By: Prashant Kr Mali

AI Assignment

Module - 1 & 2

Tasks to be performed:

Q. 1: Define a use case of Deep Learning in your current domain (if fresher, pick any of your favorites) and propose a solution through DL.

ANSWER:

- **Use case with domain: Civil Engineering**

I belong to the Civil Engineering domain. However, construction projects pose a unique set of challenges due to their scale and the number of contractors involved. That is why civil engineering companies are turning to machine learning and data science consulting to help construct and design roads, bridges, and other infrastructure projects. Historically, some machine learning algorithms were more popular than others in the field.

1 - Evaluation Computation

Evolutionary modeling or computation is an AI category based on principles and concepts of evolutionary biology (i.e., Darwinian) and population genetics. Thanks to an iterative process, it offers an effective way to tackle complex optimization problems. This machine learning technique is widely applied in design engineering to automate design production.

2 - Artificial neural networks (ANNs)

ANNs exhibit excellent performance in lots of areas, including construction. Artificial neural networks are modeled after the brain and can be trained to recognize patterns. This makes them useful for decision-making, pattern recognition, forecasting, and data analysis. Civil engineering includes all those tasks. Thus, ANNs are widely present in studying building materials, defect detection, geotechnical engineering, and construction management.

- **Solution: Smart construction design**

Constructing a building isn't a one-day task that involves lots of pre-planning. Sometimes, it may take years to bring a particular vision to life. Therefore, the planning stage in construction has a lot to benefit from smart systems i.e., Deep learning-based models combined with big data technologies.

Thus, AI-enabled tools and programs can automate the calculation and environmental analysis. Instead of manually compiling all the data, like, weather data, material properties, and others.

Moreover, artificial intelligence has strengthened the core 3D construction system called BIM, Building Information Modeling allows architects to create data-laden models based on the comprehensive information layer & machine intelligence can take the form of virtual and augmented reality.

Q. 2: A steel manufacturing plant is spending many hours in the manual inspection of their steel product (flat 2*2 ft steel plates) The inspection is a crucial part of their delivery cycle.

- **Design and detail a proposal to automate the process using DL.**
- **Please, explain your major questions to the client regarding the process will be & data.**

ANSWER:

a. Design and detail a proposal to automate the process using DL.

Quality inspection in manufacturing using deep learning-based computer vision. Improving yield by removing bad quality material with image recognition

Automation in Industrial manufacturing:

Today's increased level of automation in manufacturing also demands automation of material quality inspection with little human intervention. The trend is to reach human-level accuracy or more in quality inspection with automation. To stay competitive, modern Industrial firms strive to achieve both quantity and quality with automation without compromising one over the other. This posting takes the user through a use case of deep learning and showcases the need for optimizing the full stack (algorithms, inference framework, and hardware accelerators) to get the optimal performance.

Deep Learning for Quality inspection:

To meet industry standards quality inspectors in manufacturing firms, inspect product quality usually after the product is manufactured, it's a time-consuming manual effort, and a rejected product results in wasted upstream factory capacity, consumables, labor, and cost. With the modern trend of Artificial Intelligence, industrial firms are looking to use deep learning-based computer vision technology during the production cycle itself to automate material quality inspection. The goal is to minimize human intervention and at the same time reach human-level accuracy or more as well as optimize factory capacity, labor cost, etc. The usage of deep learning is varied, from object detection in self-driving cars to disease detection with medical imaging deep learning has proved to achieve human-level accuracy & better.

Problem Statement:

Detecting bad quality material in steel manufacturing is an error-prone & time-consuming manual process and results in false positives (detecting a bad one as a good one). If a faulty component/part is detected at the end of the production line there is a loss in upstream labor, consumables, factory capacity as well as revenue. On the other hand, if an undetected bad part gets into the final product there will be customer impact as well as market reaction. This could potentially lead to unrepairable damage to the reputation of the organization.

