UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LÚCIO PEREIRA FRANCO

Neural Networks for Solar Panels Efficiency Prediction

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engeneering

Advisor: Prof. Dr. Lucineia Heloisa Thom
Coadvisor: Dr. Pascal Hirmer
From University of Stuttgart

Porto Alegre
May 2022

The real test is not whether you avoid this failure, because you won't.
It's whether you let it harden or shame you into inaction, or whether you learn
from it; whether you choose to persevere.

— BARACK OBAMA

ACKNOWLEDGEMENTS

I want to acknowledge the institutions that held this work possible: the Federal University of Rio Grande do Sul and the University of Stuttgart. Furthermore, I am grateful to all the academics who taught me during my graduation, specifically those who guided me while developing this work: Dr. Ana Franco, Prof. Dr. Lucineia Thom, and Dr. Pascal Hirmer.

Thanks to Lucas Heck, Laura Corssac, Mariana Montoya, and Diana Farje, who were the pilar of my sanity when lockdown hit Germany. You were my bit of home there when I was away from my family. I extend my thanks to Christian Schmitz, Augusto Boranga, and Lucas Flores, whose friendship during the graduation also kept me going further.

Thanks to all friends who the university brought into my life. Henrique Valcanaia, Cristiano Lunardi, Camila Schmidt, and Rodrigo Wuerdig, thank you for all the nights we spent trying to solve (quite possible now) problems. Thank you for all the well-deserved commemorations we did and the ones we will do.

My gratitude to the founders of NiC and my best friends: MSc Felipe Nepomuceno, Ian Lane, Arthur Vedana, Rodrigo Scroferneker, and Gustavo Paese. Your partnership made me who I am today. Thank you for all the countless funny moments, hearing my complaints, and for supporting me during these years. I am really proud of loving you guys.

A thousand thanks to my beloved girlfriend, Evelyn Holz, that lived with me, giving support during all the paper publication and this work completion. You bring color to life and make everything seems possible.

My deepest love and gratitude to all my family members, especially my father, João, mother, Márcia, and two brothers, Bruno and Vitor, who did not measure efforts to support me while I focused on my studies. You are my strength in challenging times and my light through the uncertainties of life. I want to extend my special thanks to my grandfather, a UFRGS Engineer, who recently passed away. I wish you could see me graduate.

ABSTRACT

In 2018, more than 100GW worth of solar panels were installed worldwide. The increase in solar energy systems is beneficial for the environment, but some companies have trouble implementing it given its inconsistent generations of energy. This inconsistency creates energy fluctuations that can overcharge the electrical grid or complicate load demand predictions. This work proposes an approach that uses distributed data and machine learning to reduce these risks by making these energy fluctuations more predictable. We implemented neural network models to evaluate data that connects solar panels' efficiency with weather conditions. Companies and Smart Grid operators can use our approach and models with forecasting data to predict the next-day solar energy generation and adapt decisions considering fluctuations. Results show that our application achieved with an absolute error of less than 10% in some studied cases.

Keywords: Machine Learning. Neural Networks. Solar Energy. Photovoltaic Systems.

Redes Neurais para Predição da Eficiência de Painéis Solares

RESUMO

Somente em 2018, mais de 100 GW em novos painéis solares foram instalados no mundo. Mesmo que essa fonte energética seja benéfica para o meio ambiente, muitas organizações enfrentam dificuldades em implementá-la devido à sua geração de energia inconsistente. Essa inconsistência cria flutuações de energia que podem sobrecarregar a rede elétrica ou complicar o cálculo da demanda de energia. Esse trabalho propõe uma abordagem usando dados distribuídos e aprendizagem de máquina para reduzir esses riscos. Para isso, nós implementamos modelos de redes neurais para analisarem dados que relacionam a eficiência de painéis solares com condições climáticas. Companhias podem usar nossa abordagem e modelos para prever a energia que será gerada no próximo dia e se programar melhor para flutuações de energia. Os resultados mostram que a nossa abordagem atingiu níveis de precisão prósperos, tendo menos de 10% de erro absoluto em suas medições em alguns cenários estudados.

Palavras-chave: Aprendizagem de Máquina, Redes Neurais, Painéis Solares, Sistemas fotovoltaico.

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
CSV	Comma-separated values
CNN	Convolutional Neural Network
GCPVS	Grid-Connected Photovoltaic System
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Square Error
PV	Photovoltaic
RMSE	Root Mean Square Error

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1 INTRODUCTION

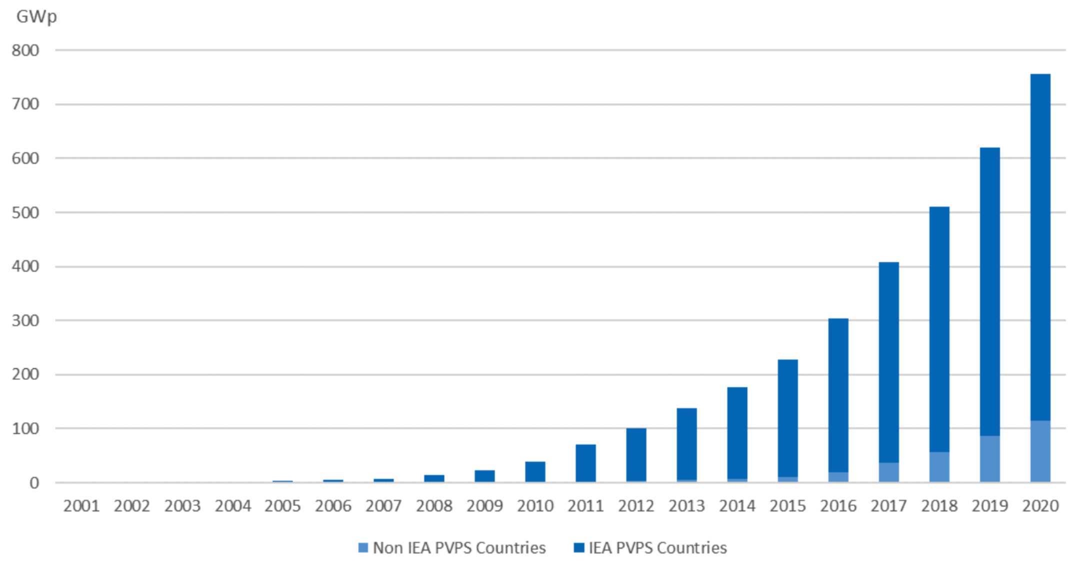
The installation of solar panels has been growing exponentially since earlier records of 2008 (DIRAC, 2019b). Despite the global COVID-19 pandemic, this growth has not slowed down. In 2019, countries from the International Energy Agency (IEA) installed more than 77,9 GW worth of solar panel systems (DIRAC, 2019b). The same group installed 97,6 GW in 2020 (DIRAC, 2019a). The USA was responsible for 19,2GW of those, an increase of approximately 44% compared to the previous year. Figure 1.1 depicts the total PV Systems' power installed over the last decade. Of these systems, 95%-99% are Grid-Connected Photovoltaic Systems (GCPVS). GCPVS utilizes the grid electricity as either an energy backup for less productive hours or as a gateway for more productive hours (DIRAC, 2019b).

The increase in solar energy systems is beneficial for the environment, having produced energy in 2020 that would otherwise have caused the emission of 800Mt of CO2 (DIRAC, 2019b), though it also raises new challenges. One of them is the unpredictability of the output power of Photovoltaic (PV) panels. These variations on GCPVS hinder the ability of energy providers to plan and manage electricity demands accurately. In fluctuation peaks, these systems can stress or overcharge the electrical grid (OBI; BASS, 2016). They also make the load demand prediction more difficult because operators cannot access *behind-the-meter* information.

Considering these challenges, models capable of predicting solar panels' output are interesting for designing a PV System or estimating load demand. Other authors have already explored the subject of Solar Energy Forecasting. Previous research in this field approached the problem using statistical models (SAFI; ZEROUAL; HASSANI, 2002). A study from 2011 used linear regression to arrive at an equation capable of establishing the relation between solar energy and weather factors (Sharma et al., 2011). A recent study has proposed Long-Short Term Neural Networks (LSTM) with Convolutional Neural Networks (CNN) to solve the same problem and achieved promising results (Hossain; Mahmood, 2020). These and similar works are discussed in Section 2.6.

The problem of predicting solar energy generation is well suited for machine learning algorithms, given its non-linearity and the high volume of data available. Adel Mellit et al. (MELLIT et al., 2020) reviewed several different machine learning approaches to solar energy prediction (e.g., Deep-NN, LSTM-NN, CNN). In their paper, they noticed a lack of variability in the studies. Most of them focus on a single location

Figure 1.1: Global evolution of cumulative PV installations.



Source: IEA PVPS

(SON et al., 2018) (HANIFULKHAIR et al., 2020), and the authors noticed that little work has been done with regional models (PIERRO et al., 2017). A model built with data from a specific location or Photovoltaic System does not adapt well when predicting events for other scenarios.

1.1 Research Questions and Goals

If the solar energy market continues to grow, applications to better manage these technologies will be fundamental for improving performance and avoiding disasters. The challenges mentioned previously and the available research on this topic raise the following research questions:

- **[RQ1:]** Is it possible to predict solar energy generation of a PV System only by using data from PV Systems around it?
- **[RQ2:]** Can we use distributed and weather forecasting information to predict regional next-day energy generation?

To answer these two research questions, we propose an approach to predicting solar energy generation with Deep Neural Networks (DNN) (MOOLAYIL, 2018). We use open data from solar panel community owners and historical weather information to train our machine learning model. Instead of analyzing a single PV System, our approach uses multiple data sources, combining several PV Systems to compose our dataset.

This work aims for a more generic model capable of predicting regional solar energy generation. With our findings, we hope smart grid operators can use weather forecasting to estimate the regional next-day energy generation and prepare themselves for grid fluctuations. We also intend to use our findings to analyze the impact of a new system in a region.

1.2 Text Organization

We organize the remainder of this work into 6 Chapters. Chapter 2 explains, using a high-level approach, all concepts, and technologies used to investigate our research questions. It also presents related works from the same field for posterior comparison. Chapter 3 describes the data used in our experimentation, its sources, and the scripts built to gather it. Then, in Chapter 4, we describe how we used these data to build a model capable of predicting solar energy generation. We present the results and evaluate the proposed scenarios in chapter 5. In addition, Chapter 5 also discusses our findings and compares them to related works. Finally, Chapter 6 concludes the text by giving an overall picture of our contributions. Chapter A, the appendix, includes all result plots from our training.

2 FOUNDATIONS

This chapter introduces the concepts and technologies used in this work. We will explain in a high-level approach the following topics: Photovoltaic Systems, Python, HTTP Requests and APIs, NumPy and Pandas, Neural Networks, and Tensorflow. We encourage the reader to look for a more extensive explanation in the referenced works.

2.1 Photovoltaic Systems

Photovoltaic (PV) modules, also called PV Panels, are composed of Solar Cells. The sunlight (photons) hits these solar cells, increasing their electrons' energy, creating current, voltage, and power (WHITE, 2014). The energy generated by the PV module yields a Direct Current (DC) that can be used to charge batteries.

PV Systems contain one or more PV Panels but are not only composed of them. Some have batteries to store the energy generated and charge controllers to prevent battery over- and under-charging. Since most of the energy we use daily is Alternate Current (AC) (e.g., our home appliances), PV systems usually contain one or more inverters. Inverters can convert DC energy into AC and vice-versa. Converting energy consumes power and impacts the performance of a PV System.

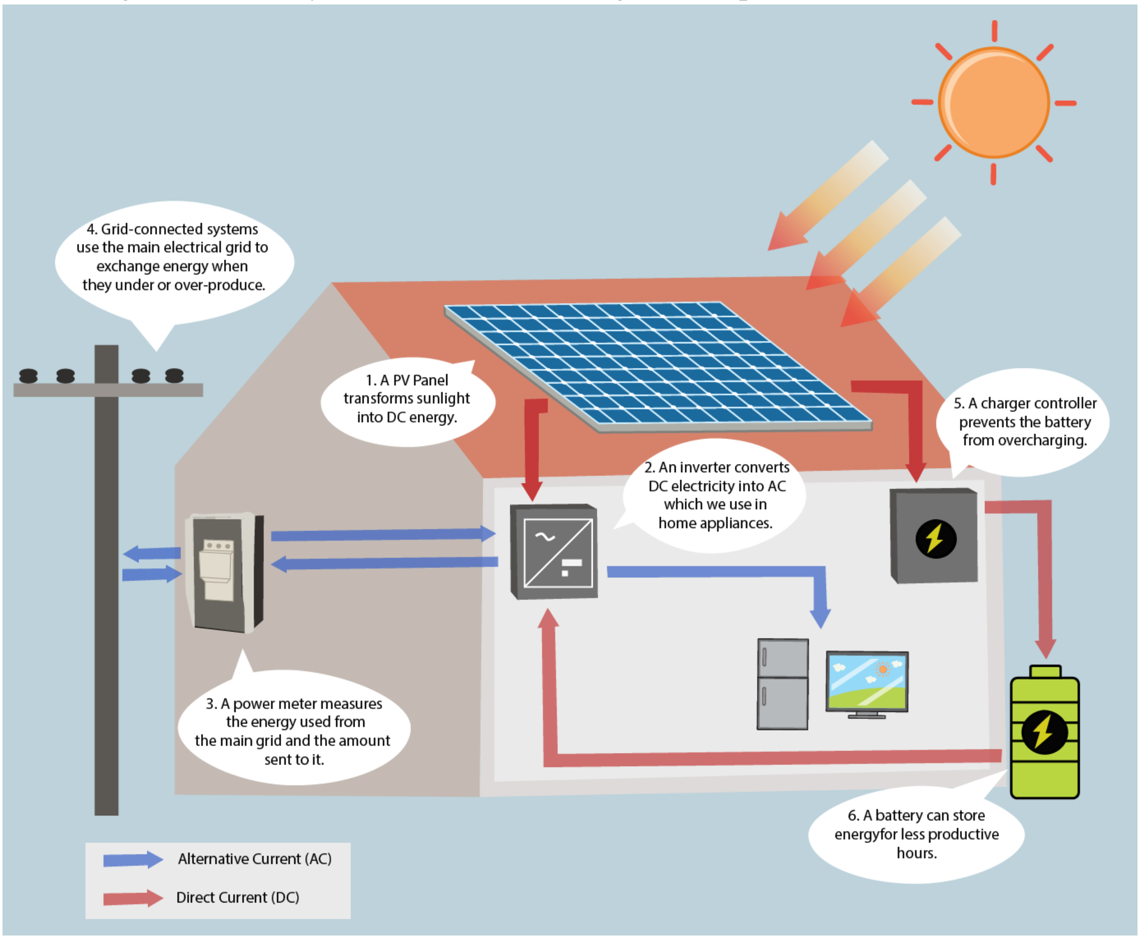
There are two types of PV systems: off-grid and grid-connected. Grid-connected Photovoltaic Systems (GCPVS) are connected to the electrical grid. They use the grid as a backup for when the PV Systems are not generating enough power (e.g., at night) and as a gateway for not storable or usable energy. A GCPVS uses a power meter to measure the energy used from the grid and the amount sent back to it. Off-grid Systems depend only on batteries, and when the batteries are complete, they need to discard the excess energy.

2.2 Python

Python is a programming language created by Guido van Rossum around 1991 (ROSSUM; BOER, 1991). It started with the core principles of being easy and intuitive, open-source, as understandable as plain English, and suitable for everyday tasks. Python is a general-purpose programming language, object-oriented, interpreted with dynamic semantics.

