Application of Neural Network Techniques for Modeling of Blast Furnace Parameters

by

Anjali Dhond

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 19, 2000

© Copyright 2000 M.I.T. All rights reserved.
Application of Neural Network Techniques for Modeling of Blast Furnace Parameters
by
Anjali Dhond

Submitted to the
Department of Electrical Engineering and Computer Science

May 19, 2000

In Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT:
This thesis discusses the predictions of various output variables in a blast furnace. It compares the ability of multi-layer perceptron neural networks for prediction with other blast furnace prediction techniques. The output variables: Hot Metal Temperature, Silicon Content, Slag Basicity, RDI, and +10 are all modeled using the MLP networks. Different solutions are proposed for preprocessing the original data and finding the most relevant input variables. The NNRUN software is used to find the best MLP neural network. Finally, methods to control the output variables in the blast furnace are examined and a derivative-based sensitivity analysis is discussed.

Thesis Supervisor: Dr. Nishikant Sonwalkar
Title: Director, Hypermedia Teaching Facility
The MIT Center for Advanced Educational Services (CAES)
Acknowledgments

I would like to thank my parents, Ramakant and Maneesha Dhond, my brother, Sameer Dhond, and the rest of my family for constant support and encouragement.

I would also like to thank Dr. Amar Gupta and Dr. Nishikant Sonwalkar for supervising the project and providing direction and guidance.

Finally, I would like to thank the members of the Data Mining group: Ravi Sarma, Victor Chang, Ashish Mishra, Benjamin Fang, Danny Lai, and Katherine Koch.
# Table of Contents

1. **INTRODUCTION** ........................................................................................................................................ 8  
   1.1 **DATA MINING OVERVIEW** ................................................................................................................ 9  
   1.2 **BLAST FURNACE BACKGROUND** ...................................................................................................... 10  
   1.3 **STEEL MAKING PROCESS** ................................................................................................................ 10  

2. **NEURAL NETWORKS** .................................................................................................................................. 14  
   2.1 **REASON FOR NEURAL NETWORKS** ................................................................................................... 14  
   2.2 **PREVIOUS RESEARCH** ..................................................................................................................... 16  
   2.3 **FEED-FORWARD NEURAL NETWORKS** .............................................................................................. 21  
   2.4 **NNRUN** .............................................................................................................................................. 26  

3. **HOT METAL TEMPERATURE** .................................................................................................................. 30  
   3.1 **NARROWING THE PARAMETERS** ....................................................................................................... 30  
   3.2 **HMT DATA PREPROCESSING** ........................................................................................................... 33  
   3.3 **RESULTS** ........................................................................................................................................... 36  
      3.3.1 1 vs. 2-Hidden Layers .................................................................................................................. 36  
      3.3.2 Automated vs. Forced Lags ........................................................................................................... 38  
      3.3.3 Moving Window vs. Hourly Averaging .......................................................................................... 42  
      3.3.4 Results Analysis ............................................................................................................................ 48  
   3.4 **COMPARING LINEAR REGRESSION TO ANNS** ................................................................................ 49  

4. **SILICON CONTENT** .................................................................................................................................. 53  
   4.1 **DATA PREPROCESSING** .................................................................................................................... 53  
   4.2 **PREDICTION AND MODELING OF SILICON CONTENT** ...................................................................... 55  
      4.2.1 Results using coke and sinter as inputs ....................................................................................... 55  
      4.2.2 Results Silicon Content without coke and sinter as inputs ....................................................... 58  

5. **SLAG BASICITY** ....................................................................................................................................... 62  
   5.1 **DATA PREPROCESSING** .................................................................................................................... 62  
   5.2 **PREDICTION AND MODELING OF SLAG BASICITY** ........................................................................ 64  
      5.2.2 Slag Basicity when datapoints removed if coke and sinter do not exist .................................... 66  
      5.2.3 Slag Basicity when placing empty values for coke and sinter ..................................................... 67  

6. **SINTER** ...................................................................................................................................................... 71  
   6.1 **SINTER DATA PREPROCESSING** ....................................................................................................... 71  
   6.2 **PREDICTION OF RDI AND +10 RESULTS** ..................................................................................... 75  
      6.2.2 +10 Results ..................................................................................................................................... 79  

7. **CONTROL** .................................................................................................................................................. 82  
   7.1 **IDENTIFYING KEY CONTROL PARAMETERS** ................................................................................... 82  
      7.1.1 Control Parameter Reduction Techniques .................................................................................... 82  
      7.1.2 Sensitivity Analysis: Derivative Approach .................................................................................... 84  
   7.2 **DITHERING VS. DERIVATIVES** ........................................................................................................ 88  
   7.3 **CONTROL SYSTEMS** ....................................................................................................................... 90  
      7.3.1 Supervised Control .......................................................................................................................... 90
List of Figures

FIGURE 1 : A TYPICAL HOT METAL CHEMISTRY ................................................................. 13
FIGURE 2 : SERIAL AND PARALLEL PRIOR KNOWLEDGE NEURAL NETWORK MODEL [19] 19
FIGURE 3 : A FEED-FORWARD NEURAL NETWORK WITH 4 INPUTS, 1 OUTPUT AND 1 HIDDEN LAYER 22
FIGURE 4 : NNRUN MAIN MENU .................................................................................... 26
FIGURE 5 : NNRUN PARAMETERS MENU ........................................................................ 27
FIGURE 6 : NMSE 1 VS. 2-HIDDEN LAYER NETWORKS ..................................................... 37
FIGURE 7 : PEARSON COEFFICIENT 1 VS. 2 HIDDEN LAYERS ........................................ 38
FIGURE 8 : PEARSON COEFFICIENT AUTOMATED VS. FORCED LAGS ........................... 39
FIGURE 9 : AVERAGE ERROR AUTOMATED VS. FORCED LAGS ...................................... 40
FIGURE 10 : NMSE ERROR AUTOMATED VS. FORCED LAG NETWORKS ........................... 41
FIGURE 11 : HMT DATA WITH MOVING WINDOW AVERAGING ......................................... 43
FIGURE 12 : HMT DATA WITH HOURLY AVERAGING ....................................................... 45
FIGURE 13 : HMT 1-HOUR PREDICTION ......................................................................... 46
FIGURE 14 : HMT 2-HOUR PREDICTION ......................................................................... 47
FIGURE 15 : HMT 4-HOUR PREDICTION ......................................................................... 48
FIGURE 16 : HMT 8-HOUR PREDICTION ......................................................................... 50
FIGURE 17 : PEARSON COEFFICIENT NEURAL NETWORK VS. LINEAR REGRESSION ..... 51
FIGURE 18 : NMSE NEURAL NETWORK VS. LINEAR REGRESSION .................................... 55
FIGURE 19 : SILICON CONTENT 1-HOUR PREDICTION WITH COKE AND SINTER AS INPUTS 56
FIGURE 20 : SILICON CONTENT 2-HOUR PREDICTION WITH COKE AND SINTER AS INPUTS 57
FIGURE 21 : SILICON CONTENT 4-HOUR PREDICTION USING COKE AND SINTER ............... 58
FIGURE 22 : SILICON CONTENT 8-HOUR PREDICTION USING COKE AND SINTER AS INPUTS 59
FIGURE 23 : SILICON CONTENT 1-HOUR PREDICTION WITHOUT COKE AND SINTER AS INPUTS 60
FIGURE 24 : SILICON CONTENT 2-HOUR PREDICTION WITHOUT COKE AND SINTER AS INPUTS 61
FIGURE 25 : SILICON CONTENT 4-HOUR PREDICTION WITHOUT COKE AND SINTER AS INPUTS 62
FIGURE 26 : COMPLEXITY OF SILICON CONTENT ................................................................ 64
FIGURE 27 : SLAG BASICITY 1-HOUR PREDICTION WHEN DATA POINTS DROPPED ............. 66
FIGURE 28 : SLAG BASICITY 2-HOUR PREDICTION WHEN DATA POINTS DROPPED ............. 67
FIGURE 29 : SLAG BASICITY 4-HOUR PREDICTION WHEN DATA POINTS DROPPED ............. 68
FIGURE 30 : SLAG BASICITY 1-HOUR PREDICTION WHEN EMPTY VALUES USED ............... 69
FIGURE 31 : SLAG BASICITY 2-HOUR PREDICTION WHEN EMPTY VALUES USED ............... 70
FIGURE 32 : SLAG BASICITY 4-HOUR PREDICTION WHEN EMPTY VALUES USED ............... 71
FIGURE 33 : NMSE SLAG BASICITY .................................................................................. 72
FIGURE 34 : NMSE FOR RDI RAW, AVERAGED, AND LINEARLY INTERPOLATED DATA 74
FIGURE 35 : NMSE FOR RDI RAW, AVERAGED, AND LINEARLY INTERPOLATED RDI DATA 75
FIGURE 36 : RDI 1-DAY PREDICTION .............................................................................. 76
FIGURE 37 : RDI 2-DAY PREDICTION .............................................................................. 77
FIGURE 38 : RDI 4-DAY PREDICTION .............................................................................. 78
FIGURE 39 : +10 1-DAY PREDICTION ............................................................................. 79
FIGURE 40 : +10 2-DAY PREDICTION ............................................................................. 80
FIGURE 41 : +10 4-DAY PREDICTION ............................................................................. 81
FIGURE 42 : NMSE RDI VS. +10 ..................................................................................... 82
FIGURE 43 : NEURAL NETWORK WITH 2 HIDDEN LAYERS .............................................. 84
FIGURE 44 : DERIVATIVE FOR CHRGDTIM & COKTOTAL ............................................ 87
FIGURE 45 : DERIVATIVE FOR ECO & EH2 ................................................................. 88
FIGURE 46 : DITHERING VS. DERIVATIVE FOR CHRGDTIM ........................................ 89
FIGURE 47 : DITHERING VS. DERIVATIVE FOR COKTOTAL ......................................... 89
List of Equations