Summary:

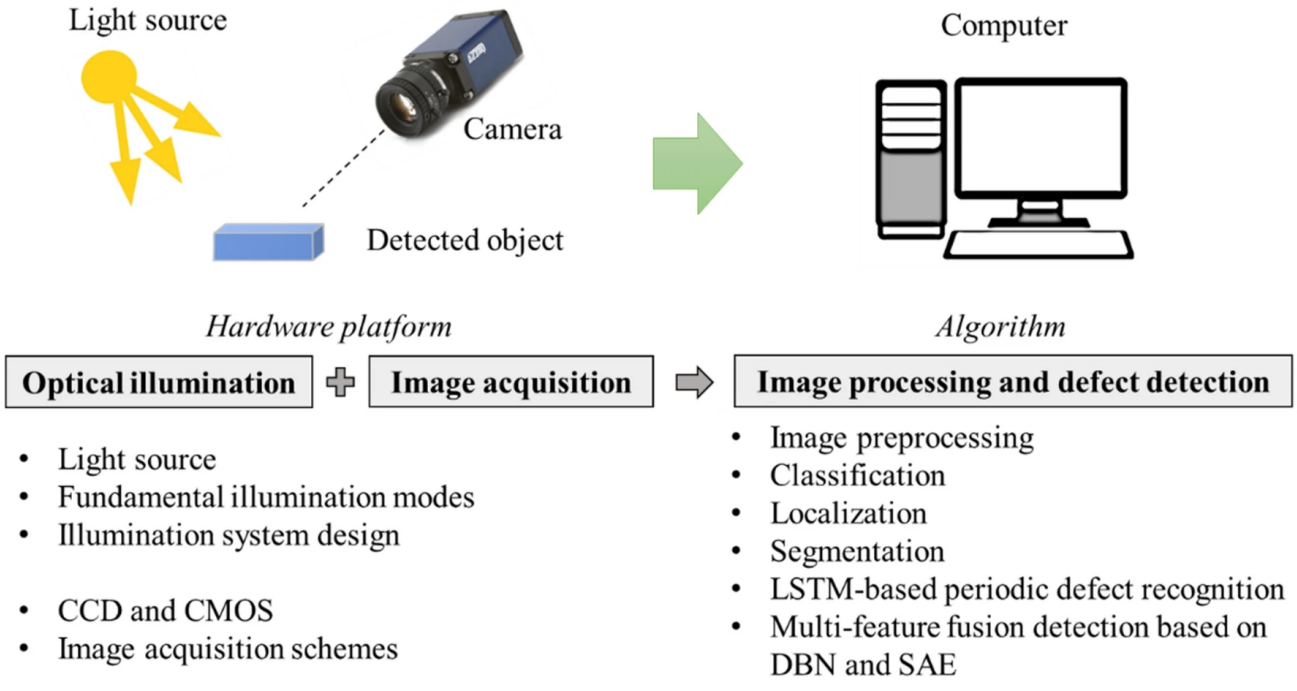
Automating defect detection on steel products using deep learning. During hardware manufacturing processes there could be damages such as scratches/cracks which make our products unusable for the next processes in the production line. Deep learning applications detected defects such as a crack/scratch in milliseconds with human-level accuracy and better as well as interpreted the defect area in the image with heat maps.

Approach:

