Spring 2023

Dynamic Predictions of Thermal Heating and Cooling of Silicon Wafer

Hitesh Kumar
San Jose State University

Dynamic Predictions of Thermal Heating and Cooling of Silicon Wafer

A Project Report

Presented to

Professor Robert Chun

Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Class

Spring-2023: CS 298

By

Hitesh Kumar

May 2023

The Designated Project Committee Approves the Project Titled

Dynamic Predictions of Thermal Heating and Cooling of Silicon Wafer

By

Hitesh Kumar

Approved for the Department of Computer Science

San Jose State University

May 2023

Dr. Robert Chun	Department of Computer Science
Dr. Navrati Saxena	Department of Computer Science
Dr. William Andreopoulos	Department of Computer Science

ABSTRACT

Neural Networks are now emerging in every industry. All the industries are trying their best to exploit the benefits of neural networks and deep learning to make predictions or simulate their ongoing process with the use of their generated data. The purpose of this report is to study the heating pattern of a silicon wafer and make predictions using various machine learning techniques. The heating of the silicon wafer involves various factors ranging from number of lamps, wafer properties and points taken in consideration to capture the heating temperature. This process involves dynamic inputs which facilitates the heating of the silicon wafer to make an IC chip. In this research, LSTMs (Long Short Term Memory) and RNNs (Recurrent Neural Network) have been used with the time series. This problem comes under the Multivariate time series analysis where time factor is taken into account as the heating goes on. This study includes the wafer heating pattern recognition, implementing LSTM and findings which leads to advancement of the silicon wafer heating so that manufacturing firms can identify heating anomalies and adjust inout parameters in their heating recipes to get the best yield. Also, these findings help to simulate the wafer heating by inputting the input parameters to get the surface temperature of the silicon wafer so that process engineers can build a fair idea of the recipe adjustments beforehand to get the best yield. The technical implementation done in this comprises use of Keras and sklearn libraries to use the machine learning capability via Python.

Index Terms – **Neural Networks, Deep Learning, Long Short Term Memory, Recurrent Neural Networks, Simulation, Wafer Heating, Keras**

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to Professor Robert Chun, for his guidance, support, and encouragement throughout my research. Professor Robert Chun's expertise and insights were invaluable in shaping the direction of my research and helping me to navigate the challenges and complexities of the research process. I am grateful for his patience and dedication, as well as for the countless hours he spent providing feedback on my work and helping me to improve my writing and research skills. Without his mentorship, this project would not have been possible, and I am truly grateful for his contribution to my academic and personal development. Also, I would like to thank Vladimir Kudriavtsev, fellow from Yield Engineering Systems for his guidance and approval of using the dataset from Yield Engineering Systems.

TABLE OF CONTENTS

TABLE OF FIGUERES

1. INTRODUCTION

LSTM or Long Short-Term Memory is a type of neural network which falls under the sub-category of Recurrent Neural networks. With time series implementation, they work effectively as they store the short terms of sequences for a long period of time. This strengthens them to identify the patterns and adjust weights accordingly. The idea of LSTM was first introduced in 1997 by 2 scientists who were working on the concept of gradient explosion in Recurrent neural network models [1]. The most basic unit in LSTM is a memory module which keep the track of short sequences. Each memory module controls the information flow and gates for the control of the information passing and flow.

Its basic structure is as follows:

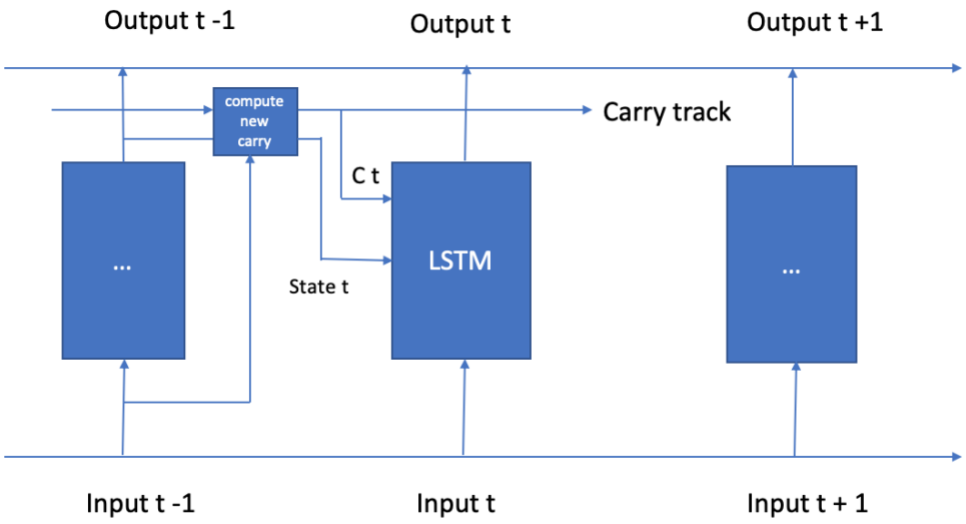


Figure 1: LSTM architecture

LSTM is called a special type or case of Recurrent neural networks. The cells in the LSTM have the capability to store the data, process it and transfer to the next layer. This transfer is done in the form of data streams within the cells [2]. The cell in LSTM is made up of 3 basic gates -

input gate, output gate and forgetting gate. By using these gates, LSTM hold the capability to process the data , make the use of the gates logic and process the data selectively. These logic gates are generally based on sigmoid neural networks.

The sigmoid layer works in the below manner:

1. Every sigmoid layer has the ability to generate number ranging from 0 to 1
2. The generation of numbers depends on the number of number of data segments passed through the layer
3. If the value is 0, it means no data passing is permitted
4. If the value is 1, this allows all data segments to be passed through

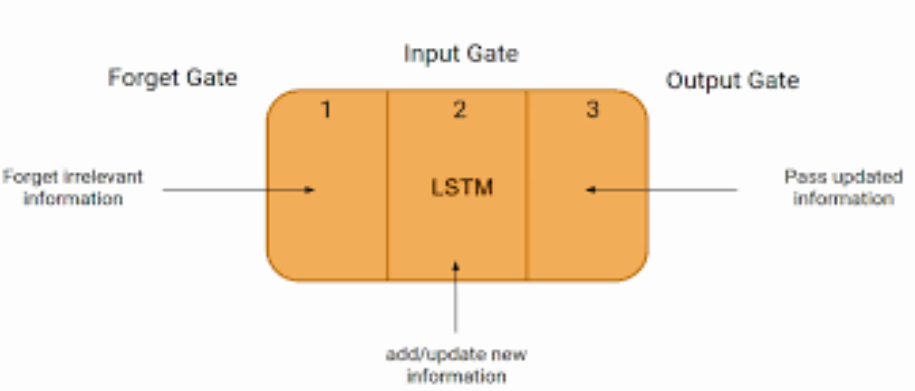


Figure 2: LSTM gates

The logic gates in LSTM play a vital role in the information storage and passing. The most interesting gate is the forgetting gate which operates on the value of 0 and 1. Here 0 and 1 means fully “reserved” and “ignored” respectively. The storage layer which is the input layer acts as the layer for storing data using the sigmoid layer as tanh [3]. The tanh has the function to create vectors that are being used by the sigmoid layer to select the values and store them in the vectors. These

vectors are then added to the state of the cell. The input gate has the function of filtering, adding and summarizing the cell state value.

1.1 Recurrent Neural Network VS LSTM

The basic concept of a neural network is that we stack different node cells on top of each other that contain multiple layers, one of them being a hidden layer. The hidden layer serves the purpose of learning and adjusting weights accordingly. The other significant layer is the dense layer for generating outputs. The major drawback of neural network is there is no feature of memory [4]. The previous result's essence is lost and it becomes difficult for sequential type of data where the core idea is to remember the previous sequence and generate the output accordingly. This brings us to Recurrent Neural Networks which work in the form of a feedback loop that functions in a form of memory. Hence, the output received are always associated with the previous output [5].

1.2 Time Series Data

Series of data compiled over a period of time steps is known as Time Series Data. As compared to regular data series, this data is meant to track events with respect to a fixed unit of time. The observations are directly correlated with increasing or decreasing time steps are arranged in a chronological patterns and behavior of a particular process [6].

1.3 Time Series Graph

The data series when plotted with an x-y axis is known as a time series graph. These types of graphs are used to analyze worthy models.

More specifically, time series data visualization offers an exploratory tool for determining whether data:

1. Reverts to being mean or exhibits explosive behavior
2. is time-trending
3. demonstrates seasonality
4. displays structural flaws

What does it mean to have reverting data? Data that "mean-reverts' ' eventually settles on a mean that is independent of time [7].

1.4 Time trending data

A specific component of time period has a lot of information to extract, which means the data for that period has a specific.

