INTERNSHIP REPORT

On Course from Coursera

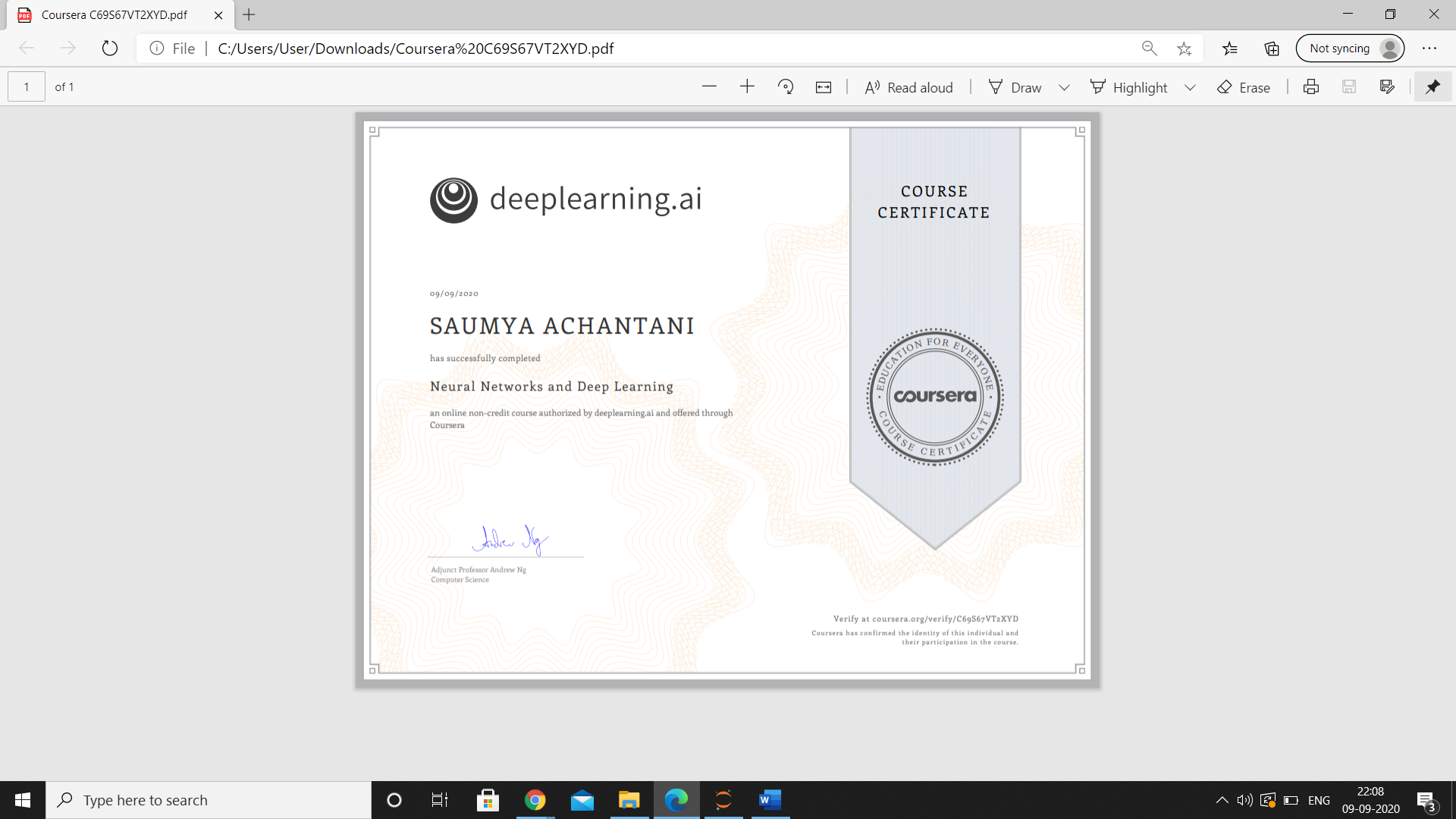
**“Neural Networks and Deep Learning”**

**SUBMITTED BY:**

**Saumya Achantani**

**18CSU194**

# **CERTIFICATE**



# **ACKNOWLEDGEMENT**

On 29 August 2020 , I completed the Neural Networks and Deep Learning course offered by deeplearning.ai on coursera.org. I would like to show my gratitude to Andrew Ng, Course Instructor, for teaching the course .I appreciate his intuitive teaching and explaining practical applications of neural networks. I would also like to express my gratitude to college faculty for providing an opportunity to do this course.

# 

# **ABSTRACT**

The part 1 of this course is based on neural networks. Simply put, a neural network is a connected graph with input neurons, output neurons, and weighted edges. Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. Using algorithms, they can recognize hidden patterns and correlations in data and – over time – continuously learn and improve. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

A neural network is a branch of machine learning called **deep learning .**Deep learning is one of many machine learning algorithms to enable a computer perform a plethora of tasks such as :

1. Some kind of prediction like stock prediction , house rent prediction or whether a person is eligible for a job or not etc.)
2. Classification ( like classifying images of dogs , cats and hotdogs or whether an employee is good or bad for the company etc.).

Deep learning uses the concept of neural networks to do its job. Just like there are neurons in the brain for passing information, there are nodes in a neural network for doing a similar task. Nodes are nothing but mathematical functions.

A neural network is of two types majorly:

1. Convolutional neural network.
2. Recurrent neural network.

All this is discussed in this course in detail.

# **INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Particulars** | **Page no.** |
|  | Introduction | **6-7** |
|  | Training objective | **8** |
|  | About the course | **9** |
|  | Instructor | **10** |
|  | Course syllabus | **11-12** |
|  | Week 1 (Outcomes and assignment) | **13-17** |
|  | Week 2 (Outcomes and assignment) | **18-28** |
|  | Week 3 (Outcomes and assignment) | **29-39** |
|  | Week 4 (Outcomes and assignment) | **40-45** |
| **10.** | Outcome on new dataset | **46** |
| **11.** | Conclusions | **47** |
| **12.** | Reference | **48** |

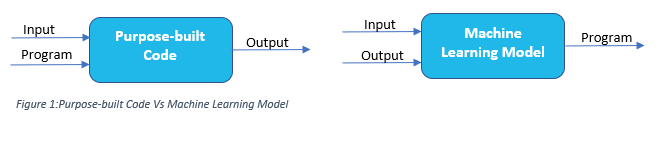
# 

# **INTRODUCTION**

John McCarthy, widely recognized as one of the godfathers of Artificial Intelligence (AI), defined AI as “the science and engineering of making intelligent machines that have the ability to achieve goals like humans do” in the year 1955. In short, Artificial Intelligence is human intelligence exhibited by Machines.

Arthur Samuel defined Machine Learning (ML) in 1959 as a large sub-field of AI dealing with the field of study that gives computers the ability to learn without being explicitly programmed. This means a single program, once created, will be able to learn how to do some intelligent activities outside the notion of programming. This contrasts with purpose-built programs whose behavior is defined by hand-crafted heuristics that explicitly and statically define their behavior. So, you can say Machine Learning is an approach to achieve Artificial Intelligence.

Fig 1. Illustrates how a machine learning model learns:



This is exactly how humans learn as well. When any kid learns to identify objects/person, we don’t tell them an algorithm/procedure to identify the features and then decide what is it. We simply show them multiple examples of that object and then our human brain automatically identifies the features (sub-consciously) and learns to identify that object. This is indeed what a Machine Learning Model does. Within the machine learning fields, there is an area often referred to as brain-inspired computation. Human brain is one of the best ‘machine’ we know for learning and solving problems. The brain-inspired technique is indeed inspired by how our human brain works. It is believed that the main computational element of our brain is neuron. The complex connected network of neurons forms the basis of all the decisions made based on the various information gathered. This is exactly what Artificial Neural Network technique does.

Within the domain of neural networks, there is an area called Deep Learning(DL), in which neural networks have more than three layers, i.e. more than one hidden layer. These neural networks used in Deep learning are called Deep Neural Networks (DNNs).So, Deep Learning is a technique for implementing Machine Learning. Thanks to Deep learning, there are many tasks that machines can now do better than humans. One such example is image classification. In 2015, the ImageNet winning entry, Reset, exceeded human-level accuracy with a top-5 error rate below 5%. Humans can classify images with error rate 5%.

Fig. 2 illustrates the relationship between Artificial Intelligence, Machine Learning and Deep Learning. If you look at it from the mathematical terms, all machine learning is AI, but not all AI is machine learning. Similarly, all deep learning is machine learning but not all machine learning is deep learning.

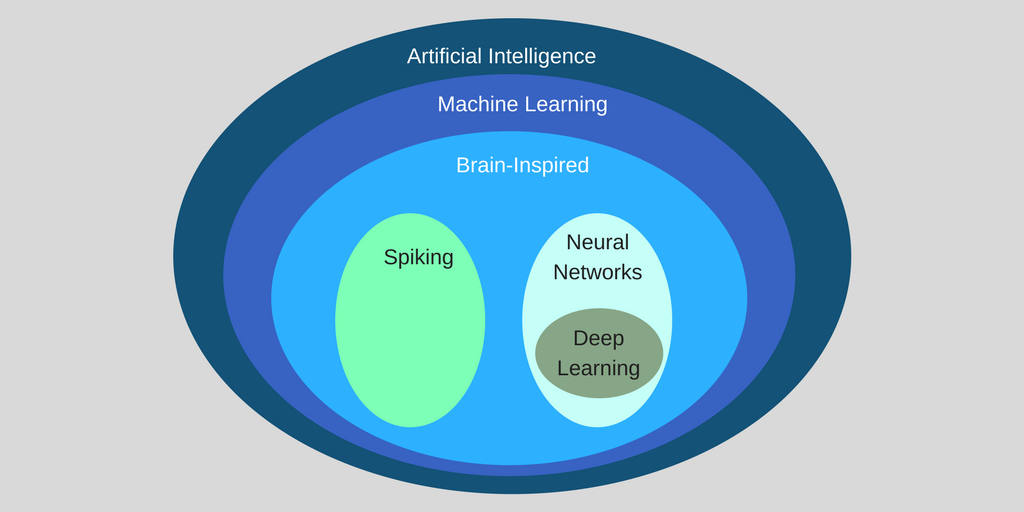


Fig 2:Relationship between Artificial Intelligence, Machine Learning and Deep Learning

To summarize:

* Artificial Intelligence is human intelligence exhibited by machines
* Machine Learning is an approach to achieve Artificial Intelligence
* Deep Learning is a technique for implementing Machine Learning

# **TRAINING OBJECTIVE**

To understand the major technology trends driving Deep Learning - Be able to build, train and apply fully connected deep neural networks - Know how to implement efficient (vectorized) neural networks - Understand the key parameters in a neural network's architecture This course also teaches you how Deep Learning actually works, rather than presenting only a cursory or surface-level description. So, after completing it, one will be able to apply deep learning to our own applications.

# **ABOUT THE COURSE**

Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions. Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.To improve the performance of a Deep Learning model the goal is to the reduce the optimization function which could be divided based on the classification and the regression problems.

In this first course, we have learn how to build a new network including a deep neural network and how to train it on data. There is a cat neem running around in deep learning. And so, we'll build a cat recognizer.

In the second week, we have learn about the Basics of Neural Network Programming.Then we have learn the structure of what we call the forward propagation and the back propagation steps of the algorithm and how to implement neural networks efficiently.

In third week we have learned the framework for neural network programming , we code up a single hidden layer neural network .Inorder to learn all the key concepts needed to implement and get to work in neural network.

Then finally in fourth week, we build a deep neural network and neural network with many layers and see it working .

# **INSTRUCTORS**

****

[**Andrew Ng**](https://www.coursera.org/instructor/andrewng)

CEO/Founder Landing AI; Co-founder, Coursera; Adjunct Professor, Stanford University; formerly Chief Scientist,Baidu and founding lead of Google Brain



[**Head Teaching Assistant - Kian Katanforoosh**](https://www.coursera.org/instructor/~24382163)

Lecturer of Computer Science at Stanford University, deeplearning.ai, Ecole CentraleSupelec



[**Teaching Assistant - Younes Bensouda Mourri**](https://www.coursera.org/instructor/younes)

Mathematical & Computational Sciences, Stanford University, deeplearning.ai

# **COURSE SYLLABUS**

The course spans over 4 weeks starting with the basics and building upon it each week. It start with the most simple 1-Layer neural network in the second week and complete an L-Layer neural network in the fourth week.

## Week 1 – Introduction to deep learning

In the first week, introductions about Neural Network and Deep Learning are discussed.

