Applied Energy 294 (2021) 117014

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy

Deep learning based solar radiation micro forecast by fusion of infrared cloud images and radiation data

Meenu Ajith *, Manel Martínez-Ramón

Department of Electrical and Computer Engineering, The University of New Mexico, NM, 87106, USA

ARTICLE INFO

Keywords:
Solar energy forecasting
Deep learning
Multimodal feature fusion
Hybrid convolutional long short term memory
Infra-red images

ABSTRACT

Solar irradiance forecasting has been gaining paramount importance in recent years due to its impact on power grids. However, solar energy harvesting over shorter periods also brings new challenges due to its intermittent and uncertain attributes. Hence, accurate forecasting has become an indispensable aspect of the effective management of power system operations. The existing models focus on using only time-series data for solar radiation forecasting. But during cloudy time instances, it fails to quickly capture the nonlinear Spatio-temporal variations in the data for shorter periods. To bridge this gap, in this paper, a multi-modal fusion network is developed for studying solar irradiance micro forecasts by using both infrared images and past solar irradiance data. Here both spatial and temporal information is extracted parallelly and fused using a fully connected neural network. The solar forecasts of the proposed methods are evaluated against benchmark models in terms of Mean Absolute Percentage Error (MAPE) and other qualitative measures. The experimental results illustrate that the multi-modal fusion networks outperform the existing methods while predicting solar irradiance for cloudy days as well as mixed days (both cloudy and sunny days). Hence a transfer learning-based classifier with 99.23% accuracy is developed to categorize the cloudy days from sunny days. In the case of higher horizon forecasts, the proposed models show the optimum trade-off between performance and test time. Moreover, the Multiple Image Convolutional Long Short Term Memory Fusion Network (MICNN-L) shows a 46.42% improvement in MAPE whereas the Convolutional Long Short Term Memory Fusion Network (CNN-L) has a 42.02% increase when compared to the benchmark machine learning and deep learning models.

1. Introduction

Solar energy is a promising renewable energy source that has grown considerably over the past decades. It has played a major role to tackle the global energy crisis and the climate change. The fraction of incident solar radiation on the Earth surface is determined by the cloudiness present in the sky. However, the solar irradiance over relatively short periods are subject to rapid fluctuations due to its dependency on the motion and distribution of the clouds [1]. This leads to ambiguity and inconsistencies in determining the amount of solar irradiance, especially on a partially or fully cloudy day. Further researches in weather conditions concluded that the most accurate forecasts were made during a clear sky day whereas perceptible fluctuations in solar radiation were observed during cloudy days [2]. Thus cloud classification is crucial to predict solar radiation [3,4]. Accurate short-term prediction of solar radiation is paramount for curbing the economic inefficiencies and disruptions in power generation. Besides, a reliable forecast also maintains the power grid stability and helps with planning backups, load and congestion management, and changing power sources [5–7].

The advent of modern services in smart grids urges the need for instantaneous availability of solar radiation data, especially within few minutes. But in the case of solar radiation micro forecast, the variability in the data is difficult to capture using the existing prediction methods [8]. Traditional solar irradiance forecasts rely on the Numerical Weather Prediction (NWP) models to simulate weather conditions and generated longer forecasts ranging from 6 h to several days [9]. Later, a variety of other features such as satellite images, total-sky images, cloud indices (CI) [10], and weather information [11,12] were used to enhance the predictions. Initially, most researchers used satellite images for solar irradiance micro forecasting, but the low resolution of these images resulted in inaccuracies during cloudy conditions. Thus, total sky imagers were introduced to model cloud motion patterns for very short-term forecast [13,14]. Although total sky images have a very high spatial–temporal resolution, they have a limited range compared to satellite-based imaging [15,16]. Hence satellite images are used for forecasting ranging from 1-6 h whereas micro forecast for 0–15 min ahead is performed using total-sky images. However,

* Corresponding author.
E-mail address: majith@unm.edu (M. Ajith).

https://doi.org/10.1016/j.apenergy.2021.117014
Received 13 January 2021; Received in revised form 8 April 2021; Accepted 23 April 2021
Available online 28 April 2021

Abbreviations

ANN	Artificial Neural Network
CI	Cloud Indices
CNN	Convolutional Neural Networks
CNN-L	Convolutional Long Short Term Memory Fusion Network
DAQ	Data Acquisition System
DNN	Deep Neural Network
ELU	Exponential Linear Unit
GHI	Global Horizontal Irradiance
GP	Gaussian Processes
GPR	Gaussian Process Regression
GRU	Gated Recurrent Unit
LGP	Linear Gaussian Processes
LSTM	Long Short-Term Memory
LSVM	Linear Support Vector Machines
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MeAPE	Median Absolute Percentage Error
MICNN-L	Multiple Image Convolutional Long Short Term Memory Fusion Network
MLP	Multilayer Perceptron
NLGP	Non-linear Gaussian Processes
NLSVM	Non-linear Support Vector Machines
NWP	Numerical Weather Prediction
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMO	Sequential Minimal Optimization
SVM	Support Vector Machines
SVM	Support Vector Machines

Kernel learning symbols

α_n^*	Scalar coefficient or Lagrange multiplier for constraints defined in negative errors
α_n	Scalar coefficient or Lagrange multiplier for constraints defined in positive errors
$\boldsymbol{\alpha}$	Column vector of coefficients α_n
$\mathbf{K}$	Kernel matrix, Gram matrix of kernel dot products between vectors $\mathbf{x}_n$
$\mathbf{k}(\cdot)$	Vector of kernel dot products
$\boldsymbol{\Sigma}$	Covariance matrix of the Gaussian prior probability distribution of the parameter vector $\mathbf{w}$ in a Gaussian process
$\mathbf{w}$	Column vector containing the parameters of a linear estimator
$\mathbf{x}^*$	Test input sample
$\delta(k)$	Kronecker's delta function
$\mathcal{H}$	Reproducing kernel Hilbert space
$\mathcal{N}(\mu, \Sigma)$	Multivariate Gaussian distribution with mean μ and covariance matrix Σ
μ_n^*	Scalar coefficient or Lagrange multiplier for constraints defined in negative errors
μ_n	Scalar coefficient or Lagrange multiplier for constraints defined in positive errors
$\phi(\cdot)$	Nonlinear mapping into a Hilbert space
σ_n^2	variance of the Gaussian model for the estimation error in a Gaussian process
c	Scalar used as a bias kernel in the covariance function of a Gaussian process
$f(\cdot)$	Estimation function
f_*	Latent prediction function evaluated at $\mathbf{x}^*$
$k(\cdot,\cdot)$	Positive definite function, Mercer's kernel
$\mathbf{1}$	Column vector of ones

Neural networks symbols

$\mathbf{b}$	Vector of biases
$\mathbf{c}_n$	Vector of functions used as inner state gate in stage n of an LSTM or GRU network
$\mathbf{F}$	Convolution filter or kernel in a convolution operation in a CNN
$\mathbf{n}$	Vector of functions used as a forget gate in stage n of an LSTM or GRU network
$\mathbf{h}_k$	Feature vector at the output of layer k of a neural network
$\mathbf{I}$	Input matrix of a convolution operation in a CNN
$\mathbf{i}_n$	Vector of functions used as input gate in stage n of an LSTM or GRU network
$\mathbf{o}$	Vector of outputs of a neural network
$\mathbf{o}_n$	Vector of functions used as output in stage n of a RNN, LSTM or GRU network
$\sigma(\cdot)$	Vector of Sigmoidal functions
$\mathbf{U}$	Matrix resulting of a convolution operation in a CNN
$\mathbf{W}_k$	Matrix or tensor of linear parameters connecting layer $k-1$ with layer k of a neural network
∇	Gradient operator
$\otimes$	Hadamard (or element-wise) product between matrices or vectors of the same dimension
$\sigma(\cdot)$	Sigmoidal function used as activation in a neural network
$\tilde{\mathbf{c}}_n$	Vector of functions used as output state gate in stage n of an LSTM network
C	Trade-off parameter that weights the trade-off between the structural and the empirical risks in an SVM
D_k	Number of nodes in layer k of a neural network
h_k, d	Feature d in layer k of a neural network
$J_{ML}(\cdot)$	Maximum Likelihood cost function
$L_p(\cdots)$	Optimization functional defined over the primal parameters of an estimator
T	Time horizon
x_t	Sample of a time series

Signals and data symbols

$\mathbf{X}$	Matrix containing a sequence of vectors $\mathbf{x}_n$
$\mathbf{x}_n$	Column vector containing a sequence of samples x_n
$\mathbf{y}$	Column vectors of desired estimation outputs
b	Bias
e_n	Estimation error at instant n

these models are not robust enough and further studies have been conducted using various image processing and statistical methods to improve the predictions. Later, cloud motion displacement vectors were used to capture the varying cloud speeds for very short time intervals [17,18]. This approach was found to be more effective than particle image velocimetry and optical flow-based feature extraction

methods. In the case of statistical methods, Artificial Neural Networks (ANN) [19], Support Vector Machines (SVM), Markov Chains, Autoregressive models [20,21], and Regression models [22,23] were analyzed to evaluate their performance under various meteorological conditions. However, these methods were ineffective due to their high dependency on historical data which restricted them to capture and model small fluctuations in the solar radiation [24]. Later, ensemble-based learning using ANNs, Gaussian Process Regression (GPR) [25] and random forest was introduced to address the drawbacks of an individual system [26]. This idea was further extended when past irradiance outputs and other statistical metrics extracted from sky images were fed into the ANNs [27]. This model was able to learn complex non-linear features from different inputs and generate comparatively better forecasts for smaller datasets [28]. However, these conventional machine learning-based models were unable to learn useful representations with when the data sizes increased. Meanwhile, deep learning-based solar irradiance forecasting gained tremendous popularity, as it relies on representational learning to improve the accuracy of the predictions. The various stacked layers extract deep features from large data samples and avoid the need for complex feature engineering. The main models that specialize in handling sequential data include recurrent neural networks (RNN), long short-term memory (LSTM) networks, gated recurrent unit (,GRU) networks, and convolutional LSTMs. CNN based approaches, which demonstrated superior performance for intra hour solar forecasting, use spatial features from the sky images [29–32]. Later, a convolutional LSTM model to extract both the spatial and temporal features from the sky images was used in [33], but the model generated larger error for higher time horizons and, hence, it failed to learn the Spatio-temporal correlations in the data. Another hybrid model that succeeded in fully integrating the merits of CNN and LSTM used only the solar radiation data as the input. At first, the CNN was used to extract the intrinsic features of the solar radiation time series, and later an LSTM was connected to learn the dependencies from the sequential data [34]. The commonly used image input to CNN for short term forecasting included total sky images, since it provides information related to the cloud movements and sky conditions. However, total sky imager equipment was expensive and the methodology was computationally intensive for cloud mapping. To alleviate these shortcomings, infra-red-based cloud imagers were developed for solar irradiance prediction [35].