1.5 Seasonality

A seasonality in terms of time series is defined as the pattern repetitive over a period of time when plotted in the form of a graph. The values don’t need to be exact but the shape of the graph should be close enough to capture an ongoing trend [8].

1.6 Structural breaks

Sometimes the time series graph follows some irregular behavior which may indicate a break in the repetitive pattern. These are essential to tweak the model and know the insights of the data.

1.7 Time-Domain VS Frequency Domain Models

In our current use-case, we will be using a time-domain approach as a function of past and future values graphs on a time axis. For the forecasting, the regression values will be used for the prediction.

1.8. Universal Approximation Theorem

The Universal Approximation Theorem is the building block of the neural network, artificial intelligence and subset of virtual reality. It is fundamentally a mathematical hypothesis which states that a neural network which has only a single layer with neurons depending on the size of the input has the capability to make approximations related to the output[1] [2]. The catch here is that the function should be of continuous nature. The precision depends on the layers and neurons architecture in the network. This concept has been pivotal in building neural networks and coming up with multiple concepts within.

This theorem was brought first in 1989 by George Cybenko and then by Kurt Hornik in 1991. The background theory behind building this concept was the concept of linear function. It says, neural network with a single layer has the capability to approximate a linear function, hence combining multiple neural networks can make it approximate continuous function.[3][4]

There are three important caveats to be taken care of . First, the function we are trying to approximate must be a continuous function bounded by a finite range. It ought to be defined on a defined set. Second, this theorem makes approximations, it does not guarantee predicting exact values. Third, the activation function must be non-constant and of continuous nature. Nowadays, the activation functions used are ReLu, sigmoid, tanh etc.[5][6]

The Universal approximation theorem has played an important role in building neural networks, solving multiple complex problems and making predictions on a large variety of datasets. For Semiconductors, heating patterns , this theorem becomes motivation and food for thought if this can be used in the fab industry.

This theorem confirms that it can make approximations, with an underlying assumption that the neural network is optimal with respect to layers, neurons and approximations [7]. Every dataset has a different configuration and this comes under the area of research to fix the model and make it work for the dataset. Hence, developing techniques for building the neural network is hit and trial and knowing the nuances of the dataset. This makes it even more important to learn the function of the data and apply good practices of building an optimized neural network.

1.9 Motivation for using Universal Approximation Theorem to approximate cosine curve

To perform and emulate the working of Universal approximation theorem, a 1-3-1 neural network model is being programmed using python where the weights and biases of the neural network are being used. The below python code comprises building a neural net from scratch:

1.9.1: Python code to emulate the cos function.

Part 1 : Creating a 1-3-1 feedforward neural network

Learning rate = 0.01

Epochs : 100000

```
[1] #### final
    #!/usr/bin/env python3
    # -*- coding: utf-8 -*-
    """
    Created on Sun Mar 19 16:17:10 2023

    @author: hiteshkumar
    """

    import numpy as np

    def G(s):
        return 1 - np.exp(-2*s) / (1 + np.exp(-2*s))

    # Derivative of G(s)
    def deriv_G(s):
        return 4 * np.exp(-2*s) / (1 + np.exp(-2*s))**2

    # Define the neural network
    class Cos_emulator:
        def __init__(self, input_layer, hidden_layer, output_layer):
            # hidden layer : random init of weights and biases
            self.W1 = np.random.randn(input_layer, hidden_layer)
            self.b1 = np.zeros((1, hidden_layer))

            # output layer : Initialize the weights and biases for the
            self.W2 = np.random.randn(hidden_layer, output_layer)
            self.b2 = np.zeros((1, output_layer))
```

Figure 3: 1-3-1 neural network

The above code does the following:

1. initializes random values to bias and network weights.
2. defines the activates function.
3. defines the derivative of the activation function for easing out the calculation.

```
[ ]     def calc_forward(self, X):
            # hidden layer
            Z1 = np.dot(X, self.W1) + self.b1
            A1 = G(Z1)

            #output layer
            Z2 = np.dot(A1, self.W2) + self.b2
            return Z2

        def calc_backward(self, X, Y, learn_rate):
            # output layer : gradients of the weights and biases
            dZ2 = self.calc_forward(X) - Y
            dW2 = np.dot(self.A1.T, dZ2)
            db2 = np.sum(dZ2, axis=0, keepdims=True)

            # hidden layer : gradients of the weights and biases
            dA1 = np.dot(dZ2, self.W2.T)
            dZ1 = dA1 * deriv_G(self.Z1)
            dW1 = np.dot(X.T, dZ1)
            db1 = np.sum(dZ1, axis=0, keepdims=True)

            # updating of weights and biases
            self.W2 -= learn_rate * dW2
            self.b2 -= learn_rate * db2
            self.W1 -= learn_rate * dW1
            self.b1 -= learn_rate * db1

        def train(self, X, Y, learn_rate, tot_epochs):
            error =[]
            for epoch in range(tot_epochs):
                temp = []
                temp.append(epoch)
                self.Z1 = np.dot(X, self.W1) + self.b1
                self.A1 = G(self.Z1)
                self.Y_hat = np.dot(self.A1, self.W2) + self.b2
                self.calc_backward(X, Y, learn_rate)
                temp.append(np.mean(np.square(self.Y_hat - Y)))
                error.append(temp)
            return error
```

Figure 4: Forward & backward pass

The above code adjusts the weights and calculates the backward and forward pass while training the wrights. Also, the functions adjust the bias values of every pass.

Learning rate = 0.001

Epochs : 10000

```
# Generating X-values and Y-values
X = np.random.uniform(low=-np.pi, high=np.pi, size=(10, 1))
Y = np.cos(X)

nn = Cos_emulator(input_layer=1, hidden_layer=3, output_layer=1)

error_array =nn.train(X, Y, learn_rate=0.001, tot_epochs=10000)

# Generate some test x-values
X_test = np.linspace(-np.pi, np.pi, num=100).reshape(-1, 1)

# Make predictions using the trained neural network
Y_hat = nn.calc_forward(X_test)

# Plot the original function and the predicted function
import matplotlib.pyplot as plt
plt.plot(X_test, np.cos(X_test), label='Actual cosine')
plt.plot(X_test, Y_hat, label='Emulated cosine')
plt.legend()
plt.show()
```

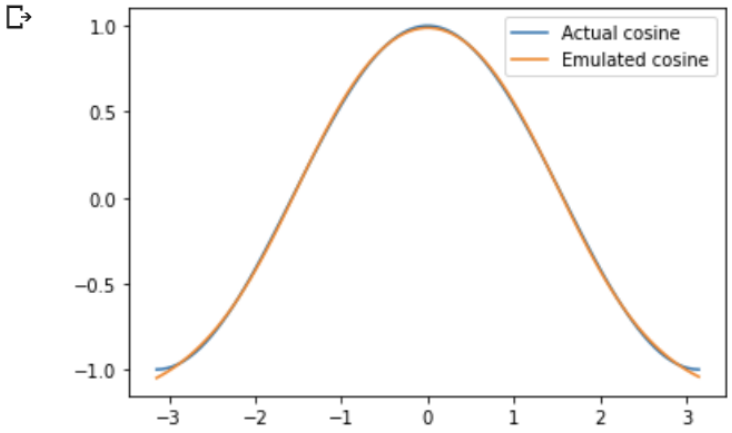


Figure 5: The plotting of cosine curve

1.9.2: Cost Function

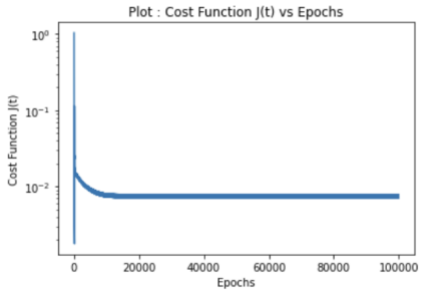
In this section, we will observe that the loss function with respect to epoch. As we can see in the "Cost function vs Epoch" image, the cost function gets lesser and reached to a saturation which denotes the model has converged.