* Introduction to Neural Network
* Supervised learning with Neural Network
* Reasons for Deep Learning taking off

## Week 2 – Neural Networks Basics

The second week is, in my opinion, the foundation week for the specialization. Many of the important concepts of Deep Learning are discussed. There are also refreshers on calculus and linear algebra.

* Derivatives
* Binary Classification
* Logistic Regression
* Cost Function
* Gradient Descent
* Computation graph and Derivatives with a Computation Graph
* Vectorization and Vectorizing Logistic Regression/Gradient Output

## Week 3 – Shallow neural networks

The lectures from third week teach how to build a neural network with one hidden layer using forward propagation and back propagation.

* Two-layer Neural Network
* Neural Network Representation
* Computing a Neural Network’s Output
* Vectorizing across multiple examples
* Activation functions, need for non-linear activation functions and derivatives of activation functions
* Gradient descent for Neural Networks
* Random Initialization

## Week 4 – Deep Neural Networks

In the last week, we learn about the key computations underlying deep learning, and use them to build and train deep neural networks, and apply it to computer vision

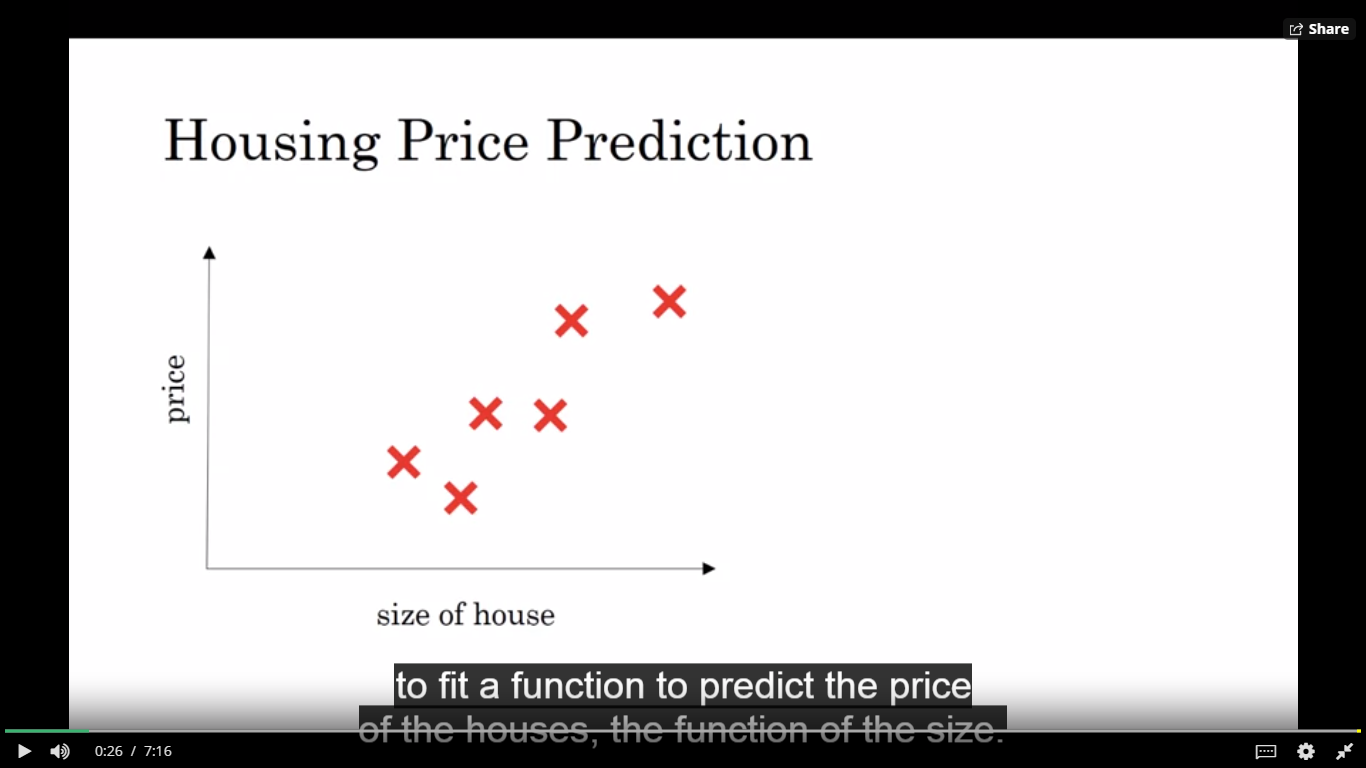
* Deep L-layer neural network
* Forward Propagation in a Deep Network
* Matrix dimensions right
* Purpose of Deep representations
* Building blocks of deep neural networks
* Forward and Backward Propagation
* Parameters vs Hyperparameters

# **Week 1:** Introduction to Deep Learning

**Key Concepts**

* Understand the major trends driving the rise of deep learning.
* Be able to explain how deep learning is applied to supervised learning.
* Understand what are the major categories of models (such as CNNs and RNNs), and when they should be applied.
* Be able to recognize the basics of when deep learning will (or will not) work well.

Neural networks were inspired by the neural architecture of a human brain, and like in a human brain the basic building block is called a Neuron. Its functionality is similar to a human neuron, i.e. it takes in some inputs and fires an output.

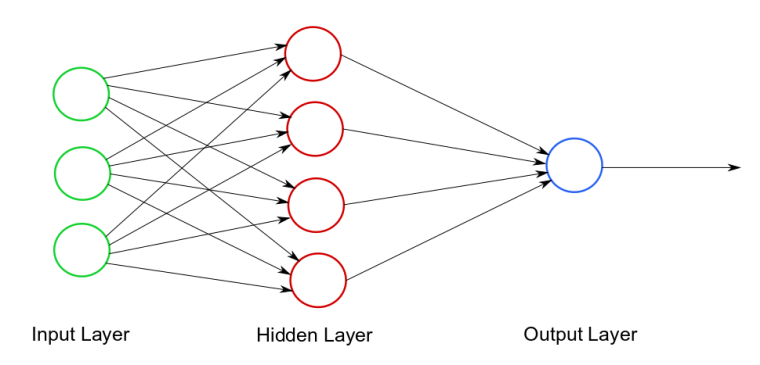


In the first video, instructor gives a very basic idea in layman terms of what a neural network is.

An example of housing price prediction is taken:

Aim -To predict the price of house given some of the inputs like size of house, number of family members ,location( pin-code) etc.

An easy understanding of neural network from this would be that we are provided with input known as features , followed by hidden layer where processing(calculation) is done. Job of the neural network is to predict the output i.e. price of house.



In the second video, difference between structured and unstructured data is explained:-

|  |  |
| --- | --- |
| Structured Data | Unstructured Data |
| 1.Follows a proper format, is stored in relational databases | 1.Does not follow a proper format(structure) and hence cannot be stored in relational database. |
| 2. **Structured data** include names, dates, | 2. **Unstructured data** includes video, audio or image files. |
| 3.20% of Enterprise data. | 3. 80% of enterprise data(according to Gartner report) |

Due to rise in neural networks, analyzing and interpreting unstructured data has become easier. The ability to process large numbers of features makes deep learning very powerful when dealing with unstructured data.

Many applications such as image processing, speech recognition make use of unstructured data.

Various applications of supervised learning with neural networks are provided such as:-

1. Online advertising

2. Photo Tagging

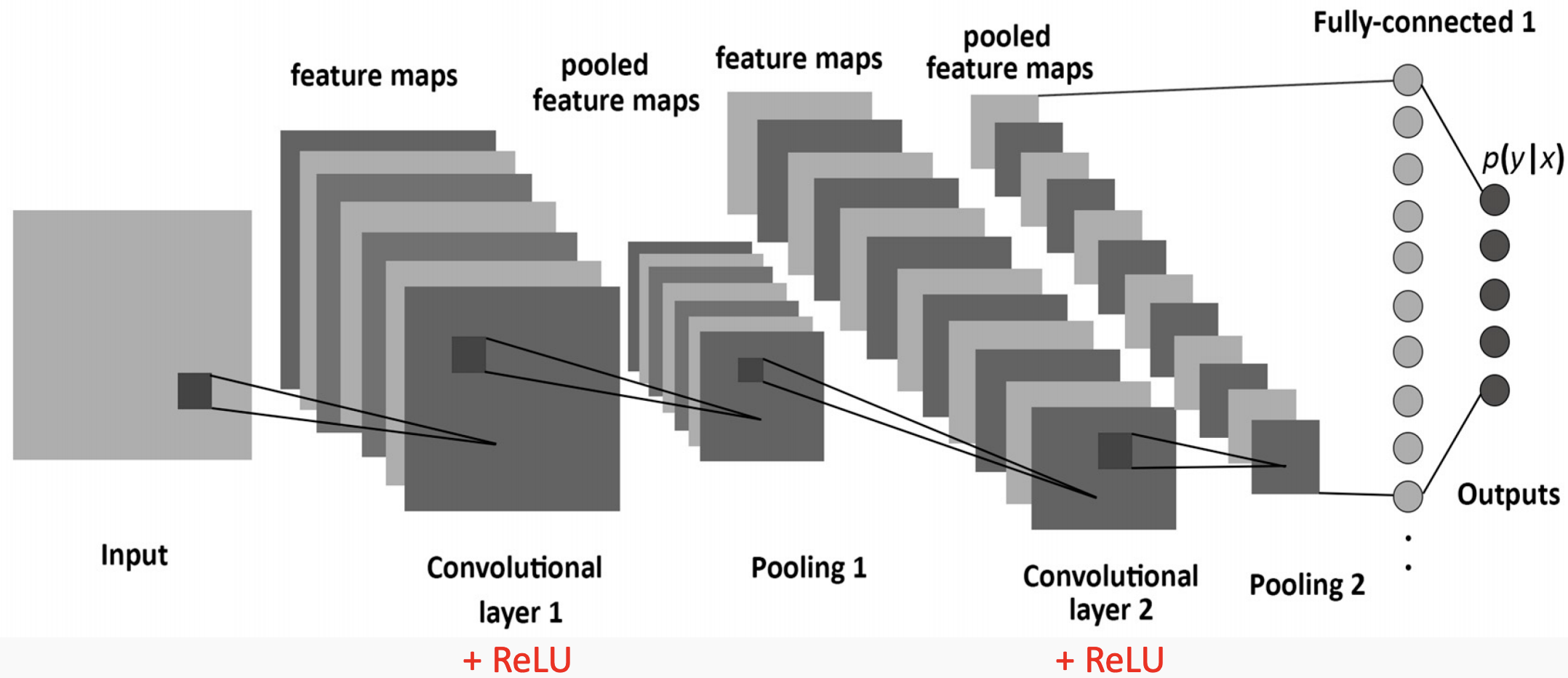
3. Speech Recognition

4. Machine Translation

Convolutional Neural Network:-

Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

It is used mainly for image processing, classification, segmentation and photo tagging.

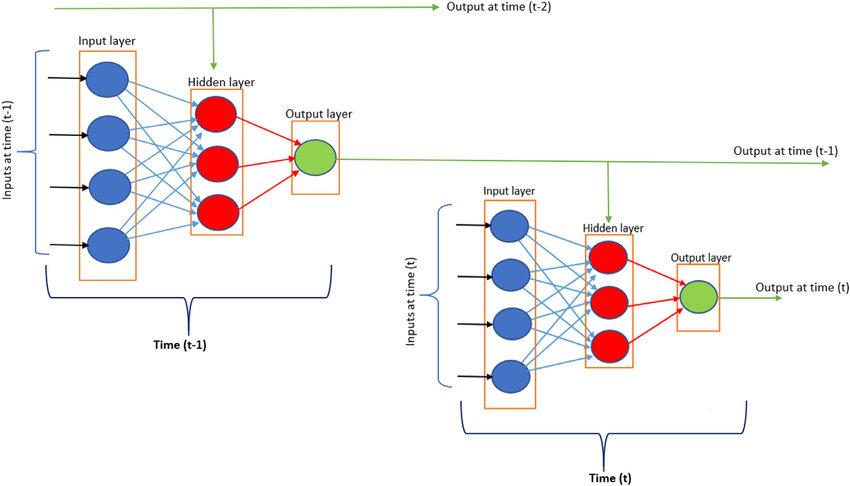


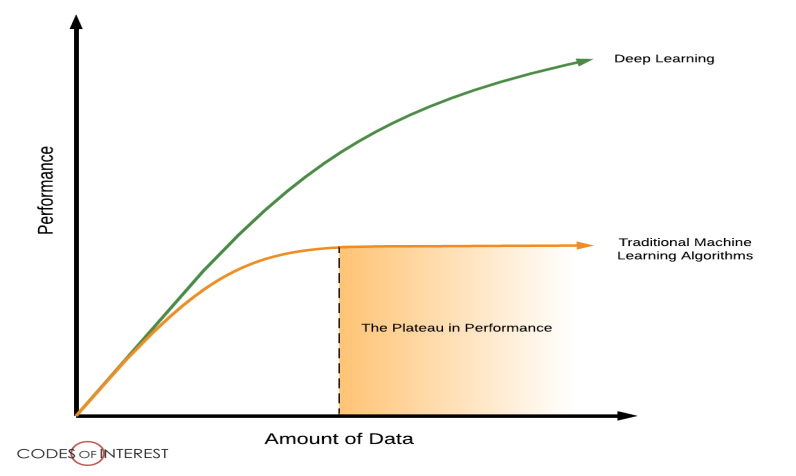
Recurrent Neural Network

RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations.

