Solar Irradiance Forecasting via Artificial Neural Networks (ANN)

Mohammad F. El Hajj Chehade – E4 ECE

Taha Koleilat – E4 CCE

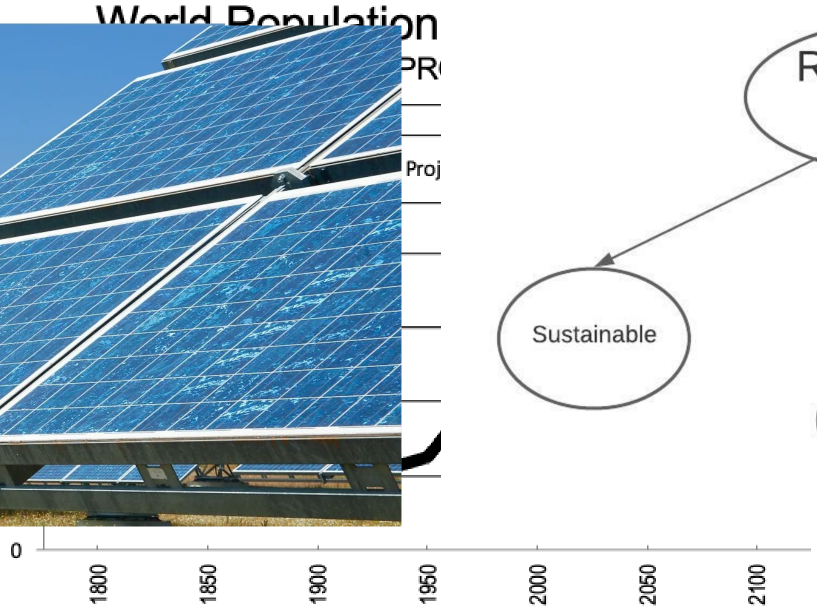
ENMG 616 – Advanced Optimization & Techniques
American University of Beirut

Nov 16, 2022

Outline

- Introduction
  - Motivation
  - Objective
- Problem formulation
  - ANN
  - Optimization Problem
- Proposed Methodology
  - Weights Initialization
  - Backpropagation
  - Iterative Descent Methods
  - Gradient Clipping
  - Data Pre-processing
- Convergence Analysis and Fine-tuning
- Discussion of results
  - Comparison between Models
  - Prediction on a recent weather sample

Motivation



Sources: United Nations World Population Prospects, 2022 & US Census Bureau

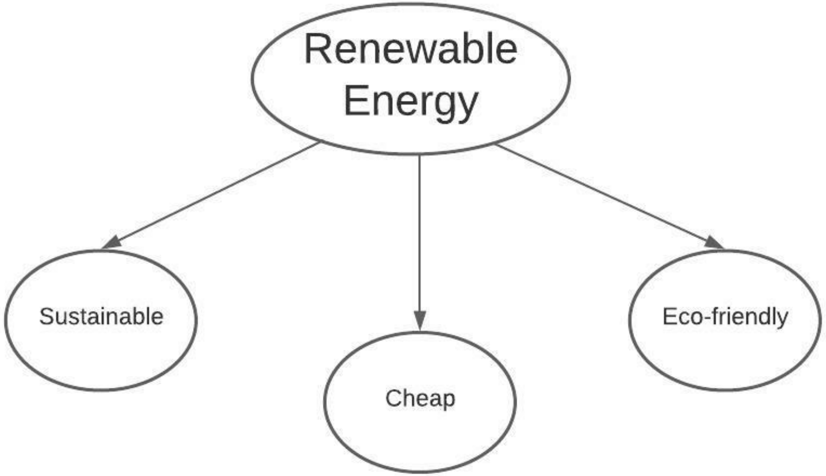
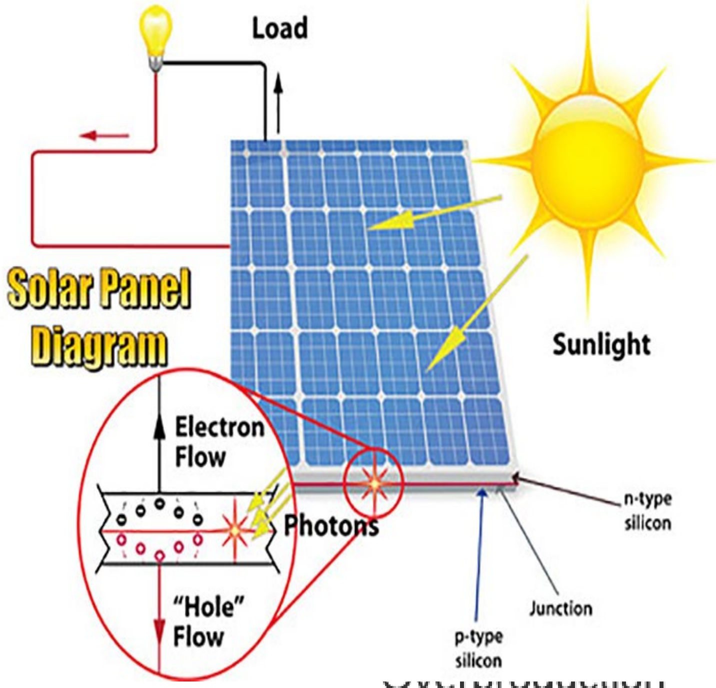
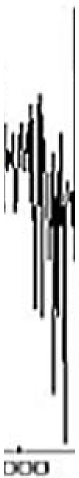
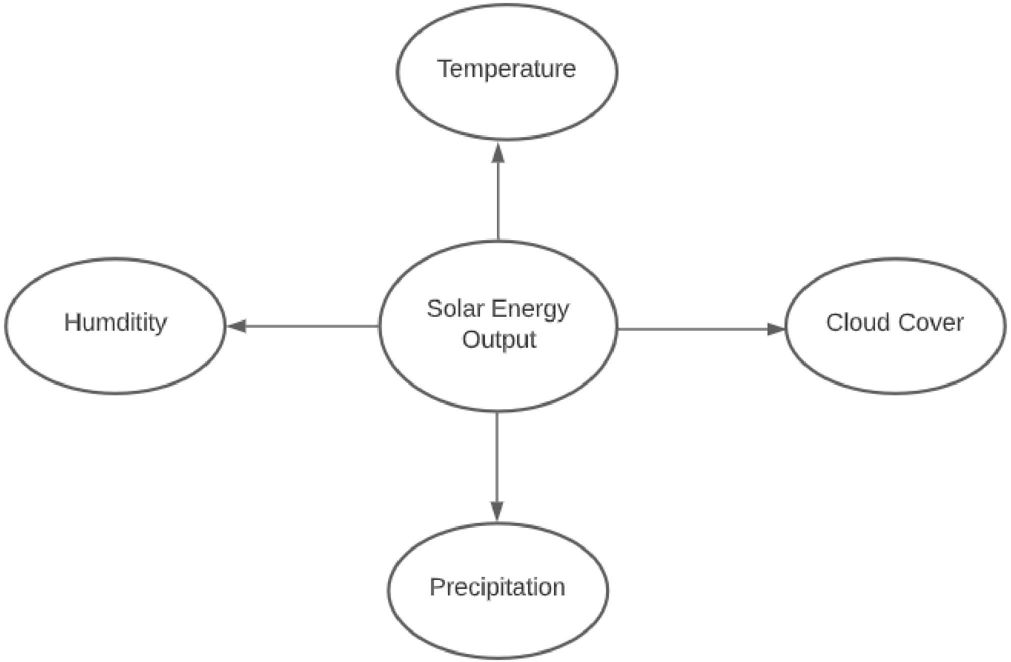