Part 3 : Plotting cost function with time : takng log of cost function to show the detailed trend since we get a dropping line

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    from matplotlib.figure import figure

    df = pd.DataFrame(error_array, columns = ['time', 'cost function'])

    # plot the dataframe
    plt.plot(df['time'],df['cost function'])
    plt.ylabel('Cost Function J(t)')
    plt.xlabel('Epochs')
    plt.title('Plot : Cost Function J(t) vs Epochs')
    plt.yscale('log')
    figure(figsize=(50, 20), dpi=200)
    plt.show()
```




```
<Figure size 10000x4000 with 0 Axes>
```

Figure 6: Cost function vs Epoch

1.10 Digital Twin

Digital Twin is an upcoming technology which every industry is embracing. It is bound to have a strong impact on the engineering part of the industry in a transformative way of how operations are done. The idea behind the digital twin is how actual physical processes are simulated in the form of digitization and a process's same replica is made, hence called a 'twin'. As a twin has the same genetic material and functions the same, the process is simulated in the form of its digital

twin which can be controlled and manipulated easily with the help of a UI and clicks. Usually, the digitals twins are dynamic in functionality, which means as we change the parameters of the inputs and see changes corresponding to its physical nature in a programmatic and mathematical language.

1.10.1 Digital Twin – Benefits

When an organization needs to monitor and check their systems in real time, the digital twin comes into picture. For example, the Fab industry works on the sensor's data coming from various processing parts and being stored in a repository. Using this data, the opportunity of error detection comes into play like anomaly detection and pattern recognition. Also, for a new design testing and proof of concept, digital twins are very useful to have an initial prototype building.

1.10.2 Digital Twin - Challenges

There are multiple challenges that come with Digital twins. One of the major is - Data integration as the data for digital-twin requires data from multiple sources of data which needs to be packed into a single model. The other challenge is that it requires SME (Subject Matter Expert) for building the Digital Twin and for its regular maintenance of it.

1.11 Digital Twin – Applications

This concept of Digital twin finds its place in multiple fields like healthcare, silicon fab industry, automobile and transportation. In this report, we will be focusing on the silicon wafer heating simulation which will help the process engineers to simulate the wafer heating mechanism and build their recipes to bake the wafer in the oven. Also, it will help the process engineers to detect the anomalies of the heating step if the process goes out of the way anyhow. In transportation, like for example in building a driverless car, digital twin helps the developers to test their products on a variety of factors to get insights of the output in terms of fuel reduction and energy conservation. Also in airline simulation, various accidents are also avoided by simulating flight movements and analyzing factors accordingly.

2. PROBLEM STATEMENT

Yield Engineering Systems (YES) manufactures wafer heating machines and uses different recipes to check the efficiency of the machine they assemble. To do this, they follow a 5 step process to heat and process a silicon wafer disk which they call it recipes. The recipes they follow are ordered steps to process the silicon which involves setting the power of heating lamps (two being used for now), the ambient temperature of the disk and emissivity value of the wafer disk. The recipe is converted to 1305 rows of data with input features and output features as the surface temperature on 4 different points on the disk. The problem statement is to input the key features manually by the process engineers and predict the temperature of the points of disk so that it can simulate the wafer heating pattern and make adjustments in the recipe during the actual processing step to get a better yield.

3. DATASET

The dataset has been derived from the recipe used at Yield Engineering Systems Inc from one of the processes known as PI Cure. The PI machine follows a set of instructions where it bakes the wafer and follows a predefined process. The recipe is a set of Steps Numbers which carries information related to temperature, pressure , gas flow and setpoint.

The dataset comprises 35000 records of process steps which are being divided into 25 files. Each file has 1400 rows of data which tells one complete process of the disk by a machine. The dataset has been made public by Yield Engineering Systems under the guidance of Vladimir Kudriavtsev who is a fellow at Yield Engineering Systems(YES).

After a analysis of the dataset, 4 significant inputs are being considered which are as follows :

1. Power of heating lamp - 1 within the machine (Watts/m^2) **[Heat Flux1_Magnitude]**
2. Power of heating lamp - 2 within the machine (Watts/m^2) **[Heat Flux2_Magnitude]**
3. Ambient Temperature (Celsius) **[Radiation_AmbientTemp]**
4. Convection Film Coefficient **[Convection_FirmCoef]**

The output values are temperature of 4 different surfaces which are being marked using a distinct number for the identification purpose.

Temp_id_22163, Temp_id_5514, Temp_id_5300

```
1 df.head(2)
```

	Time	Heat Flux1_Magnitude	Heat Flux2_Magnitude	Convection_FilmCoef	Radiation_AmbientTemp	TEMP_TEMP_id_22163	TEMP_TEMP_id_5514	TEMP_TEMP_id_5300
0	0.0	0.0	0.0	20.0	22.0	22.0	22.0	22.0
1	0.2	0.0	0.0	20.0	22.0	22.0	22.0	22.0

Figure 7: Dataset top 2 rows

For our modeling point of view, we will be using the actual data which is being generated from the actual machine at the end of the processes. We will be focusing on the information related to Heat Flux magnitude-1, Heat Flux magnitude-2, Convection Film coefficient, Radiant Ambient Temperature. The mentioned parameters are the most important for carrying on the process and as an output, we will be considering four points on the wafer disk. The four output points on the wafer are being taken at the center, on the right boundary, top boundary and bottom. The temperature on the disk must be uniform and the predicted temperature must be as close to all the points.

For our data purpose, the 4 points on the wafer have been labeled with a number with a Temperature id: 22163, 5514, 5300 and 25190. Hence the output columns names are TEMP_id_22163, TEMP_id_5514, TEMP_id_5300, TEMP_id_21590.

The input data set and output dataset visuals are below:

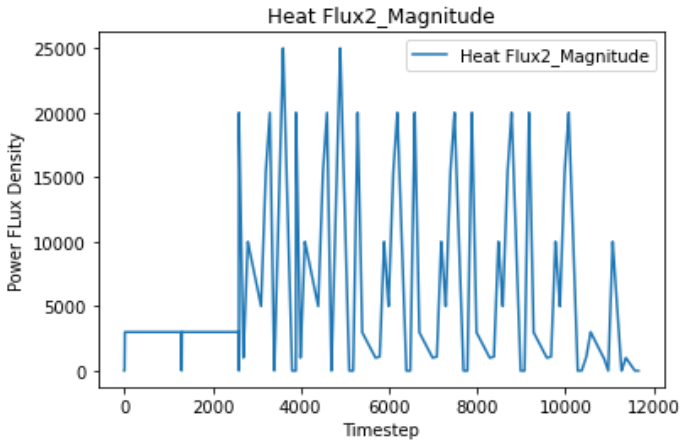


Figure 8: Heat Flux-2 input

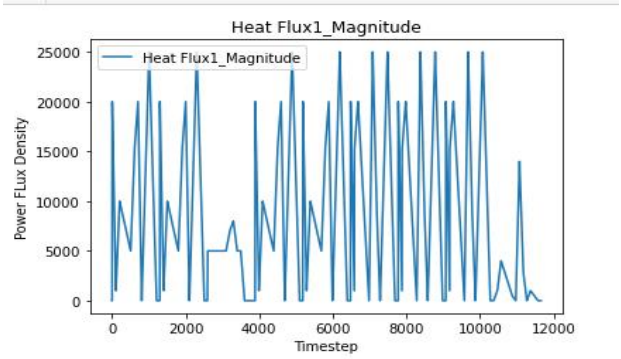


Figure 9: Heat Flux-1 input

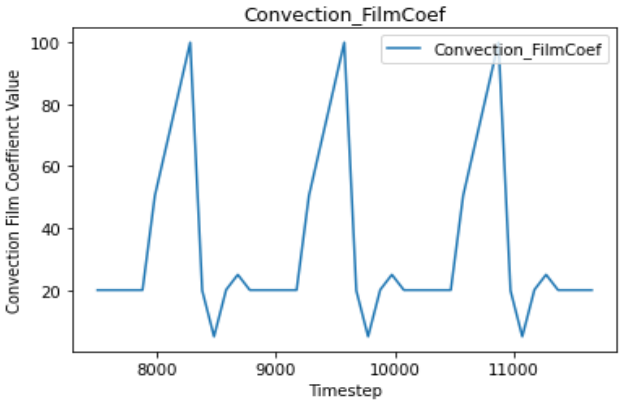


Figure 10: Convection film coefficient

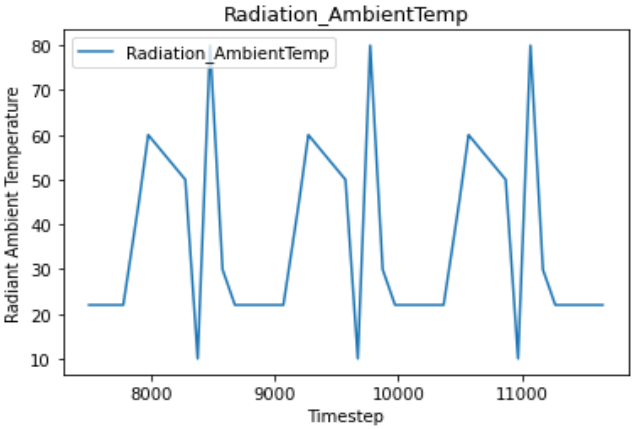


Figure 11: Radiation Ambient Temperature

The output points are as follows:

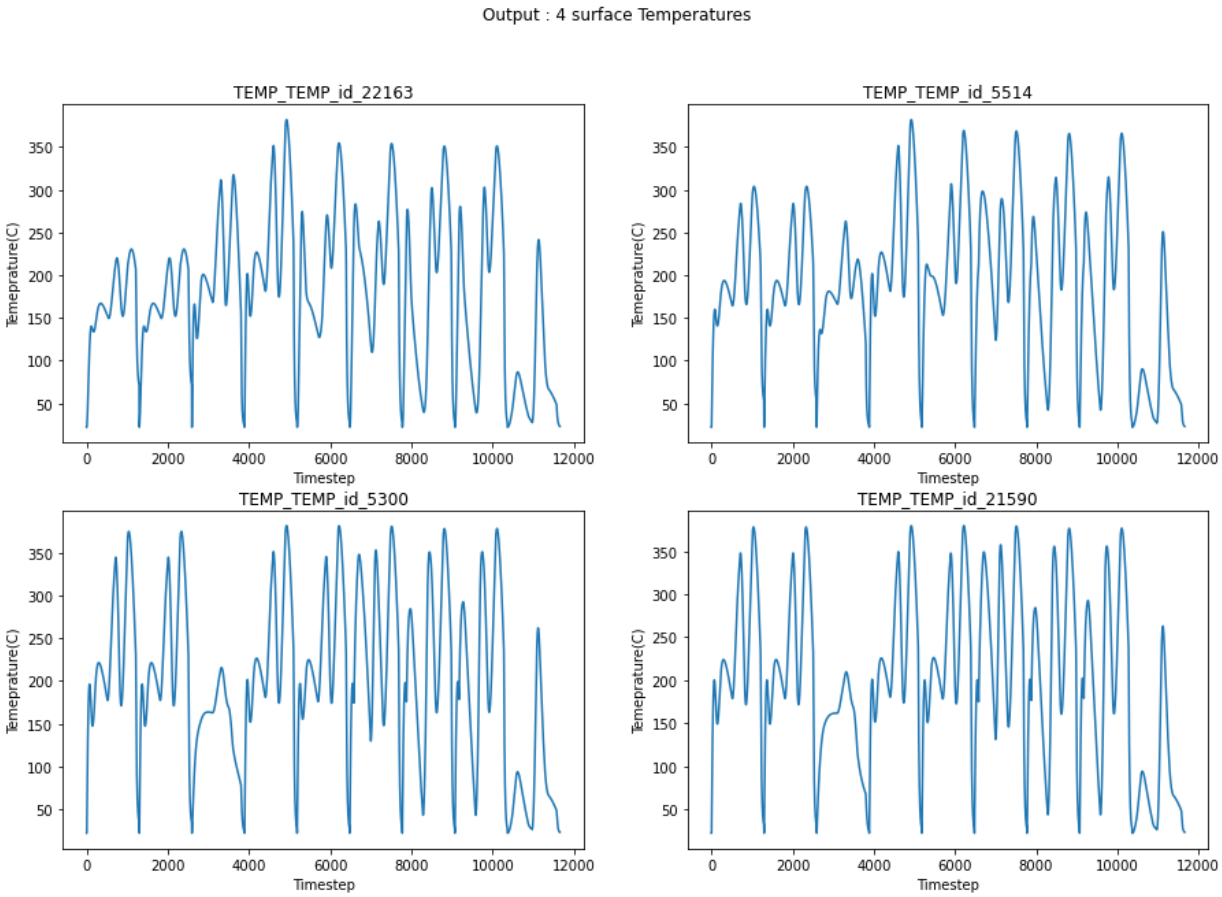


Figure 12: Output disk points

4. MODELLING APPROACHES

4.1 - Approach 1 - Using single LSTM layer

By using one LSTM, the heating pattern is easily captured. The LSTM remembers how the heating phases are changing and it predicts the temperature accordingly. Going from reflow to cooling, the LSTM has gained enough knowledge through training that the temperature is varied in terms of change of states. In this approach, I have created a basic LSTM network to check the feasibility of results in the terms of prediction and scaling [9].

Number of files used for training: 8

Number of files used for training: 1

Number of files used for testing: 2

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *
from tensorflow.keras.backend import clear_session
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam

clear_session()

model1 = Sequential()
model1.add(InputLayer((5, 5)))
model1.add(LSTM(64))
model1.add(Dense(8, 'relu'))
model1.add(Dense(1, 'linear'))
```

