G53IDS Predicting Solar Irradiance

Submitted 18th April 2019, in partial fulfillment of
the conditions for the award of the degree **BSc Hons Computer Science and Artificial Intelligence with Year in Industry.**

14262501

Supervised by Isaac Triguero

School of Computer Science

University of Nottingham

Word Count: $14,320$

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature ____________________

Date _______ / _______ / _______

Acknowledgements

I would like to express my gratitude to my superior, Dr Isaac Triguero, for his guidance and advice throughout the project. I would also like to thank the team at Elastacloud for their continued support and Sandy May for his invaluable Spark knowledge, without which large portions of the project would not have been possible. Additionally I would like thank to Benjamin Charlton for always saving me a seat in the lab and leading an ear whenever I was stuck. Finally, I would like to thank my parents for their unwavering belief in my ability to succeed.

Abstract

Adoption of renewable energy sources is a key step in curbing the effects climate change. Moving away from traditional energy generation processes presents many challenges. Unlike coal and natural gas many renewable energy sources and specifically solar operate intermittently and have abrupt changes in output. This variability presents a problem both for the generators, trying to meet contractual obligations, but also the grid operators endeavouring to ensure a consistent load. Having a system capable of producing accurate predictions of solar irradiance would be of use to both parties. This project aims to build such a system. Leveraging advancements in both the domains of machine learning and big data processing.

Contents

Background and Motivation

The Intergovernmental Panel on Climate Change (IPPC), an international body studying the effects of climate change, recently released a report outlining what is needed to be done to keep to global climate change below 1.5C. Whilst they say the goal is not yet out of reach, in order to achieve it "rapid, far-reaching and unprecedented changes in all aspects of society" are needed. One of the point's raised by the IPCC is that "renewables are estimated to provide up to 85% of global electricity by 2050" [1]. One issue holding back the adoption of renewables, as an energy source is cost. Energy producers do not want to invest in something that will not make them money [2].

In order to increase investment and adoption of renewable energy, specifically solar, it must be as profitable as possible. Unlike traditional generators, solar generation is variable and peaks during the middle of the day. The weather can also change rapidly causing supply to cut off abruptly [3]. Inaccurately forecasting output can cause the generator to over or under deliver on their commitment to the grid, pushing supply and demand away from equilibrium. In order to maintain stability, the grid operator is forced to intervene, finding alternative energy sources to make up the shortfall or requiring other producers to reduce their output. The cost of this action, the balancing cost, is passed onto the offending producer [4]. Having an accurate prediction of power generation, granular enough to be of use in both the long and short term, enables more effective balancing of the grid and reduces costs to suppliers.

Since the grid, in the short run, usually trades in 30 minute blocks having a prediction only a few hours into the future can be enough to make an impact [4]. Reliably announcing their future output producers can avoid incurring production penalties and the grid manager can more easily balance the grid.

At their core, all solar based power generation technologies convert solar radiation into power. Given perfect generation circumstances power output, for a photovoltaic (PV) panel, can simply be thought of as a function of solar irradiance and system efficiency. This is often referred to as clear sky (see Section 1.2). The main source of variability in power output is the amount of solar irradiance that falls on the generator [5]. Some of the biggest factors that affect irradiance (excluding constants such as time of day, day of year and latitude) are atmospheric / meteorological. Having an accurate picture of the weather, cloud location, density, direction and speed of movement would help facilitate the creation of a predictive solar irradiance model.

Today commercial solar farms are continuously producing vast amounts of data. Accurate measures of solar irradiance, power output as well as detailed telemetry for each component. This data is usually high resolution, often sub minute and can be streamed. This can amount to 100,000s of data points per plant being generated every day. The volume and velocity of data can necessitate the use of "big data" frameworks such as Apache Spark [6].

Many operators already leverage optimisation and machine learning techniques for uses such as panel layout, predictive maintenance and rudimentary output forecasts [7–9]. Other technologies such as thermal cameras are also employed for fault detection [10]. The fusion of data already being collected by the operator with external sources could facilitate the creation of state of the art models for production forecasting and use optimisation.

Additionally, the prevalence of home solar installations has increased in recent years. Not only are domestic solar arrays also equipped with monitoring technologies similar to those of commercial solar installations, there are also smart meters providing an accurate picture of energy usage at the household level (also enabling variable tariffs for both buying and selling energy) [11]. Battery technology is improving and its use become more prevalent. A host of "connected" high energy draw devices such as fridges, tumble dryers, and electric cars are also available to consumers.

As previously stated, a major factor in solar production (and household energy use) is the weather. It is fairly easy to access highly granular and localised weather data for historical actuals, current observations and forecasts. Commercial providers such as Weatherbit, Dark Sky and OpenWeatherMap claim to offer accurate observations for virtually any point on the planet [12–14]. One can also access rich radar and satellite imagery covering large areas of the globe available at various update frequencies (our source provides updates every three hours) [14, 15]. The type of data discussed lends itself very well to machine learning (ML). The vast majority of the data is time series, meaning there is temporal sequence, it can be possible to show correlations between variables (i.e cloud cover and solar irradiance). There are many ML approaches that have been show to work leveraging the sequential properties of this kind of data to make predictions [16–18].

There are various examples in the literature of people applying ML techniques to predict solar irradiance. Paoli et al. show that use of traditional artificial neural networks (ANNs) perform comparably to more conventional statistical approaches specifically ARIMA, Bayesian inference and Markov chains [19]. Alzahrani et al. have demonstrated the feasibility of deep learning, in the form of a recurrent neural network, to forecast solar irradiance at sub second resolution [20]. Marquez et al. have used meteorological data from the US National Weather Services and an ANN to predict solar irradiance up to 6 days out [21].

Thanks to an industry partner, Elastacloud, access to a broad range of data is possible. Metrics from numerous solar installations across the United Kingdom, both industrial scale solar farms and home solar installations. Additionally, access to a repository of rich weather data as described above is provided. Perhaps most importantly, an industry model has been provided to serve as a baseline for the proposed work. The model makes use of commercial weather forecasts and gradient boosted trees to predict irradiance (see Section 4.2).

1.1 Aims and Objectives

The main aim of the project is to build a model(s) capable of accurately predicting solar irradiance. The predictions should be at a high enough granularity to be of practical use as described above. The irradiance predictions can in turn be used to predict power output of solar generators such as photovoltaic solar farms or home installations, providing a commercial advantage. The performance of the model can be measured against the existing industry model.

Additionally, the model(s) should be efficient, simple and generalise well. The model should not be dependent on forecasts to predict into the future; doing so makes it much harder to provide confidence measures on performance at runtime as the models performance is tied to the forecast accuracy and stability.

The work is split into the following goals:

- Investigate current state of the art approaches for predicting both solar irradiance and cloud movements. Also investigate approaches to image / video frame imputation.
- Design and develop a model to predict future location of clouds from satellite and ground-based observations
- Design and develop a model to interpolate "missing frames" in an image sequence of clouds
- Design and develop a model to predict solar irradiance based on location / movements of clouds
- Combine all 3 models to give a more granular sequence of predictions

1.2 Definitions

This section will briefly clarify the meaning of some of the technical terms used throughout the document.

Irradiance vs Irradiation Solar irradiance is the measure of power per unit area radiated from the sun [22]. Since irradiance is a power metric, it is a continuous measurement ($\frac{W}{m^2}$). Solar irradiation, on the other hand, is the integral of irradiance with respect to time and is measured in $\frac{Wh}{m^2}$. The terms irradiation and irradiance are sometimes used interchangeably. In many use cases over non-instantaneous time scales they can be thought of as functionally equivalent. The raw data used is a measure of irradiation, however for the sake of simplicity it will be referred to as irradiance.

Clear Sky Clearsky represents the total solar irradiance that would radiate to earth without the interference of atmospheric and meteorological effects. It is a function of the position of the sun (solar elevation angle), and altitude [22]. It is often provided as three individual measures of GHI DHI and DNI:

– Direct Normal Irradiance (DNI): The amount of power falling directly on a plane perpendicular to the direction of the sun.

– Defuse Horizontal Irradiance (DHI): The power falling on a flat surface not directly from the sun (e.g. due to scattering and reflection).

– Global Horizontal Irradiance (GHI): The total amount of power falling on a horizontal surface, i.e. a combination of both DNI and DHI.

When referring to clearsky throughout this work, the GHI value will always be used unless stated otherwise. All clearsky values are calculated using the pvlib python library [23].

Forecast Horizon The amount of time into the future for which predicted values are available. With a forecast horizon of length n, at time-step T_0 predict values would be available from T_1 to T_n.

Related Work

In the process of developing a fully functional system there are many issues that need to be addressed. Data has to be efficiently processed and stored. Systems and endpoints have to be set up in order to integrate with existing systems and make data accessible. However, from a research perspective, the project can be split into a few main problems that need to be addressed to reach its goal.

This section will cover the following topics:

- Regression — Predicting Solar irradiance given the state of the weather (Section 2.1)
- Nowcasting — Predicting the weather data in (short term) future time steps (Section 2.2)
- Imputation — Increasing the granularity of the raw data (Section 2.3)
- Embedding — A technique for using categorical data(Section 2.4)

2.1 Regression

Solar irradiance predictions for a given point in time can be thought of as a regression problem. The goal of regression is to create a predictive model, f, that takes as input meteorological state x and outputs a scalar y representing the predicted of solar irradiance. A commonly used approach to build these kind of predictive models are artificial neural networks (ANNs). They are effective a modelling complex relationships and dealing with high dimensionality input data. They can also be expanded in ways to make them better at dealing with both sequential and spacial data.

Section 2.1.1 gives a brief overview of ANNs. More advance neural network approaches are addressed in sections 2.1.2 and 2.1.3. Section 2.1.2 introduces recurrent neural networks, an approach for dealing with sequential data. Section 2.1.3 discusses convolutional neural networks, a technique for dealing with spacial data i.e. images.