Figure 1

Photovoltaic Solar Technology

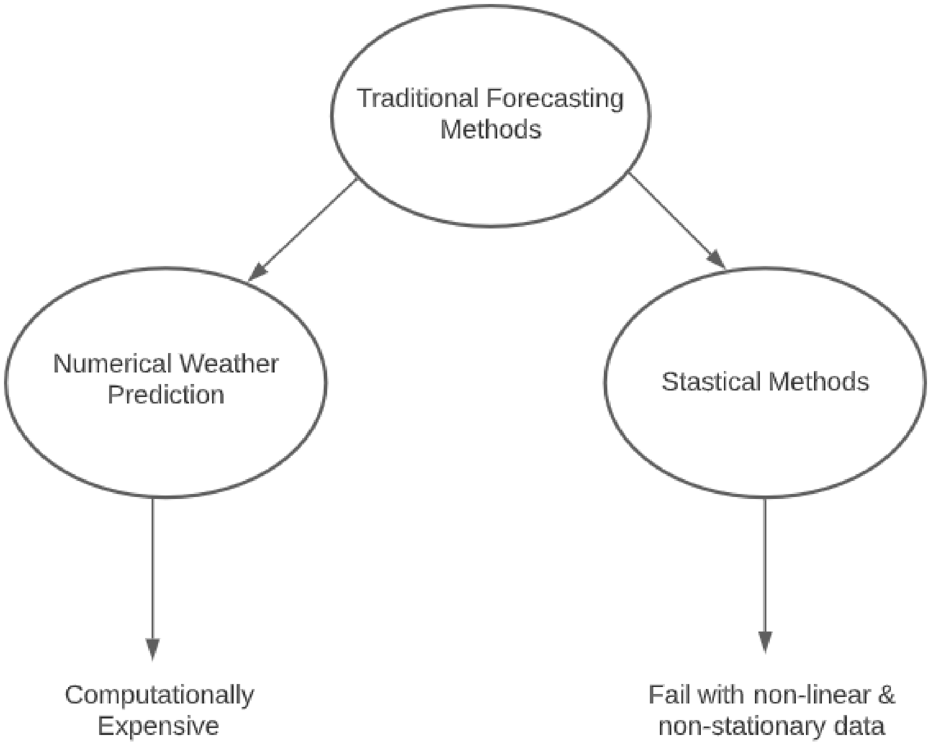




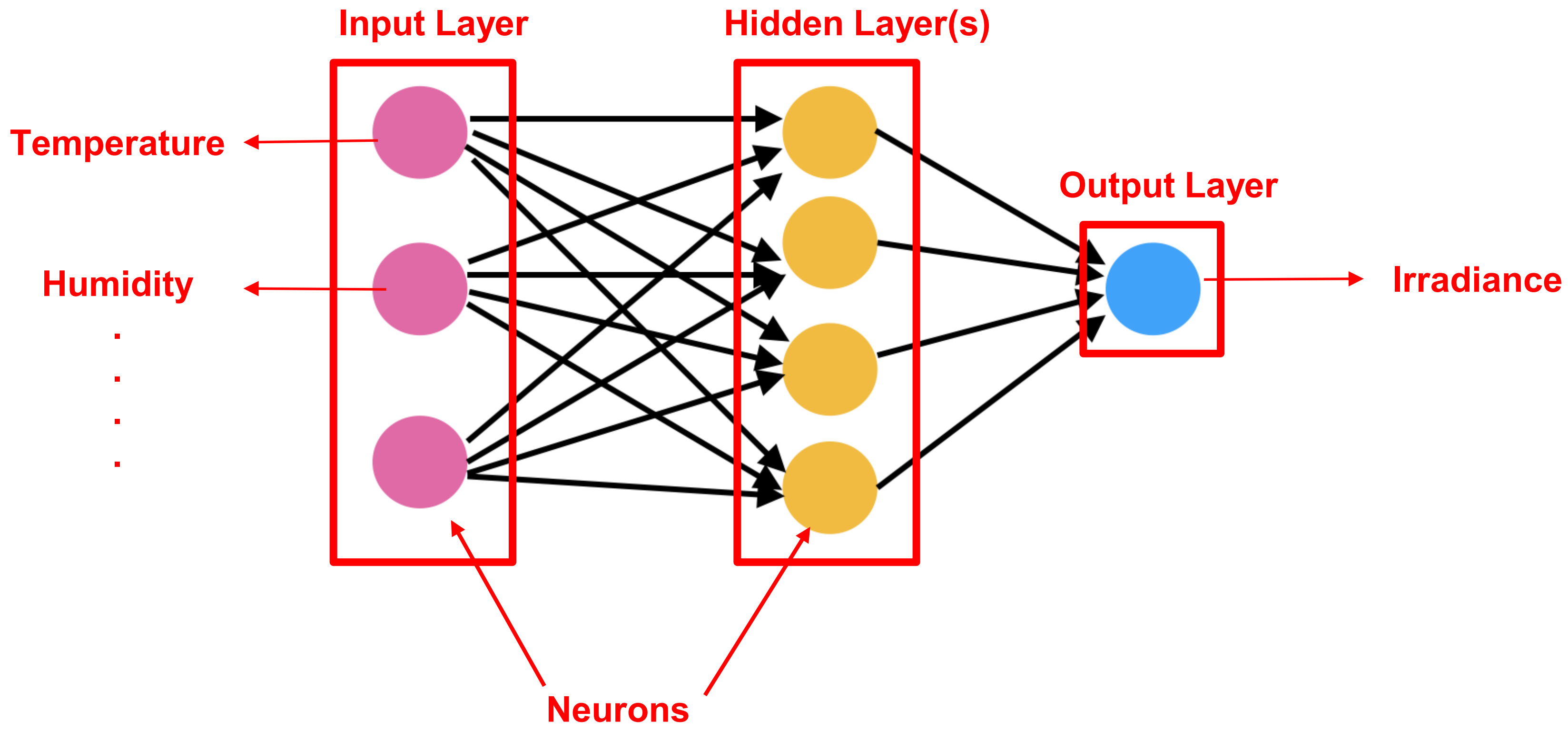


Underproduction → Outages

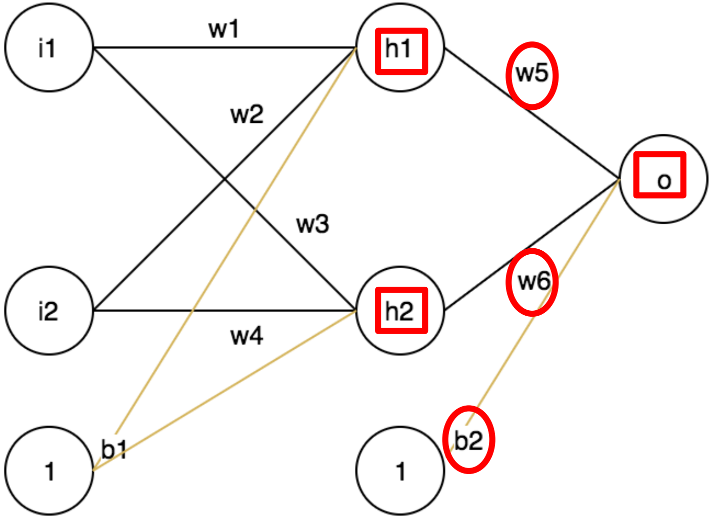
Solar Irradiance Forecasting



Artificial Neural Networks (ANN)



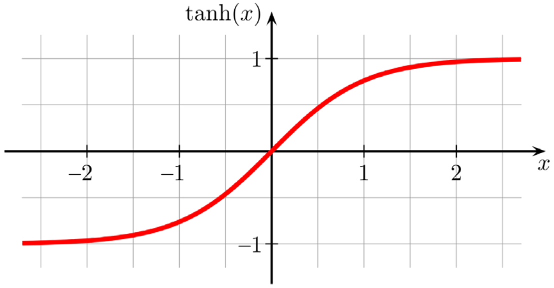
Forward Propagation



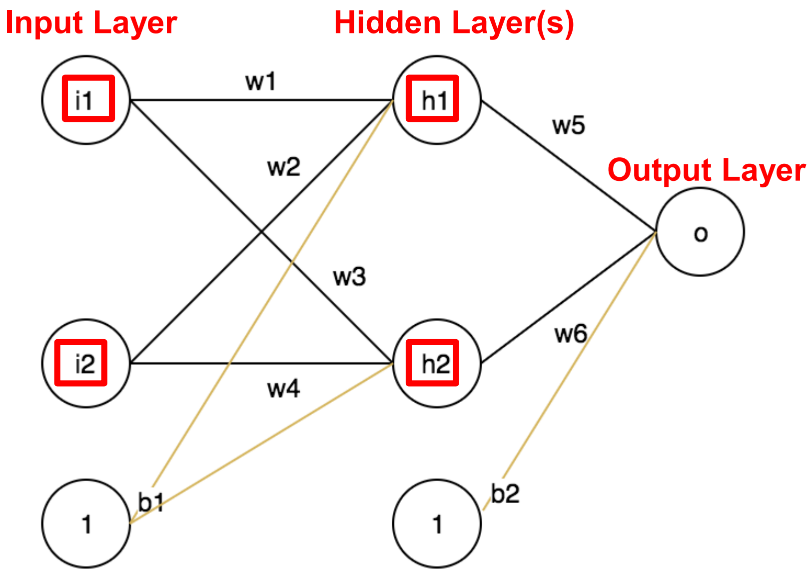
$$o = f(\underline{w_5 h_1 + w_6 h_2 + b_2})$$

activation function

tanh **sigmoid**



Forward Propagation (cont'd)



$$h1 = f(w_1 i_1 + w_2 i_2 + b_1)$$

$$h2 = f(w_3 i_1 + w_4 i_2 + b_1)$$

$$o = f(w_5 h_1 + w_6 h_2 + b_2)$$

Objective Function

MSE Loss Function

$$\underbrace{\min_{w,b}}_{\text{parameters of the network}} \frac{1}{n}\sum_{i=1}^{n}\Big(\underbrace{y_{predicted_i}}_{\text{predicted 5 minutes before t}} - \underbrace{y_{actual_i}}_{\text{actual measured value at instance t}}\Big)^2 \longrightarrow \text{convex}$$

pyrheliometer

forecasted values for weather parameters → Neural Network → forecasted value for irradiance

Gradient Computation

$$\nabla L = \left(\frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_i}, \dots, \frac{\partial L}{\partial b_1}, \dots, \frac{\partial L}{\partial b_i}, \dots\right)$$ → **Gradient Vector**

$$L(w, b) = \frac{1}{n}\sum_{i=1}^{n}\left(\underline{y_{predicted_i}} - y_{actual_i}\right)^2$$ → **Loss Function**