Frequently mentioned in the TIOBE Index (popularity) (BV, 2022), Python has an

Figure 2.1: PV System workflow showing the components' functionalities.


4. Grid-connected systems use the main electrical grid to exchange energy when they under or over-produce.
1. A PV Panel transforms sunlight into DC energy.
5. A charger controller prevents the battery from overcharging.
2. An inverter converts DC electricity into AC which we use in home appliances.
3. A power meter measures the energy used from the main grid and the amount sent to it.
6. A battery can store energyfor less productive hours.
Alternative Current (AC)
Direct Current (DC)

Source: The Authors

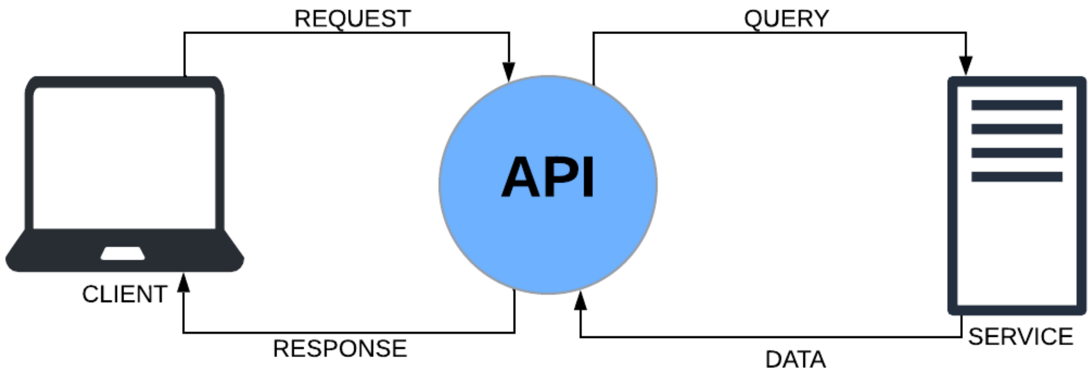
extensive community that backs up its open-source nature and creates powerful libraries. Travis E. Oliphant enumerates several of Python's qualities (OLIPHANT, 2007) that confirm its excellence for scientific computing. Its large community and available libraries for data manipulation and machine learning come in handy, so one does not need to build applications from scratch.

2.3 HTTP Requests and APIs

Hypertext Transfer Protocol (HTTP) is an internet protocol that defines a set of request methods for exchanging information (CONTRIBUTORS, 2021). Some services create an Application Programming Interface (API) that works as an abstraction layer to access these services. Clients then request information to an API via HTTP or other protocols. Figure 2.2 depicts the information exchange between a client and an API where the API communicates with the service and retrieves the data or executes an action.

The most common methods of HTTP Requests are: GET, POST, PUT, DELETE (CONTRIBUTORS, 2021). The only one we use in this work is GET. The GET method allows us to retrieve data from APIs. In Chapter 3, we explain the scripts built for gathering data using GET. The API's response can be received in JavaScript Object Notation (JSON) format. This response contains a request status (e.g., failure or success) and the requested data.

Figure 2.2: Information exchange using HTTP request via API



Source: The Authors

2.4 NumPy and Pandas

"NumPy is the fundamental package for scientific computing in Python" (HARRIS et al., 2020). NumPy is a community-developed, open-source library for Python. It combines the features of two older libraries: Numeric and Numarray. It provides multidimensional Python arrays object and array-aware functions to operate on them.

Pandas is a Python library that offers data analysis tools. It has been developed since 2008 to close the gap between Python and statistical computing platforms or database languages. It provides "dataframes" (objects to handle data) that translate Comma-separated values (CSV) files to manageable python objects.(MCKINNEY, 2010)

Both libraries are high-value tools for data analysis applications (STANčIN; JOVIć, 2019). In this work, we take advantage of their data structures and features for organizing, visualizing, filtering, and preparing our data. More about how we used these tools will be disclosed in Chapter 3.

2.5 Machine Learning

"Machine learning is about extracting knowledge from data" (GUIDO; MUELLER, 2016). Machine learning applications can learn and improve from data without being explicitly programmed. An example of a commercial application is the machine learning algorithm that recognizes people's faces on *Facebook* photos (TAIGMAN et al., 2014)

In this work, we explore the field of supervised machine learning algorithms. A supervised algorithm learns with labeled data to predict the output of unlabelled ones (GUIDO; MUELLER, 2016). Take an algorithm that learns with a dataset composed of hotdogs and sandwiches images. Each one is labeled "hotdog" or "sandwich". After the learning process, the algorithm could classify a new image as one of these two options.

2.5.1 Neural Networks

For Neural Networks and Tensorflow, we based ourselves on the book "Learn Keras for Deep Neural Networks" (MOOLAYIL, 2018).

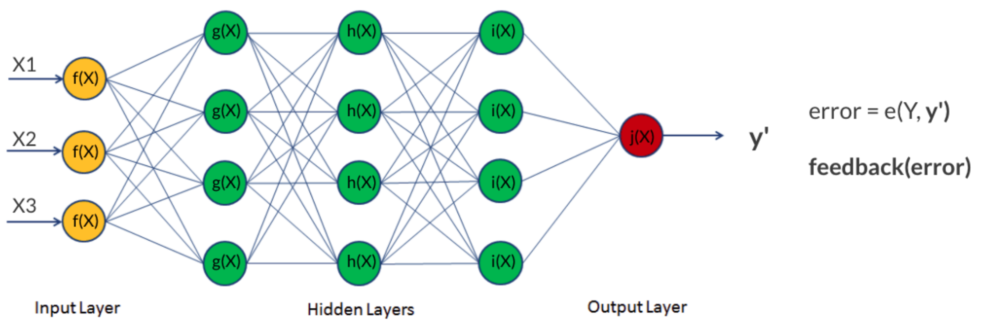
Neural networks are a field inside machine learning inspired by the brain's biological process. Neurons emit an electrical signal in the brain when stimulated by an input. In machine learning, multiple neurons compose layers that connect to create a structure capable of learning new patterns. The first layer receives all the input information, each middle layer processes the input from the previous one, and the final layer outputs a prediction. In each step, every neuron transforms the data received and sends it to every neuron of the next layer.

A learning algorithm compares the net prediction (output of the last layer) with the one observed in reality (the label). It then updates how the neurons transform data to minimize the difference between these two values. We call the process of adjusting the neurons, through the use of comparisons and feedback, *training* (GUIDO; MUELLER, 2016).

Figure 2.3 depicts a representation of a neural network with an input layer, three hidden layers, and an output layer. The information X1, X2, and X3 are inserted in the network that processes it and outputs the prediction y'. That prediction is compared with the actual label Y to calculate the error and update the model.

We validate the performance of a trained neural network by looking at the error of its predictions. Therefore, we need to decide what we define as an error. Several dif-

Figure 2.3: Example of a feedforward Deep Neural Network.



Source: The Authors

ferent metrics are available to estimate that. Some used for regression problems, such as *Absolute Error* and *Root Mean Square Error* (ANTONANZAS et al., 2016), and others more used for classification problems as *Accuracy* and *True Positives* (GOODFELLOW; BENGIO; COURVILLE, 2016). Table 2.1 details how we can calculate these examples. Choosing an appropriate metric is essential for network learning, and it is highly dependent on understanding the problem we are trying to solve with Neural Networks.

Table 2.1: Four different metrics for Error calculation.

Mean absolute error	MAE = $\frac{1}{n}\sum_{t=1}^{n}\lvert(Yobserved - Ypredicted)\rvert$
Root Mean Squared Error	RMSE = $\sqrt{\frac{1}{n}\sum_{t=1}^{n}(Yobserved - Ypredicted)^2}$
Accuracy	Accuracy = $\frac{TruePositives+TrueNegatives}{TruePositives+TrueNegative+FalsePositives+FalseNegatives}$
Precision	Precision = $\frac{TruePositives}{TruePositives+FalsePositives}$

There are two main types of neural network supervised problems: classification and regression (GUIDO; MUELLER, 2016). On the one hand, classification problems try predicting a discrete class label. If we use an image as input of our neural network and expect the output to be Dog or Cat, we have a classification problem. On the other hand, regression tries to predict a continuous quantity, like the amount of gas needed to go from one place to another given parameters, such as the car, road condition, and traffic.

Besides choosing the right error metric, a neural network needs a proper configuration to achieve optimal performance. We call the networks' setting parameters *Hyper-parameters*. Some examples of hyper-parameters are:

- **The number of Layers and Neurons:** The number of layers and neurons in each layer symbolizes the number of mathematical operations a neural network model will do. While the accuracy is directly related to the number of layers/neurons, so is the computation time.
- **Activation Functions:** It is the function each neuron uses to transform its input into output. Most famous are Sigmoid, Tanh, and Relu.
- **Learning Rate:** It represents the magnitude that the learning algorithm updates the neural network's weights (the activation functions variables). A low learning rate may delay the neural network convergence, while a big one may impossibility its convergence.
- **Batch Size:** The amount of data the learning algorithm analyses before updating parameters considering the feedback (e.g., 2, 4, 8, 16, 32, 64).
- **Epochs:** The number of times the training will run through all the data. The neural network performance improves with analyzing many times the same data, but it can cause overfitting.
- **Layer Dropout:** It is a method to avoid overfitting by canceling some neurons. Dropping out some neurons of an layer brings generalization capabilities into the neural network model.

2.5.2 Tensorflow and Keras

Tensorflow is a set of libraries for machine learning applications managed by Google (ABADI et al., 2015). It includes other libraries, like Keras (MOOLAYIL, 2018), and makes building learning models easier. Besides reducing coding time, these libraries implement parallelism and optimize their execution for the processor architecture running it.

Apart from the data, Tensorflow supports all the steps that involve Neural Networks in this work. It offers the tools to build the model structure, train it, validate it, and save it. In Chapter 4, some functions like *fit* and *evaluate* are examples of Tensorflow components.

2.5.3 Hyperas

Adjusting hyper-parameters to the data the network receives is called *tuning* (GUIDO; MUELLER, 2016). There are manual ways to choose these parameters, but it requires experience and judgment from the developer. To minimize the effort of manual searching for the optimal neural network parameters, Hyperas (PUMPERLA, 2020) has the objective of selecting the most fitting hyper-parameters for a given configuration and data. Hyperas is a Python library that works along with TensorFlow. It allows us to compare different setups of neural networks and select better values for their number of layers, activation functions, dropout percentages, batch sizes, and optimizer functions. Later, in Chapter 4, we describe how different data result in distinct neural network configurations, and Hyperas helps us with that.

2.6 Related Works

Adel Mellit et al. (MELLIT et al., 2020) classifies PV system energy forecasting methods based on four categories of *forcast horizons*: *very short-term*, *short-term*, *medium-term*, and *long-term*. The difference between them is the range of prediction. While *very short-term* ranges from a few seconds to some minutes, *short-term* predicts up to 72h, *medium-term* to weeks ahead, and the time magnitude of *long-term* goes from months to years. Most of the authors work with a day-ahead (24h ahead) forecast (ANTONANZAS et al., 2016), which stands between the *very short-term* and *short-term forecast horizons*.

The review authors also distinguish related works by data parameters and technology used. Regarding data parameters, they differentiate data with only PV energy records, data only with historical weather information, and hybrid datasets with both contents. Furthermore, they also distinguish datasets composed of one single location and those consisting of several (forming a regional dataset). Concerning technology, they list three traditional techniques of working with PV energy: physical methods, statistical/probabilistic approaches, and artificial intelligence. Inside the latest technique is machine learning, the focus of our research.

In their review, Antonanzas et al. (ANTONANZAS et al., 2016) explain that while the *forecast horizon* is the amount of time between the present and predictions, the *forecast resolution* describes the data time-frequency. Most of their referenced works used datasets with one hour of frequency between measurements. Only two works (LONG; ZHANG; SU, 2014) (OUDJANA; HELLAL; MAHAMMED, 2013) explore data containing energy generation in a daily frequency. The review also complements the difference in predicting regional PV energy generation depending on the available information. They concluded that the best scenario is when knowledge of power generation from all PV systems is known. Then, models can execute a bottom-up strategy (summing individual predictions) to arrive at a regional forecast.

The best way of classifying our work into these categories would be:

- **Time Horizon:** Day-ahead forecast;
- **Time Resolution:** 24-hour frequency;
- **Data type:** Hybrid and Regional;
- **Method:** feedforward Deep Neural Networks.

There are many methods for predicting solar energy, so finding related work that precisely matches our approach is not trivial. We did find works that share some of our approach characteristics. Son et al. (SON et al., 2018) evaluated the performance of a feedforward Deep Neural network using hybrid data. Their approach aimed to reduce the necessity of weather sensors on-site for learning experiments. Therefore, they used data composed of their PV System on-site measurements and weather forecast from a national weather service. They demonstrate that the trained model can achieve comparable or even slight results to their previous system that relies on the on-site sensor.

Long et al. (LONG; ZHANG; SU, 2014) used a *forecast resolution* of 24 hours to train an artificial neural network and other data-driven approaches. They collected data from on-site sensors of a single rooftop PV System located on the Coloane island of Macau and ran parameter analyses to select their models' inputs. Evaluating the test dataset, their artificial neural network model achieved a *Mean Absolute Percentage Error* of 11.878%.

Pierro et al. (PIERRO et al., 2017) used satellite and weather data to build models capable of predicting regional solar energy output. They divided the studied location into clusters and upscaled the prediction on these to arrive at a region's total energy generation. They concluded that each cluster behaved like a single PV System, and accuracy improved due to smoothing effects. They achieved RMSE and MAE of 7% and 4% for day-ahead prediction.

Besides the related works similar to ours, new research with more sophisticated machine learning techniques achieves exciting results. Wang et al. (WANG; QI; LIU, 2019) reviewed the difference between models that implemented Conventional Neural Networks (CNN), Long Short Term Memory (LSTM) neural networks, and a hybrid model that combined both. They show that the accuracy of the three models highly depends on the size of the available data. They achieved 2.2% of the Mean Absolute Percentage Error in their best-case scenario.

2.7 Summary

This chapter presented the foundations needed to comprehend our work. We start by explaining the process of the main subject of our study, the Photovoltaic System. Then, we explain some tools used in the conception of this work, such as Python, Pandas, Numpy, API Requests, TensorFlow, and Hyperas. In addition, we show the main concepts of neural networks, explaining how they work, their configuration parameters, and the typical problems we can use them to solve.

We establish how our approach stands in this field categories identified by review authors. We define it as a deep neural network (method) using hybrid and distributed data with a frequency of 24 hours to predict the day-ahead forecast. Then, we summarize three similar works (SON et al., 2018) (LONG; ZHANG; SU, 2014) (PIERRO et al., 2017) that share some features with ours. The first one uses feedforward deep neural networks collecting part of the data from a weather service to predict a single PV energy generation. The second uses a *forecast resolution* of 24 hours and artificial neural networks to solve the same problem. The last creates a regional model using satellite data and an up-scaling model. Furthermore, we also review a research (WANG; QI; LIU, 2019) where the authors explore the problem with more sophisticated neural network methods.

3 DISTRIBUTED DATA

This chapter describes our data parameters and the method used to collect all the information needed for training a machine learning model. The usage of distributed open data is a fundamental aspect of this work, and we desire to enable Smart Grid operators to gather meaningful data for their specific scenarios. Considering that, we will elaborate on completing the collection processes. We start by introducing our data sources. Then, we describe the scripts we developed to obtain the data and the final datasets' features and labels.

