USING ARTIFICIAL NEURAL NETWORK (ANN) TECHNIQUES FOR SOLAR IRRADIATION PREDICTIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EROL CAN AKBABA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
PHYSICS

SEPTEMBER 2019

Approval of the thesis:

USING ARTIFICIAL NEURAL NETWORK (ANN) TECHNIQUES FOR SOLAR IRRADIATION PREDICTIONS

submitted by **EROL CAN AKBABA** in partial fulfillment of the requirements for the degree of **Master of Science in Physics Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Altuğ Özpineci _____
Head of Department, **Physics**

Prof. Dr. Bülent G. Akınoğlu _____
Supervisor, **Physics, METU**

Assist. Prof. Dr. Emre Yüce _____
Co-supervisor, **Physics, METU**

Examining Committee Members:

Prof. Dr. Mehmet Parlak
Physics, METU _____

Prof. Dr. Bülent G. Akınoğlu
Physics, METU _____

Dr. Talat Özden
Faculty of Engineering, Gümüşhane University _____

_____

_____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: EROL CAN AKBABA

Signature :

ABSTRACT

USING ARTIFICIAL NEURAL NETWORK (ANN) TECHNIQUES FOR SOLAR IRRADIATION PREDICTIONS

Akbaba, Erol Can

M.S., Department of Physics

Supervisor : Prof. Dr. Bülent G. Akınoğlu

Co-Supervisor : Assist. Prof. Dr. Emre Yüce

September 2019, 34 pages

Estimation of solar energy is a task with many benefits to a diverse group of people. This purpose is pursued with many different methodologies. Artificial Neural Networks (ANNs) are the novel methods of choice in the last decade. We compare the classical solar irradiation estimation methods with different ANN schemes including different inputs, data amount and estimation target. Our analyses show that the use of ANN to predict solar irradiation reaching the Earth's surface gives similar results with that of the classical regression approaches. The small difference between these two approaches lies within the instrumentation accuracy of the measuring devices.

Keywords: solar energy, estimation, machine learning, deep learning, artificial neural network

ÖZ

YAPAY NÖRON AĞLARI (YNA) İLE GÜNEŞ RADYASYONU TAHMİNLENMESİ

Akbaba, Erol Can
Yüksek Lisans, Fizik Bölümü
Tez Yöneticisi : Prof. Dr. Bülent G. Akınoğlu
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Emre Yüce

Eylül 2019 , 34 sayfa

Güneş enerjisi tahmini bir çok farklı gruba fayda sağlayan bir bilgi. Bu amaçla bir çok analiz ve model hali hazırda geliştirilmiş bulunmakta. Güneş enerjisi tahminlerinde son yıllarda yükselişe geçen Yapay Nöron Ağları (YNA) ile klasik yöntemleri karşılaştırıyoruz. Farklı girdiler, veri miktarı ve hesap hedefleri üzerinden yaptığımız karşılaştırmalarda YNA'nın klasik yöntemlerle benzer isabet gösterdiği sonucuna vardık. İki yöntem arasındaki küçük miktarda bulunan isabet farkının enstrümentasyon hatası limitleri içinde kaldığı sonucuna vardık.

Anahtar Kelimeler: güneş enerjisi, makine öğrenimi, derin öğrenme, nöron ağı

Dedicated to the exalted pursuit of knowledge.

ACKNOWLEDGMENTS

I would like to thank;
Doğuşcan Ahiboz for many stimulating conversations,
all my friends for their cheers,
and my advisor for his infinite patience.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Estimation of daily total energy received per unit area is a task that can help in many areas including allocating future solar panel capacity, filling missing data, facilitating calculation of solar energy in regions without advanced measurement stations and more.

To facilitate this calculation many models exist in the literature, with varying degrees of accuracy in their estimation. These models, while adequate, can always be improved upon and offer great potential benefits to the growing solar energy industry and all related research disciplines if potential improvements are realized. As the machine learning approach is having a resurgence over the last decade its potential may yet prove useful in the problem of daily solar energy estimation.

Core of the estimation problem consists of obtaining a functional relationship between values like bright sunshine fraction (i.e the ratio of bright sunshine duration to total daylight time), daily average (or minimum, maximum) temperature or any other measured quantity about weather and total daily energy received on a surface. An important part of the problem is determining which subset of often used climatology variables (like temperature, latitude etc.) are required for most accurate estimations. The final concern is the accuracy pertaining to measured data when comparing performance differences in models.

As a consequence, the aim of the thesis is to estimate daily solar irradiation using machine learning and comparing these estimations with a conventional model. It

is also aimed to clarify if a machine learning algorithm produces better estimation schemes compared to the conventional model in connection with the unavoidable errors in the measurements.

1.2 Proposed Methods and Models

Many models are proposed in the literature, establishing functional relationships between daily solar energy and a long list of variables, chief among them being bright sunshine fraction (i.e the ratio of bright sunshine duration to total daylight time), and latitude. These input variables are often accompanied by some subset of other measured quantities about the climate such as average temperature during day, maximum or minimum temperatures, relative humidity, wind speed and direction and many more. These functional relationships often come in the form of a sum of elementary functions of the input variables (usually polynomials). Multiplicative relations between the real numbered powers of the inputs is also a model that is considered. Examples of these and more can be found in the works of Kumar et al. [1] and Özgören et al. [2] which also compare them to neural networks with a single hidden layer.

The methods utilized in obtaining numerical values for the parameters of these models vary as well. Most common methods are linear and non-linear regressions to local data of solar irradiation and other climatic parameters. A universal approach conducted by Akınoğlu and Ecevit [3]] ties local models together to generate estimates that are valid globally.

Machine learning in the form of neural networks are rivaling human performance in some areas and the core reason is the increased feasibility of deep neural networks which have many hidden layers. This is a result of an increasing amount of data being collected over the last decades, improvements in computing hardware and development of more efficient algorithms and implementations.

1.3 Contributions and Novelties

The contributions and novelties of the thesis are itemized in the following:

- Investigating the role of having more layers within neural network model to estimation accuracy of daily solar energy;
- Obtaining a quantitative estimate for the performance difference of more versus less data in training a neural network;
- Facilitating a comparison between machine learning based estimation and conventional approaches regarding accuracy
- Comparing the differences between models in light of measurement uncertainty

1.4 The Outline of the Thesis

Chapter 2 summarizes the conventional methods and results. Chapter 3 contains an explanation of how machine learning with deep neural networks operates. It starts with a general overview of machine learning and then continues with a more detailed take on the core parts of machine learning. Specifically the structure of a neural network and the gradient descent optimization algorithm is explained in some detail. Final section notes the reasons behind expecting deep neural networks to perform better.

Chapter 4 examines the data and its use. Which climatic variables were available, how they have been grouped for purposes of training, validation and set as well as identification of the subset which results in the most accurate estimate. It then continues with outlining the various choices made regarding the multi layer perceptron model and model's training with gradient descent. These are followed by the performance of trained networks, in comparison with classical methods.

Chapter 5 concludes the main points regarding feasibility of machine learning for solar energy estimation task. A broader view of both machine learning and solar energy is considered for the purposes of future potential.

CHAPTER 2

CONVENTIONAL APPROACHES

Scientific interest in solar energy is not a new phenomenon. As such there are many different models to estimate and to forecast. In this chapter we aim to provide a short summary for the estimation of (daily) solar energy incident on a surface.

2.1 Meteorological Variables

In review of empirical models Besharat et al [4] utilize four categories of meteorological variables that are used to estimate incoming solar energy. These are sunshine based, cloud based, temperature based and other, which may combine several variables from previous three categories.