In this paper, a deep learning-based Multi-Modal CNN LSTM fusion network is introduced for solar radiation micro forecasting. This work uses data from infra-red images of the sky to capture the cloud motion and distribution. The fusion of infrared image information and the lagged data of the forecast variable helps to overcome the shortcomings of the existing approaches. To further aid the proposed system, a transfer learning method based on MobileNetV2 [36] was introduced to classify between sunny and cloudy days. This can help to identify the cloudy time instances and it can be further used to train the multi-modal fusion network. The main contributions of this paper are as follows:

- A multipath parallel model is introduced so that the deep features extracted from the infra-red image using CNN are combined with the time series features extracted from the LSTM for multi-step solar radiation prediction during a cloudy day. The CNN extracts only the representative features, whereas the LSTM captures the temporal dynamics of the solar radiation data. Hence the most significant features are integrated to form a sequence which is further fed into the dense layers for obtaining the final prediction.
- The proposed method shows superior performance with competitive computational efficiency. It illustrates the optimal trade-off between performance and time complexity.
- This network outperforms the benchmark models by achieving significant improvement for higher time horizons.
- A classifier based on transfer learning is introduced to distinguish between sunny and cloudy time instances in a day.

The rest of the paper is formulated as follows. Section 2 introduces the data and methodology associated with the time series-based forecasting models. Section 3 explains the image processing-based methods and the overall methodology of the proposed models. Section 4 introduces the transfer learning method used for the classification of images. Section 5 outlines a comprehensive overview of the study area, data processing, and training criteria of all the methods introduced in the paper. Section 6 presents the results and discussion using the different models. Finally, the paper is concluded in Section 7.

2. Time series based forecasting methods

2.1. Time series processing and data structures

Assume a time series $x_t \in \mathbb{R}$, $1 \leq t \leq n$. The forecasting task consists of estimating future values x_{n+T} of the time series at different horizons T. If there is a correlation, or more generally, a mutual information between the past time instants and the future instants of the sequence x_t, it is reasonable to construct a parametric function $x_{n+T} = f(\mathbf{x}_n) + e_n$ where $\mathbf{x}_n = \{x_{n-D+1} \cdots x_n\}^\top \in \mathbb{R}^D$coefficients is a column vector containing a window of D past samples of the sequence and e_n is the prediction error for sample at horizon T. Consider a historical sequence recorded for ML training, consisting of input/output pairs $\{\mathbf{x}_n, x_{n+T}\}$. For notation purposes, the training inputs are organized in a matrix $\mathbf{X} = \left(\mathbf{x}_1 \cdots \mathbf{x}_N\right) \in \mathbb{R}^{DxN}$ and the training (desired) outputs are placed in a column vector $\mathbf{y} = \{x_{D+T} \cdots x_{N+T}\}^\top$ to be used as regressors in a prediction algorithm.

2.2. Kernel methods versus deep learning methods

A plethora of algorithms can be used for the forecast. We present a benchmarking with the two main state-of-the-art ML algorithms, which are kernel methods and deep learning methods. The first ones are constructed over a linear that can be formulated as

$$y_n = \mathbf{w}^\top \phi(\mathbf{x}_n) + b + e_n \tag{1}$$

where $\phi(\cdot)$ is a mapping into a Hilbert space of higher dimensionality $\mathcal{H}$ endowed with a dot product $\langle \cdot, \cdot \rangle$ that can be expressed as a positive definite map $k(\cdot, \cdot) \in \mathbb{R}$ (usually called Mercer's kernel) [37] of input samples, i.e.,

$$\langle \phi(\mathbf{x}_n), \phi(\mathbf{x}_m) \rangle = \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) \tag{2}$$

The Representer Theorem [38] states that, under non restrictive conditions, the weight vector $\mathbf{w}$ can be represented as a linear combination of the training data, this is $\mathbf{w} = \sum_{n=1}^{N} \alpha_n \mathbf{x}_n$ and hence estimator (1) admits a dual representation.

$$y_n = \sum_{m=1}^{N} \phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n) + b + e_n = \sum_{m=1}^{N} \alpha_m k(\mathbf{x}_m, \mathbf{x}_n) + b = \boldsymbol{\alpha}^\top \mathbf{k}(\mathbf{x}_n) + b \tag{3}$$

where $\boldsymbol{\alpha} = \{\alpha_1 \cdots \alpha_N\}^\top$ is the vector of dual coefficients and

$$\mathbf{k}(\mathbf{x}_n) = \{k(\mathbf{x}_1, \mathbf{x}_n), \ldots, k(\mathbf{x}_N, \mathbf{x}_n)\}^\top$$

is called a dual representation of vector $\phi(\mathbf{x}_n)$, which is simply a projection of this sample onto the subspace of all training data samples. Several criteria can be used to optimize the dual parameters, the most immediate being a regularized MMSE criterion, also called ridge regression (RR), a maximum margin criterion, supported by the statistical learning theory [39], which leads to the SVM or a posterior distribution maximization through a probabilistic model of the regressors, which leads to the Gaussian Processes (GP) for Machine Learning [40].

The second class of algorithms constitutes deep learning-based methods. Roughly speaking one can think of a deep learning algorithm as a machine that is organized in several layers of nodes, each one of them nonlinearly processes the input to produce an output intended to have a higher level of abstraction set of features. If a given layer $k-1$

outputs a set of D_{k-1} features $\mathbf{h}_{k-1} = \{h_{k-1,1}, \dots, h_{k-1,D_{k-1}}\}$, each node of next layer k produces a new set of D_k features.

$$h_{k,d} = \sigma(\mathbf{w}_{k,d}^{\top}\mathbf{h}_{k-1} + b_{k,d}) \tag{4}$$

where $\sigma(\cdot)$ is a nonlinear, monotonically increasing function called activation. Usually, weights $\mathbf{w}_{k,d}$ are grouped in a weight tensor $\mathbf{W}_k$. In this paper, these tensors have dimensions $D_{k-1} \times D_k$, this is, they are simply matrices. The first layer is the input, this is, $\mathbf{h}_0 = \mathbf{x}_n$, and the last layer is intended to approximate the desired output x_{n+T}. The structure can easily be constructed as a multitask network, which can predict several horizons at a time. These structures are formulated using these weight matrices or tensors, and they can be thought of as a cascade of adaptive basis functions. Since the representation is intrinsically nonlinear, they do not admit a dual representation and the training is usually more complex computationally. The computational burden, however, can be lower and more tractable than for the case of kernel machines when the number of training data increases. These structures offer advantages and trade-offs for kernels. They are usually trained using gradient descent to obtain a cost function expressed in terms of a cross-entropy between the input and the output, which further leads to the well-known backpropagation algorithm. Besides the standard deep neural network (DNN) that is directly defined the previously introduced functional structure, we summarize here different structures that are adapted to the sequence process, mainly when this sequence has a temporal structure that can be exploited. They are called recurrent neural networks. Variants of the recurrent neural networks are the GRU and the LSTM.

2.3. Support vector machines (SVM)

An SVM is a machine whose direct criterion consists of the maximization of a margin defined over the regression hyperplane, while at the same time, the expressive capacity of the machine is minimized to limit the machine overfitting, through the minimization of its weight norm. Given the primal estimator of Eq. (1), the primal criterion is then to minimize

$$\begin{aligned} &L_p(\mathbf{w}, b.\xi_n, \xi_n^*) = \|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n + \xi_n^* \\ &s.t.: \begin{cases} y_n - \mathbf{w}^{\top}\phi(\mathbf{x}_n) + b \le \varepsilon + \xi_n \\ -y_n + \mathbf{w}^{\top}\phi(\mathbf{x}_n) + b \le \varepsilon + \xi_n^* \\ \xi_n, \xi_n^* \ge 0 \end{cases} \end{aligned} \tag{5}$$

where ε is a given error tolerance and C is a tradeoff parameter between the minimization of the capacity of the machine (structural risk) and the minimization of the error (empirical risk). This error is represented by nonnegative slack variables $\xi_n.\xi_n^*$. If the error is positive, the excess error over the tolerance is represented by ξ_n, and for negative errors, by ξ_n^*. If one is positive, the other one is forced to be zero. A Lagrange optimization with multipliers $\alpha_n, \alpha_n^*, \mu_n, \mu_n^*$ over each one of the four sets of constraints in functional (5) leads to the dual functional.

$$\begin{aligned} &L_d(\alpha_n, \alpha_n^*) = -\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right)^{\top}\mathbf{K}\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right) + \left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right)^{\top}\mathbf{y} + \varepsilon\mathbf{1}^{\top}\left(\boldsymbol{\alpha} + \boldsymbol{\alpha}^*\right) \\ &s.t.\ \ 0 \le \alpha_n, \alpha_n^* \le C \end{aligned} \tag{6}$$

where $\mathbf{1}$ is a column vector of N ones and $\mathbf{K}$ is the matrix of kernel dot products (2) between training data. The estimator is the expressed as

$$y_n = \left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right)^{\top}\mathbf{k}(\mathbf{x}_n) + b \tag{7}$$

If the error corresponding to training sample $\mathbf{x}_m$ is positive and higher than ε, then $\alpha_m = C$, $\alpha_m^* = 0$. If it is equal to ε, then $0 < \alpha_m < C$, $\alpha_m^* = 0$, and reciprocally for negative errors. If the error is less than ε, then $\alpha_m = \alpha_m^* = 0$. The training samples with $\alpha_m - \alpha_m^* \neq 0$ are the support vectors. This dual is a quadratic function and the matrix $\mathbf{K}$ is positive definite, thus the functional is the sum of a convex function and a linear function of the dual parameters. Hence, it has a single maximum with respect to the parameters, which can be found using quadratic programming, or following the efficient Sequential Minimal Optimization (SMO) [41].

This machine has the advantage that the capacity of the machine is controlled by parameter C, and cross-validation over this parameter can lead to a good overfitting control. The drawback is that the number of elements of matrix $\mathbf{K}$ is N^2 and, typically, the optimization of this function has a computational time of N^3. It has to be done in a block, this is, it does not admit online approximations, and it is difficult to parallelize. Finally, parameters C, ε, and all the parameters of the kernel function must be cross-validated.

In this subsection, the nonlinear SVM (NLSVM) has been presented, where the nonlinearity is introduced through mapping $\phi(\cdot)$. The linear counterpart of the machine assumes that the transformation is through a rotation and translation matrix $\mathbf{A}$ onto the same space of input data $\mathbf{x}_n$, this is $\phi(\mathbf{x}_n) = \mathbf{A}\mathbf{x}_n$, hence $\mathbf{k}(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^{\top}\boldsymbol{\Sigma}\mathbf{x}_m$ is a linear dot product where $\boldsymbol{\Sigma} = \mathbf{A}^{\top}\mathbf{A}$ is a positive definite matrix, typically defined as an identity. Therefore, the linear counterpart of the SVM optimization is formally identical to the nonlinear one, and kernel matrix $\mathbf{K}$ is the Gram matrix of dot products between training data.