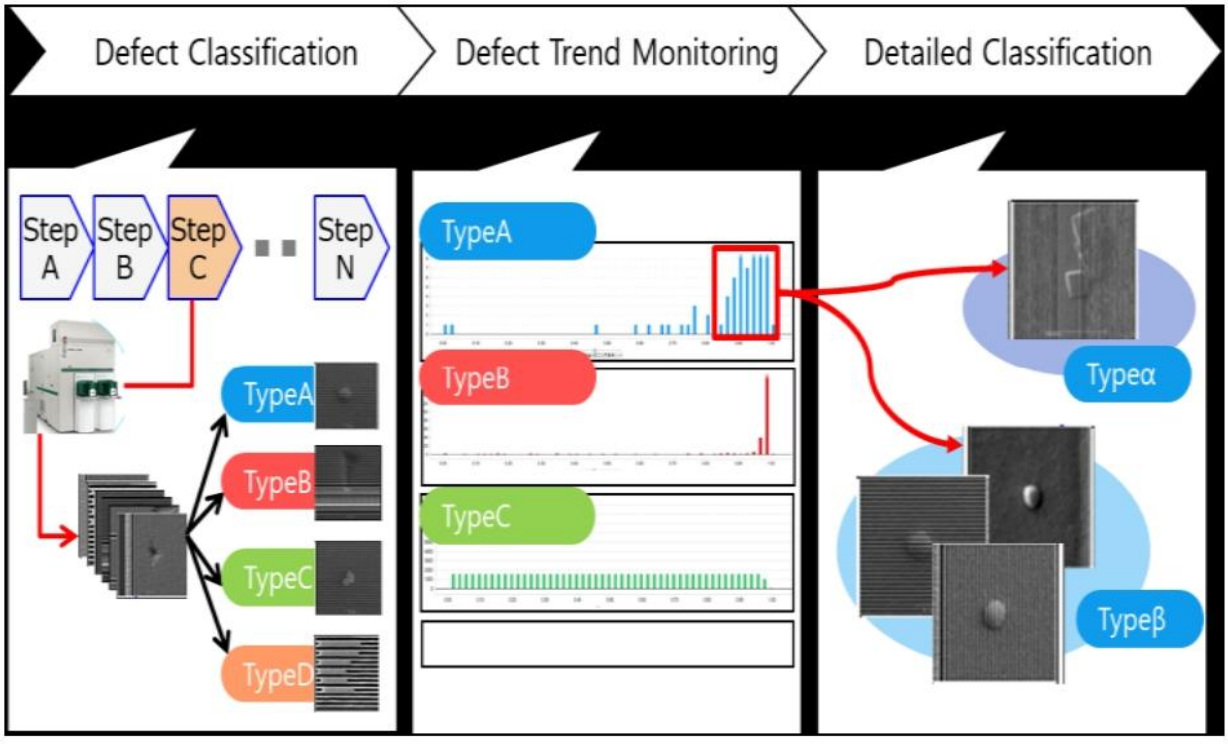
Adopting a combination of pure computer vision approach (non-machine learning methods) to extract the region of interest (ROI) from the original image and a pure deep learning approach to detect defects in the ROI.

First Extract the "Region of Interest (ROI)" with Computer Vision (Non-Machine Learning Methods). Here, we go through multiple processes on the image such as gray scaling, transformations such as eroding, dilating, closing the image, etc., and eventually curve out the ROI from the image based on use case type/product type, etc. The basic idea of erosion is just like soil erosion — it erodes the boundaries of the foreground objects. Dilating is just the opposite of erosion — it increases the size of the foreground object. Normally, in cases like noise removal, erosion is followed by dilation. An opening is just another name for erosion followed by dilation. It is useful in removing noise. Closing is the reverse of opening, with dilation followed by erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object. Gradient transformation is the difference between the dilation and erosion of an image. Overall, these steps help in opening up barely visible cracks/scratches in the original image.

Secondly, detect defects using deep neural networks (deep neural network (CNN)-Based Models) using proven CNN topologies such as Inception Net (aka Google Net), Res Net, and Dense Net:



We would need few thousand unique images labelled as defects and few thousand labelled as good ones. Augmentation is critical to avoid overfitting the training set. We did X random crops and Y rotations (1 original image results in X*Y augmented images). After augmentation we have X*Y thousand defective images and X*Y thousand good images.



b. Please, explain your major questions to the client regarding the process will be & data.

1. Production process, Material & Design details and most importantly, types of defects for the classification
2. Image data of good & bad quality product

Q. 3: A particular linear process has an input (x=5) and output (y=10)

a. Create a model using NumPy to solve for "y" and record your best error.

b. Repeat the above experiment with different hyper parameters like Learning Rate & activation function.

c. Record the various hyperparameters used in experiment and submit the table with data (x | y | y_pred | error (RMSE) | LR | activation used).