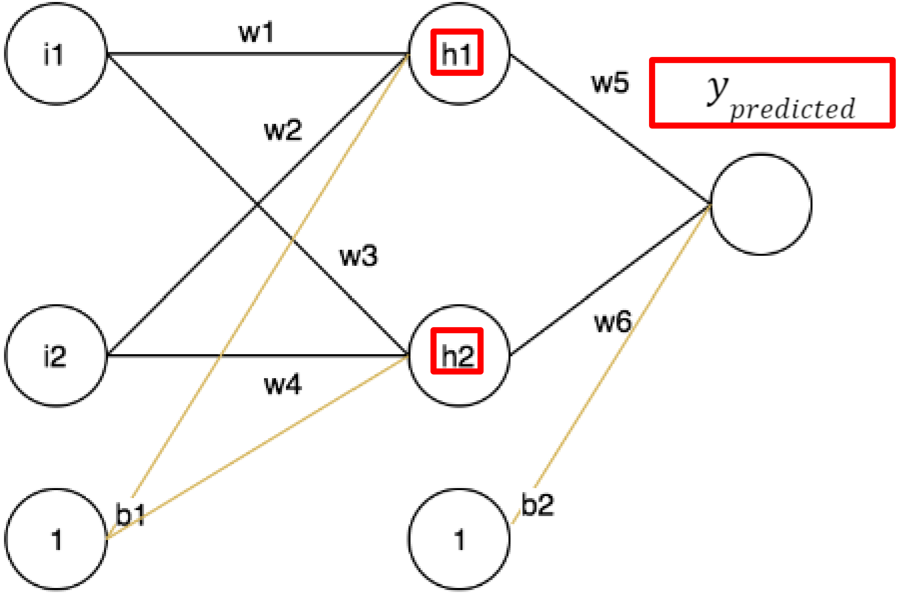
depends on w & b

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y_{predicted}} \frac{\partial y_{predicted}}{\partial w_i}$$ → **Neural Network relations**

$$\frac{\partial L}{\partial y_{predicted}} = \frac{1}{2n}\sum_{i=1}^{n}\left(y_{predicted_i} - y_{actual_i}\right)$$

Back Propagation

GOAL: Find $\frac{\partial y_{predicted}}{\partial w_i}$



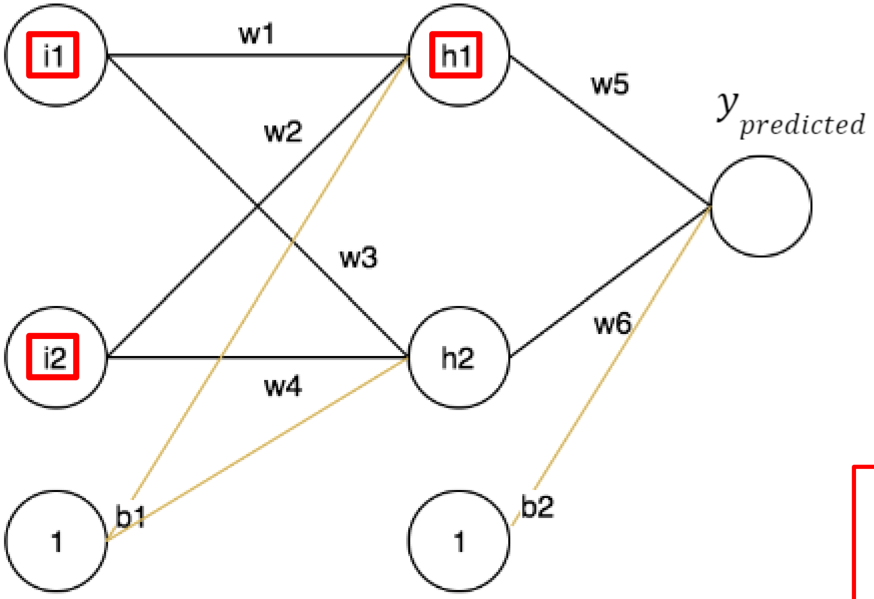
$$y_{predicted} = tanh(\underbrace{w_5 h_1 + w_6 h_2 + b_2}_{z})$$

$$y_{predicted} = tanh(z)$$

$$\frac{\partial y_{predicted}}{\partial w_5} = \frac{\partial tanh(z)}{\partial z} \frac{\partial z}{\partial w_5} = tanh'(z) \times h_1$$

Back Propagation (cont'd)

GOAL: Find $\frac{\partial y_{predicted}}{\partial w_i}$



$$h1 = tanh(w_1 i_1 + w_2 i_2 + b_1)$$

$$\boxed{\frac{\partial h_1}{\partial w_1} = \frac{\partial tanh(z)}{\partial z}\frac{\partial z}{\partial w_1} = tanh'(z) \times i_1}$$

$$y_{predicted} = tanh(w_5 h_1 + w_6 h_2 + b_2)$$

$$\boxed{\frac{\partial y_{predicted}}{\partial h_1} = \frac{\partial tanh(z)}{\partial z}\frac{\partial z}{\partial h_1} = tanh'(z) \times w_5}$$

$$\frac{\partial y_{predicted}}{\partial w_1} = tanh'(z) \times w_5 \times tanh'(z) \times i_1$$

Gradient Descent

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y_{predicted}} \frac{\partial y_{predicted}}{\partial w_i}$$

back propagation

differentiating the loss function

$$\nabla L = \left(\frac{\partial L}{\partial w_1}, \ldots, \frac{\partial L}{\partial w_i}, \ldots, \frac{\partial L}{\partial b_1}, \ldots, \frac{\partial L}{\partial b_i}, \ldots\right)$$