3.1 Data Sources

Concerning the PV systems' characteristics and past energy generation records, we collected all data from a community of solar panel owners named *PVoutput* (PVOUTPUT, 2020). In this community, users can register their PV Systems and log their daily energy generations. With numerous users contributing with high volume data from different places, this community is essential to evaluate this work proposal. Furthermore, users can also add any registered PV System to a favorite list on their platform. This feature makes it possible to create a list of pre-selected systems with locations close to each other. In Section 6, we are going to discuss the importance of that. Several filters are applicable by users to find data that serve their goals.

The weather information originates from a service named *World Weather Online* (WORLD. . . , 2020). This website contains historical climate conditions, real-time information, and forecasting. These different types of data are essential for our analysis.

3.2 Data Collection

We developed four different scripts to gather the data from these services. That was necessary because of the specifics of each API and data format. Besides requests to the websites, these scripts also filter data, eliminate corrupt entrances from the datasets, and combine different data sources to compose the final datasets. All of them are available on GitHub (FRANCO, 2020).

3.2.1 PV Data Collection

The first step in building the PV datasets is collecting a list of PV Systems we want to use for training our model. Our first script[1] starts by requesting this list from the PVoutput community using our API key and the GET HTTP method. This key allows us to access the API information and works as a reference to retrieve the systems we selected as favorites. The script then saves this list as a CSV file, creating a file that contains all characteristics of the to-be-studied PV systems.

For each system, the script makes new requests. These requests pull the daily energy generation records from the installation of every PV system until the current day. Since the PVoutput API allows requesting only 150 days interval records, we had to adapt our script to request data until it retrieved all the logs. It saves the response of every request, creating, for each system, a CSV file with all records.

3.2.2 Weather Data Collection

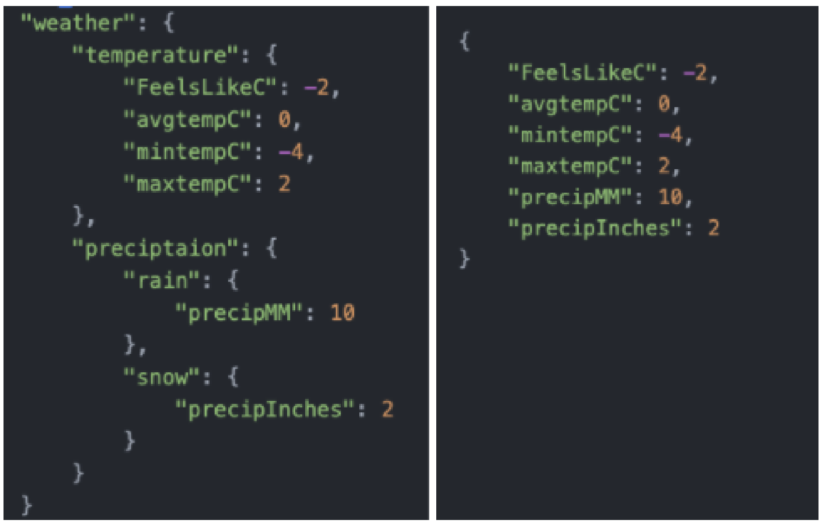
A second script[2] is responsible for requesting the data from the *World Weather Online* API. The script starts by accessing the saved information about the favorite PV Systems. It creates a dictionary for each system, containing id, installation date, latitude, and longitude information. Our algorithm uses this dictionary to construct the request's URL for the weather API.

Using GET HTTP, the script requests the historical weather information for the location of each system. Since the *World Weather Online* API also limits the data per request, we had to make multiple requests the same as we did for the system's daily output. It then returns all the weather conditions from the installation date until the current day. This API may not find the historical weather information from a pinpoint location as we requested. Instead, it will return the data from the nearby location it knows.

This API's response is a multilevel JSON, i.e., it is composed of objects that may have other objects inside them. Since we need a flat sheet as a dataset for machine learning, it may complicate our analysis. For that reason, we had to implement functions that flatten the responses. Figure 3.1 depicts an example of multilevel JSON and how it looks after being flattened.

[1]<https://github.com/lucio-lpf/SEPA/blob/main/data/pv_data.py>
[2]<https://github.com/lucio-lpf/SEPA/blob/main/data/weather_data.py>

Figure 3.1: Process of flattening a JSON response.



```
"weather": {
    "temperature": {
        "FeelsLikeC": -2,
        "avgtempC": 0,
        "mintempC": -4,
        "maxtempC": 2
    },
    "preciptaion": {
        "rain": {
            "precipMM": 10
        },
        "snow": {
            "precipInches": 2
        }
    }
}
```

```
{
    "FeelsLikeC": -2,
    "avgtempC": 0,
    "mintempC": -4,
    "maxtempC": 2,
    "precipMM": 10,
    "precipInches": 2
}
```

Source: The Authors

3.2.3 Forecasting Data

We created another script to retrieve forecasting information from the same weather service. This new script[3] requests the next five days forecasting every system on the favorites list. Then, it saves a list for each system with its characteristics and the forecasting. At the time, these datasets do not include a label. After the forecast days, we include it manually, knowing the actual energy generation.

3.3 Data Combination and Filtering

A fourth script[4] combines each energy generation record with its PV system's characteristics and the weather information on each given day. The outputs of these combinations are the neural networks' final datasets for training. In this script, we used Numpy to combine the different array of data and Pandas to select the features of our dataset and filter it.

We noticed by analyzing the data that some entries were incorrect as the PV System had not generated energy on some particular days. We also noticed that the entrances had a data feature, *Condition*, set to *Not Sure* when that happened, diverging from the

[3]<https://github.com/lucio-lpf/SEPA/blob/main/data/forecasting_data.py>
[4]<https://github.com/lucio-lpf/SEPA/blob/main/data/join_data.py>

typical cases *Sunny*, *Rainy*, and *Cloudy*. After combining the different datasets, our script filters them by dropping these corrupted entries.

The script finishes dividing the data into four datasets—one for each year's season. Mohammad et al. (Hossain; Mahmood, 2020) divided data to improve pattern recognition. That increase happens because the climate difference between seasons makes neural network pattern recognition difficult. In Chapter 5, we demonstrate the different learning performances of dividing the data or not.

3.4 Data Aspects

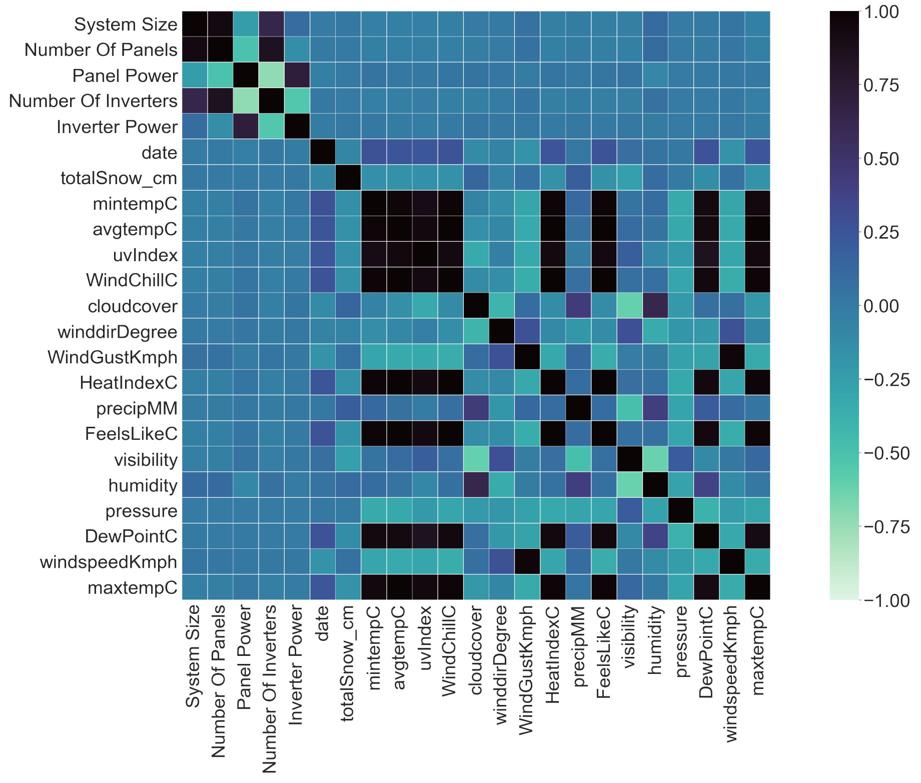
We classify the collected data into two categories: prediction target and features. Our prediction target is energy generation logs, and our features are the systems' characteristics and weather information. Table 3.1 shows a dataset sample from one of our collections. In total, our dataset contains 23 features.

3.4.1 Dataset Features

From the PVoutput community, we retrieved the following PV system's characteristics: *System Size, Number Of Panels, Panel Power, Number Of Inverters, Inverter Power*. Moreover, we also obtained the records that relate the energy generation to a specific date. The weather service API delivers the following information: *date, totalSnow_cm, mintempC, avgtempC, uvIndex, WindChillC, cloudcover, winddirDegree, WindGustKmph, HeatIndexC, precipMM, FeelsLikeC, visibility, humidity, pressure, DewPointC, windspeedKmph, maxtempC*.

We did not remove any feature based on our presumptions of importance. Instead, we let the neural network balance their weights on the predictions. The only filtering we did was to discard the categorical features, such as *Condition* because we collected more precised weather information. Then, all the parameters of our data are numerical values and represented in *float64*. Figure 3.2 depicts a heat-map with the correlation of our dataset features on a scale that ranges from -1 to 1. While a value of 1 means two features have a proportional relationship, -1 represents an inverse correlation. Correlations close to 0 symbolize a non-linearity between two features.

Figure 3.2: Heat-map with features relation.



Source: The Authors

3.4.2 Learning Label

Our prediction target is *Efficiency* (Wh/W), i.e., the amount of energy a system produces (Wh), divided by its size (W).

$$Efficiency = \frac{EnergyProduced}{SystemSize} \tag{3.1}$$

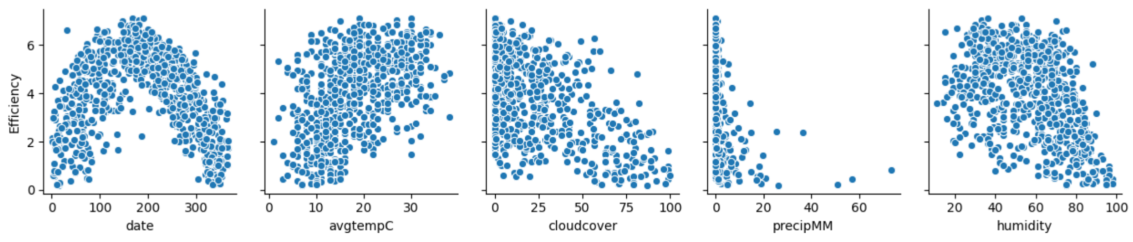
It represents the number of hours a PV System would generate power in optimal capacity. This variable makes it easier to train our model with different systems because it softens the range of energy generation using each system size. Therefore, it enables gathering a larger dataset (not filtering per size) and generalizing our results to other systems.

A study made at the University of Massachusetts Amherst (Sharma et al., 2011) shows a strong correlation between solar intensity and parameters, such as the day of the year, temperature, sky cover, precipitation, and humidity. In Figure 3.3, we can notice organized patterns representing a significant correlation between our chosen label and these parameters.

Table 3.1: Dataset aspects sample.

Feature	**Mean**	**Min**	**Max**
Efficiency	3.187760	0.0	7.552
System Size	10085.003378	3750	38190
Number Of Panels	40.540875	15.0	133
Panel Power	268.553224	185.0	360
Number Of Inverters	27.685534	1.0	133
Inverter Power	3694.008520	195.0	11400
day of the year	180.099880	1.0	366
totalSnow_cm	0.135677	0.0	45.100
mintempC	10.277382	-19.0	29
avgtempC	15.501389	-14.0	37
uvIndex	4.021695	1.0	9
WindChillC	13.145747	-23.0	34
cloudcover	40.623977	0.0	100
winddirDegree	208.379964	7.0	348
WindGustKmph	15.563434	3.0	55
HeatIndexC	14.953082	-15.0	39
precipMM	4.603667	0.0	172.400
FeelsLikeC	13.699159	-23.0	39
visibility	9.101381	0.0	10
humidity	70.152316	27.0	100
pressure	1017.818069	985.0	1042
DewPointC	8.366827	-24.0	25
windspeedKmph	10.354778	2.0	39
maxtempC	18.783425	-11.0	41

Figure 3.3: Relation between key features and the chosen label.



Source: The Authors

3.5 Summary

This chapter introduces the two sources we used in this work to collect distributed data for training our neural network models. We collected PV System energy generation records from a community of solar panel owners (PVOUTPUT, 2020) and historical/-forecasting data from a weather service (WORLD. . . , 2020). We detail how we used the PVOutput platform to select data from the to-be-studied locations. We explain our scripts' steps to request data from the sources' API and combine the different data formats into five datasets (one for each year's season and one with undivided data).

We end the chapter by listing and analyzing our datasets' aspects. We explain the meaning of our chosen label, *Efficiency*, and relate it to the energy produced by a PV System. We also show how our features correlate between themselves using a heat map and how they relate to our label. To better illustrate the data we are working with, we display an example of a dataset created from the process of data collection.

4 NEURAL NETWORK WITH CUSTOM CONFIGURATION

To investigate the research questions of this work, we implemented scripts for analyzing our data using one of the *Tensorflow* (ABADI et al., 2015) neural network libraries: *Keras* (CHOLLET et al., 2015). Furthermore, we also use other libraries, such as *Numpy, Pandas, Seaborn*, and *Matplotlib* to transform the data we collected and display the neural network results. This chapter explains how our learning algorithms work and the motivations behind their proprieties.

4.1 Data Normalization

Table 3.1 shows that our dataset comprises features with different units and ranges. These differences alone could cause the neural network to emphasize some features over others. To minimize that, the first step in the training of a neural network model is the standardization of the dataset. We chose a z-score normalization represented by Equation 4.1 where: *Z_score* is the normalized value, *O_value* is the data original value, *F_mean* is the feature mean, and *F_std* is the standard deviation. It divides the features by each range's standard deviation, containing 99% of data after three deviations. Then, all of our features fall within similar ranges, and we work with equivalent metrics (KREYSZIG, 2010).

$$Z\_score = \frac{O\_value - F\_mean}{F\_std} \tag{4.1}$$

To apply the Equation 4.1 to our dataset, we had to remove the columns that contained only constants and would create an error in the script. For example, assume that the weather data has a feature describing the snow cover, but our data comes from a tropical place. This column would be filled only by zeroes, and the division would result in *not a number* (NaN).

4.2 Hyperas

After preparing the data, the next step is configuring the neural network. Since we propose an approach that different users can use to validate their scenarios, an immutable configuration would not work. As discussed in Chapter 2, Hyperas (PUMPERLA, 2020) is a library built to find the optimal hyper-parameters for a given dataset. Using this library, we observe, in Chapter 5, that datasets from different places result in distinct configurations of neural networks.

Hyperas works along with Keras. It analyzes multiple setups of neural networks and returns the one with the best performance. Developers can establish *choices* to be evaluated by Hyperas. Alg. 4.1 shows the difference between a simple Keras algorithm and one that implements Hyperas *choices*. We implemented a script [1] that uses Hyperas to select the configuration of our models depending on the data we are input it. Table 4.1 describes the hyper-parameter we used Hyperas to find and the choices it should analyze for each one.

