MECENG 249 Project 2 Report

Understanding the Principles and types of Neural Network Model

Submitted By:

Joshua Duarte
Pavan M. Reddy

Date: 10-20-2022

Table of Contents

I. Introduction

A simplified yet intelligent model that teaches computers to process the data like a human brain is coined as a neural network. There are three types of neural networks namely, Artificial Neural Networks (ANN) Convolution Neural Networks (CNN), and Recurrent Neural Networks (RNN). In this project, the implementation and understanding of a simple artificial neural network is conducted.

An Artificial neural network is a model with connected input/output layers and hidden layers with neurons to simulate a human brain. These models are widely used in the energy sector to evaluate systems where physical prototyping and evaluation for various operating conditions are not feasible.

In the first part of the project, two models are created and trained using two different methods, the first principles and Keras. The First-principle method also known as the gradient method follows the backpropagation principle. It is the essence of neural network training where the weights are fine-tuned based on the error rate obtained in the previous epoch. Keras is a model package integrated with TensorFlow which is used for training and creating neural networks.

In the second part, the learnings from the tasks are used to analyze a working hybrid solar fossil-fuel turbine system. A system consisting of three heat sources namely, a regenerator, solar input, and a burner is considered. For achieving optimal efficiency while maintaining the safe operating limits of the system, the usage of machine learning algorithms is implemented.

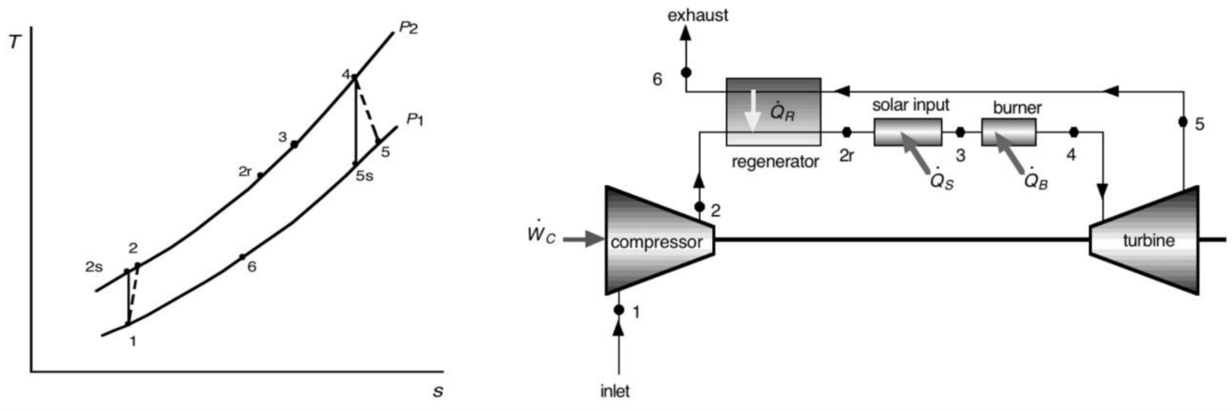

Figure 1: Hybrid solar fossil-fuel turbine system

The energy system used is shown above. In an overview, atmospheric air(T_1) is compressed in the compressor, and then the pressurized air(T_2) is heated in the regenerator to a temperature of (T_{2r}). Next, this air is heated to the next step using a solar input system to temperature(T_3). Then this air is passed through a burner where the air-fuel mixture is burnt and the heat generated by combustion is used to raise the temperature to the final temperature(T_4) before entering the turbine.

While in any turbine system thermodynamically it’s known the higher the input temperature to the turbine higher the efficiency. So the objective in the above system is to achieve the highest temperature of T_4. But the turbine blade material has a thermal limitation which has to be considered. If the temperature goes higher than the threshold the turbine parts might get damaged. Hence an optimal temperature of T_4 is to be generated.

The burner uses a mixture of methane and propane with air to help in combustion. To achieve what is discussed above we need to understand the variation of the two key parameters molar air-to-fuel ratio α and the fuel propane mole fraction γ. For understanding this we need to develop a model to predict these values for varying input conditions.

There are two ways the model can be constructed, one way is to have a physical model based on the principles of thermodynamics, fluid dynamics, and heat transfer i.e, the first principles or to develop a machine learning algorithm to take in a set of operating conditions to predict the ratios.

Considering all the pros and cons of both processes, it is concluded that the use of a machine learning model helps to visualize the data and predict values close to the real system as it accounts for multivariate data. Hence we develop a Keras model to train a set of training data initially available. Then the model is used to predict the parameters for two different sets of data. It is then used to understand the sensitivity of the model for various initial inputs.

The neural network model is constructed using three input variables, atmospheric temperature(T_1), methane to fuel mixture ratio(γ), and solar heat input(Q_s). It predicts two outputs, the air-to-fuel mixture ratio(α) and the system efficiency(η_{sys}). This baseline neural network is then evaluated for a various number of hidden layers, a number of neurons, and different activation functions.

II. Nomenclature

w	Weights assigned.

b	Bias assigned.
$\underline{E}$	Average Squared Error.
P_1, P_4	pressures.
T_1, T_4	temperatures.
η_{sys}	System Efficiency.
η_c	the compressor isentropic efficiency.
η_t	the turbine isentropic efficiency.
ε_r	the regenerator effectiveness.
m_{air}	the mass flow rate of air into the compressor.
Q_s	the rate of solar thermal heat input.
α	(moles of air)/(moles of propane and methane mixture) in the burner.
γ	(moles of propane)/(moles of propane and methane in fuel mixture) in the burner.