Audio is represented as 1 d temporal sequence ;hence speech recognition can be considered as rnn.

Similarly, language is also sequence based and a complex form of rnn is used.





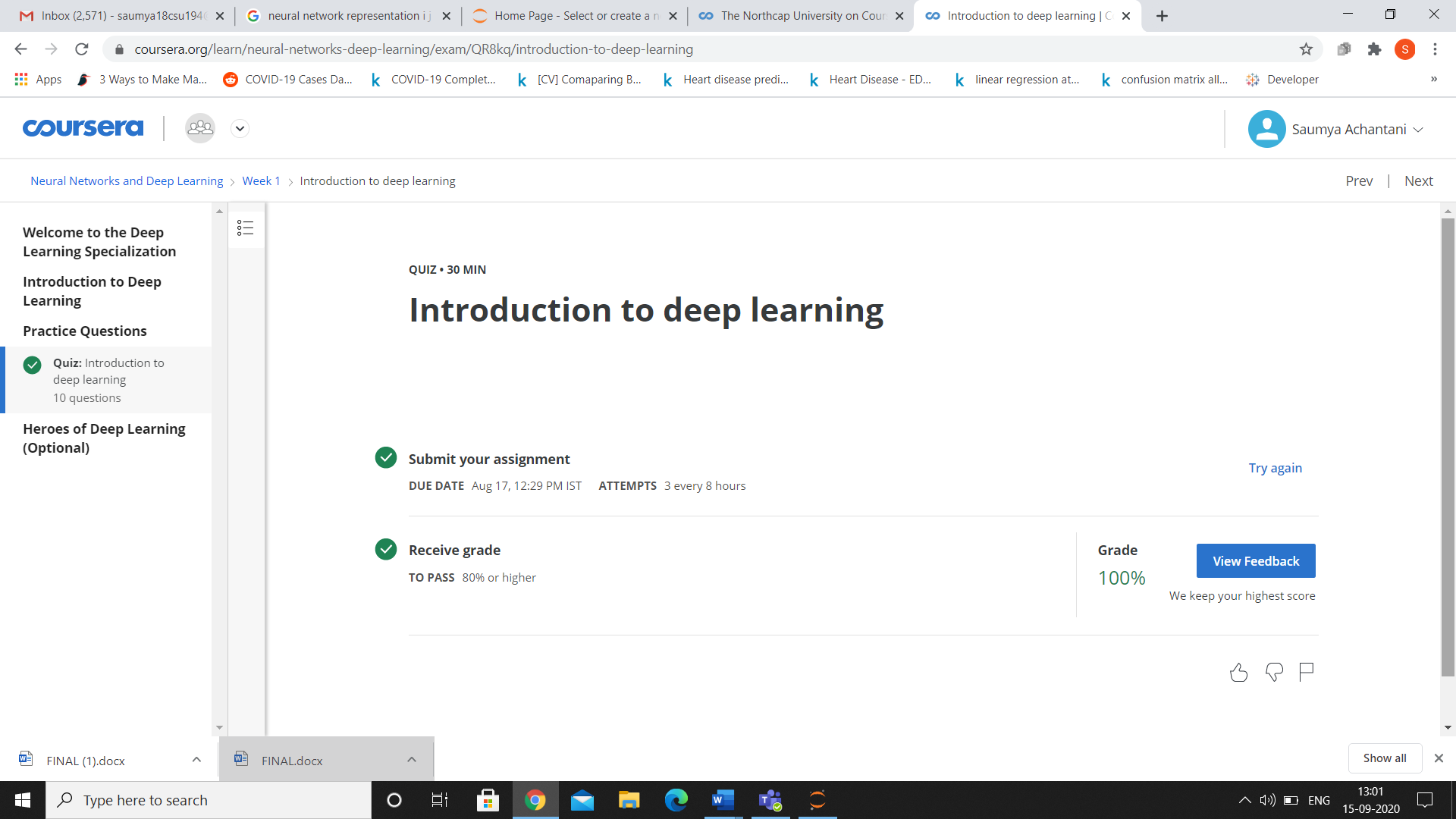
Drawback of Machine Learning:-

In Machine Learning, as we increase the data , performance increases up to a particular threshold and after that if we increase data, there is no change in performance.

In Deep learning, as we increase the data ,performance keeps on increasing.

Deep learning is based on learning data interpretations, more the data:-more learning .

QUIZ OUTCOME



# Week 2: [**Logistic Regression with a Neural Network mindset**](https://www.coursera.org/learn/neural-networks-deep-learning/programming/XaIWT/logistic-regression-with-a-neural-network-mindset)

**Key Concepts**

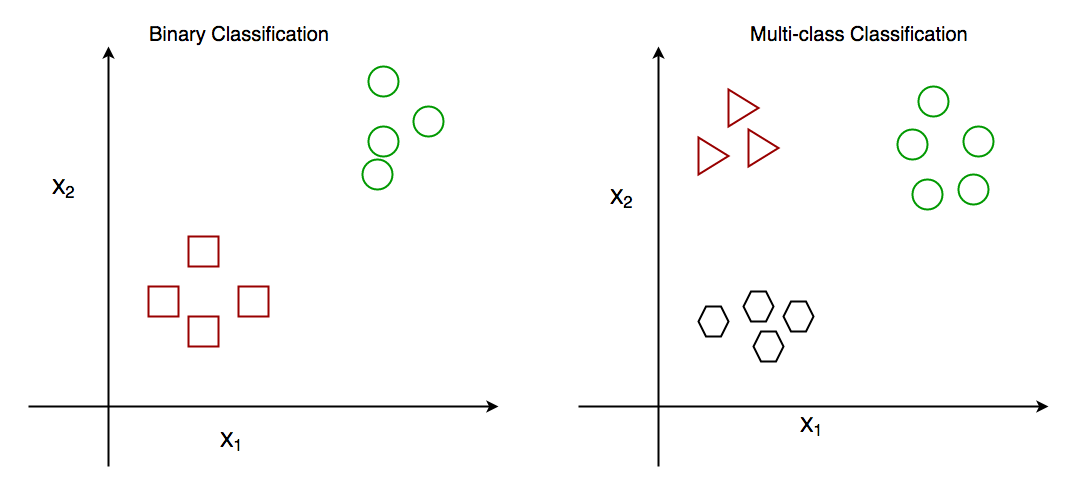
* Build a logistic regression model, structured as a shallow neural network
* Implement the main steps of an ML algorithm, including making predictions, derivative computation, and gradient descent.
* Implement computationally efficient, highly vectorized, versions of models.
* Understand how to compute derivatives for logistic regression, using a backpropagation mindset.
* Build the general architecture of a learning algorithm, including:
  + Initializing parameters
  + Calculating the cost function and its gradient
  + Using an optimization algorithm (gradient descent)
* Become familiar with Python and NumPy
* Work with Jupyter
* Be able to implement vectorization across multiple training examples.

BINARY CLASSIFICATION

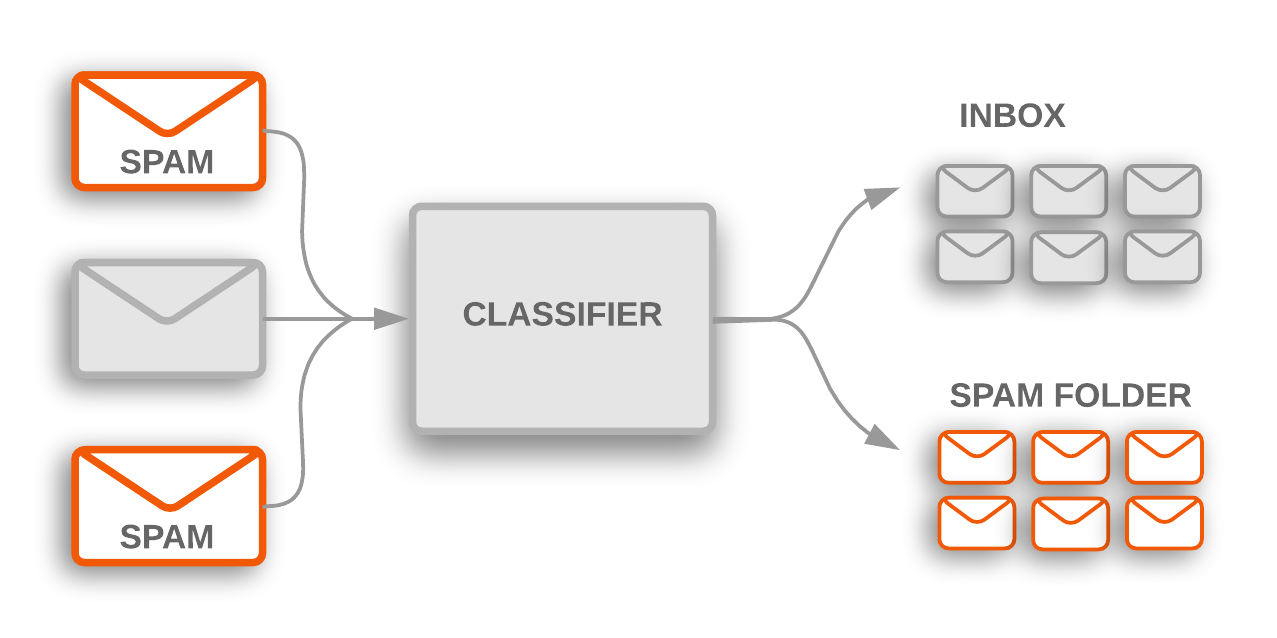
Binary classification-  The goal of binary classification is to categorize data points into one of two classes:

For example:

0 or 1, true or false, to survive or not to survive, categorizing customers into likely buyers or no etc.

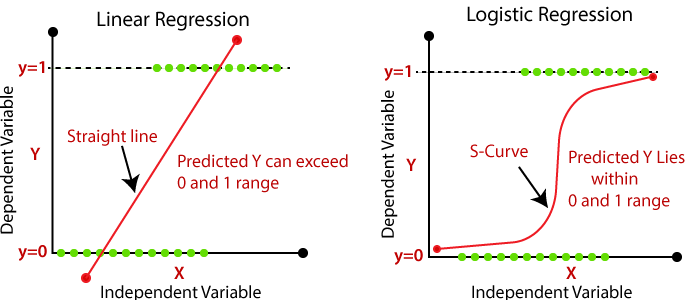


Real life application



In Machine Learning, Logistic regression can be used to solve binary classification problems.

