Modeling Carbon Monoxide Fields with Artificial Neural Networks

by

Tharshikan Srikannathasan

A thesis submitted in conformity with the requirements
for the degree of Masters of Science
Graduate Department of Physics
University of Toronto

Abstract

Modeling Carbon Monoxide Fields with Artificial Neural Networks

Tharshikan Srikannathasan
Masters of Science
Graduate Department of Physics
University of Toronto
2016

Current techniques for inverse modeling carbon monoxide emissions have large biases due to errors in atmospheric transport and tracer gas chemistry in atmospheric models. In addition, current observations used to constrain emissions on regional scales suffer from limited data and low precision measurements. Even with the multiple data stream approach, the covariances between different datasets can be challenging to quantify. A statistical approach offers a means of inverse modeling atmospheric carbon monoxide while avoiding the challenges of model errors We examine the use of a statistical model to simulate atmospheric concentrations of carbon monoxide via feed forward neural networks. This is a a necessary first step toward the use of such networks for the inverse modeling of carbon monoxide emissions. The GEOS-Chem chemical transport model is used to train the network. We show that a maximally connected network with a single hidden layer is sufficient to fit a time series of carbon monoxide concentrations over New York City.

Acknowledgements

Many thanks to Professor Dylan Jones for teaching a **dope** introductory class on data assimilation, and for taking me under his supervision. I look forward to our continued collaboration over the length of my doctoral training.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Carbon Monoxide

Carbon monoxide (CO) is an odorless, colorless, and toxic gas regulated in Canada. CO is the product of incomplete combustion and a byproduct of hydrocarbon oxidation. It is removed in the atmosphere by reacting with the hydroxide radical OH

$$CO + OH \rightarrow CO_2 + H \tag{1.1}$$

Similar reactions also remove other pollutant gases like methane (CH_4) and is thus responsible for the oxidative capacity of the atmosphere. Recent data shows that CO concentrations have increased by two to three times the pre-industrial level, largely due to the impact of anthropogenic activity [Haan et al., 1996]. CO is a known precursor of tropospheric ozone. Therefore, it is vital to understand the dynamics of CO and its sources in order to predict future atmospheric conditions and understand climate change. The lifetime of tropospheric CO is one to two months, long enough to track individual pollution events. CO sources are relatively large compared to other tracer gases. Thus, CO concentrations are easy to detect relative to the background levels. These features render CO a prime candidate for air quality studies.

1.2 Sources of CO

[Duncan et al., 2007] calculated the global budget for sources of CO from 1988 to 1997. The results are summarized in Table 1.1. Note that the most important surface sources are from the combustion of

Source	Emissions
Fossil Fuels	464-487
Bio-fuels	189
Biomass Burning	451-573
Biogenic NHMC	354-379
Methane Oxidation	778-861

Table 1.1: Global budget of CO sources from 1988 to 1997 (Tg/year) [Duncan et al., 2007]

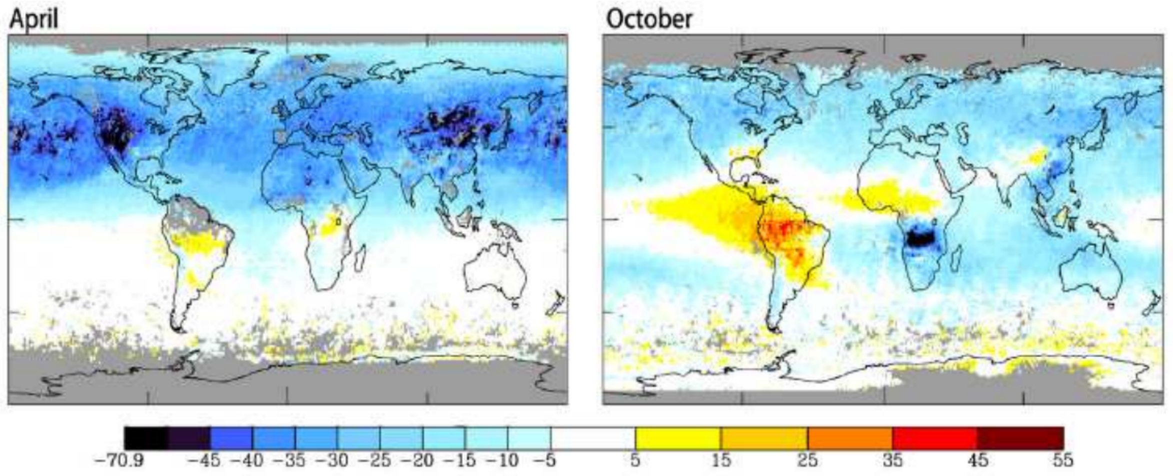

Figure 1.1: Differences between the chemical transport multi-model mean CO and MOPITT V3 CO for 2000-2004 (ppbv) at 500 hPa. The background CO field is 200 ppbv. Plot from [Shindell et al., 2006].

fossil fuels, bio-fuels, and biomass. In the troposphere, the oxidation of methane provides the bulk of the background CO concentration. Non-methane hydrocarbons (NHMCs) come from surface vegetation and are a source of CO when oxidized.

There are large uncertainties with the estimation of regional CO sources. These uncertainties propagate through atmospheric models and lead to sizable errors in the simulation of atmospheric CO fields. [Bian et al., 2007] used six different models of biomass burning emissions to simulate CO concentrations. The range of the uncertainty in biomass burning was found to be 129 Tg/year. This only accounts for 6% of the the total CO emissions, but regional emission estimates varied by factors of four or more. The uncertainties resulted in large variability between the various simulations. Another study, conducted by [Shindell et al., 2006] compared 26 different chemical transport models (CTMs) and found that they successfully reproduced the large scale spatial and temporal features of CO. However, the study also revealed significant biases between modeled and observed CO (See Figure 1.1). Note that the average CO concentration is underestimated in the northern hemisphere during April and over the biomass burning regions of Africa during October. It turns out that CO concentrations are consistently underestimated in the northern hemisphere throughout the year [Shindell et al., 2006]. This is due to the underestimation of CO surface emissions caused by anthropogenic activity. Furthermore, the variability between the individual models was large even when the mean agreed with the observations. Again, this is due to incorrect estimations of emissions from NHMCs and gas oxidation. The study also found that transport was not as important as differences in NHMC estimates. [Liu et al., 2010] used the GEO-Chem CTM to analyze CO variation over tropical latitudes. The results were compared to the Microwave Limb Sounder (MLS) and Tropospheric Emission Satellite (TES) retrievals of CO. Again, spatial and seasonal CO variability was captured but there were underestimations in the biomass burning in the southern latitudes. The study also showed a temporal displacement in CO seasonal maxima in the troposphere over South America. The findings were linked to erroneous vertical transport and NHMC sources. Another study was conducted by [Ott et al., 2011]) concerning the influence of convective transport on CO fields. They perturbed the GEOS-5 convective parameters in an ensemble of eight simulations. The parameter changes caused large variability on CO concentrations over Africa, Indonesia, and South America.

These areas notably feature significant biomass burning and convection in the lower atmosphere. The aforementioned results point to similar issues with modeling CO distribution: uncertainties in source estimates and in the transport. It is clear that one must better constrain estimates of CO sources to increase the accuracy of the CO simulations that rely on them.

1.2.1 Estimation of CO Sources

The two most common methods of calculating CO emissions are referred to as the bottom-up, and top-down approaches. The former method relies on regional and international energy statistics for estimates of both fossil fuel and biofuel sources. This data is combined with geo-information such as population statistics, road type, and land cover data to approximate anthropogenic sources of CO. However, these data are often extrapolated in space and time. For example, statistics from countries with limited data are replaced with information from western countries [Streets et al., 2003]. For biomass burning, satellite observations are combined with ecosystem types to produce CO emission estimates. Despite this, it is still difficult to constrain burned area and fuel loads. [Van der Werf et al., 2010] showed that biomass burning calculations are uncertain to a minimum of 20%.

The top-down approach uses ideas from the field of inverse modeling. This method attempts to minimize the differences between CTM simulations and observed CO concentrations by adjusting sources of CO. Most studies have used low-resolution Bayesian methods. That is, emissions are grouped by region and continent. Emissions are then scaled with respect to each unit to minimize the residual error between the model and the observations. However, this process leads to aggregation errors which have been shown to significantly impact emission estimates [Jiang et al., 2011]. Another top down approach considers techniques that fall under variational methods. Four dimensional variational data assimilation (4D-Var) is one such technique. 4D Var uses so called adjoint methods, which have the advantage of allowing emission estimates to be made at the native resolution of the CTM. These methods have immediate problems concerning data resolution limits in both the spatial and temporal domain. Aircraft and surface based observations don't guarantee a well constrained estimate on CO sources. [Jones et al., 2009] used CO retreivals from TES and MOPITT to constrain sources using the GEOS-Chem CTM. They found that differences in *a posteriori* emission estimates were roughly 20% of the *a priori* source information.

1.3 Objective of Manuscript

To summarize the previous sections, it is apparent that model biases are a challenge for the inverse modeling of CO emissions. Both satellite and surface CO data streams are insufficient to quantify emissions on regional subscales. Combining datasets is one solutions of overcoming the data sparsity problems. However, calculating covariances between the different streams of data can prove to be challenging. In this manuscript we use statistical methods to model CO fields. These methods avoid calculating model biases and covariances. This allows combinations of multivariate data to be included in the modeling of CO concentrations. It will also circumvent the challenges associated with model biases. This gain comes at the cost of losing information about the structure of the model. In particular, the free parameters of statistical models are often difficult to interpret outside of a handful of toy models. Our long term goal is to use a statitical approach to estimate CO sources. As a first step, the work presented in this report is focused on predicting CO concentrations using neural networks. If we can successfully predict CO fields then that information will lead to better estimates of CO sources.