Figure 13: Keras sequential model with single LSTM

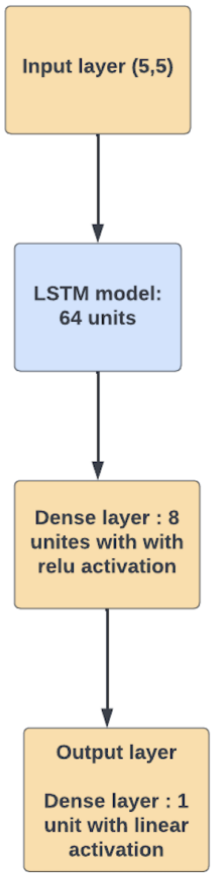


Figure 14: Flow diagram of base model architecture

Blue Line: Predicted temperature

Orange Line: Actual temperature

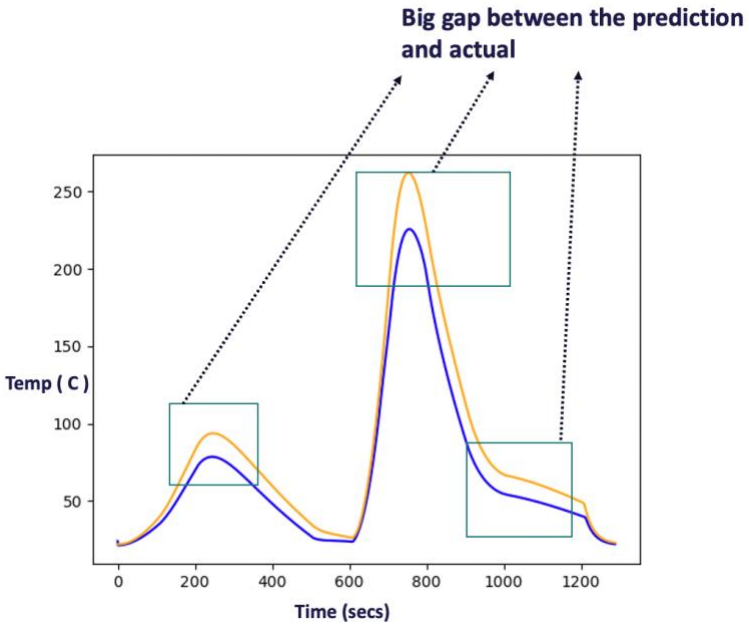


Figure 15: Prediction graph for heating pattern of test file

I tried to experiment different process flows to check segment wise capturing of pattern and scaling of the model. To verify each segment of data, I have broken down pieces in graph where it is off the track.

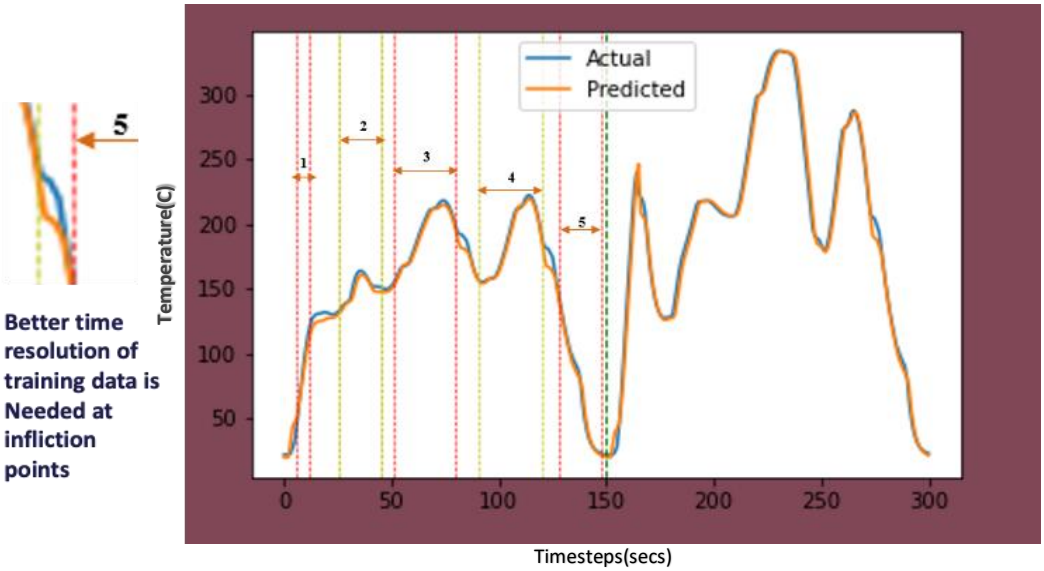


Figure 16: Phase wise tracking of graph

4.2 Approach 1 – Drawbacks

The drawback is, the scaling factor. The temperature is off at the high spikes as shown in the diagram. This is due to the use of single LSTM being used. The scope of improvement here is use of other layers and bidirectional LSTM. Current with this approach, the RMSE value is ranging from 25-35.