EQUATION 1 : NEURAL NETWORK OUTPUT ................................................................. 23
EQUATION 2 : SIGMOID FUNCTION ........................................................................ 23
EQUATION 3 : OUTPUT OF NEURON ..................................................................... 23
EQUATION 4 : MEAN SQUARED ERROR ................................................................. 24
EQUATION 5 : NUMBER OF NETWORK CONFIGURATIONS .................................. 27
EQUATION 6 : NORMALIZED MEAN SQUARED ERROR ........................................ 28
EQUATION 7 : PEARSON CORRELATION COEFFICIENT ..................................... 32
EQUATION 8 : NORMALIZATION FORMULA .......................................................... 34
EQUATION 9 : AVERAGE ERROR FORMULA ........................................................ 36
EQUATION 10 : LINEAR REGRESSION MODEL ....................................................... 50
EQUATION 11 : INFORMATION CONTENT ........................................................... 84
EQUATION 12 : DERIVATIVE OF 2-LAYER NEURAL NETWORK .......................... 86
1. Introduction

This thesis focuses on using neural networks to model and predict the Hot Metal Temperature, Silicon Content, Slag Basicity, and Sinter in a blast furnace. A blast furnace is very difficult to model due to the complex flow conditions relating to the mass and heat transfer inside. There are currently no universally accepted methods for accurately controlling blast furnace operation and predicting the outcome.

Hot Metal Temperature, Silicon Content, Slag Basicity, and Sinter are important indicators of the internal state of a blast furnace as well as of the quality of the pig iron being produced. The production of pig iron involves complicated chemical reactions and causes complex relationships between the various chemicals used. These relationships are non-linear and difficult to estimate using statistical techniques [36]. Artificial Neural Networks have been used to predict the silicon content of pig iron [8]. This work has been extended to further improve the results of Silicon Content prediction and apply the neural network technique to predict Hot Metal Temperature, Slag Basicity, and Sinter. This thesis examines how the input variables for each model are chosen and how prediction depends on the architecture of the network used. Various methods for preprocessing data sets are analyzed and results are displayed. It describes NNRUN, a software package that finds the best multi-layer perceptron network based on a number of input parameters.
The second part of this thesis discusses the control of the Hot Metal Temperature in the blast furnace. The first step towards control is to detect the most influential variables for the output HMT. Using the optimally trained neural network, the input variables are ranked based on their impact on the hot metal temperature. The input variables that are ranked at the top of this list can later be used to implement a control system.

1.1 Data Mining Overview

Data mining is the process of discovering correlations and trends in large amounts of data using mathematical and knowledge-based techniques. Data mining is related to artificial intelligence and machine learning. It has been applied to many fields including financial [28] and marketing analysis [25]. Previously, data had only been used for user searches while recently the amount of data collected has grown exponentially. There has been an increased interest in "mining" these data to learn new information. Artificial neural networks are one of many techniques that can be used in the analysis of data. An artificial neural network can find nonlinear relationships in data without a pre-defined model. The great increase in the computational power of computers has made training and testing of ANNs on large amounts of data possible. Neural networks are often more accurate than commonly used learning algorithms [10].
1.2 Blast Furnace Background

The blast furnace is the first step in producing steel from iron oxides. Blast furnaces were first used in the 14th Century and produced one ton a day. Modern blast furnaces produce 13,000 tons per day [11]. The processes inside the blast furnace have not changed even though higher production rates can be achieved today [11].

Blast furnace operators combine iron ore, limestone, and coke in the blast furnace to produce molten iron. The blast furnace chemically reduces and physically converts iron oxides into liquid iron called hot metal. The blast furnace is a large stack lined with brick. Thousands of cubic meters of materials can be held in the blast furnace [24]. Iron ore, coke, and limestone are put into the top and take six to eight hours to descend to the bottom of the furnace. They then become the final product of liquid slag and liquid iron. Heated air is blown into the bottom of the furnace. This comes to the top in six seconds after going through several chemical reactions. Once started, a blast furnace will run continuously for four to ten years with only short stops for maintenance [11].

1.3 Steel Making Process

The quality of the pig iron produced determines how expensive it will be to produce steel from this pig iron. This quality varies depending on the amount and composition of any impurities and the temperature of the hot metal when it is tapped.
from the blast furnace [21]. Iron oxides come to the blast furnace plant in the form of raw ore, pellets, or sinter. Oxygen is removed from the iron oxides to produce pure liquid iron. The raw ore is removed from the earth and made into small pieces. This ore is either Hematite (Fe₂O₃) or Magnetite (Fe₃O₄) and iron rich ore can be placed directly into a blast furnace without any further processing. Iron ore that contains a lower iron content must be processed to increase its iron content.

Sinter is produced from fine raw ore, small coke, limestone, and numerous other steel plant waste materials that contain some iron. These materials are mixed together, heated by a furnace, and fused by the heat from the coke fines into larger size pieces. The iron ore, pellets, and sinter then become the liquid iron produced in the blast furnace. Any remaining impurities go to the liquid slag.

The coke is produced from a mixture of coals. The coal is crushed and ground into a powder and then heated in an oven. The coke is made up mostly of carbon, with some ash and sulfur. The coke with a high energy value provides heat and gases which are required to reduce and melt the iron ore, pellets, and sinter.

The final raw material in the iron-making process is limestone. The limestone is removed from the earth by blasting with explosives. The limestone can contain large amounts of calcium or magnesia. Since the limestone is melted to become the slag which removes sulfur and other impurities, the blast furnace operator may blend the different stones to produce the desired slag chemistry that create optimum properties such as has a low melting point and a high fluidity.
These materials are placed into the furnace top and go through many chemical and physical reactions while descending to the bottom of the furnace. The iron ore, pellets and sinter are reduced, meaning that oxygen in the iron oxides is removed by a series of chemical reactions. These reactions occur as follows:

1) \( 3 \text{Fe}_2\text{O}_3 + \text{CO} = \text{CO}_2 + 2 \text{Fe}_3\text{O}_4 \) Begins at \( 850^\circ F \)
2) \( \text{Fe}_3\text{O}_4 + \text{CO} = \text{CO}_2 + 3 \text{FeO} \) Begins at \( 1100^\circ F \)
3) \( \text{FeO} + \text{CO} = \text{CO}_2 + \text{Fe} \)
   or \( \text{FeO} + \text{C} = \text{CO} + \text{Fe} \) Begins at \( 1300^\circ F \)

As the iron oxides are going through these reactions, they start to melt into liquid iron through the coke to the bottom of the furnace. The coke descends to the bottom of the furnace to the level where the hot blast enters. The coke reacts to generate heat as follows:

\[ \text{C} + \text{O}_2 = \text{CO}_2 + \text{Heat} \]

The carbon dioxide is reduced to carbon monoxide as follows:

\[ \text{CO}_2 + \text{C} = 2\text{CO} \]

The product of this reaction, carbon monoxide, is necessary to reduce the iron ore as seen in the previous iron oxide reactions. The limestone descends in the blast furnace and goes through the first reaction below. This starts at about \( 1600^\circ F \).
The CaO formed is used to remove sulfur from the iron (second reaction) which must occur before the hot metal becomes steel.

\[
\begin{align*}
\text{CaCO}_3 &= \text{CaO} + \text{CO}_2 \\
\text{FeS} + \text{CaO} + \text{C} &= \text{CaS} + \text{FeO} + \text{CO}
\end{align*}
\]

The CaS becomes part of the slag. The slag is also formed from left over SiO₂, Alumina, Al₂O₃, MgO, or CaO that entered with the iron ore, pellets, sinter or coke. The liquid slag then goes to the bottom of the furnace where it floats on top of the liquid iron since it is less dense.