First, the reader will be introduced to notion of artificial neural networks. The notion of training a neural network will be discussed, along with a derivation of the back-propagation algorithm. The network will applied to a nonlinear univariate regression problem. Then we attempt to model a Lorentz dynamical system using a neural network generator and discuss some limitations of FFNNs. Finally, we train the network on the GEOS-Chem chemical transport model to predict yearly CO fields.

Chapter 2

Artificial Neural Networks

2.1 What is a Neural Network?

In the field of biology, a *neuron* is a cell that processes information by way of electrical and chemical stimulation. These cells are at the core of nature's computing architecture, the nervous system. Neurons are made up of a soma, axons, and dendrites. The soma is the body of the neuron while signals are captured by dendrites and transmitted through the axons. When a neuron receives a signal, it will output a response if the input is higher than some internal threshold. A mesh of many of these neurons is called a neural network. The human brain has an estimated 100 billion such neurons interconnected in complex networks. It is believed the dense computing power of the brain is a direct result of this topology. In statistics and machine learning, an *artificial neural network* (ANN) is a network of computing units loosely inspired by the aforementioned systems seen in biology. ANNs are used to approximate functions and classify patterns. An artificial neuron is a simple computing node that takes in a signal s and fires according to an internal activation rule

$$n(s) = \begin{cases} 1 & \text{if } s \geq t \\ 0 & \text{if } s < t \end{cases}$$

where t is the threshold of the neuron. The signal comes from the outputs of other neurons

$$\begin{aligned} s &= w_1x_1 + w_2x_2 + ... + w_Nx_N \\ &= \mathbf{x}^T\mathbf{w} \end{aligned}$$

where $\mathbf{x}^T\mathbf{w}$ is the euclidean inner product between a vector of weights $\mathbf{w}$ and a vector of inputs $\mathbf{x}$. In this language the neuron can be defined as

$$n(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \quad \text{where } z = \mathbf{x}^T\mathbf{w} + b \tag{2.1}$$

where b is the so called bias of the neuron. With this definition, one can pick weights such that the neuron can model a NAND gate. Recall that a NAND gate is an inverted AND gate. Therefore we wish

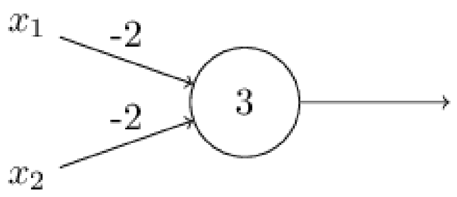

Figure 2.1: A single neuron NAND gate with two identical weights, $w_1 = w_2 = -2$ and a bias $b = 3$

to take two inputs A and B and output NOT(A and B). Consider a neuron that takes two binary inputs (x_1, x_2) and outputs a binary response according to the architecture in Figure 2.1. In this simple case, the weights and bias are trivial to choose. For example, if $(x_1, x_2) = (1, 0)$ then we have

$$\begin{aligned} n\left(w_1x_1 + w_2x_2 + b\right) &= n\Big((-2)(1) + (-2)(0) + 3\Big) \\ &= n(1) \\ &= 1 \end{aligned}$$

The other permutations of inputs clearly describe the properties of the function we wished to recreate. The natural question to ask is one of generality, how does a step function model complex time series data or classify animals in images? The answer lies in the organization of many neurons and the resulting topology that is formed from the grouping.

2.2 Architectures, Activation, and Assimilation

The *architecture* of a neural network refers to the structure that exists between individual nodes of a network. This paper will be discussing *feed-forward neural networks* (FFNNs). A FFNN consists of L layers of neurons with each layer containing n_l single neurons, where $l = 1, 2, ..., L$. The first layer of neurons are called the input layer. This layer is unique in the sense that it does not compute, but only sends the initial information into the network. From this point, the information is sent from one layer to the next and each neuron in a succeeding layer will fire according to it's activation function and input signal from the preceding layer of neurons.

The *activation function* of a FFNN refers to a non-linearity $\phi(\cdot)$ that is applied after the node receives a signal. For practical purposes, a smooth approximation to the step function is preferred. Hence ϕ is usually some sort of sigmodial function such as the logistic function or the tanh function (Figure 2.2). These functions have convenient analytic properties that one can take advantage of to train the network. The last layer of the network is called the output layer. This layer takes the signal from the last layer of hidden neurons and sends the signal through a final transfer function, which translates the output of the network to a result that can be understood within the context of the problem. Such transfer functions include an identity mapping (for regression) or a soft-max mapping for multi-label classification. FFNN neurons are all maximally connected. This means that every neuron in layer l is connected to the neurons in layers $l-1$ and $l+1$. The network from Figure 2.1 can be considered a FFNN with 0 hidden layers, 2 input neurons and 1 output neuron. More generally, the strength of connectivity between neuron j in layer $l-1$ and neuron k in layer l is denoted by the weight w^l_{jk} and neuron j in layer l carries a bias of

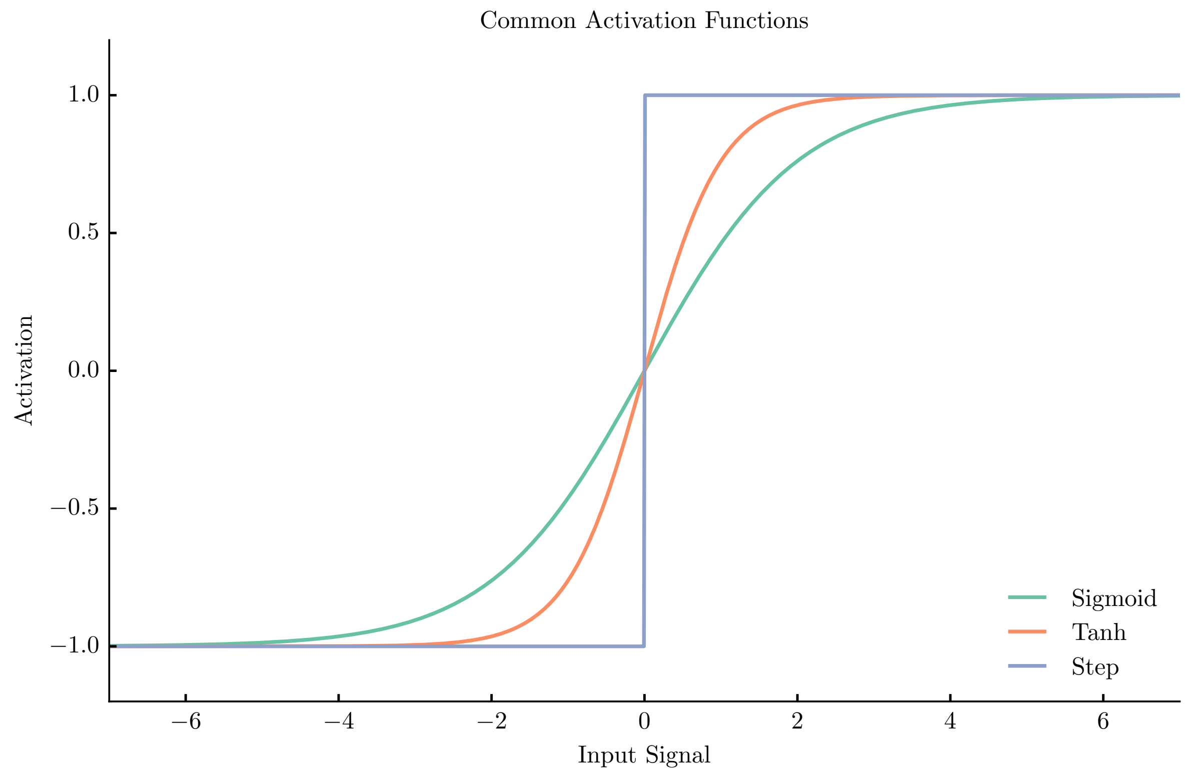

Figure 2.2: Three common activation functions used in neural networks.

b_j^l. One of the most interesting aspects of FFNNs is the ability to compute ***any*** function with a single hidden layer of neurons. That is, given any function f there exists a FFNN that can closely approximate f. This is stated formally as the **Universal Approximation Theorem** (UAT).

2.2.1 The Universal Approximation Theorem

Let $g(\cdot)$ be a non-constant, bounded, and monotonically-increasing continuous function. Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of continuous functions on I_m is denoted by $C(I_m)$. Then, given any function $f \,\epsilon\, C(I_m)$ and $\epsilon > 0$, there exists an integer n, real constants α_i, $\beta_i \,\epsilon\, \mathbb{R}$ and real vectors $\boldsymbol{\omega}_i \,\epsilon\, \mathbb{R}^m$, where $i = 1, ..., n$, such that we may define

$$F(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, g(\boldsymbol{\omega}_i^T \mathbf{x} + \beta_i)$$

as an approximate realization of the function f where f is independent of g_i; that is,

$$|F(x) - f(x)| < \epsilon$$

for all $x \,\epsilon\, I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$. Furthermore, this theorem still holds when replacing I_m with any compact subset of $\mathbb{R}^m$.