4.3 Approach 1 - Scope of improvement

(a) The missing spikes during the phase change can be experimented with different training batch sizes, adding a TimeDistributed layer and adding multiple LSTMS.
(b) Adding new neurons to each layer
(c) We can also experiment to add Bidirectional LSTM so that the information from both the sides of last 5 temperatures and next 5 temperatures can be used to determine the next temperature

4.4 Approach 2 - Using Bidirectional LSTM

The use of bidirectional LSTMs can solve the spike and phase change issue. The reason is pretty straightforward. The bidirectional LSTM keeps track of previous and next values to make a better prediction. We can experiment with different layers like the TimeDistributed layer [10]. It takes a temporal slice with respect to the time series inputted to the corresponding layer [11].

-

```
#Best 1
model1 = Sequential()
model1.add(LSTM(256, input_shape = (5,5), activation='relu', return_sequences=True))
model1.add(TimeDistributed(Dense(1)))
model1.add(Bidirectional(LSTM(128, return_sequences=True)))
model1.add(TimeDistributed(Dense(1)))
model1.add(LSTM(64, return_sequences=True))
model1.add(TimeDistributed(Dense(1)))
model1.add(Flatten())
model1.add(Dense(1, 'linear'))
```

LSTM Layer
Time Distributed layer
Bi-Directional Layer
Time Distributed layer
LSTM Layer
Time Distributed layer
Dense Layer

Figure 17: Possible network to be experimented

5. EXPERIMENTS

5.1 Single LSTM layer model architecture with look back = 1

In very first experiment, we will be using a single LSTM layer of 50 neurons and a single Dense layer. The purpose of this architecture is to measure the level of accuracy of the LSTM and whether it is able to predict the right trend of the heating pattern. Also, I will be using the loss function as Mean Absolute Error(MAE) with an optimizer as “adam”. Also, for the first experiment, the result’s analysis will give us in which direction we need to proceed to build a better model architecture with respect to the data and the problem.

Using a single layer will also help to avoid overfitting with model being trained quickly.

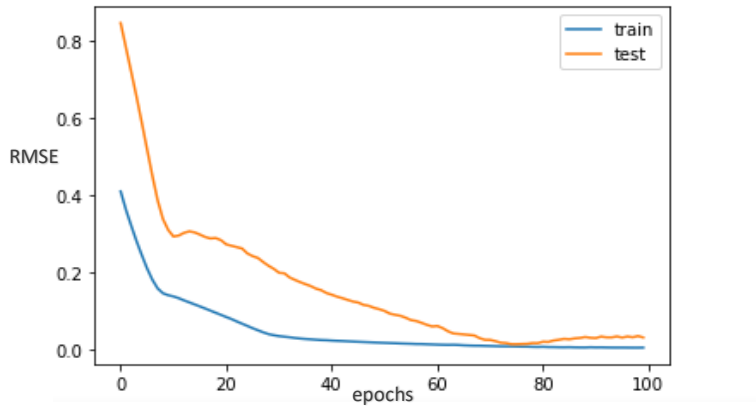


Figure 18: Training of exp-1

```
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
```

Figure 19: Model architecture of exp-1

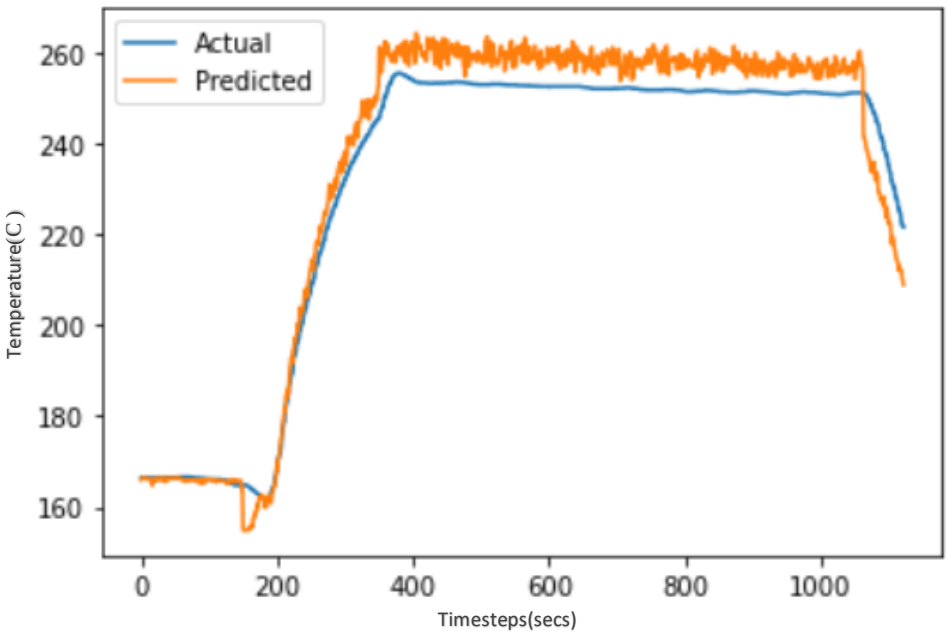


Figure 20: Predicted vs actual of exp-1

5.1.1 Analysis

With a single LSTM layer, the model is able to predict the trend of the heating pattern. The key observation here is that whenever there is a sharp change in the heating pattern, the model does not perform well. For example, in the above image, near timestamp 200, there is a change in the state of the wafer surface and the model is not performing well in those instances.

Another key observation is, whenever the heating temperature is rising linearly, the model is making good predictions along with the time. Hence a look back of 1 performs well for a particular state of the heating (linear rising or constant temperature) but makes bad predictions when there is an abrupt change in the state.

Another key observation is the start and ending timestamps of the wafer heating. The model is making extra predictions for the last few time stamps which needs to be fixed in the architecture.

For timestamp 400-1000, the model has identified the correct pattern but it's making predictions larger than the actuals. This makes sense because we have added only a single layer of LSTM which is essentially not good enough to capture the complex relationships of the inputs with respect to time.

The RMSE value 6.347 is high for this experiment, hence we will need to change our approach in another experiment. Also by observing the test and train loss, we can concluse that we can use the LSTM layer in our architecture experiment with different neurons and layers in our another experiment.

5.2 Modeling on Delta Temperature, single LSTM layer with look back = 1

Our primary objective is to make the temperature on the surface of 4 points as close as possible for each time stamp with a good accuracy.. We can introduce the delta-temperature and use it as an input feature(the only input) and observe the predicting pattern. For adding this as an input feature, 'diff_temp' has been introduced which takes the absolute difference of 4 points on the surface of the wafer with maximum temperature and minimum temperature.

For example in the figure :

At timestamp 0, the temperature on the 4 surfaces is 22, 22, 22, 22 respectively. Hence the delta temperature is 0.

At timestamp = 4, the temperature on the 4 surfaces is 23.121931, 22.34, 22, 22. Hence the delta-temperature is 1.21931 degree celsius.

Heat Flux1_Magnitude	Heat Flux2_Magnitude	Convection_FilmCoef	Radiation_AmbientTemp	Radiation_AmbientTemp .1	file_num	diff_temp
0.000	0.0	20.0	22	22	1.0	0.00000
0.000	0.0	20.0	22	22	1.0	0.00000
0.000	0.0	20.0	22	22	1.0	0.00000
10000.000	1500.0	20.0	22	22	1.0	1.21931
20000.000	3000.0	20.0	22	22	1.0	3.67546
19040.500	3000.0	20.0	22	22	1.0	25.84860
18081.000	3000.0	20.0	22	22	1.0	44.74679
16181.000	3000.0	20.0	22	22	1.0	69.57054
14281.000	3000.0	20.0	22	22	1.0	83.61014

