Introduction to IoT and Its Industrial Applications (CS698T)
Indian Institute of Technology Kanpur
Assignment 2

Member Names: Pranshu Sahijwani, Shivam Tripathi
Member Emails: pranshus21@iitk.ac.in, sivamtr21@iitk.ac.in
Member Roll Numbers: 21111048, 21111408
Date: November 3, 2021

---

1 Building a Smart Irrigation System - Approach

We are supposed to build four independent units of irrigation system all of which will contain two sensors namely Servo motor and DHT22. The control flow of the project is :

1. We read the information of the LDR sensor which is utilized for telling the iridescent force falling upon it , we have set the limit worth to 50% , if lux power is underneath Threshold ,it will predicts Night and won't play out any activity and on the off chance that lux intenstiy is more noteworthy than 50%, it will show Day time and will begin the further activity.

2. If Day time , the DHT22 sensor starts sensing the amount of temperature and humidity in the surroundings using the in-built function ***dht[i].readTemperature() and dht[i].readHumidity()*** from the library ***#include"DHT.h"***

3. Then, at that point, this information is taken care of to the prepared ML model()()() which returns the waterflow that should be provided to the ranch's servo engines. The measure of waterflow is returned in rate that we are showing on the 2x16 LCD, then we have converted this percentage into degrees using the following formula:

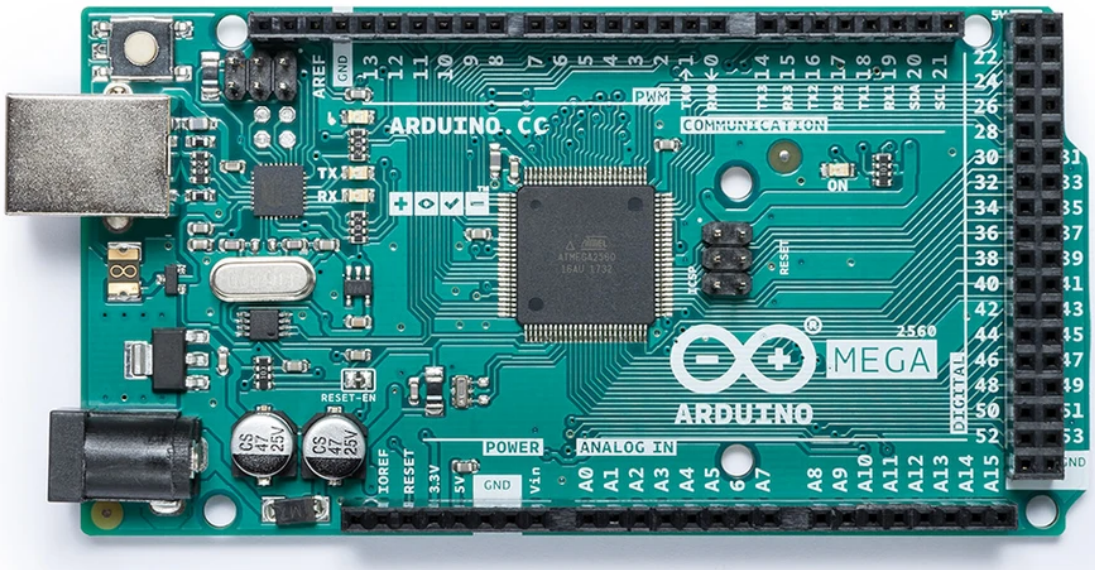
$$Servo\ angle = \frac{water\ percentage * 180}{100}$$

4. Each of the four units are independent, so changing the sensor values of Dht22 for a particular servo motor will affect the waterflow rate, **only**, of that particular servo motor.

2 Components

2.1 Arduino-Mega Board [3]

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.



2.2 Sensors [4]

(a) Photo resistor (LDR)

The resistance of a photoresistor decreases with increase in incident l intensity; in other words, it exhibits photoconductivity. The sensor has 4 pins namely VCC, GND, DO and AO. It can give both digital and analog reading depending on which pin among DO and AO is used for recording the readings.