Code variable	**Write-up variable**
w01	w_{01}
w01n	$w_{01,n}$
b1	b_1
b1n	$b_{1,n}$
E3ti	E_3 batch total squared error
E3	E_3 batch average squared error
dE3dw01ti	sum of $\partial E_3 / \partial w_{01}$ for batch
dE3dw01	batch average $\partial E_3 / \partial w_{01}$
gamma	γ

III. Part One

A.

The task is oriented toward understanding the implementation of the backpropagation equations. Provided code was completed using the equations within the red boxes of the project description document. The total batch squared cutout limit was then converted from 0.001 to 0.00035. The script ran until a batch-squared error of less than 0.00035 was obtained. To achieve the best fit, hyperparameters gamma and number of epochs keeping the initial values fixed were adjusted. The results achieved by this task provide the best fit using the first-principles model. This task is performed individually among the team members to obtain the results and has been documented.

Member 1: Pavan M. Reddy

For achieving the best fit the learning rate gamma value was changed with various epoch values. Initially, for low gamma value, the epoch was increased to a high number and then gradually reduced to achieve the best fit with small incremental steps. After iterations, the best fit was achieved with a batch-squared error of 0.0002866 with the learning rate initially kept at 0.165 and then after lower error reduced to 0.08 with 200 epochs.

Member 2: Joshua Duarte

The primary factor in reducing the error was in increasing the number of epochs. This was done by lowering the bat he squared cutout limit to 0.00034 from 0.00035. This minute change with a slight change in the gamma to 0.07, created an RMS value of 0.018.

B.

With the first-principles model viewed, we next move to find the results using the Keras neural network model. We analyze the same data set as the first-principles model with the data normalized the same way. The neural network layers use the same activation functions as the first-principle model, and the initial values of the weights and biases are assigned to be the same as the first-principle model.

I. A simple load and run task is performed in this section. The assigned code is run to achieve a loss value below 0.03. Initially, cell 1,2,3 are run sequentially and the epoch is set to 400 on cell 4 where the cell is run multiple times to attain a loss of less than 0.014. After reaching a low loss we vary the learning rate to further reduce the loss achieved on running cell 4.

II. Once the lowest loss is achieved, a separate cell is run to analyze the results and the data is used to fill the Keras section of the table below.

	First principles	Keras
w01	1.1630778885373607	1.2245948
w02	0.41545711694896326	0.30143216
w03	0.7012193472321739	0.72449493
b1	-0.1475266383815076	-0.13787363
w12	0.699236459210893	0.7087467
b2	-0.11887959054981592	-0.10063175
w23	0.7091125013528717	0.68293154
b3	0.010512651193809102	0.03686729
	rms=$\sqrt{.00028667222349910103}$ =0.0169314	mae= loss=0.01367854326963424

Table 1: Pavan Reddy First Principles and Keras Weights and Bias

x01	x02	x03	y3 data	First principles predicted y3	Keras predicted y3
20.0	13.0	310.8	30.97	31.090793	31.5234
20.0	14.5	308.0	32.3	31.678835	31.909126
20.0	15.3	306.0	31.5	31.973926	32.09619
20.2	13.0	310.8	30.91	31.275793	31.713541
20.0	14.5	308.0	32.5	31.678834	31.909126
20.0	15.3	306.0	31.4	31.973926	32.286304
24.0	13.0	310.8	35.59	34.790793	35.326298
36.0	14.5	308.0	46.4	46.478832	47.1207

Table 2: Pavan Reddy First Principles and Keras Predicted Values Comparison

	First principles	Keras

w01	1.216357733025874	1.2239491
w02	0.4125302702148731	0.2895847
w03	0.6884058318375348	0.7234253
b1	-0.16095176378660467	-0.14015044
w12	0.714042336036248	0.70647335
b2	-0.127856005074306	-0.10168917
w23	0.6939304618676472	0.6799924
b3	0.004522231198836544	0.03707751
	rms=0.018035753115350762	mae=loss=0.01374276727437973

Table 3: Joshua Duarte First Principles and Keras Weights and Bias

x01	x02	x03	y3 data	First principles predicted y3	Keras predicted y3
20.0	13.0	310.8	30.97	31.111950636608388	31.044891
20.0	14.5	308.0	32.3	31.696596380009723	31.408806
20.0	15.3	306.0	31.5	31.990227141614145	31.584368
20.2	13.0	310.8	30.91	31.305292042148874	31.233511
20.0	14.5	308.0	32.5	31.696596380009723	31.408806
20.0	15.3	306.0	31.4	31.990227141614145	31.772987
24.0	13.0	310.8	35.59	34.97877874741809	34.817287
36.0	14.5	308.0	46.4	47.163908823248555	46.498375

Table 4: Joshua Duarte First Principles and Keras Predicted Values Comparison

III. Using the predicted values from each model, plots were created against the initial data on a logarithmic plot using NumPy and matplotlib. The scattered plot provides data to analyze the deviation of the predicted data from the initial data available. The plots for each model are shown below. The scattered plots provide validation for the variation that is seen in the predicted data from the two models for the same initial values assigned to the weights and biases.