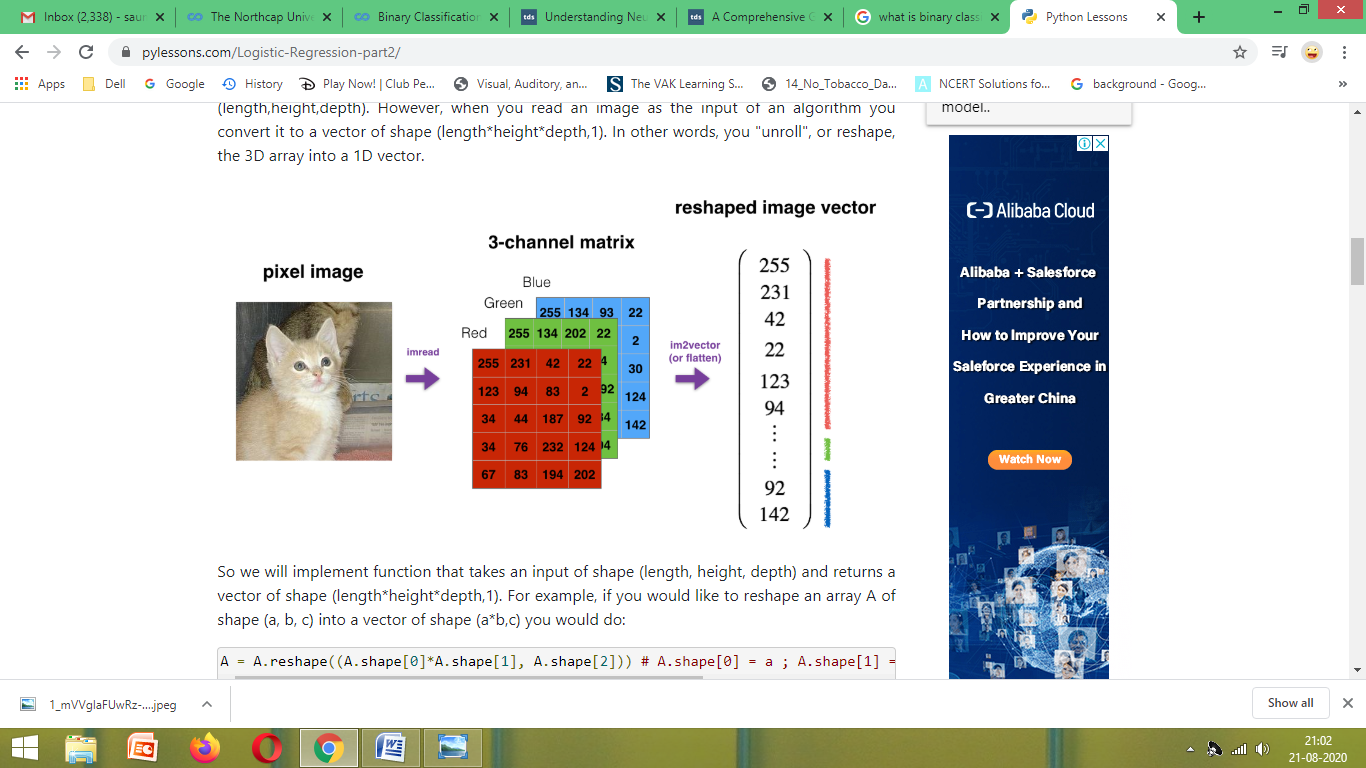
Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.



Difference between Linear and Logistic Regression

1. In case of **Linear Regression,** the outcome is continuous while in case of **Logistic Regression** outcome is discrete (not continuous)
2. To perform**Linear regression,** we require a linear relationship between the dependent and independent variables. But to perform **Logistic regression** we do not require a linear relationship between the dependent and independent variables.
3. **Linear Regression** is all about fitting a straight line in the data while **Logistic Regression** is about fitting a curve to the data.
4. **Linear Regression** is a **regression**algorithm for Machine Learning while **Logistic Regression** is a **classification**Algorithm for machine learning.

Problem : To classify an image as cat or not cat.



 Explanation of image above:

An image is represented by a 3D array(R,G,B) of shape (length,height,depth). However, when we read an image as the input of a code, we convert it to a vector of shape (length\*height\*depth,1). In other words, we reshape, the 3D array into a 1D vector[*1D Input feature matrix].*

If the image size is 64\*64 ,then dimension of input vector x will be:-

n=64\*64\*3 (since 3 matrices)

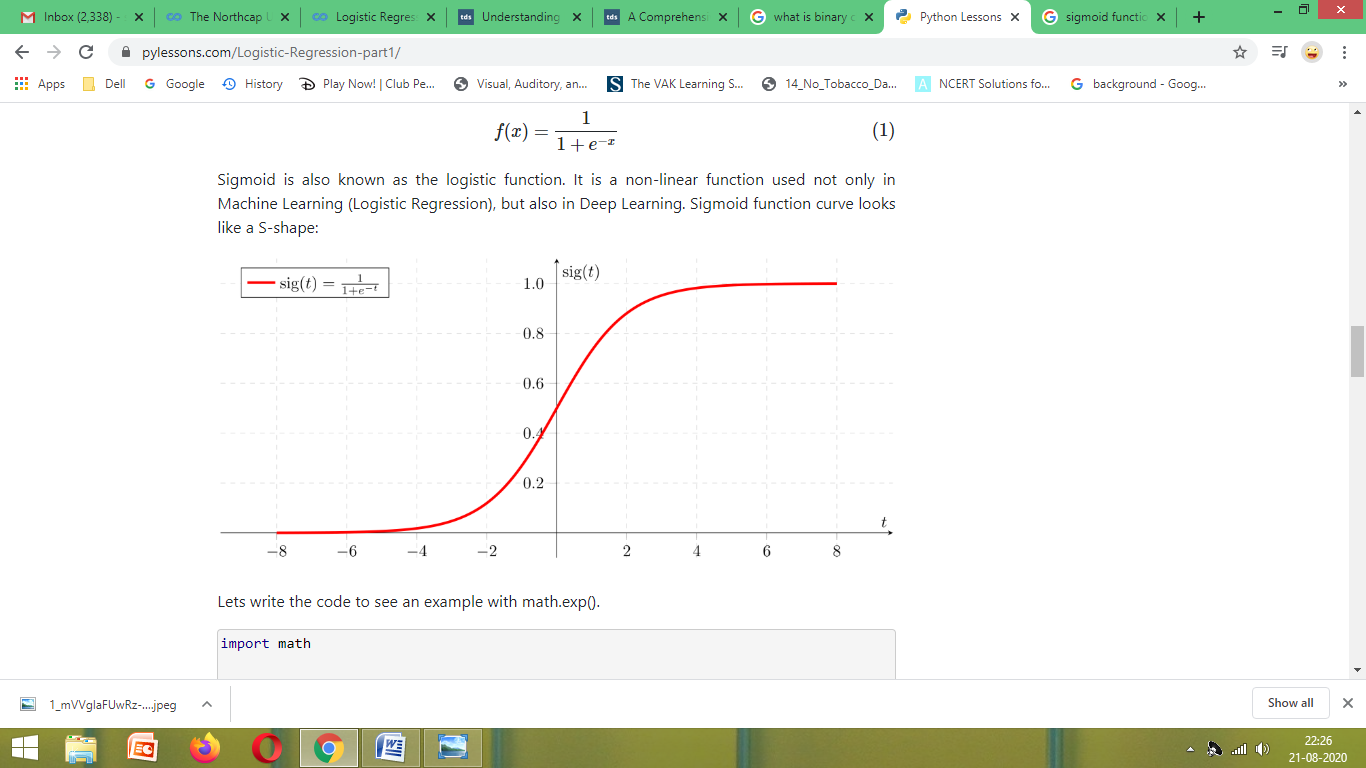
Shape=12288

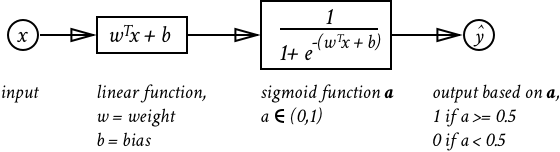
Output=0(Not cat) or 1(cat)

y^=P(y=1/x)

(Chance that image is cat image ,given the image)

* We use sigmoid function: - Since the output is confined in between 0 to 1 we convert this in the form of probability which is also confined between 0 and 1. Sigmoid function helps to achieve that.





**LOGISTIC REGRESSION LOSS FUNCTION AND COST FUNCTION**

We cannot calculate the perfect weights for a neural network; there are too many unknowns. Instead, the problem of learning is cast as a search or optimization problem and an algorithm is used to navigate the space of possible sets of weights the model may use in order to make good or good enough predictions.

**What Is a Loss Function and Loss?**

In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the objective function.

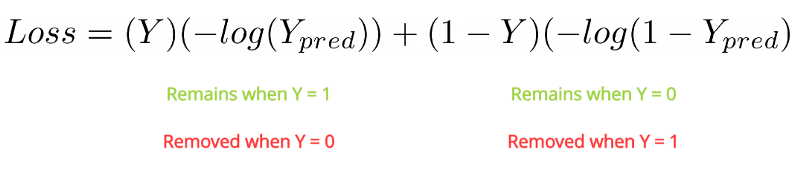
We may seek to maximize or minimize the objective function, meaning that we are searching for a candidate solution that has the highest or lowest score respectively.

Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply “*loss*.”

1.For linear functions

 (where y: actual value; y^: predicted value)

2.For non-linear functions



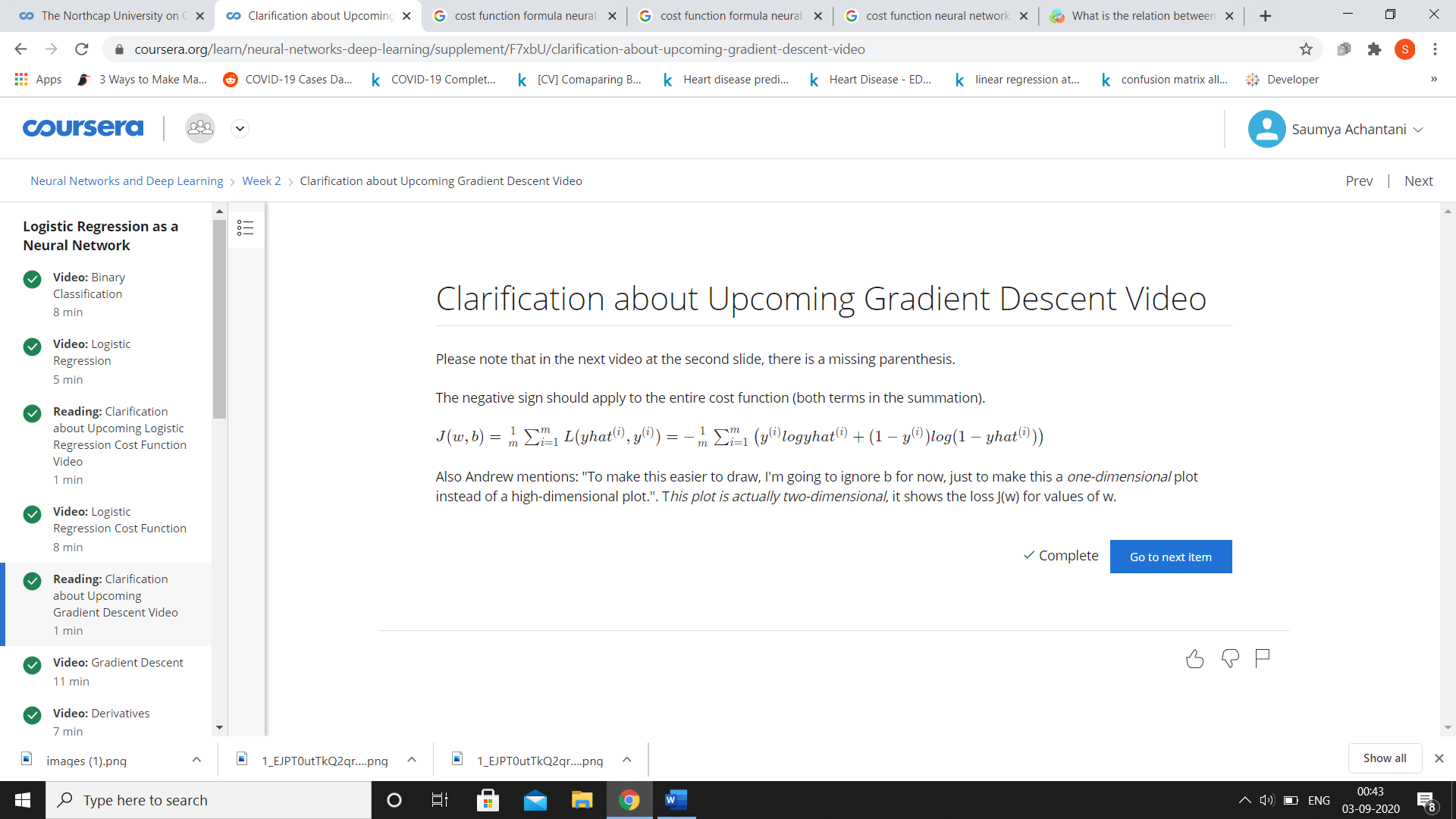
**COST FUNCTION**

The terms *cost* and *loss* functions almost refer to the same meaning. But loss function mainly applies for a single training set as compared to the cost function which deals with a number of training sets or the complete batch. It is also sometimes called an error function.

In short, we can say that the loss function is a part of the cost function. The cost function is calculated as an average of loss functions. The loss function is a value which is calculated at every instance.

So, for a single training cycle loss is calculated numerous times, but the cost function is only calculated once.

To calculate cost function:-



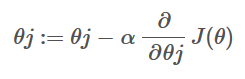
**Derivative**

The derivative is instantaneous rate of change of a function with respect to one of its variables, equivalent to finding the slope of the tangent line to the function at a point. It is basically the slope of a function at a time.