2.1.1 Feedforward Neural Networks

A neural network is comprised of multiple nodes called neurons. Although inspired, in part, by the biological neuron they are distinct. Each neuron takes multiple values (x) inputs, performs a weighted sum and applies an activation function on the result to determine the value of its output. The classical ANN is a Feedforward Neural Networks, simply meaning there are no cycles in the network (graph) of neurons. The network is built by combining multiple neurons into layers and stacking layers on top of one another. The first layer (layer 0) takes its values from the input, for layer n the results of the activation functions of all neurons at layer $n-1$ are taken as inputs [24].

Appropriate values for all the weights (ω) of the network are usually "learned" through back propagation. There is a large body of existing work exploiting ANNs to perform regression for various use cases [24].

2.1.2 Recurrent Neural Networks

A major limitation of standard neural networks is their assumption of independence among examples. After an example is processed, all state is lost. This presents a problem if the data is related in time or space. Recurrent neural networks (RNNs) aim to address this issue for time sequence data. By having the ability to "selectively pass information across sequence steps, while processing sequential data one element at a time," [17] they can model non-independent sequences of elements. Simply put, a RNN takes as input both an element from time step T_n as well as its previous sate when processing element from step T_{n-1}.

By taking in its state from pervious time steps RNNs are able to efficiently model sequence and time dependencies as information can be passed from one step to another. This is distinctly deferent from approaches where multiple time steps are concatenated together. They set a fixed bound on the learnable sequence, thus "precluding modelling long-range dependencies " [17]. RNNs avoid this issue and can model arbitrary length dependencies.

2.1.3 Convolution Neural Network

In a traditional neural network all neurons are fully connected, each neuron takes as input all outputs from the preceding layer. A neuron at layer n receives inputs from all nodes at the pervious layer $(n-1)$, and so requires a corresponding number of weights. A [10X10X3] image as input would results in 300 weights per neuron in the first layer. As images increase in resolution the number parameters will increase accordingly. Using images of a more realistic resolution ([100X100X3]), combined with many neurons per layer and multiple layers, the number of parameters for the network (ω) will quickly grow to an unreasonable size. Having too many parameters increases training complexity and can lead to overfitting [25].

Convolutional Neural Networks (CNNs) exploit the spacial nature of images to reduces the number of connections and so parameters. Rather than looking at all inputs, each neuron of a convolution layer looks at (takes as input) a $3D$ subsection of the previous layer. Additionally, pooling layers are used to perform downsampling. This reduces the overall dimensionality of the data, reducing the number of parameters and the amount of compute needed [25]. Additionally, this helps to reducing over fitting by forcing the network to extract only relevant features.

2.2 Nowcasting

Nowcasting, in the meteorological context, is a process that "maps the current weather, then uses an estimate of its speed and direction of movement to forecast the weather a short period ahead; assuming the weather will move without significant changes" [26]. Various approaches for nowcasting exist and the specific one used depends on the timescale and the meteorological measure to be forecast [9]. A major challenge of nowcasting is dealing with the chaotic nature of the weather. In the context of machine learning this can be thought of as a regression problem.

As described above, taking a set of know states to feed through an ANN in order to predict solar irradiance is a proven way to build a regressive model. This, however, relies on the data used to make the prediction already existing. As described in the motivation, being able to predict how much irradiance a solar farm will receive *now* is of little use. By combining the approach stated above with nowcasted values, we can provide accurate values at least a few hours into the future.

2.2.1 Existing Approaches

One approach for short range weather predictions is to use numerical models [27]. By combining images and other measurements of the weather at T_0 and running them through a simulation of the physical equations in an atmospheric model it is possible to calculate expected values for $T_1, T_2 \ldots T_n$. However, this is computationally expensive and often reserved for longer term predictions [9].

Another approach is to use to use optical flow. Taking weather images, ether from radar or satellite and extrapolating movement forward in time [28]. This approach is explored in more detail in further sections.

2.2.2 Novel Approaches

Deep learning and other machine learning techniques have been used, with success for nowcasting [9]. An interesting example of this is the use of a Convolution LSTM (ConvLSTM) network [27]. Weather radar image sequence can be thought of having both spacial and temporal aspects. LSTMs are known to be effective at modelling temporal sequences [17]. The use of a fully connected LSTM, in this instance however, results in "redundant" connections. Adding a convolution in the input to state, and state to state transitions of the model, the spacial aspect of the data can also be represented. Using this technique with weather radar data from airports they were able to outperform existing approaches. The authors also discuss how this approach can extend to more generic spatiotemporal sequence predictions.

2.3 Imputing / Interpolating Data

The granularity of the raw data, for some sources is much lower than the target output. For example, the satellite imagery is only updated once every three hours, a much lower frequency than the required output granularity of one hour. There are two ways to approach this, add a "flag" to the predictive model so that it can learn to adjust its predictions for data that is misaligned to its output target time. Alternatively, one could attempt to impute (fill in) the missing data.

This approach also has the advantage of potentially simplifying switching to alternate, more accurate or granular data sources down the line. If for example, a new source of satellite imagery was used that had a much higher granularity it could be simply be swapped in place of the imputation.

There are various statistical methods for imputing sequence data. Filling in using the last value, taking the midpoint between $t-1$ and $t+1$. However, for this use case they have two major issues. Firstly, the meteorological data used is represented as a 2D matrix, meaning there is no simple midpoint between the two points. Secondly, they required a point in the future, e.g. $t+1$, to be known. Whilst this is possible at training time, any production system would not have that luxury and so would have to use a predicted value (either nowcasted by the system using the approaches above or from external forecast providers). This approach could be practical however it would add more uncertainty to the system. A more ideal solution would be to create a single model that can take in the lower frequency raw data and output imputed and predicted values in the target frequency.

An interesting comparison that addresses both the 2D and the temporal nature of the data is video. A lot of work, in recent years, has been put into both next frame prediction and intra frame prediction. This comparison was the inspiration for [27] and avenue explored as a potential solution.

The following sections will look at approaches that leverage the sequential nature of the data to attempt to impute the missing values. Section 2.3.1 gives an overview of optical flow. Section 2.3.2 explores the use of Generative Adversarial Networks.

2.3.1 Optical Flow

Optical flow is a technique commonly used in computer vision (CV). "The goal of optical flow estimation is to compute an approximation to the motion field from time-varying image intensity" [29]. By taking in a sequence of images, a $2D$ motion vector in the image plane can be calculated for each point. Using these motion vectors, each points next position can be calculated and the next frame generated. This approach can be used to predict the future movements of objects within a scene based on their existing trajectories. Optical flow has been applied to many CV problems such as object detection and tracking, movement detection and robot navigation [30] Optical flow has also been used for nowcasting, predicting the next "frame" of weather radar images [27, 28].

Whilst not a perfect fit, it would be possible to take the lower frequency input data and produce values with a higher granularity. One potential shortfall of this approach is that optical flow does not account for the chaotic nature of weather.

2.3.2 Generative Adversarial Networks

A relatively new approach for image generation are Generative Adversarial Networks (GANs). Initially described by Goodfellow et al, a GAN is comprised of two networks a "Generator" and a "Discriminator" [31]. The Discriminator D attempts to predict the probability that an image was created by Generator G or drawn from real examples. Both the Generator and Discriminator are trained simultaneously. The Generator learns by attempting to output images that D cannot confidently classify, if G is successful D's confidence for all images is 0.5. At the same time the Discriminator learns to differentiate images produced by G and those drawn from "true" sample space.

By using complex and novel architectures for both G and D, GANs have successfully been used to generate highly photo realistic images [32].

Mathieu, Comprie, and LeCun have extended the use of GANs for video frame interpolation and next frame prediction [33]. They also attempt to address a perceived fuzziness of generated images by combining both a standard GAN with an additional loss to "based on the image gradients, designed to preserve the sharpness of the frames." By combining the confidence level of D with this new loss to train G they were able to generate objectively sharper images. This approach is interesting as, for weather data imputation, the additional loss could be a measure of prediction accuracy.

There are two major differences between the uses above and the proposed use:

- The time scales are vastly different. In the intra frame generation, the differences are a less than a second. The cloud images are over hours.

- There is no simple measure of ground truth. Only data from time step T_{3n} is available.

2.4 Embedding

Categorical data can present challenges when training an ANN.The data must be converted into a numerical form that the network can use. It can be represented as a one-hot encoded vector, or with an integer label. However, these representation of categories can often obscure information and relations inherent in the data.

Embedding is a form of dimensionality projection. It is an approach often used in Natural Language Processing (NLP) where integer representations of words are projected into a vector space [34]. As the embedding layer learns the mapping between discreet and vector space, it can model relationships between similar words or concepts. Labels that are deemed to be similar are projected near to one another in vector space (e.g. Dog and Puppy).

This method can be applied to any type of categorical data that can be represented using integers. Brébisson et al. used embedding for traditional categorical data to good effect. They have also shown that it can be effective for data that could be considered continuous, e.g. location [35].

Design

This section provides an overview of the system design, it contains both a high level overview of the production system as well as details the proposed models. The main technology of the production system is built on the Microsoft Azure cloud, so many of the systems leverage functionality provided by products offered by Microsoft. Section 3.1 outlines the data collection pipelines and Section 3.2 provides and overview of the production system. Section 3.3 outlines high level aspects of the model while Section 3.4 details the specifics of the models.

3.1 Data Collection

The raw data is collected from a variety of sources. Whist the specific details are beyond the scope of this project, it is worth providing an overview of the data pipelines. The main sources are industry databases for plant measures (primarily used for the irradiance) and commercial weather providers for both satellite imagery and ground observations. A more in-depth description of the data collected is in Section 4.3.

The data collection system is fairly simple. It ingests all raw data and lands it into the data lake, a central repository for all data that can scale as needed. It is composed of two separate systems. A set of Azure Function that run periodically, every 30 minutes collecting weather data and every three hours for satellite imagery from various sources. The second process runs as a scheduled Databricks notebook (Apache Spark). It loads the raw data, reshapes it, and saves it in parquet format. This pipeline can also be extended to run and export predictions to a database or other "sink" as needed. An overview of the pipelines can be seen in Figure 1.