Sunshine based estimations have their start at Angstrom-Prescott relation;

$$\frac{H}{H_0} = a + b(\frac{n}{N}) \tag{2.1}$$

where H is the daily solar radiation on a horizontal surface, H_0 is the extraterrestrial radiation at the periphery of Earth's atmosphere, n is bright sunshine duration in hours and S_0 is the total daylight hours. a and b are the empirical parameters to be determined from data. This equation implicitly depends on latitude through H_0 and N, which are calculated. n is determined from measurements using sunshine recorders.

A great number of researchers have utilized this equation with the local data they have available to make estimations. Besharat et al cite dozens of models obtained from the

premise of the Angstrom-Prescott equation. Geographical scope of the models range from being specific for a single meteorological station or city to encompassing an entire nation.

Akınoğlu and Ecevit [3] combines one hundred local coefficients of the linear Angstrom-Prescott relation over the globe to obtain a quadratic relation, presented below:

$$\frac{H}{H_0} = 0.145 + 0.845\frac{n}{N} - 0.280(\frac{n}{N})^2. \tag{2.2}$$

In cloud based estimations satellite or ground observation of cloud cover is utilized in making estimations about solar energy. Proposed empirical forms include polynomials (up to degree 3) and exponential relations.

Temperature based models have more diversity in mathematical form as well as independent variables. Inputs for these are usually minimum and maximum temperature at the location for a given day, with the average also included in some models.

Other models include various climatological measurements like relative humidity, mean precipitation and more. In general, models with diverse set of measurements were more often utilized in large scale (up to half of a continent's area or more) models.

In their case study Besharat et al. list best performing model for their location. With respect to variables sunshine and temperature based models outperform other considerations, with "other" category model a close third with variables of daily sunshine fraction, maximum and minimum temperature of day.

In an examination of sunshine based models Yorukoğlu [5] reports that out of the five models considered (linear, quadratic, cubic, logarithmic and exponential) the cubic polynomial is the most accurate. However, quadratic and linear models are also close in accuracy. In final analysis Yorukoğlu concludes that the linear Angstrom-Prescott model has similar accuracy and only half the parameters.

In review of artificial neural network (ANN) performance in solar radiation estimation Yadav [6] reports a number of models with different inputs considered. These inputs are more varied and numereous than previous empirical models. Yadav reports that

ANNs that produce best results in their analysis include the inputs of temperature, relative humidity, atmospheric pressure and wind speed after sunshine duration. Their work considers neural networks with a single hidden layer and two hidden layers.

Shamshirband et al [7] examine the usefulness of extreme learning machines (ELM) for solar energy estimation task. ELMs are neural networks that utilize radial basis functions introduced by Huang et al [8]. Shamshirband et al study uses ELMs with one and two hidden layers.

In general, previous studies either utilize a proposed functional form which has its parameters determined by regression or neural networks with up to two hidden layers.

CHAPTER 3

WHAT IS MACHINE LEARNING?

3.1 Introduction

Machine Learning (ML) is a cluster of algorithms that are used to obtain models from data. Current state of the art consists of artificial neural networks (or just neural networks) that can learn useful representations from data or environment interaction. Machine learning is not a new methodology, having gone through a boom and bust cycle[9]. Current resurgence is attributed to more data being available, more efficient algorithms and hardware improvements.

Machine learning consists of various different approaches that can be quite distinct from each other. Constructing a walker with a few sticks and joints then evolving them according to a fitness criteria (e.g distance covered) is under the umbrella of machine learning, as well as video recommendation engines and spam detectors.

There are two main components to a machine learning application. First is the model and second is the optimization procedure to optimize the model by changing the parameters based on data. Our model is a neural network (specifically a multi layer perceptron) while our optimization procedure is a variant of gradient descent called AdaGrad[10].

Our type of problem is called a regression problem as opposed to a classification problem. In instances where we're aiming to approximate a continuous function it's called regression, an example of which is estimation of housing prices. If discrete labels or categories are output then it is termed classification which would be exemplified by email spam filters.

There are also two different kinds of learning. One is where "true" results are available (as is in our case or spam filters) which is called "supervised learning" and where such data is not available (like video recommendation) which is called "unsupervised learning".

As for the optimization procedure, backpropagation and variants of gradient descent are the current staples of training a neural network. Gradient descent can be understood via an analogy to gravity. Over a surface (called loss surface) a point particle is dropped under gravity. As such the particle will move in the direction of lower elevation, eventually settling to a (often local) minima if there is one. The surface is an analogy to how similar obtained model is to what the data represents, whereas gradient descent is analogous to gravity. On the surface higher points are where the model and data diverge, lower points are where they agree. The main task of machine learning is to find the lowest possible valley. Of course the dimensionality of neural networks is significantly greater than 3, but the analogy still remains valid.

To wit, this work is about supervised training of an artificial neural network to obtain a model to estimate daily solar irradiation with different input variables using the aforementioned AdaGrad as the optimization algorithm.

3.2 Neural Networks

Neural networks are originally inspired from the structure of our brains, with neurons connecting to each other and sending signals if neuron is "active". Software equivalent of this structure is expressed through a "graph" (as in graph theory, mathematics of pairwise connected objects). An example graph is provided in figure 3.1.

In figure 3.2 you can see an example of a neural network with 2 hidden layers. Each circle is a "node" corresponding to a neuron. Connections are made between one group of neurons (layer 1 neurons) to another group (layer 2 neurons). This structure of layered units, with inputs connecting to inner neurons which in turn are connected to the output node is the most basic structure of a neural network with a single hidden layer. These inner neurons are called "hidden" because they don't interact with input or output data directly. Any neural network with more than 1 hidden layer is called a

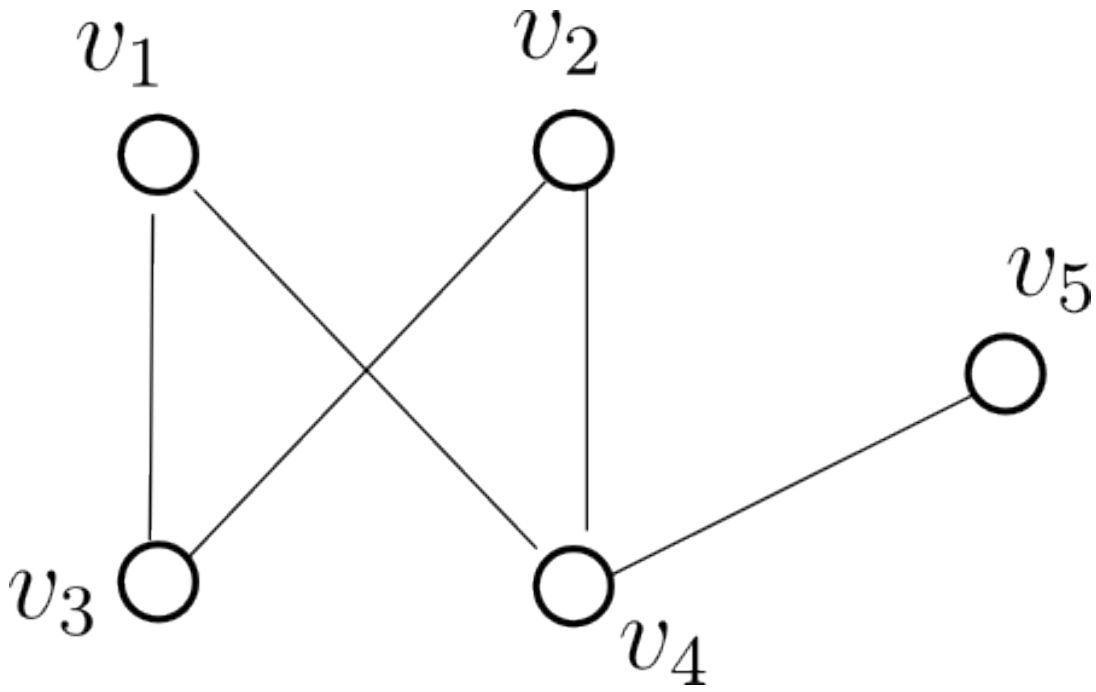

Figure 3.1: © Taiyaki1228 (https://commons.wikimedia.org/wiki/File:Wikibooks_graph_theory.png), Wikibooks graph theory, https://creativecommons.org/licenses/by-sa/3.0/legalcode

An example graph, with 5 nodes labeled with v_i. In a mathematical contexts the name given to nodes is vertex, the connections are called edges. In machine learning we will call them neurons and weights, respectively. Also, we will not have connections like the one between v_1 and v_3 in a neural network since a connection like that prohibits formation of layered neurons.