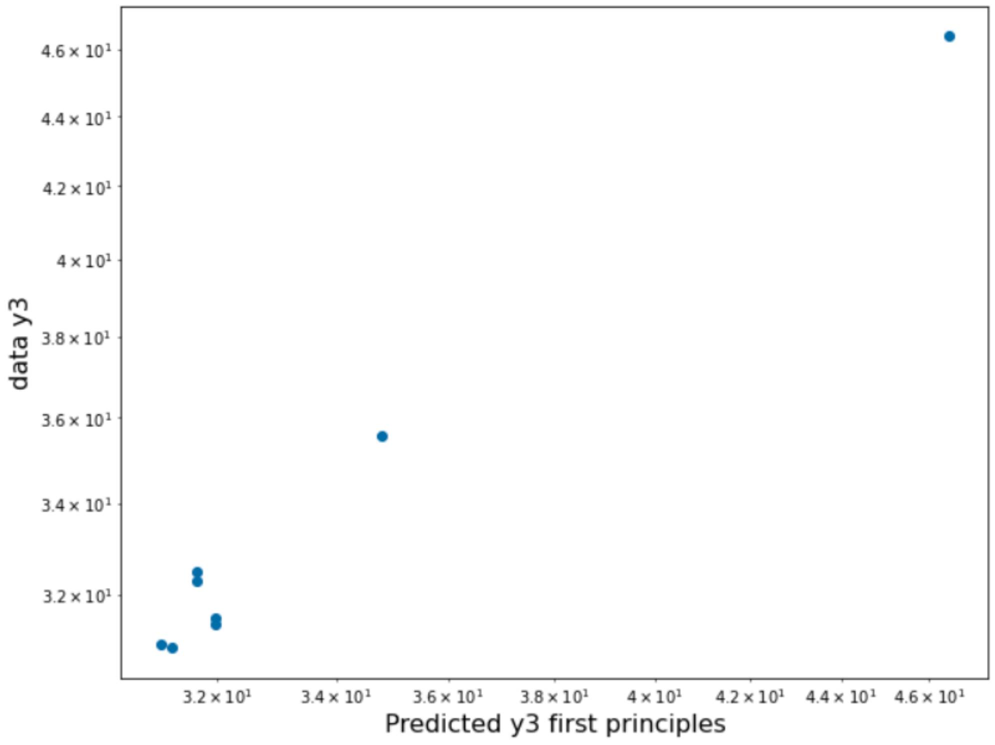

Figure 2: Predicted vs Data for First-Principles Artificial Neural Network Models

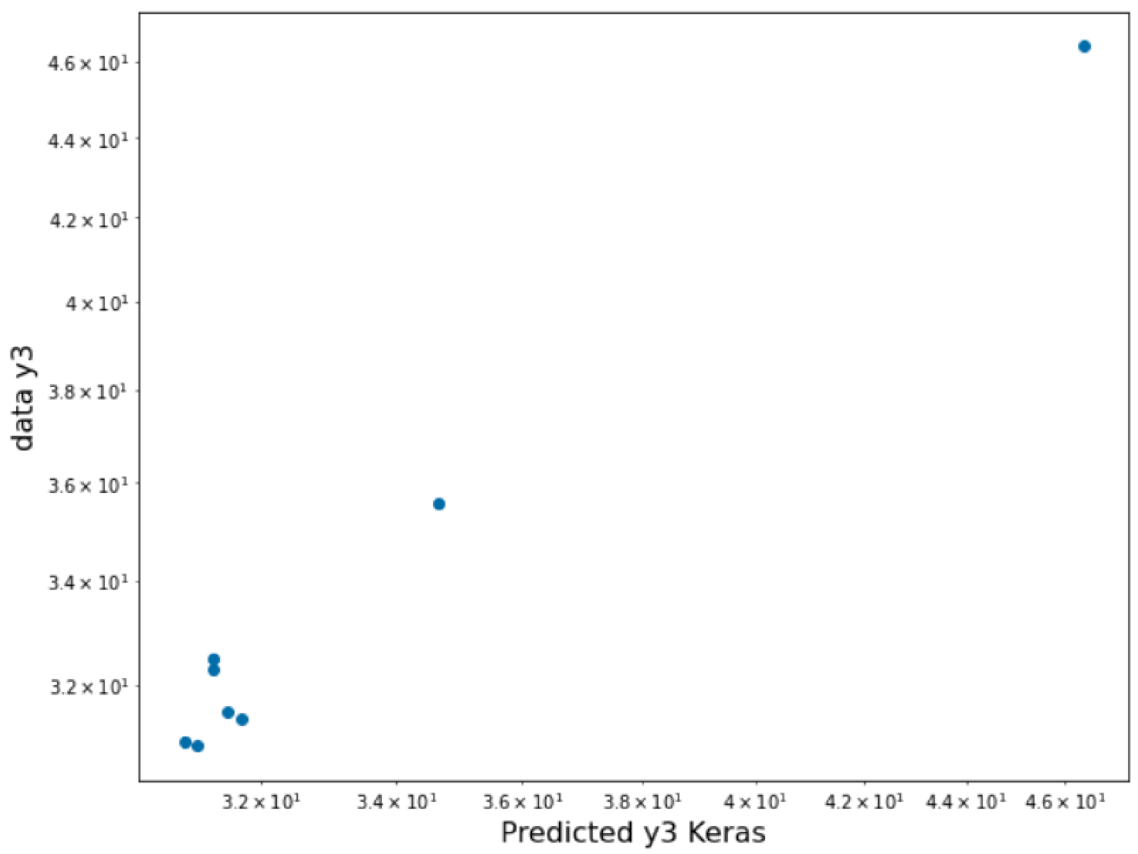

Figure 3: Predicted vs Data for Keras Artificial Neural Network Models

IV. Finally, using the selected first principles and Keras models, the initial values of the weights and biases are modified to 1.2 times the initial. Then the models are trained to achieve the best fit with a loss of less than 0.025. Using the predicted data comparison is done with the earlier predicted values to analyze the sensitivity of the model to the weights and biases. What was determined is that the model is not as sensitive to the change of the initial values, but may require more epochs to reach a satisfactory error. Below, the final weights and biases are shown for both the initial test and the initial test that was 20% greater. The final values are slightly different, but meet the error requirements.