2.4. Gaussian processes (GP)

The parameter validation drawbacks do not exist in GP regression. Instead, we must construct a probabilistic model for the data and the set of primal parameters. The GP assume that the set of parameters $\mathbf{w}, b$ are a latent random variable, whose prior probability distribution is a Gaussian $\mathcal{N}\left(\mathbf{0}, \boldsymbol{\Sigma}_p\right)$ with zero mean and arbitrary covariance matrix $\boldsymbol{\Sigma}_p$. The estimation error in (1) is assumed to be an independent and identically distributed (i.i.d.) Gaussian random variable of zero mean and variance σ_n^2. Therefore, regressor y_n is also a Gaussian random variable. The covariance of this regressor can be computed as

$$\begin{aligned} \mathbb{E}\left[y_n y_m\right] &= \mathbb{E}\left[\left(\mathbf{w}^{\top}\phi(\mathbf{x}_n) + b + e_n\right)\left(\mathbf{w}^{\top}\phi(\mathbf{x}_m) + b + e_m\right)\right] \\ &= \begin{pmatrix} \phi(\mathbf{x}_n) \\ 1 \end{pmatrix}^{\top}\mathbb{E}\left[\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}^{\top}\right]\begin{pmatrix} \phi(\mathbf{x}_m) \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \phi(\mathbf{x}_n) \\ 1 \end{pmatrix}^{\top}\boldsymbol{\Sigma}_p\begin{pmatrix} \phi(\mathbf{x}_m) \\ 1 \end{pmatrix} + \sigma_n^2\delta(n-m) \end{aligned} \tag{8}$$

where $\delta(n-m)$ is the Kronecker delta function, which appears by assuming that the error is drawn from a zero-mean i.i.d. Gaussian distribution. This expression is a positive definite function of $\mathbf{x}_n$, $\mathbf{x}_m$, thus, by the Mercer's theorem, it can be considered a kernel dot product between data and, reciprocally, any Mercer's kernel can be considered a covariance function of y_n.

$$\mathbb{E}\left[y_n y_m\right] = k(\mathbf{x}_n, \mathbf{x}_m) + c + \sigma_n^2\delta(n-m) \tag{9}$$

where c is, from the construction of expectation (8), the last term of covariance $\boldsymbol{\Sigma}_p$. Therefore, the covariance matrix of process $\mathbf{y}$ is

$$\mathbb{E}\left[\mathbf{y}\mathbf{y}^{\top}\right] = \mathbf{K} + \sigma_n^2\mathbf{I} \tag{10}$$

where a matrix of constants $c\mathbf{1}\mathbf{1}^{\top}$ has been included in side matrix $\mathbf{K}$. Now, assume a test sample $\mathbf{x}^*$, different from the training data. The corresponding prediction is $f_* = \mathbf{w}^{\top}\phi(\mathbf{x}*)$. Using the same strategy as in expectation (8), the following covariances can be found:

$$\begin{aligned} \mathbb{E}\left(\mathbf{y}f_*\right) &= \mathbf{k}(\mathbf{x}^*) \\ \mathbb{E}\left(f_*^2\right) &= k(\mathbf{x}^*, \mathbf{x}^*) \end{aligned} \tag{11}$$

By assuming that the joint process $\mathbf{y}, f_*$ is a Gaussian process of zero mean and covariance function

$$\mathbb{E}\left[\left(\mathbf{y}^{\top}f_*\right)^{\top}\left(\mathbf{y}^{\top}f_*\right)\right] = \begin{pmatrix} \mathbf{K} + \sigma_n^2\mathbf{I} & \mathbf{k}^{\top}(\mathbf{x}^*) \\ \mathbf{k}^{\top}(\mathbf{x}^*) & k(\mathbf{x}^*, \mathbf{x}^*) \end{pmatrix} \tag{12}$$

Then the posterior probability of f_* given the training data $\mathbf{X}$, $\mathbf{y}$ can be found, and it is a univariate Gaussian distribution with mean and variance given by the following equations:

$$\begin{aligned}\bar{f}_* &= \mathbf{y}^\top \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{k}(\mathbf{x}^*) \\ \sigma_*^2 &= k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^\top(\mathbf{x}^*) \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{k}(\mathbf{x}^*)\end{aligned} \tag{13}$$

This expression gives a mean and a confidence interval for the prediction and has several free parameters, which are the ones of the kernel function, including c, and the error variance σ_n^2. These parameters are optimized by maximizing the logarithm of the likelihood function of y for these parameters. Hence, this criterion does not need parameter-based cross-validation, but a more efficient gradient ascent algorithm over the likelihood. In exchange, it needs a Gaussian model of the error. The summarized GP algorithm is linear, the nonlinear version of this procedure is obtained by simply assuming that $\phi(\mathbf{x}_n) = \mathbf{A}\mathbf{x}_n$ where in this case, $\mathbf{A}^\top\mathbf{A} = \sigma_p$ is represented by the prior covariance matrix in Eq. (8).

2.5. Recurrent neural networks (RNN)

A block of an RNN is shown in Fig. 1(a). The block represents a neural network of one layer with connections $\mathbf{W}_r$ and biases $\mathbf{b}_r$, whose input is $\mathbf{x}_n$ concatenated with the output $\mathbf{h}_n$ of a previous RNN block. The result of the product of the inputs with the weight vector plus the bias is passed through a nonlinear activation to produce hidden output $\mathbf{h}_{n+1}$ [42]. The equations of the recurrence are

$$\mathbf{h}_n = \sigma\left(\mathbf{W}_r^\top \begin{bmatrix} \mathbf{x}_n \\ \mathbf{h}_{n-1} \end{bmatrix} + \mathbf{b}_r\right) \tag{14}$$

$$\mathbf{o} = \sigma\left(\mathbf{W}_0^\top \mathbf{h}_n + \mathbf{b}_0\right) \tag{15}$$

where $\sigma(\cdot)$ is an array of sigmoidal functions, and $\mathbf{o}$ represents the output of the network. The new state $\mathbf{h}_{n+1}$ contains information of the state produced by the old sample in order to use the temporal structure of the data.

A log likelihood function can be used as a cost function for optimization purposes as

$$J_{ML}(\theta) = -\frac{1}{N} \sum \log p(\mathbf{x}_n | \mathbf{y}_1, \ldots \mathbf{y}_n) \tag{16}$$

where $\mathbf{y}_n$ is the desired output at instant n, which is assumed to be dependent on the sequence of inputs $\mathbf{x}_1 \cdots \mathbf{x}_n$. The derivation of the backpropagation algorithm is then similar to that of a standard feedforward neural network, where the updates are simply

$$\begin{aligned}\nabla_{\mathbf{b}_0} J_{ML} &= \sum_n \nabla_{\mathbf{o}_n} J_{ML} \\ \nabla_{\mathbf{b}_r} J_{ML} &= \sum_n \left(\mathbf{1} - \mathbf{h}_n \otimes \mathbf{h}_n\right) \otimes \nabla_{\mathbf{h}_n} J_{ML} \\ \nabla_{\mathbf{W}_o} J_{ML} &= \sum_n \nabla_{\mathbf{o}_n} J_{ML} \mathbf{h}_n^\top \\ \nabla_{\mathbf{W}_r} J_{ML} &= \sum_n \left(\mathbf{1} - \mathbf{h}_n \otimes \mathbf{h}_n\right) \otimes \nabla_{\mathbf{h}_n} J_{ML} \begin{pmatrix} \mathbf{x}_n \\ \mathbf{h}_n \end{pmatrix}\end{aligned} \tag{17}$$

$\otimes$ being the elementwise product operator and $\mathbf{1}$ is a vector of ones. The gradient $\nabla_{\mathbf{h}_n} J_{ML}$ with respect to the hidden output $\mathbf{h}_n$ is computed recursively as

$$\nabla_{\mathbf{h}_n} J_{ML} = \mathbf{W}_{r,h} \nabla_{\mathbf{h}_{n+1}} J_{ML} \otimes \left(\mathbf{1} - \mathbf{h}_{n+1} \otimes \mathbf{h}_{n+1}\right) + \mathbf{W}_0^\top \nabla_{\mathbf{o}_n} J_{ML} \tag{18}$$

where $\mathbf{W}_{r,h}$ is the part of $\mathbf{w}_r$ in Eq. (14) that multiplies to the hidden output $\mathbf{h}_n$. Last, we need to compute gradient $\nabla_{\mathbf{o}_n} J_{ML}$. We previously assumed that a negative log likelihood function is used as cost function. Thus, we must assume that all elements of output $\mathbf{o}_n$ are passed though a softmax function to produce outputs $\hat{y}_{i,n} = \frac{o_{i,n}}{\sum_j o_{j,n}}$, and the gradient is simply

$$\nabla_{\mathbf{o}_n} J_{ML} = \hat{\mathbf{y}}_n - \mathbf{y}_n \tag{19}$$

where $\mathbf{y}_n$ is the vector of desired outputs $y_{i,n}$ at instant n.

2.6. Long short term memory (LSTM) networks

LSTMs were introduced by Hochreiter & Schmidhuber (1997), to address the problem of long-term dependencies [43]. The architecture of LSTM, shown in Fig. 1(b) represents a type of RNN which are explicitly modeled for long-term temporal sequences. Unlike conventional RNNs, the recurrent hidden layer of the LSTM comprises blocks named memory cells. These blocks are connected in the form of a chain, which enables them to pass significant information sequentially across the network. The control of information that goes in and out of the memory cells is done by special units called gates. The gates are modeled to add relevant information and remove redundancies. There are three gates present in each memory cell namely the input gate, forget gate and the output gate [44].

In an LSTM, the primary decision is made by a sigmoid layer named forget gate. It analyzes the information from the previous hidden layer $\mathbf{h}_{n-1}$ and the present input $\mathbf{x}_n$ and helps to select the values that needs to kept and those that needs to be forgotten.

$$\mathbf{f}_n = \sigma(\mathbf{W}_{fh}\mathbf{h}_{n-1} + \mathbf{W}_{fx}\mathbf{x}_n + \mathbf{b}_f) \tag{20}$$

In the next step the cell state is updated using a sigmoid layer and a tanh layer. The input gate acts as the sigmoid layer and takes in the previous hidden state $\mathbf{h}_{n-1}$ and the current input $\mathbf{x}_n$ to output values between 0 and 1. The same input is further passed through the tanh layer to generate values between -1 and 1 ($\tilde{\mathbf{c}}_n$). The output from the input gate and tanh layer are multiplied to produce an update to the cell state.

$$\mathbf{i}_n = \sigma(\mathbf{W}_{ih}\mathbf{h}_{n-1} + \mathbf{W}_{ix}\mathbf{x}_n + \mathbf{b}_i) \tag{21}$$

$$\tilde{\mathbf{c}}_n = \tanh(\mathbf{W}_{ch}\mathbf{h}_{n-1} + \mathbf{W}_{cx}\mathbf{x}_n + \mathbf{b}_c) \tag{22}$$

In order to calculate the new cell state $\mathbf{c}_n$, the forget gate output is multiplied to the old cell state and the output is added with $\mathbf{i}_n \otimes \tilde{\mathbf{c}}_n$.

$$\mathbf{c}_n = \mathbf{c}_{n-1} \otimes \mathbf{f}_n + \mathbf{i}_n \otimes \tilde{\mathbf{c}}_n \tag{23}$$

Finally, the output gate is used to determine the next hidden state. In the first step, it takes in the previous hidden state and present input and passes it through a sigmoid layer. Next, the new cell state is passed through a tanh layer and this output is multiplied to the output from the sigmoid layer to determine the information present in the hidden layer [45].

$$\mathbf{o}_n = \sigma(\mathbf{W}_{oh}\mathbf{h}_{n-1} + \mathbf{W}_{ox}\mathbf{x}_n + \mathbf{b}_o) \tag{24}$$

$$\mathbf{h}_n = \mathbf{o}_n \otimes \tanh(\mathbf{c}_n) \tag{25}$$

Here $\mathbf{W}_{fh}, \mathbf{W}_{fx}, \mathbf{W}_{ih}, \mathbf{W}_{ix}, \mathbf{W}_{ch}, \mathbf{W}_{cx}, \mathbf{W}_{oh}, \mathbf{W}_{ox}$ represent the weights corresponding to the hidden and input layers of different gates, $\mathbf{b}_f, \mathbf{b}_i$, $\mathbf{b}_c$ and $\mathbf{b}_o$ denote the biases.