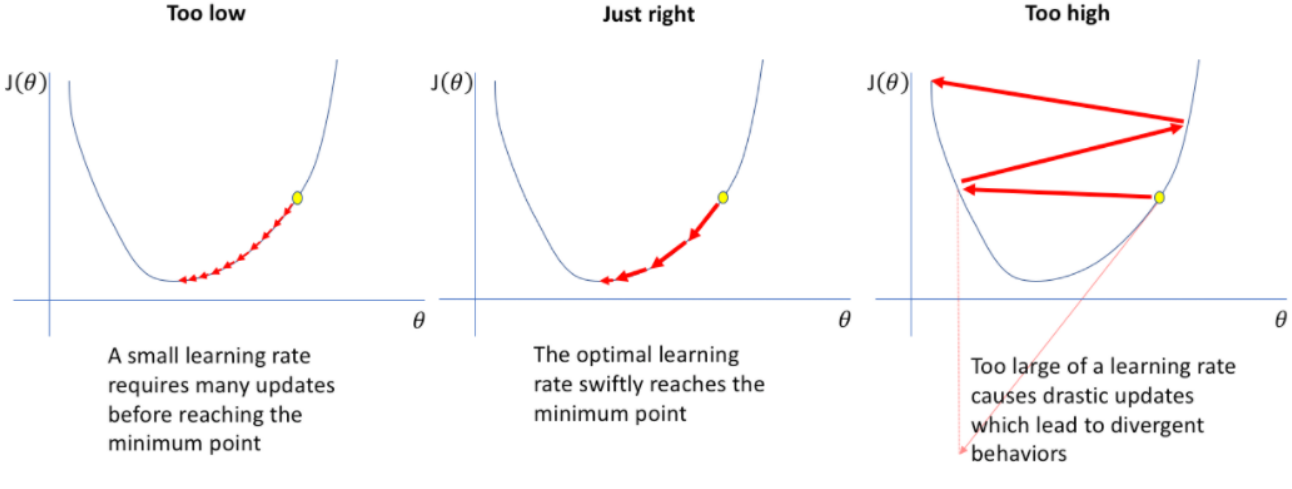
ANSWER: Please refer to the file: *DL_LinearRegressionModel.ipynb in the folder*

Q. 4: Define the role of Learning Rate in Neural Networks and why they are important using an experiment?

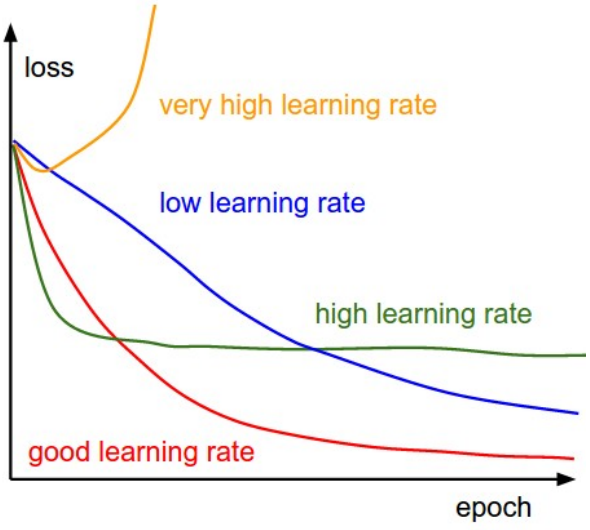
ANSWER:

Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. The lower the value, the slower we travel along the downward slope. While this might be a good idea (using a low learning rate) in terms of making sure that we do not miss any local minima.

$$w^{+} = w - \eta * \frac{\partial Etotal}{\partial w}$$



As such, it's often hard to get it right. The below diagram demonstrates the different scenarios one can fall into when configuring the learning rate.



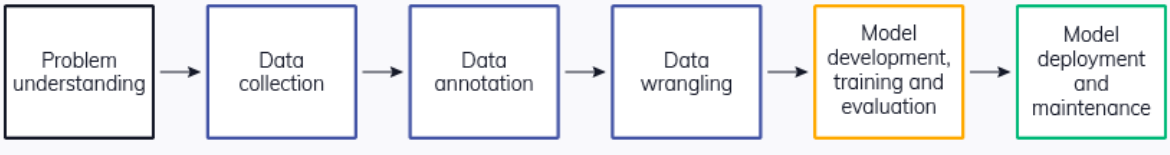
Q. 4_a: Use an experiment like above one with LR and one without LR.

ANSWER: Every optimizer has a default learning rate of 0.001, however, I have explained & visualize the 'make_blob' dataset with different LR ranging (0.01, 1.0).

Refer the LR_Impact.ipynb file in the same folder.

Q. 5: Explain the 3-step life cycle of Deep Learning projects with an example use case.

The life cycle of a machine learning project can be represented as a multi-component flow, where each consecutive step affects the rest of the flow. Let's look at the steps in a flow on a very high level:



1. Problem understanding (aka business understanding).
2. Data collection.
3. Data annotation.
4. Data wrangling.
5. Model development, training and evaluation.
6. Model deployment and maintenance in the production environment.

