

# ID2223 Project

Salman Niazi and Shadi Issa

Jan 10, 2017

# Project Description

- ▶ Predict the solar radiation near Earth surface

## Data Samples

- ▶ **0.4 million samples**
- ▶ A typical sample looks like

lev	p	T	q	lwhr
0	19.231	-80.0	0.0	0.122
1	57.692	-80.0	0.0	0.451
2	96.154	-70.874	0.029	-1.229
3	134.615	-51.083	0.262	-2.732
4	173.077	-36.489	0.977	-3.429
5	211.538	-25.816	2.211	-3.574
6	250.0	-17.87	3.756	-3.536
7	288.462	-10.404	5.431	-3.802
8	326.923	-6.608	4.226	-2.198
9	365.385	-2.388	8.776	-4.203
10	403.846	1.264	10.375	-3.567
11	442.308	4.462	11.895	-3.146
12	480.769	7.318	13.347	-2.829
13	519.231	9.903	14.733	-2.598

# Data Samples

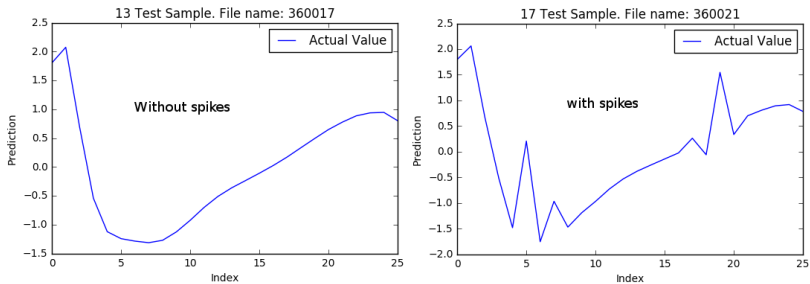


Figure 1: Data Samples

# Solution

- ▶ Regression Problem
  - ▶ with 26 outputs
- ▶ Could be implemented using
  - ▶ Multivariate Regression
  - ▶ **Feed Forward Neural Networks**
  - ▶ **Convolution Neural Networks**

# Feedforward Neural Network

- ▶ Feedforward neural networks with a single hidden layer can approximate continuous functions
- ▶ This can be efficient to replace an analytical model

# Feedforward Neural Network Model

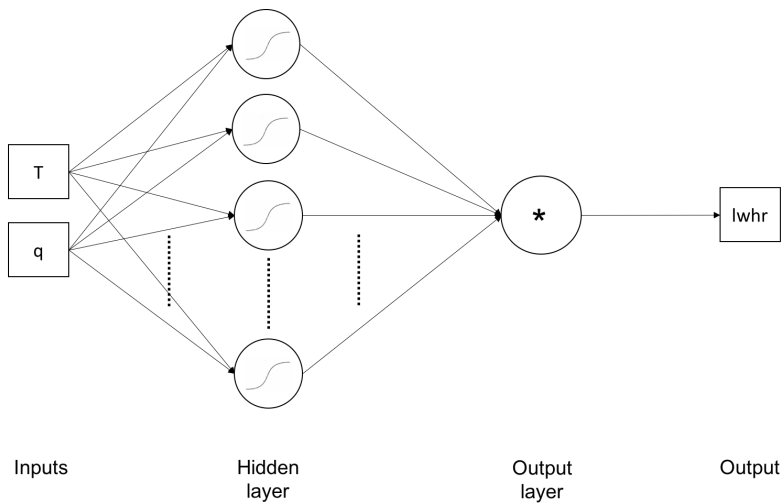


Figure 2: Architecture of feedforward neural network

# Feedforward Neural Network Setup

- ▶ Training data set size 1,400,000 (70%).
- ▶ Test data set size 600,000 (30%).
- ▶ Max number of Epochs 14000
- ▶ Batch Size 100
- ▶ Weights and biases are initialized to zeros