Table 4.1: Hyper-parameters and the evaluated choices.

Parameter	**Choices**
Number of Hidden Layers	2, 3, 4, 5
Number of neurons per Layer	128, 256, 512, 1024
Layer Dropout Percentage	0 - 1
Layer Activation Function	relu, sigmoid, tanh, linear
Optimizer	rmsprop, ADAM, sgd
Training Batch Size	4, 16, 32, 64, 128

Algorithm 4.1: Example of Hyperas usage.

```
# PLAIN KERAS
model_layers.append(layers.Dense(256, activation='sigmoid'))
model_layers.append(layers.Dropout(0.14325834324324327))

# KERAS WITH HYPERAS
 model_layers.append(layers.Dense({{choice([128, 256, 512, 1024])}},
                                  activation={{choice(['relu',
                                                       'sigmoid',
                                                       'linear'])}}))
model_layers.append(layers.Dropout({{uniform(0, 1)}}))
```

4.3 Neural Network Training

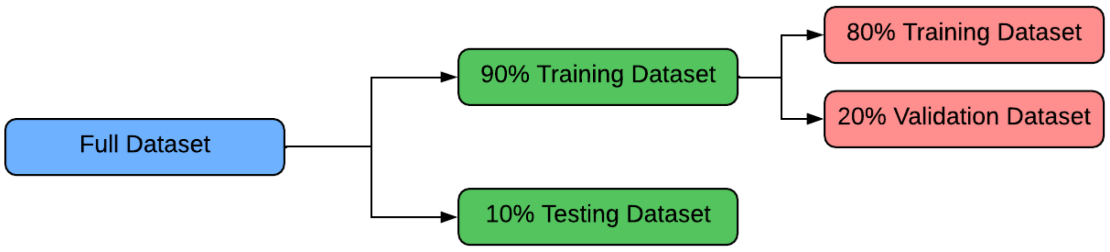
The script responsible for creating the neural network model[2] starts by normalizing the dataset and dividing it into 90% for training and 10% for testing. This division

[1]<https://github.com/lucio-lpf/SEPA/blob/main/hyperas_select.py>
[2]<https://github.com/lucio-lpf/SEPA/blob/main/neural3.py>

is critical for testing a trained model and guaranteeing that it is not overfitting our data. Overfitting happens when a model gets too used to the training dataset, predicts it precisely, but can not predict new data it has never seen before (MOOLAYIL, 2018). Then, our script builds a neural network model using an input shape equal to our data and the hyper-parameters chosen by Hyperas.

After building the model, our script calls the function that executes training iterations. Keras function *fit* trains our model using the normalized dataset and its labels. It also receives some parameters: training batch size, number of epochs, and the percentage of data designated for validation purposes. An early stop function uses this validation data to monitor the training performance indicator, e.g., RMSE or MAE. It forces the training to stop when this indicator stops improving. Figure 4.1 depicts all the data divisions during the execution of this script.

Figure 4.1: Data designations for training, testing and validations.

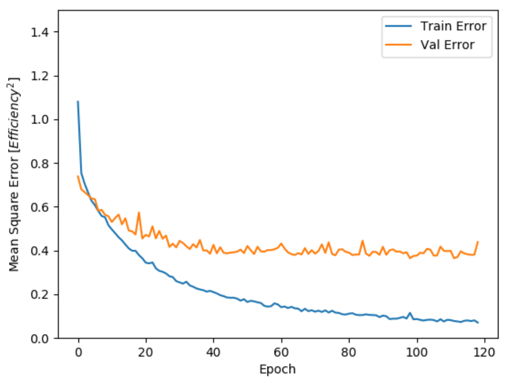


Source: The Authors

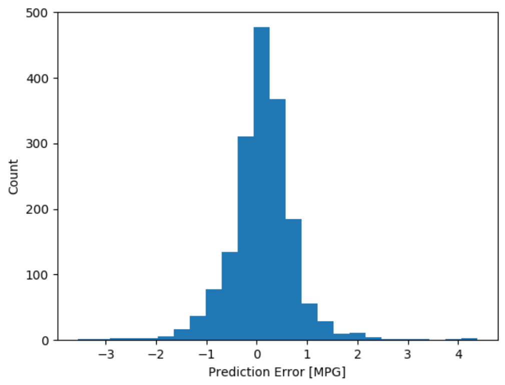
Finally, our script tests the trained model with the Keras function *evaluate*. It uses 10% of the original dataset separated for testing. If the test results look similar to the training, our model did not overfit our data, and we can save it for future predictions.

We compiled some of our training and testing information into plots to analyze our model performance. Our script produces the following plots every time it trains and evaluates a neural network: true values x predicted values, prediction errors, and loss progression over epochs. Figure 4.2 depicts examples of these plots. The loss progression plot, shown in Figure 4.2a, is important to investigate if a model is not overfitting the dataset. It depicts the training performance along the epochs. Figure 4.2b depicts the errors distribution of our model that resembles a Gaussian curve. In Figure 4.2c, the blue line places the 1:1 proportion between the axis. The closer the dots are from that line, the better the neural network predicted the testing dataset.

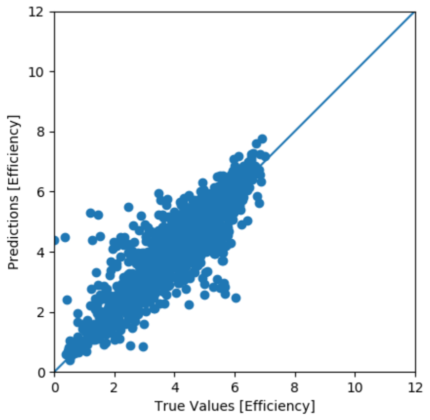
Figure 4.2: Plots describing training performance.



(a) Loss Progression



(b) Prediction Errors



(c) True Values x Predicted

Source: The Authors

4.4 Prediction Features

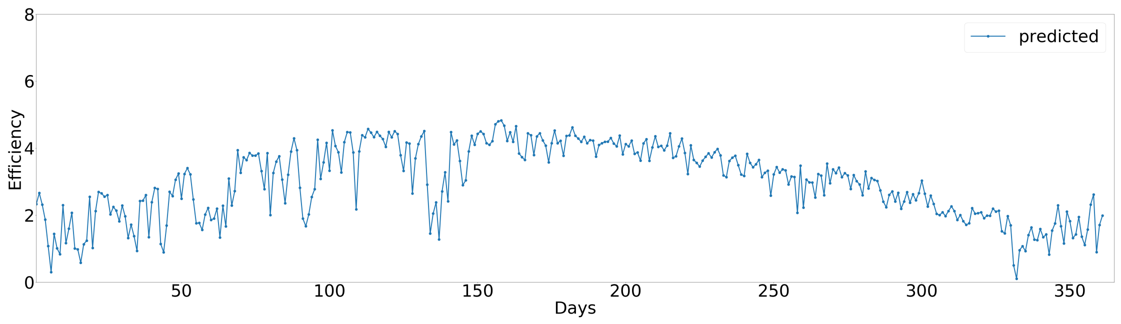
We created two more scripts[3][4] to test our two research questions in Section 1.1. Both of them use a saved neural network model from the training step. For both cases, we used the Keras function *predict*, which receives data features and outputs our chosen label.

4.4.1 Impact Analysis

By training our neural network models with data from a determined location, we can evaluate the impact of a new PV System near that place. We created a dataset with a

[3]<https://github.com/lucio-lpf/SEPA/blob/main/new_system.py>
[4]<https://github.com/lucio-lpf/SEPA/blob/main/predictions.py>

Figure 4.3: Evaluation of a new system in a known location.



Source: The Authors

new PV System's configuration and a year of weather information from one of the nearby systems used in the training process. It then uses a models trained with undivided data to predict the energy the new system would generate that year. Figure 4.3 depicts an example of our script output plot. The energy fluctuation in the plot marks the different seasons of the year. The climate information came from the northern hemisphere, where the highest energy generation is in the middle of the year. We hope to enable better planning for building a PV System with this feature. Smart Grid operators can evaluate the impact of a new system in a region, and customers can check what configuration satisfies their needs.

4.4.2 Day-ahead Forecasting

Our second script uses a weather forecasting dataset to predict the day-ahead energy generation of all systems from a location. It predicts each system's next day *Efficiency* and multiplies these values by the correspondent system size to arrive at the total energy generation. The script outputs a value representing the regional generation for a day ahead by combining all systems' expected energy generations. We created this feature hoping that energy suppliers can plan for energy fluctuations instead of reacting to them.

4.5 Summary

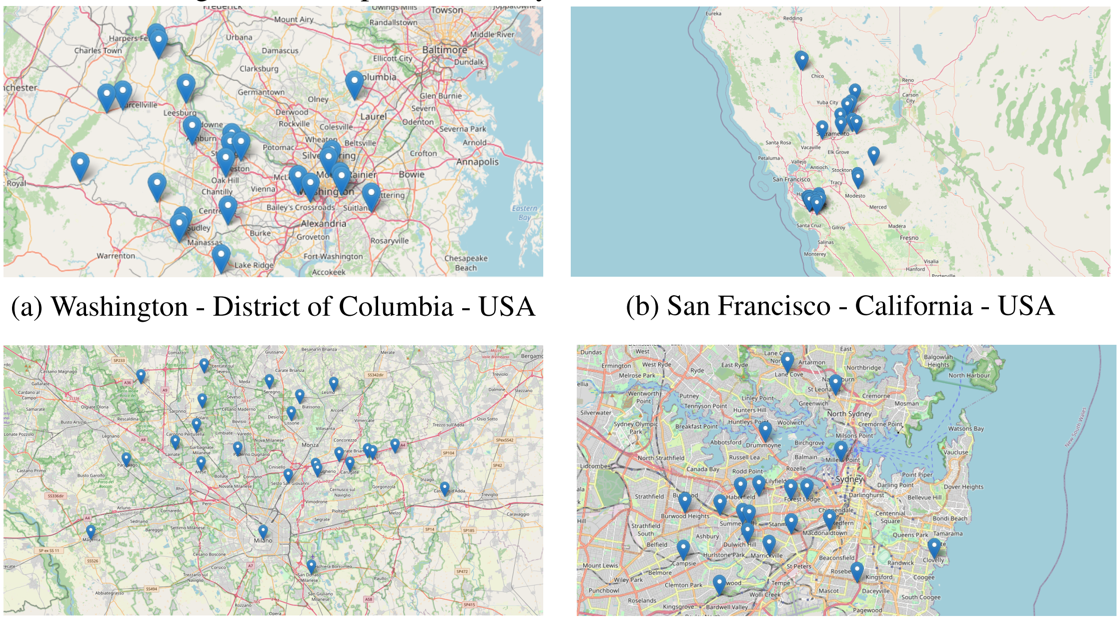
This section details our process of creating, training, and testing neural network models using a Tensorflow (ABADI et al., 2015) library called Keras(CHOLLET et al., 2015). It describes how we normalized our dataset using a standard deviation strategy and how we had to drop features depending on the studied location. It follows to explain how we used Hyperas (PUMPERLA, 2020) to ease the hyper-parameters selection. Hyperas evaluates our datasets and chooses the best options for the number of layers, number of neurons per layer, layer dropout percentage, layer activation functions, learning optimizer, and training batch size. Then, the chapter detail the steps our script take to train neural network models and the plots produced by each training.

This chapter also describes the prediction features we developed to answer our research questions in Section 1.1: day-ahead forecasting and impact analysis. The first uses weather data from a location to study how a system would behave there. The second uses forecasting information to predict the energy generated by all systems at a location. It then combines the measures to arrive at a regional day-ahead prediction.

5 EVALUATION

To evaluate our approach, we selected distributed data from four different locations. Each location resulted in five different datasets (one for each season and one for all data combined). Figure 5.1 depicts the locations we chose to evaluate our approach, where each pinpoint symbolizes one PV System. Our datasets comprise a total of 175074 entries from 122 PV Systems. Table 5.1 shows precisely the number of systems and the amount of data from each location.

Figure 5.1: Pinpoint of PV Systems in the selected locations.



(a) Washington - District of Columbia - USA

(b) San Francisco - California - USA

(c) Milan - Lombardy - Italy

(d) Sydney - New South Wales - AUS

Source: The Authors

Table 5.1: Number of PV Systems and collected data from each location.

Location	**Number of Systems**	**Total Entries**
A	35	33001
B	28	36385
C	25	48876
D	34	56812

5.1 Neural Network Models Configurations

We used Hyperas to evaluate each dataset and find their neural network's hyper-parameters. Table 5.2 shows each location's chosen neural network settings. In addition to the layers described in that table, all models contain a final layer composed of one neuron that outputs our regression result.

Table 5.2: Chosen hyper-parameters for each dataset using undivided data.

Location	Optimzer	Batch Size	Layer	n Neurons	Droptout	Activation Func.
A	ADAM	16	1	256	0.0723	Relu
			2	1024	0.1548	Relu
			3	512	0.4480	Linear
B	ADAM	8	1	1024	0.0523	Tanh
			2	256	0.0030	Sigmoid
			3	512	0.1045	Relu
C	RMSprop	32	1	1024	0.317	Relu
			2	128	0.3063	Tanh
			3	256	0.0005	Tanh
D	RMSprop	32	1	1024	0.052	Tanh
			2	256	0.03015	Sigmoid
			3	128	0.1545	Relu

The Early Stop function we implemented monitors the validation loss (MSE) and has a *patience* of 10. This patience means our function will look for improvements in the validation loss until it remains 10 epochs without improving, then it stops the training. We set the training epochs to 200, meaning our learning algorithm will finish after going through the entire training dataset 200 times or being interrupted by the Early Stop. Most models did not complete all epochs.

Regarding metrics, we chose to evaluate our models using *Mean Absolute Error* (MAE), *Root Mean Square Error* (RMSE), and *Mean Absolute Percentage Error* (MAPE) (ANTONANZAS et al., 2016). While the MAE and RMSE unit is *Efficiency* (Wh/W), MAPE is a percentage value. Their formulas are represented by equations 5.1, 5.2, and 5.3, respectively. Equation 5.4 represents the dimensionless weighted percentage of MAE. We chose to add this metric because MAPE can be misleading when dealing with values close to zero (KIM; KIM, 2016).

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|(Y_{observed} - Y_{predicted})| \tag{5.1}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(Y_{observed} - Y_{predicted})^2} \tag{5.2}$$

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}|\frac{(Y_{observed} - Y_{predicted})}{Y_{observed}}| \tag{5.3}$$

$$MAE\% = 100\% \times \frac{MAE}{Mean} \tag{5.4}$$

5.2 Training Results

We trained 20 neural network models to evaluate all datasets. Table 5.3 compares the testing results of the neural network models of each location. The table's values represent results from data we did not input into the models for training. We included an evaluation of the undivided datasets to analyze the influence of this strategy. The best performance location and season was San Francisco in the summer. Figure 5.2 depicts the three plots that our script outputs after successfully training our best performing model. All other plots are available in Appendix A.1.

5.3 Prediction Results

We selected the model that better performed in training to evaluate our proposed prediction features. We used the saved model that evaluated the undivided dataset for the impact analyses. We selected the model from the season the forecasting was from for the day-ahead prediction.