In general in the blast furnace solids descend and gases ascend. Many chemical and physical reactions occur in a blast furnace to produce the final product which is hot metal[11]. A typical hot metal chemistry is:

<table>
<thead>
<tr>
<th>Element</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iron (Fe)</td>
<td>93.5 - 95.0%</td>
</tr>
<tr>
<td>Silicon (Si)</td>
<td>0.30 - 0.90%</td>
</tr>
<tr>
<td>Sulfur (S)</td>
<td>0.025 - 0.050%</td>
</tr>
<tr>
<td>Manganese (Mn)</td>
<td>0.55 - 0.75%</td>
</tr>
<tr>
<td>Phosphorus (P)</td>
<td>0.03 - 0.09%</td>
</tr>
<tr>
<td>Titanium (Ti)</td>
<td>0.02 - 0.06%</td>
</tr>
<tr>
<td>Carbon (C)</td>
<td>4.1 - 4.4%</td>
</tr>
</tbody>
</table>

*Figure 1: A Typical Hot Metal Chemistry*
2. Neural Networks

The computer science concept of an artificial neural network is based on the biological neural network. The brain has about 10 billion interconnected neurons which use biochemical reactions to receive, process, and transmit information. If the input to the neuron is greater than the threshold value then the neuron fires transmitting an output signal. The biological neuron can be simulated by a simple model. The inputs each have a weight that they contribute to the neuron, if the input is active. The neuron can have any number of inputs; neurons in the brain can have as many as a thousand inputs. Each neuron also has a threshold value. If the sum of all the weights of all active inputs is greater than the threshold, then the neuron is active [13].

2.1 Reason for Neural Networks

This section discusses why neural network models were chosen for prediction in the blast furnace. Conventional methods for solving problems involve determining an algorithm that implements a solution for that problem. The main differences between neural networks and conventional methods are that neural networks are trained with examples while with conventional methods a model is used to derive an algorithm. Neural networks are useful for problems where no direct algorithmic solutions exist and when examples of desired responses are available for training the neural network [3].
Steel production is a complex process that involves many nonlinear reactions. Deterministic models and equations based on the chemistry involved in the blast furnace have been made [34] in the past. However, these models have not been very successful for control or prediction in the furnace. Biswas concludes that it is almost impossible to achieve accurate prediction and control of blast furnace variables based only on domain level models [34]. There is very little reliability in using the chemical equations that describe the reactions in the blast furnace because it is very difficult to attain accurate input parameter measurements. The variations in the composition of the raw materials and other input uncertainties further deter the modeling process [1].

Neural networks were chosen for modeling in the blast furnace for a few reasons. They have proven to be more capable of capturing complex relationships than standard regression methods [29]. Neural networks learn by example and are trained using large amounts of data rather than using domain knowledge and the chemical relationships. They are useful for models that change over time. The static algorithmic or regression models cannot adapt to changes in the environment [3]. Finally, they have proven to be successful in many different applications. At MIT, neural networks have been used to optimize inventory levels [25][26] for a major pharmaceutical company. They were able to reduce the required amount of inventory by 50% while still guaranteeing that there was a 95% chance that a random customer would be able to find the drug that they were looking for. Also at MIT, neural networks were successfully used for character recognition on bank checks [47].
2.2 Previous Research

Researchers have made many attempts to predict the variables in a blast furnace based on the conditions inside the furnace. Modeling the relationships between various variables in the blast furnace has been difficult using standard statistical techniques [5]. This is because non-linearities exist between the different parameters used in pig iron (hot metal) production. Therefore, many people have turned to neural networks to predict various blast furnace parameters. This section gives a brief overview of the major work done in this area using neural networks.

Multiple layered feed-forward artificial neural networks were used to predict the silicon content of hot metal from a blast furnace by Abhay Bulsary, Henrik and Bjorn Saxen [8]. Time-dependencies (time lags) between each of the various inputs used and the output (silicon content) were found to be significant. Therefore, each input was lagged from the output by its optimal amount, as found by the correlation coefficient. The input variables used included: blast, blast volume, calculated heat loss, oil injection and the silicon content of the previous two taps. Feed-forward networks with one, two, and three hidden layers were tried and the results found that the feed-forward ANNs produced considerably better results than standard linear time-series prediction.

Several different artificial neural network models were tried by Himanshu Singh, Nallamal Venkata and Bramha Deo [6] in order to predict the silicon content of hot metal. The following inputs were used: coke rate, hot blast temperature, slag
rate, top pressure, slag basicity and the logarithm of blast kinetic energy. The networks consisted of three layers (input, hidden and output) and the number of hidden nodes varied from six to eleven. Back-propagation was used to update the weights. They also tried using a dynamic learning rate model, functional link model, and a fuzzy neural network. The fuzzy neural network showed the best results, while the back-propagating model was next. The conclusion of this paper was that ANNs increased the predictive power of silicon content compared to other models.

Bulsari and Saxen [5] used feed-forward neural networks when trying to classify the state of a blast furnace based on the measurement of blast furnace temperatures. Since the knowledge relating blast furnace temperature to gas distribution within the furnace is non-linear and had only been known by people who had experience operating the blast furnace, a neural network seemed like a good solution. Based on the measurements of the blast furnace temperature, the network could classify the state of the gas distribution in the blast furnace into one of 6 categories. A feed-forward network using back-propagation was constructed. They found that larger networks, with more hidden nodes and hidden layers, showed better results than smaller networks. As the number of hidden nodes decreased, the accuracy of the network also decreased. The worst model was the linear regression model. Since more hidden nodes provided better results larger networks were deemed to be more capable of capturing the nonlinear relationships between the variables in the system in order to classify the state of the blast furnace more accurately.
Bhattacharjee, Dash, and Das of Tata Steel used neural networks to predict the Reduction Degradation Index in the blast furnace. The network architecture that was used had fifteen input nodes, seven hidden nodes, and the one output, RDI. They found that the network predicted the correct range for RDI 82% of the time. The network not only predicted the general level of RDI but also the magnitude of the variation[12].

Also at Tata Steel, researchers have used neural networks to model an air separation unit that produces high purity oxygen, nitrogen, and argon [31]. Single hidden layer neural networks with three inputs were capable of predicting the gaseous oxygen production rate.

Thompson and Kramer of MIT’s Department of Chemical Engineering have a method to model chemical processes using prior knowledge and neural networks. Artificial neural networks have the ability to approximate complex functions and have been verified to be universal function approximators [22]. They are able to model a process without detailed background knowledge. Because of this they have been used to model chemical processes. Since the neural network does not have a process-based internal structure it is at a disadvantage when trained with sparse, noisy data [19]. Inadequate data will decrease the accuracy of the neural network since the network must rely on the data completely. Prior knowledge about the function may help to increase the reliability of the neural network model. Thompson and Kramer present two ways in which prior knowledge can be used. In the serial approach shown in Figure 2, the output is made to be consistent with the prior
knowledge, which is represented as function $y$. In the parallel approach the prior model is used as a guide [23].

![Figure 2: Serial and Parallel Prior Knowledge Neural Network Model [19]](image)

Thompson and Kramer used a radial basis function network (RBFN). The RBFN has one hidden layer and one output layer. The hidden layer nodes have multivariate Gaussian activation functions and the output layer is a simple linear function. Their results showed that the RBFN model compensates for the noisy and sparse data. Prior knowledge improves performance by acting as a default estimate when training data is absent [19].

Expert systems have been used in the steel industry to improve production efficiency and quality assurance [24]. Steel-makers have not been making closed
control loops for control steel production because of the nature the steel-making process. The most important aspect of using control systems is to standardize production. Expert systems are used to predict abnormal situations and keep temperatures stable. Blast furnace operators can adjust the temperatures in the furnace in many ways such as: changing the ore-to-coke ratios, fuel-injection levels, and blast furnace moisture. Since the reaction time of the blast furnace operators and the way in which they respond to a problem are very different, it is difficult to standardize the control of the furnace. An expert system would build a model from previous data values and have rules about when and how the operator should react. Nippon Steel's Artificial and Logical Intelligence System was one of the first expert systems used [24]. In the experiment the system improved performance by 25% as opposed to 7% for operator-only control. The last chapter in this thesis discusses control and how a control system can be implemented in a blast furnace.
2.3 Feed-forward Neural Networks

Artificial neural networks can model complicated functions between inputs and outputs when the form of the function is unknown [2]. Artificial neural networks are known for recognizing patterns from noisy, complex data, estimating their nonlinear relationships and learning by example. The examples consist of pairs of inputs and outputs that are known to go together. Learning consists of determining the weights so that for every input the network generates the corresponding output. It can learn the general relationships and patterns in the data by training on a set of examples. Since neural networks are capable of modeling non-linear relationships, these models have been and continue to be quite useful in modeling the complex relationships between various chemicals and parameters in the blast furnace [2].

Our research focused on the multi-layer perceptron feed-forward neural network architecture. The hidden layer of nodes increases the complexity of the problems that can be solved. The goal of a feed-forward network is to model a function mapping the inputs to the outputs (i.e. the network models the function \( y = f(x) \) where \( y \) is a vector of outputs and \( x \) is a vector of inputs). The output from the neural network can be single or multiple. In our research we have multiple inputs mapped to a single output.
Figure 3: A feed-forward neural network with 4 inputs, 1 output and 1 hidden layer

The boxes in figure x form the input layer, which accepts inputs from the outside. The inputs are then fed into the neurons of the second layer, referred to as the hidden layer because it is hidden from the outside. More than one hidden layer may be present in a neural network. As the number of hidden layers become larger, the mapping function between the input and outputs increases in complexity [13]. This output is then fed into the next layer of processing neurons. In figure x, the next layer of processing neurons is the output layer.