gradient vector

$$w \leftarrow w - \alpha \nabla L$$

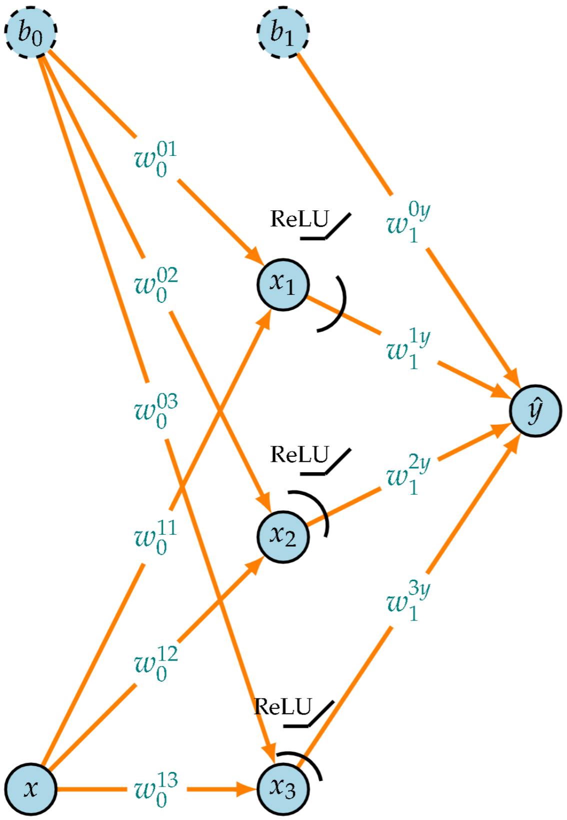
update rule

Weights Initialization

- Speeds up convergence
- Mitigates Exploding and Vanishing gradient problems
- Takes into account the weights' variance
- Utilized Weights Initialization proposed by Kaiming He et al.

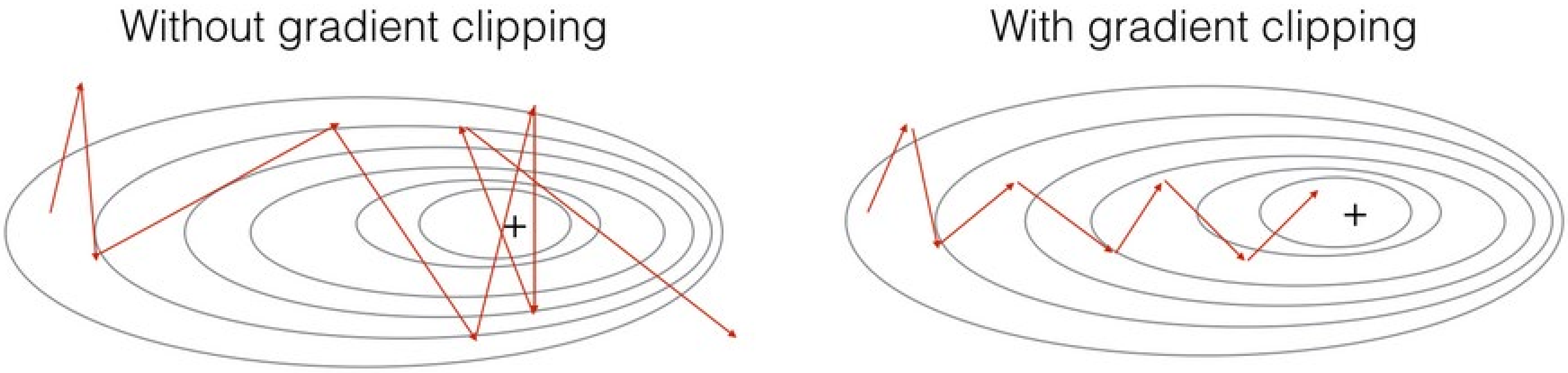
$$W \sim \mathcal{N}\left(0, \frac{2}{n^l}\right)$$

Kaiming Initialization



Gradient Clipping

- Trim a gradient if it grows too large in order to maintain an acceptable value
- If $\|\nabla L\| \geq T$, then $\|\nabla L\| \leftarrow T$
- Ensures that the update for the parameters isn't too large to cause divergence