(b) DHT22

The DHT22 is a basic digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin, no analog input pins needed. It has a VCC pin(5v positive pin), SDA pin(signal line goes to a digital pin), NC(not connected) and GND(ground -ve line).

(c) Servo motors

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors. It has 3 pins – ground pin, v+ pin and PWM pin(signal pin). We can turn its arm by 180, this can be done in a looping manner or just once.

(d) LCD (20*4) I2C

This is a 20x4 LCD display screen with I2C interface. It is able to display 20*4 characters i.e. 20 characters on 4 lines (including white characters). VCC, GND, SDA, SCL. SDA is the serial data pin and SCL is the clock pin.

3 Evaluation Metrics [2]

(a) **Coefficient of Determination(R^2) regression score function**

Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a score of 0.0.

If $\bar{y}$ is the mean of the observed data:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

The sum of squares of residuals, also called the residual sum of squares:

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2\}$$

The total sum of squares (proportional to the variance of the data):

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2\},$$

The most general definition of the coefficient of determination is

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}\}$$

Our R^2 Score is **0.870758881631629**

(b) **Root Mean Squared Error(RMSE)**

$$\sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(\frac{d_i - f_i}{\sigma_i}\Big)^2}$$

where, d_i are data values, f_i is the mean, σ_i is the standard deviation, n is the total number of observations

Our RMSE(Root Mean Squared Error) is **11.594032609946861**

(c) **MAPE(Mean Absolute Percentage Error)**

$$\frac{100}{n} \times \sum_{t=1}^{n} \left|\frac{A_t - F_t}{A_t}\right|$$

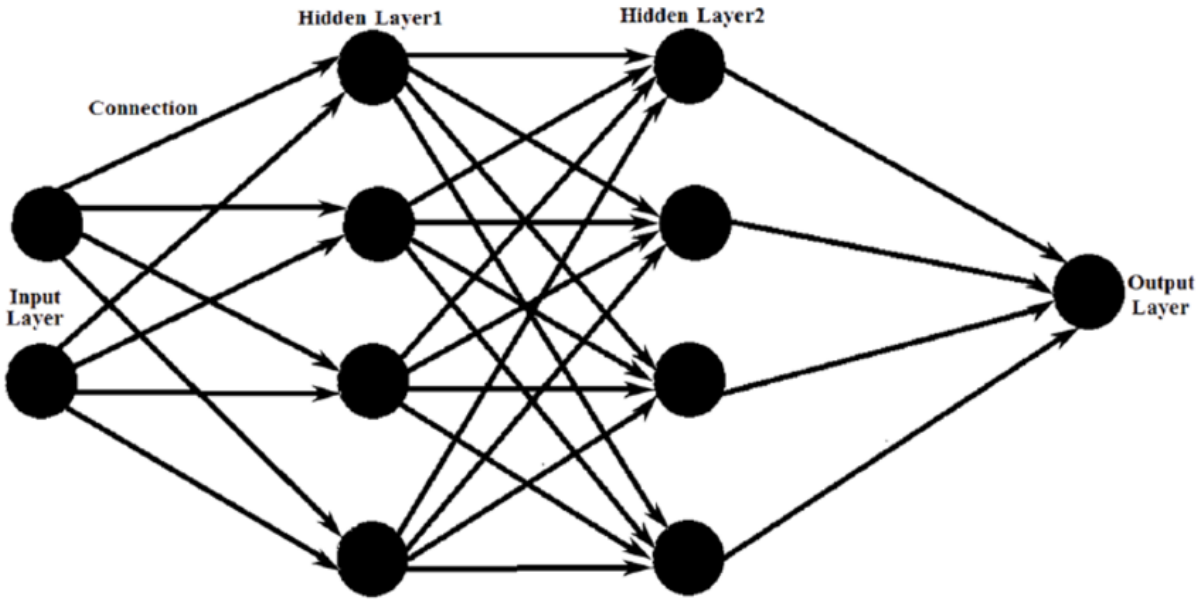
where At is the actual value and Ft is the forecast value. Their difference is divided by the actual value At. The absolute value in this ratio is summed for every forecasted point in time and divided by the number of fitted points n.