The basic building block of a neural network is the neuron. Each neuron has an $n$ dimensional input vector and one output. In a feed-forward neural network the output is not fed back to the input of the neuron. The output of the neuron is a
nonlinear function of the weighted sum of all the inputs. Each link is weighted with a weight \( w_i \). The output of the neuron would be in the form:

\[
Y = f(\sum w_i a_i)
\]

- \( w_i \) is the weight on each link
- \( a_i \) is the input at that link.

\[\text{Equation 1 : Neural Network Output}\]

Generally, this nonlinear function \( f \) maps the weighted sum of inputs to a value between 0 and 1. The transforming function usually used is a sigmoid \([2]\), which has the form:

\[
F(u) = \frac{1}{1 + e^{-u}}
\]

\[\text{Equation 2 : Sigmoid Function}\]

As \( u \) goes to \(-\infty\), the sigmoid function approaches zero. When \( u \) approaches \( \infty \), the sigmoid function approaches one. Thus, the output of a processing neuron \( j \) takes the form:

\[
O_j = \frac{1}{1 + e^{-s}} \quad \text{where } s = \sum w_i a_i, \quad a_i \text{ is the } i\text{th input into the node and } w_i \text{ is the weight of the link carrying the } i\text{th input}
\]

\[\text{Equation 3 : Output of neuron}\]
Thus, individual neurons only perform a simple signal processing function. However, when the neurons are organized into layers with a distinct topology, the network as a whole can produce very complex functions between the inputs and outputs.

The first phase of a neural network is learning. During the first phase, the neural network is presented with a series of examples. An example consists of sets of input and output pairs. The network will learn to associate a given output with a given input by adapting its weights. During the learning phase, the back-propagation learning algorithm adjusts each weight of each link so as to minimize the mean squared error of the network. This error is defined as:

\[
E = \frac{N}{N} \sum_{n=1}^{N} (O_n - y_n)^2
\]

Equation 4: Mean Squared Error

where \( N \) is the number of examples that the network uses to train [2]. \( O_n \) is the predicted value for the \( n \)th example. \( y_n \) is the actual value for the \( n \)th example. \( E \) is the mean of the errors squared, where the error is the difference between what the network predicts given a set of inputs and the actual output for that example. The goal of training is to find the combination of weights that minimize the error.
Back-propagation starts by making a random guess for a set of weights for the network. Zero cannot be used as a starting for a weight because it may stay stuck at zero throughout the backpropagation procedure [2]. The network will then run through all the values in the training set (called an epoch) and adjust the weights to minimize the error. At the beginning of each epoch, the network maps the inputs of the example into outputs using the current weights. The error is then calculated at the output. Each output node decides how each of its weights should change to reduce the error. The output node then propagates its error back through its links to the previous hidden nodes. Each hidden node will then decide how it should change its weights in the same way. The node will calculate its error derivative, which is the partial derivative of the node’s error with respect to one of its weights: \( \delta E/\delta w_i \). \( E \) is the network error described earlier and \( w_i \) is a particular link weight. The actual change in the weight \( \Delta w_i \) is proportional to the magnitude of the error derivative. Thus, if changing the weight a little reduces the error by a large amount, that particular weight should be reduced by a relatively large amount.

This cycle is repeated hundreds of times in training a network. One cycle through all the training examples is known as an epoch, and often training is done on hundreds of epochs. The testing dataset is used to decide when to stop training. Overfitting occurs when the error is decreasing on the training data, but increasing on the testing data. This means that the network is memorizing the examples in the training data, but is not able to generalize to other examples. To avoid overfitting, the training is stopped when the Mean Squared Error on the testing set stops decreasing from one epoch to the next.
2.4 NNRUN

The NNRUN system was developed to help implement Artificial Neural Networks using Matlab’s NNET toolbox. It greatly reduces the time needed to search for the optimal neural network configuration for a given dataset. NNRUN is made up of the ANN training and testing tool, NNTESTER, and the prediction tool, NNPREDICTOR. The NNTESTER module finds the best neural network architecture to model the data being used. The initial design and development of NNRUN was performed by Vladislav Gabovich and Bradley Banks [21].

```
NNRUN menu
----------
(q) Quit
(t) Train ANNs using NNTESTER
(h) View help
(p) Predict using NNPREDICTOR

Enter selection character:
```

Figure 4: NNRUN Main Menu

The system gives the user many options on how to find the best ANN architecture. It allows the user to choose a one or two hidden layer model and the minimum and maximum number of nodes in each layer. The data is partitioned into a training set (80%) and a testing set (20%). The training set will be used to find the optimal weights for the ANN while the testing set is used to find the accuracy of the weights found. The software also allows the user to specify the number of runs. The more runs which are used for training allows the network to be more generalized.
The user also has the choice for the runs to be randomly or sequentially partitioned. After running several ANN models it was found that the best networks are trained with a large number (three or four) randomized runs. The user can also specify the partitions that the network should run on or opt to have them be selected randomly or sequentially.

----NNRUN Parameters Menu---------
(1) Number of runs
(2) Learning rate
(3) Minimum gradient
(4) Mse change
(5) Number of training epochs
(6) Goal training error
(7) Fraction of pts used for test set
(8) Randomize runs
(v) View current selection
(q) Quit to NNT ESTER menu

Entering any selection number will allow you to view the current value of the selected parameter, acceptable range and recommended ranges. If desired, the current value can be changed.

-----------------------------
Enter selection:

Figure 5: NNRUN Parameters Menu

The software will train every network within the range of the specified number of nodes and choose the best one based on the Normalized Mean Square Error. The number of network configurations that will be trained and tested would be:

\[[\text{Max}(\text{Layer1}) - \text{Min}(\text{Layer1})]\times[\text{Max}(\text{Layer2}) - \text{Min}(\text{Layer2})]\times\text{NumberRuns}\]

Equation 5: Number of Network Configurations
The Normalized Mean Square Error used to rank the performance of the networks is:

\[ NMSE = \frac{\sum_{t=1}^{n} (\text{actual}_t - \text{predicted}_t)^2}{\sum_{t=1}^{n} (\text{actual}_t - \text{mean}_t)^2} \]

Equation 6: Normalized Mean Squared Error

If more than one randomized run is used then the network that has the lowest average NMSE over all runs is the optimal network. NNRUN avoids overfitting the neural network. Overfitting occurs when instead of generalizing, the network starts memorizing the examples in the dataset. The MSE of an overfitted network will continue to decrease when using the training data, but increase when using the testing data. NNRUN stores in memory the last \( n \) number of trained networks. If the MSE for testing is increasing while for training it is decreasing for the last \( n \) networks then it determines that these networks are overfitted. It will discard these \( n \) networks and use the last network that was saved before this trend started occurring. The optimal value of \( n \) was found to be five. That is, once five consecutive epochs showed the trend that the MSE decreased for the training data but increased with the testing data, the optimal weight configuration has been found.

The NNPREDICTOR module uses a previously trained ANN and an input dataset to make the predictions of the output variables based on the ANN trained with NNTESCTOR.
When a dataset is processed for training use by NNTESTER, the parameters necessary to process the future data in exactly the same way are stored. Then, when the user wants to predict new data, the preprocessing utilities are passed the stored information to generate a dataset with the same processing methods used.
Hot metal is also referred to as pig iron and its temperature is in the range of 1450 - 1500 degrees Celsius. The hot metal temperature is used in determining the quality of the pig iron and its temperature is an important parameter in both the control and the prediction of the blast furnace [1]. The temperature of hot metal is measured whenever pig iron is tapped which occurs at irregular intervals.