As you can see, the entire cycle consists of 6 consecutive steps. Each step is unique, with its own nature. These differences lead to variations in resources, time and team members needed to complete each step. Let's have a detailed look at few important components in the life cycle as mentioned below and see what it's all about.

1. Problem understanding & Data collection.
2. Model development, training and evaluation
3. Model deployment and maintenance in the production environment.

Step 1: Problem understanding & Data collection.

Ideally, a clear problem definition should be numerically described. Numbers not only provide an ability to know where your starting point is, but also let you track the effect from the changes later on.

For example, In the organization, the management figured out that if we want to decrease the costs for a given manual operation by 20%, we should decrease the number of manual processing from 100% to at least 70%. This means that 30% of all operations should be processed automatically. Knowing that can help us narrow down the scope for a project, letting us understand that we only need to target a portion of a problem, not the whole problem.

Next, the manual operation we wanted to target was decomposed into pieces. Knowing how much each piece costs in terms of time (and money), the team was able to come up with a list of proposals for the tasks that might be automated.

Discussing this list with the machine learning team, they picked a few tasks that can be solved via supervised machine learning algorithms if proper data is available.

Finally, the problem understanding is complete: each team in the company knows what they're targeting and why. The project can begin.

Data is power. When the problem is clear, and an appropriate machine learning approach is established, it's time to collect data.

Collected data is messy. There are many problems that machine learning engineers face when dealing with raw data. Here are the most common issues needed to be fixed:

- Relevant data should be filtered. Irrelevant data should be cleaned up.
- Noise, delusive and erroneous samples should be identified and removed;
- Outliers should be recognized and eliminated.
- Missing values should be spotted and either removed or imputed by proper methods.
- Data should be converted to proper formats.

Step 5: Model development, training and evaluation

By this step you should have a complete dataset that's ready to be fed into the model. What's next? It's time to make a decision about the future model and assemble it.

Tune the model architecture accordingly, It's important to note that a pre-trained model that we import needs to be modified to reflect the specific task we're doing. It's important to note that a pre-trained model that we import needs to be modified to reflect the specific task we're doing.

Experiment a lot, Experiments are in the nature of what we do. If your computation resources aren't limited, you should definitely take advantage of it. The outcomes that you might get can be quite unexpected. Who knows, maybe a new state-of-the-art model configuration will come from one of your experiments.

Evaluate properly, Evaluation during training is performed on a separate validation dataset. It tracks how good our model is at generalization, avoiding possible bias and overfitting.

It's always good practice to visualize model progress during the training job. Tensorboard is the first and most basic option to consider. Take your time to find an experiment tracking tool that fits your particular needs. You will save a ton of time and improve your overall workflow when you get one.

Step 6: Model deployment

Excellent! Now, engineers deploy a train model and make it available for external inference requests.

Deployed models need monitoring. You need to track deployed model performance, to make sure it continues to do the job with the quality that the business requires. We all know about some negative effect that might happen over time: model degradation is one of the most common ones.

Another good practice would be to collect samples that were wrongly processed by the model to figure out the root cause reasons for why it happened and use it for retraining the model making it more robust to such samples. Such little continuous research will help you better understand possible edge cases and other unexpected occurrences that your current model isn't prepared for.

==========Q 4 End==========

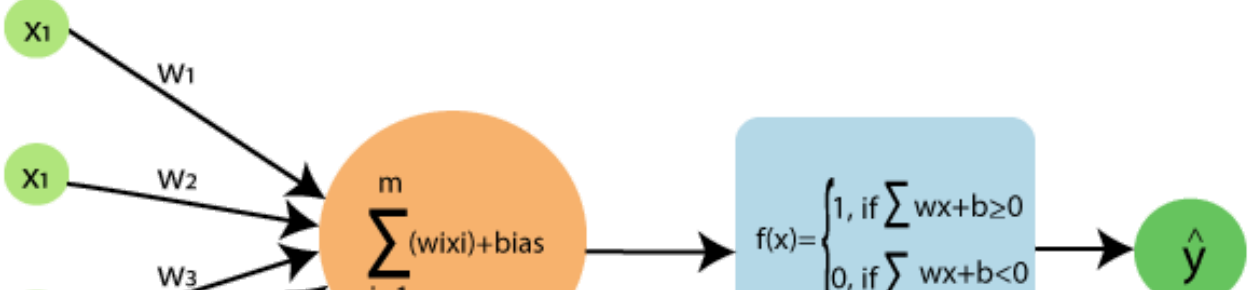
Q. 6: Mathematically, derive the complete process of a single perceptron from an input to output and perform the below task:

a. Explain the above in 3 phases of Feed Forward, Error Calculation, Back Propagation mathematically.
b. Explain the role and importance of each and every parameter other than x and y.