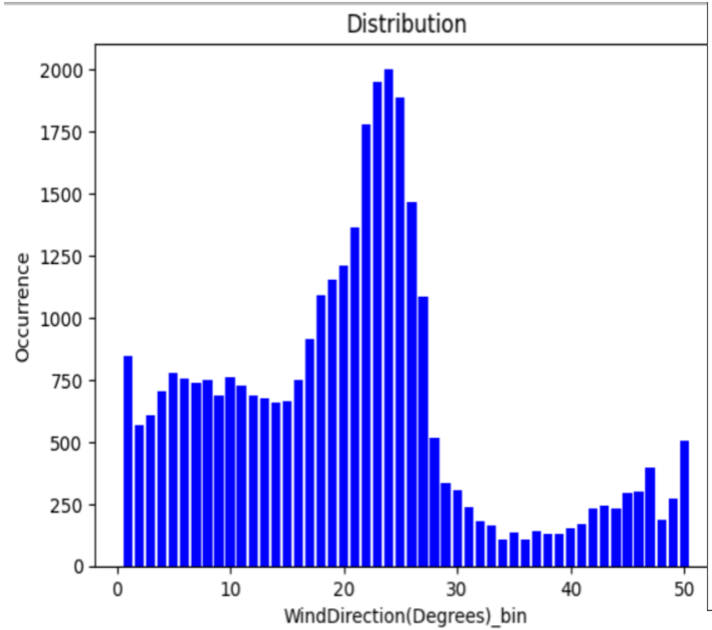
Data Pre-processing

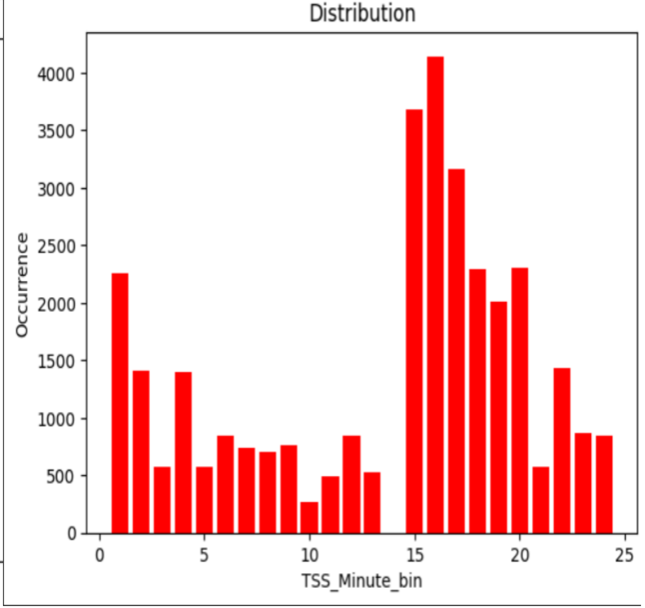
	Data	Time	Temperature	Pressure	Humidity	WindDirection(Degrees)	Speed	TimeSunRise	TimeSunSet
0	9/29/2016 12:00:00 AM	23:55:26	48	30.46	59	177.39	5.62	06:13:00	18:13:00
1	9/29/2016 12:00:00 AM	23:50:23	48	30.46	58	176.78	3.37	06:13:00	18:13:00
2	9/29/2016 12:00:00 AM	23:45:26	48	30.46	57	158.75	3.37	06:13:00	18:13:00
3	9/29/2016 12:00:00 AM	23:40:21	48	30.46	60	137.71	3.37	06:13:00	18:13:00
4	9/29/2016 12:00:00 AM	23:35:24	48	30.46	62	104.95	5.62	06:13:00	18:13:00
...	...	...	...	...	...	...	...	...	...
32681	12/1/2016 12:00:00 AM	00:20:04	44	30.43	102	145.42	6.75	06:41:00	17:42:00
32682	12/1/2016 12:00:00 AM	00:15:01	44	30.42	102	117.78	6.75	06:41:00	17:42:00
32683	12/1/2016 12:00:00 AM	00:10:01	44	30.42	102	145.19	9.00	06:41:00	17:42:00
32684	12/1/2016 12:00:00 AM	00:05:02	44	30.42	101	164.19	7.87	06:41:00	17:42:00
32685	12/1/2016 12:00:00 AM	00:00:02	44	30.43	101	83.59	3.37	06:41:00	17:42:00

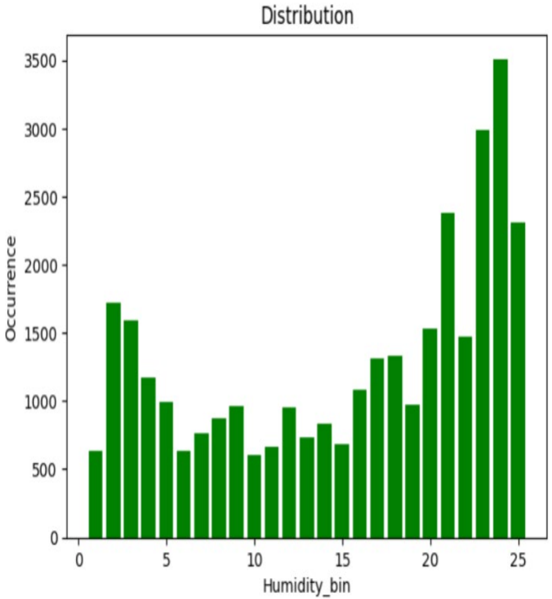
32686 rows × 9 columns

	Radiation
0	1.21
1	1.21
2	1.23
3	1.21
4	1.17
...	...
32681	1.22
32682	1.17
32683	1.20
32684	1.23
32685	1.20

32686 rows × 1 columns


Distribution
Occurrence
WindDirection(Degrees)_bin
0
250
500
750
1000
1250
1500
1750
2000
0
10
20
30
40
50

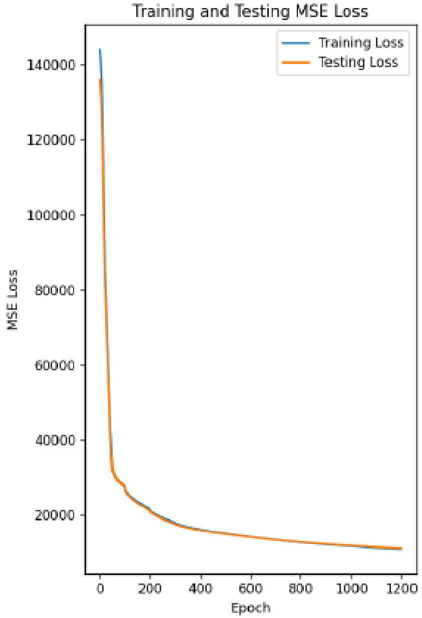

Distribution
Occurrence
TSS_Minute_bin
0
500
1000
1500
2000
2500
3000
3500
4000
0
5
10
15
20
25

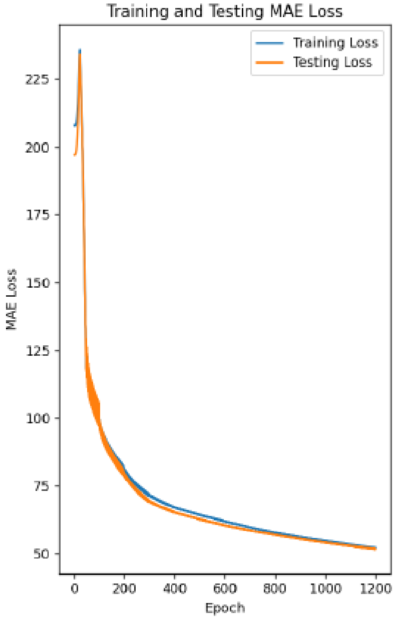

Distribution
Occurrence
Humidity_bin
0
500
1000
1500
2000
2500
3000
3500
0
5
10
15
20
25

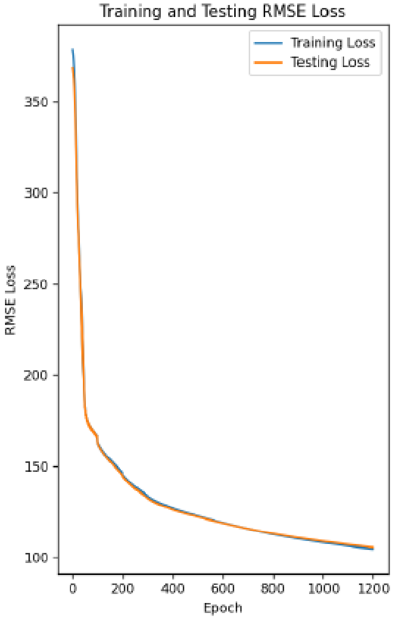
Convergence Analysis and Fine-tuning

- Full Batch Gradient Descent
- Diminishing step size
- 0.001 Learning Rate
- 1200 epochs