While the *rigorous* proof of the UAT is far beyond the scope of this manuscript, the following discussion will sketch how the UAT connects to FFNNs. Consider a FFNN with a single hidden layer with n hidden neurons. For simplicity let the input be two dimensional and the output one dimensional. Let us denote these variables as $\mathbf{x} = (x_1, x_2)$, and y. The first step to computing the signals z_i into the hidden layer of neurons, where $i = 1, 2, ..., n$

$$\begin{aligned} z_1 &= w^2_{11}x_1 + w^2_{21}x_2 + b^2_1 \\ z_2 &= w^2_{12}x_1 + w^2_{22}x_2 + b^2_2 \\ &\vdots \\ z_n &= w^2_{1n}x_1 + w^2_{2n}x_2 + b^2_n \end{aligned}$$

The hidden layer has a continuous output $a_i = \phi(z_i)$, where ϕ is a sigmoidal non-linearity. These activations are sent from the hidden layer to the output layer via the 2nd set of weights. Here, the final non-linearity is applied to give the output y

$$y = \psi\left(w^3_{11}a_1 + w^3_{21}a_2 + ... + w^3_{n1}a_n\right)$$

If ψ is the identity mapping[1], then we have exactly the construction of a function as shown in the UAT

$$\begin{aligned} y &= w^3_{11}\phi({\mathbf{w}^2_1}^T\mathbf{x} + b^2_1) + ... + w^3_{n1}\phi({\mathbf{w}^2_n}^T\mathbf{x} + b^2_n) \\ &= \sum_{i=1}^{n} w^3_{i1}\phi({\mathbf{w}^2_i}^T\mathbf{x} + b^2_i) \end{aligned} \tag{2.2}$$

Therefore by the UAT the construction above will approximate any one dimensional function $f \in \mathbb{R}$.

The astute reader might be left with more questions at the end of the preceding analysis. How many hidden neurons does it take to approximate a function? How does one choose the weights and biases such that the approximation is a *good* one? While the UAT guarantees the existence of such a network, little is claimed about how to go about constructing the architecture. The problem of choosing parameters becomes intractable as the number of hidden neurons increases.

In a FFNN of one dimensional inputs and outputs, the number of adjustable parameters is given by $N_p = 3n$, where n is the number of hidden neurons in a single hidden layer. *Deep neural networks* are often used to classify images. In the problem of digit classification, a network ingests an input image of size 28x28 which means the input space is 784 dimensional. The output space is 10 dimensional which corresponds to the 10 possible digits that can be recognized. Suppose further that a four layer FFNN is used to solve this problem. Even if there are only 10 neurons in each hidden layer, that corresponds to over 8000 adjustable weights and biases. Thus, in order to proceed we must consider an algorithmic way of optimizing these values to produce the *best* network possible. This is referred to as the *learning problem* in machine learning and it brings us to the final aspect of ANNs, assimilation.

2.3 The Learning Problem

Suppose we are given a set of training examples $S = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, ..., N | (\mathbf{x}_i, \mathbf{y}_i) \,\epsilon\, \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}\}$. The goal is to construct an L layer FFNN that takes $\mathbf{x}_i$ as an input and outputs an estimate $\boldsymbol{\eta}_i(\Theta) = \boldsymbol{\eta}(\mathbf{x}_i|\Theta)$,

[1]This holds for all ψ, not just linear functions

where $\Theta = \{\mathbf{W}, \mathbf{b}\}$ is the set of weights and biases describing the network. We seek a parameter set $\Theta = \Theta^*$ such that $\boldsymbol{\eta}_i(\Theta^*) \approx \mathbf{y}_i$. Since Θ is a random variable, Θ^* must be correspond to the parameter set that is most likely to be observed given what is known about the training data. We construct this probability distribution under the following assumptions. Assume each $\boldsymbol{\eta}_i$ is independent and identically distributed (i.i.d) and normally distributed around the target $\mathbf{y}_i$. Furthermore, we constrain the weights to be normally distributed around $\mathbf{0}$ to avoid over-fitting of the training data and all biases are assumed to be drawn from a uniform distribution over some subset of the real line. Bayes Theorem can now be invoked to calculate the posterior distribution of Θ given the training examples

$$P(\Theta|S) = \frac{P(S|\Theta)P(\Theta)}{\int P(S|\Theta)P(\Theta)d\Theta} \tag{2.3}$$

Maximizing this function is equivalent to minimizing $-ln\left[P(\Theta|S)\right]$. If we discard additive constants, the problem of maximizing $P(\Theta|S)$ transforms into the problem of minimizing the so called *cost function*

$$E(\Theta|\mathbf{X}) = \frac{1}{2N}\left[\boldsymbol{\eta}(\mathbf{X}|\Theta) - \mathbf{Y}\right]\left[\boldsymbol{\eta}(\mathbf{X}|\Theta) - \mathbf{Y}\right]^T + \frac{\lambda}{2N_w}\sum_l \mathbf{W}_l\mathbf{W}_l^T$$

The notation has been vectorized for brevity. $\mathbf{X}$ is the $N \times d_x$ input data matrix and $\mathbf{Y}$ is the $N \times d_y$ target data matrix. $\boldsymbol{\eta}(\mathbf{X})$ is the output of the network, N_w is the total number of weights, λ is the regularization hyper-parameter, and $\mathbf{W}_l$ are the matrix of weights connecting layers $l-1$ and l such that each element of $\mathbf{W}_l$ is w^l_{jk}. Now that we have a well defined objective in the learning problem, we discuss the algorithms used to find Θ^*.

2.3.1 Minimizing $E(\Theta)$

In order to computationally minimize $E(\Theta)$ it is necessary to know the gradient of the cost with respect to all elements of Θ. The *back-propagation algorithm*, published in 1986 by [Rumelhart et al., 1988] describes an algorithmic procedure for computing the gradient of the cost function. Note that for our network from section 2.3 we have $\Theta = \{\mathbf{W}_2, \mathbf{W}_3, ..., \mathbf{W}_L, \mathbf{b}_2, ..., \mathbf{b}_L\}$. Let us denote the inputs and outputs to layer l as $\mathbf{z}_l$ and $\mathbf{a}_l$ respectively. The signals are related through the activation function[2]

$$\mathbf{a}_l = \psi(\mathbf{z}_l),\, \mathbf{z}_l = \mathbf{a}_{l-1}\mathbf{W}_l + \mathbf{b}_l \tag{2.4}$$

Note that $\mathbf{z}_1$ does not exist, as there is no signal going into layer 1, $\mathbf{a}_L = \boldsymbol{\eta}(\mathbf{X})$[3], and $\mathbf{a}_1 = \mathbf{X}$. The recursive structure of the network implies that derivatives in layer l depend on layers *ahead* of them. To see this note that for $l = L$ we have

$$\frac{\partial E}{\partial \mathbf{W}_L} = \frac{1}{N}\mathbf{a}_{L-1}^T\left[\boldsymbol{\eta}(\mathbf{X}) - \mathbf{Y}\right] \odot \psi'(\mathbf{z}_L) + \frac{\lambda}{N_w}\mathbf{W}_L$$

$$\frac{\partial E}{\partial \mathbf{b}_L} = \frac{1}{N}\mathbf{1}^T\left[\boldsymbol{\eta}(\mathbf{X}) - \mathbf{Y}\right] \odot \psi'(\mathbf{z}_L)$$

[2]Note that the sum between the weighted components and the bias is broadcast row-wise.
[3]The Θ has been suppressed here for brevity.

Where $\odot$ represents the element-wise product, $\psi'(\cdot)$ is the derivative of the transfer function applied element-wise, and $\mathbf{1}$ is a vector of 1's. Defining $\boldsymbol{\delta}_L = \frac{1}{N}\left[\boldsymbol{\eta}(\mathbf{X}) - \mathbf{Y}\right] \odot \psi'(\mathbf{z}_L)$ we have

$$\frac{\partial E}{\partial \mathbf{W}_L} = \mathbf{a}_{L-1}^T \boldsymbol{\delta}_L + \frac{\lambda}{N_w}\mathbf{W}_L$$

$$\frac{\partial E}{\partial \mathbf{b}_L} = \mathbf{1}^T \boldsymbol{\delta}_L$$

A pattern reveals itself as we consider $l = L-1$

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{W}_{L-1}} &= \frac{1}{N}\mathbf{a}_{L-2}^T \boldsymbol{\delta}_L \mathbf{W}_L^T \odot \psi'(\mathbf{z}_{L-1}) + \frac{\lambda}{N_w}\mathbf{W}_{L-1} \\ &= \mathbf{a}_{L-2}^T \boldsymbol{\delta}_{L-1} + \frac{\lambda}{N_w}\mathbf{W}_{L-1}\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{b}_{L-1}} &= \frac{1}{N}\mathbf{1}^T \boldsymbol{\delta}_L \mathbf{W}_L^T \odot \psi'(\mathbf{z}_{L-1}) \\ &= \mathbf{1}^T \boldsymbol{\delta}_{L-1}\end{aligned}$$

where $\boldsymbol{\delta}_{L-1} = \frac{1}{N}\boldsymbol{\delta}_L \mathbf{W}_L^T \odot \psi'(\mathbf{z}_{L-1})$. This is extended by induction until $l = 2$ and we have the complete gradient of the cost function

$$\boldsymbol{\delta}_l = \frac{1}{N}\boldsymbol{\delta}_{l+1}\mathbf{W}_{l+1}^T \odot \psi'(\mathbf{z}_l) \tag{2.5}$$

$$\frac{\partial E}{\partial \mathbf{W}_l} = \mathbf{a}_{l-1}^T \boldsymbol{\delta}_l + \frac{\lambda}{N_w}\mathbf{W}_l \tag{2.6}$$

$$\frac{\partial E}{\partial \mathbf{b}_l} = \mathbf{1}^T \boldsymbol{\delta}_l \tag{2.7}$$

Notice that $\boldsymbol{\delta}_l$ can be cast as a derivative of the cost with respect to the signal input to layer l, $\partial E/\partial \mathbf{z}_l$. Therefore, one can consider $\boldsymbol{\delta}_l$ as being a measure of the error in layer l of the network. With this result, the back-propagation algorithm can be stated in its entirety.