Our Mean Absolute Percentage Error is **0.7489201061953561**

4 Architecture

We have used a two-layer perceptron machine learning model [5] (i.e. input layer, 2 hidden layers, and output layer) to predict the servo motor's signal. The activation function that we have used is ReLu(Rectified Linear Unit), with total iteration as 10000 and the number of neurons in the first layer as 16 and in the second layer as 8.

4.1 Diagram



5 ML Model Parameters

(a) **Information on Epochs**: We have used 10000 epochs in the form of argument max_iter = 10000 as it gives optimal accuracy. this was giving us a better values for our evaluation metrics

(b) **Hidden Layers**: We have used 1 input layer + 2 hidden layer + 1 output layer as mentioned in the question. Hidden Layer Sizes : The hidden layer sizes denotes the number of neurons present in each hidden layer. The size of the first layer is **16 neurons** followed by **8** for the next. We have taken powers of 2 as this works best for the CPU and GPU of a system. These sizes are also the result of continuous testing.

(c) **Seed Information**: Random state is provided so that the output of an experiment can be reproduced anytime for validation purposes. Providing the same random number will yield the same output. It is provided in the form of argument random_state = 1.

(d) **ReLu Activation function**: The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. The formula and image below show a simple Relu function. [1]

$$relu(x) = max(0, x)$$

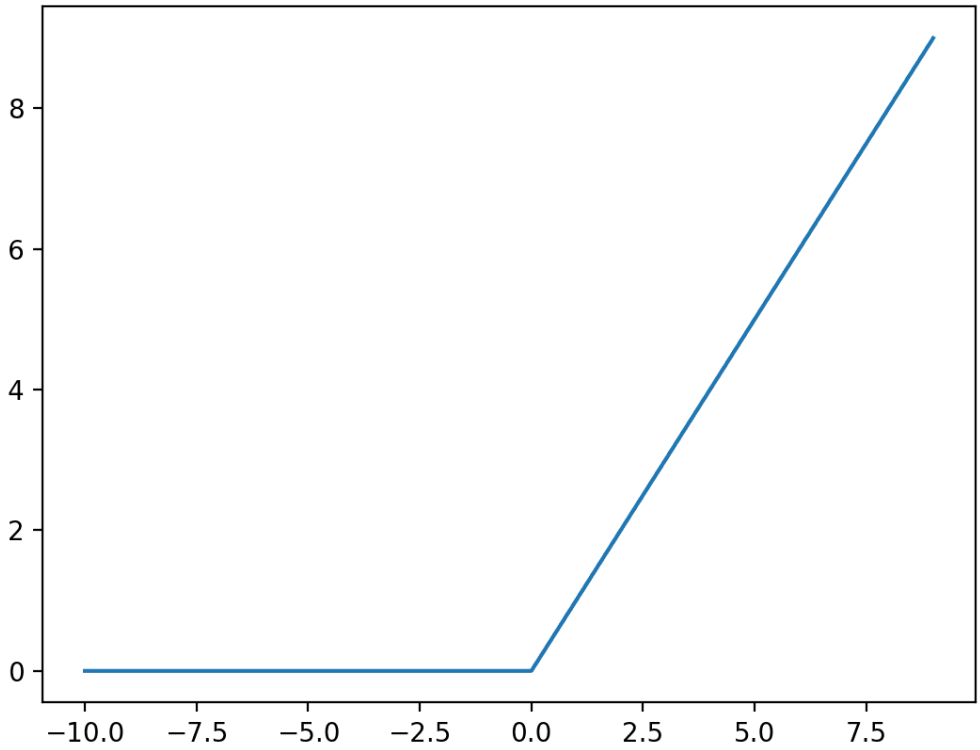


Figure 1: Rectified-Linear Unit Activation Function

6 ML Model - Training and Hyper Parameter Tuning

6.1 Code - Python

```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
import pandas as pd
import math

#reading the given data file
df = pd.read_csv('IOT_Assignment_2_data_regression_sensor_range.csv')

#column renaming to more logical names
df.columns = ['humidity_perc', 'temp_c', 'waterflow_perc']

#min-max normalisation
max_h = df['humidity_perc'].max()
max_t = df['temp_c'].max()
```

```
min_h = df['humidity_perc'].min()
min_t = df['temp_c'].min()

df['humidity_perc'] = df['humidity_perc'].apply(lambda x:(x-min_h)/(max_h-min_h) )
df['temp_c'] = df['temp_c'].apply(lambda x: (x-min_t)/(max_t-min_t))

#splitiing the dataframe into inputs and the output
X = df[['humidity_perc', 'temp_c']]
y = df.waterflow_perc

#splitiing the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=1,
    test_size=0.25)

# training the MLP model with the following params
# Hidden layer 1: 16 neurons
# Hidden layer 2: 8 neurons
# maximum iterations: 10000
reg = MLPRegressor(hidden_layer_sizes=(16,8),activation="relu" ,random_state=1,
    max_iter=10000).fit(X_train, y_train)

y_pred=reg.predict(X_test) # prediction using the trained model

#Evaluation metrics
print("The R2 Score with ", (r2_score(y_pred, y_test)))
print("The RMSE with ", (math.sqrt(mean_squared_error(y_pred, y_test))))
print("The mean_absolute_percentage_error with ",
    (mean_absolute_percentage_error(y_pred, y_test)))

#final weights and biases
print('The weights are', reg.coefs_)
print('The biases are', reg.intercepts_)
```