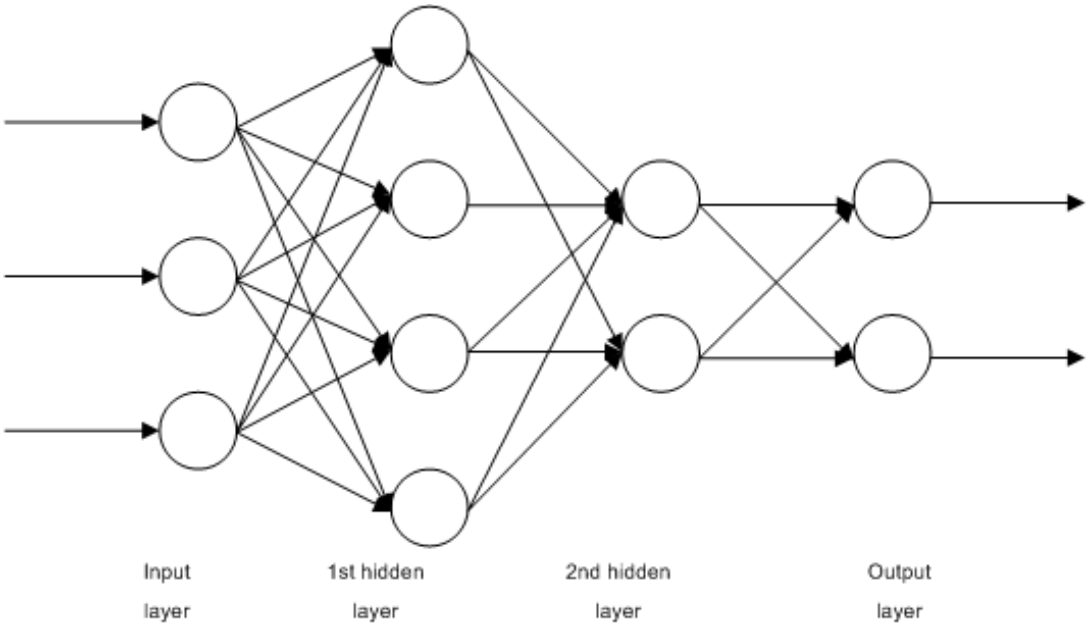

Figure 3.2: © John Salatas (https://commons.wikimedia.org/wiki/File:Multilayer_Neural_Network.png), Multilayer Neural Network, https://creativecommons.org/licenses/by-sa/3.0/legalcode
A neural network consisting of 2 hidden layers. It has 3 inputs and 2 outputs. Unlike the general graph, this structure has clear layers which can be deduced from the lack of connections between neurons in the same layer.

deep neural network.

Similar to a neuron activating, our mathematical model needs a gating function to characterize activity of a node. Most historically appropriate function for this purpose is called the sigmoid function while the most recent and state of the art choice is rectified linear unit (relu). Another option for a neuron's activation is hyperbolic tangent. These are shown in fig 3.3.

The capacity of neural networks to approximate any continuous function is established via the universal approximation theorem. Chapter 4 of Michael Nielsen's book [11] demonstrates a graphical approach to how neural networks can approximate any continuous function. There are also analytical proofs from Cybenko [12] and Hornik et al. [13].

As an example, for a single layer neural network with 5 inputs, 10 hidden neurons and 1 output the entire model under consideration is as follows:

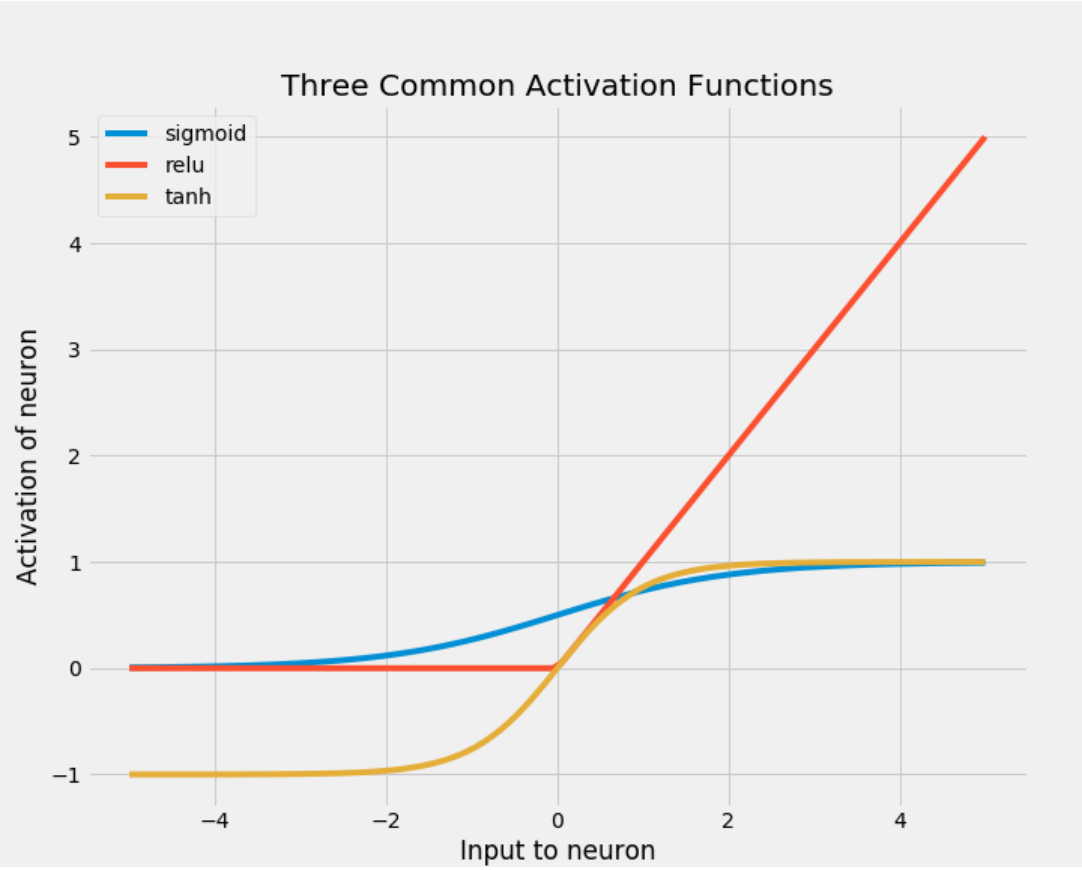

Figure 3.3: Activation of a single neuron as a function of its input.

$$N = \sum W_{2i} a_i + b_2 \tag{3.1a}$$

$$a_i = \sigma(\sum W_{1j} X_j + b_1) \tag{3.1b}$$

where W_1 and W_2 are connection weights, and b_1, b_2 are the bias terms. W_1 is for the connections from input to hidden layer and W_2 for the connections from hidden layer to output. As we have 5 inputs and 10 hidden neurons, shape of W_1 is (5, 10) as a matrix. Shape of W_2 is (10, 1) so in total this network contains 62 parameters, including two bias terms.

Neural networks are initialized with random weights, so their output is completely random in the first run. The only meaning of the formulation comes through universal approximation theorem. Once a neural network learns from data, the resulting model is encoded in the activity of the nodes. This means neural networks are black boxes by default. Their universality, however, makes them worthwhile.