2.7. Gated recurrent unit (GRU) networks

GRU is a variant of LSTM introduced by Kyunghyun Cho in 2014 [46]. Unlike LSTM, GRU does not have a separate memory cell and hence has lesser parameters during training [47]. But GRU has multiple gates that help to regulate the information flow. The representation block and the generated final output h_n is shown in Fig. 1(c).

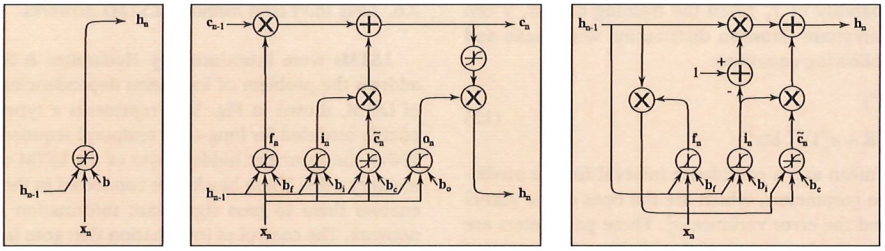
$$\mathbf{i}_n = \sigma(\mathbf{W}_{ih}\mathbf{h}_{n-1} + \mathbf{W}_{ix}\mathbf{x}_n + \mathbf{b}_i) \tag{26}$$

$$\mathbf{f}_n = \sigma(\mathbf{W}_{fh}\mathbf{h}_{n-1} + \mathbf{W}_{fx}\mathbf{x}_n + \mathbf{b}_f) \tag{27}$$

$$\tilde{\mathbf{c}}_n = \tanh(\mathbf{W}_{cx}\mathbf{x}_n + \mathbf{W}_{ch}(\mathbf{f}_n \otimes \mathbf{h}_{n-1}) + \mathbf{b}_c) \tag{28}$$

$$\mathbf{h}_n = (1 - \mathbf{i}_n) \otimes \mathbf{h}_{n-1} + \mathbf{i}_n \otimes \tilde{\mathbf{c}}_n \tag{29}$$

Here $\mathbf{i}_n$ denotes the update gate which helps the network to control the amount of information from the past that needs to be kept and passed on to the future. On the other hand, the reset gate $\mathbf{f}_n$ helps to combine the current input $\mathbf{x}_n$ with the previous memory. Hence both these gates are trained in order calculate $\mathbf{h}_n$ which either maintain all the past information or remove information insignificant for the prediction.



(a) RNN (b) LSTM (c) GRU

Fig. 1. Basic blocks of RNN, GRU and LSTM.

3. Image processing based methods

CNN structures are very popular and successful in their applications to image processing and, therefore, we only consider these structures in the present study. CNNs are a class of neural networks that were widely used for computer vision tasks [48]. Unlike the multilayer perceptron (MLP), the CNNs demonstrated that it was able to extract low-level features from the input layers and progressively map them into high-level features. These features were trained and optimized to produce excellent performance in applications such as image recognition, speech processing, video analysis, and neural style transfer due to their ability of weight sharing [49]. Feature learning, weight sharing, receptive fields, and downsampling are the main elements that make the network tolerant to translation variations. The convolutional layers constitute the basic building block of the CNN architecture. These layers scan an input image I and produce output feature maps by performing the convolution operation using the filters. The two hyperparameters of the network include the filter size and stride. The convolution operation is represented using the following equation.

$$\mathbf{U} = \mathbf{I} * \mathbf{F} \tag{30}$$

Here U, I, and F denote the convolution layer output, input matrix, and filter. Next, the pooling layer takes the output from the convolutional layer and helps to reduce the dimensionality of the feature maps. It sums up the information and downsamples the feature maps. The most widely used pooling layers perform either average pooling or max pooling. In average pooling, the entire elements present in the feature maps are averaged to obtain the output, whereas in max-pooling the maximum value of a region R_{ij} is determined from all the elements. The max-pooling and average pooling operations are illustrated using the below equations.

$$\mathbf{Y}_{kij} = \max_{p,q \in \mathbf{R}_{ij}} \mathbf{A}_{kpq} \tag{31}$$

$$\mathbf{Y}_{kij} = \frac{1}{|\mathbf{R}_{ij}|} \sum_{p,q \in \mathbf{R}_{ij}} \mathbf{A}_{kpq} \tag{32}$$

Here $\mathbf{Y}_{kij}$ is the output of kth feature map at (i,j) whereas $\mathbf{A}_{kpq}$ is the input at (p, q) within the region R_{ij}. Finally, the fully connected layers are used towards the end of the CNN architecture. This layer has a defined set of neurons that takes in a flattened input and outputs a vector. The fully connected layer aggregates the data extracted from the preceding convolutional and pooling layers to generate the desired output [50].

$$\mathbf{H}_k = \sum_{l=1}^{m} \mathbf{W}_{kl}\mathbf{S}_l + \mathbf{B}_k \tag{33}$$

In the above equation $\mathbf{S}_l$ is the flattened input vector of length m and $\mathbf{H}_k$ is the output of length k. Further, the weight matrix and bias are denoted by $\mathbf{W}_{kl}$ and $\mathbf{B}_k$ respectively. To add nonlinearity to the network, activation functions are used after the convolutional, pooling, and fully connected layers. This work uses the Exponential Linear Unit (ELU) [51] as the activation function since it does not suffer from dying neurons, vanishing, and exploding gradients. ELU is continuous and differentiable at all points and if the input value is positive it outputs the same value. But, when the input is less than zero, the output depends on a parameter α and the input value. The mathematical formulation of the ELU activation function is represented as follows:

$$ELU(z) = \begin{cases} z & , z \geq 0 \\ \alpha(e^z - 1) & , z < 0 \end{cases} \tag{34}$$

3.1. Multi modal fusion methods

The proposed model is a multimodal deep learning technique that combines CNN and LSTM for solar radiation micro forecast. In multimodal learning, the information from various sources is combined to create an efficient system. It also contributes to better feature extraction from different sources. In this work, two channels of information such as image and time series data are used to identify the patterns for improving the performance of prediction. Here feature extraction from different modalities is independent of one another, hence a parallel system is created so that CNN can extract the image features whereas the LSTM can extract the time-series information. Finally, the extracted features are integrated to form a shared representation for prediction.

The overall structure consists of 3 modules namely: (i) Image feature extraction block (ii) Time series feature extraction block (iii) Multimodal feature fusion and the prediction block. Fig. 2 represents a structure which inputs a single image whereas Fig. 3 inputs time distributed images. This fusion network consists of 16 layers that combine both image and time-series information to predict the desired output. The image feature extractor block is a conventional CNN with 4 convolutional layers, 4 max-pooling layers, 1 dropout layer, and 2 fully connected layers (dense layer). This block inputs and processes infrared images of the sky of size 60 × 80. The first convolutional layer has 16 filters of size 3 × 3, followed by the ELU activation function. The number of filters in the following convolutional layers is lower. The first convolutional layer extracts more complex high-level features but as it gets deeper into the layers, the network learns more features and hence we increase the number of filters by two times as compared to the previous layers. Thus, there are 32, 64, and 128 filters in the succeeding convolutional layers. Next, we add the max-pooling layer of size 2 × 2 to downsample the features. This model utilizes a smaller filter size for each of its layers since the extracted features are highly localized and do not provide a generic representation of the image. But this helps the network to learn complex and more nonlinear features. Further, a smaller filter size is computationally more efficient since these filters have a lesser number of weights. Hence the succeeding convolutional layers and pooling layers also use filter sizes of 3 × 3 and 2 × 2 respectively. But the number of filters in the following convolutional layers varies since the initial layer extracts complex high-level features. The overfitting of the model is prevented by including a dropout layer after flattening the output from the final pooling layer. Hence a 20% dropout is added so that it acts as a regularizer and randomly drops out the features during each update of the training phase. Finally, the

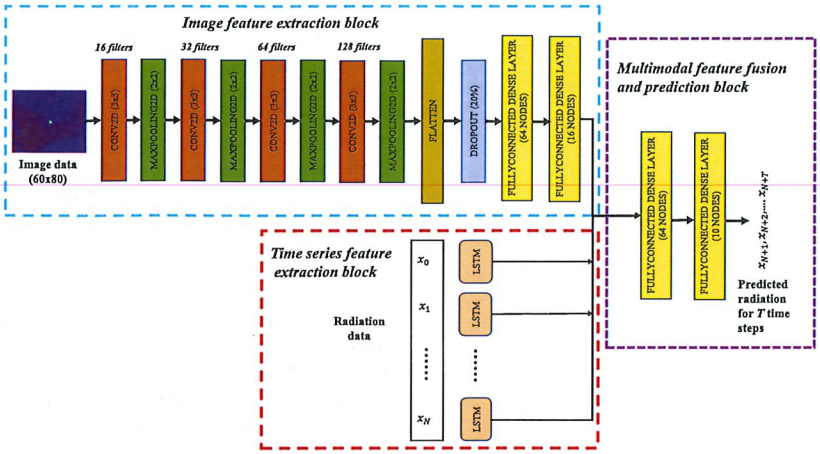


Fig. 2. The network architecture for a convolutional long short term memory fusion network (CNN-L) which uses a single image.