3.2 Production System

As discussed earlier, the main aim of the project is to build a performant model. However, it is worth providing an overview of what a production system would look like as it helps inform design decisions. A high-level overview of how various pipelines, processes, and models fit together can be seen in Figure 2. Knowing the model must be usable in production provides some high level NFR and FR for the model. Additionally, other aspects of productionisation must considered such as version control and model management.

3.2.1 Requirements

Non-Functional Requirements

- Minimal practical latency between new data entering the system and predictions being available

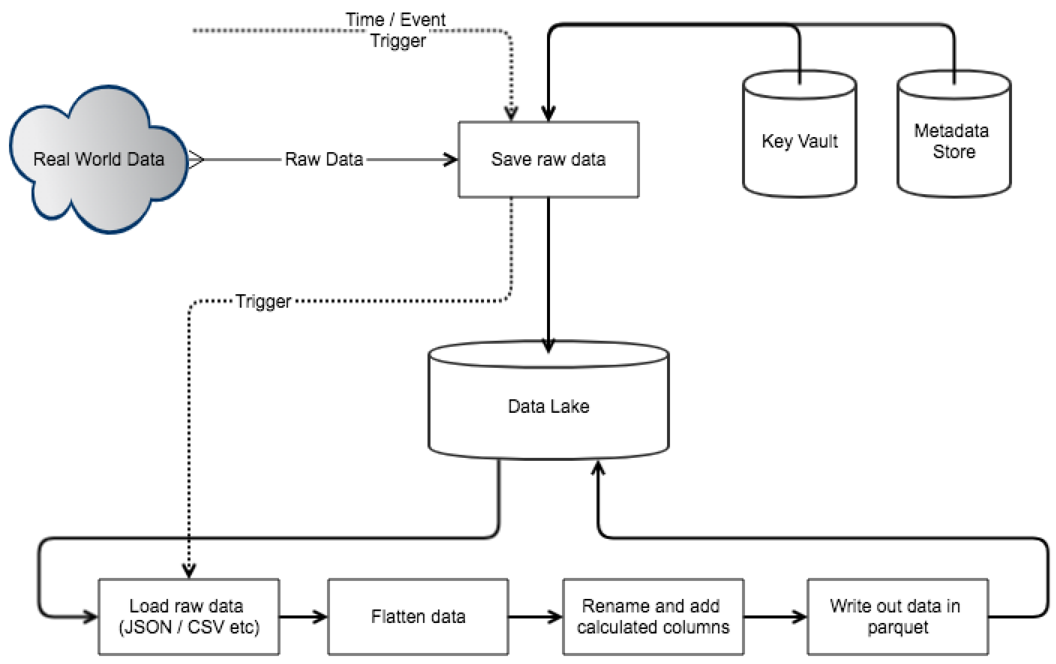


Figure 1: An overview of data collection pipeline

- Easy expandability of the model, adding new data sources or expanding the scope of its predicting e.g a new location
- Comparable performance to existing approaches

Functional Requirements

- Provide irradiance predictions for all locations with existing predictions
- Exported model be able to function in a distributed framework

3.3 Predictive Model

This section details the high level decisions made for the model(s).

3.3.1 Feature Selection

As with all machine learning, feature selection is a vital part of model design. For the proposed use, practical considerations must made. In order for any model to be used in production, the features selected must be available to the model at inference time. If the model can achieve a high performance level at training time but uses data with a high update lag it is of no practical use.

An example of this would be the approach described by Alzahrani et al. [20]. Their model makes use of the irradiance measures from the plant. Whist this data will be available eventually, to productionize the model would require a low latency feed from the plant. In some cases, this might be a possibility, but it cannot be guaranteed. It is for this reason that a more traditional autoregressive time series approach is not used[1].

[1]A traditional autoregressive model would include the last n irradiance steps to predict the future irradiance values

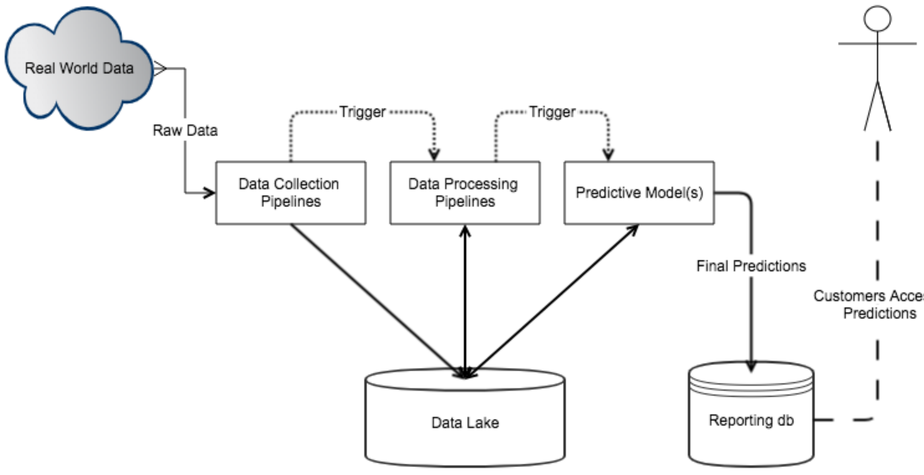


Figure 2: Overview of the main components of the system outlining the data flow

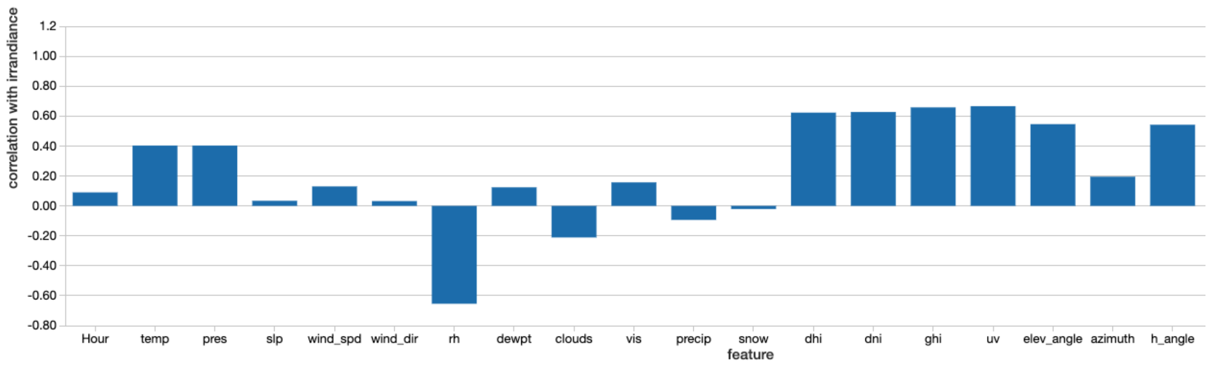


Figure 3: Correlation statistics for Irradiance compared to weather measurements. The feature codes are defined Table 1

As such, the model will only use weather data. However, unlike the industry model, the proposed models will use satellite image layers. The layers, a subset of all available, were chosen based on chosen base on the correlation statistics shown in Figure 3.[2] They all represent fairly standard meteorological measurements that would be available from alternative providers. Table 2 describes the chosen layers. Additionally, metadata such as latitude, longitude, time of day and day of year is used as appropriate.

3.3.2 Global vs Local Models

Another key decision to make is whether to build a global or local model. There are four basic conceptual approaches for the predictive model. They are comprised of global and local inputs and global and local outputs. Global inputs encompass the whole system, in this case weather images covering a large portion of the UK.Local inputs are made up of images covering a smaller geographic area. Local outputs map to a specific instance (in this case a single plant) while global outputs predict values for all instances at once. This section will provide an overview of the four approaches and touch on some of their conceptual and practical pros and cons

Local Local Simply put this is a model per plant and is the approach taken by the industry model. On-boarding of new plants is simple, one just trains a new instance of the model. However this requires that the plant have enough

[2]These statistics were provided by Elastacloud. They were calculate using a different plant set and ground based weather stations.

historic data to effectively train a model (upwards of a year). Whist a model per plant may be practical for a small number of plants there are some major drawbacks. The amount of compute required for inference grows with the number of plants. When improvements to the model architecture are made, the model for all plants must be retrained. The approach does not scale efficiently, if one wanted to scale up to predictions for domestic installations, a model learned per house would not be practical.

Local Global A single model that can be used to predict for all plants. This is a generalised version of a local model as it still only predicts output for one plant at a time. This approach solves some of the scaling issues that local local models have. Only a single model is created reducing overall training time. Additionally it is potentially possible for the model to produce predictions for unseen locations. The approach can also handle plants with a lack of data. However, the amount of compute used for inference still increase in line with the number of points that require a prediction. Additionally poor generalisation could cause a reduction in performance.

Global Local A single model that predicts all plants at once. As input it takes images covering a much larger geographic area. It outputs a vector with size equal to the number of plants in the training set. This approach has the inverse scale issues as that of the Local Global. On-boarding requires a change the model and so a full retraining. Plants without a full history present issues as they would cause missing data in the training set. However, inference compute is fixed as one step produces predictions for all plants. The model has the potential to capture relationships between plants

Global Global Functionally very similar to Global Local. It takes the larger images as input and predicts all plants at once. However its predictions are embedded in a world representation (see Section 4.4.3). This approach gets around the ridged architecture of a Global Local model by output a world representation. Conceptually this approach is the best of both worlds as it combines the strengths of both Global Local and Local Global models. However, finding an effective world representation can be challenging. Depending on the representation this output might require a downstream refinement step to convert the output to an exact prediction.

A Local Global and Global Global approach have been chosen as they provide a balance of flexibility and potential for generalisation beyond any specific plant. Additionally, a Local Local model is built for direct comparison to the industry model.

3.4 Architecture

Initial architectures envisioned a system comprised of two models. A GAN to both nowcast and impute data, and a separate regression model to produce irradiance values from the GANs output (Figure 4). However, this was predicated on the ease with which one could build a strong regressor to act as a baseline for the GAN. It became clear early on that doing so would not be easy. As such building a strong Discriminator to train the Generator became untenable in the timeframe.