3.1 Narrowing the Parameters

The initial data contained thirty-five input parameters. Some of the input variables were redundant and others were not useful in predicting HMT or Silicon content. A sensitivity analysis was performed in order to discover which variables were the most important. The way this was done was to calculate the correlation coefficient between each input variable and the corresponding output variable (HMT). The higher the correlation between a particular input and HMT, the more effect that particular input variable must be in determining HMT. Therefore, such a variable should be included in the data set. Using correlation relationships and information from the blast furnace experts at SteelCorp, the number of input variables were narrowed down from thirty-five to eleven. The original variables were:

<table>
<thead>
<tr>
<th>Input Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAFT_TH</td>
<td>RAFT temperature</td>
</tr>
</tbody>
</table>
The final eleven variables found to be the most important by the correlation coefficient were total coke, carbon oxide, hydrogen, steam, group 1 heat flux, group 2 heat flux, actual coke injection, % oxygen enrichment, ore/coke ratio, hot blast temperature (degrees C), and charge time for 10 semi charges. In addition to these eleven variables the previous hot metal temperature was also used as an input to predict the future HMT value.
Time-dependencies in the data were also taken into account. It is known that there exists a time lag between the time when the value of an input parameter is changed and the time when the output reacts to that changed input. The effect of the changing input parameter on the HMT is not instant. The optimal lag for a variable is the time lag between the variable and the output such that the correlation between the input variable and the output is the highest. In order to find the optimal lag for each output variable, the Pearson coefficient was calculated between each input, lagged from the output by anywhere from 0-8 hours, and HMT. The maximum lag of eight hours was chosen based on domain expert advice. The Pearson coefficient is as follows:

\[
\text{Corr}(x,y) = \frac{\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \sqrt{\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}}
\]

Equation 7: Pearson Correlation Coefficient

The maximum Pearson coefficient was recorded for each variable as well as the lag at which it occurred. At the same time information was collected from the domain experts on what the actual lag should be. These actual lags and the optimal lags found by the coefficient method are compared in the table below:
<table>
<thead>
<tr>
<th>Variable</th>
<th>Optimal Lag (determined by Pearson coefficient)</th>
<th>Lag (determined by domain experts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charge Time</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Coal Injection</td>
<td>3</td>
<td>3-4</td>
</tr>
<tr>
<td>Group 1 Heat Flux</td>
<td>8</td>
<td>2-3</td>
</tr>
<tr>
<td>Group 2 Heat Flux</td>
<td>8</td>
<td>3-4</td>
</tr>
<tr>
<td>Hot Blast</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Temperature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ore/Coke</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Oxygen Flow</td>
<td>8</td>
<td>1-4</td>
</tr>
<tr>
<td>Steam Flow</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Top gas CO</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Top gas CO₂</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Top gas H₂</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Wind Flow</td>
<td>1</td>
<td>1-3</td>
</tr>
</tbody>
</table>

In most cases it seems that the optimal lag predicted by the Pearson coefficient was much higher than that suggested by the domain experts. Two different datasets were created and used to train and test the network. One applied the optimal lags found by the Pearson coefficient and the other used the lags supplied by the domain experts. In the case where the correct lag could not be used (i.e. when predicting future values at time $t+4$, but the lag is two hours and therefore the value at $t+2$ is needed) the current value (at time $t$) was used. The results of both models are below.

### 3.2 HMT Data Preprocessing

The raw data from the blast furnace were not in the correct form for direct use. Problems in the data ranged from missing or very abnormal values to not taking into account the effect of time lags in the production process. Several steps
were involved in preprocessing the raw data into a dataset that would be suitable for training an artificial neural network.

Extremely abnormal data values were adjusted to make the data more consistent. Values that were more than two standard deviations from the mean were modified so that they would be two standard deviations away from the mean. In some cases a minimum value for a variable was specified. If two standard deviations below the mean is smaller than the minimum then the data was adjusted to the minimum value. This process removed any outliers from the data set.

Neural networks are sensitive to the scale and variance of the input variables. For this reason, all the input variables were normalized between zero and one using the following formula:

\[
\frac{X_{ij} - \text{Min}\{X_i\}}{\text{Max}\{X_i\} - \text{Min}\{X_i\}}
\]

- \(X_{ij}\) is the \(j\)th value of the \(i\)th input (the value to be normalized)
- \(\text{Min}\{X_i\}\) and \(\text{Max}\{X_i\}\) are the minimum and maximum values of the \(i\)th input variable

**Equation 8 : Normalization Formula**

By placing all the data points within the same range noise from the wide scale and distribution of data values is minimized.
A major problem with the original data was inaccurate values of HMT for many of the datapoints. HMT can be measured only approximately once every hour while the other datapoints were taken every five minutes. Linear interpolation between measurements of HMT was used to approximate values for the missing data points.

The raw data from the blast furnace contained a total of 9100 data points taken every five minutes. This five minute level data may see some inputs changing rapidly from one value to another, but since the temperature changes slowly over a longer period of time these short term changes do not have a noticeably affect on the output. Domain knowledge from SteelCorp indicated that an effective unit for considering the data would be in blocks of one hour. Therefore groups of twelve data points were averaged to create one data point which represented one hour block. While hourly averaging of the data improved the predictive ability of the network, it had a side effect of greatly reducing the number of data points available for training the networks. The hourly averaging reduced the number of data points to approximately 760. The results from the hourly averaged (760 data points) are shown below.

A moving window technique can be used to increase the size of the dataset. The moving window takes the first twelve data points and averages them, but in the next step it shifts over by a five-minute interval and averages the new data point with the previous eleven data points. The window continues to slide one data point at a time, until the end of the set is reached. This technique allows the use of almost the
same number of data points as in the original dataset. If the initial data set has \(i\) data points and a window size of \(j\) is used the resulting set of data will have \(i-j\) data points in total. The neural network could then train and test on this adjusted data.

### 3.3 Results

#### 3.3.1 1 vs. 2-Hidden Layers

After creating the datasets, we trained and tested the networks using both one and two hidden layers. Previous research shows that a two hidden layer network can predict values more accurately than a one-layer network [14]. We modeled and predicted future HMT values with both types of networks. When using NNRUN, for a one hidden layer network we allowed the software to search for the best network having within the range of one to thirty hidden nodes. For a two hidden layer network we allowed each layer to have between one and twelve hidden nodes. This is because as the number of hidden layers increase, the amount of processing time to find the best network goes up significantly. NNRUN uses the normalized mean square error to find the best network. Other ways to evaluate the performance of a network include the Pearson coefficient and the average error (below).

\[
\text{Average Error} = \frac{\sum_{t=1}^{n} |\text{actual}_\text{value}(t) - \text{predicted}_\text{value}(t)|}{n}
\]

where \(n\) is the number of testing data points

**Equation 9 : Average Error Formula**
The average error shows the average difference between the actual HMT value and the predicted HMT value over all datapoints in the testing set. The Pearson coefficient shows how the model predicts trends. When its value is 1 it means that the model is predicting trends at the highest accuracy possible. If it is -1 then it is not predicting the trends at all.

![Figure 6: NMSE 1 vs. 2-Hidden Layer Networks](image)

The chart above shows the normalized mean squared error for a one and two-hidden layer network. It compares this error across the prediction horizon of one to eight hours. The one and two-hidden layer networks have a similar NMSE when the model is predicting less than three hours into the future. However when the prediction horizon is longer the one-hidden layer network outperforms the two-hidden layer network. The following chart compares the Pearson coefficient for the one and two-hidden layer networks.
The Pearson coefficient also shows that a one-hidden layer network more accurately predicts the trends in the data than a two-hidden layer network for predictions further in the future. This could mean that a simpler network configuration can more accurately predict hot metal temperature for large prediction horizons. Another possibility is that because we have to set the range for the number of nodes in two hidden layers to be much smaller than for one-hidden layer, the optimal two-hidden layer is never found. The one-hidden layer network is given a much larger node configuration range causing its accuracy to be greater.

3.3.2 Automated vs. Forced Lags

As mentioned previously, the optimal lags found by the Pearson coefficient did not match the optimal lags that were determined by the domain experts. Several different models were created, some using the automated (optimal lags by Pearson coefficient) and others using the forced lags (optimal lags by domain
experts). The Pearson coefficient, normalized mean squared error, and average error were compared for the automated and forced lags data sets.

![Figure 8: Pearson Coefficient Automated vs. Forced Lags](image)

The Pearson coefficient shows similar trend prediction for automated and forced lags for future predictions of up to 4 hours. After 4 hours, the automated lags seems to be more accurate. The average error and normalized mean squared error are compared below.
Figure 9: Average Error Automated vs. Forced Lags
Both the Pearson coefficient and NMSE show similar trends as the average error above. The performance of both networks are similar for prediction a small amount of time into the future. However, as the prediction horizon increases, the automated lagged network outperforms the forced lagged network. This makes sense since the automated lagged network is using the lags that it found to be most highly correlated with the output. In other words, when the lags are forced, the network is using values that it did not find to be highly correlated with the output. As the model tries to predict further out into the future, it cannot depend on the previously known HMT value as much, and must depend on the other inputs. This makes the difference in performance between the two networks larger for longer prediction horizons.
3.3.3 Moving Window vs. Hourly Averaging

Previously the moving window technique was mentioned as a way to increase the data set size. However, one problem with this approach is that each new data point is not capturing much new information. Rather, each new set of twelve points has just one new averaged data point than the previous set. The moving window data set will be much smoother because of the overlapping data points. The HMT values from moving window averaged data is compared with hourly averaged data below.
Figure 11: HMT Data with Moving Window Averaging

HMT Values From Moving Window Averaged Data

Figure 12: HMT Data with Hourly Averaging

HMT Values From Direct Hourly Averaged Data
The figures above show that moving window averaging causes the HMT value curve to be smoother than the averaged data. The artificial neural network will be able to predict the values for the smoother curve more easily because the network will rely more on previous HMT values. It will be more difficult for the network to predict the jagged hourly averaged data. Since the moving window averaging is almost like cheating the network these results use the hourly averaged data.