2.3.2 The Back-propagation Algorithm

1. Set $\mathbf{a}_1 = \mathbf{X}$
2. Compute $\mathbf{z}_l = \mathbf{a}_{l-1}\mathbf{W}_l + \mathbf{b}_l$ and $\mathbf{a}_l = \psi(\mathbf{z}_l)$ for $l = 2, 3, ..., L$
3. Compute $\boldsymbol{\delta}_L = \frac{1}{N}\left[\boldsymbol{\eta}(\mathbf{X}) - \mathbf{Y}\right] \odot \psi'(\mathbf{z}_L)$
4. Compute $\boldsymbol{\delta}_l = \frac{1}{N}\boldsymbol{\delta}_{l+1}\mathbf{W}_{l+1}^T \odot \psi'(\mathbf{z}_l)$ for $l = L-1, L-2, ..., 2$
5. $\nabla E(\mathbf{W}, \mathbf{b})$ is given by $\frac{\partial E}{\partial \mathbf{W}_l} = \mathbf{a}_{l-1}^T \boldsymbol{\delta}_l + \frac{\lambda}{N_w}\mathbf{W}_l$, and $\frac{\partial E}{\partial \mathbf{b}_l} = \mathbf{1}^T \boldsymbol{\delta}_l$ for $l = L, L-1, ..., 2$

We can now use one of many gradient based optimization techniques to find the optimal parameters for the network. In all of the applications that follow, the Broyden - Fletcher - Goldfarb - Shanno (BFGS) algorithm is used to minimize the cost function. The BFGS method is a hill climbing technique that requires the Jacobian of the cost function (provided by the back-propagation algorithm). Given 0th and 1st order information, the algorithm implemented by the python library `scipy.optimize` also approximates the Hessian of the cost function. In practice BFGS converges significantly faster than batch gradient descent and hence is a favorable algorithm to use for the problems outlined in this study. Future work will include developing more complex algorithms to handle the minutiae of climate data.

2.4 ANNs in Practice

2.4.1 Example: Nonlinear Fitting

Consider the toy problem of fitting a one dimensional function. There are a multitude of existing techniques to solve this problem, the most common being linear regression. However, these models usually need a specific characterization of the function. Neural networks have the advantage of a fluid functional form. The architecture is the only information that needs to be fed into the system. There is an art in picking an optimal architecture, but many architectures are redundant and will model a training set equally well. In this example we sample 100 training and 100 testing points from the function

$$g(x) = \exp(-x^2)\sin(5\pi x) \tag{2.8}$$

for $x \in [-1, 1]$. This function has both changing amplitude and oscillatory structure, two important features that appear in many physical datasets. White noise is added to both sets of data to test the robustness of the network. The FFNN is set up with two hidden layers of 10, and 5 neurons respectively. See Figure 2.3 for a plot of the training process. Note that the generalization of the model is good, as there is little over-fitting of the training set. The resulting fit is shown in Figure 2.4. The model is an excellent fit for the testing data.

2.4.2 Example: The Lorentz Model

Next, we consider the dynamical problem of modeling differential equations. A *Lorentz system* is a set of nonlinear coupled differential equations

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} \sigma(y-x) \\ x(\rho - z) - y \\ xy - \beta z \end{pmatrix} \tag{2.9}$$

where $\dot{u} = du/dt$. The Lorentz system is characterized by having chaotic solutions and large sensitivities to initial conditions. The *Lorentz attractor* is a particular set of solutions, made famous by it's resemblance to a butterfly in 3-space (See Figure 2.5). In this example, the vertical dynamical component is replaced by a FFNN

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} \sigma(y-x) \\ x(\rho - z) - y \\ \mathcal{N} \end{pmatrix} \tag{2.10}$$

where $\mathcal{N} = \mathcal{N}(x, y, z, \dot{x}, \dot{y})$.

We first run the complete Lorentz system for a total time of 16 units. The system is integrated using a 4th order Runge-Kutta scheme with parameters $(\sigma, \rho, \beta) = (0, 8/3, 28)$ and an initialization of $(x_0, y_0, z_0) = (1.508870, -1.531271, 25.46091)$. These conditions result in a strange attractor solution. Figure 2.6 is a component plot of the solution. The multi-modal nature of the system is quite obvious, having an approximate period of five seconds. This feature is visualized in 3-space by the two winged structure of the attractor. After experimenting with different layer architectures, a single hidden layer with 50 neurons was found to have the best fit. Plots of the cost functions and the residual errors are shown in Figures 2.7 and 2.8. The costs have been minimized and the generalization error is relatively low. However, the network is clearly having trouble following the trajectory of the model. This first starts occurring approximately five seconds into the simulation, corresponding with a change in mode. The network continues the simulation assuming the model should still be in the first mode.

This is symptomatic of a larger problem concerning FFNNs. The network does not have any internal memory, meaning that it cannot assimilate temporal information. Which is paramount to understanding how the model oscillates between modes. If the number of neurons in the hidden layer are increased, the network tracks the model further. However, the network will always eventually lose the signal after a certain number of mode changes. In order to further improve statistical performance, a new type of architecture must be considered. *Recurrent neural networks* (RNNs) are a class of ANNs that have been shown to better constrain highly non-linear time series [Seidl and Lorenz, 1991]. RNNs have connections between neurons that are closed cycles. The architecture gains a notion of memory, where information from previous time steps can be fed back into the network to further constrain the model.

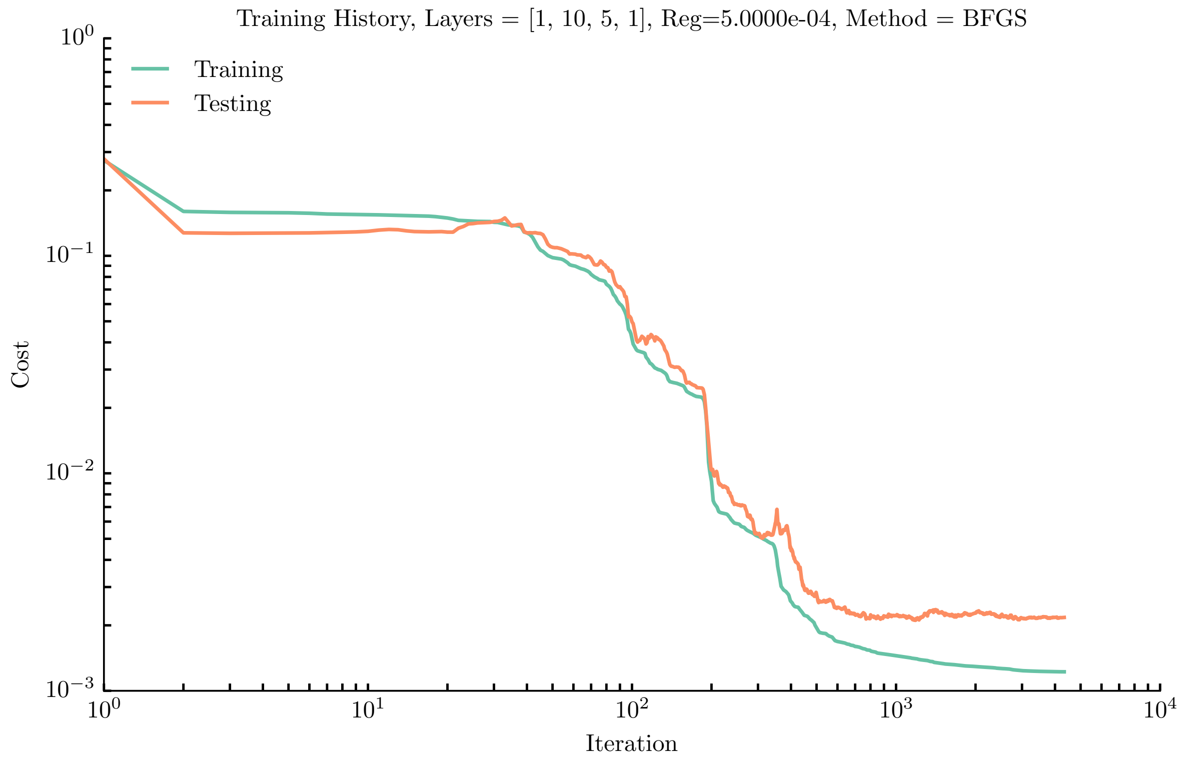

Figure 2.3: Plots of the cost function during the training of the FFNN. Note the minimal over-fitting illustrated by the testing cost curve.

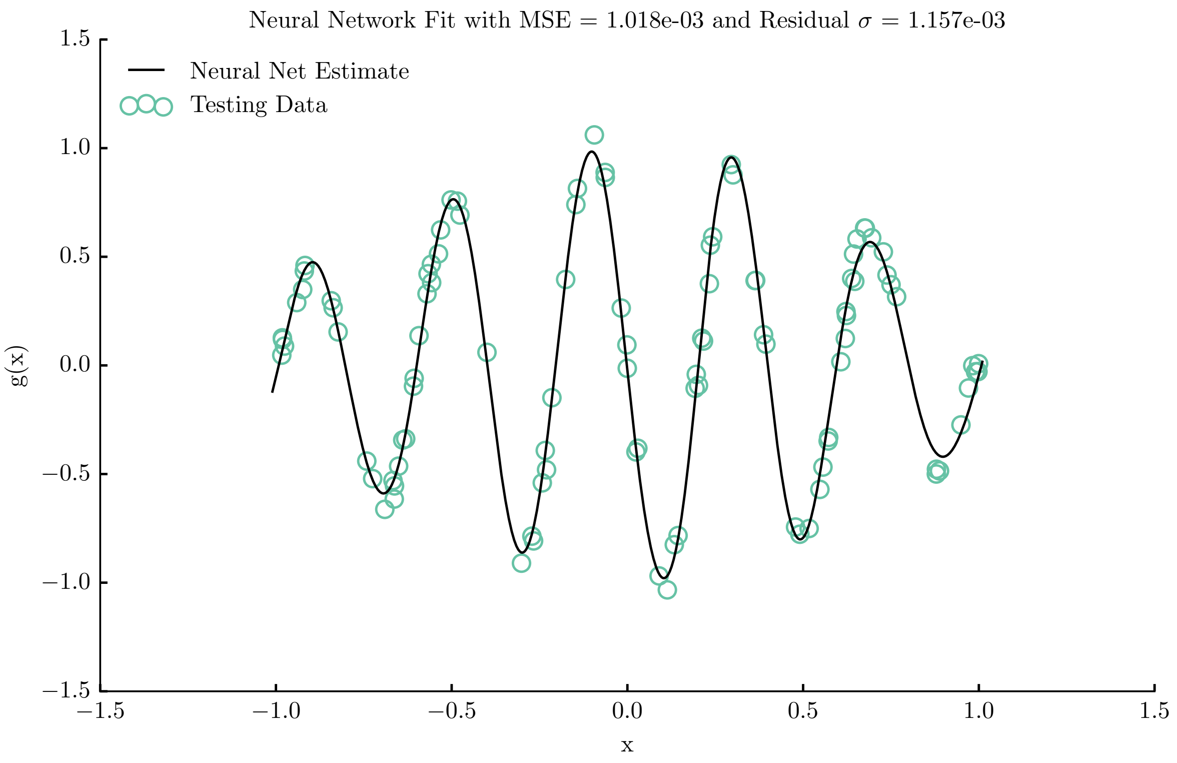

Figure 2.4: The neural network fit of equation 2.8.