6.2 Obtained Weights and Biases

```
The weights are:
       1. Size: 2*16
       [array([[-5.59418921e-001, 5.68105385e+000, 3.46751235e-001,
        1.86789661e-191, -1.98833942e+000, -4.46480581e-001,
       -9.32164878e-169, -1.34262509e-001, 1.33698637e-001,
        2.01027369e+000, -1.11078127e-217, 7.07340485e+000,
        8.41761117e-172, 2.03528446e+000, 4.20290993e-001,
       -1.63419267e-197],
      [ 1.20408795e+000, 1.45136443e+000, 9.76418092e-001,
       -1.12745754e-170, 1.91922139e+000, 1.45782895e+000,
       -6.42633031e-194, 9.59700165e-001, 1.11916115e+000,
        1.57075215e+000, -2.94658617e-152, -3.34198642e+000,
        9.66679560e-166, 1.31832568e+000, -2.06059725e+000,
        1.75487135e-218]]

       2. Size: 16*8
        array([[-2.54884246e-001, 3.74398549e-169, -1.66044117e-152,
       -3.47973049e-002, -4.72585411e-001, 8.10771533e-001,
       -1.01561042e-001, -1.64428059e-156],
```

```
[-2.14019684e-001, 6.15562924e-170, -2.15918750e-218,
  7.47343693e+000, 7.10978591e+000, 6.05545211e+000,
  1.06349417e-001, 3.57778216e-219],
[-2.74705701e+000, 3.42393478e-226, 5.54175040e-204,
 -1.69943972e-002, 2.60638831e-001, 1.17596739e+000,
  2.17795792e-001, -8.33412683e-169],
[-9.18990343e-169, -2.50811767e-177, -3.48223846e-216,
 -6.53259304e-173, 3.29583642e-159, 5.99732666e-207,
  3.05749116e-187, 3.56764791e-192],
[ 1.61973781e+000, -4.23656131e-212, 3.29629181e-187,
 -1.78189037e+000, -1.78890484e+000, 2.52172851e+000,
 -1.77703652e-001, -4.75974455e-156],
[-4.24856004e-001, -1.01683236e-212, -3.70179026e-165,
  2.50047754e-001, -1.27643010e-001, 1.12509924e+000,
 -1.92286425e-001, -2.98275879e-186],
[ 5.94400978e-163, 1.34502227e-222, -3.58525783e-150,
 -3.93624455e-213, 3.55631809e-202, 3.06899610e-227,
  5.56907627e-165, -1.09148978e-208],
[-4.91763770e-001, -5.08826590e-212, 1.53023443e-151,
  5.13347576e-001, 4.28724467e-001, 1.30351001e+000,
 -4.34280951e-001, -6.35216117e-169],
[-3.30331943e-001, -1.63414142e-197, 1.00281498e-158,
  5.99470244e-001, 7.07202364e-001, 9.97026299e-001,
  1.09036296e-001, -7.04991970e-167],
[-3.23257913e+000, -2.36039185e-153, 2.10520988e-153,
 -5.95411026e-002, 9.79315264e-001, 3.81577910e-001,
 -5.26470724e-002, -1.11589781e-171],
[-6.50135738e-167, 7.36841267e-193, -4.05177814e-219,
 -2.24775672e-153, -7.77105078e-223, -1.33274802e-154,
  1.66105971e-178, 1.48105983e-184],
[-2.16330902e+000, 3.16101271e-214, -1.94788532e-170,
 -4.54832129e+000, -4.04888313e+000, -7.02168421e+000,
