**Solar Forecasting using Deep Learning**

**Renewable Energy Project**

Submitted by:

### **Arpit Mathur (9915103112)**

### **Ravi Gaurav (9915103128)**

### **Gaurav Singh (9915103252)**

Under the supervision of:

**Dr. Vinay Tikkiwal**

**Jaypee Institute of Information Technology University, Noida**

**Semester VI, 2018**

# **ACKNOWLEDGEMENT**

The completion of any inter-disciplinary project depends upon cooperation, co-ordination and combined efforts of several sources of knowledge. We are grateful to Dr. Vinay Tikkiwal for his even willingness to give us valuable advice and direction whenever we approached him with a problem. We are thankful to him for providing us with immense guidance for this project.

We are also thankful Vinay Tikkiwal sir for giving his valuable time for evaluating our project. We would also like to thank our College authorities and Head/Dean for giving us the opportunity to pursue our project in this field and helping us to successfully complete this project.

**Signature:**

**Names:**

Arpit Mathur (9915103112) – F4

Ravi Gaurav (9915103128) – F4

Gaurav Singh (9915103252) – F8

**PROBLEM STATEMENT**

Power forecasting of renewable energy power plants is a very active research ﬁeld, as reliable information about the future power generation allow for a safe operation of the power grid and helps to minimize the operational costs of these energy sources. Deep Learning algorithms have shown to be very powerful in forecasting tasks, such as economic time series or speech recognition.

Up to now, Deep Learning algorithms have only been applied sparsely for forecasting renewable energy power plants. By using different Deep Learning and Artiﬁcial Neural Network Algorithms, such as LSTM, we introduce these powerful algorithms in the ﬁeld of renewable energy power forecasting.

Our motive is to show the forecast strength of these algorithms compared to a standard MLP and a physical forecasting model in the forecasting the energy output of 21 solar power plants and compare our results with results obtained from Artiﬁcial Neural Networks as well as other reference models such as physical models.

**DATASET EXPLAINED: -**

The GermanSolarFarm data set contains 21 photovoltaic facilities in Germany. Their installed nominal power ranges between 100kW and 8500kW. The PV facilities range from PV panels installed on rooftops to fully fledged solar farms. They are distributed throughout Germany as shown in the attached image.

For each facility historical NWP data and the produced power in a three-hour resolution for 990 days are available. All-time series in the data set, except the measured power output, are normalized between 0 and 1 using the minmax normalization.

The target variable, the measured power output, is normalized using the nominal output capacity of the corresponding PV facility. Therefore, allow the comparison of the forecasting performance without taking the size of the PV facilities into account.

**

Fig1 Solar Farm

Locations

**TECHNIQUES AND ALGORITHMS USED**

* 1. ***PCA (Principle Component Analysis)****: -*

Principal Component Analysis (PCA) is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set.

The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

METHOD:

1. Calculate the covariance matrix
2. Calculate the eigenvectors and eigenvalues of the covariance matrix
   1. Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalues for this matrix. These are rather important, as they tell us useful information about our data. by this process of taking the eigenvectors of the covariance matrix, we have been able to extract lines that characterize the data.
3. form a feature vector, which is just a fancy name for a matrix of vectors.
   1. This is constructed by taking the eigenvectors that you want to keep from the list of eigen vectors and forming a matrix with these eigenvectors in the columns.



1. Deriving the new data set



* 1. *ANN (Artificial Neural Network): -*

**Artificial neural networks** (**ANNs**) or **connectionist systems** are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" (i.e. progressively improve performance on) tasks by considering examples, generally without task-specific programming.

An ANN is based on a collection of connected units or nodes called artificial neurons (a simplified version of biological neurons in an animal brain). Each connection (a simplified version of a synapse) between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it.



* 1. Libraries Used: -

Scikit-Learn

Scikit-learn (formerly scikits.learn) is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language.[[3]](https://en.wikipedia.org/wiki/Scikit-learn#cite_note-jmlr-3) It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [k-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy).

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

Pandas

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

PVlib

 PVLIB Python is a community supported tool that provides a set of functions and classes for simulating the performance of photovoltaic energy systems.

Itertools

Itertools is a module for the Python language which contains high level functional constructs for working with iterable objects and generators. itertools is a module for the Python language which contains high level functional constructs for working with iterable objects and generators.

Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level, mathematical functions to operate on these arrays.

**IMPLEMENTATION: -**

Code:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import tensorflow

dataset = pd.read\_csv('pv\_01.csv')

X = dataset.iloc[:, 0:40].values

y = dataset.iloc[:, 40].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.decomposition import PCA

pca = PCA(n\_components = 10)

X\_train = pca.fit\_transform(X\_train)

X\_test = pca.transform(X\_test)

explainvar = pca.explained\_variance\_ratio\_

import keras

from keras.models import Sequential

from keras.layers import Dense

predictor = Sequential()

predictor.add(Dense(activation="sigmoid", input\_dim=10, units=15, kernel\_initializer="uniform"))  # input layer and the first hidden layer  #Dense(output\_dim = 6, init = 'uniform', activation = 'relu', input\_dim = 11)

predictor.add(Dense(activation="sigmoid", units=10, kernel\_initializer="uniform"))  #Seecond hidden layer Dense(output\_dim = 6, init = 'uniform', activation = 'relu')

#predictor.add(Dense(activation="relu", units=15, kernel\_initializer="uniform"))  #Seecond hidden layer Dense(output\_dim = 6, init = 'uniform', activation = 'relu')

predictor.add(Dense(activation="sigmoid", units=20, kernel\_initializer="uniform"))  #Seecond hidden layer Dense(output\_dim = 6, init = 'uniform', activation = 'relu')

predictor.add(Dense(activation="sigmoid", units=1, kernel\_initializer="uniform"))  # Output LayerDense(output\_dim = 1, init = 'uniform', activation = 'sigmoid')

predictor.compile(optimizer = 'adam', loss = 'mean\_squared\_error', metrics = ['accuracy'])

predictor.fit(X\_train, y\_train, batch\_size = 10, nb\_epoch = 500)

1. **RESULTS: -**
2. **EVALUTATION AND COMPARISON: -**
3. **CONCLUSION: -**
4. **REFERENCES: -**
5. Gensler, A., Henze, J., Sick, B., & Raabe, N. (2016, October). Deep Learning for solar power forecasting—An approach using AutoEncoder and LSTM Neural Networks. In Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on (pp. 002858-002865). IEEE.
6. Reno, M. J., Hansen, C. W., & Stein, J. S. (2012). Global horizontal irradiance clear sky models: Implementation and analysis. SANDIA report SAND2012-2389.