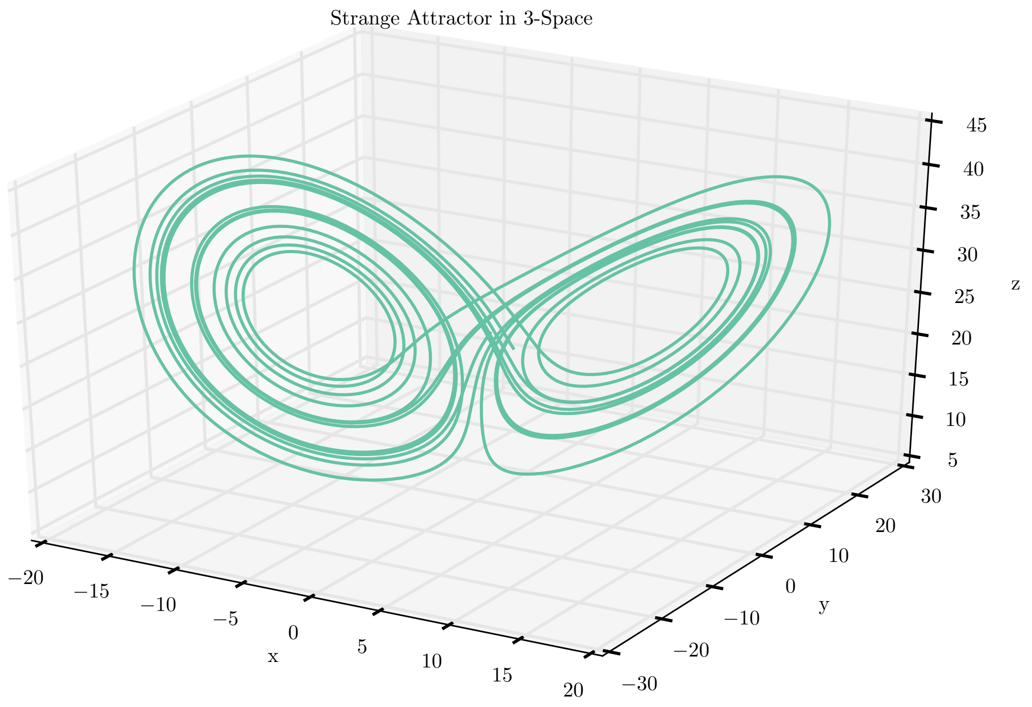

Figure 2.5: The strange attractor solution to the Lorentz system.

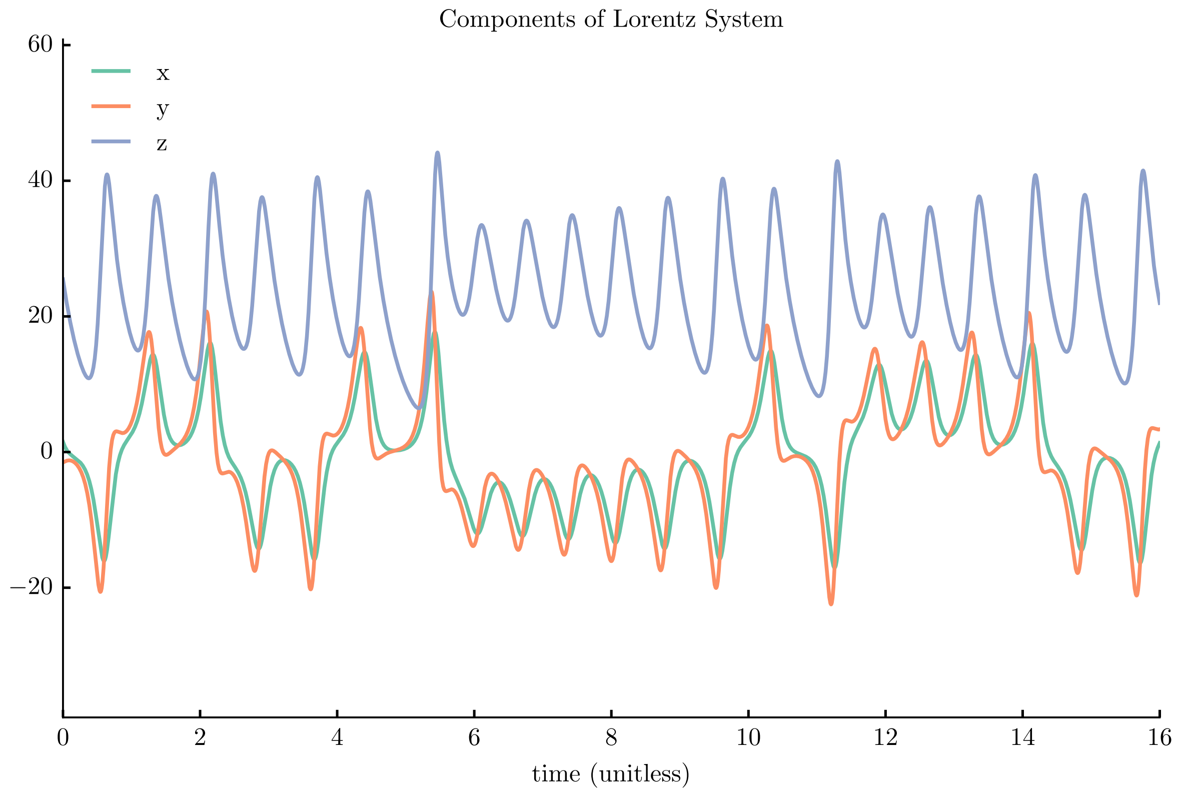

Figure 2.6: The individual components for the strange attractor solution. There is a unique multi-modal structure in which the solution persists between the two modes.

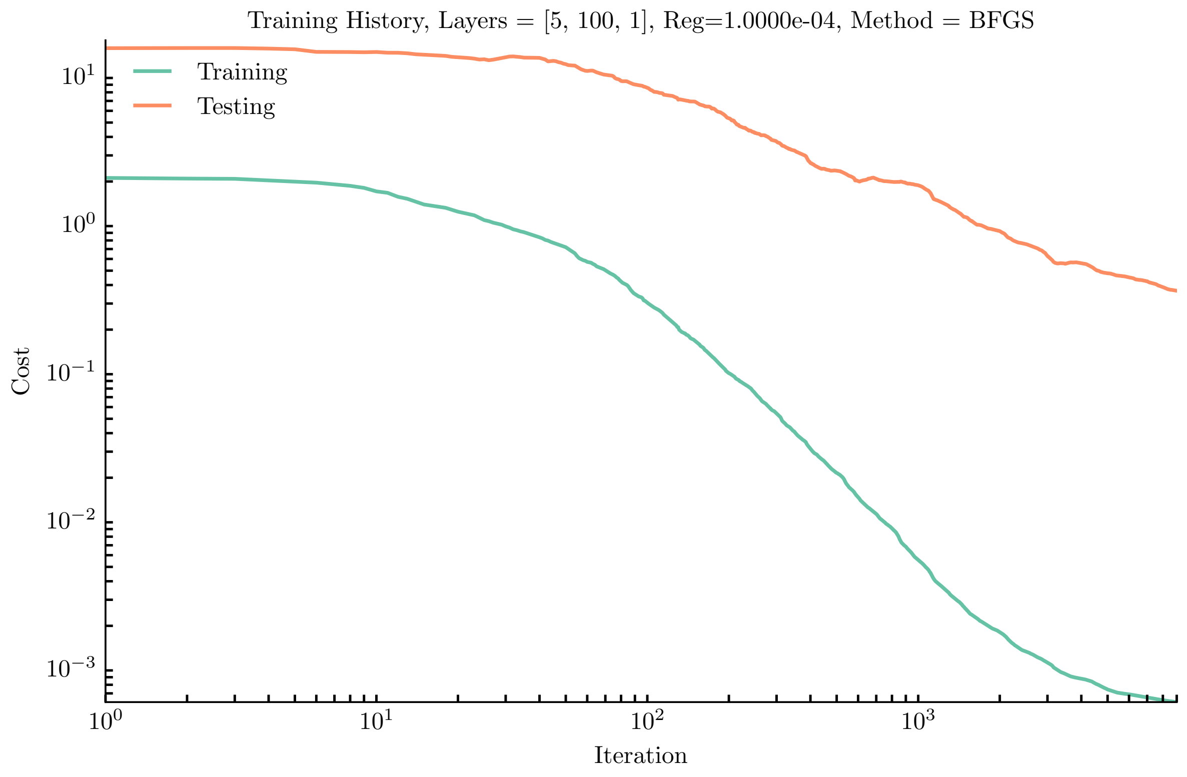

Figure 2.7: The cost function history for training the Lorentz network. The minimal over-error near the end is relatively negligible.