 -4.82127303e-001, 4.04619449e-166],
[ 1.58552600e-155, 4.85288551e-164, 4.08484233e-183,
 -1.88314547e-194, -3.65370072e-223, 3.54753242e-151,
 -4.81479936e-159, -9.43019837e-154],
[-3.21399022e+000, 3.79858738e-179, 6.24552015e-188,
  4.74998659e-001, 5.91226416e-001, 4.26979011e-001,
  3.62862124e-001, 5.72903947e-171],
[ 2.86625633e+000, 3.00939395e-229, 1.35604237e-212,
  1.67395024e+000, -5.28749140e-002, -2.75650442e+000,
 -4.95416402e-001, -1.84072673e-228],
[-3.32463256e-215, -4.63799837e-223, 1.11583830e-200,
 -2.82076252e-151, -8.10520464e-220, -1.29639822e-212,
 -4.09655966e-224, -2.02622064e-188]]),

 3. Size: 8*1
 array([[-4.32607544e+000],
[ 2.59915637e-150],
[ 1.44939016e-149],
[ 4.93008226e+000],
[ 4.60747737e+000],
[ 5.88102834e+000],
[-3.23551018e-001],
[ 9.76717628e-129]])]
```

```
The biases are:
       1. Size: 16
       [array([ 0.40546025, -4.42774979, 0.08486776, -0.2130242 , 0.53810184,
       0.56266578, -0.55623279, 0.540329 , 0.94469664, -0.05598933,
      -0.25352144, -0.28231778, -0.45815514, -0.13210869, 2.71258716,
      -0.23831385]),

       2. Size: 8
       array([ 1.19471381, -0.23508044, -0.43366517, 1.19420209, 0.74977472,
      -0.22586172, 0.15849943, -0.43346352]),

       3. Size: 1
       array([1.28557373])]
```

6.3 Instructions to train the ML Model and print Weights and Biases

1. Install the following python packages:
   (a) sklearn
   (b) pandas
   (c) math
2. Extract the zip file
3. Run the python file provided

NOTE: We have used Python version 3.8.8 to execute the training of MLP model. Please make sure you use the same or compatible Python version.

7 Results and Overall Summary

We have successfully built our smart irrigation system. The water percentage to be spilled using servo motors is calculated using forward propagation on the multi-layer perceptron model. This percentage value is mapped to appropriate angle value between 0-180 by scaling.

Since we know the land requires large amount of water when the temperature outside is high and humidity is low. We can clearly verify this by testing our model and passing the appropriate temperature and humidity values.

The illumination of area can be recorded using the LDR (Light Dependent Resistor (LDR). We have set a threshold of 50 lux to differentiate between the day and night time. The irrigation system works only during the day time, during the night time we turn off all the servos and don't irrigate the area.