ANSWER:

Single Layer Perceptron in TensorFlow

A perceptron is a neural network unit that does a precise computation to detect features in the input data. Perceptron is mainly used to classify the data into two parts. Therefore, it is also known as **Linear Binary Classifier**.



Q. 6_a. Explain the above in 3 phases of Feed Forward, Error Calculation, Back Propagation mathematically.

```
import numpy as np
import matplotlib.pyplot as plt
x1 = [10, 8, 15]
x2 = [8, 20 , 25]
y = [30, 15, 20]

# Estimate best values of w1, w2, b
```

```
def forward_pass(w1, w2, b):
    y_pred = []
    total_error = 0
    for i in range(n):
        y_hat = w1 * x1[i] + w2 * x2[i] + b
        y_pred.append(y_hat)

        E = (y[i]-y_hat)**2
        total_error += E
    return y_pred, total_error

def weight_update(w1, w2, b, y_pred):
    for i in range(n):
        # alpha = 0.01
        dJ_dw1 = -2*(y[i]-y_pred[i])*x1[i]
        dJ_dw2 = -2*(y[i]-y_pred[i])*x2[i]
        dJ_db = -2*(y[i]-y_pred[i])

        w1 = w1 - alpha * dJ_dw1
        w2 = w2 - alpha * dJ_dw2
        b = b - alpha * dJ_db
    return w1, w2, b

#CONSTANTS
iterations = 100
n = len(x1)
alpha = 0.0001

#INIT of WEIGHTS
random_init = np.random.randn(3)
w1, w2, b = random_init[0], random_init[1], random_init[2]
```

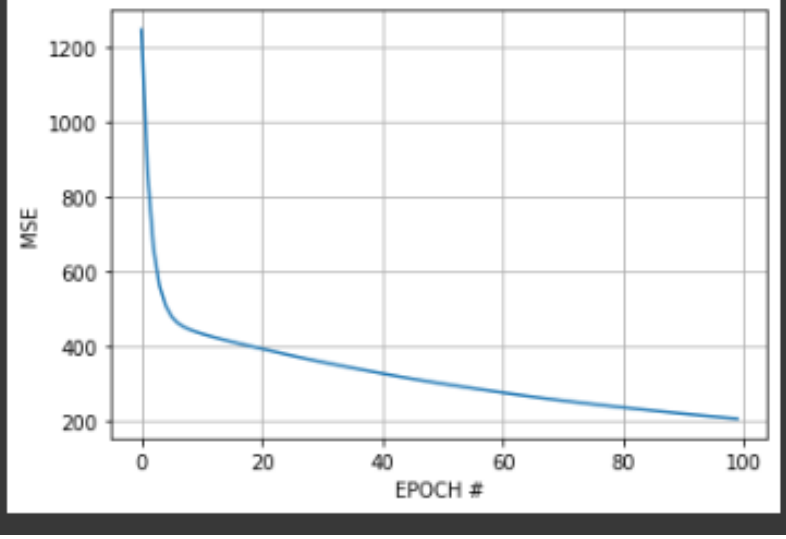
```
print(w1, w2, b)

# MODEL
error = []
for iter_num in range(iterations):
    y_pred, total_error = forward_pass(w1, w2, b)
    print(f"Epoch #: {iter_num} | MSE = {round(total_error,2)}")
    error.append(total_error)
    w1, w2, b = weight_update(w1, w2, b, y_pred)

print("Result:",[w1, w2, b])
```

```
-0.27904128228462466 0.26521819721651096 1.218518843806807
Epoch #: 0 | MSE = 1248.89
Epoch #: 1 | MSE = 861.51
Epoch #: 2 | MSE = 663.84
Epoch #: 3 | MSE = 561.87
Epoch #: 4 | MSE = 508.19
Epoch #: 5 | MSE = 478.9
Epoch #: 6 | MSE = 461.92
Epoch #: 7 | MSE = 451.19
Epoch #: 8 | MSE = 443.64
Epoch #: 9 | MSE = 437.71
Epoch #: 10 | MSE = 432.63
Epoch #: 11 | MSE = 428.0
Epoch #: 12 | MSE = 423.63
Epoch #: 13 | MSE = 419.41
Epoch #: 14 | MSE = 415.3
Epoch #: 15 | MSE = 411.26
Epoch #: 16 | MSE = 407.29
Epoch #: 17 | MSE = 403.37
Epoch #: 18 | MSE = 399.5
Epoch #: 19 | MSE = 395.69
Epoch #: 20 | MSE = 391.92
```

```
plt.plot(error)
plt.xlabel("EPOCH #")
plt.ylabel("MSE")
plt.grid()
plt.show()
```