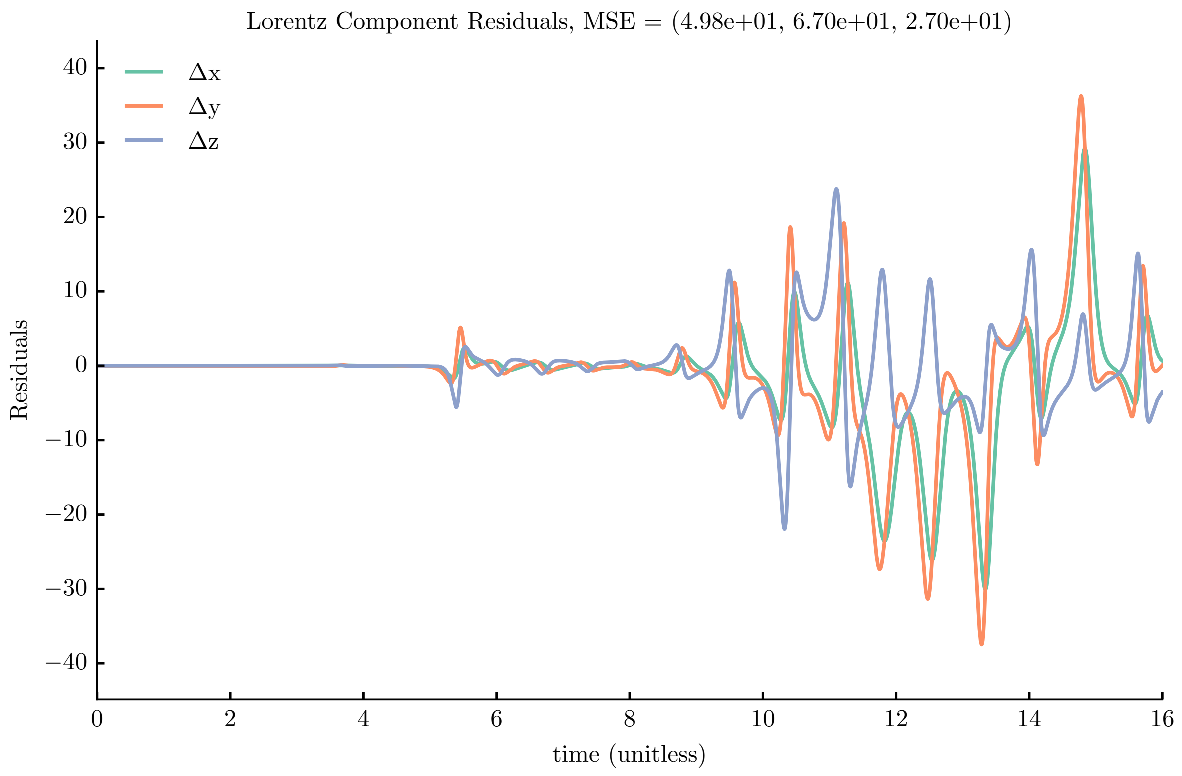

Figure 2.8: The residuals between the trained network and the true dynamical state of the Lorentz system. The network has a good fit until the mode changes, where it loses track of the solution and errors start to accumulate quickly.

Chapter 3

Modeling CO Fields with Neural Networks

3.1 Input Data

The GEOS-Chem CTM was used to generate CO and atmospheric data covering a span of two years from 2006 to 2008. The model has a resolution of 4 degrees by 5 degrees and 47 vertical levels from the surface to 0.01 hPa. However, we only use data from the middle and lower troposphere (up to level 29), which have been archived hourly. The attributes selected were: day of the year, surface CO emissions, CO concentrations, wind speeds, temperature, humidity, surface pressures, and planetary boundary layer (PBL) heights. Data was restricted to a two-gridbox (8 by 5 degree) area over the New York City region to isolate mainly anthropogenic sources of CO emissions. Each of the attributes were averaged over this area and the first 17 vertical levels to reduce the dimensionality of the problem. Each attribute was standardized by its mean and variance to remove large scale differences between the data streams.

3.2 Network Setup

The FFNN was setup to ingest a time-attribute array of data and output a time-target array of CO fields. After testing with multiple layer structures, a single hidden layer with eight neurons was selected as the best model. Including more neurons returned diminishingly smaller improvements to the model, suggesting that more preprocessing work is to be done to extract features and reduce the number of degrees of freedom for this problem. Many hidden layers also lead to similar negligible improvements, with significantly longer training times. L2 regularization was imposed on the network with a regularization coefficient of 1.00×10^{-3} to reduce possible over-fitting of the training data. The network was trained for data over 2006 and tested with data over 2007 to capture seasonal structures within the attributes.

3.3 Results

A cost analysis for the eight neuron architecture is shown in Figure 3.1. The trained network has a good level of generalization, as seen from trajectory of cost cost curves. This is due to the relatively low complexity of the parameter space (81 parameters). Fitting results are summarized in Figures 3.2 and 3.3. The residuals are normally distributed, indicating the FFNN captured most of the useful information and did not over-fit the testing set by assimilating the noise. The errors correlate to changes in season, but stay within approximately 12.7% of the mean CO field. Although these results indicate that smaller architectures have higher accuracy, larger networks are necessarily harder to train. Therefore, different training conditions may have to be imposed on different architectures to truly make meaningful comparisons between the different models. These networks were trained in serial on a personal computer, parallelization of the code and longer training times will likely lead to increases in prediction capabilities. All residuals were smoothed with windows of 7 days to show larger scale structure in the fit. The challenge still remains to fit the minute details in hourly variability. Figures 3.4 and 3.5 compare two different smoothing windows, one for 28 days and another for 1 day. The large window shows that seasonal variability is captured by the network and furthermore, that accuracy is shared between all the architectures. Note that the spread of the data is larger for the smaller window, illuminating the problem of over-fitting. Large networks need to be trained more for the same level of generalization, and In this case all three networks were trained in similar settings. Therefore, we notice a higher magnitude of noise in the fit as the number of weights increases. The raw fit without smoothing is plotted in Figure 3.6. There is a length of time during the summer months where the network has particular trouble fitting the small scale structure of the field. This has been a common occurrence throughout the different network architectures and will be a focus of future work on this project.

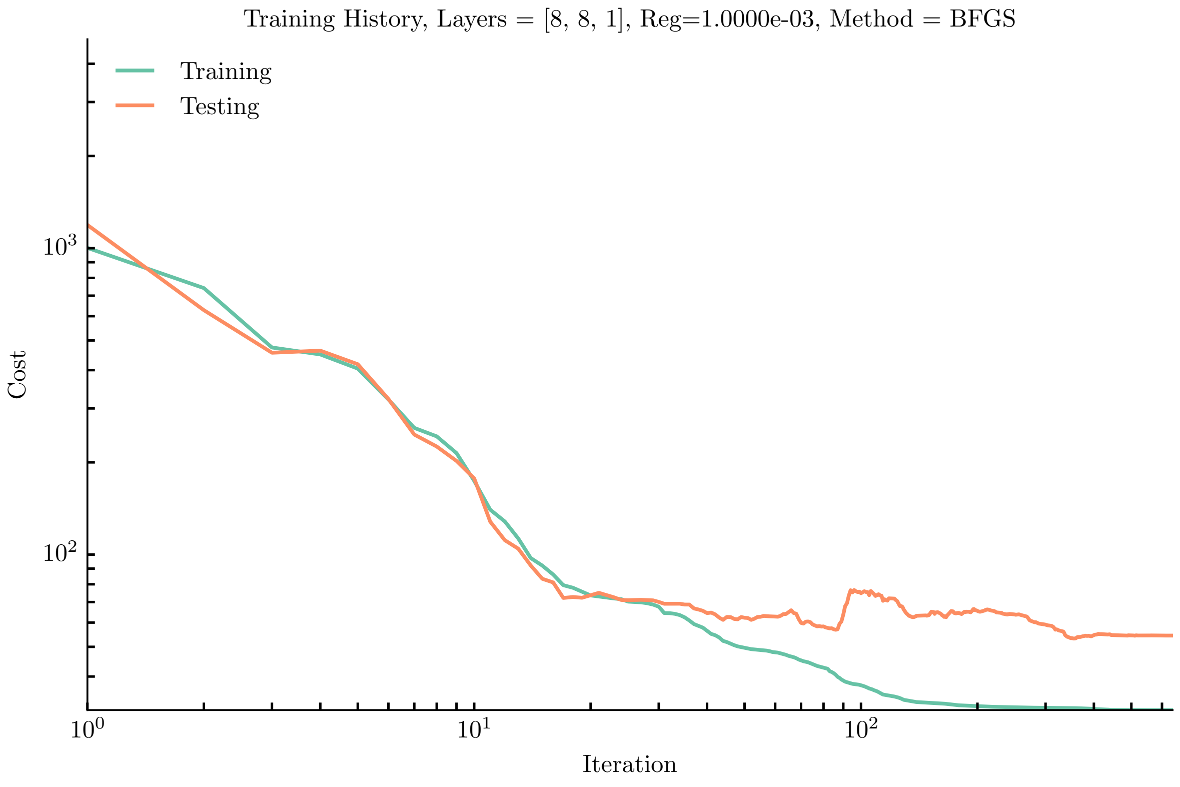

Figure 3.1: The cost function history for training a FFNN on the GEOS-Chem CTM data.