The following graphs show hot metal temperature predictions one, two, four, and eight hours into the future. These predictions used the optimal lags found by the correlation coefficient (automated lags) and the hourly averaged data (760 datapoints).
Figure 13: HMT 1-Hour Prediction

The normalized mean squared error is .00049 and there are 7 hidden nodes in this model. For all of the following models, the dashed line is the actual HMT value and the solid line is the predicted HMT value.
Figure 14: HMT 2-Hour Prediction

The normalized mean squared error is .0012 and the number of hidden nodes is 20 for this model. The error for this model is larger than the 1 hour prediction, and the predicted value seems to lag the actual value more. However the model is still able to predict HMT value to a fair degree of accuracy.
The normalized mean squared error for this model is .0013 and the number of hidden nodes is 20. The error has increased slightly from the 2 hour model and the predicted value for this model also lag the actual values by a significant amount.

**Figure 15: HMT 4-Hour Prediction**
The normalized mean squared error is .0013 and the number of hidden nodes is 25. The complexity of the models has increased significantly as the prediction horizon is extended more into the future. The complexity is reflected in the optimal number of hidden nodes, as this number increases, the complexity of the model increases. The prediction values for the 8 hour prediction model tend to stay closer to the mean than the other models. There is also a significant lag in this model.

3.3.4 Results Analysis

Several points come up when analyzing the networks:

1. The accuracy of each network falls as we predict further into the future. Noise and other factors may distort the relationship between HMT and its input variables.
2. As we predict further into the future the predicted values lag the actual values by larger amounts. This means that the networks heavily depend on the previously known HMT value when predicting the future values.

3. The complexity of the networks (number of hidden nodes) also increases as the prediction horizon increases. The networks are trying to capture the relationships between all of the input variables and the HMT output rather than just the previously known HMT value for predictions further into the future.

4. As we predict further into the future, the predictions are closer to the mean. The model is not able to catch all the extreme values and tends to make predictions near the mean.

Overall the models are able to predict the major trends in the HMT output.

3.4 Comparing Linear Regression to ANNs

Previous work has shown that linear modeling methods have not proven to be successful for modeling parameters in a blast furnace. Linear regression is used to approximate a linear function between multiple inputs and one output. To implement a linear regression model, first the inputs in the system are placed in a $n \times p$ matrix $X$ where each column represents a specific input variable and each row represents a combination of data input values at each point in time ($n$). The goal is to find a vector of coefficients that minimizes the error over all points $n$. This vector, $\beta$, has $p$ rows and one column where each row is a coefficient for a particular input variable.
In the output vector $y$ each row represents an output at a specific point in time. The equations for the model are as follows [50]:

$$y = X\beta + \varepsilon$$

$$b = \beta = (XX)^{-1}Xy$$

$$\hat{y} = Xb = X(XX)^{-1}Xy$$

**Equation 10 : Linear Regression Model**

The predicted output is $\hat{y}$ which is based on the set of inputs. The Pearson coefficient and normalized mean squared errors are compared for the linear regression and artificial neural network models using the direct hourly averaged data.

![Pearson Coefficient of Neural Network vs. Linear Regression Predictions](image)

**Figure 17 : Pearson Coefficient Neural Network vs. Linear Regression**
Both models start at approximately the same place with the one-hour prediction model. The results show that the one-hour prediction model relies heavily on the last known HMT value because coefficient for the last known HMT value is significantly higher than the coefficients for the other input variables. After the one-hour prediction the NMSE for the models diverge significantly. The NMSE for the eight-hour prediction model is 0.506 for the ANN model and 1.597 for the linear regression model. The NMSE for the linear regression model rises above one at the two-hour prediction level and remains there through the eight-hour prediction. This means that the linear regression model performs worse from two to eight hours than just predicting the mean value each time.

The Pearson coefficient shows similar performance results. The one-hour linear regression model can predict trends at a comparable level as the ANN model.
However for the larger prediction horizons the accuracy of the linear regression model goes down significantly.

The results of this section show that the artificial neural network model outperforms the linear model at all prediction horizons and is therefore more successful for prediction in a blast furnace.
Hot metal silicon content is also important both for quality and control purposes. It reflects the internal state of the high-temperature lower region of the blast furnace. In this region any change in the silicon content corresponds to a similar change in the hot metal temperature. This reaction makes it very useful in blast furnace control. The extent of silica reduction indicates the how far other reactions have gone. The accurate prediction of silicon content can help stabilize the blast furnace operations. The silicon content of the hot metal is measured when pig iron is tapped. After the pig iron leaves the blast furnace, it goes through another process become steel[1].

4.1 Data Preprocessing

The data sets used to train the silicon content ANNs was extracted from the hot metal chemistry data and the data used for the hot metal temperature (HMT) model. Two distinct type of data sets were created in order to model future silicon content. The first type of data set consisted of input/output columns of the HMT data. These variables were used as the inputs in order to predict the output variable, Si%, which is a column that was extracted from hot metal chemistry data.

Since Si% was measured at a less frequent rate than the HMT input variables, the addition of the silicon column as the output column resulted in having large regions of the output variable that had the same constant value. Therefore,
linear interpolation and hourly averaging were performed. In addition, the best lags for each input column were implemented along with filling of missing values with previous values, and the normalization of each column.

The second type of silicon data set was processed the same way, but includes additional variables as inputs. The additional inputs used were taken from the Coke and Sinter data sets, and include the following: Coke Ash, Coke V.M., C.S.R., C.R.I., RDI, CaO, SiO₂, MgO, Al₂O₃, FeO and Fe. Domain experts had suggested that these additional input variables may be helpful in better explaining silicon content.
4.2 Prediction and Modeling of Silicon Content

4.2.1 Results using coke and sinter as inputs

Figure 19: Silicon Content 1-Hour Prediction with coke and sinter as inputs

The normalized mean squared error is .034 and the number of hidden nodes is 12. This model is able to predict the silicon content fairly accurately.
Figure 20: Silicon Content 2-Hour Prediction with coke and sinter as inputs
The normalized mean squared error is .039 and the number of hidden nodes is 20.
The error has increased slightly in this model.

Figure 21: Silicon Content 4-Hour Prediction using coke and sinter
The normalized mean squared error is .046 and the number of hidden nodes is 20.
The models for silicon content have a higher NMSE than the HMT models. The complexity of the models increases as the prediction horizon is pushed farther into the future. The one, two, and four-hour prediction models are accurately predicting the output, while the eight hour prediction model is only able to catch the trends.
4.2.2 Results Silicon Content without coke and sinter as inputs

The following graphs show the Silicon content predictions without coke and sinter as inputs. These models were created to see if the extra inputs were actually just adding noise rather than increasing the prediction accuracy.

Figure 23: Silicon Content 1-Hour Prediction without coke and sinter as inputs
The normalized mean squared error is .035 and the number of hidden nodes is 12. The performance of this model is very close to the 1-hour prediction model using coke and sinter as inputs. The coke and sinter input variable seem irrelevant to the model.
Figure 24: Silicon Content 2-Hour Prediction without coke and sinter as inputs
The normalized mean squared error is .037 and the number of hidden nodes is 13. This error is slightly better than the model with coke and sinter as inputs, but the complexity is much smaller. This model has only 13 hidden nodes as compared to 20 for the 2-hour prediction model with coke and sinter as inputs.
Figure 25: Silicon Content 4-Hour prediction without coke and sinter as inputs

The normalized mean squared error is .037 and the number of hidden nodes is 12. The performance of this network is significantly better than the 4 hour prediction network with coke and sinter as inputs. The reason for this could be because the data is less noisy with fewer inputs.

The models without coke and sinter as inputs perform the same as with these inputs for one and two hour predictions. However, at four hours the networks without the extra inputs perform better (NMSE of .037 vs. .046). This suggests that the extra inputs create more noise in the data. As shown in the figure below, the complexity of the models increases significantly with the extra inputs.
The complexity of the networks start out the same at the one-hour prediction level. This is because the one-hour prediction relies heavily on the last known HMT value. As the prediction horizon increases the number of hidden nodes necessary for the network increases rapidly for the model with coke and sinter as inputs. The extra inputs increase the complexity of the model when the number of data points stays constant. As the number of inputs increase, the number of data points used to train the network should also increase. A limited number of data points caused the model to prefer a fewer number of input variables. According to the model coke and sinter do not help the prediction of the network but rather hamper it because the data set size is limited.
Slag is produced when impurities in the iron ore, such as silicon dioxide, react with limestone. Once produced, it floats on top of the molten iron and can be drawn off separately. Slag plays an important role in reduction processes where liquid iron is produced. It absorbs undesirable components contained in the raw material. The quality of the hot metal is determined by the concentration of silicon, manganese, phosphorus and sulfur which depend on temperature, slag basicity and the ratio of the slag and metal quantities [1].