Q. 6_b. Explain the role and importance of each and every parameter other than x and y.

- **Input value or One input layer:** The input layer of the perceptron is made of artificial input neurons and takes the initial data into the system for further processing.
- **Hidden layer:** Input and output layers get separated by hidden layers. Depending on the type of model, there may be several hidden layers. There are several neurons in hidden layers that transform the input before actually transferring it to the next layer. This network gets constantly updated with weights in order to make it easier to predict.
- **Weights and Bias:**
  Weight: It represents the dimension or strength of the connection between units. If the weight to node 1 to node 2 has a higher quantity, then neuron 1 has a more considerable influence on the neuron.
  Bias: It is the same as the intercept added in a linear equation. It is an additional parameter which task is to modify the output along with the weighted sum of the input to the other neuron.
- **Net sum:** It calculates the total sum.
- **Activation Function:** A neuron can be activated or not, is determined by an activation function. The activation function calculates a weighted sum and further adding bias with it to give the result.

1. What are activation functions and why are they required in NN?

b. Create python functions for sigmoid, tanh, relu & softmax.

Q. 7: What are activation functions and why are they required in NN?

a. Create python functions for sigmoid, tanh, relu & softmax.

ANSWER:

In the process of building a neural network, one of the choices you get to make is what Activation Function to use in the hidden layer as well as at the output layer of the network.

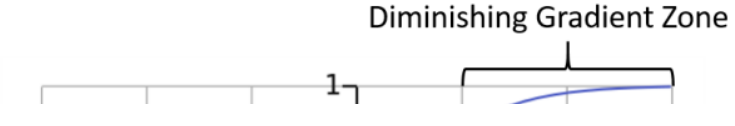
Input Layer: This layer accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.

Hidden Layer: Nodes of this layer are not exposed to the outer world; they are part of the abstraction provided by any neural network. The hidden layer performs all sorts of computation on the features entered through the input layer and transfers the result to the output layer.

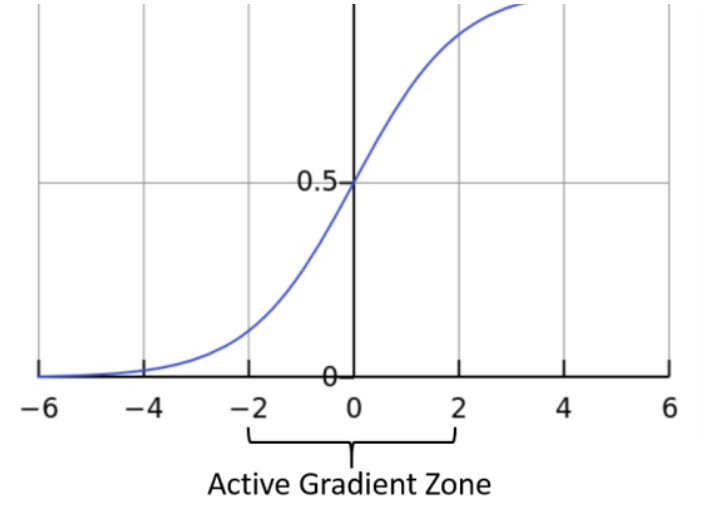
Output Layer: This layer brings up the information learned by the network to the outer world.