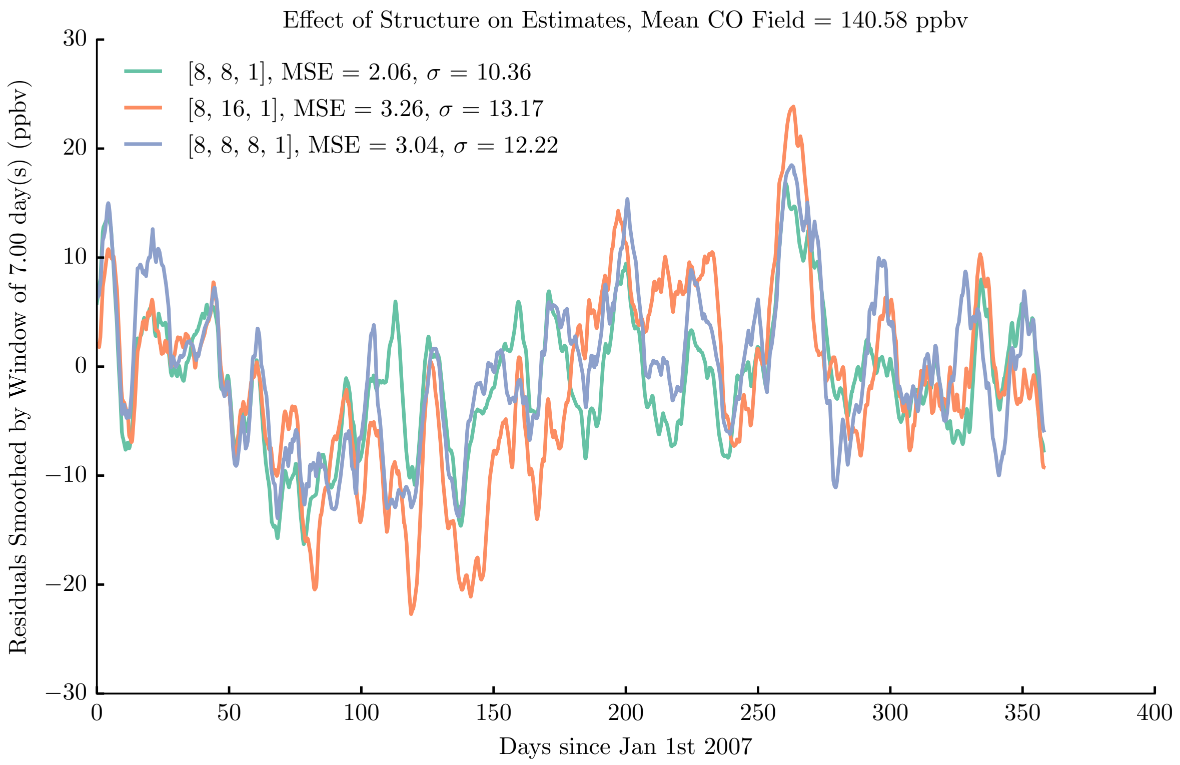

Figure 3.2: A plot of residual data between the GEOS model and multiple neural network fits. Larger architectures do not necessarily provide proportionate improvements to the model.

3.4 Conclusions and Future Work

The statistical approach taken in these experiments have proved to be an advantageous way of modeling CO concentrations. Accurate models were constructed given short training times, small architectures, and little preprocessing of data. It is clear that given more computational resources, neural networks will be a valuable tool to use alongside modern data assimilation techniques. The next problem is one of scale. The data set is currently one dimensional in each attribute. The total GEOS-Chem CTM dataset is multivariate and of very high dimension. In order to extend this analysis to whole map modeling, efficient methods of dimension reduction must be investigated. N-way methods of tensor factorization will be considered to reduce the number of dimensions but also to retain correlation information in both spatial and temporal domains. There has been recent interest to apply convolutional neural networks to high dimensional spatial data and to use recurrent neural networks for modeling time series data. It would seem plausible then, to combine the two architectures to attack problems with both spatial and temporal correlations. Lastly, more efficient ways of training will be considered to deal with larger sets of data, as training on one year may not be enough to quantity large scale temporal activity in the data. Weight dropout, simulated annealing, and genetic algorithms will all be considered to help reach global minima in spaces that not convex and of very large size.

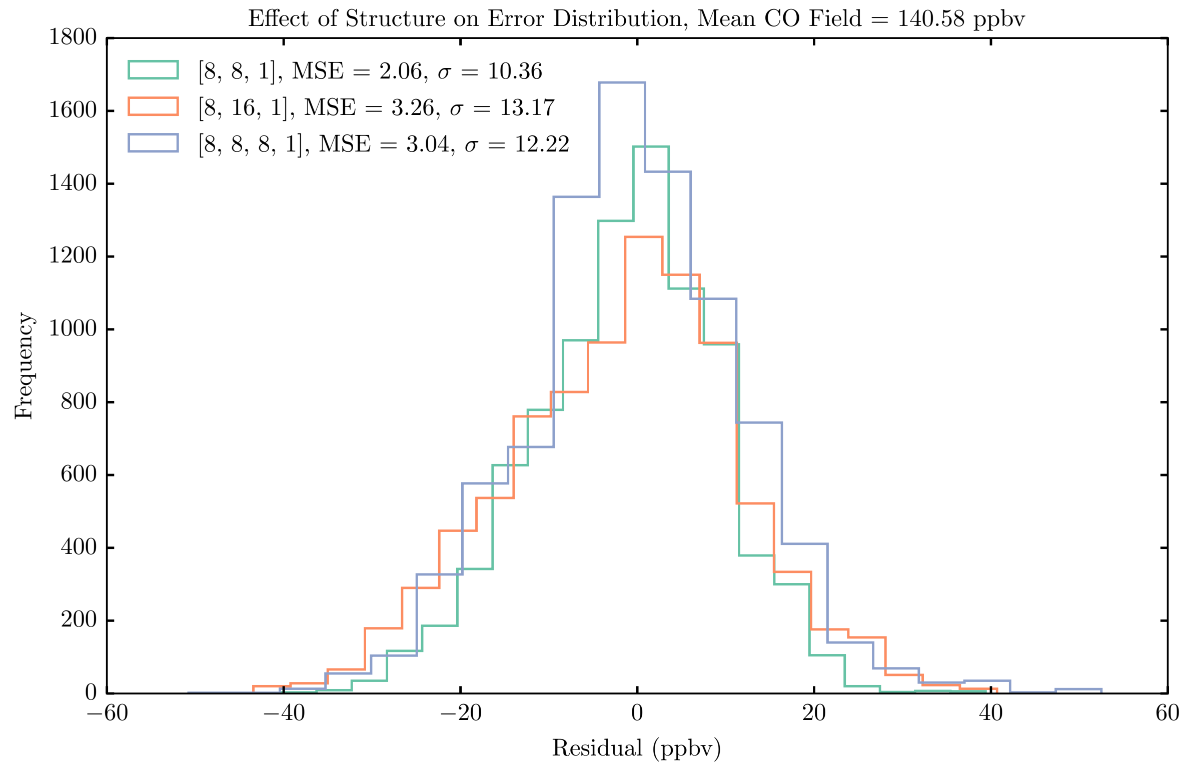

Figure 3.3: A histogram of residuals data between the GEOS model and multiple neural network fits.

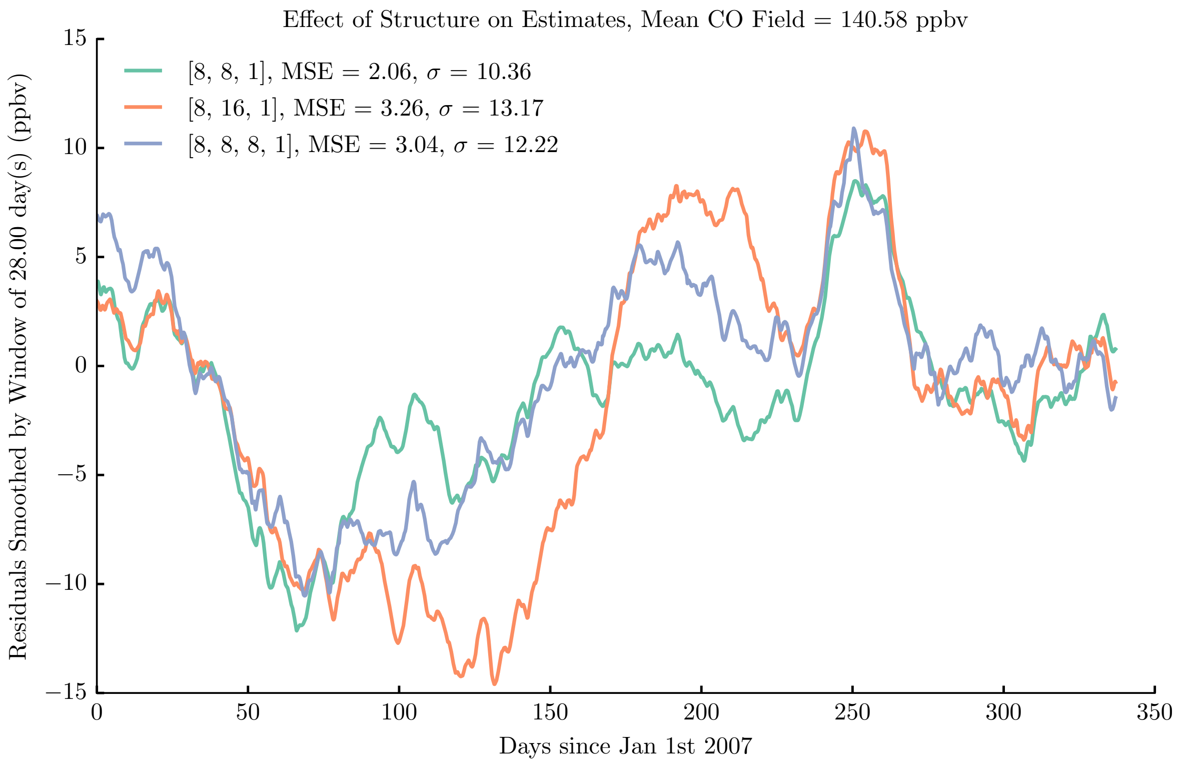

Figure 3.4: Residuals between the GEOS-Chem CTM data and network outputs. A large smoothing window shows that all three networks capture seasonal changes without major discrepancies.