Figure 21: Feature engineering of exp-2

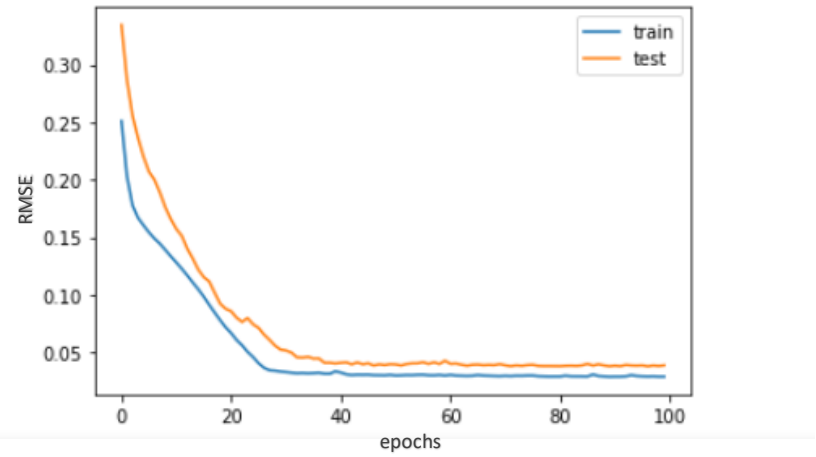


Figure 22: Training of exp-2

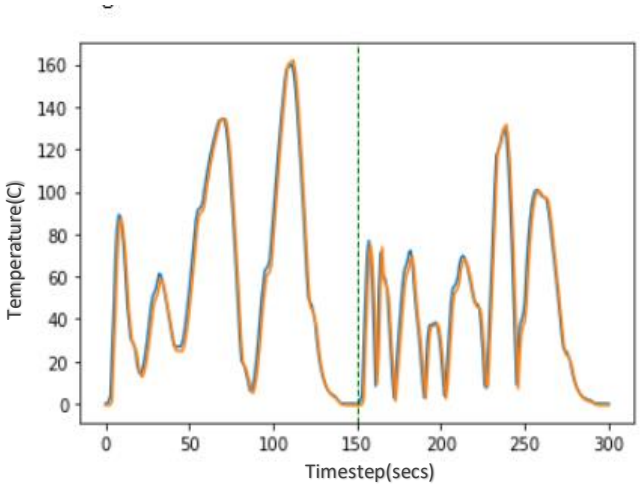


Figure 23: Predicted(orange) vs actual(blue) of exp-2 (overfitting)

5.2.1 Analysis

In this experiment, the model is trained only on the input of delta temperature to experiment the predictions. We can clearly see in the figure, the model is overfitting and, we cannot take this feature as a sole input for our model but can add it with our input feature to experiment with. We can check if adding this feature will give us any significant boost in our model's performance.

The train RMSE is low and test RMSE is quite high, which is a clear indication of overfitting.

In this experiment, we have taken only a single LSTM layer with 50 neurons and a dense output layer.

5.3 Experimenting with the layers, neurons, look back and learning rate

In this experiment, we will be adding an input layer, adding more neurons to the LSTM layer and adding 2 dense layers of 'relu' and 'linear' and will keep the look back value as 1.

```
model1 = Sequential()
model1.add(InputLayer((5, 5)))
model1.add(LSTM(64))
model1.add(Dense(8, 'relu'))
model1.add(Dense(1, 'linear'))

model1.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 64)                17920

 dense_4 (Dense)             (None, 8)                 520

 dense_5 (Dense)             (None, 1)                 9

=================================================================
Total params: 18,449
Trainable params: 18,449
```

Figure 24: Model architecture of exp-3

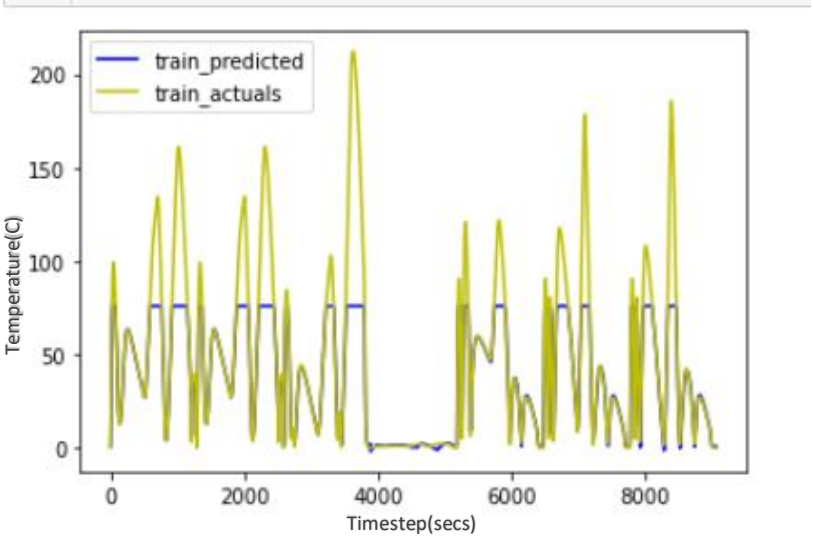


Figure 25: Training phase of exp-3

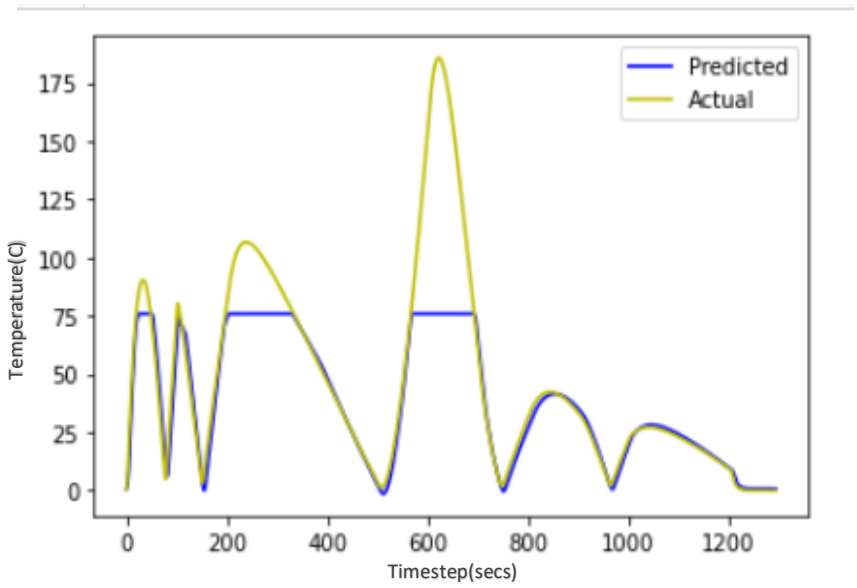


Figure 26: Predicted vs actual of exp-3

5.3.1 Analysis

In the above architecture mentioned in the image, the model is able to predict the linear up and down heating but the spike regions (hill tops) are missed by a huge margin.

We added ‘relu’ and ‘linear’ dense layers in the model which captures the non-linearity of the complex input data, but as the time goes, it is missing the changes in the state and transits to the new state through a straight line, hence missing the curve nature. The train and test RMSE is very high due to the missing mountain top curves. The Test RMSE is 23.98311.

In this experiment, LSTM layer is taking care of the temporal dependencies, the dense layers is building up the non linear relationships of input and output for each heating state. The relu is responsible for adding the non-linearity to the model’s architecture. The linear dense layer as the final layer is responsible for continuous output predicted by the model.

By changing the model architecture in this case, we are able to fix the start and end points of the prediction and make better predictions during the linear increase and decrease on the surface temperature. Also, the dense layer of linear paired with relu, assisted in improving the model performance and learning more complex patterns.

5.4 Experimenting Bidirectional & TimeDistributed layers with look back = 5

In our final experiment as seen in the figure, we will be introducing the Bidirectional LSTM layers, Time Distribution layer and will be experimenting with multiple LSTM layers with different numbers of neurons. Multiple time distribution layer and LSTM layers have been experimented and the best architecture is mentioned below. The model architecture with parameters is also included here:

```
model1 = Sequential()
model1.add(LSTM(256, input_shape = (5,5), activation='relu', return_sequences=True))
model1.add(TimeDistributed(Dense(1)))
model1.add(Bidirectional(LSTM(128, return_sequences=True)))
model1.add(TimeDistributed(Dense(1)))
model1.add(LSTM(128, return_sequences=True))
model1.add(TimeDistributed(Dense(1)))
model1.add(LSTM(64, return_sequences=True))
model1.add(TimeDistributed(Dense(1)))
model1.add(Flatten())
model1.add(Dense(1, 'linear'))

model1.build(X_train1.shape)
model1.summary()
```

Figure 27: Model architecture of exp-4

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 5, 256)	268288
time_distributed (TimeDistributed)	(None, 5, 1)	257
bidirectional (Bidirectional)	(None, 5, 256)	133120
time_distributed_1 (TimeDistributed)	(None, 5, 1)	257
lstm_2 (LSTM)	(None, 5, 128)	66560
time_distributed_2 (TimeDistributed)	(None, 5, 1)	129
lstm_3 (LSTM)	(None, 5, 64)	16896
time_distributed_3 (TimeDistributed)	(None, 5, 1)	65
flatten (Flatten)	(None, 5)	0
dense_4 (Dense)	(None, 1)	6