The approach was adapted and rather than using an adversarial training process to train a separate GAN and regressor, they were combined into a unified model (Figure 5) and trained using standard supervised learning techniques. This approach is flexible enough to be used for all classes of local and global models and deal with arbitrary differences in input and output data frequencies. The fundamental architecture is comprised of three main components.

A Convolutional Module Take as input the raw satellite images and performs initial feature extraction, reducing the overall dimensionality of the data in the process.

An Imputation Module Up-sample and project the low frequency representations across time.

A Regression Module Regress over the projected internal representation combined with meta data to output irradiance predictions.

3.4.1 Convolutional Module

As input this module takes a set of images comprised of the raw satellite images normalised to values between $[0-1]$. The module performs initial feature extraction and reduces the overall dimensionality of the data before it is fed into the imputation module. All layers in the module are time distributed, meaning the same transformations are applied to all input image stacks i.e. the feature extraction process is time invariant. Since there are multiple images being fed in at once there are two straightforward ways for a CNN to deal with them:

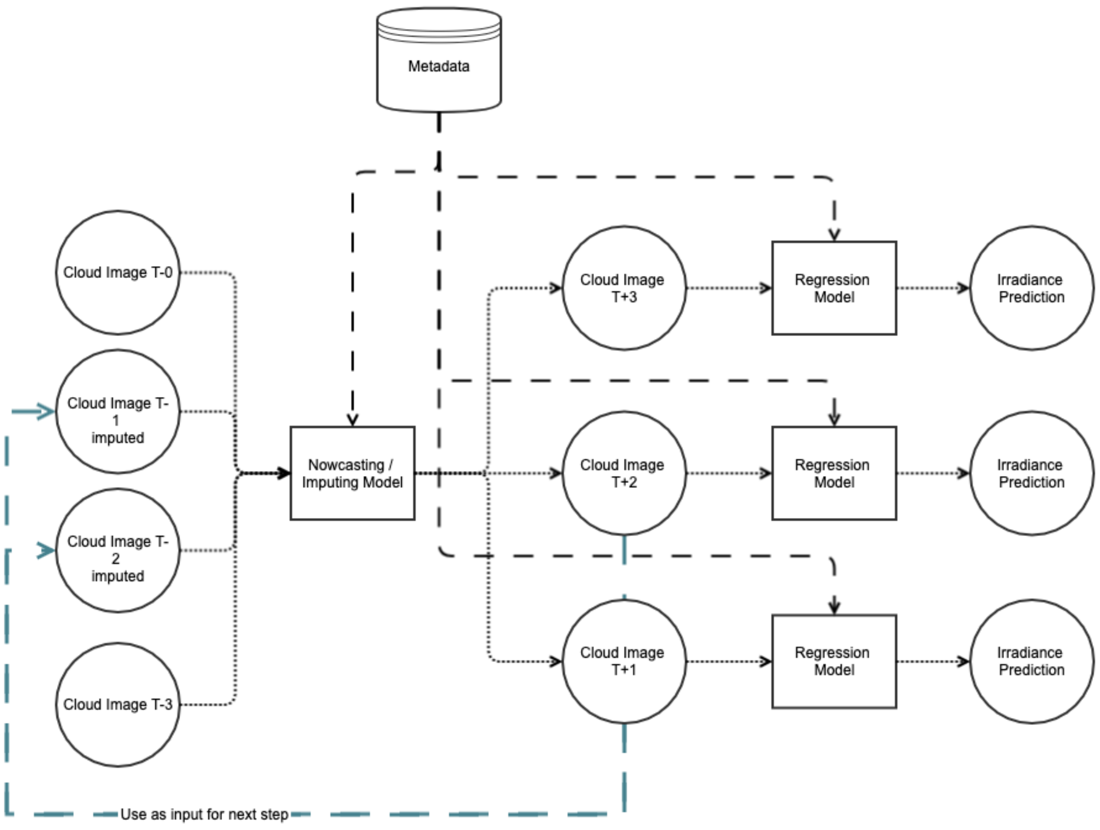


Figure 4: Overview of the envisioned architecture

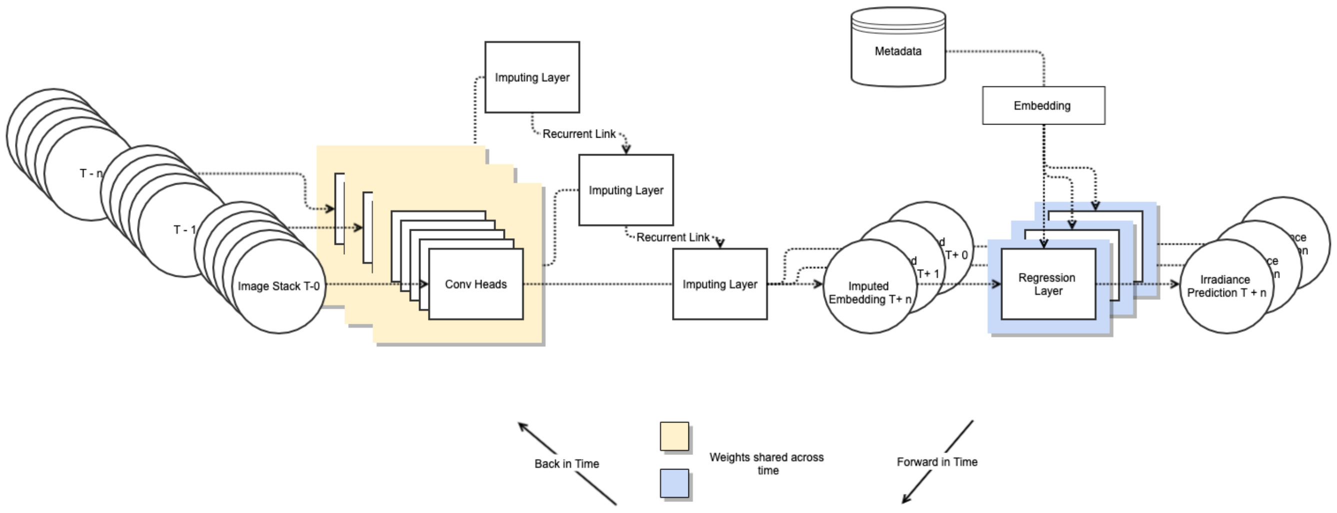


Figure 5: The fundamental model architecture used. Combining imputation and regression. Coloured boxes represent weights shared across time

Multi Channel Convolutions The images are stacked along the channel dimension resulting in a $3D$ tensor with layers $[RGB \dots RGB]$. Convolutions are then applied to all images at once.

Multi Head Convolutions A separate convolution is learned for each image. The results of each convolutional head are concatenated together before being passed to the next module.

Canizo et al. have explored both approach for initial feature extraction with multi input time series data and concluded that multi head is superior, as such this is the approach taken [36].

3.4.2 Imputation Module

The imputation module serves two functions. It upsamples the satellite imagery to match the target frequency, imputing the missing values. It also enables forecasts to be made, projecting into the future. It is built using ConvLSTM layers, combining aspects of both CNNs and RNNS, allowing the network to build an internal spatial representation across time. The penultimate layer of the imputation performance a convolution with the same number of filters as time steps to predict (a sum of the intra-image steps and the forecast horizon). This results in a $3D$ object where its third dimension represents the forecasted internal representation for the regression. The advantage of this approach is that it is time scale agnostic. Within reason, this approach can deal with an arbitrary difference in frequencies between images and predictions. It can also easily scale to support longer forecast horizons. The final step of the module expands the $3D$ time distributed object axis into a set of $2D$ objects. Each $2D$ slice is then flattened and feed into 256 wide dense layer. This step reduces the dimensionality of the input to the regressor

3.4.3 Regression Module

Both the convolution and imputation modules are architecture agnostic, they are the same regardless of whether a local or global approach is being used. The overall architectural flexibility comes from changes in the regression module. For the sake of simplicity, the regressor for both the local and global models are similar in structure. When a local output approach is being used a traditional, fully connected ANN is used with a final layer size of one. For the output global architecture, fully connected layers are used in addition to convolutional upsampling to produce an output grid.

Implementation and Experimental Process

This section will provide an overview of the experimental frameworks used as well as provide implementation details. Section 4.1 will provide an overview of the performance metrics and validation steps used. Section 4.2 provides an overview of the industry model, the performance of which is used as a reference point. Section 4.3 discussed the raw data used to train the models with Section 4.4 outlining how it was processed. Section 4.5 provides implementation details of the various proposed architectures.

4.1 Metrics and Validation

The ability to compare performance of models is paramount. In order to do so, clear and robust measures must be used. Metrics provide a measure of performance and enable comparison between various approaches. Additionally, validation ensures the results are robust and accurate comparisons can be made. Section 4.1.1 describes the metrics used and Section 4.1.2 outlines the validation steps taken.

4.1.1 Metrics

Clearly defined performance metrics that can be used for all models facilitates comparisons between them. It is also important to make sure that the metrics encompass various aspects of the model's performance, e.g. accounting for outliers etc. whilst being easy to understand and objectively compare. There exist numerous metrics for evaluating the performance of regression models throughout the literature. They primarily provide a summary of the error distribution where error is defined as the difference between the observed and predicted value [37, 38]. Additionally, there exist many ways to measure forecast accuracy [39]. A common feature of forecast errors is a scaling of the error enabling comparison across distributions.