5.3.1 Impact Analyses

To investigate a new system's impact, we searched the PVOutput community and selected another system from Location B. This way, we could demonstrate this feature and evaluate it. While Figure 5.3 depicts the typical output of our script, Figure 5.4 depicts a

Table 5.3: Neural Network models performance when predicting test data.

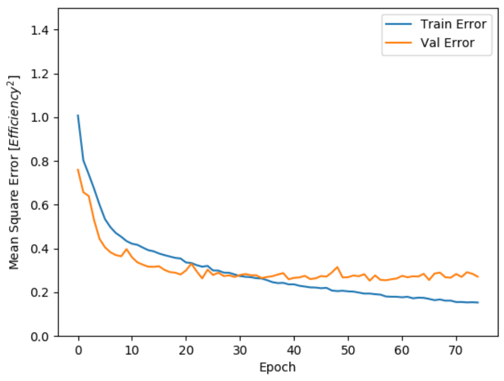
Location	Season	MAE	RSME	MAPE	MAE%
A	Summer	0.39	0.57	15.53%	9.46%
	Fall	0.31	0.46	21.59%	14.36%
	Winter	0.35	0.55	27.51%	15.34%
	Spring	0.38	0.55	12.08%	9.41%
	Undivided	0.48	0.66	27.08%	14.90%
B	Summer	0.34	0.52	9.76%	6.87%
	Fall	0.37	0.58	30.72%	13%
	Winter	0.49	0.68	30.34%	17.52%
	Spring	0.47	0.67	14.32%	9.49%
	Undivided	0.47	0.66	21.98%	12%
C	Summer	0.47	0.71	16.73%	10.85%
	Fall	0.37	0.54	35.33%	20.6%
	Winter	0.36	0.52	30.30%	17.4%
	Spring	0.52	0.75	19.57%	13.9%
	Undivided	0.46	0.67	30.23%	14,5%
D	Summer	0.59	0.86	22.91%	14.35%
	Fall	0.51	0.76	35.33%	16.75%
	Winter	0.40	0.64	21.14%	15.28%
	Spring	0.57	0.85	20.99%	13.90%
	Undivided	0.50	0.78	23.89%	14.44%

comparison between the actual *Efficiency* and our model predictions for the year 2021.

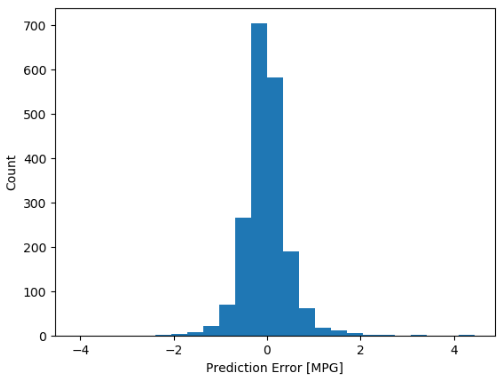
5.3.2 Day-ahead

Of the 28 Systems, we used data to build the Location B dataset, 24 are active and posting their daily energy generation. We used their data to test the regional day-ahead energy prediction. For that, we gathered the on-day-ahead weather forecasting for each system location. We waited for the actual next-day output and built a forecasting dataset. All the 24 PV Systems combined generated 1,040,185 Wh of energy. Our neural network model predicted these systems would generate 1,099,775 Wh that day, an error of 5.73%.

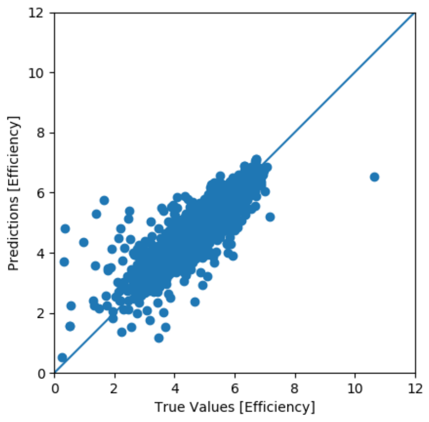
Figure 5.2: Result plots from Location B with summer dataset.



(a) Loss Progression



(b) Prediction Errors



(c) True Values x Predicted

Source: The Authors

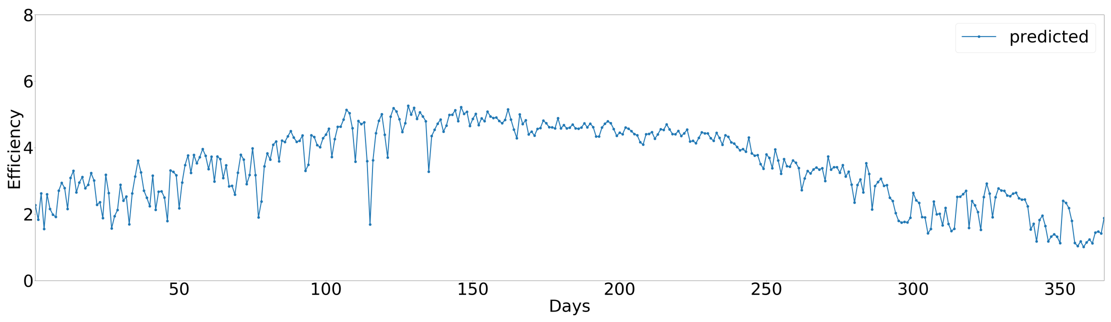
5.4 Discussion

Different causes may have affected the results of our models. The weather data that composes our datasets do not necessarily inform the climate conditions of the PV systems' positions. It searches for the nearest point with accessible data given the coordinates of each system. Also, the energy-generated data is uploaded by community users, making it impossible to confirm the measurements. Even after filtering, we could find some *bad* data that potentially harmed our training.

We noticed a strong correlation between the results and the location seasons during experimentation. While separating the data into four datasets helped improve the performance of seasons such as Summer and Spring, the performance of Winter and Fall usually went worse than the undivided test. Figure 5.5 and Figure 5.6 [1] depict that, in San

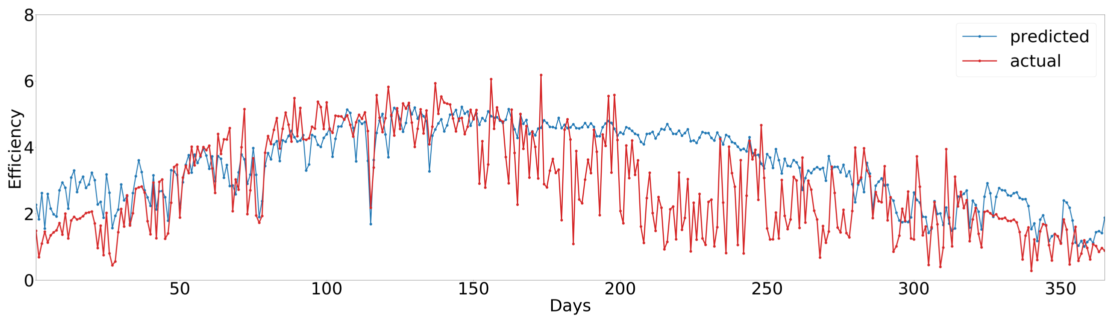
[1]Required reference: © WeatherSpark.com

Figure 5.3: Impact Analyses of new system at Location B.



Source: The Authors

Figure 5.4: Comparison between impact analyses and real system output for 2021.



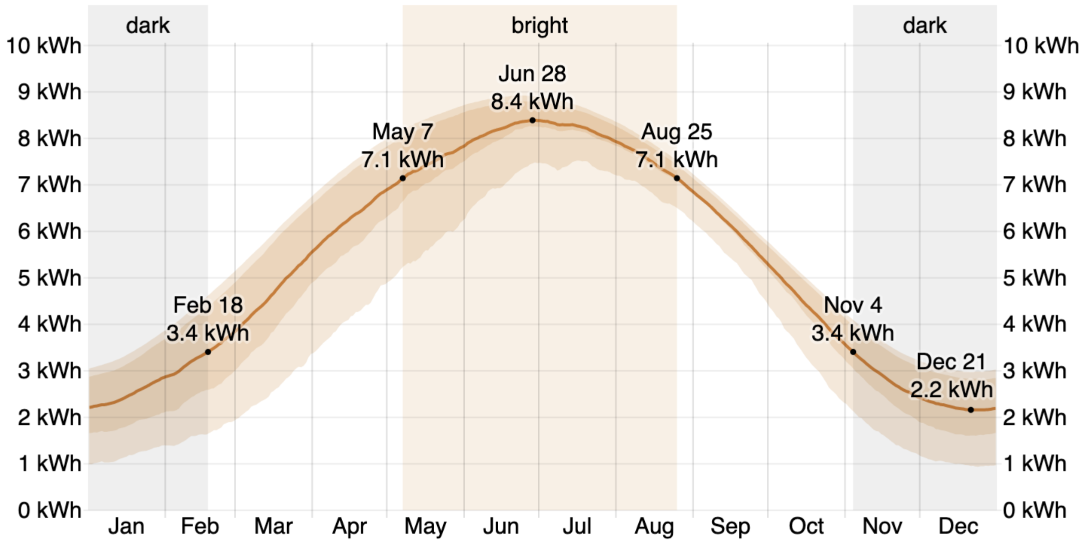
Source: The Authors

Francisco, the winter season is characterized by lower solar incidence and rainfalls. The first characteristic reduces the average energy generation, requiring a more precise prediction to lower losses. The second one brings more variations that increase the difficulty of a neural network model to recognize patterns. In contrast, when we analyze Location D, Sydney, we notice that all models performed similarly. We can relate that to a more even rain distribution throughout the year, as depicted in Figure 5.7.

Regarding the dynamic hyper-parameters approach, Hyperas chose a distinct configuration for each location. The training with the chosen configurations showed signs of overfitting depending on the season evaluated, e.g., as shown in Figure A.4. This overfitting reveals that even though Hyperas found suitable configurations, we still need to make fine adjustments to neural networks for the same location.

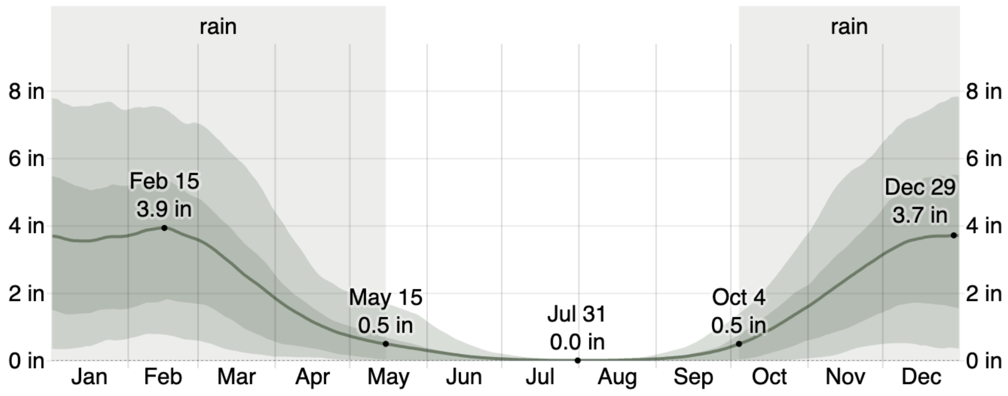
Analyzing the day-ahead feature, our model seems to perform better when predicting generation using forecasting. However, what happened was just an advantage of working with distributed PV Systems. As we observe in Figure 5.2b, while our model has overpredicted some energy generations, others it underpredicted. When we combine all to calculate the regional prediction, our worst-case scenario would be similar to the

Figure 5.5: Average daily incident shortwave solar energy in San Francisco.



Source: WeatherSpark

Figure 5.6: Average monthly rainfall in San Francisco.



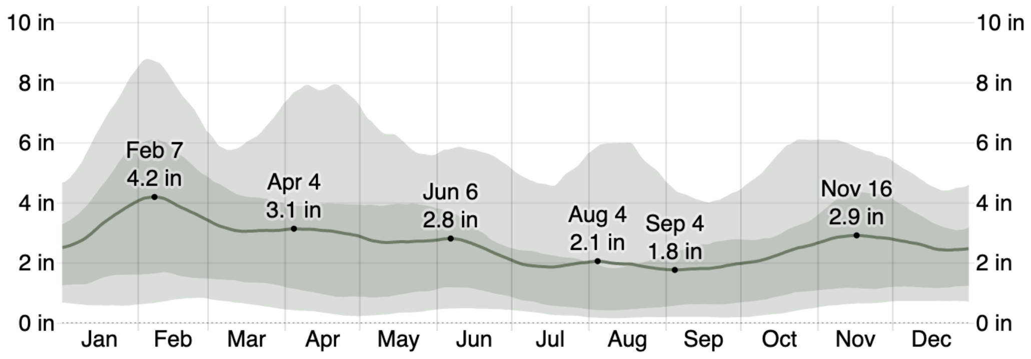
Source: WeatherSpark

training MAE, but these prediction errors may cancel themselves in the best case.

Most of the authors present their results in units such as power (W) or energy generation(Wh) (MELLIT et al., 2020) (ANTONANZAS et al., 2016), but that does not interfere when comparing our results with them. Equation 3.1 shows the direct relationship between our label and these units. Therefore, we can compare the percentage error values instead of absolute ones. When comparing our results to other works, the most significant complication is the different characteristics between our approach and others.

Son et al. (SON et al., 2018) used a similar machine learning method and also used weather data from outside their system's environment. The main difference between our approaches is that they used information from a single PV System instead of multiples and their data had a *time resolution* of one hour. Their model outperformed ours, achieving an

Figure 5.7: Average monthly rainfall in Sydney.



Source: WeatherSpark

MAE% of 2.9%.

Long et al. (LONG; ZHANG; SU, 2014) shared the same daily frequency for *time resolution*. Even though their artificial neural network model predicted data for a single location with on-site sensors, some of our models achieved similar results to their MAPE of 11.878%.

5.5 Next Steps

The PVOutput community (PVOUTPUT, 2020) has a significant number of systems enrolled. Although we selected PV systems that, at first sight, appeared to contain healthy data, further analyses into the community to separate proper systems for building a dataset are needed. In addition, we could employ feature selection in our data (LONG; ZHANG; SU, 2014). These investigations may result in the drop of some features that do not improve the prediction capabilities of their models.

Even though Hyperas eased our model hyper-parameter selection, we implemented a basic feedforward Deep Neural Network to analyze our data. We could try evaluating the same data with more sophisticated methods, such as Convolutional Neural Networks, Long Short Term Memory, or a hybrid combination (WANG; QI; LIU, 2019) (Hossain; Mahmood, 2020).

The process of data collection we created is still unstable and time-consuming. Some unexpected errors may complicate who tries to use them. E.g., a PV System information may not contain its installation date, or there are energy records previous to the installation date that will conflict when matching weather and energy data. An application in collaboration with the PV Output community to ease data collection would allow more research using their data without unexpected struggles.

Finally, despite our focus on documenting all our scripts' steps, they may still be unfriendly to some users. A user interface that integrates our scripts could lower our approach's complexity and learning curve.

5.6 Summary

This chapter shows the four locations we collected data to build our training datasets and specifies the amount and the number of systems. Then, the chapter details the model's hyper-parameters chosen by Hyperas (PUMPERLA, 2020) for each location, the early stop configuration of our training, and the metrics with which we evaluate our models. It displays the five datasets' testing results of every location (one for each year's season and one containing all data). Our best performance model achieved 6.87% of MAE% predicting summer energy generation. We used this model to demonstrate our two prediction features. It predicted the day-ahead energy generation with an error of 5.73%.