**Gradient Descent**

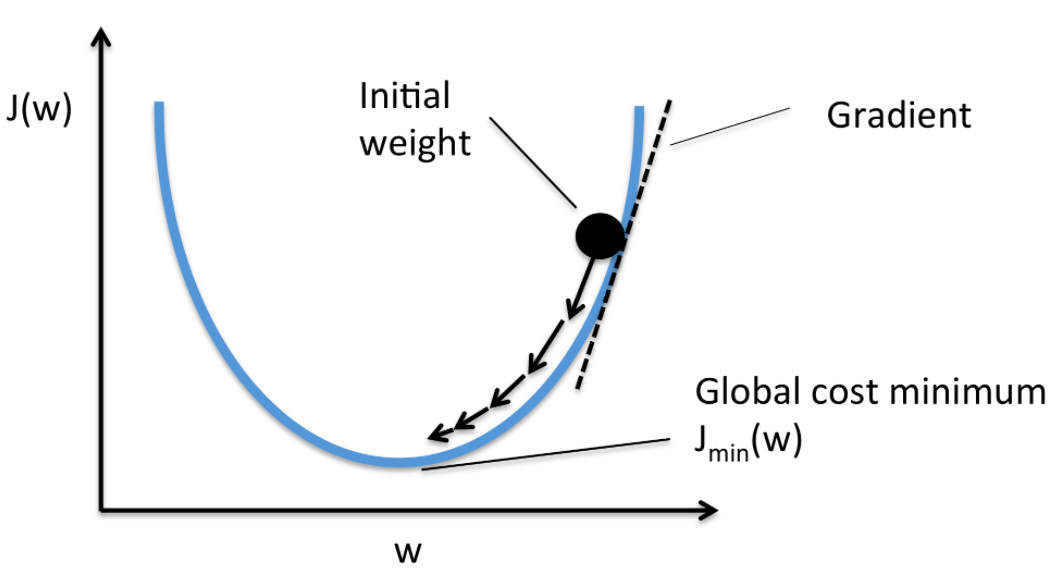
The model with a given set of weights is used to make predictions and the error for those

predictions is calculated. The gradient descent algorithm seeks to change the weights so that the next evaluation reduces the error, meaning the optimization algorithm is navigating down the gradient (or slope) of error.

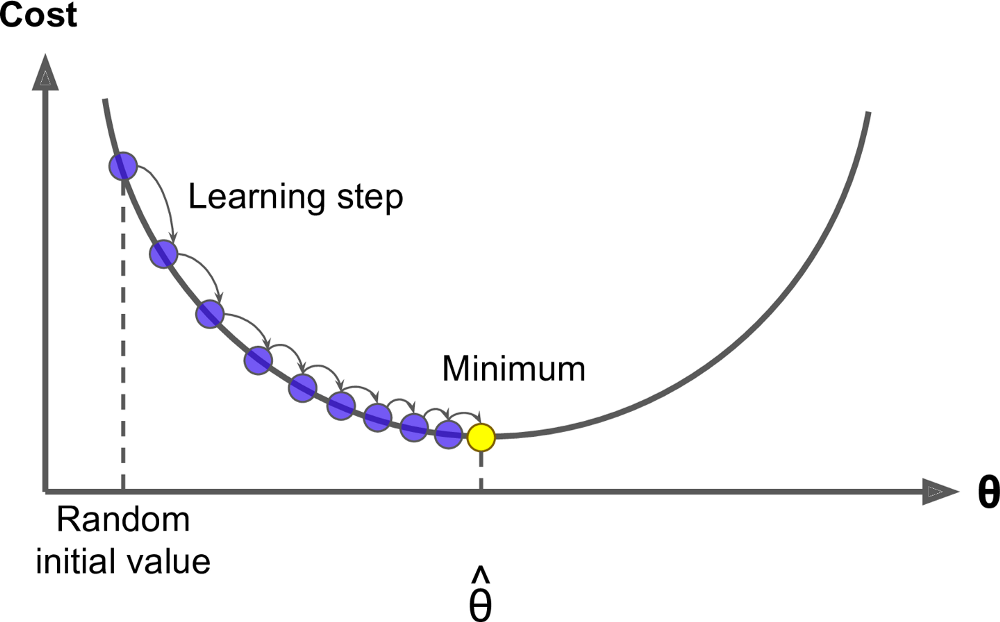


Objective:-To minimize cost function

How:-Using Gradient Descent



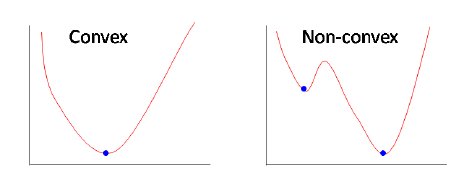
We will initialize both the parameters w and b ;Gradient descent starts at an initial point as shown in above diagram and keeps taking a downward step till we reach global minimum.



Learning step controls the size of step we take on each gradient descent.

What is convex and non-convex function?

In convex function, there is one global minimum which means only 1 minimum is there .Unlike in non-convex, where there are multiple local minima, as shown in diagram below.



If the objective function is convex, that comes with nice guarantees. Most importantly, if a function is strictly convex, it is guaranteed to have a**unique global minimum**, and it can be found by various standard methods.  
Non-convex functions may have several local minima, that is multiple points satisfying that they are the best point in their local neighborhood, but which are not globally optimal. Therefore, if you have a non-convex problem, there is usually no way to test if the solution you have found is indeed the best solution.  
Furthermore, if a problem is convex, it is usually easier to analyse the asymptotic behavior of the algorithm, that is how fast it converges as you observe more and more data.

Our cost function is convex because we want only 1 minimum value for parameters w and b.

COMPUTATIONAL GRAPH

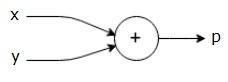
### What is a Computational Graph?

A computational graph is defined as a directed graph where the nodes correspond to mathematical operations. Computational graphs are a way of expressing and evaluating a mathematical expression.

For example, here is a simple mathematical equation −

p=x+yp=x+y

We can draw a computational graph of the above equation as follows.

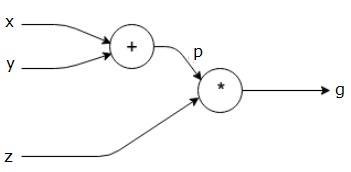


The above computational graph has an addition node (node with "+" sign) with two input variables x and y and one output q.

Let us take another example, ,We have the following equation.

g=(x+y)∗Z

The above equation is represented by the following computational graph.



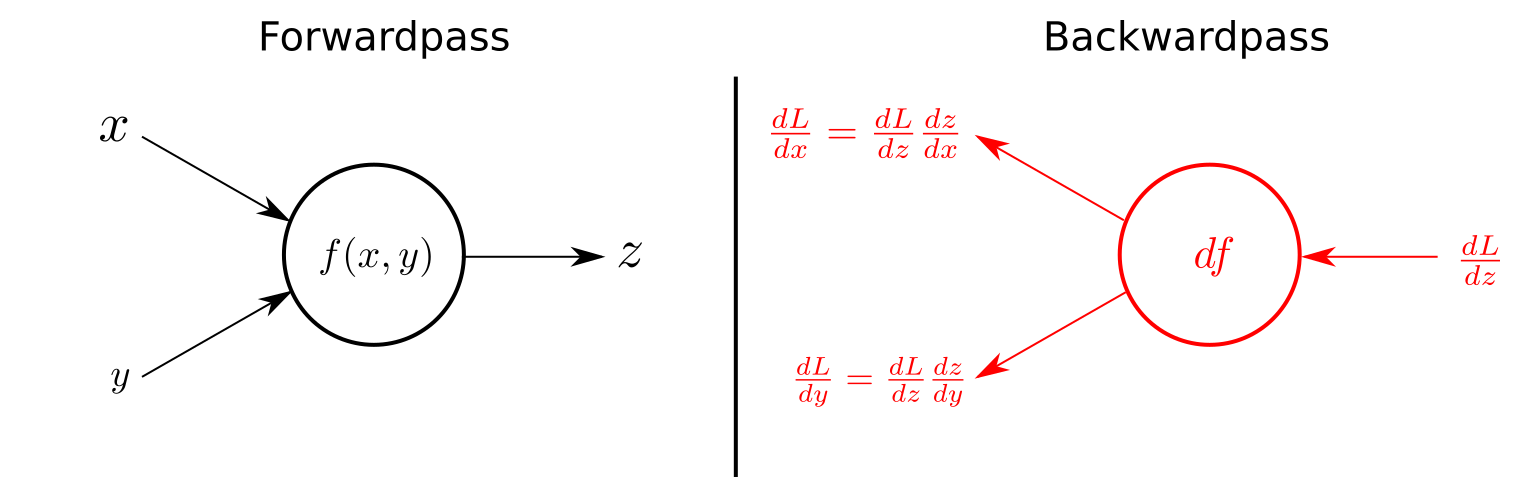
Forward Propagation

Forward propagation (or forward pass) refers to the calculation and storage of intermediate variables (including outputs) for a neural network . The input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.

Backpropagation

Backpropagation refers to the method of calculating the derivatives or gradient of neural network parameters. In short, the method traverses the network in reverse order, from the output to the input layer, according to the chain rule from calculus. We repeatedly adjust the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector.

Intuitive diagram:-



**The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.**

**Vectorization**

Vectorization is a technique by which you can make your code execute fast. In machine learning, there’s a concept of an optimization algorithm that tries to reduce the error and computes to get the best parameters for the machine learning model.

Most of the NumPy library methods are vectorised version.

**Broadcasting**

Broadcasting is a mechanism which allows tensors with different numbers of dimensions to be added or multiplied together by (virtually) replicating the smaller tensor along the dimensions that it is lacking.

**Example:**

|  |
| --- |
| import numpy as np  a = np.array([17, 11, 19]) # 1x3 Dimension array  print(a)  b = 3  print(b)  c = a + b  print(c) |

INPUT:

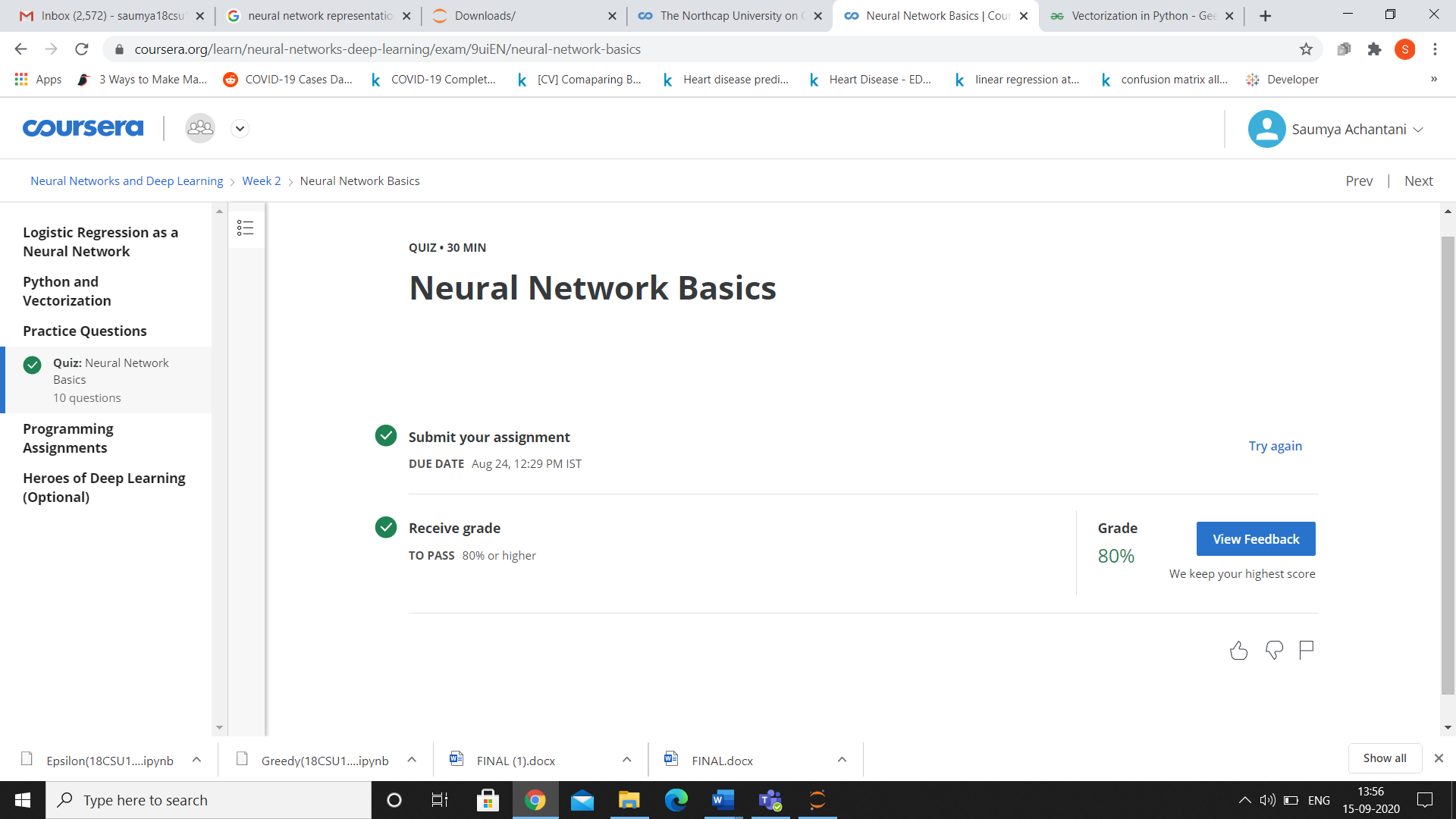
[17 11 19]