The following performance metrics are used to evaluate and compare the various models:

- Mean Absolute Error (MAE)
- Root Mean Square Error (RMSE)
- Coefficient of determination (R^2)
- Mean absolute scaled error (MASE)

For a dataset comprised of n examples where $Y = [y_0, \ldots, y_n]$ represents the observed value and $\hat{Y} = [\hat{y_0}, \ldots, \hat{y_n}]$ the predicted values. The performance metrics are defined as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=0}^{n} |y_i - \hat{y_i}| \tag{1}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n} (y_i - \hat{y_i})^2} \tag{2}$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=0}^{n} (y_i - \hat{y_i})^2}{\sum_{i=0}^{n} (y_i - \bar{Y})^2} \tag{3}$$

$$\mathrm{MASE} = \frac{\mathrm{MAE}}{\mathrm{MAE}_{\text{naïve forecast}}} \quad \text{where naïve forecast: } \hat{y_t} = y_{t-24} \tag{4}$$

MAE and RMSE are easy to understand scale-dependent metrics, they provided an error measure in the same unit as that of predicted value. They provide a high-level overview of performance but can be misleading when comparing distributions with different ranges, for example performance throughout the day. A scale invariant metric enables comparison between various output distributions. Both R^2 and MASE are scale invariant enabling more nuanced comparison between models. A downside to scaled error metrics is their behaviour when values approach 0 can become difficult to interpret.

4.1.2 Validation

Another key part of evaluating the performance is validation. Validation ensuring the model will generalise well to new data and is not overfitting the training set. In other words, making sure the metrics are representative of future performance. This is done by measuring the learned model's performance on unseen data.

Given a large enough dataset one could simply use a fixed train test split to validate the model. However this requires a gargantuan dataset to be effective and as such is rarely possible. It is often desirable to make use of the full dataset for both training and validation. *k*-fold cross validation (CV) is a widely used approach to measure how well a model generalises especially with relatively small datasets. Cross validation enables the use of the full dataset for both training and validation by learning successive models form a distinct random subset of the data and using the held out set for validation. This produces *k* metrics that can be averaged to calculate a generalisation error [24].

The random nature of *k*-fold could potentially cause issues for time series models. Since data is grouped into windows, if the data is randomly split points could end up in both the training and test sets. As such it is common to use approaches such as last block / out-of-sample evaluation. These approaches are attractive as they closely model the systems final use and preserve the natural temporal dependencies within the data i.e the future depends on the past. However, a major drawback is that they fail to make full use of all available data. Bergmeira et al have shown that "for purely autoregressive models, the use of standard *k*-fold CV is possible" and can provided a meaningful generalisation error [37, 38].

Another approach proposed by Bergmeira et al. is to adapt *k*-fold and use holdout sequences. Rather than randomly shuffle all data points, instead split the data into blocked independent sequences and remove *k* at random. Due to its simplicity and relative ease of implementation across datasets holdout *k*-Fold cross validation will be used. All models are trained using week long blocks split on T_0.

4.2 Industry Model

A key motivation for this work is to provided alternative methods for industry use. In order to prove the approaches proposed have value, their performance will be compared against existing industry techniques. This section provides a high-level overview of the industry model that will act as a baseline.

4.2.1 Model Architecture

The industry model makes use of Gradient Boosted Trees (GBTs). A GBT expands on a basic decision trees and functions similarly to random forest combining the prediction from multiple trees. However, unlike a random forest which trains multiple trees in parallel on different subsets of the data, GBTs, as their name suggests, use gradient boosting. Trees are sequentially trained with each new tree being used to reduce the error of the previous trees [40].

The production model uses GBTs to perform regression across the input features to predict irradiance. Irradiance is inferred every hour. A local local architecture is used meaning each plant has its own model.

Some of the major advantages of this approach are that it is easy and fast to add a new plant. However, each new plant added needs to have enough historic data to train on, minimum of at least a year. Any time changes to the model are made, each plants model needs to be retrained. The time this takes will grow linearly with the number of plants.

4.2.2 Data

The raw data used to train the industry model is provided by [12]. Each plants data is sourced from the nearest ground-based weather station. The training set consists of the features in Table 1. Minimal data preprocessing is performed; all features are either directly provided or derived from the data. The only exception is the imputation of missing values for *h_angle*, *precip* and *snow*.

Feature Name	Description
clouds	Cloud cover (%)
dewpt	The dew point (c)
dhi	Clear sky direct horizontal irradiance ($\frac{W}{m^2}$)
dni	Clear sky direct normal irradiance ($\frac{W}{m^2}$)
ghi	Clear sky global horizontal irradiance ($\frac{W}{m^2}$)
h_angle	Solar hour angle (degrees)
precip	Precipitation (mm / hr)
pres	Pressure (mb)
rh	Relative humidity (%)
slp	Sea Level Pressure in (mb)
snow	Amount of snow fall in (mm/hr)
temp	Temperature (c)
uv	UV index (1 - 11)
vis	Visibility (KM)
wind_dir	Wind direction (degrees)
wind_spd	Wind speed (m/s)
season	Season (1 = Winter, 2 = Spring, 3 = Summer, 4 = Autumn)
tod	Bucketed time of day (1 = 06h - 11h, 2 = 12h - 17h, 3 = 18h - 23h, 4 = 00h - 05)
hour	Hour (0 - 23)
day	Day of month (1 - 31)
month	Month of year (1 - 12)
year	Year

Table 1: Features used by the GBT

4.2.3 Forecasting

A fundamental difference between the industry approach and the proposed architecture is the ability to forecast. The industry model performs flat regression over data, i.e. it makes the prediction for T_0. In order to be of use in industry a forecast of prediction are needed. In production this issue is circumvented by regressing over weather forecasts. However, in order to enable comparison between the various methods an alternative approach was used. A lag was introduced to the dataset between the feature and label columns. A unique tree was trained for each of the corresponding forecast horizon $T_0 \dots T_{+12}$.

4.3 Raw Data

Irradiance Data The irradiance data is comprised of 24 solar farms distributed across the UK (see Figure7). Each plant has a corresponding sequence of irradiance measurements taken at a frequency of up to five minute intervals. For training one hour frequencies where used with measures taken from $2018\text{-}02\text{-}15\ 02{:}00{:}00$ to $2019\text{-}02\text{-}14\ 16{:}00{:}00$ resulting in $\approx 200,000$ irradiance points.

Image Data Once every three hours, satellite measured meteorological layers are updated. The layers are provided as $256\text{px} \times 256\text{px}$ RGB tiles at various zoom levels. A total of 48 individual tiles at zoom level 7 are collected and composited for each layer every time step. The compost image covers the whole UK with an area approximately $49N$ to $62N$ and $-14E$ to $3E$ at a resolution of 1590px X 2048px. The image layer dataset spans the same date range as that of the irradiance. Table 2 describes the layers used and Figure 6 provides an example of each layer.

Layer	Description
APM	Atmospheric pressure on mean sea level (hPa)
CL	Cloudiness (%)
WND	Joint display of speed wind (color) at an altitude of 10 meters and wind direction (arrows), received by U and V components (m/s)
HRD0	Relative humidity (%)
TA2	Air temperature at a height of 2 meters (C)

Table 2: Satellite image layers used

Meta Data In addition the main data, the flowing metadata is used in full or part by the models

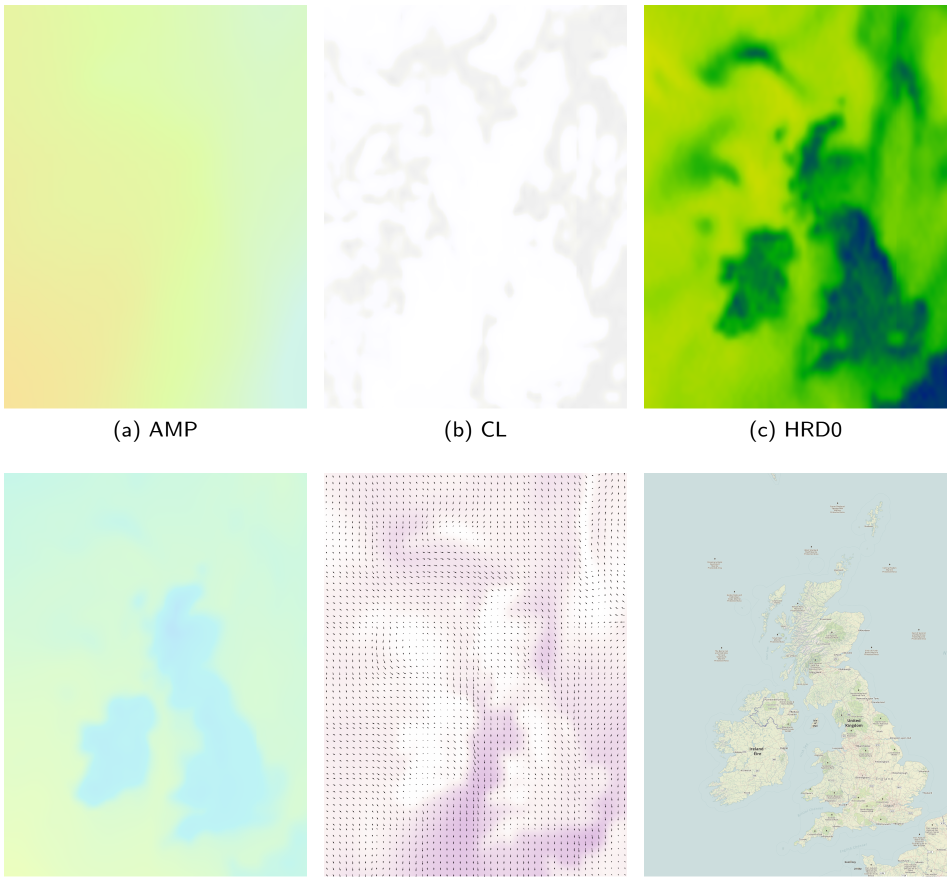
- Day of year (DOY)
- Hour
- Clear sky
- Latitude
- Longitude

4.4 Data Processing

This section will outline the data preprocessing steps common to all models. Section 4.4.1 and 4.4.2 will discuss data cleaning. Section 4.4.3 provides an overview of the world representation used by the Global Global model. And Section 4.4.5 outlines how the data is shaped for training.