3.3 Gradient Descent

Once we have a neural network with randomly initialized weights, we are supposed to optimize the weights to obtain a model that matches given data.

Optimization is a process aimed at a goal. In the case of machine learning that goal is called loss function, or objective function. This function is a measure of how close our approximation is to the data. There are various different choices that can be made, with increasing complexity and nuance as the problem domain becomes harder for computers. In the process of optimization we aim to reduce loss as close to 0 as possible. Here it will suffice to demonstrate one of the most common loss functions, the squared error:

$$L = \frac{1}{2M} \sum_m (N^m - y^m)^2 \tag{3.2}$$

where y^m is the target value found in data for $m - th$ element of the data set, N^m is corresponding output from neural network, M is the total number of data points

in the data set used to calculate L. This loss value acts as a guide to how well our model is doing in the process of optimization. As the loss decreases our model starts to resemble the data more accurately.

Gradient descent is an optimization algorithm that relies on derivatives of the loss function with respect to the parameters of our model. If we label the parameters of our model as W_{ij} basic structure of the gradient descent algorithm looks like this:

Initialize eqn. 3.1 with random W_{ij}
while *loss* > *desired loss* **do**
  Compute derivative of L wrt parameters, $\frac{\partial L}{\partial W_{ij}}$
  Change the values of W_{ij} to $W_{ij} - l\frac{\partial L}{\partial W_{ij}}$
end

Algorithm 1: Basic structure of gradient descent

Here l is called the learning rate. It controls how far we move in the direction opposite to the gradient in a single update step(gradient itself points in the direction of increasing L with respect to the parameters of our model). There are practical considerations in choosing l, two obvious cases being too large or too small values for l. In the former case this algorithm never converges and in the latter case it always converges but time to completion may be too long. Usually there's a suitable intermediate value between large and small l. Common practice is to reduce the learning rate as training goes on, either automatically by the algorithm or by manually arranging its value.

Learning rate is the first example of a class of variables called "hyperparameters". The constants used in optimization algorithm, number of layers, number of neurons in the layers and many more belong to this category. In general any parameter that requires tuning and isn't a weight in our neural network will be a hyperparameter. They can be assigned manually, or chosen through a "grid search" of several potential values. The method we use, namely hyperband, is explained later.

There are various different modifications to the basic setup of gradient descent. These include adding a momentum term [14], adaptive methods that scale learning rate automatically [15] and each weight specifically [16]. All of them share the same underlying idea as the basic variant.

Backpropagation is the technique of moving the gradient of the loss function from output nodes towards input nodes. Since error of any node by itself doesn't make sense except for the output, every parameter that eventually culminate in loss calculation gets judged by its contribution to the loss function. Below is backpropagation of errors in equation form for our 1 layer neural network in eqn 3.1,

$$\frac{\partial L}{\partial W_{ij}} = \frac{1}{M}\sum_{m}(N^m(W_{ij}) - y^m)\frac{\partial N^m(W_{ij})}{\partial W_{ij}} \tag{3.3a}$$

$$\frac{\partial N^m(W_{ij})}{\partial W_{ij}} = a_i\sigma\prime(\sum_{j}(W_{1j}X_j + b1))X_j \tag{3.3b}$$

Here M is the total number of data points, with m denoting a single element of that set. $\sigma\prime$ denotes the derivative of the specific activation function used. It is because of this step that the activation functions in neural networks must be differentiable. With differentiable activations, calculating derivatives with backpropagation scales linearly with the number of data points. Doing the same task using the usual numerical differentiation methods would have resulted in $\mathcal{O}(n^2)$ scaling. This linear scaling with input, combined with parallel computing is what enables use of "big data".

One final note regarding the use of gradient descent is that when passing through the data only a small portion of that data is seen at a single update step. This considerably speeds up the optimization process, even though each step is somewhat random.

3.4 Quality of Fit

The procedure explained above is quite simple but in practice there are many pitfalls. Because neural networks contain a large number of parameters, it's possible for them to overfit the data ("memorizing" data), fitting so closely that the model matches the noise in the data and end up generalizing poorly. It is also possible to have a network fail to extract all information from the data, which is called underfitting. These can be identified via observation of loss value over time during training.

For purpose of testing generalization, a small subset of the total data called a "validation set" is used. The neural network never sees this portion during training, only

the loss value over this subset of data is calculated as well as the loss value over the training data (or training set). During the training these two values should decrease together. A decrease in loss value of training data while loss value of validation data is either increasing or remaining the same implies overfitting. The last iteration at which loss value of training data and validation data decreased together is the optimal time to stop training. Anytime before this would be underfitting, anytime after is overfitting. This method is called early stopping.[17]

There are many other methods to control overfitting. Most straightforward option is just having more data, as it's difficult to memorize all the training data when there is too much of it. Another method is called (weight) regularization, in which a term proportional to the weights or the square of the weights is added to the loss function. This penalizes using too many weights unnecessarily at the expense of a control constant, the regularization hyperparameter. This hyperparameter controls how strongly weights contribute to the total loss. Searching and using a value as close to optimal for weight regularization yields models that generalize with more accuracy.

Another method is called dropout, where during training some random portion of the neurons in a layer are disabled by temporarily setting their activations to zero for a single iteration. This forces the network to use less of its available capacity at each iteration without sacrificing potential benefits from redundancy. The proportion of dropped neurons is another hyper-parameter, however it's almost always optimal to use p=0.5 for dropout rate. [18]

When training is finished, further tests of the model's quality are done on a separate, third set of data called the test set. Since validation data is used to stop training at an optimal point, model is optimized over the validation set implicitly. There could be a configuration of parameters that give higher loss on validation set, but lower in a much more diverse data set. Those configurations aren't chosen with early stopping, but the procedure identifies overfitting to training data and is a staple of the machine learning approach.

Checking the quality of the final model can be done through many ways. While optimization procedure uses loss function (specifically mean squared error) as the metric of choice, any statistical metric is appropriate. These may be mean absolute

error, mean squared error, linear correlation coefficient, lin's concordance correlation and many others.

A way to probe the precision of the model is sampling initial values. Upon acquiring an initial weight configuration the network is trained until desired conditions are reached. This process is repeated with everything held exactly the same except the initial values. Several trained networks can then be used to estimate the standard deviation of the model. The causes that would contribute to such a variation are initial values and different local minima the network ends up in after training. Since the training and validation data, optimal stopping condition, network structure and optimization parameters are held constant, this amounts to estimating model uncertainty. It should also be noted here that in our case, the loss value is convex and as a result has a global minima which also is the only local minima. There are, however, many different configurations of weights that would yield results close to this value. The outlined method samples exactly these.

3.5 Hyperparameter Choice

Before actual training starts, we need to set the hyperparameters to some certain values that will serve us in the best manner possible. To facilitate this, most straightforward approach is to use brute-force search over many possible values for each hyperparameter. Let's say we are concerned with a starting learning rate l_0 and a rate reduction parameter τ to be used in $\frac{l_n}{1+\tau n}$ for the $n-th$ update step. In this case sampling 5 points each for a total of 25 possible settings is quite feasible. This would be followed with a grid of values with less difference between them, to hone in the hyperparameters. As an example, first search for learning rate could be for values $10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1$ while second search, assuming the first returned 10^{-3}, would include $5 \times 10^{-3}, 3 \times 10^{-3}, 10^{-3}, 9 \times 10^{-4}, 7 \times 10^{-4}, 5 \times 10^{-4}$.