	First-principles (initial)	First-principles (initial x 1.2)	Keras (Initial)	Keras (initial x 1.2)
w01	1.16307788853736	1.330454505834746	1.2245948	1.2100601
w02	0.41545711694896	0.518082059970551	0.3014321	.37602067
w03	0.70121934723217	0.819484485376314	0.7244949	0.7082452
b1	-0.1475266383815	-0.19661969807460	-0.137873	-0.14468023
w12	0.69923645921089	0.794065168503205	0.7087467	0.7057381
b2	-0.1188795905498	-0.16722390789335	-0.100631	-0.11462832
w23	0.70911250135287	0.561876617810140	0.6829315	0.6842814
b3	0.01051265119380	-0.02143072177226	0.0368672	0.01542608

Table 5: Comparison of the weights and biases for the initial and modified values

IV. Part Two

The task aims to analyze the performance of the hybrid solar fossil-fuel gas turbine system using a neural network model. The main objective of the neural network model is to help predict the required air-fuel ratio to achieve the turbine inlet temperature T_4 for varying conditions. These results would help the controller present in the burner to effectively set the fuel flow rate to the proper value.

A. As explained above the system is trained to fetch the air-to-fuel ratio α and the efficiency of the system η_{sys} as dictated by the operating conditions initial inlet temperature T_1, the ratio of propane in the fuel mixture γ, and the rate of solar thermal heat input Q_S.

1. Using the data points given in CodeP2.322.ipynb file, the five data point version is edited to incorporate all the points. Then both the input and output data points i.e, xdata and ydata respectively, are printed out to verify the code.
2. Next, using NumPy functions, the median of each parameter in the data set is calculated. Each data point in the data set is then divided by the respective calculated median value to normalize the data. With the input data points normalized, the data is ready to train the neural network model.

B. With the CodeP2.422.ipynb file, a neural network with four layers is created. Having three input variables defined as the operating conditions fed into the neural network, an input hidden dense layer is created with a three-input shape and 16 neurons with a rectified linear units (relu) function. Two dense layers are added with 32 and 16 neurons respectively, with activation functions set the same as the input layer i.e. relu.

With the neural network layers set, the first two cells are run and checked for any errors prior to commencing the next step.

C. With no errors present, the normalized data set is used to train the neural network. Two parameters, the learning rate to 0.001 and the number of epochs to 600 were manipulated and then sequentially ran with all four cells. Here, the model is trained using successive forward passes for each data point in the set and a backpropagation pass to update the weights and biases at the end of each epoch. The model is trained until there is a mean absolute error of less than 0.05. To achieve this low error after a sequential run of all four cells, only the fourth cell is run repeatedly which pushes the weights and biases generated at the last epoch of the previous run as the initial weights and biases for the next. This extends the number of epochs and narrows down the mean absolute error to a lower value.

Results:

best epoch = 184

smallest loss = 0.03583650613824527

D. In obtaining a trained neural network, the neural network was used to determine the variation of α for a set of operating conditions. At a fixed γ value equal to 0.25, $268 < T_1 < 318$ K and $500 < Q_s < 2500$ kW, a surface plot is created for α as a function of T_1 and Q_s.Figure 2: Surface Plot for α as a function of T_1 and Q_s

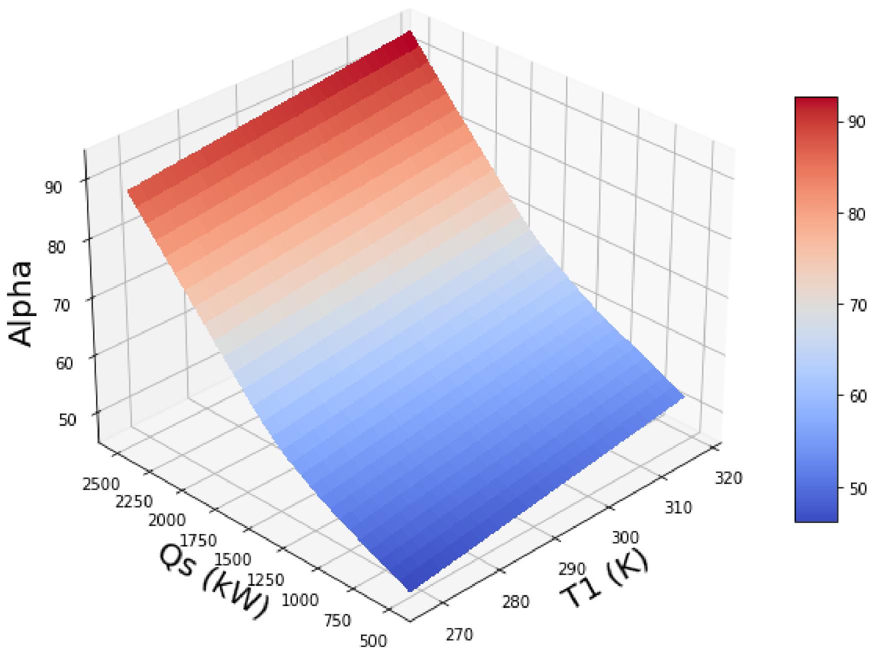

Figure 4: Surface Plot for α as a function of T_1 and Q_s

E. Having a clear idea of how the neural network model works and how to determine the outputs from the earlier task, a distinct set of test data is used to determine the α predicted value and is used to compare the results with the α test data provided. Firstly, the test data is initialized, and then using the median value determined for the training data, normalization is performed. Next, the normalized data set is inserted into an array which is passed into the model·predict function to determine the output values. The point to remember is that the output values given by the model are to be multiplied by the respective median to obtain the physical output values. Using the predicted output values a log-log plot against the corresponding test data values is created.