	Temperature (° Celsius)	Humidity (%)	Waterflow (%)
Servo 1	80.0 °C	8.5%	67%
Servo 2	21.2 °C	10.5%	0%
Servo 3	39.5 °C	15.5%	19%
Servo 4	59.9 °C	53.0%	1%

Table: Example values of temperature, humidity and prediction of water percentage

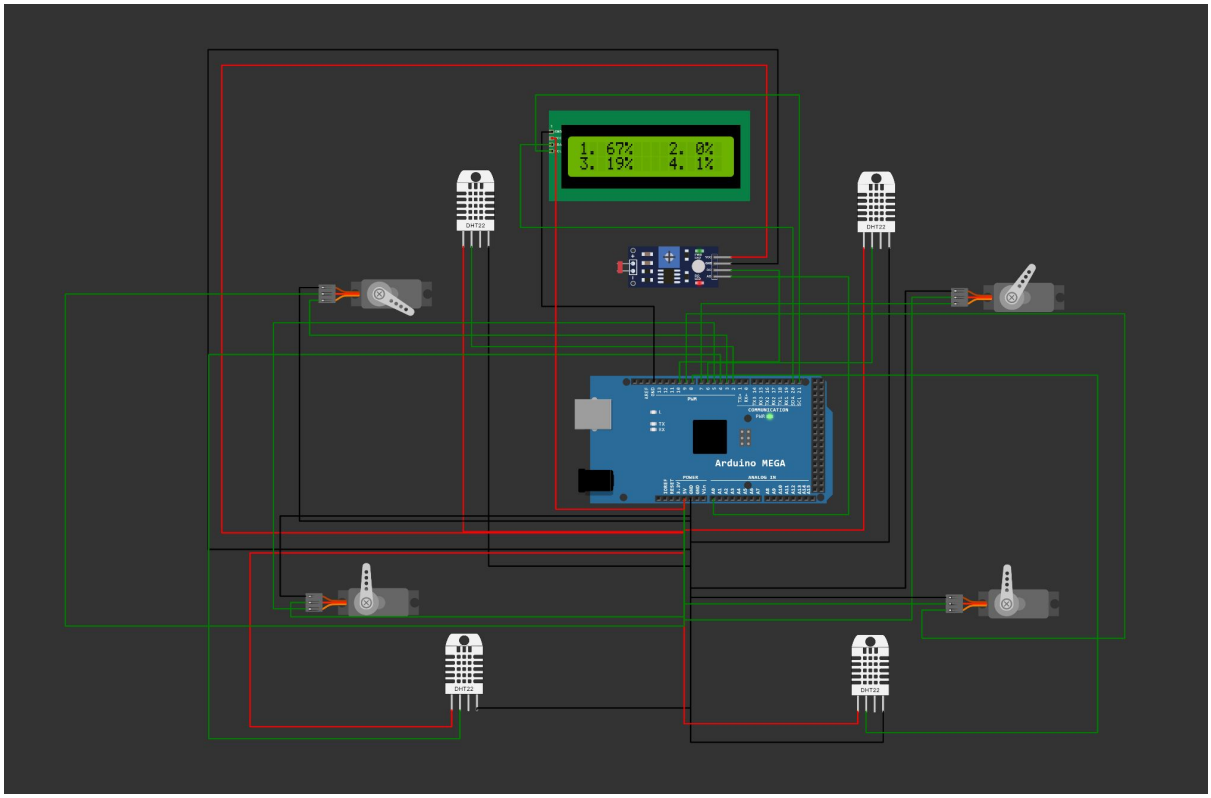


Figure 2: Example of Simulator Showing the Servos Irrigating the Area based on the Calculated Water Percentage (displayed on the LCD)

8 Links

- Video Demonstration of Smart Irrigation System YouTube
- Link to the Simulated Project WokWi

References

[1] *A Gentle Introduction to the Rectified Linear Unit (ReLU)* https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural

[2] *Scikit-learn Wikipedia* https://en.wikipedia.org/wiki/Scikit-learn

[3] *Arduino Mega 2560* https://store.arduino.cc/products/arduino-mega-2560-rev3

[4] *Wokwi Documentation* https://docs.wokwi.com/

[5] *Multilayer perceptron and neural networks*https://www.researchgate.net/publication/228340819_Multilayer_perceptron_and_neural_networks