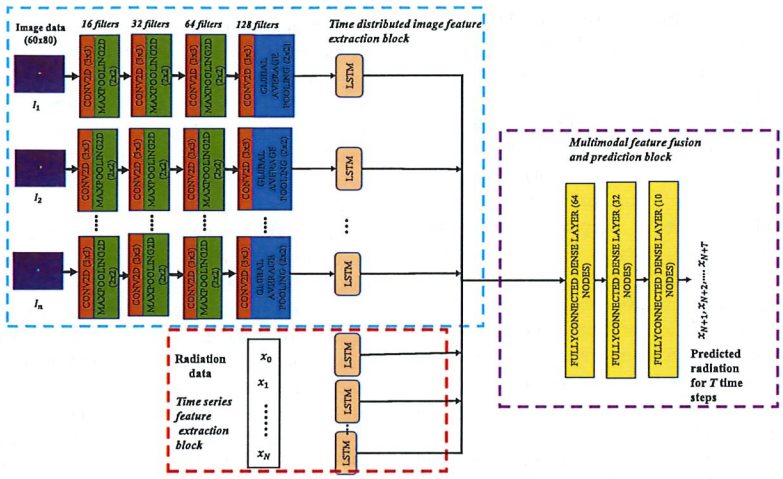


Fig. 3. The network architecture for a multiple image convolutional long short term memory fusion network(MICNN-L).

extracted features are passed through two fully connected layers of sizes 64 and 16 to output a feature vector that represents the most significant information from the infra-red image.

Next, the time series feature extraction block consists of a single layer LSTM which inputs 10 past timesteps of solar radiation values. Then the outputs from the image feature extraction block and time series feature extraction block are concatenated to combine the visual features with the 1-dimensional time-series features. This operation is performed inside the multimodal feature fusion and prediction block, which uses the feature concatenation layer to transform the features in different columns into a single column. Further, this feature vector is passed through 2 fully connected layers with 64, and 10 nodes to generate the final output forecast. This block also uses the ELU activation function on the first fully connected layer whereas the final dense layer uses the linear activation function to predict the desired solar radiation value. In the case of single-step prediction, we use only 1 dense layer at the end of the multimodal feature fusion and the prediction block. But for a multistep forecast, we increase the size of the dense layer to 10, to make predictions for longer time horizons. The parameters of the proposed model are shown in Table 1

Another special case implemented after inspiration from the above network is shown in Fig. 3. The MICNN-L as the name suggests input a sequence of images compared to a single image as in the case of CNN-L. The image feature extraction block in this system is modified to process time-distributed images. The network uses parallel convolutional blocks to extract the temporal features. In this case, we have a set of images with clouds captured every 15 s. The motion of the clouds across the sky over time helps to identify whether they are occluding the sun or not. This helps to aid the predictive analysis since the time series feature extraction block does not provide any information related to the clouds. In comparison to the CNN-L model, this network passes each image through 4 convolutional layers. Following this, global average pooling is performed to encapsulate the features present in the image and reduce the dimensionality of the feature maps. Further, the image feature extraction block is completed by adding LSTM cells towards the end to maintain the time dependencies in the image sequences. The time series feature extraction block on the other hand is similar to the one represented in Fig. 2. Finally, features from both the blocks are combined and pass into the fully connected layers. It was found that the network with multiple image inputs shows comparable performance to the one which inputs a single image. Here the structure was tested using a maximum of 5 image sequences for the multiple-image case. Overall, this network was implemented to incorporate the visual features from the sky and the clouds to accurately predict solar radiation.

4. Transfer learning for image classification

Transfer learning uses the knowledge learned from the source domain to construct a target model. In transfer learning, a pre-trained

Table 1
The model architecture and parameter description for the CNN-L network.

Model	Output size	Number of Parameters	Activation Function
Convolution 2D	(60,80,16)	448	ELU
Maxpooling 2D	(30,40,16)		
Convolution 2D	(30,40,32)	4640	ELU
Maxpooling2D	(15,20,32)		
Convolution 2D	(15,20,64)	18,496	ELU
Maxpooling 2D	(7,10,64)		
Convolution 2D	(7,10,128)	73,856	ELU
Maxpooling 2D	(3,5,128)		
Flatten	(1920)		
Dropout	(1536)		
Dense layer	(64)	122,944	ELU
Dense layer	(16)	1040	Linear
LSTM	(10,1)	480	ELU
Concatenate	(26)		
Dense layer	(64)	1728	ELU
Dense layer	(10)	650	Linear

model is created by training the network on a large-scale benchmark dataset with multiple categories. The weights from these pre-trained models are extracted and used by another network, so that it does not require its model to be trained from scratch. The main advantage of transfer learning over traditional deep learning is that it does not require a large dataset for training and hence its computational burden is less. Additionally, the first layers of the network use the pre-trained weights and it only needs to learn the weights of the last layers.

The MobileNet architecture consists of depthwise separable convolutions, which are formed by factorizing a standard convolution into depthwise and pointwise convolution. The 3×3 depthwise convolution uses a single filter for each input channel, whereas the 1×1 pointwise convolution combines the outputs of the depthwise convolution. The architecture of MobileNetV1 [52] does not use max-pooling, but each layer is followed by an activation function and batch normalization. Towards the end of the network there exists an average pooling layer. It is followed by a fully connected layer that uses softmax for doing the classification. The recently introduced MobilenetV2 also has an expansion layer, called a depthwise convolution layer, and a projection layer. The expansion layer consisting of 1×1 convolutions to expand the number of channels in the data before passing it to the depthwise convolutional layer. Consequently, the output of this layer is given to the projection layer, which uses pointwise convolutions to reduce the dimensionality of the data. A residual connection was also newly introduced in the MobileNetV2 structure. During the performance analysis between the two versions, the MobilenetV2 showed better performance with a lesser number of parameters. Due to this extra speed gain, MobileNetV2 eliminated the resource liabilities for mobile devices. Hence these models are widely used for different computer vision tasks such as image classification, and object detection [53,54].

5. Experiments

5.1. Study area and data processing

The data used for the experiments described below were collected using a camera and a pyranometer placed at the roof of the Mechanical Engineering Building of the University of New Mexico, located at the center of Albuquerque, NM, United States of America. The climate of Albuquerque is arid semi-continental, with few rains, more likely in the summer months. The city center is at 1500/1600 m (4900/5200 ft). Between mid May and mid June, the sky is clear or partly cloudy during the 80% of the time. Approximately, 170 days of the year are sunny, with less than 30% of cloud coverage, and 110 are partly sunny, with 40% to 80% cloud coverage. Temperatures range from an minimum of −4 °C in winter to a maximum of about 33 °C in summer. The collected rain and snow are about 11 inches per year.

The proposed Multi-Modal CNN LSTM fusion network for solar radiation forecasting uses both images as well as time-series information. These data are obtained simultaneously. A system captures the circumsolar IR images whereas another system estimates the Global Horizontal Irradiance (GHI) using a pyranometer. A solar tracker is been used in the Data Acquisition system (DAQ) to maintain the position of the sun at the center and to update its pan and tilt every second. The IR sensor yields a uniform thermal image. It has a wavelength from 8 to 14 μm and it is equipped with a FLIR Lepton® camera with radiometry [55,56]. Every 15 s, a sequence of 10 consecutive images are recorded and averaged in order to reduce the noise, and stored in png format. The pyranometer signal is filtered using an analog antialiasing filter adjusted to allow a sample rate of 4 to 6 samples per second. The data is averaged in windows of one second and stored. Before using this signal for training purposes, the data is digitally filtered to provide antialiasing and noise reduction, and further subsampled to a sample rate of one sample every 15 s. The dataset con be downloaded from repository [57].

5.2. Training

The main aim of the proposed method is to predict solar radiation by using the infra-red images of the clouds. The past information from the pyranometer readings is not sufficient to accurately predict the solar radiation during the following scenarios: (i) When the clouds occlude the sun suddenly during a sunny day. (ii) During a completely cloudy day. In these cases, the visual information of the clouds can be used as additional information along with the pyranometer readings to reduce the prediction error. The clouds moving towards the sun show a high probability of occluding the sun. Hence a classifier based on transfer learning was developed to distinguish between sunny and cloudy days. Following this, the proposed model was designed to use the perceptual information from the clouds along with the past values of the pyranometer. A multi-step model was created and applied on a large dataset of sky images, and forecasts were made up to 10-time steps ahead. GHI measurements from the physical model were used for evaluating the forecast performance.

Different experiments were conducted to evaluate the performance of the network. At first, cloudy days were classified using MobileNetV2 and this data was passed on the proposed network as input. In this application of classifying cloudy days, MobileNetV2 outperforms other transfer learning models such as VGG16, InceptionV3, and ResNet50. The data used for this experiment consist of the IR images of both cloudy and sunny time instances. During the training, the base model is created from the pre-trained MobilenetV2. The initialization of the base model is done by inputting the size of the images, that is 60×80. In this approach, only the last layers undergo retraining, resulting in an accelerated training speed. Finally, the model is trained on a small dataset by using the same weights from Imagenet. Transfer learning extensively helps in enhancing the generalization performance of the model in the target domain. Nevertheless, transferring information from an unrelated source could result in a negative transfer. This could be overcome by improving the source data quality, target data quality, and reducing the domain divergence. However, in this case, since the learning tasks in both domains are similar, the model was able to accurately classify the images. Further, the overall performance of the classification model was evaluated by plotting the confusion matrix.

Later, the proposed multi-modal fusion networks were trained on cloudy days and tested on cloudy, sunny, and mixed days. The criteria for training involved minimization of the mean square error using the adaptive moment optimization (Adam) [58], though several other optimizers such as Stochastic Gradient Descent (SGD), RMSprop [59], and AdaDelta [60] were analyzed. Adam optimizer showed the best performance. It uses the weighted average of the past gradients and the weighted average of the squares of past gradients to update the weight matrix and calculate the current gradients. Hence this optimizer utilizes

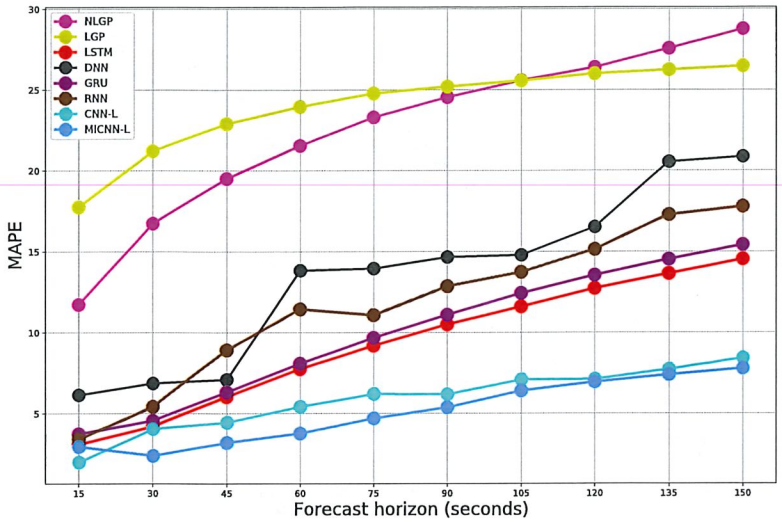


Fig. 4. Comparison of MAPE using different forecasting methods. The previous 10 time steps were used to predict the time horizon ranging from 15–150 s. The training and testing data comprised only cloudy days.

the concept of both Stochastic Gradient Descent with momentum and RMSprop and provides better results than the other commonly used optimizers. Further, no learning rate was specified since Adam has adaptive learning rates. The overall training and optimization were done in batches of 50 for 200 iterations.