# Evaluation

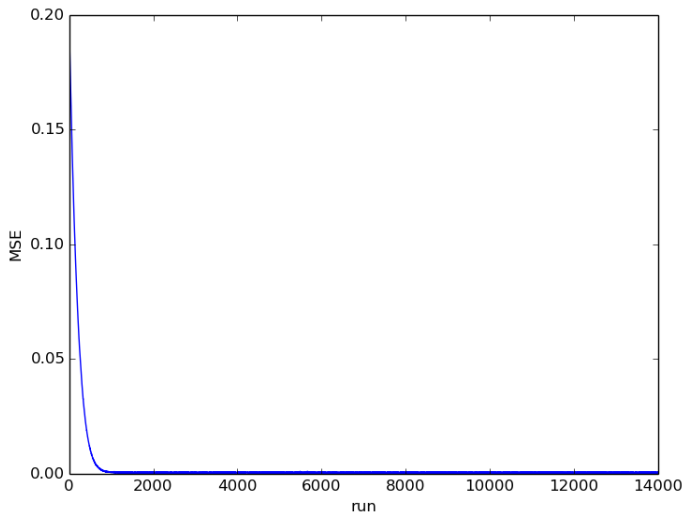


Figure 3: MSE of the FFN

# Evaluation

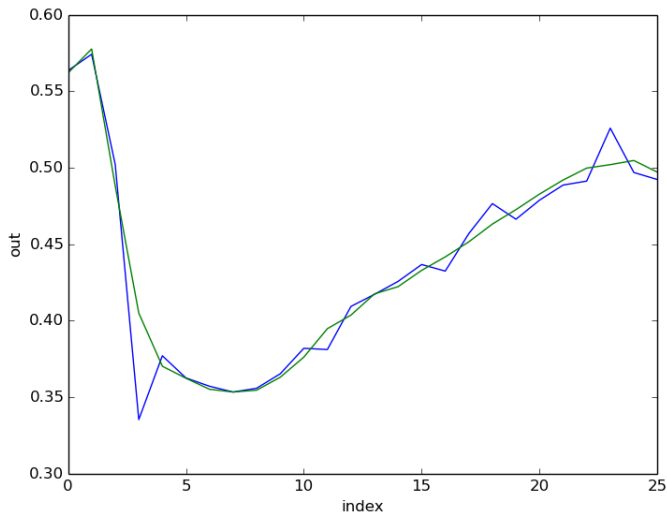


Figure 4: MSE of the FFN

# Discussion

- ▶ low MSE
- ▶ does not capture the spikes

# Convolution Neural Network

- ▶ Spikes are more affected by adjacent values
- ▶ To try to capture the spikes we opt to CNNs
- ▶ Kernels within CNN can detect local patterns

# Input

- ▶ The input can be morphed into  $8 \times 8$  matrix
  - ▶ padding is needed as there are only 52 input features

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	0	0	0	0
0	0	0	0	0	0	0	0

Figure 5:  $8 \times 8$  Input Matrix

# Convolution Neural Network Model

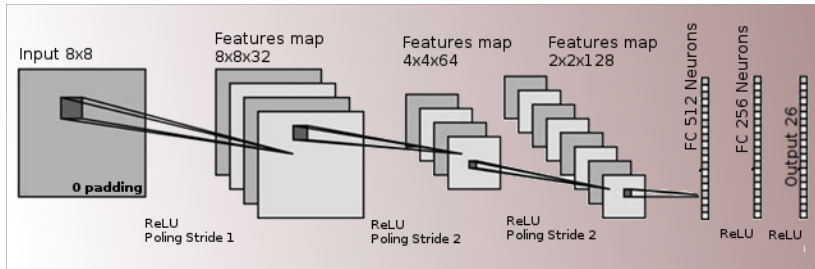


Figure 6: Architecture of convolution neural network

## Model Complexity

Layer	Size	Memory	Weights	Bias
Input	8x8x1	64	0	0
CONV	8x8x32	$8 \times 8 \times 32 = 2048$	$2 \times 2 \times 1 \times 32 = 128$	32
POOL	8x8x32	$8 \times 8 \times 32 = 2048$	0	0
CONV	8x8x64	$8 \times 8 \times 64 = 4096$	$2 \times 2 \times 1 \times 64 = 256$	64
POOL	4x4x64	$4 \times 4 \times 64 = 512$	0	0
CONV	4x4x128	$4 \times 4 \times 128 = 2048$	$2 \times 2 \times 1 \times 128 = 512$	128
POOL	2x2x128	$2 \times 2 \times 128 = 512$	0	0
FC	1x512	512	$2 \times 2 \times 128 \times 512 = 262144$	512
FC	1x256	256	$512 \times 256 = 131072$	256
OUT	1x26	26	$26 \times 256 = 6656$	26