3

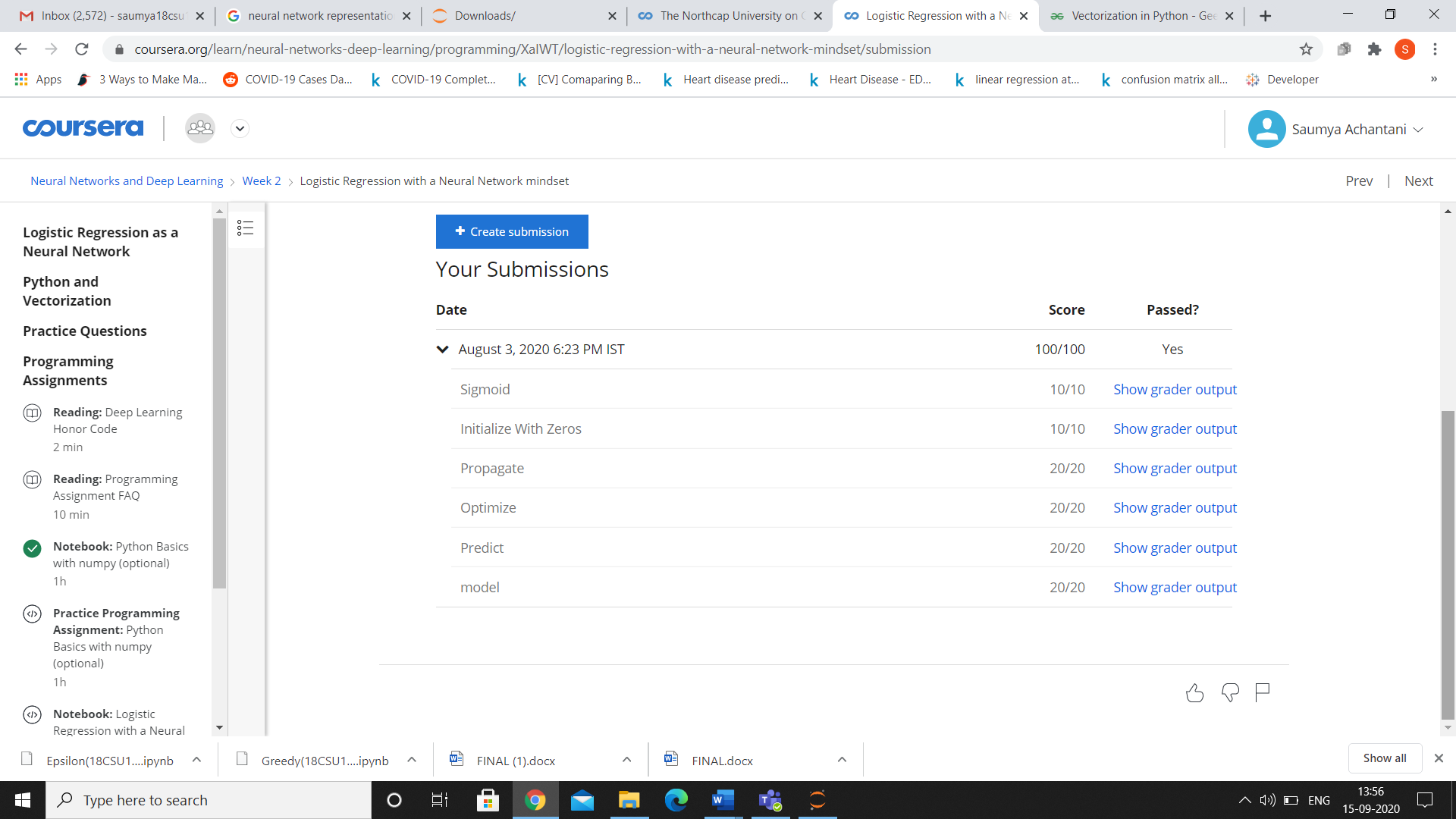
OUTPUT:

[20 14 22]

**QUIZ OUTCOME**



**ASSIGNMENT OUTCOME**



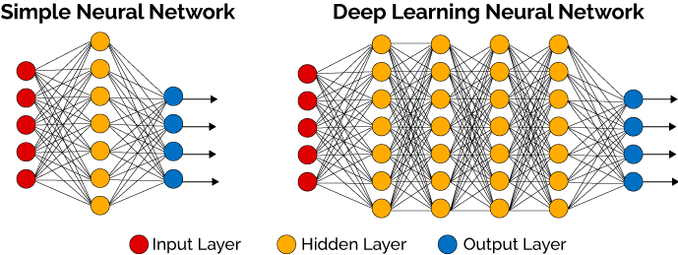
# **Week 3:** [**Planar data classification with a hidden layer**](https://www.coursera.org/learn/neural-networks-deep-learning/programming/wRuwL/planar-data-classification-with-a-hidden-layer)

**Key Concepts**

* Become fluent with Deep Learning notations and Neural Network Representations
* Understand hidden layers
* Be able to apply a variety of activation functions in a neural network.
* Build your first forward and backward propagation with a hidden layer
* Apply random initialization to your neural network
* Build and train a neural network with one hidden layer.

## How does a Hidden Layer work?

Hidden layers, simply put, are layers of mathematical functions each designed to produce an output specific to an intended result.

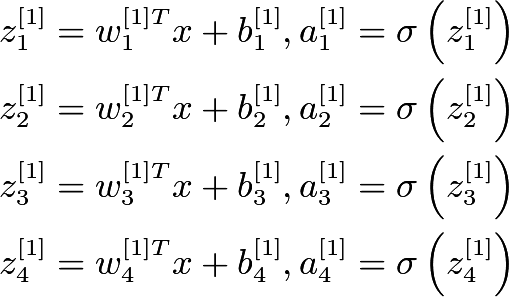


Hidden layers allow for the function of a neural network to be broken down into specific transformations of the data. Each hidden layer function is specialized to produce a defined output. For example, a hidden layer functions that are used to identify human eyes and ears may be used in conjunction by subsequent layers to identify faces in images. The deeper layers of a neural network are typically computing more complex features of the input than the earlier layers.

Easy analogy of hidden layers:

If you want a computer to tell you if there's a bus in a picture, So your bus detector might be made of a wheel detector (to help tell you it's a vehicle) and a size detector (to tell you it's too big to be a car). These are the three elements of your hidden layer: they're not part of the raw image, they're tools you designed to help you identify buses. If all those detectors are active, then there's a good chance you have a bus in front of you.

Representations in neural network



1. The superscript number ***[i]*** denotes the layer number and the subscript number ***j*** denotes the neuron number in a particular layer.
2. ***X*** is the input vector consisting of n features.
3. ***W[i]j***is the weight associated with neuron ***j*** present in the layer ***i***.
4. ***b[i]j***is the bias associated with neuron ***j*** present in the layer ***i***.
5. ***Z[i]j***is the intermediate output associated with neuron ***j*** present in the layer ***i***.
6. ***A[i]j***is the final output associated with neuron ***j*** present in the layer ***i***.
7. ***Sigma*** is the sigmoid activation function.

Activation function

Activation function is a function that is added into an artificial neural network in order to help the network learn complex patterns in the data. It is a transfer function that is used to map the output of one layer to another.

Uses of Activation Function:

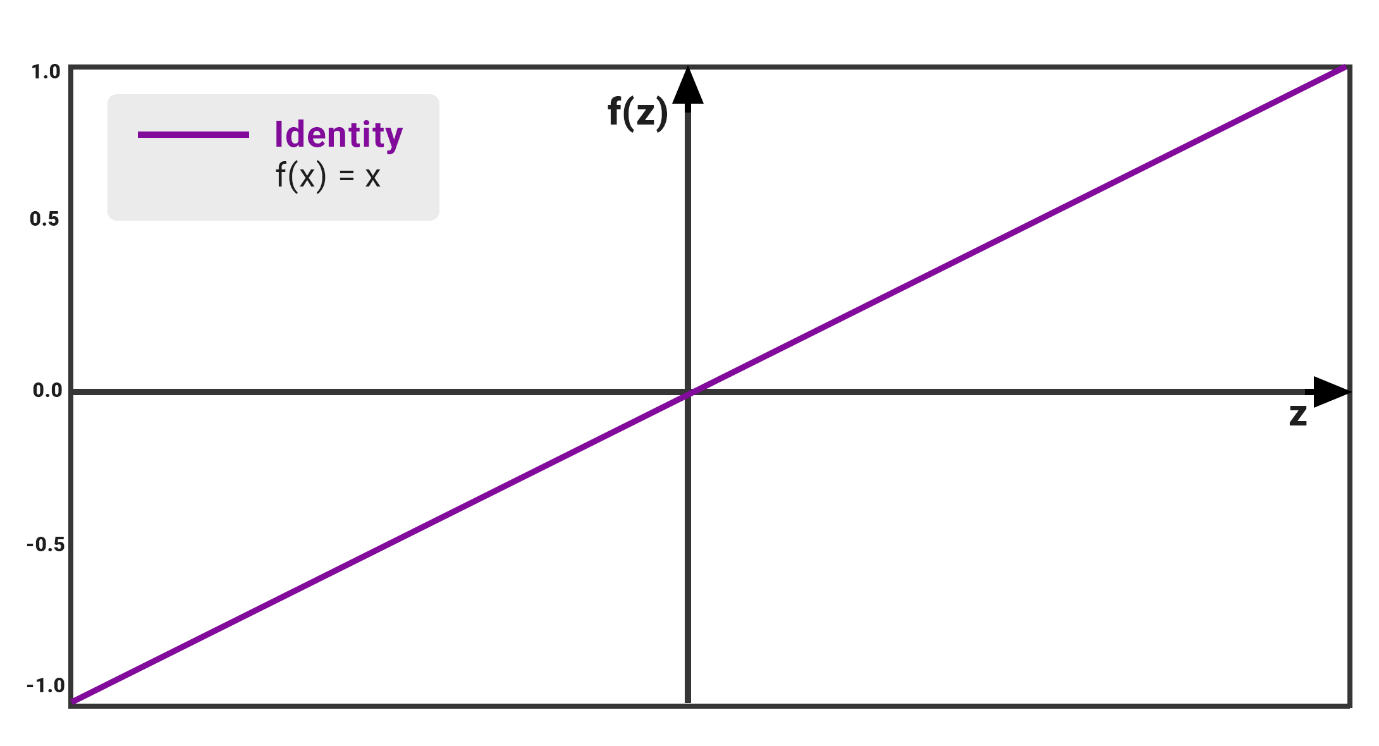
1.Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.

The purpose of the activation function is to introduce non-linearity into the output of a neuron.

2. It is important because input into the activation function is W\*x + b where W is the weights of the cell and the x is the inputs and then there is the bias b added to that. This value if not restricted to a certain limit can go very high in magnitude especially in case of very deep neural networks that have millions of parameters. This will lead to computational issues.

What is the problem with linear functions?

The value of *f(z)* increases proportionally with the value of *z*. The input value is the weighted sum of the weights and biases of the neurons in a layer. The linear function solves the issue of a binary step function where it reports only a value of 0 and 1.



**Fig :**Performance of Linear or Identity Activation Function

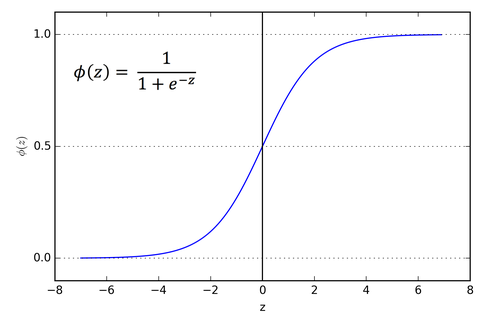
The output of the function is not confined between any range; that is, the value of *f(z)* can go from which not necessarily a problem as we can proceed into the next or final layer by taking the max value of the neurons that have fired after com. Apart from that, the linear activation function has its set of disadvantages such as:

* We observe that the function’s derivative is a constant. That means there is constant gradient descent occurring since there is no relation to the value of z.
* Our model is not really learning as it does not improve upon the error term, which is the whole point of the neural network.
* Since the activation is linear, nesting in 2 or N number of hidden layers with the same function will have no real effect. The N layers could basically be squashed into one layer.