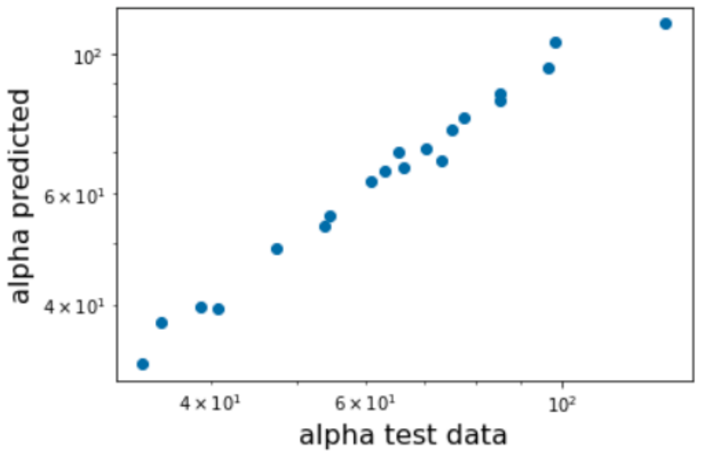

Figure 5: Alpha test data vs. Alpha Predicted Scatter Plot

We also determine the RMS deviation between the prediction and test data collection using the below-stated equation to be 4.991060819 units.

$$RMS = \sqrt{\frac{\sum_{i=1}^{20} \quad (\alpha_{pred} - \alpha_{test})^2}{20}}$$

F. In the previous task, a set of test data is used to evaluate the outputs. Now, in this task, a new set of operating conditions of T_1 and Q_s anticipated over a typical summer day is provided. For two conditions of γ=0 and γ=0.5, α is predicted over time.

Similar to the earlier task, test data is initialized and then normalized using the median value of the training data. This normalized data is then inserted into an array and then passed into the model·predict function to determine the output value for both γ conditions.

Using the predicted output values a scatter plot against the corresponding time course of the day is created.

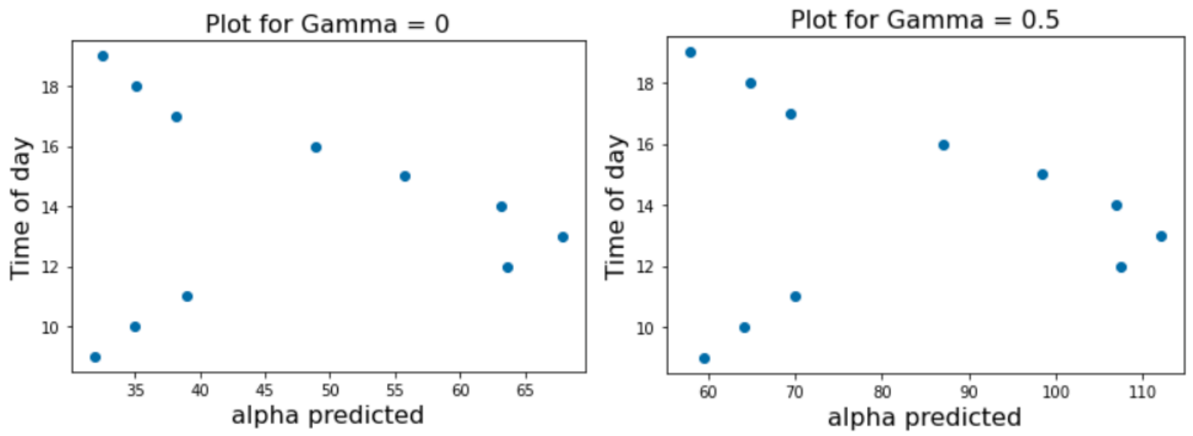

Figure 6: Alpha Predicted Values vs Time Over the Course of the Day

G. The final task is analyzing the baseline neural network model for four different modified variations. For each variation the minimum loss obtained and the number of epochs run is noted down to indicate the convergence steps taken.

The baseline network used in part two for training the data set is modified by the following modifications and run to achieve a minimum loss and then compared to understand the best design variation.

1. The baseline neural network with the relu activation function changed to ELU.

   The activation functions used in task 2.2 cell1 are changed to the exponential linear units function and then the cells are run sequentially. The best epoch and minimum loss are noted.
   - best epoch = 830
     smallest loss = 0.030023944626251856

   Observation: The minimum value was achieved with a minimum number of epochs compared to the baseline neural network model. The activation function elu, providing some gradient for negative z values improved the training efficiency.

2. The baseline neural network with an added layer and the number of neurons in the layers set to 12, 24, 12, 12, 2 (activation functions will be relu for all, as in the baseline).

   In this step, the whole neural network layers are changed. The input layer is set to 12 neurons followed by three hidden layers with 24, 12, and 12, and finally, an output layer with 2 neurons. All the layers are set with an activation function Rectified Linear Units(relu) and run to find the loss.
   - best epoch = 1411
     smallest loss = 0.0274755934874216721

Observation: Increasing the number of neurons significantly changed the loss by converging to the lowest. As known, the relu activation function was highly efficient in training in the shortest training time compared to all other modified models.

3. The baseline neural network with the numbers of neurons changed to 8, 16, 8, and 2.

   The model is changed to the baseline network back again but the number of neurons changed to 8,16, 8, and 2 for an input, two hidden, and an output layer respectively.
   - best epoch = 1074
     smallest loss = 0.03805743008852005

   Observation: Keeping the activation function the same but just changing the number of neurons to a lower value increased the number of iterations required to achieve a close minimum loss value but not as close to the second model due to a reduced hidden layer.

4. The baseline neural network with the number of neurons in the layers set to 20, 40, 20, 2
   - best epoch = 806
     smallest loss = 0.034501071895162265

   Observation: The number of neurons plays a role in achieving a minimum error with minimal runs of the cell 4. Though the number of neurons were high compared to the last modified model the convergence rate did not change after the second run.