Moreover, the experiments were also conducted on eight-time series-based models for comparative analysis. The time series-based solar irradiance forecasting models include LSVM, NLSVM, LGP, NLGP, DNN, RNN, GRU, and LSTM. The dataset used for this experiment includes the GHI values captured from the pyranometer. Here the predictions were made using the past 10 values of the solar radiation. The primary preprocessing of the data involves standardization, which rescales the distribution of values to have mean 0 and standard deviation 1. The LSVM uses a linear kernel whereas the NLSVM uses the radial basis function (RBF) kernel for mapping the data to a higher dimension to help create a better fit for the data. The regularization parameter C is set to 1 in both cases. On the other hand, LGP used a linear kernel whereas the NLGP used a combination of kernels which involved the sum of the linear kernel and Matérn 5/2 kernel [61] with a variance of 1. The combined kernel equation was given as follows:

$$k(r) = \sigma^2(1 + \sqrt{5}r + \frac{5}{3r^2})exp{-\sqrt{5}r} \tag{35}$$

$$k(x, y) = \sigma^2 xy \tag{36}$$

$$K = k(r) + k(x, y) \tag{37}$$

where K is the combined kernel, $K(r)$ is the Matérn 5/2 kernel, $k(x, y)$ is the linear kernel, r is the Euclidean distance between the input points and σ^2 is the variance parameter. In the case of machine learning-based models such as DNN, RNN, GRU and LSTM, the training is done by minimizing the mean square error value by using the Adam optimizer. They are trained during 100 epochs with a batch size of 70 samples. A validation set is created by using 20% of the training data and the best model is selected according to the minimum validation loss. Overall, the experiments were divided and performed on a laptop with NVIDIA GeForce GTX 1060 GDDR5 6.0 GB GPU and supercomputers from The Center for Advanced Research Computing of the University of New Mexico.

6. Results and discussions

The performance of the proposed model was determined by using different evaluation metrics. The main criteria included Mean Absolute Percentage Error (MAPE), Median Absolute Percentage Error (MeAPE), Mean Absolute Error (MAE), Coefficient of Determination (R^2), Root Mean Square Error (RMSE) and t-statistics (t-score). The mathematical representation of these metrics are as follows:

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{\hat{y} - y}{\hat{y}}\right| \tag{38}$$

$$MeAPE = Median(\left|\frac{\hat{y} - y}{\hat{y}}\right| \times 100) \tag{39}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y} - y| \tag{40}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y} - y)^2}{\sum_{i=1}^{N}(\mu_y - y)^2} \tag{41}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y} - y)^2} \tag{42}$$

$$t = \frac{\mu_{\hat{y}} - \mu_y}{\sqrt{\frac{\sigma_{\hat{y}}^2 + \sigma_y^2}{N}}} \tag{43}$$

Here $\hat{y}$, y, μ_y, $\mu_{\hat{y}}$, σ_y^2, $\sigma_{\hat{y}}^2$, N represents the actual output, predicted output, mean value of predicted output, mean value of actual output, variance of predicted output, variance of actual output and number of samples respectively.

6.1. Comparative analysis using MAPE

To measure micro forecast performance using cloudy days, three different experiments were conducted and the findings were verified using MAPE as the evaluation metric. The first experiment used 11,487 samples for training out of which, 20% was used for validation. The data for training and validation included data samples from only cloudy days. In the case of test data, 2419 samples were belonging to cloudy days. The multistep prediction from this data was evaluated by using time series-based methods as well as hybrid methods which used both time series and image data. The time series models included Linear Gaussian process (LGP), Non-linear Gaussian process (NLGP), DNN, RNN, GRU, and LSTM whereas the proposed CNN-L and MICNN-L belonged to the hybrid model. In Fig. 4 the MAPE was calculated for the future 10 time steps and it was observed that the hybrid models outperformed the time series-based models by a significant margin for

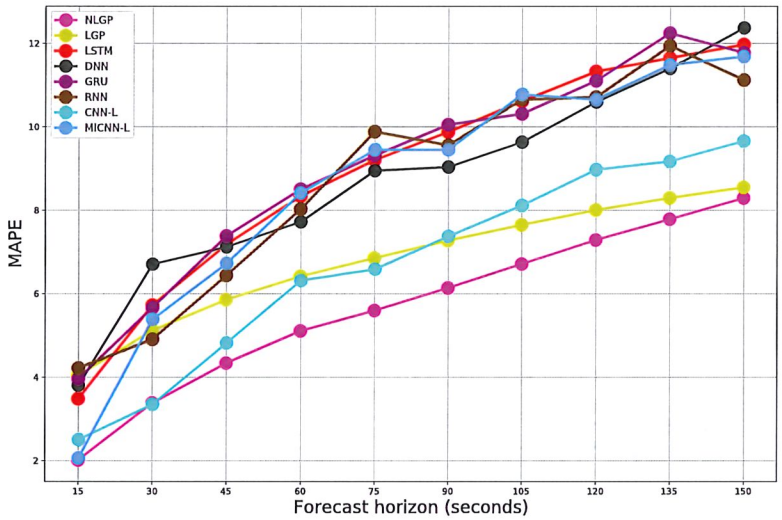


Fig. 5. Comparison of MAPE using different forecasting methods. The previous 10 time steps were used to predict time horizons ranging from 15–150 s. The training data consist of cloudy days, and testing data comprises only sunny days.

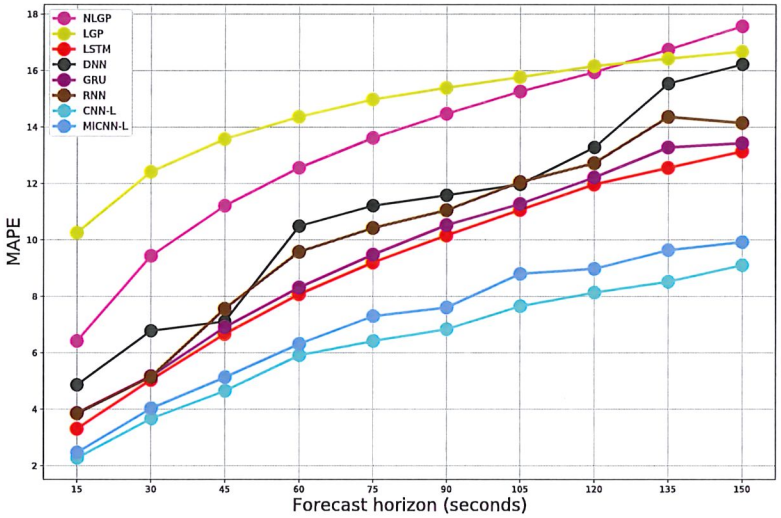


Fig. 6. Comparison of MAPE using different forecasting methods. The previous 10 timesteps were used to predict the time horizon ranging from 15–150 s. Here the training data consist of cloudy days and testing data comprises of both cloudy and sunny days.

higher time horizons. The proposed models were able to extract the most important information from the infra-red images and successfully combine it with the time-series data to predict future radiation values.

In the case of higher horizons, the best performance was shown by MICNN-L with a MAPE of 7.79. Here 5 time distributed images and 10 past values of solar radiation were given as input to the model to generate the desired forecast. The single image hybrid method also showed competitive performance by generating a MAPE of 8.43 for higher horizons. On the contrary, the time-series based methods experienced a drop in performance as the MAPE as from 14.54 to 28.77. Both linear and non-linear Gaussian processes showed an error almost 4 times higher than the one of the proposed models. The DNN, RNN, GRU, and LSTM based models also resulted in a higher error margin. Thus, for cloudy days it was observed that the prediction using only time-series data is almost 2 to 5 times worse than the one using both image and time-series data.

The second experiment involved training using cloudy days and testing using sunny days. Here the same number of cloudy samples were used for training, whereas the test dataset consisted of 3468 samples. The validation was done similarly to the previous experiment and it was observed that most of the methods showed comparable performance for all the time horizons. The lowest MAPE was obtained by NLGP for 150 s, followed by LGP and MICNN-L. The NLGP used a combination of linear kernel and Matérn 5/2 kernel. Further, the training consisted of minimizing the negative log marginal likelihood by using the Scipy optimizer. Thus the trained NLGP model using these parameters were able to make more accurate predictions during the sunny days. The results obtained are shown in Fig. 5.

Finally, the third experiment consists of training with cloudy days and testing using mixed days (both cloudy and sunny days). This illustrates the most likely case in which the model is required to make predictions of a heterogeneous mixture of days. The training data consisted of 11,487 samples and the test data had 5887 samples. During the evaluation, it was found that the image-based methods outperformed the time series-based methods by a higher margin for all the predicted time horizons as shown in Fig. 6. Although the GP based

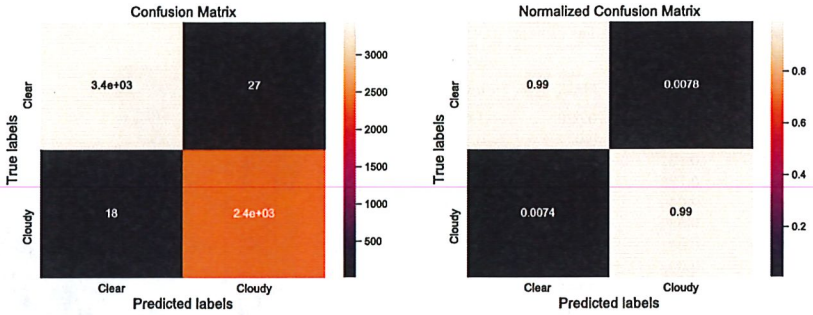


Fig. 7. Confusion matrix for cloud classification using transfer learning (MobileNetV2).

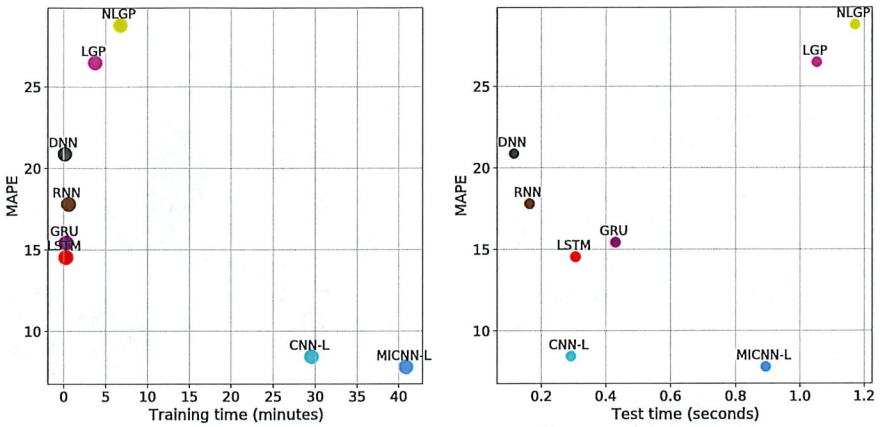


Fig. 8. Cloudy day training and test computational complexity Vs Performance graph for a horizon of 150 s.

methods showed high performance using only sunny days as the test data, it showed a significant degradation in a forecast using mixed days. The information from the past values of solar radiation was found to be insufficient to make error-less forecasts, thereby resulting in higher values of MAPE for the benchmark models.