We see that this function is not fit to handle complex. So, in order to fix this issue, we use non-linear functions to enable our model to learn iteratively.

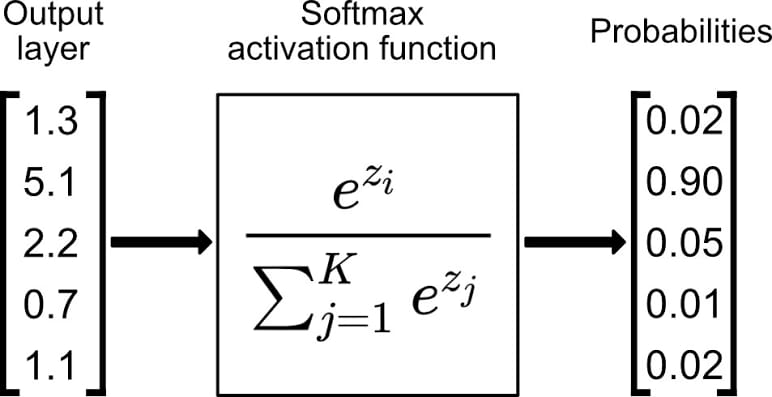
**VARIANTS OF ACTIVATION FUNCTION :-**

**1). Sigmoid Function :-**

****

* It is a function which is plotted as **‘S’** shaped graph.
* **Equation :**  
  A = 1/(1 + e-z)
* **Value Range :**0 to 1
* **Uses :**Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

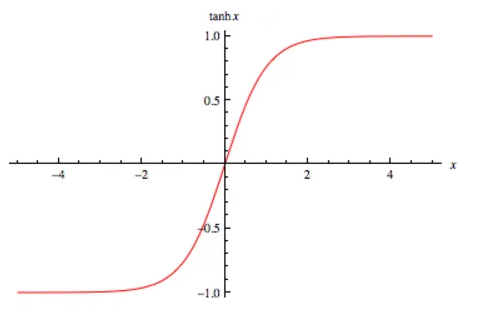
**2). SoftMax Function :-**

****

The SoftMax function is also a type of sigmoid function but is handy when we are trying to handle multi classification problems. This function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

* **Nature :-**non-linear
* **Uses :-**Usually used when trying to handle multiple classes. The SoftMax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.
* **Output:-**The SoftMax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

**3). Tanh Function :-**

****

The activation that works almost always better than sigmoid function is Tanh function also knows as **Tangent Hyperbolic function**. It’s actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.

 **Equation :-**

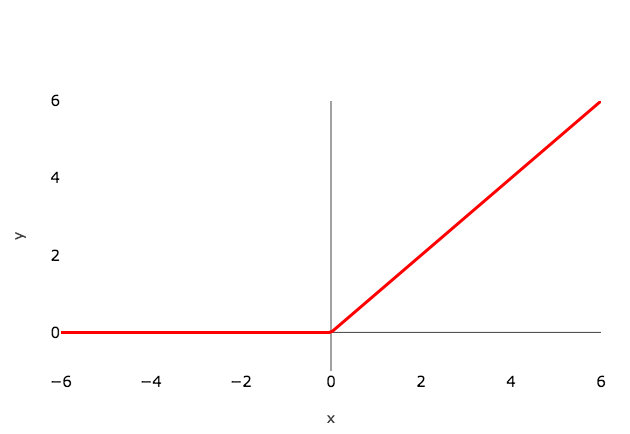
f(x) = tanh(x) = 2/(1 + e-2x) - 1

OR

tanh(x) = 2 \* sigmoid(2x) - 1

* **Value Range :-**-1 to +1
* **Nature :-**non-linear
* **Uses :-**Usually used in hidden layers of a neural network as its values lies between **-1 to 1**hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.

**4). RELU :-**

****

Stands for Rectified linear unit. It is the most widely used activation function. Chiefly implemented in hidden layers of Neural network.

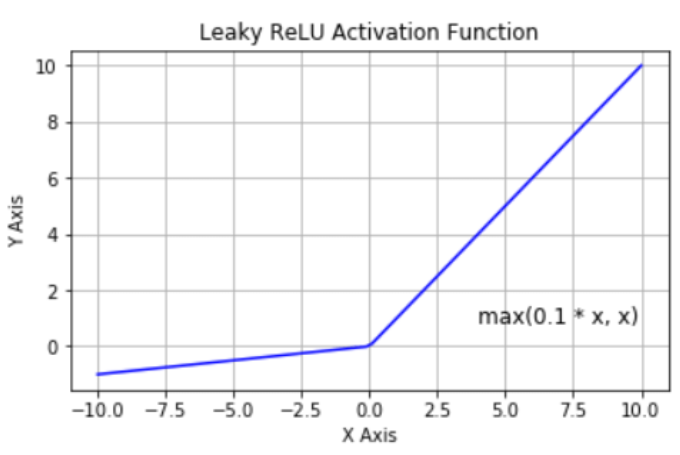
* **Equation :- *A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
* **Value Range :-**[0, inf)
* **Nature :-**non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
* **Uses :-**ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation. In simple words, RELU learns *much faster* than sigmoid and Tanh function.

This is a widely used activation function, especially with Convolutional Neural networks. It is easy to compute and does not saturate and does not cause the Vanishing Gradient Problem\*( the **gradient** will be vanishingly small, effectively preventing the weight from changing its value.).

**Issues:**

* It has just one issue of not being zero centered. It suffers from **“dying ReLU”** problem. Since the output is zero for all negative inputs. It causes some nodes to completely die and not learn anything.
* Another problem with ReLU is of exploding the activations since it higher limit is, well, inf. This sometimes leads to unusable nodes. Once a ReLU ends up in this state, it is unlikely to recover, because the function gradient at 0 is also 0, so gradient descent learning will not maximize the weights. **“Leaky” ReLU** with a small positive gradient for negative inputs (y=0.01x when x < 0 say) **are one attempt** to address this issue and give a chance to recover.

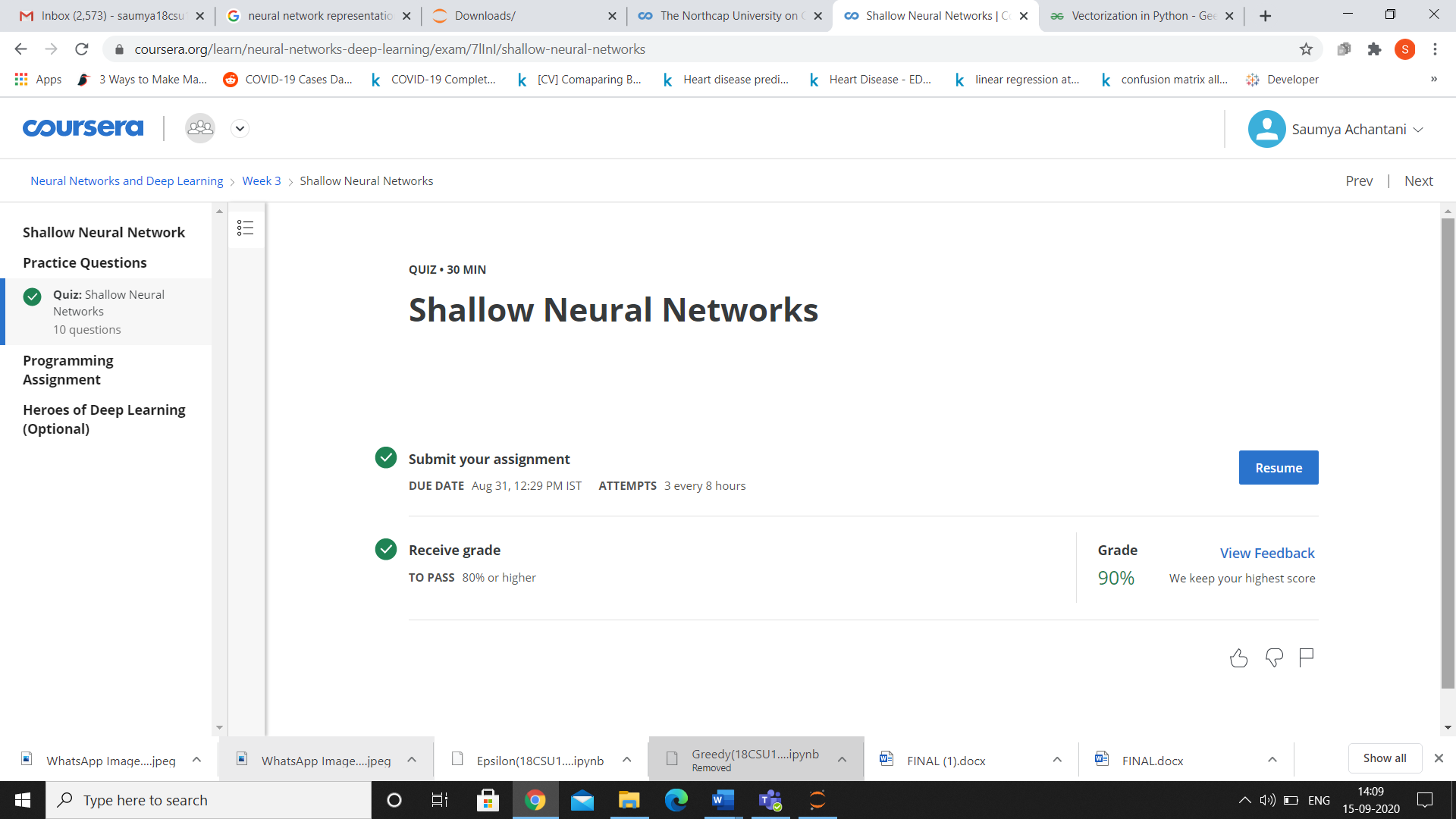
5. **Leaky ReLU and Parametric ReLU**: It is defined as **f(x) = max(αx, x)**



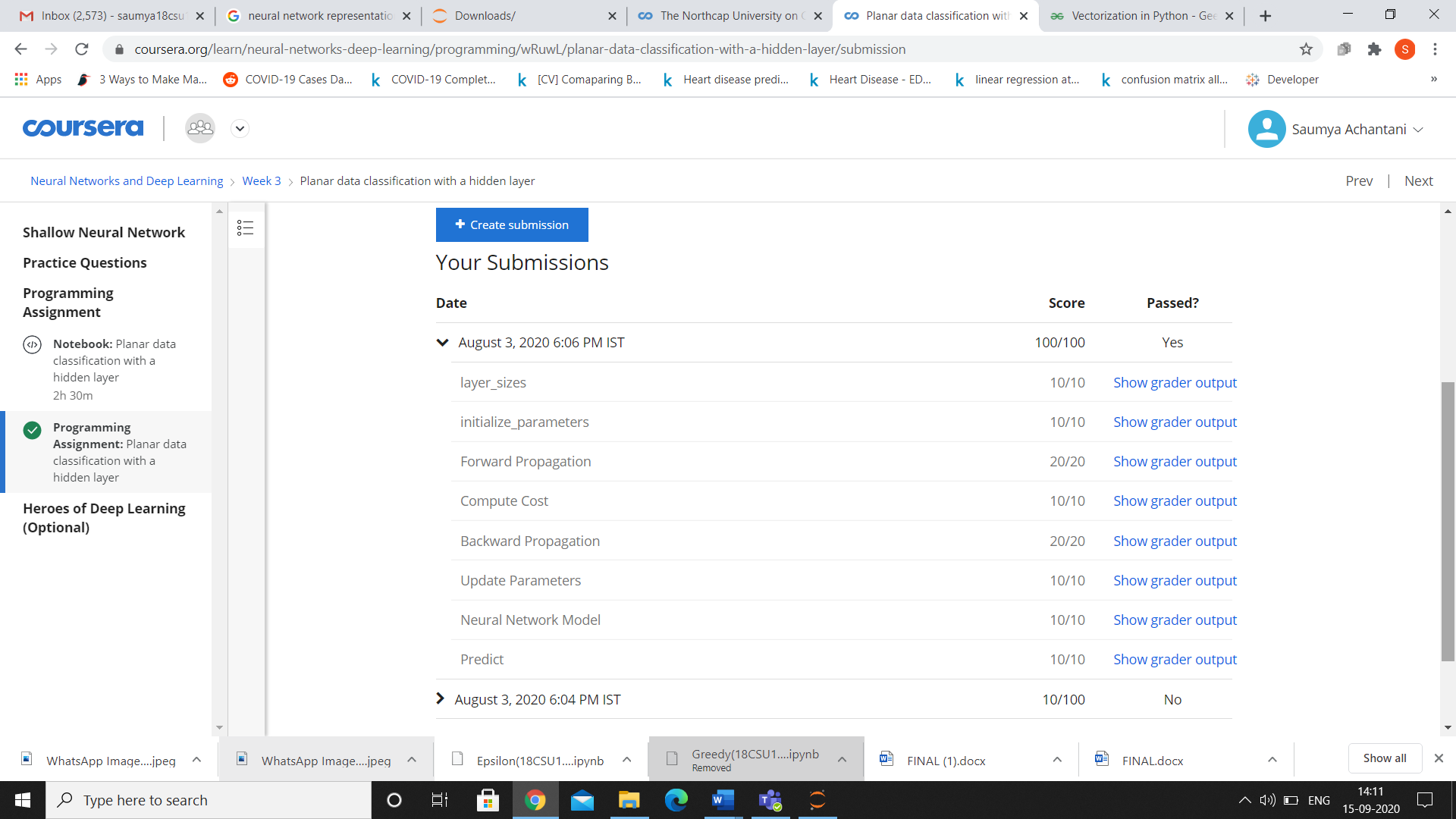
Clearly, Leaky ReLU solves the **“dying ReLU”**problem to some extent. Note that, if we set α as 1 then Leaky ReLU will become a linear function f(x) = x and will be of no use. Hence, the value of **α is never set close to 1.**If we set **α**as a hyperparameter for each neuron separately, we get **parametric ReLU** or **PReLU**.