Conclusion: The activation function Rectified linear units is best used to obtain or train the data in a short time period with a minimum number of epochs. The number of neurons in the hidden layer has a slight effect on the convergence rate if the activation function is not modified.

V. Work Distribution

Throughout the project timeline, the work was individually done and then compared as most tasks were to be done by every member. Using GitHub, and living close in proximity, the team was able to easily collaborate. The report was worked on collaboratively.

VI. Conclusion

This project allowed for the development of familiarity with artificial neural networks through the development, application, and implementation within a working hybrid solar fossil-fuel turbine system. Both the first-principles and the Keras model were observed and both were able to achieve results that were nearly identical. However, the Keras model obtained results through a script that could be more easily manipulated and altered to meet specific requirements.

VII. Appendices

1. Figures

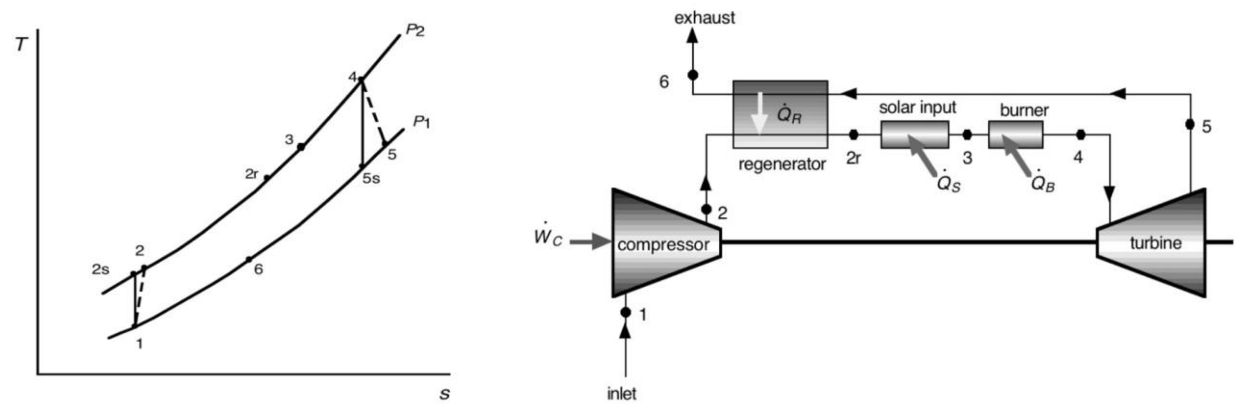

Figure 1: Hybrid solar fossil-fuel turbine system.

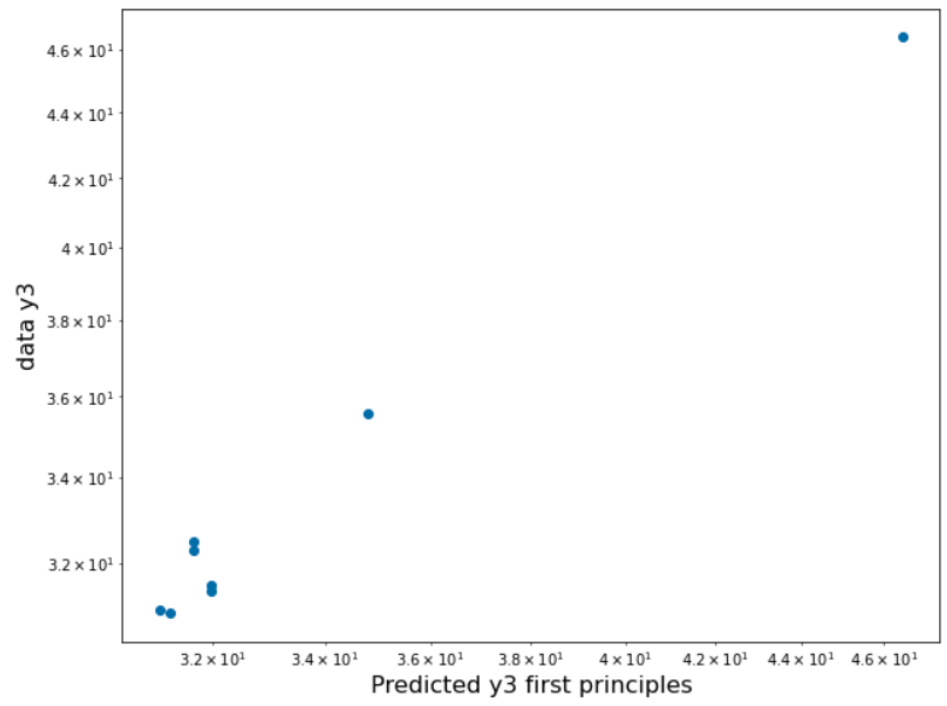

Figure 2: Predicted vs Data for First-Principles Artificial Neural Network Models