The chapter discusses our findings, assigning some possible limitations of our approach. It then analyzes the benefit of dividing the data into the season, identifying that the strategy showed better results when we collected data from places with rainy seasons. Furthermore, it compares our results with similar works discussing the main difference between our approach and the ones of other authors.

Finally, the chapter concludes by listing the possible next steps of our work. Further research can explore our approach using more sophisticated neural network methods. We also discuss the necessity of user interfaces to interact with our developed scripts and a better way of capturing data from the PVOutput community.

6 CONCLUSION

This work proposes an approach to predict regional solar energy generation using distributed data and neural networks. Our approach aims to reduce the risks imposed by solar energy fluctuations. We collected open data from distributed Photovoltaic Systems and historical weather to accomplish that. The main source that allowed our analysis was a community of solar panel owners named PVOutput. This work describes our steps to create deep neural network models with custom configurations for each data group and evaluate our collected data.

We developed scripts in Python using machine learning, data manipulation, and web request libraries. These scripts were responsible for collecting, filtering, and combining data from different sources. These scripts also created, configured, trained, and tested deep neural network models, to evaluate the obtained data.

We demonstrate the training results of our neural network models with data from four locations (Washington DC, San Francisco, Milan, Sydney). We concluded that our models performed unsatisfactorily when predicting energy generation for seasons with unstable weather. Besides the training evaluations, we also used our trained models to answer our research question. We created scripts that, using these models, predict day-ahead energy generation and impact analysis of new systems. We predicted the day-ahead energy generation with satisfactory accuracy using our best-performing model.

We identified possible limitations to our approach regarding our chosen learning method and the collected data. We set a feedforward deep neural network while recent studies use more advanced techniques such as LSTM and CNN. In addition, we can not confirm the measures each user uploads to the PVOutput community leading to possible misslabeled data. Furthermore, we used weather data from a service and not from on-site sensors. This service may not deliver data precisely from the point where PV Systems situate.

To continue this work, we propose evaluating more sophisticated machine learning methods or investing in data filters to select data that best correlate to our goals. Further investigation can help us better understand if the limitations of our approach are in our learning method or our data. Moving towards accessibility, we also believe in the benefits of user interfaces to collect data and train models using these data.

The most significant contribution of this work is the demonstration that it is possible to create models for regional solar energy prediction using open distributed data from a community of solar panel users. Furthermore, this work also shows how Hyperas can be used to reduce the friction of selecting hyper-parameters and discuss the benefit of separating data into seasons.

Part of this work was accepted for publication by the IEEE PES Innovative Smart Grid Technologies (ISGT NA 2022) under "Regional PV Energy Forecasting using Distributed Data and Deep Neural Networks."

REFERENCES

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <http://tensorflow.org/>.

ANTONANZAS, J. et al. Review of photovoltaic power forecasting. **Solar Energy**, v. 136, p. 78–111, 2016. ISSN 0038-092X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0038092X1630250X>.

BV, T. S. **TIOBE Index for February 2022**. 2022. <https://www.tiobe.com/tiobe-index/>. Accessed: 2022-02-14.

CHOLLET, F. et al. **Keras**. 2015. <https://keras.io>.

CONTRIBUTORS, M. **HTTP request methods**. 2021. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>. Accessed: 2022-02-14.

DIRAC, P. A. M. **Snapshot 2021**. [S.l.]: Clarendon Press, 2019. (International series of monographs on physics). ISBN 9780198520115.

DIRAC, P. A. M. **Trends in photovoltaic applications**. [S.l.]: Clarendon Press, 2019. (International series of monographs on physics). ISBN 9780198520115.

FRANCO, L. **Solar Energy Prediction Application.** 2020. <https://github.com/lucio-lpf/SEPA>. Accessed: 2022-04-24.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA, USA: MIT Press, 2016. <http://www.deeplearningbook.org>.

GUIDO, S.; MUELLER, A. C. **Introduction to machine learning with python: A guide for data scientists**. [S.l.]: O'Reilly Media, 2016. ISBN 9781449369415.

HANIFULKHAIR, K. b. et al. One day ahead prediction of pv power plant for energy management system using neural network. In: **2020 International Seminar on Intelligent Technology and Its Applications (ISITIA)**. [S.l.: s.n.], 2020. p. 107–112.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

Hossain, M. S.; Mahmood, H. Short-term photovoltaic power forecasting using an lstm neural network and synthetic weather forecast. **IEEE Access**, v. 8, p. 172524–172533, 2020.

KIM, S.; KIM, H. A new metric of absolute percentage error for intermittent demand forecasts. **International Journal of Forecasting**, v. 32, n. 3, p. 669–679, 2016. ISSN 0169-2070. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0169207016000121>.

KREYSZIG, E. **Advanced Engineering Mathematics**. John Wiley & Sons, 2010. 1011-1015 p. ISBN 9780470458365. Disponível em: <https://books.google.com.br/books?id=UnN8DpXI74EC>.

LONG, H.; ZHANG, Z.; SU, Y. Analysis of daily solar power prediction with data-driven approaches. **Applied Energy**, v. 126, p. 29–37, 2014. ISSN 0306-2619. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306261914003249>.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). **Proceedings of the 9th Python in Science Conference**. [S.l.: s.n.], 2010. p. 56 – 61.

MELLIT, A. et al. Advanced methods for photovoltaic output power forecasting: A review. **Applied Sciences**, v. 10, n. 2, 2020. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/10/2/487>.

MOOLAYIL, J. **Learn Keras for Deep Neural Networks: A Fast-Track Approach to Modern Deep Learning with Python**. 1st. ed. USA: Apress, 2018. ISBN 1484242394.

OBI, M.; BASS, R. Trends and challenges of grid-connected photovoltaic systems – a review. **Renewable and Sustainable Energy Reviews**, v. 58, p. 1082 – 1094, 2016. ISSN 1364-0321. Disponível em: <http://www.sciencedirect.com/science/article/pii/S136403211501672X>.

OLIPHANT, T. E. Python for scientific computing. **Computing in Science Engineering**, v. 9, n. 3, p. 10–20, 2007.

OUDJANA, S. H.; HELLAL, A.; MAHAMMED, I. H. Power forecasting of photovoltaic generation. **International Journal of Electrical and Computer Engineering**, World Academy of Science, Engineering and Technology, v. 7, n. 6, p. 627 – 631, 2013. ISSN eISSN: 1307-6892. Disponível em: <https://publications.waset.org/vol/78>.

PIERRO, M. et al. Data-driven upscaling methods for regional photovoltaic power estimation and forecast using satellite and numerical weather prediction data. **Solar Energy**, v. 158, p. 1026–1038, 2017. ISSN 0038-092X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0038092X17308617>.

PUMPERLA, M. **Keras + Hyperopt: A very simple wrapper for convenient hyperparameter optimization.** 2020. <https://github.com/maxpumperla/hyperas>. Accessed: 2022-04-24.

PVOUTPUT. 2020. <https://pvoutput.org/>. Accessed: 2022-04-24.

ROSSUM, G. van; BOER, J. de. Interactively testing remote servers using the python programming language. **CWI Quarterly**, v. 4, n. 4, p. 283–304, dez. 1991.

SAFI, S.; ZEROUAL, A.; HASSANI, M. Prediction of global daily solar radiation using higher order statistics. **Renewable Energy**, v. 27, n. 4, p. 647–666, 2002. ISSN 0960-1481. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0960148101001537>.

Sharma, N. et al. Predicting solar generation from weather forecasts using machine learning. In: **2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)**. [S.l.: s.n.], 2011. p. 528–533.

SON, J. et al. Sensorless pv power forecasting in grid-connected buildings through deep learning. **Sensors**, v. 18, p. 2529, 08 2018.

STANčIN, I.; JOVIć, A. An overview and comparison of free python libraries for data mining and big data analysis. In: **2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. [S.l.: s.n.], 2019. p. 977–982.

TAIGMAN, Y. et al. Deepface: Closing the gap to human-level performance in face verification. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 1701–1708.

WANG, K.; QI, X.; LIU, H. A comparison of day-ahead photovoltaic power forecasting models based on deep learning neural network. **Applied Energy**, v. 251, p. 113315, 2019. ISSN 0306-2619. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306261919309894>.

WHITE, S. **Solar Photovoltaic Basics**. Routledge, 2014. Disponível em: <https://doi.org/10.4324/9781315770116>.

WORLD Werather Online. 2020. <https://www.worldweatheronline.com/developer/>. Accessed: 2022-04-24.

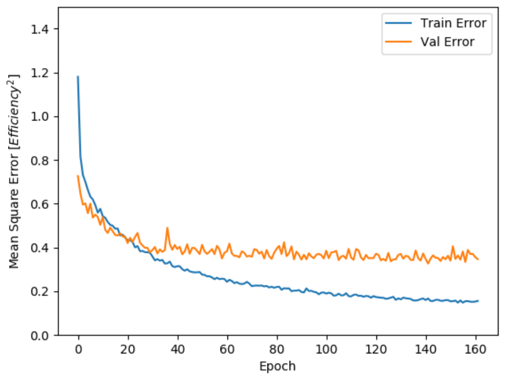
APPENDIX A — APPENDIX

This chapter includes all plots resulting from our approach's evaluation. Each figure contains the plot generated for each season and undivided data. Furthermore, all these plots and the models that produced them are available on our Github page (FRANCO, 2020).

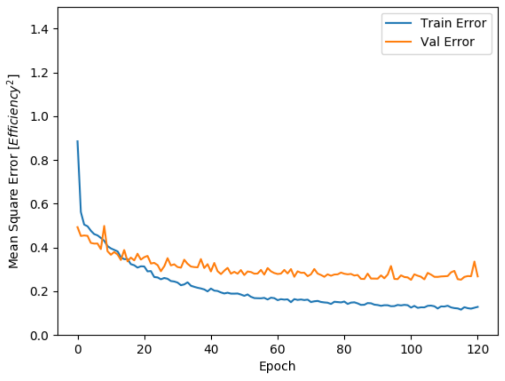
A.1 Training Result Plots

A.1.1 Washington DC

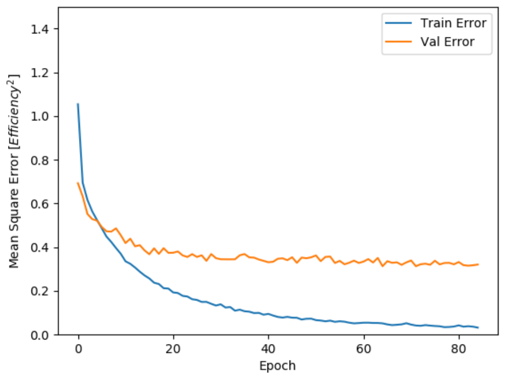
Figure A.1: Location A Loss Progression in different season.



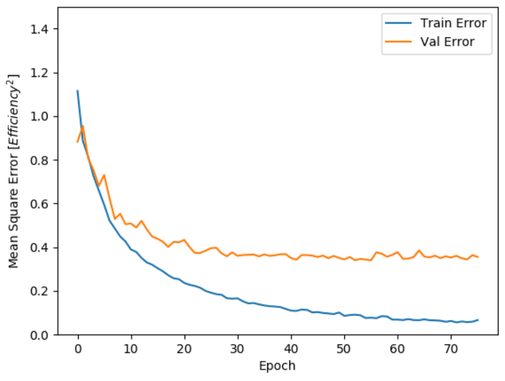
(a) Summer



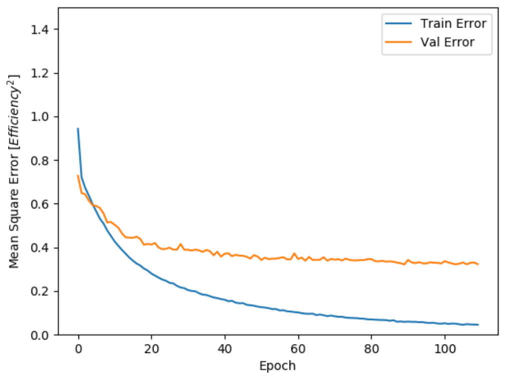
(b) Fall



(c) Winter



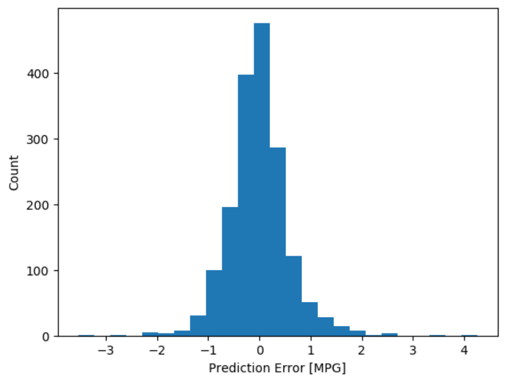
(d) Spring



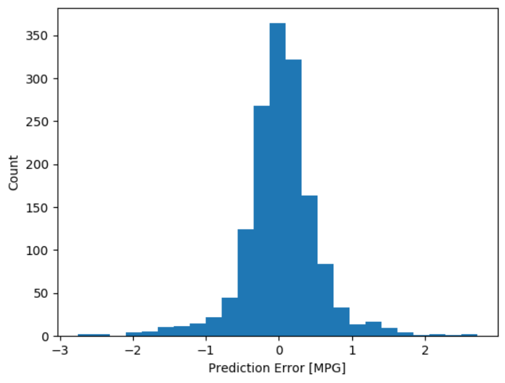
(e) Undivided

Source: The Authors

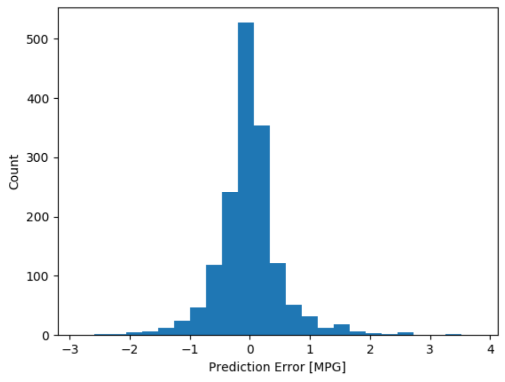
Figure A.2: Location A Pediction Erros in different season.