4.4.1 Image Slicing

The raw $1590px$ X $2048px$ PNGs are too big to be used by any of the models. In the case of the local input models, a 256px slice centered on the plant is taken from the raw image. For the global model a 1024px is taken. The image slice


(a) AMP (b) CL (c) HRD0

(d) TA (e) WND (f) UK Map for reference

Figure 6: Image layers used by the model (2019-02-02 06:00:00)

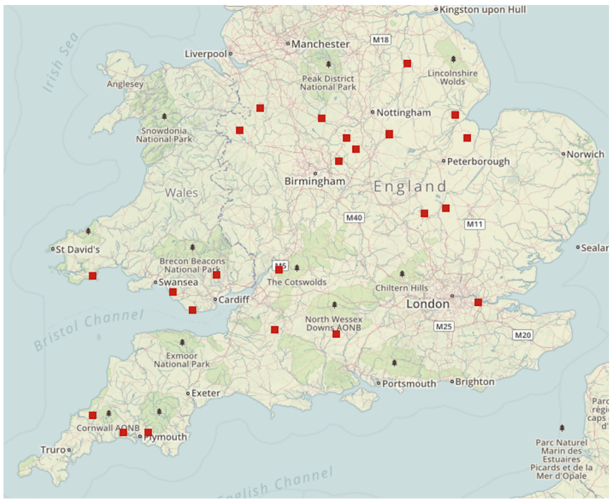


Figure 7: Map of all plants

is then downsampled by a factor of 4. This results in image layers of $64\mathrm{px}$ and $256\mathrm{px}$ respectively being used for training and inference.

4.4.2 Irradiance Data Cleaning

Whilst minimal data cleaning was performed a few necessary steps were taken. The raw data is reported in timezone Europe/London and so changes between GMT and BST.This causes alignment issues when grouping for forecast horizon and regression windows described in Section 4.4.5. To fix this, timestamps where converted to UTC using the python tz database.

All The irradiance data is clipped, all negative values where set to 0. Additionally, plants with clearly erroneous sensor readings e.g. negative values in the middle of the day had that respective days data removed from the dataset.

4.4.3 Grid Irradiance

A core part of the Global Global model is the world representation. In this case a 64×64 grid covering the same area of the UK as that of the image slice was used. At each time step an irradiance grid is created. Each plants latitude and longitude are mapped linearly to a cell in the grid by equation 5 [3]. The pants irradiance is filled into the grid, if multiple plants lie within one cell the average of their values is ascribed.

$$\begin{aligned} x &= \left\lfloor (\mathrm{lon} - \mathrm{min\ lon}) \times \frac{64}{\mathrm{lon\ range}} \right\rfloor \\ y &= -\left\lfloor (\mathrm{lat} - \mathrm{min\ lat}) \times \frac{64}{\mathrm{lat\ range}} \right\rfloor \end{aligned} \tag{5}$$

This results in a sparsely filled out grid. In order to fill in missing values an adapted form of Weighted K Nearest Neighbours (wKNN) was used [41]. Given the K nearest plants P to a point in the grid, the wKNN is defined in equation 6. In order to minimise error in the world representation, a few adaptations to wKNN where tested. A distance weight decay was applied, equation 7 (1/dist) and equation 8 (sqrt). Additionally, the Y term in the standard L2 distance was scaled equation 9. Figure 8 visualises the effects and Table 3 outlines the various approaches performance. Values where calculate by averaging the error of successively holding out a single plant, K was equal to the total number of plants.

$$wknn(x, y) = \sum_{i=0}^{K} \frac{Dist(P_i)}{\sum Dist(P)} \times Irradiance(P_i) \tag{6}$$

$$Dist = \exp(\frac{1}{\mathrm{Distance}}) \tag{7}$$

$$Dist = \exp(\frac{1}{\sqrt{0.065 * \mathrm{Distance}}}) \tag{8}$$

$$L2(x_1, x_2, x_2, y_2) = \sqrt{(x_1 - x_2)^2 + 2(y_1 - y_2)^2} \tag{9}$$

The resulting grid represents irradiance measures over the whole UK and is used for training. The rational for this approach is twofold. It enables the network to learn whilst retaining the geo-special relationships with all the plants. The final architecture is flexible enough to support the addition and removal of plants over time. It allows plants with a relatively small amount of historic data to take advantage of the synthetic values produced by plants with more data.

[3] The use of liner mapping causes distortion and differs from the standard Mercator projection used by the image layers but should still retain a semblance of geo-spacial relation

SCALE	RMSE	MAE
sqrt	104.62	66.18
sqrt-2Y	106.33	67.62
1/dist	114.03	72.62
1/dist-2Y	114.25	72.78
mean	113.97	74.91

Table 3: Errors of the various grid imputation methods

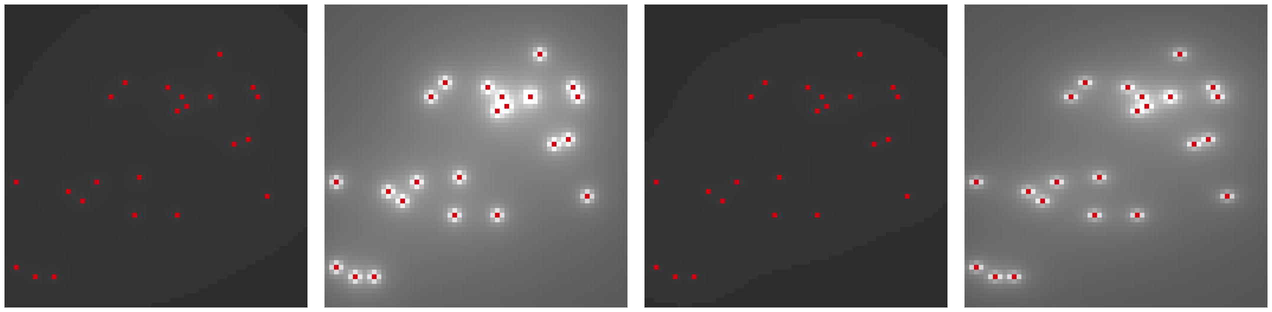

(a) 1/dist (b) sqrt (c) 1/dist-2Y (d) sqrt-2Y

Figure 8: Visualisations of the various imputation methods

4.4.4 Grid Clearsky

As shown by the correlation statistics in Figure 3, clear sky values are likely to be a useful feature for the model. They serve as a theoretical upper bound of irradiance. Similarly to the grid irradiance described above, a clear sky grid is also created. Its covers the same geographic area as the irradiance grid but at a lower resolution of 16×16. Each cells value is filled in with the calculated clear-sky of the latitude and longitude of the cells midpoint. Only the global model makes use of the full grid, the local models simply use their corresponding grid value.

4.4.5 Data Shaping

In order to train the models, the data must be shaped appropriately. Since the architecture consists of recurrent steps, the data must be windowed into a recurrent lag. Additionally as the model outputs forecasts, the data must also be windowed into forecast horizons.

Since the set of all image layers (an image stack) is updated once every three hours. While the irradiance, clear sky measures and other meta data, collectively a metric group, are updated at a much higher frequency (once every hour). Assembling a row of training data requires two steps.

Each image stack has a corresponding forecast window of length f. An image stack (at T_0) is joined to a set of metric groups coving the time-steps $T_0 \dots T_{+f}$ of its forecast window. This composite group is then joined together to with composite groups from $T_{-3}, T_{-6} \dots, T_{-3r}$. This produces a recurrent window of size r.

The final dataset consists of rows with images $T_{-3}, T_{-6} \dots, T_{-3r}$ and corresponding metrics from $T_{-3r} \dots T_{+f}$. Listing 1 shows some sudo code for how a composite group is assembled. Due to the nature of the joins there can be overlapping metric groups. As can be seen in Figure 9 metrics at $T_{-3,-2,-1}$ and $T_{0,1,2}$ fall into the forecast windows of both image stack $T_{-6,-3}$ and $T_{-3,0}$ respectively. This approach can be adapted to support both metrics and images with a wide range of update frequencies.

```sql
SELECT datetime, agg-list(images) FROM Images GROUP BY datetime

JOIN ON datetime

SELECT datetime, metric, lead(1, metric) \ldots lead(f, metric) FROM metrics ORDER BY datetime
```

Listing 1: SQL sudo code for a composite group

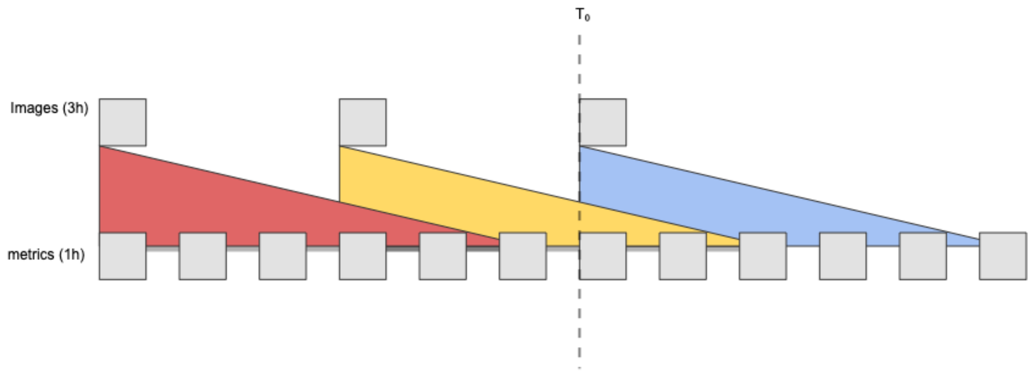


Figure 9: The structure of a row of training data where $f = 6$ and $r = 3$

4.5 Implementation

This section describes the implementation of the proposed models and states the hardware used to train them. As outlined in the design, the convolution and imputation modules are architecturally the same for all models. They only difference in implementation is the size of the input.

Convolutional Module Each convolution head is comprised of two stacked convolution and max pool layers. The input to the convolutions are zero padded. All convolutional layers use 4 kernels of size 3×3 and a stride of 1, a ReLU activation is applied and then a max pool of size 2 (Figure 10a). There is one head for every image layer (5) with each head using the same weights for all recurrent window.