5.1 Data Preprocessing

The inputs of the data sets used to train slag basicity prediction have the HMT data as their inputs, as well as the following columns of the coke and sinter data: Coke Ash, Coke V.M., C.S.R., C.R.I., RDI, CaO, SiO₂, MgO, Al₂O₃, FeO and Fe. The output was taken from the slag chemistry data and is defined as:

\[(\text{CaO} + \text{MgO})/\text{SiO}_2\]

The data sets for slag basicity were preprocessed in the same way as the silicon content data sets. Slag basicity was also measured at a less frequent rate than the HMT input variables and the addition of the slag basicity column as the output column resulted in having large regions of the output variable that had the same value. Therefore, linear interpolation and hourly averaging were performed. In addition, the usual practices of implementing the best lags for each input column,
filling of missing values with previous values, and normalization of each column were also implemented.

Two different data sets were also created for slag basicity. In one data set there were a significantly less number of data points. This is because coke and sinter data points needed to be attached to the existing HMT data set. However for each HMT data point the corresponding coke and sinter values do not always exist. Two data sets were created: one which drops the data point from the new data set and another which fills in empty values for the coke and sinter data when they do not exist. The comparison of the models produced by these different sets of data will show if it is better to use more data points that may be incomplete or fewer data points that are complete.
5.2 Prediction and Modeling of Slag Basicity

5.2.1 Slag Basicity when removing data points when coke and sinter do not exist

Figure 27: Slag Basicity 1-Hour Prediction when data points dropped
Data points are dropped if coke and sinter do not exist. The normalized mean squared error is .022 and the number of hidden nodes is 16. The actual slag basicity values are represented by the dashed line and the network-predicted values are represented by the solid line.
Figure 28: Slag Basicity 2-Hour Prediction when data points dropped
The normalized mean squared error is .021 and the number of hidden nodes is 17.
5.2.2 *Slag Basicity when datapoints removed if coke and sinter do not exist*

The models for slag basicity do not seem to follow a clear trend as for HMT and silicon content. The number of hidden nodes are larger for the one hour prediction than for the four hour prediction and the normalized mean squared error is similar for one, two, and four hours.
The models below show slag basicity predictions when an empty value is filled in for coke and sinter when their values are not known. There are significantly more data points in this data set because no data points are dropped as above.

5.2.3 Slag Basicity when placing empty values for coke and sinter

![Graph showing slag basicity predictions](image)

**Figure 30 : Slag Basicity 1-Hour Prediction when empty values used**

Data points are filled with empty values if sinter and coke are missing. The normalized mean squared error is .026 and the number of hidden nodes is 15. The complexity and error for this model is similar to the 1-hour prediction model when data points are dropped.
Figure 31: Slag Basicity 2-Hour Prediction when empty values used
Empty values are filled in for coke and sinter when they do not exist. The
normalized mean squared error is .028 and the number of hidden nodes is 10.
Figure 32: Slag Basicity 4-Hour Prediction when empty values used
Empty values are filled in if values are missing. The normalized mean squared error is 0.036 and the number of hidden nodes is 12.

The models with empty values for coke and sinter when their data did not exist performed worse than when the data point was dropped for all prediction horizons. This suggests that coke and sinter are extremely important in slag basicity predictions and when empty values are placed used when their values do not exist, the data set is too noisy.
The model containing only data points that include coke and sinter values outperforms the other model even at the one-hour prediction level. The complexity of this model is also significantly less than the model with empty values.
Sinter quality is represented in terms of its granulometry (cumulative $+10\text{mm}\%$) and the reduction degradation index (RDI). Sinter quality is a function of the chemical reactions that take place in the furnace. These include transfer of heat between gas and solids, combustion of coke breeze, and the decomposition of limestone [1].

6.1 Sinter Data Preprocessing

The two sinter output variables are granulometry (cumulative $+10\text{mm}\%$ or $+10$) and the reduction degradation index (RDI).

The raw data received for the analysis of the sinter output variables were not measured at consistent intervals. The problem with this is that the NNRUN software assumes that there is a constant lag between each data point and the data set contains these data points in order.

Three solutions were proposed to solve this problem:

1. The first was to modify the NNRUN software so that it accepts a date field. This change would allow the networks to take into account variable lags between data points.
2. The second solution was to place empty values in the data set when a value was missing at a constant lag.

3. The third method was to average the data points daily and fill in the missing values using linear interpolation or the step function.

The first solution is the cleanest in that it allows us to train networks using the exact data produced at the steel plant. However adding a date field to the software would involve a large amount of time and resources and add complexity to the system. It is will probably be incorporated in future versions of the software. The second solution was also not feasible because of the lower quality of data used, approximately 80% of the data set would become empty values.

The third solution was the most practical. The two proposed methods for this solution were linear interpolation and the step function to fill in the missing values. Linear interpolation requires two values, the last known value and the next value to interpolate the in between value. Since we are trying to predict future values using previous data points the data should not be altered in a way that it can use future data points. However the step function causes more inaccuracies in the data because if there is long period without any measured data points, the step function would fill in all these missing values with the last known previous value. This would cause that value to become overly represented in the model. The best way to avoid the problems associated with both of these methods is to use an extrapolation method using gradient calculations from the previously known data points. This
possibility may be explored in the future. The networks produced for preliminary results and analysis used linear interpolation.

There were originally twenty-nine input variables. Fifteen input variables were chosen by SteelCorp based on domain knowledge. They are: machine speed, ignition hood temperature, main suction, damper opening, lambda, bed height, percentage of coke breeze (trimming), percentage of mixed flux, percentage of lime (trimming), percentage of calcium oxide (CaO), percentage of silicon oxide (SiO₂), percentage of phosphorus oxide (P₂O₅), percentage of manganese oxide (MgO), percentage of aluminum trioxide, and iron oxide (FeO). The variables that were removed were not highly correlated with the sinter output variables.

Three distinct data sets were created to model the sinter conditions. The sinter models are different from the previous models because they are used to predict the sinter output variables one, two, and four days in the future, rather than hours. There were six models that were processed producing six different ANN networks each with a different configuration optimized for each data set. There are six different models because there are two outputs, RDI and +10, each with three different types of processed data sets. These data sets consist of the raw data, averaged data, and linearly interpolated data.
The following graph compares the NMSE for the 3 different data sets for RDI predicting one day into the future.

![Graph showing NMSE for RDI Raw, Averaged, and Linearly Interpolated Data]

**Figure 34 : NMSE for RDI Raw, Averaged, and Linearly Interpolated Data**

This shows that the models trained with the raw data are the most accurate in predicting future values. This could be because the averaged and linearly interpolated data reduce the variations in the data sets and make it more difficult for the model to catch the trends. The best performing network used the data in its original form rather than data that was changed in any way.
The following graph compares the NMSE for the three different data sets for +10 predicting one day into the future.

![Graph showing NMSE for Raw, Averaged, and Linearly Interpolated RDI Data]

Figure 35 : NMSE for Raw, Averaged, and Linearly Interpolated RDI Data

6.2 Prediction of RDI and +10 Results

The +10 ANN models had different results than the RDI models. For +10, the linearly interpolated data set ANN models were the most accurate. Noise among the input variables in the data is reduced by linear interpolation and averaging. However the variation in the data is also reduced.
6.2.1 RDI Results

The normalized mean squared error is 0.032 and the number of hidden nodes is 13 in this network. The actual values are represented by the dashed line and the predicted values are the solid line.

**Figure 36: RDI 1-Day Prediction**

The normalized mean squared error is 0.032 and the number of hidden nodes is 13 in this network. The actual values are represented by the dashed line and the predicted values are the solid line.
Figure 37: RDI 2-Day Prediction
The normalized mean squared error is .043 and the number of hidden nodes is 13.

Figure 38: RDI 4-Day Prediction
The normalized mean squared error is .047 and there are 18 hidden nodes.
The prediction models for RDI showed much poorer results than for the previous output variables. The ANNs were sometimes able to catch the trend for future values, but often predicted values close to the mean. The NMSE error got significantly worse as the models tried to predict further into the future.
6.2.2 +10 Results

Figure 39: +10 1-Day Prediction
The normalized mean squared error is .029 and the number of hidden nodes is 13.
The normalized mean squared error is .039 and the number of hidden nodes is 13.

The normalized mean squared error is .033 and there are 18 hidden nodes.
The following graph compares NMSE for +10 and RDI for one, two, and four day future predictions.

![Graph comparing NMSE for +10 and RDI](image)

**Figure 42: NMSE RDI vs. +10**

The +10 ANN models showed slightly better results than RDI. The normalized mean squared errors were lower for each +10 network as compared to RDI. The networks seemed to be catching the major trends in the data.
The next step is to apply data mining techniques to control the conditions in the blast furnace. A prediction can indicate the future condition of the blast furnace based on the current conditions, but this is only useful if the output variable can be controlled when its value is predicted to be out of desirable range. Control of blast furnace parameters such as hot metal temperature is difficult because there are many inter-dependent variables involved each with different non-linear relationships with the output. Our Artificial Neural Network HMT prediction models had eleven input variables that were used to predict the HMT output. For control of HMT we will need to narrow this list down to two or three key parameters.