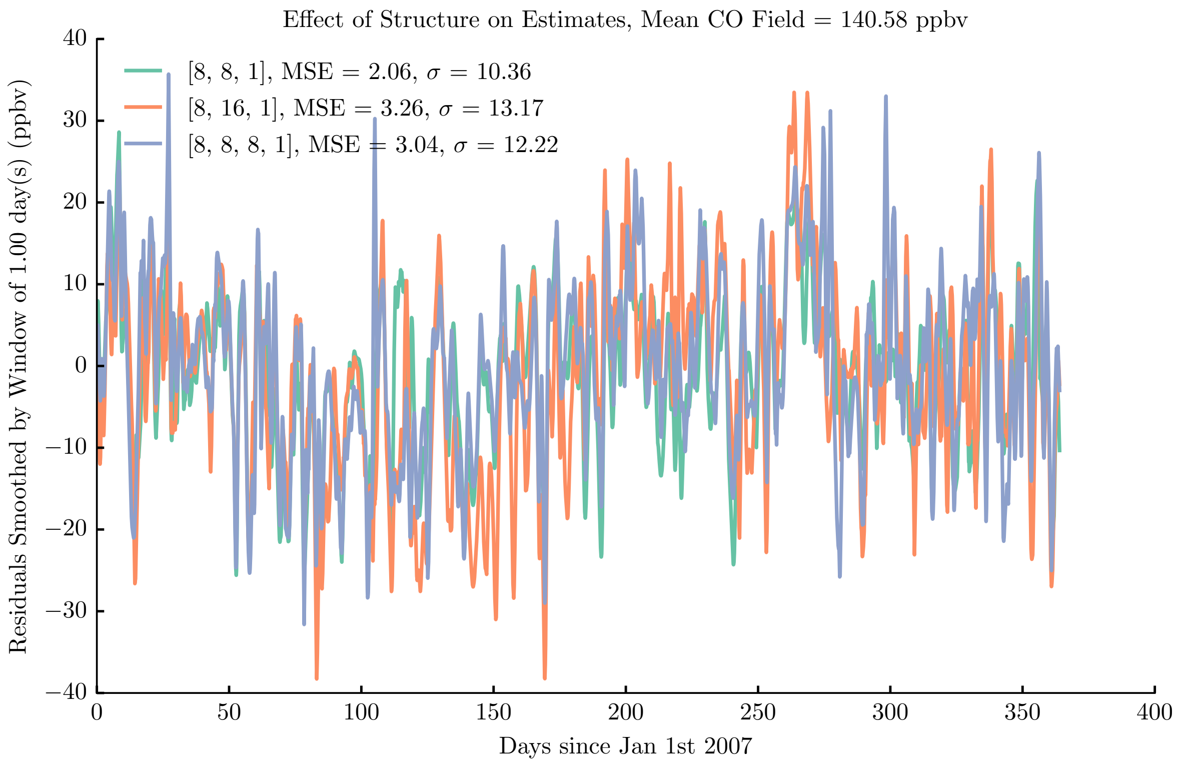

Figure 3.5: Residuals between the GEOS-Chem CTM data and network outputs. A short smoothing window allows daily variability to leak into the data and illuminate the problem of over-fitting.

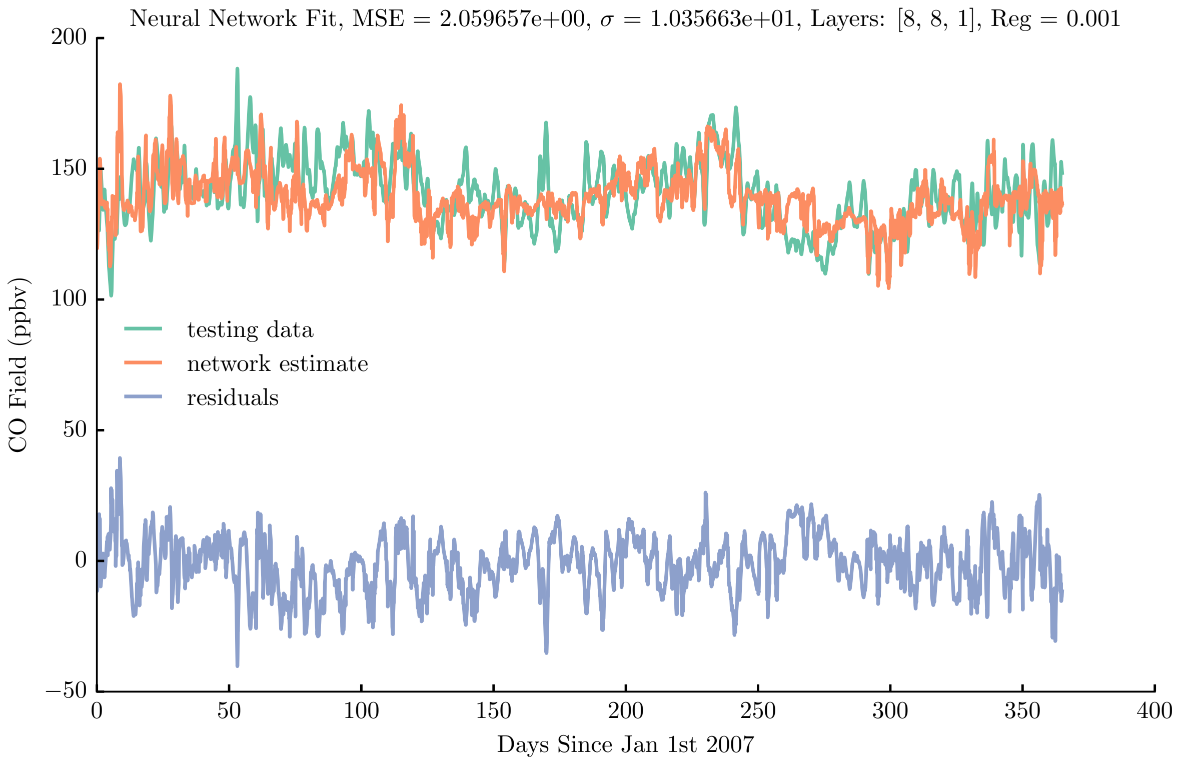

Figure 3.6: The raw fit of the testing data. Note the errors during the summer season.

Bibliography

[Bian et al., 2007] Bian, H., Chin, M., Kawa, S., Duncan, B., Arellano, A., and Kasibhatla, P. (2007). Sensitivity of global co simulations to uncertainties in biomass burning sources. *Journal of Geophysical Research: Atmospheres*, 112(D23).

[Duncan et al., 2007] Duncan, B., Logan, J., Bey, I., Megretskaia, I., Yantosca, R., Novelli, P., Jones, N. B., and Rinsland, C. (2007). Global budget of co, 1988–1997: Source estimates and validation with a global model. *Journal of Geophysical Research: Atmospheres*, 112(D22).

[Haan et al., 1996] Haan, D., Martinerie, P., and Raynaud, D. (1996). Ice core data of atmospheric carbon monoxide over antarctica and greenland during the last 200 years. *Geophysical research letters*, 23(17):2235–2238.

[Jiang et al., 2011] Jiang, Z., Jones, D., Kopacz, M., Liu, J., Henze, D. K., and Heald, C. (2011). Quantifying the impact of model errors on top-down estimates of carbon monoxide emissions using satellite observations. *Journal of Geophysical Research: Atmospheres*, 116(D15).

[Jones et al., 2009] Jones, D., Bowman, K., Logan, J., Heald, C., Liu, J., Luo, M., Worden, J., and Drummond, J. (2009). The zonal structure of tropical o 3 and co as observed by the tropospheric emission spectrometer in november 2004–part 1: Inverse modeling of co emissions. *Atmospheric Chemistry and Physics*, 9(11):3547–3562.

[Liu et al., 2010] Liu, J., Logan, J. A., Jones, D., Livesey, N., Megretskaia, I., Carouge, C., and Nedelec, P. (2010). Analysis of co in the tropical troposphere using aura satellite data and the geos-chem model: insights into transport characteristics of the geos meteorological products. *Atmos. Chem. Phys*, 10(24):12–207.

[Ott et al., 2011] Ott, L., Pawson, S., and Bacmeister, J. (2011). An analysis of the impact of convective parameter sensitivity on simulated global atmospheric co distributions. *Journal of Geophysical Research: Atmospheres*, 116(D21).

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[Seidl and Lorenz, 1991] Seidl, D. R. and Lorenz, R. D. (1991). A structure by which a recurrent neural network can approximate a nonlinear dynamic system. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, pages 709–714. IEEE.

[Shindell et al., 2006] Shindell, D., Faluvegi, G., Stevenson, D., Krol, M., Emmons, L., Lamarque, J.-F., Petron, G., Dentener, F., Ellingsen, K., Schultz, M., et al. (2006). Multimodel simulations of carbon monoxide: Comparison with observations and projected near-future changes. *Journal of Geophysical Research: Atmospheres*, 111(D19).

[Streets et al., 2003] Streets, D. G., Bond, T., Carmichael, G., Fernandes, S., Fu, Q., He, D., Klimont, Z., Nelson, S., Tsai, N., Wang, M. Q., et al. (2003). An inventory of gaseous and primary aerosol emissions in asia in the year 2000. *Journal of Geophysical Research: Atmospheres*, 108(D21).

[Van der Werf et al., 2010] Van der Werf, G. R., Randerson, J. T., Giglio, L., Collatz, G., Mu, M., Kasibhatla, P. S., Morton, D. C., DeFries, R., Jin, Y. v., and van Leeuwen, T. T. (2010). Global fire emissions and the contribution of deforestation, savanna, forest, agricultural, and peat fires (1997–2009). *Atmospheric Chemistry and Physics*, 10(23):11707–11735.