Set	MSE	MAE	RMSE
Training Set	10,919.83	N/A	N/A
Testing Set	11,119.12	51.67	105.44



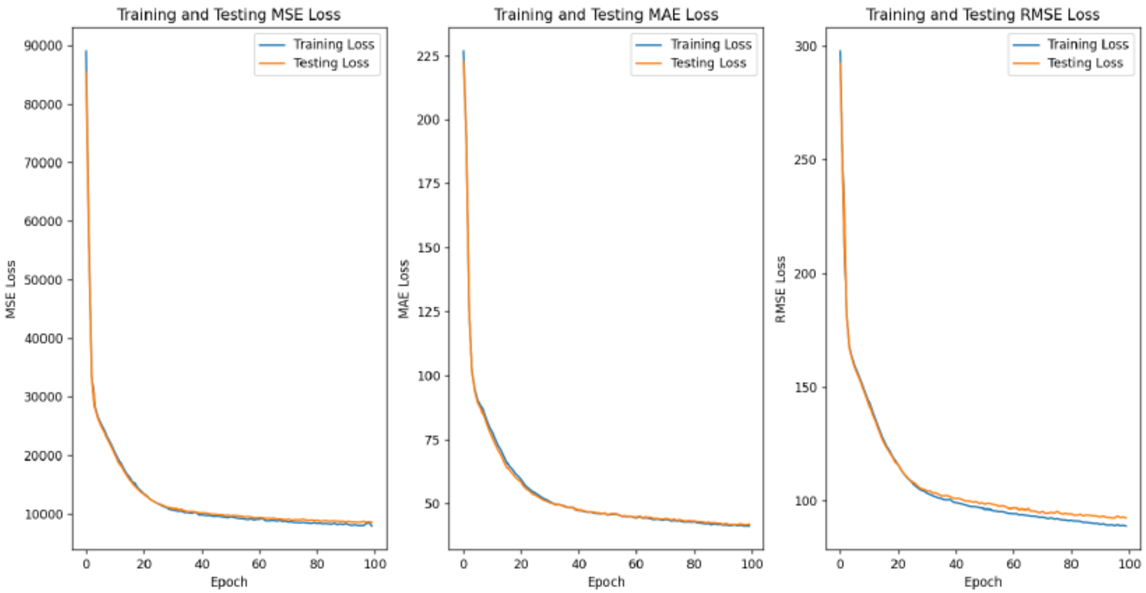




Convergence Analysis and Fine-tuning (cont'd)

- Mini Batch Gradient Descent
- 0.0001 Learning Rate
- 128 Batch Size
- 100 epochs

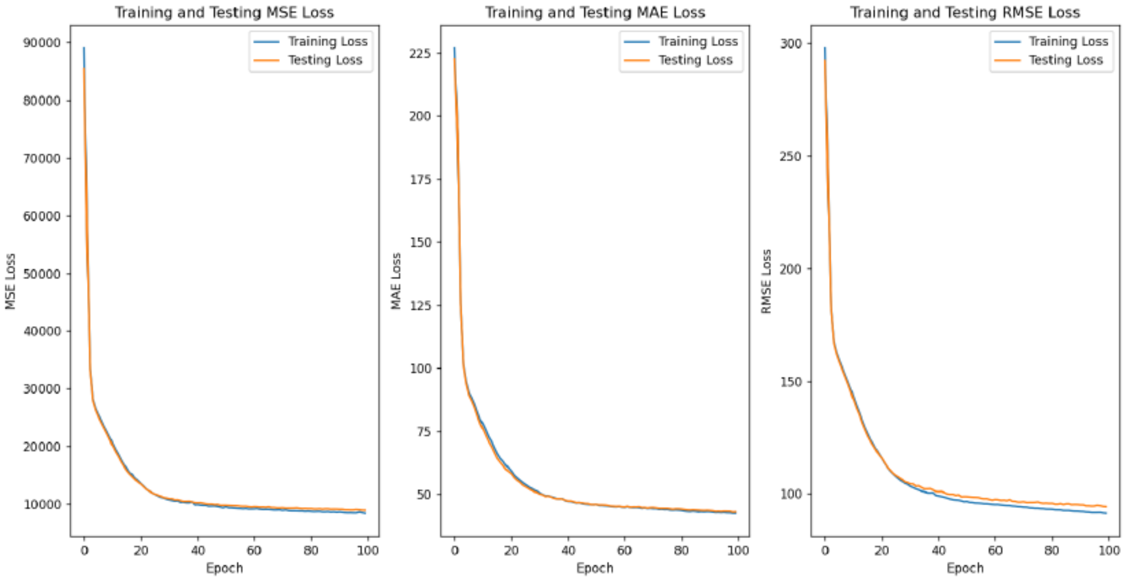
Set	MSE	MAE	RMSE
Training Set	7806.26	N/A	N/A
Testing Set	8559.51	41.79	92.52



Convergence Analysis and Fine-tuning (cont'd)

- Mini Batch Gradient Descent
- 0.001 Learning Rate
- Diminishing Step Size
- 128 Batch Size
- 100 epochs