Total params: 485,578
Trainable params: 485,578

Figure 28: Model detailed summary of exp-4

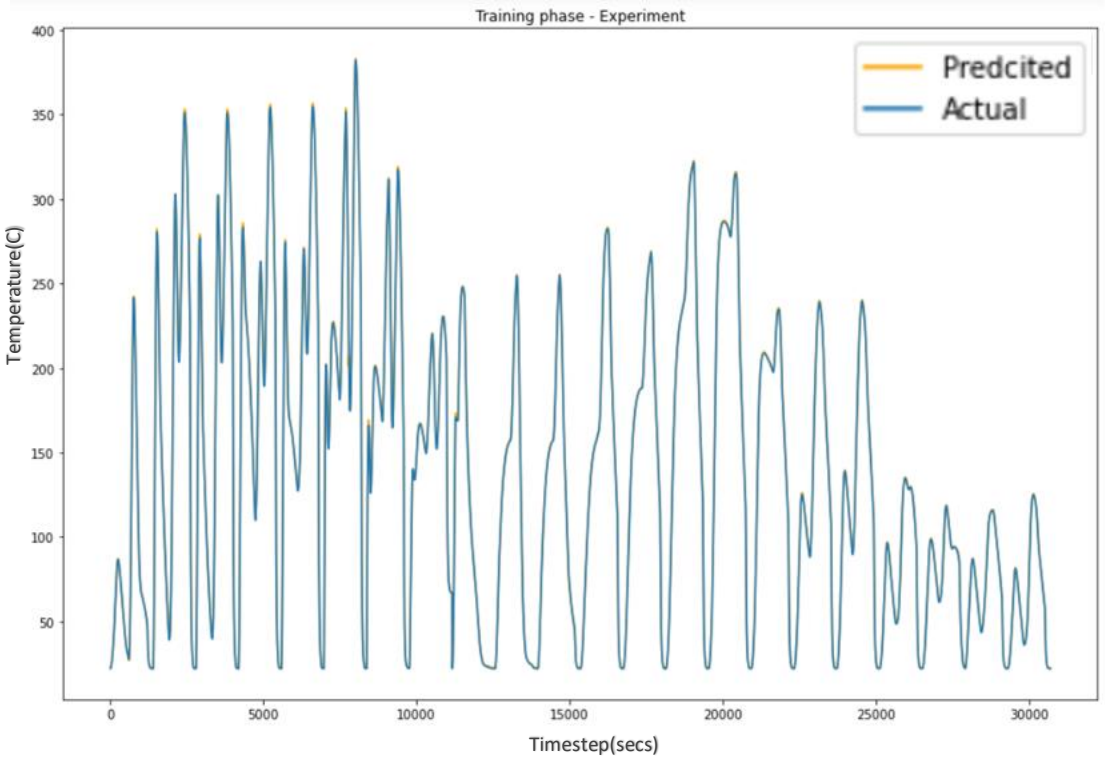


Figure 29: Training phase of exp-4

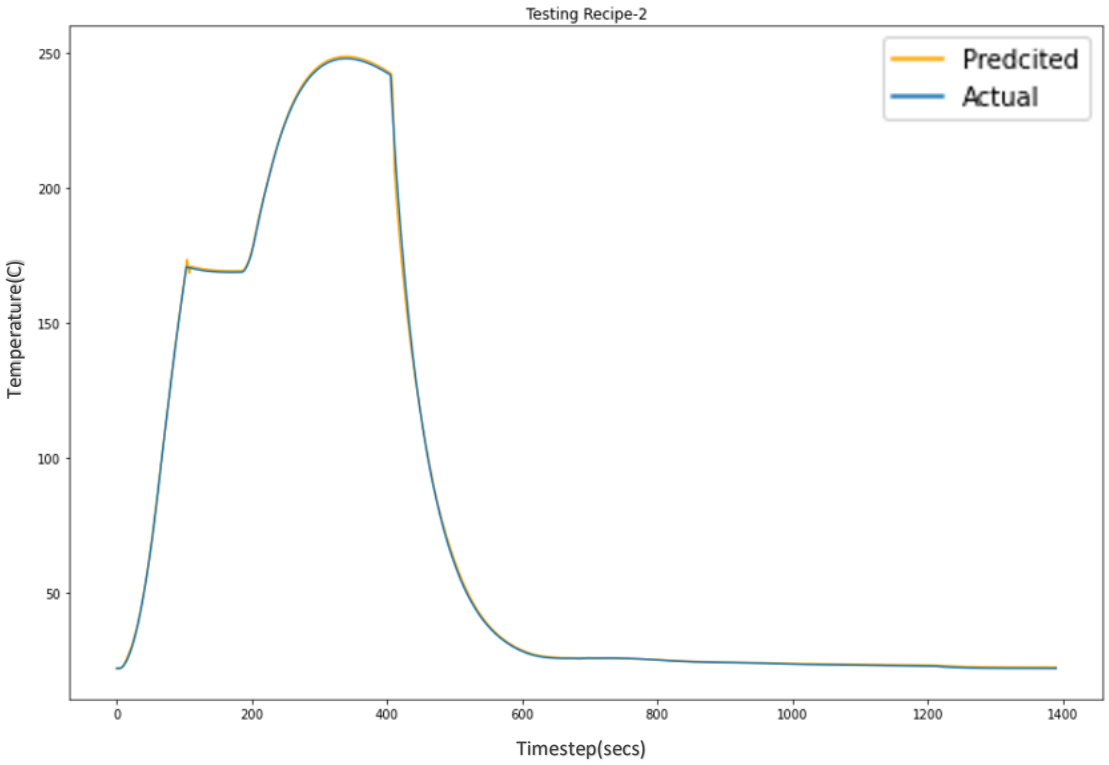


Figure 30: Predicted vs actuals of exp-4

5.4.1 Analysis

In this experiment, the look back value is 5, which means the model will check the last 5 values predicted at any time step. We received the best results with Training RMSE as 1.2370, Validation RMSE 0.7959 and Test RMSE 1.1686. To verify the results, I have used the validation dataset too.

Here the game changers are TimeDistributed layer and Bidirectional layer. For each sequence, the timedistributed layer is applied to each time step. This is useful as it applies operation to each timestep independently. This enhances the model's representational power and helps to build and understand the complex pattern happening between different time slots. As one observed in the validation test figure , the time stamp 150-250 is accurately captured by the model architecture. In our previous experiments, we were missing to catch up the hilltop curve. Also in the test set,

between 350-550 timestamp, the change in the state is being predicted by the model architecture quite accurately. The Bidirectional layer is responsible for processing the sequences inputted from both the directions, and it uses two different hidden states for performing such an operation. In the context of time series, it becomes essentially important to capture the temporal dependencies in the previous time and in future as well. Hence this helps the model to make better predictions by taking in account both forward and backward output values.

6. RESULTS & CONCLUSIONS

We first started with emulating a cosine function using the neural network which confirms the fact that a pattern of different curves can be predicted using a neural network. Then we started with the single LSTM layer and learned that the trend can be captured by a single LSTM layer though the values were off by a large margin. Then we tried the feature engineering of the delta-temperature and observed that using solely in the LSTM model can rise to the case of overfitting. Then we experimented with multiple neurons in the LSTM layer and were able to achieve a fair prediction. We missed the abrupt transition phases in the prediction as the model fails to capture the complex nature of the different phases. This is evident in the visualization that is being created in experiment-3. By now, the look-back has been kept 1. This led us to other experiments in the model architecture and introducing the Bidirectional layer and time-distribution layer. This experiment yielded us the best prediction and we observed that combining time-distribution and bidirectional assists the model to learn more about the backward and forward patterns in the dataset. Also, keeping a look-back of values of 5 yielded us best results in our experiment-4. We can conclude that in the context of LSTM, adding Bidirectional layer along with the Time - Distributed improves the ability to learn and adapt to more temporal patterns in the more complex form. Also, the cons of this are hypertuning the parameters to find the best fit. This also leads to more computation time required to perform the prediction which itself is an area of future research. The results can be summarized in below table :

Experiment No.	Description	Result	Conclusion
1	1-3-1 neural network	Cosine curve fitted	Cosine curve well with 3 hidden layers
2	Single LSTM layer with all inputs	Test RMSE: 6.347(subpart), 54.17 (complete curve)	The curve missed phase transitions, more layers, and experiments required
3	Only Surface Temperature, single LSTM later	Test RMSE: 50.12	Overfitting, train RSME low
4	Different neurons, LSTM layers, dropout layers, Dense layers	RMSE: 23.98	Missed the transitions during peak temperature
5	Added Bidirectional and Time Distributed later	RMSE: 1.1686	Best model

Table 1: Experiments summary

7. FUTURE WORK

One of the futuristic aspects of this domain's work can be application of this model architecture to different domains like manufacturing, simulation of artificial vehicles etc. The interpretability of this architecture could be used to explore networking-based models where the parameters are changing dynamically over the period of time. Since this work is based on dynamic hyperparameters, we can extend this work to the post processing of the silicon wafer and identify which step or process needs to be fixed [23] [24]. With the dynamic predictions and the post processing image of the silicon wafer, a hybrid model can be built to identify faults in the processing of the wafer. [25] [26]

Moreover, this model can be extended to a live prediction of the wafer heating mechanism which can be the base for heating anomaly detection framework in the fab industry. Many industries require a live prediction mechanism or real time prediction which can be adapted with LSTMs and time series prediction, which can be pivotal for this purpose. For example, during the reflow process of silicon wafer, it is very critical to keep the temperature to a certain threshold. This model with the look back value of 5 can predict 5 time steps before and maintain the heating lamps heat parameters below a particular threshold as learned by the model. Hence this model can be a helping hand for the lab workers and Subject Matter Experts to simulate the wafer heating patterns beforehand or while during the actual ongoing process [27] [28].