In practice, number of hyperparameters are higher than 2. How many data points are included in a single update (batch size), regularization parameter (if there is one), number of layers of the network, number of neurons to consider in these layers and many more can be considered. Even the choice of which adaptive gradient descent

algorithm can be included in this kind of search. As a result of this, the number of dimensions for hyperparameters grows and with it potential configurations grow in a multiplicative manner. This results in brute-force grid searches to be costly and time consuming. If we had 5 hyperparameters with 5 options each, we would need to check 5^5 possible settings, and then another time with smaller differences between values.

Hyperband is an algorithm that takes a different approach to hyperparameter tuning. While it takes the entire set of possibilities, it samples different amounts of configurations in many iterations of the algorithm. Hyperbands starts with most exploratory setting, selecting as many different configurations as possible and then tests them minimally. Then a fraction $1/r$ of the top performing results are chosen and trained longer by r times. This approach is repeated several times, managing the trade-off between need of exploration and finite resources. Each iteration after the first selects fewer configurations and trains them longer. Li et. al. report that hyperband is 20 times faster than random sampling. [19]

3.6 Why Deep?

When it comes to neural networks, there's an important difference in how the universal approximation theorem is proved and how the real algorithm works with finite size and resources. In theory any neural network with at least 1 hidden layer approximates universally, at least for all partially continuous functions. In practice, however, deeper networks work better by a significant margin compared to shallower ones. Indeed, the latest resurgence in machine learning is mostly because of increasing practicality of deep learning. Shallower, smaller networks don't need as much data to avoid overfitting, computational efficiency in training isn't as big a concern since number of parameters are small, some of them can be analytically solved so newer algorithms aren't needed. Despite all these deep networks manage to do things single layer networks simply can't.

This can be easily understood by pointing out and explaining the most important step of machine learning before deep networks became popular. Feature extraction,

coming up with numerical summaries of data that are most useful for the end goal of the practitioner was an integral step. It's this step that deep learning does by itself.

Let us pursue this from a visual analogy of identifying faces. We can do it almost immediately in our daily lives. A neural network does this by identifying distinct portions of the input pixels to specific names. What happens in a deep network is, on penultimate layers a face is broken up into a collection of facial features. Eye shapes, ears, mouth and overall geometry of a face are combined together. For Pinochio those are a nose that's too long, thin overall shape with small ears and mouth. Each of these features is also broken down in smaller units, an ear shape is a collection of curves at certain angles and coloration. All penultimate features are crafted from a learned list of basic shapes, layer by layer. Right after the input layer, the only features to be found are lines that delineate edges in the picture. From a combination of these, curved features are made. From the curved features, specific and different shapes for eyes, ears etc are obtained. At the last step, a combination of these features help us identify Pinochio.

The visual analogy is relatively straightforward to understand. This kind of examination becomes quite difficult once more abstract input-output relations are considered. For the main study in this thesis, geometrical insights in specific layers don't enlighten us since we don't *see* daily solar energy with eyes collecting bright sunshine fraction and average temperature. However, the principle behind deep learning remains the same. During training a deep neural network will learn features directly relevant to the task it's made to do. More layers mean a neural network can learn more complicated, specific features.

From a physics based viewpoint, progressing through the layers of a neural network we identify more specific distributions of light. Each layer has less entropy, hence carves a more precise slice from the space of possible features. This phenomenon is behind deep neural networks' success.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Data

Data used in our study is provided via courtesy of Devlet Meteoroloji İşleri (DMİ). It covers 39 cities with a total of 46 measurement stations distributed in them.

Main data consists of average temperature per day and total solar power per ten seconds. The solar power is subjected to the filter of 120 $Wm^{-}2$ cutoff criteria for the purpose of determining bright sunshine hours [20] and summed for a given day. A similar sum is applied to solar energy values as well. Daylength is calculated according to formula 4.1 below. Solar energy incident on the outer layer of the atmosphere is calculated according to eqn 4.2. Latitude data for each station is obtained from DMİ.

$$N = \frac{2}{15} cos^{-1}(-tan\phi tan\delta) \tag{4.1}$$

$$H_0 = \frac{24 \times 3600 G_s c}{\pi}(1 + 0.033cos(\frac{360n}{365}))(cos\phi cos\delta sin\omega_s + \frac{\pi\omega_s}{180} sin\phi sin\delta) \tag{4.2}$$

In these equations ϕ is latitude, δ is the declination angle, and ω_s denotes the hour angle at sunset. These angles can be seen in fig 4.1.

For the purposes of training a neural network, we divided total data over 39 cities into three distinct sets. These are the training set which is used to train the network, the validation set which serves as a check on memorization and the test set, which the

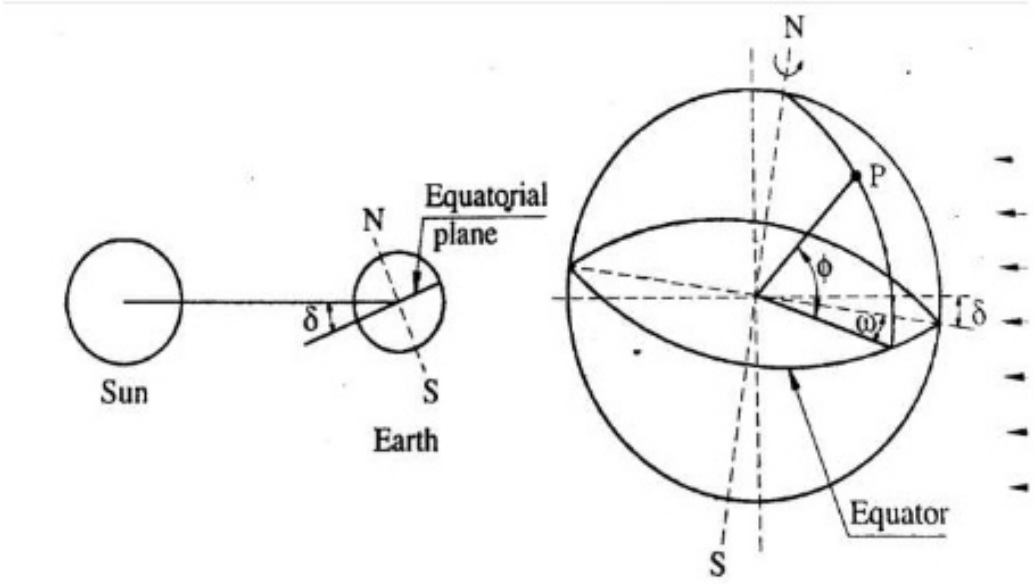

Figure 4.1: © https://image.slidesharecdn.com/earthsunangle-150224223317-conversion-gate02/95/earth-sun-angle-3-638.jpg?cb=1424819965
Sun-Earth angles; δ is declination angle, ϕ is latitude and ω is hour angle.

network never interacts with during training. We chose five cities to be in validation set and ten for the test set. For training set, two distinct subsets are used to compare with each other, one where all 24 of the remaining cities are present and another with a single city from each major climate region in Turkey. This last division is aimed at testing the data requirement for obtaining an accurate estimate. Division of stations into the mentioned clusters is listed in table 4.1

4.2 Details of the Neural Network

The main architecture utilized was multi-layer perceptron (MLP). Number of layers are determined with the hyperband algorithm [19]. All considerations included in the hyperparameter configurations are tabulated in table 4.2

Once the algorithm ran its course, top 20 hyperparameter configurations out of the 196 sampled are sorted from least error to highest. Modal average of the hyperparameters in this list are taken as the result. These results determine the best optimizer to use is $adamax$ [21] for minimization of mean absolute error (MAE). The lowest errors

Table 4.1: Names of meteorological stations included in the data. In training set bolded stations are the ones used for smaller version, all of it for larger version.

