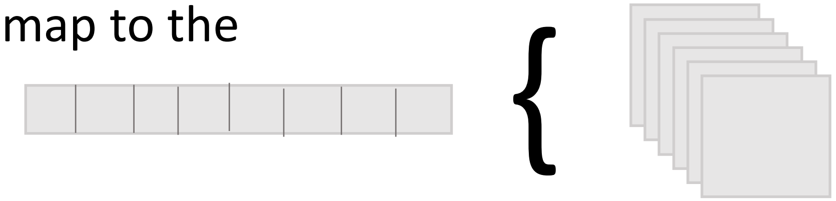
- CNN performs a combination of layers
  - Convolution Layer
    - Compares a feature with all subsets of the image
    - Creates a map showing where the comparable features occur
  - Rectified Linear Units (ReLU) Layer
    - Goes through the features maps and replaces negative values with 0
  - Pooling Layer
    - Reduces the size of the rectified feature maps by taking the maximum value of a subset

- And, ends with a final layer
  - Classification (Fully-connected layer) layer
    - Combines the specific features to determine the classification of the image

Steps



- These layer can be repeated multiple times.
- The final layer converts the final feature map to the classification.



Example: MNIST Data

- The MNIST data set is a collection of hand-written digits (e.g., 0 – 9).
- Each digit is captured as an image with 28x28 pixels.
- The data set is already partitioned into a training set (60,000 images) and a test set (10,000 images).
- The tensorflow packages have tools for reading in the MNIST datasets.
- More details on the data are available at http://yann.lecun.com/exdb/mnist/

Image borrowed from *Getting Started with TensorFlow* by Giancarlo Zaccone

Coding CNN: General Steps

1. Load the data
2. Preprocess the data.
   2a. Capture the sizes
   2b. Reshape the data
3. Design the Network Model
4. Train the model
5. Apply the model to the test data
6. Display the results

1. Load the Data

Python

```
(X_train, Y_train), (X_test, Y_test)
= mnist.load_data()
```

2a. Pre-process the Data: Capture the sizes

Python

```
numTrain = train_images.shape[0]
numTest = test_images.shape[0]

image_width = train_images.shape[1]
image_height = train_images.shape[2]
image_channels = 1
```

2b. Pre-process the Data: Reshape

Python

```
train_images = train_images.reshape((numTrain,
image_height, image_width, image_channels))
train_images = train_images.astype("float32") / 255

test_images = test_images.reshape((numTest,
image_height, image_width, image_channels))
test_images = test_images.astype("float32") / 255
```

3. Design the Network model: Part 1/4

Python

```
def define_model(image_shape):
    from tensorflow import keras
    from tensorflow.keras import layers

    image_width = image_shape[2]
    image_height = image_shape[1]
    image_channels = image_shape[3]

    #Define the input shape
    inputs = keras.Input(shape=(image_height, image_width,
image_channels))
```

3. Design the Network model: Part 2/4

Python

```
    #Define the hidden layers to be applied to the inputs & 
subsequent layers
    x = layers.Conv2D(filters=32, kernel_size=3, 
activation="relu")(inputs)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Conv2D(filters=64, kernel_size=3, 
activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Conv2D(filters=128, kernel_size=3, 
activation="relu")(x)
    x = layers.Flatten()(x)
```

3. Design the Network model: Part 3/4

Python

```
#Define the output layer
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)


model.compile(optimizer="rmsprop",
          loss="sparse_categorical_crossentropy",
          metrics=["accuracy"])


return(model)
```

3. Design the Network model: Part 4/4

Python

```
image_shape = train_images.shape
model = define_model(image_shape)
```

4. Train the model

Python

```
num_epochs = 5
batch_size = 64

model.fit(train_images, train_labels,
epochs=num_epochs,
batch_size=batch_size)
```

5. Apply Model to Test Data

Python

```
test_loss, test_acc =
model.evaluate(test_images, test_labels)
```

Display the results

Python

```
print(f"Test accuracy: {test_acc:.3f}")
```

Activity: CNN Program

- Make sure that you can run the CNN code:

Python
`05_CNN.ipynb`