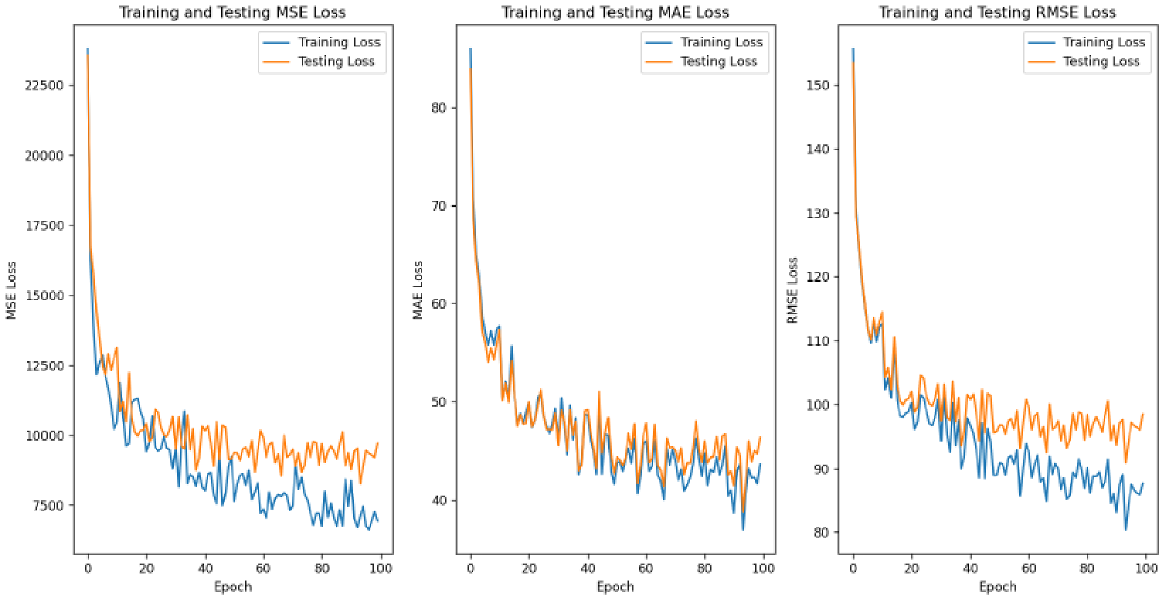
Set	MSE	MAE	RMSE
Training Set	8351.33	N/A	N/A
Testing Set	8887.98	43.08	94.27



Convergence Analysis and Fine-tuning (cont'd)

- Mini Batch Gradient Descent
- 0.001 Learning Rate
- 128 Batch Size
- 100 epochs

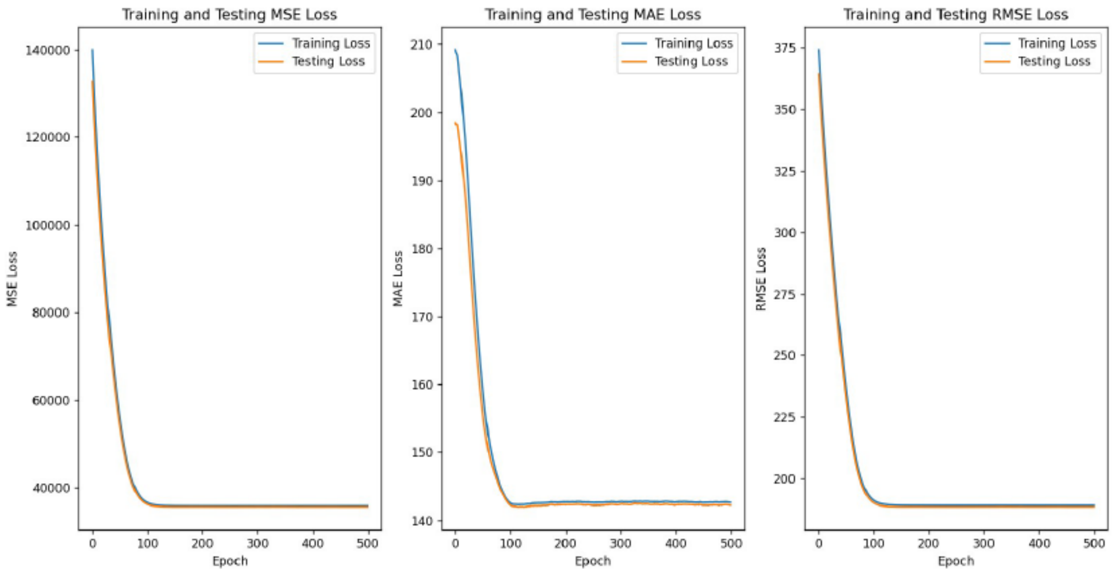
Set	MSE	MAE	RMSE
Training Set	6936.05	N/A	N/A
Testing Set	9705.06	46.37	98.52



Comparison between Models

- Mini Batch Gradient Descent
- 0.001 Learning Rate
- 32 Batch Size
- 100 epochs

Models	MSE	MAE	RMSE
Neural Network SGD	8752.26	40.83	93.55
Neural Network GD	11119.12	51.67	105.44
Linear Regression SGD	35658.31	141.90	188.83



Prediction on a recent weather sample

Sat 12 | Day

54°

Record High	Average High
86°	**69°**
Sunrise	Sunset
6:56 am	**5:29 pm**

Sat 12 | Night

31°

Record Low	Average Low
23°	**49°**
Moonrise	Moonset
9:40 pm	**11:55 am**

Waning Gibbous

Prediction on a recent weather sample (cont'd)

Predicting solar irradiance acccording to weather forecast in Dallas, Texas

```
X = pd.DataFrame(data={"Data": '12/11/2022 12:00:00 PM', "Time":'10:18:00', "Temperature":54, "Pressure":29.97, "Humidity":41, "WindDirection(Degrees)":0,
        "Speed":15, "TimeSunRise":'06:54:00', "TimeSunSet":'17:27:00'},index=[0])
X['TSR_Minute'] = pd.to_datetime(X['TimeSunRise']).dt.minute
X['TSS_Minute'] = pd.to_datetime(X['TimeSunSet']).dt.minute
X['TSS_Hour'] = np.where(pd.to_datetime(X['TimeSunSet']).dt.hour==18, 1, 0)

X['Month'] = pd.to_datetime(X['Data']).dt.month
X['Day'] = pd.to_datetime(X['Data']).dt.day
X['Hour'] = pd.to_datetime(X['Time']).dt.hour
X['Minute'] = pd.to_datetime(X['Time']).dt.minute
X['Second'] = pd.to_datetime(X['Time']).dt.second
X = X.drop(['Data', 'Time', 'TimeSunRise', 'TimeSunSet'], axis=1)
X['WindDirection(Degrees)_bin'] = np.digitize(X['WindDirection(Degrees)'], np.arange(0.0, 1.0, 0.02).tolist())
X['TSS_Minute_bin'] = np.digitize(X['TSS_Minute'], np.arange(0.0, 288.0, 12).tolist())
X['Humidity_bin'] = np.digitize(X['Humidity'], np.arange(32, 3192, 128).tolist())

X = StandardScaler().fit_transform(X)
np.squeeze(net1.predict(X))
```

32]

```
array(600.)
```

Prediction on a recent weather sample (cont’d)