From a simulation point of view, this model can be extended with the dynamic as well as static features if enough data is at disposal [29]. The data received from other sensors such as humidity, flow rate of chemicals can enhance the predictive power of the model. The initial prototype has been built and shown in this report which makes live predictions on an input set of files [30].

8. ACCOMPLISHMENTS

1. Successfully formalized the problem statement and found the correct dataset for it.
2. Researched about the topic in depth and founded the scope of solution with respect to correct tools and language.
3. Read multiple research papers to gain knowledge regarding the problem's solution and map it with my own problem statement.
4. Learnt about Recurrent Neural Network and Long Short-Term Memory using Keras
5. Gained knowledge about use of multiple LSTMs and different layers to make the model better prediction.
6. Experimented different scenarios to deep dive into the problem statement by breaking the problem into sub-parts.
7. Achieved results for the test files with the scope of making the results better.
8. Identified areas of improvement to work during the winter break and next semester.
9. Worked on the model architecture and design with more than 20 experiments related to layers and neurons.
10. Experimented different activation functions and training sets to identify best fitting model.
11. Experimented different learning rates with each place in the model disk.
12. Combined the 4 different points into a single model.
13. Experimented different window sizes for the model output and changed the model architecture.
14. Built a real-time application to predict the next timestamps of temperature so that it can be treated as an anomaly detection framework.

REFERENCES

[1]: S. D. Kumar and D. Subha, “Prediction of Depression from EEG Signal Using Long Short Term Memory(LSTM),” 2019, pp. 1248–1253, doi: 10.1109/ICOEI.2019.8862560.

[2]: G. Zheng, Q. Ni, K. Navaie, H. Pervaiz, and C. Zarakovitis, “Efficient Pruning-Split LSTM Machine Learning Algorithm for Terrestrial-Satellite Edge Network,” 2022, pp. 307–311, doi: 10.1109/ICCWorkshops53468.2022.9814494.

[3]: D. Lee et al., “Long short-term memory recurrent neural network-based acoustic model using connectionist temporal classification on a large-scale training corpus,” vol. 14, no. 9. pp. 23–31, 2017, doi: 10.1109/CC.2017.8068761.

[4]: H. Takase, K. Gouhara, and Y. Uchikawa, “Time sequential pattern transformation and attractors of recurrent neural networks,” 1993, vol. 3, pp. 2319–2322 vol.3, doi: 10.1109/IJCNN.1993.714189.

[5]: J. K. Ryeu, H. Y. Tak, N. W. Heo, and H. S. Chung, “Recognition of Korean spoken digit using single layer recurrent neural networks,” 1993, vol. 1, pp. 267–270 vol.1, doi: 10.1109/IJCNN.1993.713908.

[6]: L. Li, S. Huang, Z. Ouyang, and N. Li, “A Deep Learning Framework for Non-stationary Time Series Prediction,” 2022, pp. 339–342, doi: 10.1109/CVIDLICCEA56201.2022.9824863.

[7]: J. Xie, “Time Series Prediction Based on Recurrent LS-SVM with Mixed Kernel,” 2009, vol. 1, pp. 113–116, doi: 10.1109/APCIP.2009.37.

[8]: J. Zhang and K. F. Man, “Time series prediction using RNN in multi-dimension embedding phase space,” 1998, vol. 2, pp. 1868–1873 vol.2, doi: 10.1109/ICSMC.1998.728168.

[9]: L. Wenya, “Cooling, Heating and Electric Load Forecasting for Integrated Energy Systems Based on CNN-LSTM,” 2021, pp. 808–812, doi: 10.1109/ICPRE52634.2021.9635244.

[10]: S. Biswas, “Stock Price Prediction using Bidirectional LSTM with Attention,” 2022, pp. 1–5, doi: 10.1109/ICAIC53980.2022.9896969.

[11]: D. Kim, M. Kim, and W. Kim, “Wafer Edge Yield Prediction Using a Combined Long Short-Term Memory and Feed- Forward Neural Network Model for Semiconductor Manufacturing,” vol. 8. pp. 215125–215132, 2020, doi: 10.1109/ACCESS.2020.3040426.

[12]: G. Cybenko, "Approximation by superpositions of a sigmoidal function," Mathematics of Control, Signals, and Systems, vol. 2, no. 4, pp. 303-314, Dec. 1989.

[13]: K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural Networks, vol. 4, no. 2, pp. 251-257, Mar. 1991.

[14]: A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," IEEE Transactions on Information Theory, vol. 39, no. 3, pp. 930-945, May 1993.

[15]: S. Haykin, Neural networks: A comprehensive foundation, 2nd ed. Prentice Hall, 1999.

[16]: I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT Press, 2016.

[17]: G. Cybenko, "Neural networks and their applications," Proceedings of the IEEE, vol. 82, no. 10, pp. 1580-1587, Oct. 1994.

[18]: K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks, vol. 2, no. 5, pp. 359-366, 1989.

[19]: R. van der Meijden, “Digital twin: Manufacturing excellence through virtual factory replication,” IFAC-PapersOnLine, vol. 50, no. 1, pp. 14806–14811, 2017.

[20]: L. Xu, Y. Liu, and H. Hao, “Digital twin driven predictive maintenance for intelligent manufacturing,” Journal of Intelligent Manufacturing, vol. 32, no. 1, pp. 113–126, 2021.

[21]: H. Wu, S. Zhu, and J. Li, “Digital twin-based simulation and optimization for the energy Internet of things,” IEEE Access, vol. 9, pp. 6001–6012, 2021.

[22]: M. Shen, H. Wang, and Y**. Liu, "Digital** twin driven healthcare: A survey," Journal of Medical Systems, vol. 44, no. 7, pp. 1–14, 2020.

[23]: X. Meng, X. Ma, X. Wei and Z. Zeng, "Heat transfer simulation for radiant floor heating system with air-source heat pump," 2011 International Conference on Electric Technology and Civil Engineering (ICETCE), Lushan, China, 2011, pp. 3198-3201, doi: 10.1109/ICETCE.2011.5776200.

[24]: Junxia and Ruili, "Numerical simulation of indoor comfort in heating room with local radiant floor," 2011 International Conference on Electrical and Control Engineering, Yichang, China, 2011, pp. 6017-6020, doi: 10.1109/ICECENG.2011.6058287.

[25]: Dan Wang, Qiwu Dong and Minshan Liu, "Numerical simulation research on larger industrial heat exchanger," *2011 Second International Conference on Mechanic Automation and Control Engineering*, Hohhot, 2011, pp. 4199-4202, doi: 10.1109/MACE.2011.5987929.

[26]: K. Adam, K. Smagulova and A. P. James, "Memristive LSTM network hardware architecture for time-series predictive modeling problems," *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Chengdu, China, 2018, pp. 459-462, doi: 10.1109/APCCAS.2018.8605649.

[27]: A. Wang and C. Ren, "Prediction of receiving field strength based on SVM-LSTM hybrid model in the coal mine," *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, Beijing, China, 2021, pp. 813-816, doi: 10.1109/CISCE52179.2021.9445892.

[28]: H. Deng *et al.*, "Electricity Price Prediction Based on LSTM and LightGBM," *2021 IEEE 4th International Conference on Electronics and Communication Engineering (ICECE)*, Xi'an, China, 2021, pp. 286-290, doi: 10.1109/ICECE54449.2021.9674719.

[29]: D. Zhi-hong, Z. Song, L. Zi-fan, R. Chang, Y. Zhi-feng and W. Bin, "Sensor Fault Diagnosis Based on Wavelet Analysis and LSTM Neural Network," *2022 IEEE 20th International Power Electronics and Motion Control Conference (PEMC)*, Brasov, Romania, 2022, pp. 249-255, doi: 10.1109/PEMC51159.2022.9962935.

[30]: S. Becker, F. Durst and H. Lienhart, "LDA system for in-flight local velocity measurements on airplane wings," *ICIASF 99. 18th International Congress on Instrumentation in Aerospace Simulation Facilities. Record (Cat. No.99CH37025)*, Toulouse, France, 1999, pp. 25/1-25/7, doi: 10.1109/ICIASF.1999.827165.