7.1 Identifying Key Control Parameters

Identifying the key parameters for the prediction of the blast furnace variables is very important in building a control system. Researchers have found that control systems such as neuro-controllers or neural networks are much more accurate with a small number of control parameters [9][32].

7.1.1 Control Parameter Reduction Techniques

Principal Components Analysis is one technique for reducing the number of input parameters in a model. The PCA method involves reducing the inputs to an input combination that is a linear combination of the original inputs. The inputs in
this new set are orthogonal to each other and are ranked by their importance [48].
The control system can now determine what changes can be made to the inputs to get a desired output. The problem with this method is that the inputs are now combinations of the original inputs. Therefore, any change that is made will have to be transformed back to the original inputs. This is not very practical in the blast furnace environment.

Dithering is another method that can be used to determine the importance of input variables. This method was implemented by Bhattacharjee [12] at Tata Steel. This approach involves changing one input variable while keeping all the other inputs constant. The change in the output is measured and the importance of the input variable is determined by the magnitude of the change in the output. Bhattacharjee mentioned, however, that one serious drawback is that this method does not take into account the non-linearities in the system. This method was implemented on the Reduction Degradation Index output variable.

Another method used by Bhattacharjee [12] of Tata Steel is Information Theoretic Analysis. This technique ranks the input parameters based on the information content of the output. Given a universe of messages: \( M = \{m_1, m_2, \ldots, m_n\} \) and a probability: \( p(m_i) \), the information content of a message in \( M \) is:
\[ I(M) = \sum_{i=1}^{n} - p(m_i) \cdot \ln(p(m_i)) \]

Equation 11 : Information Content

Since there are multiple input parameters, many combinations can result in a desired output value. The ID3 algorithm is used to classify the input parameters. The initial set of inputs was reduced from twenty-four to fifteen and the results showed that the rises and falls in HMT were predicted fairly well.

7.1.2 Sensitivity Analysis: Derivative Approach

Our research focuses uses a derivative approach to do a sensitivity analysis on the input variables with the output. This approach uses the neural network to try to find which variables are the most important by calculating the derivative of the output with respect to each input variable. This is based on the derivations of Takenaga et al [20] for a one hidden layer neural network. The topology of the neural network is depicted in the figure below:
FEED-FORWARD NEURAL NETWORK WITH 2 HIDDEN LAYERS

Input Layer (Layer 1)  Hidden Layer (Layer 2)  Hidden Layer (Layer 3)

- $X_i$ is the input to neuron $i$ at the input layer
- $X_i(j)$ is the total input to neuron $i$ at layer $j$.
- $Y_i(j)$ is the output of node $i$ in layer $j$.
- $W_{ij}(k)$ is the weight from node $i$ in layer $k$ to node $j$ in layer $k+1$.

$$X_i(j) = \left[ \sum_{k=1}^{n(j-1)} W_{ki}(j-1) Y_k(j-1) \right]$$

$n(i)$ nodes in layer $i$

Figure 43: Neural Network with 2 Hidden Layers
To find the derivative of the output with respect to the input the one hidden layer formula derived by Takenaga [20] was extended.

\[ \frac{\partial Y}{\partial X_i} = \]

\[
\frac{\partial (f(X(4)))}{\partial X(4)} \ast \sum_{p=1}^{n(3)} W_{p1}(3) \left[ \sum_{j=1}^{n(2)} W_{j}(1) \left( \frac{\partial f(X_{j}(2))}{\partial X_{j}(2)} \right) \right] \frac{\partial (f(X_{p}(3)))}{\partial X_{p}(3)}
\]

**Equation 12 : Derivative of 2-Layer Neural Network**

This formula is applicable to the two hidden layer neural network shown above. The derivative that is calculated varies over time since the input and output values are always changing. By graphing this derivative over time we can see the general trend and the approximate magnitude of a particular input’s derivative as compared to others.

Some samples of the derivative changes for each variable are shown below.
Figure 44: Derivative for CHRGDTIM & COKTOTAL

Figure 45: Derivative for ECO & EH2
The following table shows the magnitude of the derivative of the HMT output with each input variable.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Variable Name</th>
<th>Average Magnitude of Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O2ENR_PV</td>
<td>0.2642</td>
</tr>
<tr>
<td>2</td>
<td>OREBYCOKE</td>
<td>0.2348</td>
</tr>
<tr>
<td>3</td>
<td>INJ_ACT</td>
<td>0.1610</td>
</tr>
<tr>
<td>4</td>
<td>ECO</td>
<td>0.1303</td>
</tr>
<tr>
<td>5</td>
<td>FIC46</td>
<td>0.1122</td>
</tr>
<tr>
<td>6</td>
<td>G1HF</td>
<td>0.1115</td>
</tr>
<tr>
<td>7</td>
<td>TIC91</td>
<td>0.1023</td>
</tr>
<tr>
<td>8</td>
<td>G2HF</td>
<td>0.1011</td>
</tr>
<tr>
<td>9</td>
<td>COKETOTAL</td>
<td>0.0963</td>
</tr>
<tr>
<td>10</td>
<td>CHRGDTIM</td>
<td>0.0914</td>
</tr>
<tr>
<td>11</td>
<td>EH2</td>
<td>0.0432</td>
</tr>
</tbody>
</table>

From the graphs and the table above we can see that the most influential variable in controlling the HMT output will be the O2ENR_PV and OREBYCOKE (ore by coke ratio).

7.2 Dithering vs. Derivatives

A method called dithering was used to confirm the derivative results. With dithering each input variable is changed by a small amount (.01) and an approximate derivative is calculated by dividing the change in output by the change in input. The following graphs compare the results of the dithering and derivative approach.
Figure 46: Dithering vs. Derivative for CHRGDTIM

Figure 47: Dithering vs. Derivative for COKTOTAL
For the each input the derivative and sensitivity graphs show the same trends. The derivative analysis is able to catch more of the extreme values as compared to dithering, but the fact that dithering and derivative analysis produce very close results provides verifies that derivative results are correct.

### 7.3 Control Systems

There are various different technologies that can be used for the control system. The controller could use neural networks, genetic algorithms, or decision trees [32]. Analysis of three types of neurocontrollers is presented below [32].

#### 7.3.1 Supervised Control

Supervised control is useful if there is a person with domain knowledge present who knows how to control the system. A neural network receives as input the error between the process (being controlled) output and the desired reference signal and outputs a control signal. The supervisor receives the same input and also produces a control signal. The error between the supervisor's control signal and the ANN control signal is sent back to the ANN as input. The ANN can now adjust its weights based on this control error. Hence, the neural network is being trained by the supervisor. Supervised control would be difficult to implement in a blast furnace environment because there is no expert supervisor who would know how to control a blast furnace well enough to accurately train a neural network.
7.3.2 Direct Inverse Control

Direct inverse control tries to find the inverse ANN function. If an inverse function is found then controlling the system would involve inputting the desired output into the inversed ANN. The ANN would be able to produce the inputs that could result in a particular output.

This approach seems to be the most applicable to the blast furnace problem. For example if the desired hot metal temperature is known, then the inverse function should be able to tell which input variables to change and how much to change them to get this final temperature.

Since blast furnace output parameters are influenced by multiple variables, it is difficult to find an inverse function for this problem. There are a number of input combinations that can produce one single output. The inverse function most likely will not converge to a result.

7.3.3 Control Conclusions

This paper has looked at two popular neurocontrol techniques that may apply to the blast furnace control problem. The limitations of these methods seem to conclude that there is no current solution using neurocontrollers for the blast furnace control problem.
8. Conclusion

The first part of this thesis examines the capability of feed-forward neural networks for modeling hot metal temperature, silicon content, slag basicity, and sinter in a blast furnace. Different preprocessing techniques along with software that automatically finds the best multi-layer perceptron neural network are presented. Some enhancements and improvements that have been made from previous work in this area include: the expansion of prediction to other parameters besides hot metal temperature, the analysis of automatic vs. domain knowledge lag detection, and the comparison of one vs. two hidden layer networks, moving window vs. hourly averaging, and linear regression vs. artificial neural networks. As shown by the results, neural network methods have been able to successfully predict the hot metal temperature (3.3.4), silicon content (4.2), slag basicity (5.2), and sinter (6.2). The network models for hot metal temperature will be implemented at SteelCorp to improve the production process and quality of steel. Future work in this area could include the application of neural network techniques to other complex processes. In the area of blast furnaces the results can be further improved upon to get higher accuracies for the prediction of slag basicity and sinter.

The final part of this thesis discusses control in a blast furnace. A derivative sensitivity analysis is presented to find the variables that most influence hot metal temperature in a blast furnace. Different types of neurocontrollers are discussed. The prediction and control of blast furnace conditions will lead to improved quality of pig iron.
9. References


[42] Lu, Y. Meeting the Challenge of Intelligent System Technologies in the Iron and Steel Industry, Iron and Steel Engineer, September 1996.