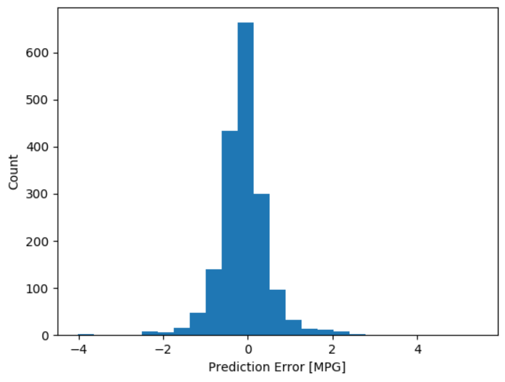
(a) Summer



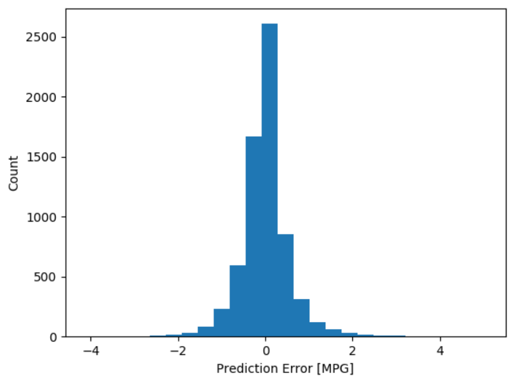
(b) Fall



(c) Winter



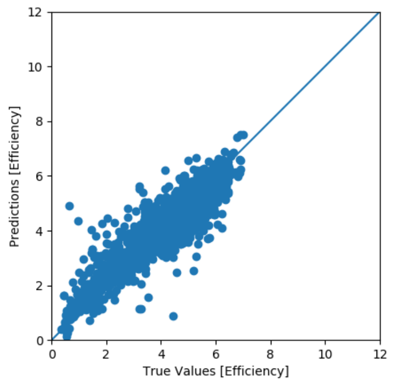
(d) Spring



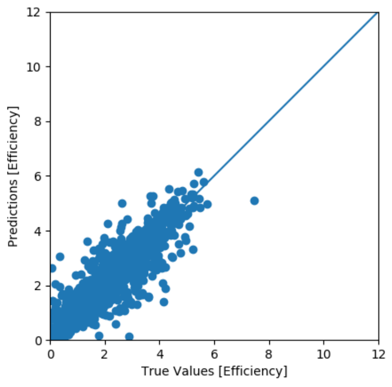
(e) Undivided

Source: The Authors

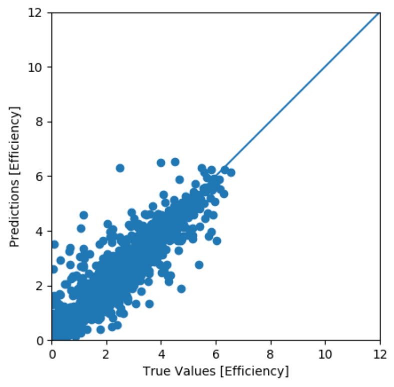
Figure A.3: Location A True x Predicted in different season.



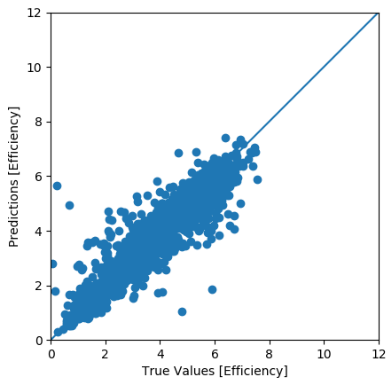
(a) Summer



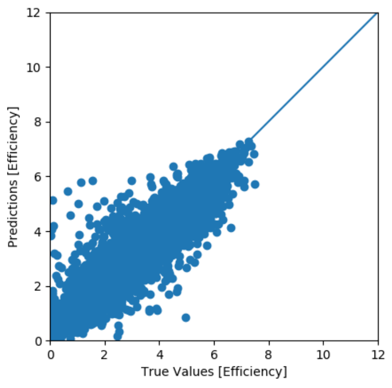
(b) Fall



(c) Winter



(d) Spring

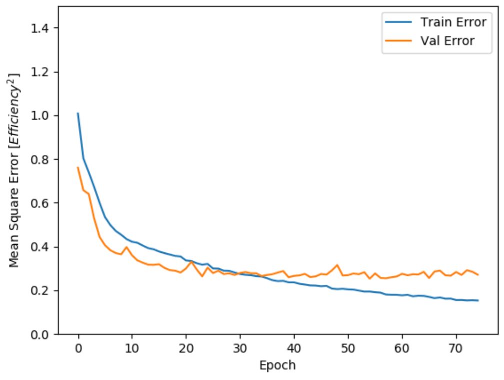


(e) Undivided

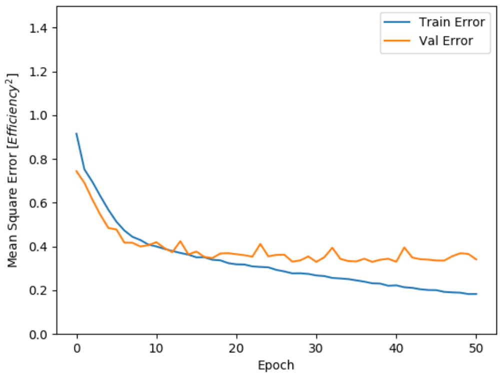
Source: The Authors

A.1.2 San Francisco

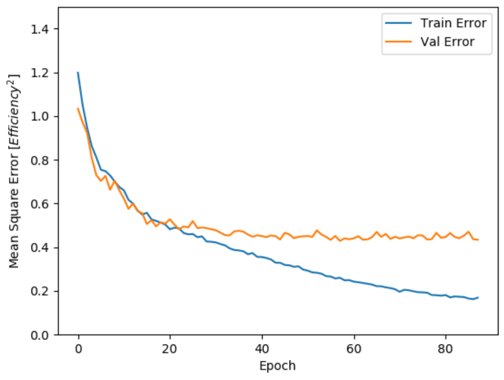
Figure A.4: Location B Loss Progression in different season.



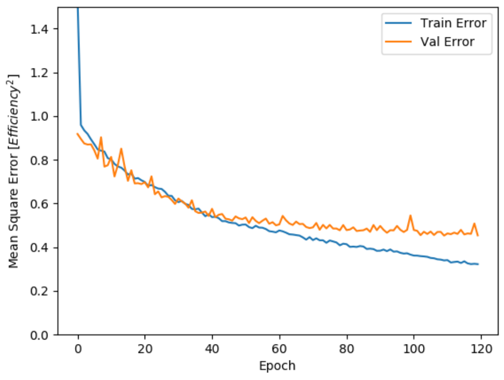
(a) Summer



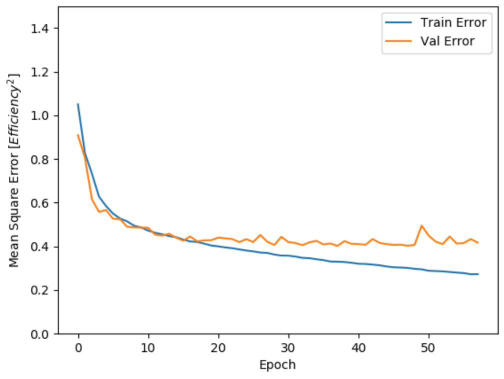
(b) Fall



(c) Winter



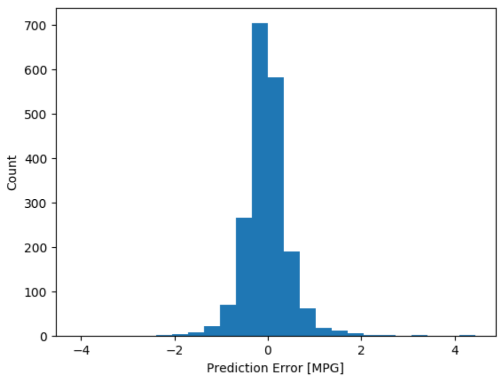
(d) Spring



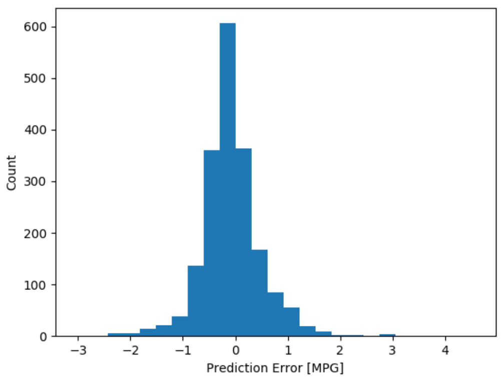
(e) Undivided

Source: The Authors

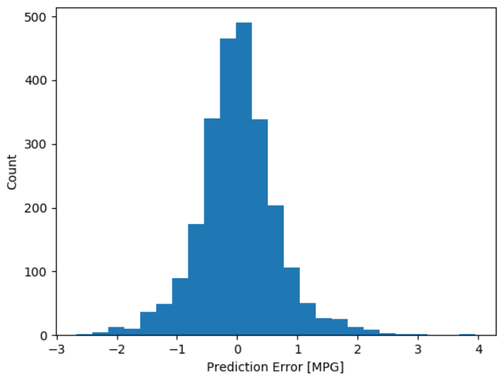
Figure A.5: Location B Pediction Erros in different season.



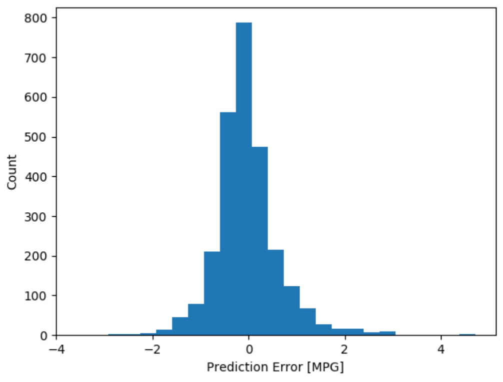
(a) Summer



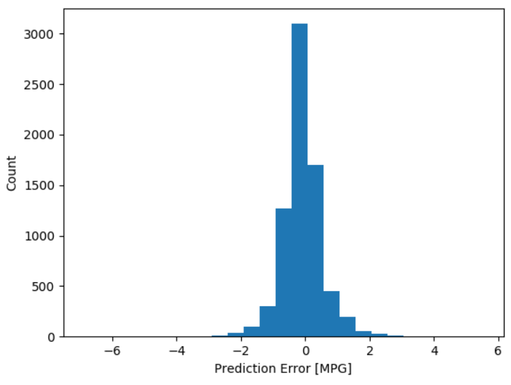
(b) Fall



(c) Winter



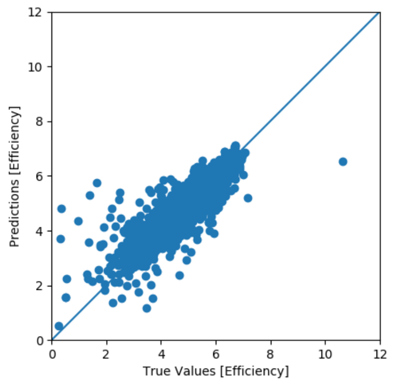
(d) Spring



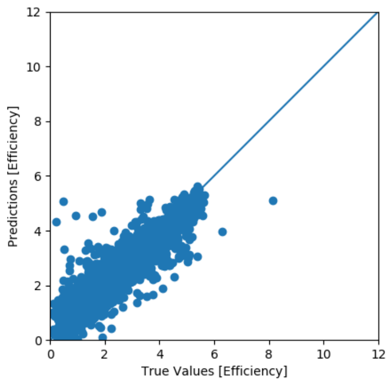
(e) Undivided

Source: The Authors

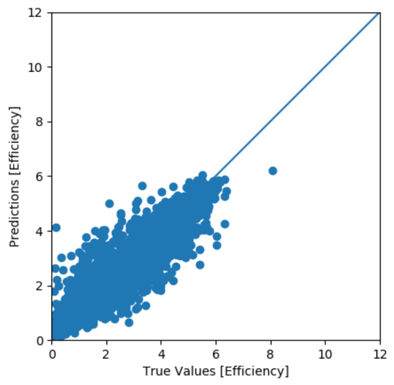
Figure A.6: Location B True x Predicted in different season.



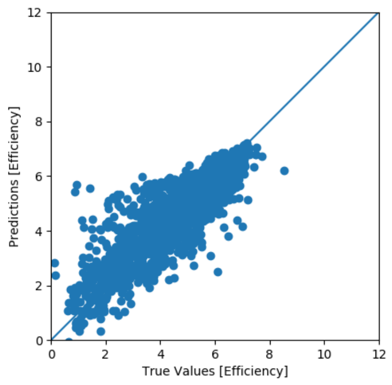
(a) Summer



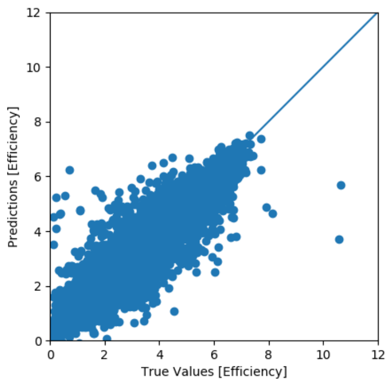
(b) Fall



(c) Winter



(d) Spring

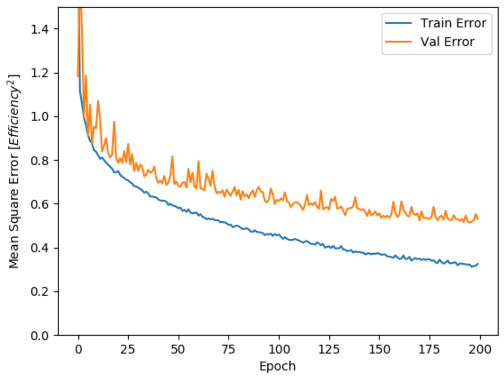


(e) Undivided

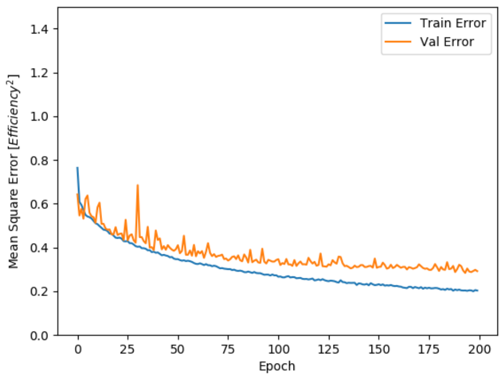
Source: The Authors

A.1.3 Milan

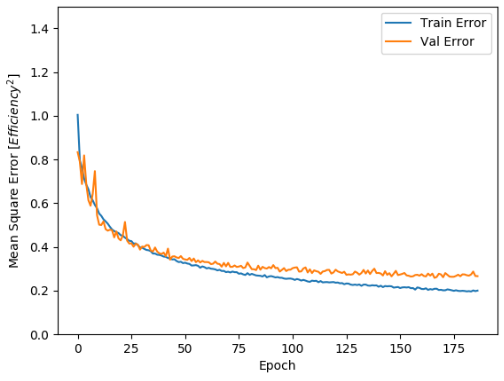
Figure A.7: Location C Loss Progression in different season.