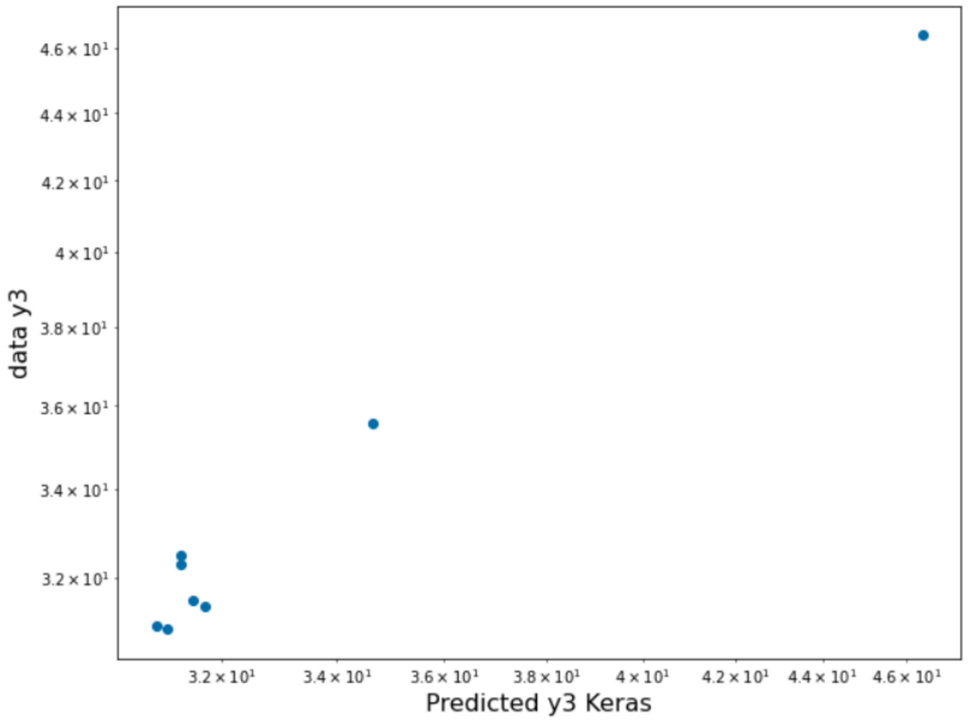

Figure 3: Predicted vs Data for Keras Artificial Neural Network Models

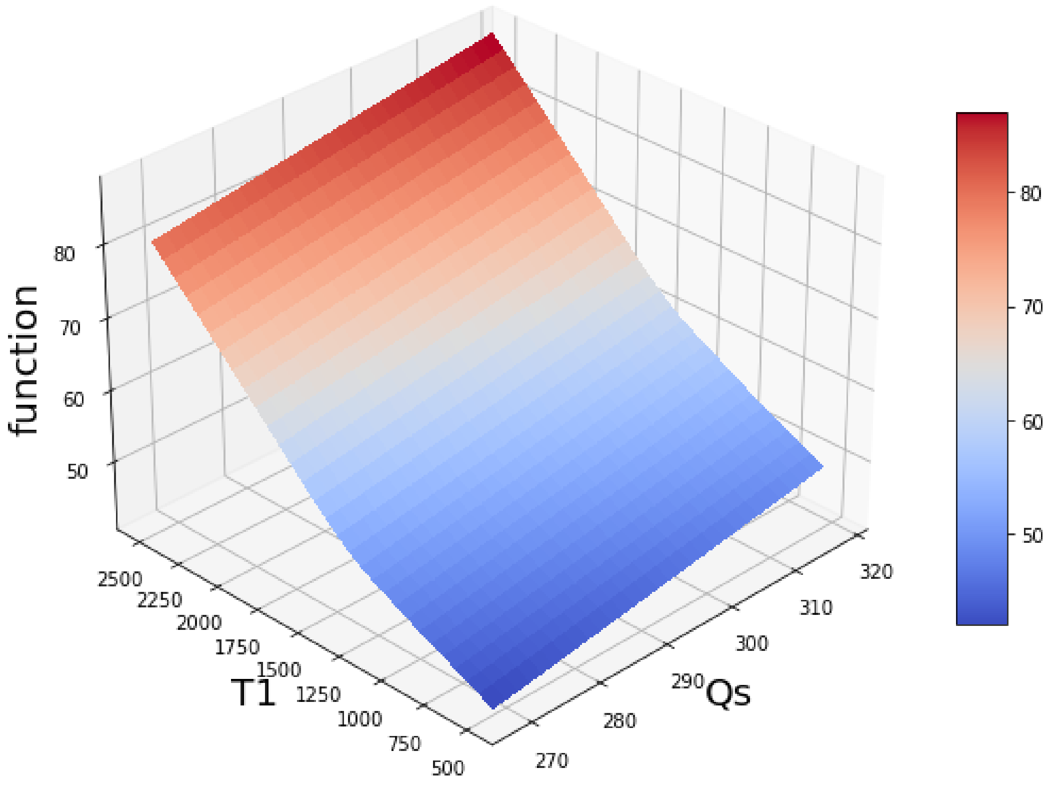

Figure 4: Surface Plot for α as a function of T_1 and Q_s

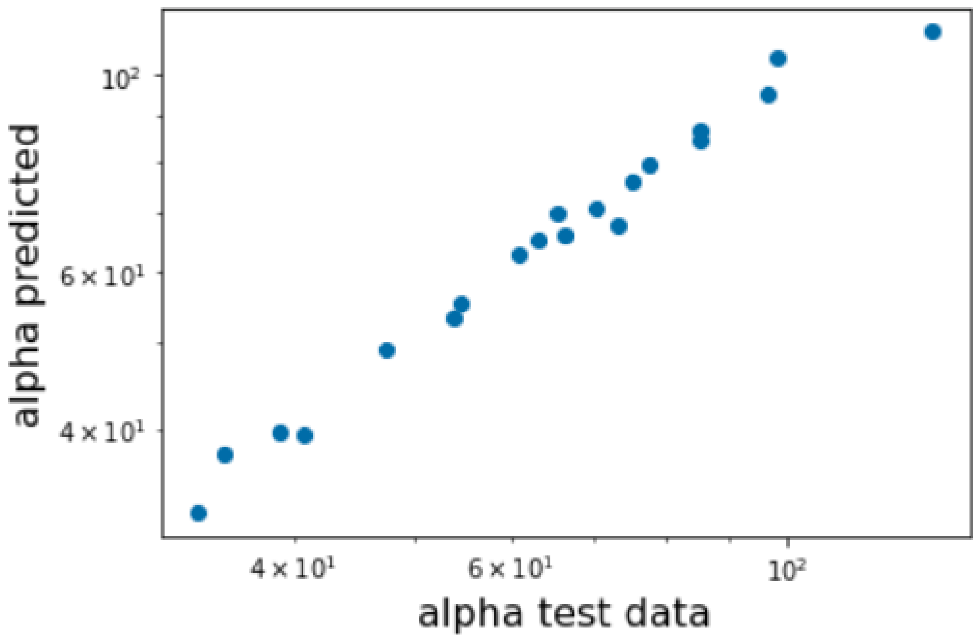

Figure 5: Alpha test data vs. Alpha Predicted Scatter Plot

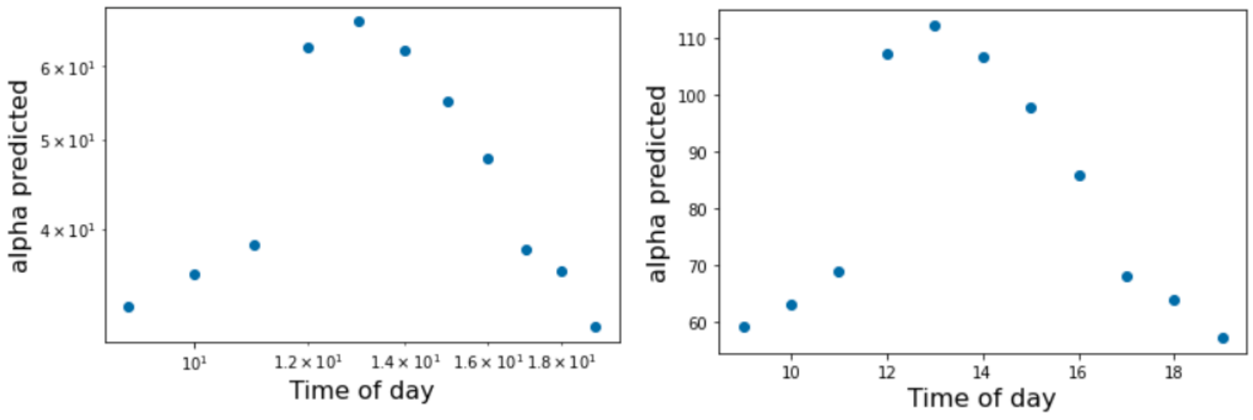

Figure 6: Alpha Predicted Values vs Time Over the Course of the Day

2. Tables

	First principles	Keras
w01	1.1630778885373607	1.2245948
w02	0.41545711694896326	0.30143216
w03	0.7012193472321739	0.72449493
b1	-0.1475266383815076	-0.13787363
w12	0.699236459210893	0.7087467
b2	-0.11887959054981592	-0.10063175
w23	0.7091125013528717	0.68293154
b3	0.010512651193809102	0.03686729
	rms=$\sqrt{.00028667222349910103}$ =0.0169314	mae= loss=0.01367854326963424

Table 1: Pavan Reddy First Principles and Keras Weights and Bias

x01	x02	x03	y3 data	First principles predicted y3	Keras predicted y3
20.0	13.0	310.8	30.97	31.090793	31.5234
20.0	14.5	308.0	32.3	31.678835	31.909126
20.0	15.3	306.0	31.5	31.973926	32.09619
20.2	13.0	310.8	30.91	31.275793	31.713541
20.0	14.5	308.0	32.5	31.678834	31.909126
20.0	15.3	306.0	31.4	31.973926	32.286304
24.0	13.0	310.8	35.59	34.790793	35.326298
36.0	14.5	308.0	46.4	46.478832	47.1207

Table 2: Pavan Reddy First Principles and Keras Predicted Values Comparison

	First principles	Keras
w01	1.216357733025874	1.2239491
w02	0.4125302702148731	0.2895847
w03	0.6884058318375348	0.7234253