Imputation Module The imputation model concatenates the output of all convolutional heads along the channel dimension. It then applies a ConvLSTM with 16 kernels of size 3×3 and a stride of 1 before a ReLU activation and a max pool of 2. It then zero pads the values before applying another ConvLSTM with $(\text{Intra-Image} + \text{Forecast Horizon})$ kernels of size 5×5 followed by a ReLU.The resultant tensors values are then recorded, combining the channel dimension with time dimension. The resultant time distributed $2D$ tensors are then flattened before being passed through a 256 wide dense layer. This process is outlined in Figure 10b.

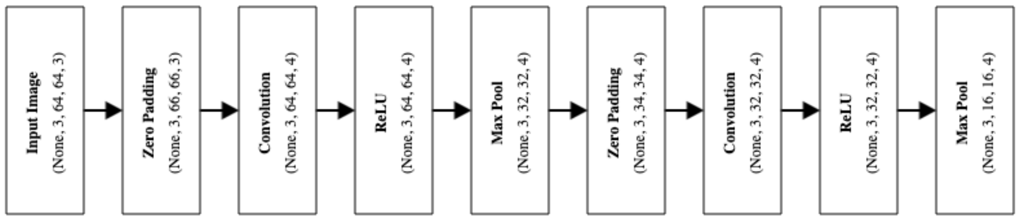
Local Regressor The local regressors are comprised of three fully connected layers of size 256 followed a layer of size one (Figure 11a). The activation function for all layers, including the output, is a ReLU.[4] The input consists of the 256 wide output of the Imputation Module concatenated with the embedded metadata and raw clear sky value. In the case of a Local Global model the all metadata is used, in the Local Local models only DOY and Hour are used. All embeddings output a vector of length 10.

Global Regressor The core difference between the local and global model is the output size, a 1D scalar vs a 2D matrix. The output of the imputation layer is fed through the same successive dense layers described above. However, rather than reducing the size of the output to 1, it must be increased. As such, after the dense layers the global model reshapes the data into 16×16 matrix and "upsamples" the output with nearest-neighbor interpolation followed by a padded convolution. Odena, et al. have demonstrated that this approach is more robust than a more traditional "transposed convolution" and produces fewer low resolution artefacts [42]. three successive upsamples, zero pads, convolutions and ReLUs are applied scaling the output from 16×16 to 32×32 and finally 64×64 (Figure 11b)

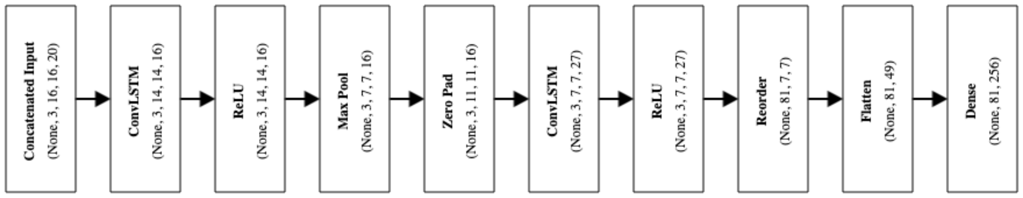
Training All models models where trained with the setting specified in Table 4. Table 5 describes variations of the Local Global and Global Global models trained.

All GBTs were run on an Apache Spark cluster with 1 driver and 2 worker nodes.

[4]It is possible to use a ReLU on the output as irradiance has a lower bound of 0.



(a) A single convolutional head



(b) Imputation Module

Figure 10: A single convolutional head and imputation module for a Local model with forecast horizon of 27 and a recurrent window of 3. Each box contains the layer name and output shape.

Parameter	Value
learning rate	0.0003
loss	MSE
optimiser	Adam
epocs	10
batch size	2
dropout	0.2

Table 4: Parameters used to train all models

Driver 14.0 GB Memory, 4 Cores

Worker 28.0 GB Memory, 8 Cores

The proposed models where trained in Google Colab with GPU acceleration

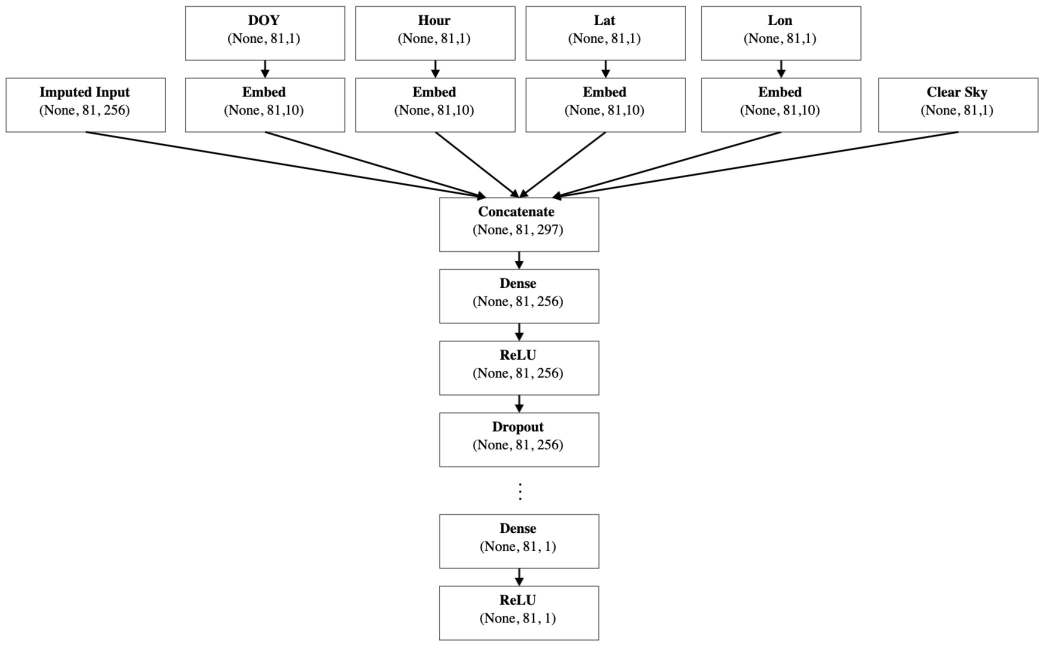
- 2 Cores, 13.0GB GB Memory, Nvidia K80 GPU
- Tensorflow 1.13.1 2.2.4-tf

Results and Analysis

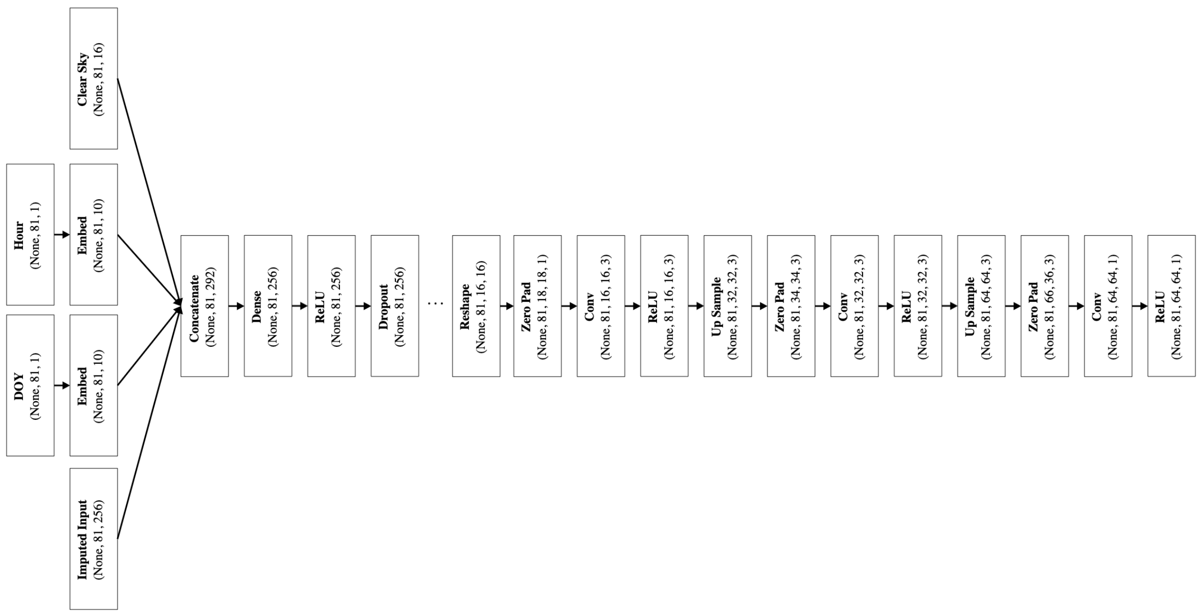
This section will present preliminary results and analyse the performance of the models described in the previous section. The models performance can be broken down into two aspects. Firstly its regressive performance i.e. the models performance at any given time step. Secondly its ability to forecast future irradiance values. Section 5.1 will discuss the performance at time step T_0. Section **??** looks at the models ability to forecast. Additionally, Section 5.3 will discuss the practicality of the various approaches.

All results where obtained using the validation approach described in Section 4.1.2. However due to time constraints, in the case of the LL models, only two full fold where trained.

Table 6 present the overall results. The LG8 performed the best, in absolute terms outperforming the industry model by almost 18%. Interestingly, the longer forecast horizon version LG27 is the next best beating the LL approach. A sample of the predictions can be see in Figure 12.



(a) Local Global regressor



(b) Global regressor

Figure 11: Example of regression modules with forecast horizon of 27 and a recurrent window of 3. Each box contains the layer name and output shape.