In order to further improve the reliability of the overall system, MobileNetV2 based classifier was developed to sort the cloudy days from sunny days. The training data consisted of 6191 IR images of both cloudy and sunny days whereas the test data had 5887 images. The classifier has an overall accuracy of 99.23% as shown in the confusion matrix in Fig. 7. Using this system, we are able to group the cloudy images at various time instances and feed into the proposed models for more accurate prediction.

6.2. Training and test time analysis

Fig. 8 shows the training and test time comparisons of several state-of-the-art methods, along with the proposed networks. The running time of all the baseline methods and the proposed multi-modal CNN LSTM fusion networks were calculated on the same machine (laptop with NVIDIA GeForce GTX 1060 GDDR5 and 6.0 GB GPU) whereas the multiple-image CNN LSTM fusion network with more than 5 images was trained on the supercomputer (Nvidia Tesla K40M with 64 GB GPU). But through experimental analysis, it was found that the optimum number of images required for this network was 5. Here, the MAPE was plotted against the running time to illustrate the trade-off between performance and speed. In the case of training time, several algorithms such as DNN, RNN, GRU, and LSTM takes less than 1 min at the cost of a higher MAPE ranging from 14.54 to 20.86. The methods based on GP experiences a further degradation in performance with a training time of 3.73 and 6.75 min for LGP and NLGP, respectively. Further, the same experiments were also repeated using LSVM and NLSVM. But, despite having the least training and test time, these algorithms generated the highest MAPE of 43.95 and 73.06, thereby showing the worst performance.

Our proposed deep learning-based method, on the other hand, shows the best performance, but at expenses of higher training time. The CNN-L and MICNN-L have a training time of 29.50 and 40.82 min, with the best MAPE of 8.43 and 7.79. Although the baseline methods are several times faster than the proposed methods during training, the test time analysis shows that both CNN-L and MICNN-L have a lesser computational burden. Although the number of trainable parameters of the proposed methods was several orders of magnitude higher than the compared methods, their test times are less than 1 s. This makes them equivalently competitive in terms of speed. However, the performance improvement was maximized. Hence, the proposed networks show a good balance in speed and achieve the best prediction with minimal error. The results also suggest that the usage of image information along with radiation data improve performance.

6.3. Quantitative performance analysis

Table 2 illustrates the single-step prediction analysis of the forecasting systems using different evaluation metrics. Here we calculate the MAPE, R^2, RMSE, MAE and t-statistics for the proposed networks and other benchmark models for cloudy days. Primarily, it is observed that the proposed CNN-L and MICNN-L models outperform the time series-based methodologies with the least MAPE of 2.00 and 2.96 respectively. Similarly, the MAE was decreased by 43.75% (0.016 to 0.009) in the case of CNN-L and 31.25% (0.016 to 0.011) for MICNN-L when compared to the benchmark model with the best performance. From the table, it can be further seen that both LGP and NLGP shows the worst performance with higher values for MAPE, RMSE, MAE, and a much lower value for R^2. Additionally, the normal distribution of

Table 2
Study of various network configurations using different evaluation metrics.

METHODS	ANALYSIS METRIC				
	MAPE	R^2	RMSE	MAE	t-test
LSVM	5.57	0.992	0.034	0.020	−1.097
NLSVM	5.66	0.992	0.033	0.022	−0.872
LGP	17.75	0.852	0.132	0.062	−3.428
NLGP	11.73	0.850	0.133	0.064	−0.722
DNN	6.14	0.983	0.040	0.028	−2.505
RNN	3.41	0.992	0.032	0.018	1.075
GRU	3.74	0.992	0.031	0.016	1.375
LSTM	3.09	0.992	0.033	0.017	0.915
CNN-L	**2.00**	**0.993**	**0.030**	**0.009**	**0.390**
MICNN-L	**2.96**	**0.993**	**0.029**	**0.011**	−0.430

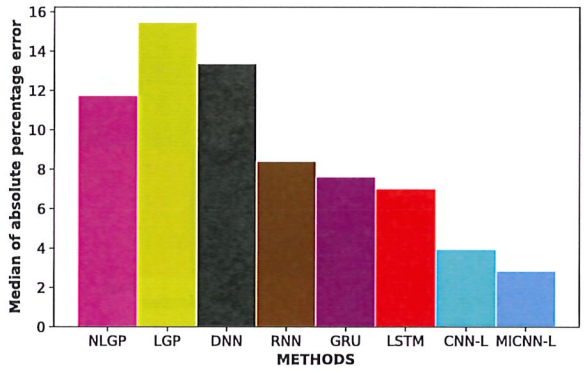
the data was evaluated using a chi-square test. Here the calculated p-value is greater than the alpha value (0.05), and hence we accept the null hypothesis that the data came from a normal distribution. Further, the skewness was 0.5, and kurtosis was -0.6. Consequently, the positive skewness shows that the data is skewed right, and the negative kurtosis is a measure that the data is lightly tailed to a normal distribution. Following these assumptions, the t-test was used to illustrate the statistical difference between the actual and the predicted value. Thus, a smaller t-score indicates that the two categories are similar. Henceforth, in this case, both CNN-L and MICNN-L have the least t-score implying that they show the smallest differences between the distributions of both predicted and actual outputs. But for both LGP and DNN, since the absolute value of the t statistic is greater than the critical value of 1.96 as observed in the t-test table, there exists a higher probability of having statistically different means.

Fig. 9 represents the bar graph corresponding to the multi-step forecasting of different models based on MeAPE for various prediction horizons such as the next 15, 75, and 150 s. It is first observed that the proposed models generate more accurate predictions for all horizons compared to the rest of the time series-based models. Specifically, for higher horizons, the MeAPE value falls under 6 for both CNN-L and MICNN-L which is between 2 to 3 times better than the benchmark networks. Further, for all the models the error value increases as the horizon time increases, implying the decrease in efficiency for higher horizons. However, the proposed models learn the long-term dependencies more efficiently unlike the rest of the models, since they shows a slower increase in error as the prediction time increases.

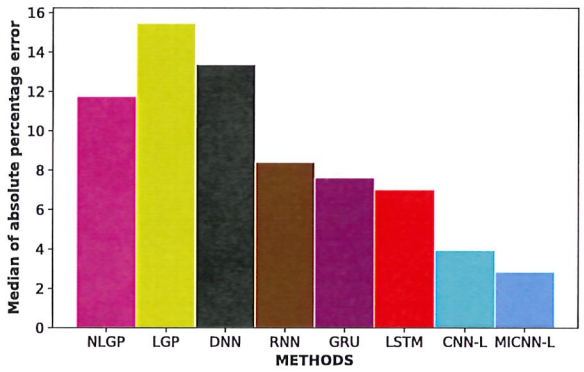
7. Conclusion and future work

Photovoltaic energy is intrinsically stochastic due mainly to cloud coverage. Therefore, forecasting solar radiation is not straightforward, and it needs to be performed at various time horizons, ranging from seconds to hours or days. In the case of micro-forecast, this problem has been extensively tackled by prediction of the solar radiance from historical time series of measured radiance at various time horizons and through the observation of the sky with imaging systems. Though a large quantity of work has been devoted to the use of total sky visible light images for solar radiation forecast, more recently attention has been paid to the use of infrared images.

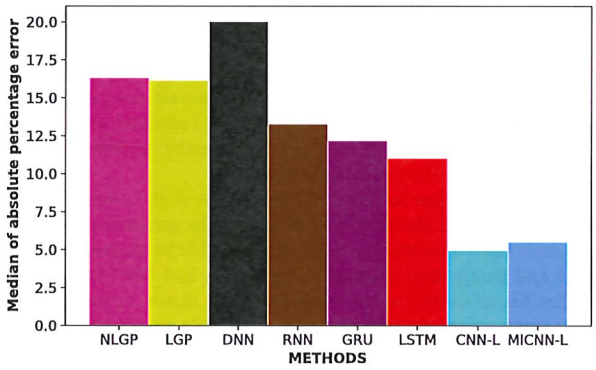
In this work, a methodology that uses information fusion of radiance data and infrared images is presented. Infrared images are more informative than visible images because they convey more information related to the cloud density and they are more invariant to the sun illumination than visible images [55]. Instead of total sky imaging, a solar tracker is used that produces images of the circumsolar area every 15 s. The data also includes the time series of the solar radiation both as an input and as a desired forecast output. The method is intended to perform information fusion of a window of past samples of the radiation measurements and the images of the sky to produce a micro forecast of future samples of the radiation. The method uses an LSTM that extracts features from the time series, and a CNN that extracts features from the images. These features are then concatenated and processed through a dense neural network to produce a multi horizon forecast ranging from 15 to 150 s. The methodology has been compared to methods that use only the radiation data as an input and methods that use images only. The baseline machine learning techniques include kernel methods as SVM and Gaussian processes, both tested with linear and kernel dot products in their formulations, and deep learning methods, including a standard neural network, an RNN, an LSTM, and a GRU. We also introduce an extension that combines a sequence of images with a sequence of radiation samples. Though this extension shows an increased performance, during training it has a higher computational burden. Further comparisons with other methodologies show that our



(a) 15 s



(b) 75 s



(c) 150 s

Fig. 9. Comparative analysis of Median of absolute percentage error for different algorithms during prediction of future time step (a) 15 s (b) 75 s (c) 150 s.

structure produces the best multi horizon forecast on cloudy days with a computation time of about 0.3 s. The proposed CNN-L and MICNN-L had the minimum error while computing the MAPE, MeAPE, MAE, and RMSE values compared with the rest of the models. Moreover, the data has a normal distribution, and the following t-test shows that both CNN-L and MICNN-L also have the least difference between the distributions of the actual and observed values. In the case of sunny days, the nonlinear GP shows a better performance, but it has a higher computational burden in the test. The best overall performance is shown by the multi-modal fusion networks introduced in this paper. Thus, the use of low-resolution images is sufficient to produce reasonable forecasts at a low computational burden. But in any case, a classifier trained using transfer learning is also developed to distinguish between both sunny and cloudy days, to select the best possible predictor.

The future work will consist of providing intra-hour forecasting using the proposed methods on cloudy days. The detection of clouds moving towards the sun using adaptive convolutional kernels is also a possible direction for the upcoming research. Moreover, different feature extraction methods on the images need to be investigated to improve the reliability of the forecasts.

CRediT authorship contribution statement

Meenu Ajith: Software, Formal analysis, Methodology, Investigation, Writing - original draft, Visualization. **Manel Martínez-Ramón:** Writing - review & editing, Conceptualization, Supervision, Project administration, Resources, Funding acquisition, Data curation, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by NSF EPSCoR, United States grant number OIA-1757207 and the King Felipe VI endowed Chair. Authors would like to thank the UNM Center for Advanced Research Computing, supported in part by the National Science Foundation, for providing the high performance computing resources used in this work.