b1	-0.16095176378660467	-0.14015044
w12	0.714042336036248	0.70647335
b2	-0.127856005074306	-0.10168917
w23	0.6939304618676472	0.6799924
b3	0.004522231198836544	0.03707751
	rms=0.018035753115350762	mae=loss=0.013742767274379 73

Table 3: Joshua Duarte First Principles and Keras Weights and Bias

x01	x02	x03	y3 data	First principles predicted y3	Keras predicted y3
20.0	13.0	310.8	30.97	31.111950636608388	31.044891
20.0	14.5	308.0	32.3	31.696596380009723	31.408806
20.0	15.3	306.0	31.5	31.990227141614145	31.584368
20.2	13.0	310.8	30.91	31.305292042148874	31.233511
20.0	14.5	308.0	32.5	31.696596380009723	31.408806
20.0	15.3	306.0	31.4	31.990227141614145	31.772987
24.0	13.0	310.8	35.59	34.97877874741809	34.817287
36.0	14.5	308.0	46.4	47.163908823248555	46.498375

Table 4: Joshua Duarte First Principles and Keras Predicted Values Comparison

	First-principles (initial)	First-principles (initial x 1.2)	Keras (Initial)	Keras (initial x 1.2)
w01	1.16307788853736	1.330454505834746	1.2245948	1.2100601
w02	0.41545711694896	0.518082059970551	0.3014321	.37602067
w03	0.70121934723217	0.819484485376314	0.7244949	0.7082452
b1	-0.1475266383815	-0.19661969807460	-0.137873	-0.14468023
w12	0.69923645921089	0.794065168503205	0.7087467	0.7057381
b2	-0.1188795905498	-0.16722390789335	-0.100631	-0.11462832
w23	0.70911250135287	0.561876617810140	0.6829315	0.6842814
b3	0.01051265119380	-0.02143072177226	0.0368672	0.01542608

Table 5: Comparison of the weights and biases for the initial and modified values

3. Code