**CHOOSING THE RIGHT ACTIVATION FUNCTION**

* The basic rule of thumb is if you really don’t know what activation function to use, then simply use *RELU* as it is a general activation function and is used in most cases these days.
* If your output is for binary classification then, *sigmoid function* is very natural choice for output layer.

**QUIZ OUTCOME**

**PROGRAMMING ASSIGNMENT OUTCOME**



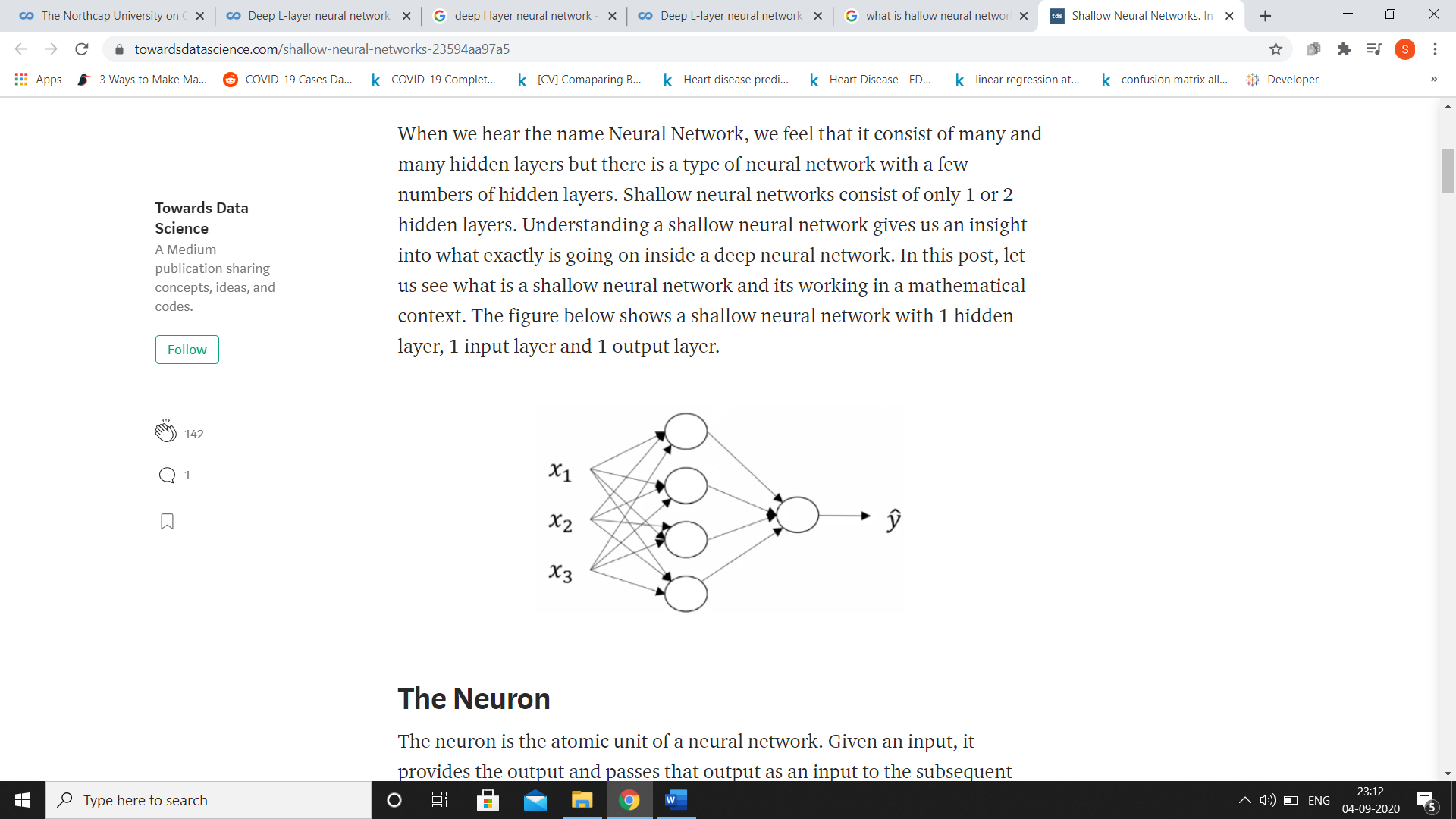
# Week 4:[Building your deep neural network](https://www.coursera.org/learn/neural-networks-deep-learning/programming/c4HO0/building-your-deep-neural-network-step-by-step) and Application of Deep Learning

**Key Concepts**

* See deep neural networks as successive blocks put one after each other
* Build and train a deep L-layer Neural Network
* Analyze matrix and vector dimensions to check neural network implementations.
* Understand how to use a cache to pass information from forward propagation to back propagation.
* Understand the role of hyperparameters in deep learning

SHALLOW NEURAL NETWORK

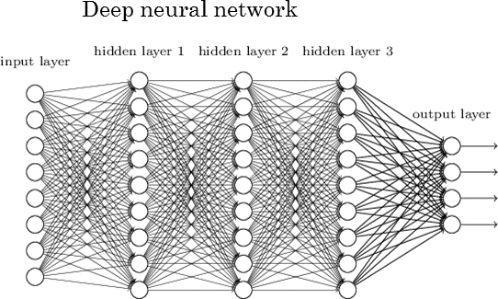
 Shallow neural networks consist of only 1 or 2 hidden layers.

The figure below shows a shallow neural network with 1 hidden layer, 1 input layer and 1 output layer. 

When we count layers in a neural network, we don't count the input layer, we just count the hidden layers and the output layer. So, this would be a 2-layer neural network and is quite shallow.

DEEP NEURAL NETWORK

A **deep neural network**  a n**eural network** with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship.



Difference between a shallow and a deep neural network

1.Deep models are able to extract/build better features than shallow models and to achieve this they are using the intermediate hidden layers. The idea of having more layers is to extract more finer features of the input vector . So generally, as we increase the depth of the model, we increase the power of the model at the cost of the computational complexity .

2. A shallow neural network which takes only input as vectors, deep neural networks, is able to take raw data such as images and text and automatically extract the necessary features to learn the data better.

What is the "cache" used for in implementation of neural networks?

Cache records values from the forward propagation units and sends it to the backward propagation units because it is needed to compute the chain rule derivatives.

We use it to pass variables computed during forward propagation to the corresponding backward propagation step. It contains useful values for backward propagation to compute derivatives.

Model Hyper-parameter and parameter tuning

A model parameter is a configuration variable that is internal to the model and whose value can be estimated from data.

* They are required by the model when making predictions.
* They value define the skill of the model on your problem.
* They are estimated or learned from data.
* They are often not set manually by the practitioner.
* They are often saved as part of the learned model.

 Model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data. Hyperparameters are the variables which determines the network structure(E.g.: Number of Hidden Units) and the variables which determine how the network is trained(ego: Learning Rate).

**Hyperparameters** are **set before training**(before optimizing the weights and bias).

* They are often used in processes to help estimate model parameters.
* They are often specified by the practitioner.
* They can often be set using heuristics.
* They are often tuned for a given predictive modeling problem.

When a machine learning algorithm is tuned for a specific problem, such as when you are using a grid search or a random search, then you are tuning the hyperparameters of the model or order to discover the parameters of the model that result in the most skillful predictions.

Some examples of model hyperparameters include:

* The learning rate for training a neural network.
* Choice of activation function

Many hidden units within a layer with regularization techniques can increase accuracy. Smaller number of units may cause **underfitting**.

Hyper parameters control the ultimate parameters W and B .

**1.Learning rate:**

The amount that the weights are updated during **training** is referred to as the step size or the “**learning rate**.” Specifically, the **learning rate** is a configurable hyperparameter used in the **training** of **neural networks** that has a small positive value, often in the range between 0.0 and 1.0.

## 2.Number of epochs

Number of epochs is the number of times the whole training data is shown to the network while training.

The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches.

The number of epochs is traditionally large, often hundreds or thousands, allowing the learning algorithm to run until the error from the model has been sufficiently minimized. You may see examples of the number of epochs set to 10, 100, 500, 1000, and larger.

**3.Sample**

A sample is a single row of data.

It contains inputs that are fed into the algorithm and an output that is used to compare to the prediction and calculate an error.

A training dataset is comprised of many rows of data, e.g. many samples. A sample may also be called an instance, an observation, an input vector, or a feature vector.

**4.Batch**

The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters.

A training dataset can be divided into one or more batches.

When all training samples are used to create one batch, the learning algorithm is called batch gradient descent. When the batch is the size of one sample, the learning algorithm is called stochastic gradient descent. When the batch size is more than one sample and less than the size of the training dataset, the learning algorithm is called mini-batch gradient descent.

* **Batch Gradient Descent**. Batch Size = Size of Training Set
* **Stochastic Gradient Descent**. Batch Size = 1
* **Mini-Batch Gradient Descent**. 1 < Batch Size < Size of Training Set

In the case of mini-batch gradient descent, popular batch sizes include 32, 64, and 128 samples. You may see these values used in models in the literature and in tutorials.

**What if the dataset does not divide evenly by the batch size?**

This can and does happen often when training a model. It simply means that the final batch has fewer samples than the other batches.

Alternately, you can remove some samples from the dataset or change the batch size such that the number of samples in the dataset does divide evenly by the batch size.

**What Is the Difference Between Batch and Epoch?**

The batch size is a number of samples processed before the model is updated.

The number of epochs is the number of complete passes through the training dataset.

The size of a batch must be more than or equal to one and less than or equal to the number of samples in the training dataset.

The number of epochs can be set to an integer value between one and infinity. You can run the algorithm for as long as you like and even stop it using other criteria besides a fixed number of epochs, such as a change (or lack of change) in model error over time.

They are both integer values and they are both hyperparameters for the learning algorithm, e.g. parameters for the learning process, not internal model parameters found by the learning process.

Example: if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch.

You must specify the batch size and number of epochs for a learning algorithm.

# **OUTCOMES ON NEW DATASET**

**Aim:** To Predict the Clean and Messy Room **.**

**Description:**

Scene recognition is one of the determining tasks of the computer vision. Our project focuses on this task.

The dataset contains train and validate folder ,inside which there are 2 more folders for clean and messy room images.

****  ****

**Project link :-**

# **CONCLUSION**

By the end of this specialization course, I have a good understanding of deep learning and neural networks. Not only was I able to execute deep learning using python but I was also able to understand how the calculations are done in neurons and how the output of a given problem is generated. This course gave me a basic idea of neural networks. The concepts taught in this course were to the point and assignments were very interesting.

# **REFERENCE**

* [www.towardsdatascience.com](http://www.towardsdatascience.com)
* [www.analyticsvidhya.com](http://www.analyticsvidhya.com)
* [www.mygreatlearning.com](http://www.mygreatlearning.com)
* [www.wikipedia.org](http://www.wikipedia.org)