References

[1] Ai Y, Peng Y, Wei W. A model of very short-term solar irradiance forecasting based on low-cost sky images. AIP Conf Proc 2017;1839(1):020022.
[2] Ogliari E, Dolara A, Manzolini G, Leva S. Physical and hybrid methods comparison for the day ahead PV output power forecast. Renew Energy 2017;113:11–21.
[3] Yang H, Huang C, Huang Y, Pai Y. A weather-based hybrid method for 1-day ahead hourly forecasting of PV power output. IEEE Trans Sustain Energy 2014;5(3):917–26.
[4] Shi J, Lee W, Liu Y, Yang Y, Wang P. Forecasting power output of photovoltaic systems based on weather classification and support vector machines. IEEE Trans Ind Appl 2012;48(3):1064–9.
[5] Voyant C, Notton G, Kalogirou S, Nivet M-L, Paoli C, Motte F, Fouilloy A. Machine learning methods for solar radiation forecasting: A review. Renew Energy 2017;105:569–82.
[6] Hammer A, Heinemann D, Lorenz E, Lückehe B. Short-term forecasting of solar radiation: a statistical approach using satellite data. Sol Energy 1999;67(1–3):139–50.
[7] Reikard G. Predicting solar radiation at high resolutions: A comparison of time series forecasts. Sol Energy 2009;83(3):342–9.
[8] Hontoria L, Rus-Casas C, Aguilar JD, Hernandez JC. An improved method for obtaining solar irradiation data at temporal high-resolution. Sustainability 2019;11(19).
[9] Perez R, Lorenz E, Pelland S, Beauharnois M, Van Knowe G, Hemker Jr K, Heinemann D, Remund J, Müller SC, Traunmüller W, et al. Comparison of numerical weather prediction solar irradiance forecasts in the US, Canada and Europe. Sol Energy 2013;94:305–26.
[10] Chu Y, Pedro HT, Li M, Coimbra CF. Real-time forecasting of solar irradiance ramps with smart image processing. Sol Energy 2015;114:91–104.
[11] Garcia-Hinde O, Gómez-Verdejo V, Martínez-Ramón M, Casanova-Mateo C, Sanz-Justo J, Jiménez-Fernández S, Salcedo-Sanz S. Feature selection in solar radiation prediction using bootstrapped SVRs. In: 2016 IEEE congress on evolutionary computation. IEEE; 2016, p. 3638–45.
[12] a Hinde OG, Terrén-Serrano G, Hombrados-Herrera M, Gómez-Verdejo V, Jiménez-Fernández S, Casanova-Mateo C, Sanz-Justo J, nez Ramón MM, Salcedo-Sanz S. Evaluation of dimensionality reduction methods applied to numerical weather models for solar radiation forecasting. Eng Appl Artif Intell 2018;69:157–67.
[13] Pelland S, Remund J, Kleissl J, Oozeki T, De Brabandere K. Photovoltaic and solar forecasting: State of the art. 2013.
[14] Ahmed R, Sreeram V, Mishra Y, Arif M. A review and evaluation of the state-of-the-art in PV solar power forecasting: Techniques and optimization. Renew Sustain Energy Rev 2020;124:109792.
[15] Yang L, Gao X, Hua J, Wu P, Li Z, Jia D. Very short-term surface solar irradiance forecasting based on FengYun-4 geostationary satellite. Sensors 2020;20(9):2606.
[16] Akhter MN, Mekhilef S, Mokhlis H, Shah NM. Review on forecasting of photovoltaic power generation based on machine learning and metaheuristic techniques. IET Renew Power Gener 2019;13(7):1009–23.
[17] Wang F, Zhen Z, Liu C, Mi Z, Hodge B-M, Shafie-khah M, Catalão JP. Image phase shift invariance based cloud motion displacement vector calculation method for ultra-short-term solar PV power forecasting. Energy Convers Manag 2018;157:123–35.
[18] Wang F, Xuan Z, Zhen Z, Li Y, Li K, Zhao L, Shafie-khah M, Catalão JP. A minutely solar irradiance forecasting method based on real-time sky image-irradiance mapping model. Energy Convers Manage 2020;220:113075.
[19] Kamadinata JO, Ken TL, Suwa T. Sky image-based solar irradiance prediction methodologies using artificial neural networks. Renew Energy 2019;134:837–45.
[20] Wan C, Zhao J, Song Y, Xu Z, Lin J, Hu Z. Photovoltaic and solar power forecasting for smart grid energy management. CSEE J Power Energy Syst 2015;1(4):38–46.
[21] Lipperheide M, Bosch J, Kleissl J. Embedded nowcasting method using cloud speed persistence for a photovoltaic power plant. Sol Energy 2015;112:232–8.
[22] Sobri S, Koohi-Kamali S, Rahim NA. Solar photovoltaic generation forecasting methods: A review. Energy Convers Manage 2018;156:459–97.
[23] Sreekumar S, Bhakar R. Solar power prediction models: Classification based on time horizon, input, output and application. In: 2018 International conference on inventive research in computing applications, 2018. p. 67–71.
[24] Dong J, Olama MM, Kuruganti T, Melin AM, Djouadi SM, Zhang Y, Xue Y. Novel stochastic methods to predict short-term solar radiation and photovoltaic power. Renew Energy 2020;145:333–46.
[25] Sheng H, Xiao J, Cheng Y, Ni Q, Wang S. Short-term solar power forecasting based on weighted Gaussian process regression. IEEE Trans Ind Electron 2018;65(1):300–8.
[26] Feng C, Zhang J. Hourly-similarity based solar forecasting using multi-model machine learning blending. In: 2018 IEEE power energy society general meeting, 2018. p. 1–5.
[27] Dolara A, Leva S, Mussetta M, Ogliari E. PV hourly day-ahead power forecasting in a micro grid context. In: 2016 IEEE 16th international conference on environment and electrical engineering, 2016. p. 1–5.
[28] Sun Y, Venugopal V, Brandt AR. Short-term solar power forecast with deep learning: Exploring optimal input and output configuration. Sol Energy 2019;188:730–41.
[29] Ryu A, Ito M, Ishii H, Hayashi Y. Preliminary analysis of short-term solar irradiance forecasting by using total-sky imager and convolutional neural network. In: 2019 IEEE PES GTD grand international conference and exposition Asia. IEEE; 2019, p. 627–31.
[30] Feng C, Zhang J. SolarNet: A sky image-based deep convolutional neural network for intra-hour solar forecasting. Sol Energy 2020;204:71–8.
[31] Sun Y, Venugopal V, Brandt AR. Short-term solar power forecast with deep learning: Exploring optimal input and output configuration. Sol Energy 2019;188:730–41.
[32] Zhao X, Wei H, Wang H, Zhu T, Zhang K. 3D-CNN-based feature extraction of ground-based cloud images for direct normal irradiance prediction. Sol Energy 2019;181:510–8.
[33] Kong W, Jia Y, Dong ZY, Meng K, Chai S. Hybrid approaches based on deep whole-sky-image learning to photovoltaic generation forecasting. Appl Energy 2020;280:115875.
[34] Ghimire S, Deo RC, Raj N, Mi J. Deep solar radiation forecasting with convolutional neural network and long short-term memory network algorithms. Appl Energy 2019;253:113541.
[35] Mammoli A, Ellis A, Menicucci A, Willard S, Caudell T, Simmins J. Low-cost solar micro-forecasts for PV smoothing. In: 2013 1st IEEE conference on technologies for sustainability, 2013. p. 238–43.
[36] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018. p. 4510–20.

[37] Aiserman M, Braverman E, Rozonoer L. Theoretical foundations of the potential function method in pattern recognition. Avtomat i Telemeh 1964;25(6):917–36.
[38] Schölkopf B, Herbrich R, Smola AJ. A generalized representer theorem. In: International conference on computational learning theory. Springer; 2001, p. 416–26.
[39] Vapnik V. The nature of statistical learning theory. Springer science & business media; 2013.
[40] Rasmussen CE. Gaussian processes in machine learning. In: Summer school on machine learning. Springer; 2003, p. 63–71.
[41] Platt J. Sequential minimal optimization: A fast algorithm for training support vector machines. Tech. rep. MSR-TR-98-14, Microsoft Research; 1998, p. 21.
[42] Fan C, Wang J, Gang W, Li S. Assessment of deep recurrent neural network-based strategies for short-term building energy predictions. Appl Energy 2019;236:700–10.
[43] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9:1735–80.
[44] Chollet F. Deep learning with python. 1st ed. USA: Manning Publications Co.; 2017.
[45] Kim H, Won CH. Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. Expert Syst Appl 2018;103:25–37.
[46] Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 conference on empirical methods in natural language processing. Doha, Qatar: Association for Computational Linguistics; 2014, p. 1724–34.
[47] Wang F, Yu Y, Zhang Z, Li J, Zhen Z, Li K. Wavelet decomposition and convolutional LSTM networks based improved deep learning model for solar irradiance forecasting. Appl Sci 2018;8(8):1286.
[48] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature Cell Biol 2015; 521(7553):436–44.
[49] LeCun Y, Bengio Y. Convolutional networks for images, speech, and time series. In: The handbook of brain theory and neural networks. Cambridge, MA, USA: MIT Press; 1998, p. 255–8.
[50] Rawat W, Wang Z. Deep convolutional neural networks for image classification: A comprehensive review. Neural Comput 2017;29(9):2352–449.
[51] Clevert D, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear Units (ELUs). In: Bengio Y, LeCun Y, editors. 4th International conference on learning representations, conference track proceedings, 2016.
[52] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. MobileNets: Efficient convolutional neural networks for mobile vision applications. 2017, CoRR abs/1704.04861.
[53] Ajith M, Kurup AR. Pedestrian detection: Performance comparison using multiple convolutional neural networks. In: Perner P, editor. Machine learning and data mining in pattern recognition. Cham: Springer International Publishing; 2018, p. 365–79.
[54] Kurup A, Soliz P, Nemeth S, Joshi V. Automated detection of malarial retinopathy using transfer learning. In: 2020 IEEE southwest symposium on image analysis and interpretation, 2020. p. 18–21.
[55] Terrén-Serrano G, Martínez-Ramón M. Data acquisition and image processing for solar irradiance forecast. 2020, arXiv:2011.12401.
[56] Terrén-Serrano G, Bashir A, Estrada T, Martínez-Ramón M. Girasol, a sky imaging and global solar irradiance dataset. Data Brief 2021;106914.
[57] Terrén-Serrano G, Bashir A, Estrada T, Martínez-Ramón M. Girasol, a sky imaging and global solar irradiance dataset. 2021, http://dx.doi.org/10.5061/dryad.zcrjdfn9m, Dryad, Dataset.
[58] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, arXiv preprint arXiv:1412.6980.
[59] Dauphin Y, De Vries H, Bengio Y. Equilibrated adaptive learning rates for non-convex optimization. Adv Neural Inf Process Syst 2015;28:1504–12.
[60] Zeiler MD. Adadelta: an adaptive learning rate method. 2012, arXiv preprint arXiv:1212.5701.
[61] Genton MG. Classes of kernels for machine learning: a statistics perspective. J Mach Learn Res 2001;2(Dec):299–312.