**Total memory =  $413908 \times 4$  bytes (*float32*)  $\times 2$  (back propagation) =  $3311264 = 3.1$  Megabytes**

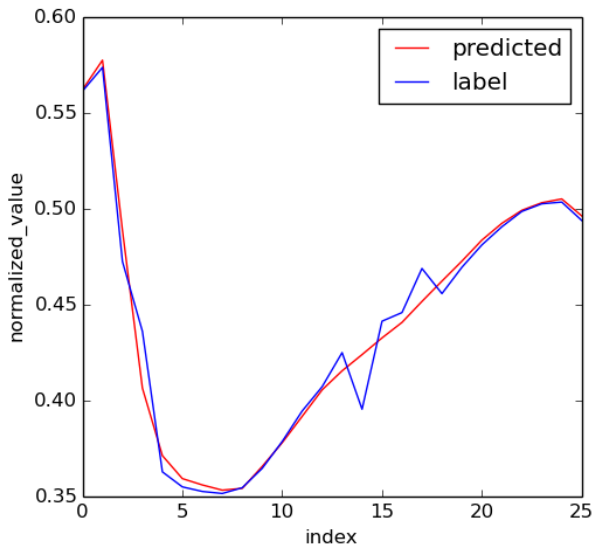
# Evaluation Setup

- ▶ Training data set size 360,000.
- ▶ Test data set size 40,000.
- ▶ Inputs are normalized using max-min scaling
  - ▶  $X_{\text{norm}} = (X - X_{\text{min}}) / (X_{\text{max}} - X_{\text{min}})$
  - ▶  $X_s = (X - \text{Input}_{\text{mean}}) / (\text{Input}_{\text{std}})$
- ▶ Learning Rate 0.001
- ▶ Dropout 0.95
- ▶ Max number of Epochs 120000
- ▶ Batch Size 3
- ▶ Weights were randomly initialized such that the random numbers had *mean*=0.1 and *stddev*=0.3
- ▶ Bias were also randomly initialized such that the random numbers had *mean*=0 and *stddev*=0.03



## Results

- MSE drops to 0.003 but the network failed to predict the spikes



# Dying ReLU Problem

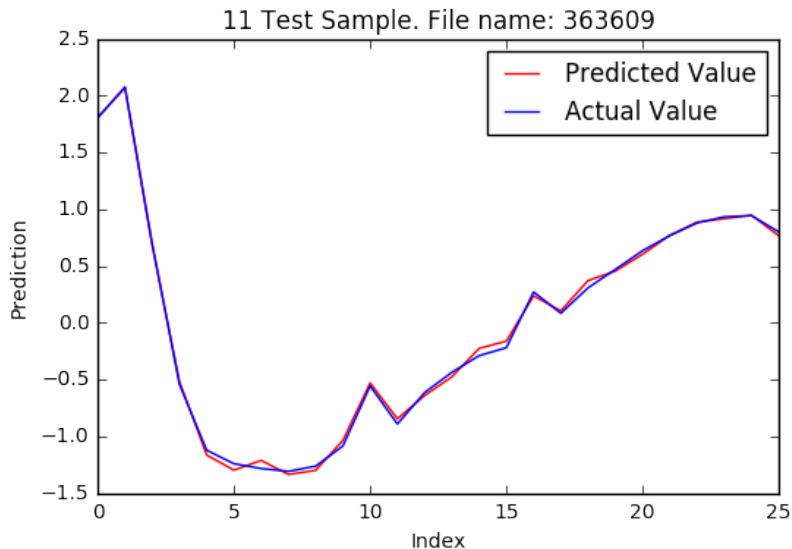
- ▶ Inaccurate data.
  - ▶ Rounding Errors
- ▶ “ReLU units can be fragile during training and can “die”. For example, a large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any datapoint again.”<sup>1</sup>

---

<sup>1</sup><http://cs231n.github.io/neural-networks-1/>

# Solution Leaky ReLU

- ▶ Use a Leaky ReLU
  - ▶ Slope 0.001



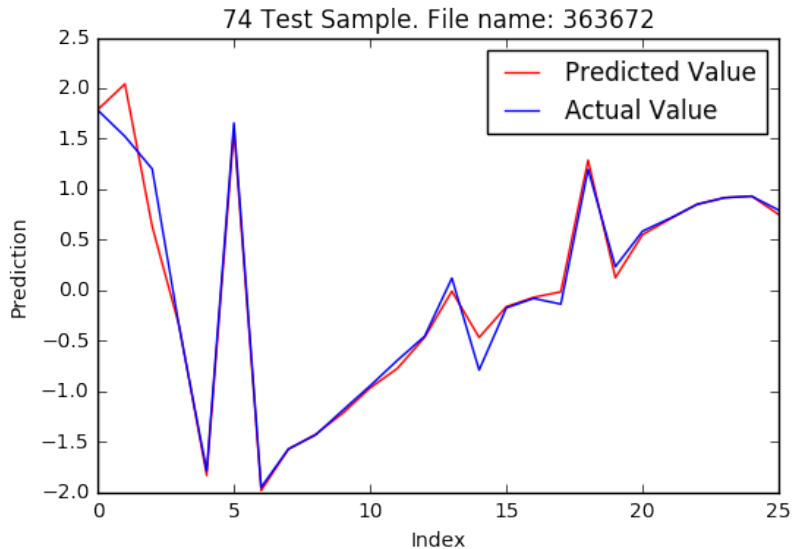


Figure 9: Sample output

Questions ?