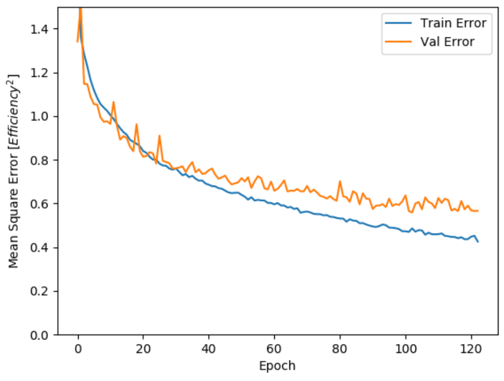
(a) Summer



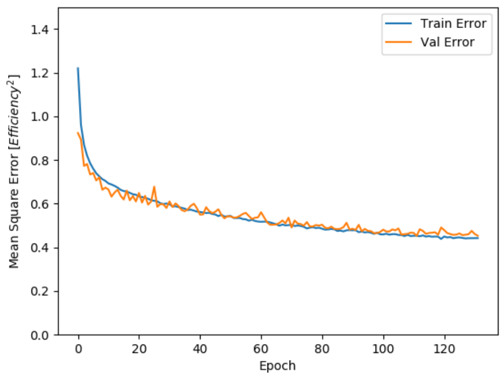
(b) Fall



(c) Winter



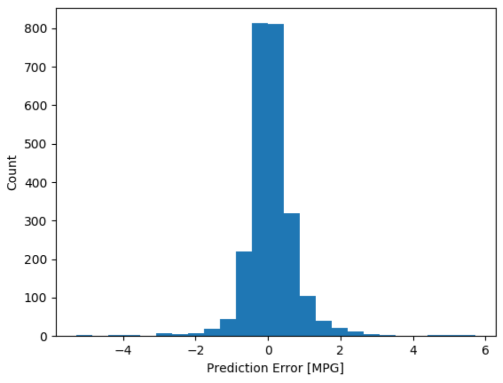
(d) Spring



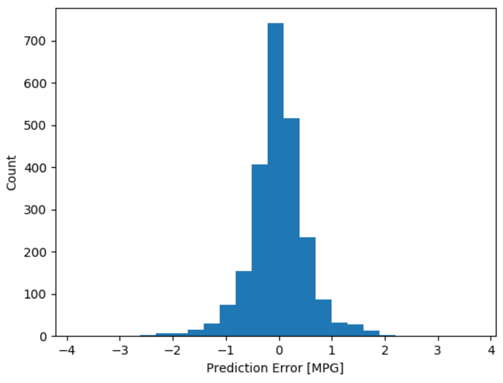
(e) Undivided

Source: The Authors

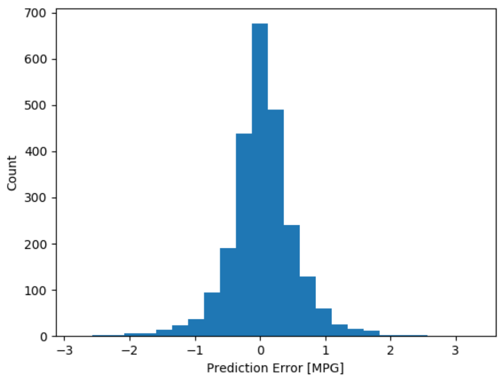
Figure A.8: Location C Pediction Erros in different season.



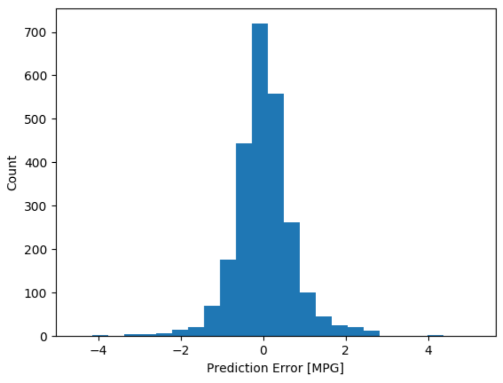
(a) Summer



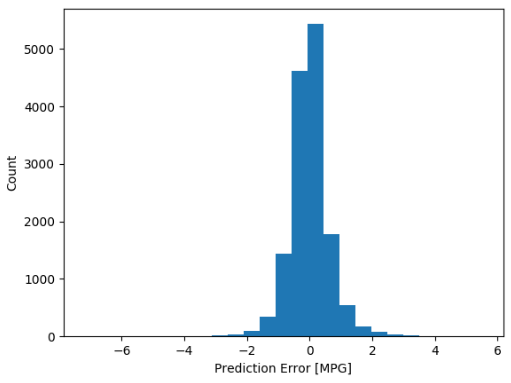
(b) Fall



(c) Winter



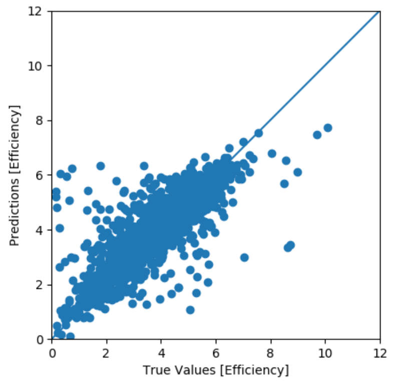
(d) Spring



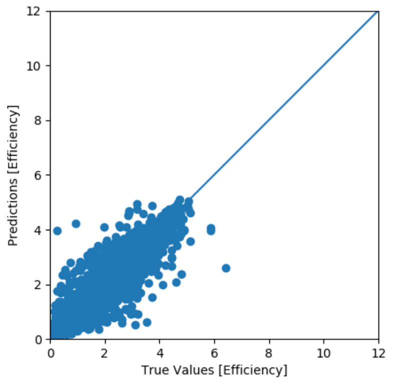
(e) Undivided

Source: The Authors

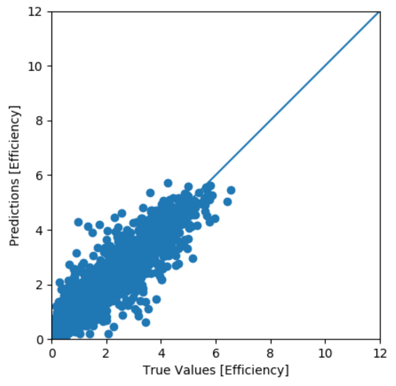
Figure A.9: Location C True x Predicted in different season.



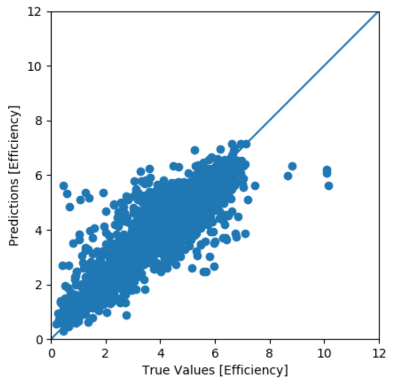
(a) Summer



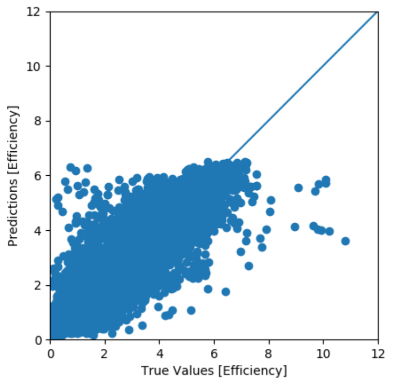
(b) Fall



(c) Winter



(d) Spring

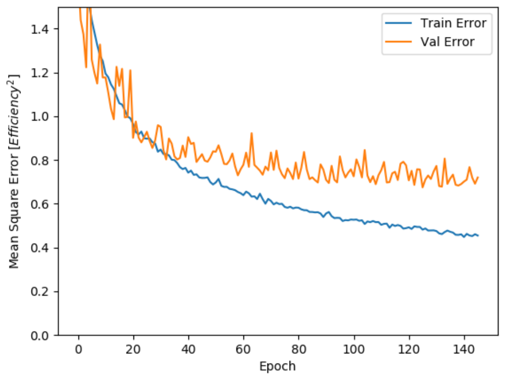


(e) Undivided

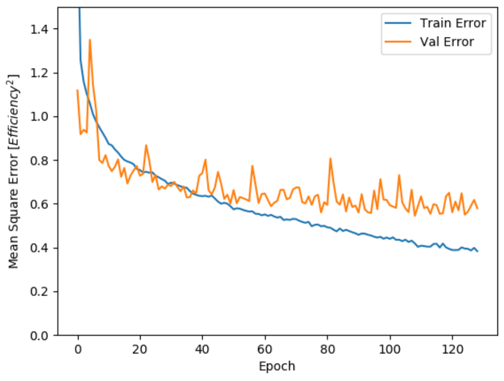
Source: The Authors

A.1.4 Sydney

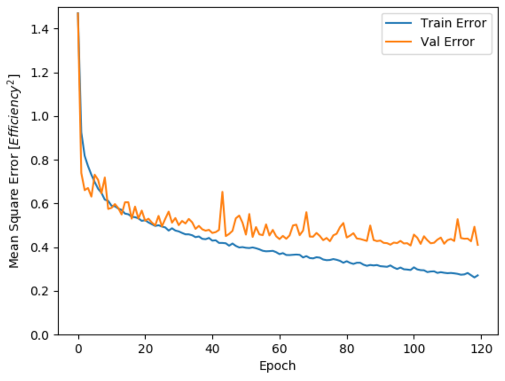
Figure A.10: Location D Loss Progression in different season.



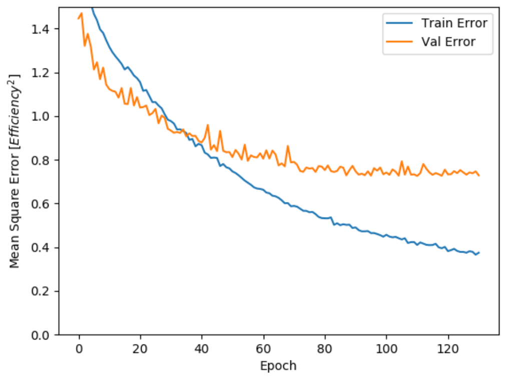
(a) Summer



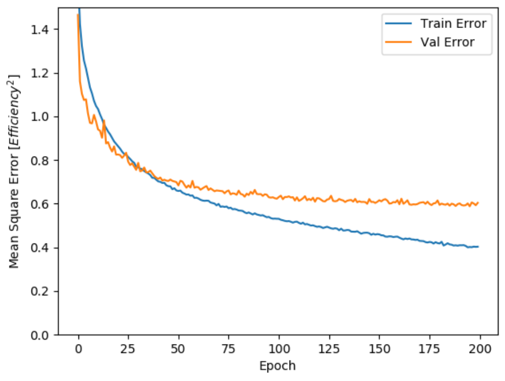
(b) Fall



(c) Winter



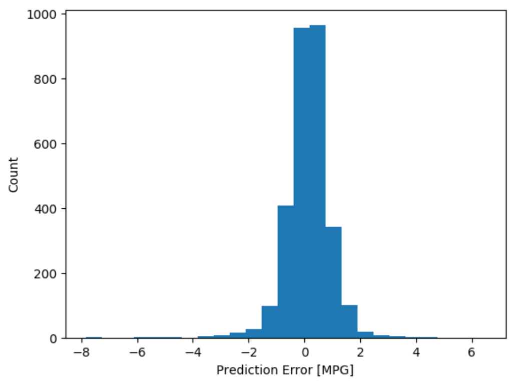
(d) Spring



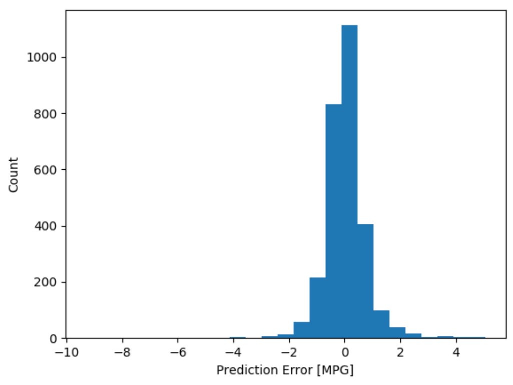
(e) Undivided

Source: The Authors

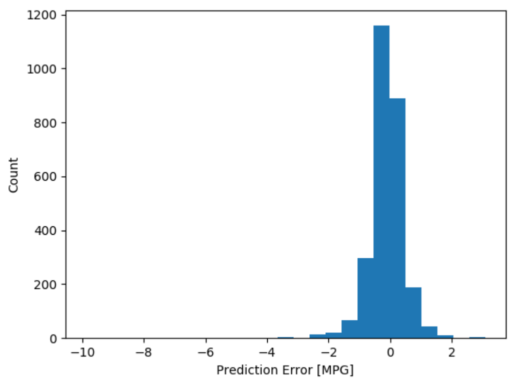
Figure A.11: Location D Pediction Erros in different season.



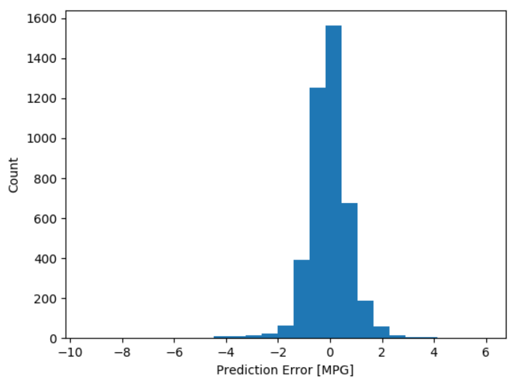
(a) Summer



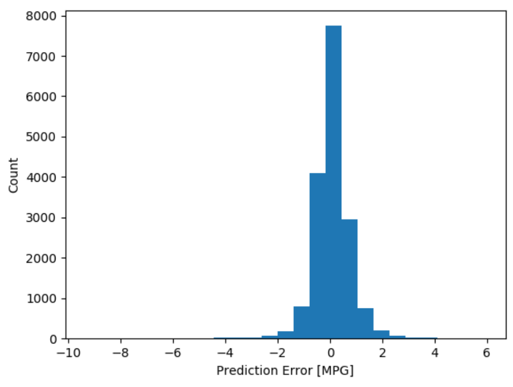
(b) Fall



(c) Winter



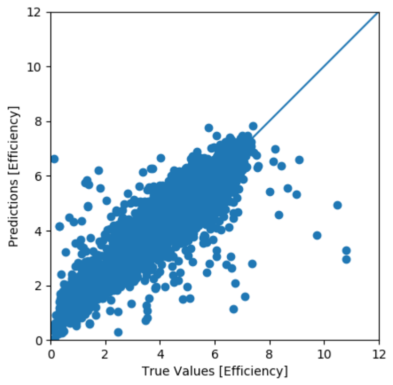
(d) Spring



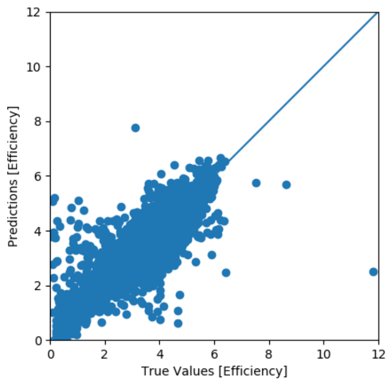
(e) Undivided

Source: The Authors

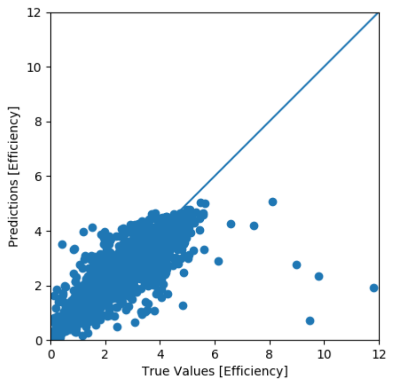
Figure A.12: Location D True x Predicted in different season.



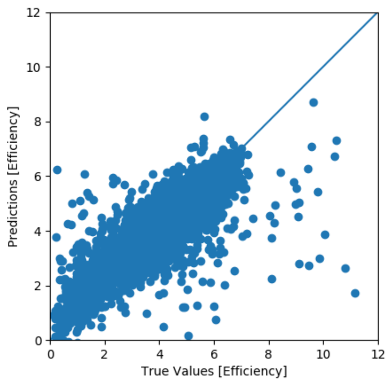
(a) Summer



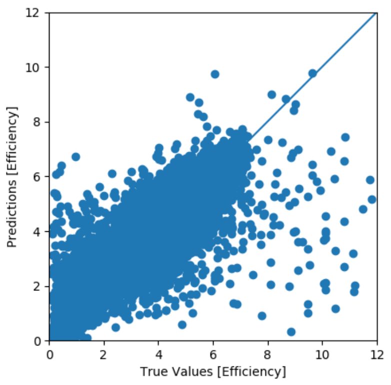
(b) Fall



(c) Winter



(d) Spring



(e) Undivided

Source: The Authors