Stations in Training Set	**Çanakkale, Isparta, Karaman, Van Bölge, Ergani, Ünye**, Rize, Artvin, Bolu, Tokat, Gümüşhane, Ağrı, Doğubeyazıt, Bursa, Gemerek, Divriği, Afyonkarahisar, Malatya, Akşehir, Kulu, Hakkari, Adana Bölge, Tercan, Kemalpaşa, Menemen, Malazgirt, Palu, Develi, Göksun, Ulukışla, Bozova
Stations in Validation Set	Kırklareli, Tarsus, Şırnak, Tortum, Boğazlıyan
Stations in Test Set	Ceylanpınar Tigem, Kilis, Elmalı, Burhaniye, Kastamonu, Kars, Solhan, Suşehri, Beyşehir, Aksaray

Table 4.2: Full list of hyperparameter configurations utilized in Hyperband algorithm. Parameters used for actual training are in bold.

Scaler	**StandardScaler**, RobustScaler, MinMaxScaler, MaxAbsScaler
Layer Count	1, **2**, 3, 4
Initialization Distribution	Uniform, **Normal**, Glorot Uniform, Glorot Normal, He Uniform, He Normal
Batch Size	16, 32, **64**, 128
Loss Function	**Mean Absolute Error**, Mean Squared Error
Optimizer	Adam, AdaGrad, **Adamax**, Nadam

were obtained with 2 layers. We also observed that dropout [18] as a method to counter overfitting was useful with random 40% of the neurons at the penultimate layer ignored on each update. First layer contained batch normalization [22] to keep gradients in range (0,1) so training doesn't slow down.

Guided by the hyperband results, we trained a two layer MLP with parametric ReLU activations [23] where the first layer has 20 and the second has 15 neurons respectively. Validation set has been used to determine the optimal time to stop training [17].

We considered subsets of potential inputs as well as a distinction for each input subset of directly estimating daily solar energy per squared meter (H) versus daily solar energy per squared meter as a fraction of extraterrestrial radiation ($\frac{H}{H_0}$). We trained 50 networks each with the chosen hyperparameters and inputs, with different initializations to gauge model variance. Table 4.3 contains the mean performance results according to several different metrics.

4.3 Performance Analysis

Our complete performance results are presented according to several metrics, averaged over fifty ANNs trained per result. Table 4.3 contains these while table 4.4 lists the standard deviations around the means.

In the aforementioned tables, MAE stands for Mean Absolute Error and is computed according to $MAE = \frac{1}{M}\sum_m |N^m - y^m|$ where N^m is the estimate of the ANN (or other model) and y^m is the measured value for $m-th$ datum of all M values in data. RMSE stands for Root Mean Squared Error, calculated according to $RMSE = \frac{1}{2M}\sum_m (N^m - y^m)^2$. These two errors are the main drivers in our analyses.

Other metrics are squared linear correlation coefficient (R^2) which is calculated according to eqn 4.3, percentage of explained variance (EVS) calculated with eqn 4.4, fraction within error (FIE) for 0.5, 1 and 1.5 MJs within RMSE calculated with eqn

Table 4.3: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) in MJm^{-2}. R^2 is squared correlation coefficient. EVS is Explained Variance Score. FIE(K) refers to Fraction in Error K where K is a numerical amount in MJm^{-2}. For example, FIE(1) shows the fraction of predictions within 1 MJm^{-2} of the measurements. All comparisons are between predictions and measurements. Input column shows the subsets of data used in training. n is bright sunshine duration, N is daylength, T is average temperature, ϕ is latitude. Rows including H refer to networks trained to estimate daily solar energy directly rather than the fraction of extraterrestrial radiation $\frac{H}{H_0}$. Rows with 6 and 31 are for networks trained with 6 or 31 stations respectively.

Input	RMSE	MAE	R^2	EVS	FIE(0.5)	FIE(1)	FIE(1.5)
$n, N-6$	3.944	1.974	0.788	0.793	0.271	0.494	0.651
$n, N-31$	3.875	1.943	0.796	0.799	0.277	0.505	0.659
$n, N-H6$	4.338	2.223	0.743	0.760	0.218	0.414	0.572
$n, N-H31$	3.992	2.072	0.783	0.796	0.245	0.456	0.615
$n, N, \phi-6$	3.986	2.197	0.784	0.788	0.214	0.402	0.554
$n, N, \phi-31$	4.134	2.136	0.767	0.779	0.246	0.460	0.615
$n, N, \phi-H6$	4.378	2.367	0.738	0.755	0.199	0.377	0.526
$n, N, \phi-H31$	4.123	2.112	0.768	0.782	0.244	0.452	0.610
$n, N, T-6$	4.049	1.935	0.777	0.781	0.292	0.524	0.682
$n, N, T-31$	3.886	1.884	0.794	0.798	0.299	0.534	0.690
$n, N, T-H6$	4.210	2.136	0.758	0.772	0.236	0.443	0.608
$n, N, T-H31$	4.015	2.020	0.780	0.793	0.258	0.476	0.639
$n, N, T, \phi-6$	4.099	2.144	0.771	0.777	0.228	0.428	0.588
$n, N, T, \phi-31$	4.043	1.941	0.777	0.782	0.292	0.524	0.680
$n, N, T, \phi-H6$	4.297	2.238	0.748	0.763	0.221	0.412	0.568
$n, N, T, \phi-H31$	4.094	2.074	0.772	0.787	0.244	0.457	0.620
Universal2.2	4.164	2.231	0.770	0.771	0.231	0.428	0.588
Quadratic in s	4.375	2.374	0.746	0.773	0.224	0.426	0.580
Quadratic in s, T	4.323	2.332	0.752	0.776	0.230	0.438	0.593

Table 4.4: Standart Deviations of ANNs obtained by 50 different initializations each

Input	RMSE	MAE	R^2	EVS	FIE(0.5)	FIE(1)	FIE(1.5)
$n, N-6$	0.088	0.020	0.010	0.010	0.006	0.008	0.007
$n, N-31$	0.048	0.015	0.005	0.005	0.003	0.004	0.004
$n, N-H6$	0.266	0.062	0.032	0.035	0.015	0.023	0.026
$n, N-H31$	0.112	0.045	0.013	0.012	0.010	0.014	0.015
$n, N, \phi-6$	0.113	0.055	0.012	0.011	0.010	0.015	0.019
$n, N, \phi-31$	0.220	0.177	0.026	0.019	0.020	0.033	0.037
$n, N, \phi-H6$	0.226	0.134	0.028	0.027	0.015	0.026	0.031
$n, N, \phi-H31$	0.222	0.061	0.026	0.027	0.013	0.018	0.018
$n, N, T-6$	0.086	0.025	0.010	0.009	0.007	0.010	0.008
$n, N, T-31$	0.061	0.020	0.006	0.006	0.006	0.007	0.007
$n, N, T-H6$	0.193	0.054	0.023	0.023	0.016	0.024	0.023
$n, N, T-H31$	0.118	0.059	0.013	0.013	0.017	0.023	0.021
$n, N, T, \phi-6$	0.099	0.066	0.011	0.011	0.015	0.024	0.028
$n, N, T, \phi-31$	0.151	0.051	0.017	0.015	0.008	0.012	0.011
$n, N, T, \phi-H6$	0.232	0.082	0.027	0.028	0.014	0.022	0.025
$n, N, T, \phi-H31$	0.135	0.054	0.015	0.016	0.017	0.024	0.024

4.5.

$$R^2 = \frac{(M * \sum_{m=1}^{M} N^m y^m - \sum_{m=1}^{M} N^m \sum_{m=1}^{M} y^m)^2}{M(\sum_{m=1}^{M}(N^m)^2 - (\sum_{m=1}^{M} N^m)^2)(M(\sum_{m=1}^{M}(y^m)^2 - (\sum_{m=1}^{M} y^m)^2)} \tag{4.3}$$

$$EVS = 1 - \frac{Var(N-y)}{Var(y)} \tag{4.4}$$

$$FIE = \frac{number\ of\ data\ where\ (N^m - y^m > E)}{total\ number\ of\ data} \tag{4.5}$$

In calculation of FIE in eqn 4.5 E stands for the difference of RMSE desired. In our calculations 3 different values of E are used; 0.5, 1.0 and 1.5 MJ.