Variants of Activation Function

- **Sigmoid Function**



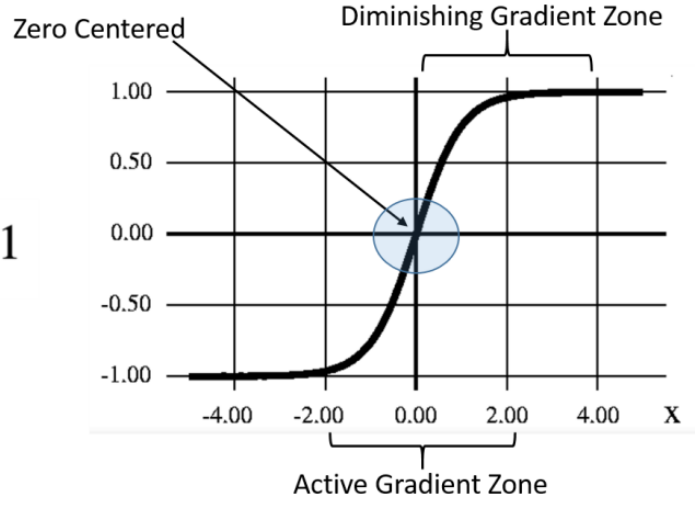
$$A = \frac{1}{1+e^{-x}}$$



1. It is a function which is plotted as 'S' shaped graph.
2. **Equation:** A = 1/(1 + e-x)
3. **Nature:** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
4. Value Range: 0 to 1
5. **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.

- **Tanh Function**

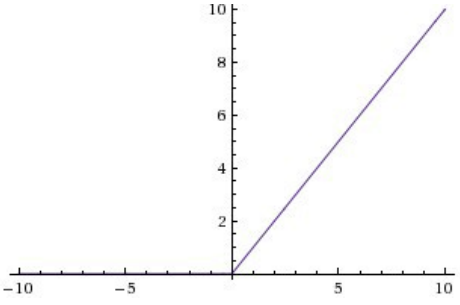
$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



1. 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.
2. **Value Range:** -1 to +1
3. **Nature:** non-linear

4. **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 Zero. This makes learning for the next layer much easier.

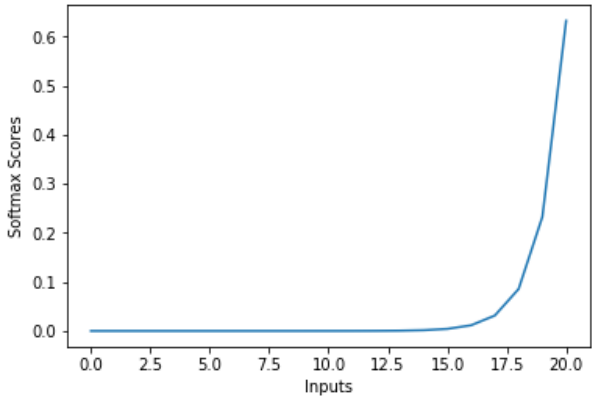
- **RELU Function**



1. It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.
2. **Equation:** *A(x) = max(0, x)*. It gives an output x if x is positive and 0 otherwise.
3. Value Range [0, inf)
4. **Nature:** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
5. **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.

- **Softmax Function**



- **Equation:**

$$softmax(Z_i) = \frac{\exp(Z_i)}{\sum \exp(Z_i)}$$

- **Nature:** non-linear
- **Uses:** Usually used when trying to handle multiple classes. the softmax function was commonly found in the output layer of image classification problems. 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.
- 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 in hidden layers 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.
- If your output is for multi-class classification then, Softmax is very useful to predict the probabilities of each classes.

Q. 7_a: Create python functions for sigmoid, tanh, relu & softmax.

ANSWER: Refer the 'Actvation_Function_python_Implmentation.ipynb' file for the notebook file.

```python
import numpy as np

# Sigmoid Function
def sigmoid(a):
  return 1/(1 + np.exp(-a))

# tanh Fuction
def tanh(a):
  t=(np.exp(a)-np.exp(-a))/(np.exp(a)+np.exp(-a))
  dt=1-t**2
  return t,dt

# ReLU Function
def relu(a):
  data = [max(0,value) for value in a]
  return np.array(data, dtype=float)

# softmax Function
def softmax(a):
 e = np.exp(a)
 return e / e.sum()
```