The main takeaways that we derive from table 4.3 are as follows. There is a small difference between calculating H vs $\frac{H}{H_0}$ where the latter option is more accurate. Inclusion of latitude explicitly results in higher error. Using more data mostly leads to better accuracy. Detailed analyses are presented in the following paragraphs.

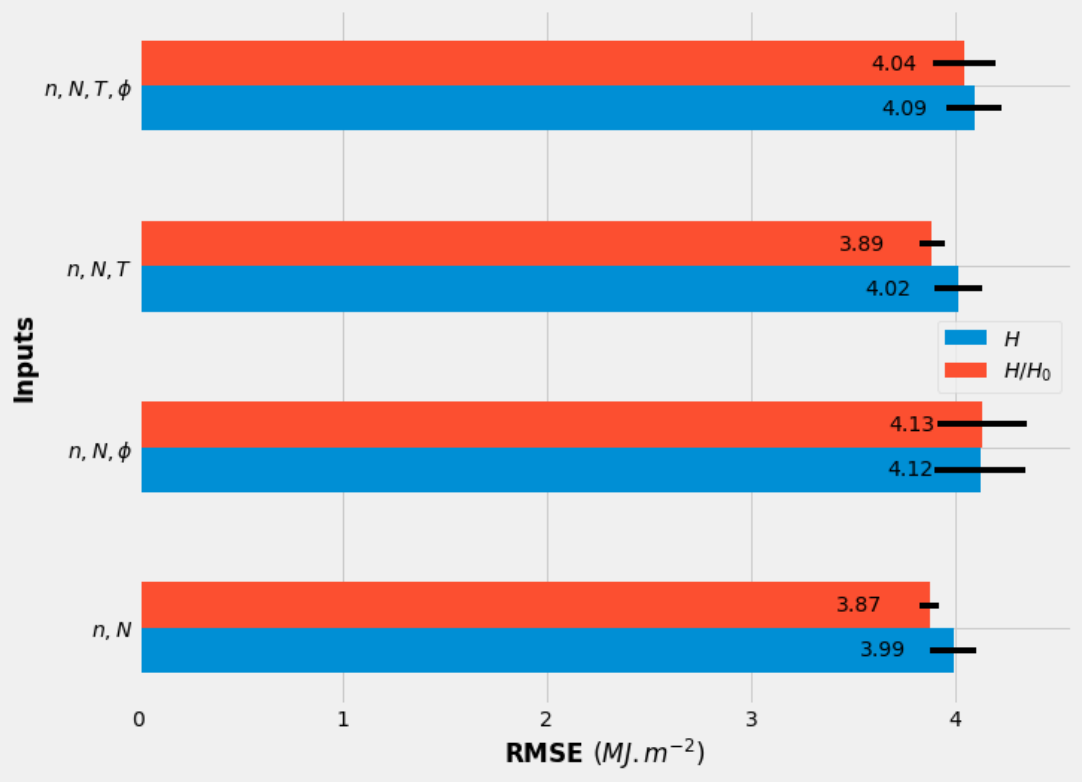

Figure 4.2: Fractional or Direct Estimation

Figure 4.2 shows the comparisons for estimating H directly or as a fraction of extraterrestrial radiation. Difference is small but consistent. The reason for this could be that fractional results are between $(0, 1)$ and neural networks train better when the inputs and outputs are in a smaller bounded region (strictly between 0 and 1) versus a larger unbounded one(between 0 MJ and an unspecified high value).

In the comparison of training data amounts, which is shown in 4.3, outlook is mostly as expected. More data leads to more accurate estimations as well as lower variation. The exception here is with the explicit inclusion of latitude ϕ in the inputs. ϕ is already accounted for in calculating daylength N and extraterrestrial radiation H_0 so it's already a part of the model implicitly. We speculate that the mechanism outlined in Wilson et. al. [24] is responsible for increasing errors with more data when ϕ is included explicitly. It is also noteworthy that standard deviations only go higher when $\frac{H}{H_0}$ is the target to estimate, however, since inclusion of ϕ lowers overall estimation accuracy, lower variation isn't useful for any application.

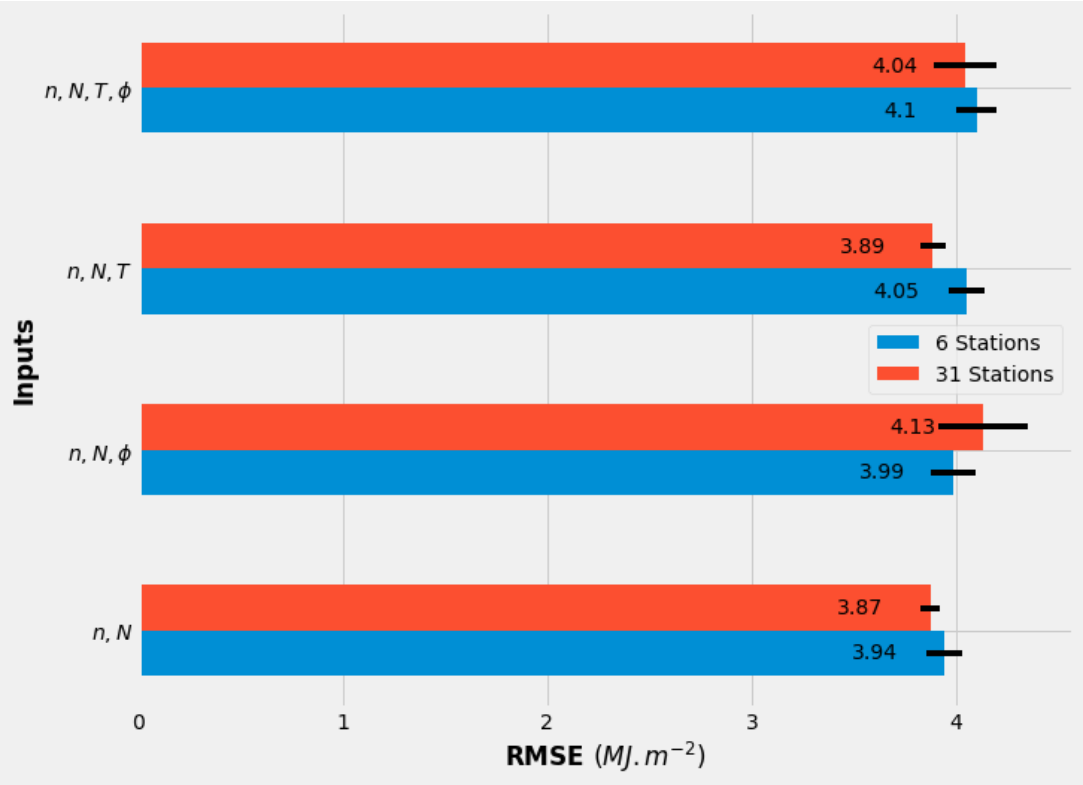

Figure 4.3: Data Amounts

Table 4.4 holds another interesting information. We observe that estimating H vs $\frac{H}{H_0}$ results in higher variance, again with the exception of explicit ϕ in the inputs.

The most important comparison is between classical methods of estimation versus the ANNs. In figure 4.4 the quadratic model with universal coefficients (eqn 2.2) proves slightly more accurate than a fit made to local data by a margin of 0.24 MJ. Inclusion of T as input slightly improves accuracy for polynomial model but not for ANNs. The main comparison is to be made between mean ANN performance and the universal model where the ANN outperforms by a margin of 0.29 MJ in RMSE, in favor of ANNs.

To put this number into context, the measurement error of most common sunshine recorders regarding daily solar energy is 2% around the reported value. This gives, for our dataset, a value of 0.32 MJ mean measurement error for daily totals over all stations.

We observe one final point about input variables. Inclusion of daily average temperature T yields negligible accuracy loss with ANNs, and a similar amount of accuracy gain with empirical fits using classical regression. We suggest that average temperature during day isn't too important by itself in explaining the variation of daily solar energy, especially when n and N are already present and account for much of the variance by themselves.

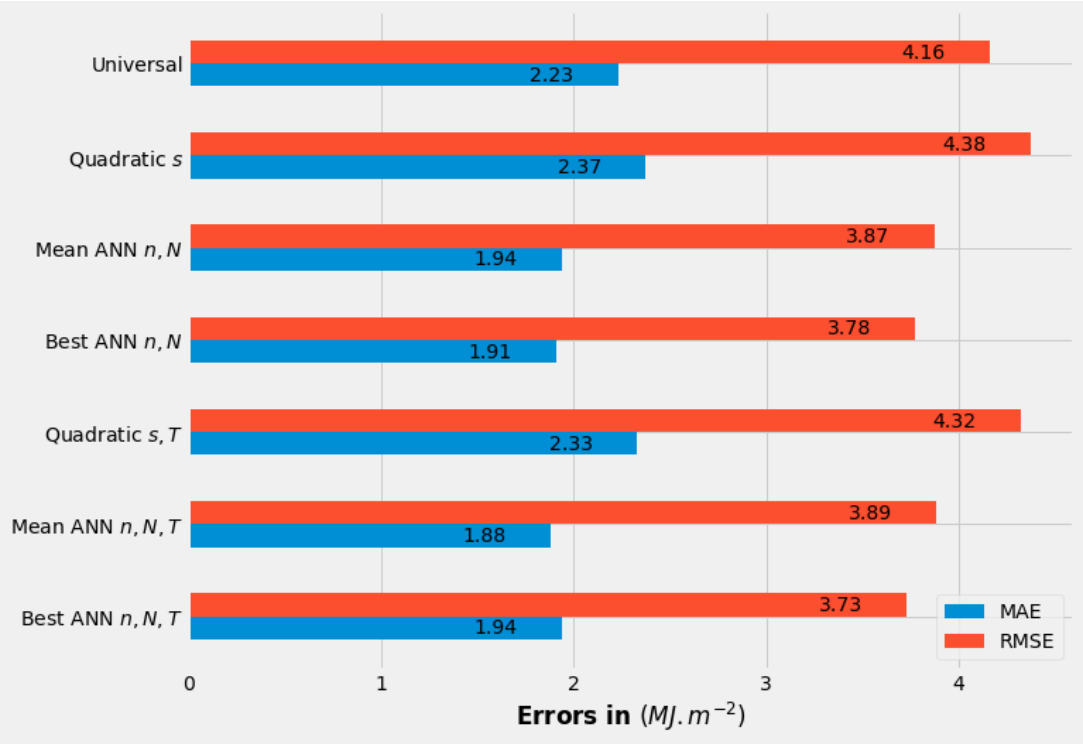

Figure 4.4: ANN vs Classical

CHAPTER 5

CONCLUSION

We use the most basic and fundamental climatology variables in estimation of daily solar energy totals, using classical regression, universal relation and best multi layer perceptron (MLP) setup we managed to setup. We find that explicit dependence on latitude is detrimental and that except for explicit ϕ more data leads to lower errors. H/H_0 as the target of estimation instead of H also leads to lower errors. For the most important comparison however, we can state the following as a succinct summary: MLPs offer modest benefits in accuracy but incur a high cost on practicality.

For purposes deemed important, use of MLPs can offer advantages not just in accuracy but also in ease of estimation of variance in the predictions. Although machine learning appeared very early in the literature, modern hardware improvements within the last decade made the training of ANNs a less time intensive task. In this view, being able to sample many different initializations to gauge how varied the results get is a valuable tool for any decision maker who would like to know how uncertain a given model is.

The accuracy benefit, however, remains small - right around the ballpark of measurement error. Of course inclusion of more variables may help reduce prediction errors.

One of the reasons why deep learning is so effective is its suitability to model causal events. For this reason it would be an interesting study to increase directly related weather parameters as input to a neural network and try to estimate solar energy from those values only. The resulting network would need to be a model of how climate works, albeit an incomprehensible one for humans.

Main limitation in our study, however, is the architecture of the network. While MLPs

suffice for an estimation task like this, the line of study mentioned above could and should be pursued with state of the art architectures, like the transformer. The main benefits of such an architecture would be its simplicity regarding hyperparameter tuning, its capacity to handle both sequential and non-sequential relations in the data at the same time. This would allow a network like that to learn the cyclic patterns year-over-year and daily, long term trends with possibility of such a model to learn causal effects of geographical distance.

Whatever the future may bring, it certainly holds immense potential for climate modeling and forecasting. Estimation on the other hand, is better served by classical methods except in niche cases.

REFERENCES

[1] J. Rajesh Kumar, R.K.Aggarwal, "Comparison of regression and artificial neural network models," *Renewable and Sustainable Energy Reviews*, vol. 52, pp. 1294–1299, 2015.

[2] M. Ozgoren, M. Bilgili, and B. Sahin, "Estimation of global solar radiation using ANN over Turkey," *Expert Systems with Applications*, vol. 39, pp. 5043–5051, apr 2012.

[3] B. Akinoğlu and A. Ecevit, "Construction of a quadratic model using modified Ångstrom coefficients to estimate global solar radiation," *Solar Energy*, vol. 45, pp. 85–92, jan 1990.

[4] F. Besharat, A. A. Dehghan, and A. R. Faghih, "Empirical models for estimating global solar radiation: A review and case study," *Renewable and Sustainable Energy Reviews*, vol. 21, pp. 798–821, may 2013.

[5] M. Yorukoglu and A. N. Celik, "A critical review on the estimation of daily global solar radiation from sunshine duration," *Energy Conversion and Management*, vol. 47, pp. 2441–2450, sep 2006.

[6] A. K. Yadav and S. Chandel, "Solar radiation prediction using Artificial Neural Network techniques: A review," *Renewable and Sustainable Energy Reviews*, vol. 33, pp. 772–781, may 2014.

[7] S. Shamshirband, K. Mohammadi, P. L. Yee, D. Petković, and A. Mostafaeipour, "A comparative evaluation for identifying the suitability of extreme learning machine to predict horizontal global solar radiation," *Renewable and Sustainable Energy Reviews*, vol. 52, pp. 1031–1042, dec 2015.

[8] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Nodes," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 17, no. 4, p. 879, 2006.

[9] A. Aggarwal, "Genesis of AI: The First Hype Cycle," 2018.

[10] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[11] M. A. Nielsen, "Neural Networks and Deep Learning," 2015.

[12] G. Cybenko, "Degree of approximation by superpositions of a sigmoidal function," *Approximation Theory and its Applications*, vol. 9, pp. 17–28, dec 1993.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, jan 1989.

[14] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, pp. 145–151, jan 1999.

[15] M. Riedmiller, M. Riedmiller, and H. Braun, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," *IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS*, vol. 16, pp. 586—-591, 1993.

[16] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," dec 2012.

[17] Y. Yao, L. Rosasco, and A. Caponnetto, "On Early Stopping in Gradient Descent Learning," *Constructive Approximation*, vol. 26, pp. 289–315, aug 2007.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[19] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," mar 2016.

[20] W. M. O. World Meteorological Organization. and W. M. O. (WMO), *Guide to meteorological instruments and methods of observation.* WMO, World Meteorological Organization, 2018 editi ed., 2008.

[21] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," dec 2014.

[22] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," feb 2015.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," feb 2015.

[24] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The Marginal Value of Adaptive Gradient Methods in Machine Learning," may 2017.