Name	Architecture	Forecast Horizon	Recurrent Window	Trainable Parameters
GBT	Local Local	-	-	-
LL	Local Local	8	3	$195,249$
LG8	Local Global	8	3	$201,649$
LG27	Local Global	27	4	$298,625$
GG	Global Global	6	3	$487,878$

Table 5: Models Trained

Model	MAE	RMSE	R^2
GBT	50.17	99.94	0.7866
GG	57.47	109.7	0.7305
LG27	45.33	89.30	0.8448
LG8	**41.83**	**81.46**	**0.8728**
LL	48.12	91.89	0.8278

Table 6: Overall results

5.1 Regression performance

This section will look at the various models performance at T_0. In the case of the GBT this is its standard operation with no time lag. For all the proposed models the results are calculated by aggregating their output from step 0, 1 and 2. This is due to the fact that the images only update once every three hours and as such step 0 would only provide predictions for hours $[0, 3, 6, 9 \dots 21]$. The analysis is divided into three slices:

- Aggregate performance per plant
- Aggregate perforce throughout the day
- Aggregate performance throughout the year

5.1.1 Plant Performance

An important aspect of the models performance, especially in industry, is how stable predictions are from one plant to another. A model that performs, in aggregate marginally worse, but is consistent for all plants would be preferable over a model with a large intra-plant variance. As can be seen in Figure 13, LG8 is constantly the most stable approach on a per plant bases. All approaches with a global component, (LG8, LG27 and GG) have a much lower intra plant variance when compared to the purely local approaches.

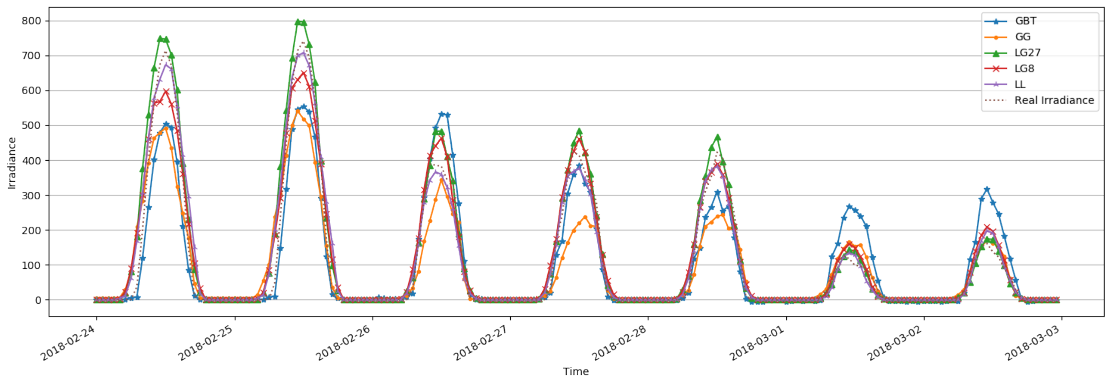
As can be seen in Tables 7, 8, 9 and 10 the LG8 is also often the best performing. It is worth noting that the both the GBT and LL contain individual plants with comparable, if not better results. The GBT especially has two outliers, Box Road and Moss Electrical.

One possible explanation for why models with a global component are more stable is that they take advantage of the increase in training data to learn a more robust regressor at the expense of taking advantage of certain plant specific trends. Alternatively the local models could be able to take advantage of plant specific trends that don't exist for all locations.

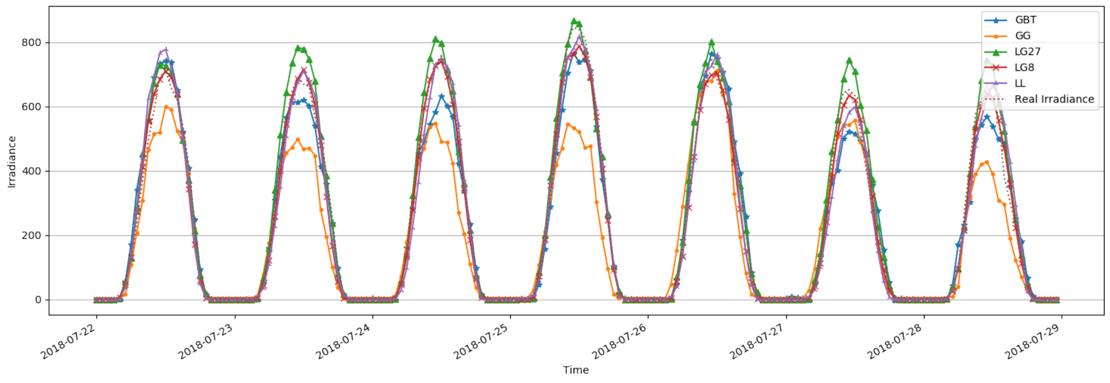
5.1.2 Hourly Performance

Throughout the day both the MAE and RMSE increase for all models in line with the overall irradiance (Figure 14a and 14b). The LG8 is constantly the best in performer, in absolute terms, throughout the day with performance being comparable at dusk and dawn for all models save the GG.

However when scaled it is not as clear (Figure 14c and 14d). During the peak irradiation hours, between 08 and 15, the LG8 for the most part performs the best. Conversely it appears to have relative poor performance during dusk and dawn. This reduction in performance is true for all models however it is much less pronounced for the GBT.

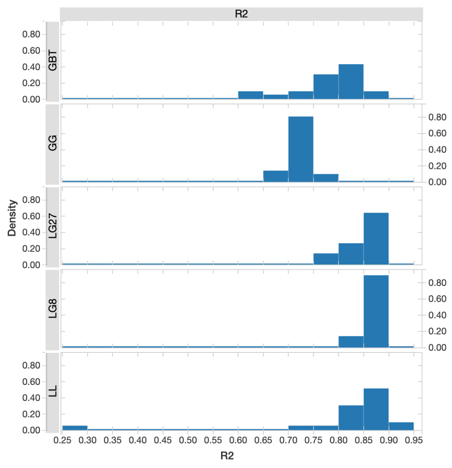


(a) Week long stretch from late February 2018, the period colloquially know as "The Beast From The East"

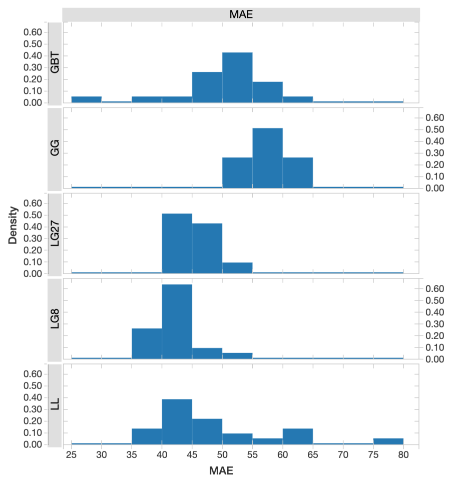


(b) Week long stretch from late June 2018, begging of the summer heatweave

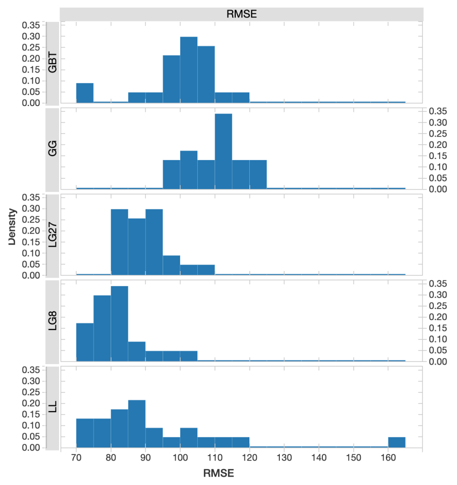
Figure 12: Sample of predictions at T_0



(a) R^2 - Higher is better



(b) MAE - Lower is better



(c) RMSE - Lower is better

Figure 13: Distribution of average errors per plant at T_0

	GBT		GG		LG27		LG8		LL	
plant	MAE	std	MAE	std	MAE	std	MAE	std	MAE	std
asfordby a	54.16	8.194	56.12	11.09	41.82	6.349	**40.00**	2.789	44.00	2.704
asfordby b	52.09	5.442	56.12	11.09	41.67	4.250	41.43	6.032	**36.99**	0.4709
ashby	55.54	9.366	54.61	10.81	41.82	6.211	**40.66**	2.858	42.18	3.843
bake solar farm	51.26	5.472	60.41	15.46	48.86	5.922	47.24	2.630	**46.78**	5.677
box road	**26.23**	9.178	58.62	13.09	44.49	3.970	42.69	3.385	40.42	8.591
caegarw	48.20	4.057	57.28	14.68	46.29	6.894	**41.86**	3.577	44.20	8.096
caldecote	54.34	5.077	63.22	14.50	45.99	5.014	**42.03**	3.385	43.57	3.475
clapham	52.55	2.926	60.81	12.34	44.40	2.654	**39.79**	3.778	41.80	3.357
combermere farm	49.07	8.451	52.95	13.82	42.35	5.087	38.45	5.732	38.40	3.071
crumlin	51.65	13.76	50.99	11.32	42.69	4.279	37.52	5.029	40.81	3.428
far dane's	52.84	2.511	55.29	11.25	43.71	4.771	40.49	4.013	41.62	7.243
grange farm	56.22	9.141	59.90	11.71	45.17	7.361	43.64	5.778	36.65	7.522
kelly green	45.09	3.450	58.21	16.55	46.07	3.356	41.99	6.401	54.57	3.993
kirton	55.39	5.980	61.22	12.28	46.10	4.001	44.71	3.189	48.00	2.189
lains farm	49.12	4.933	59.27	14.31	45.19	6.639	40.11	6.467	43.60	5.197
magazine	51.88	6.951	54.31	10.73	42.54	2.298	37.76	4.405	53.80	9.872
moor	56.36	2.792	55.67	13.51	46.12	9.185	**42.44**	4.213	61.32	24.01
moss electrical	**35.95**	5.224	52.33	12.26	49.84	6.542	40.48	4.260	46.45	3.305
nailstone	63.34	2.681	58.08	12.21	44.31	6.36	**41.86**	3.435	61.63	13.57
newnham	54.17	6.569	63.32	14.96	52.73	4.706	**50.36**	6.234	60.15	13.46
roberts wall solar farm	51.02	4.472	64.96	16.65	52.81	3.968	**48.67**	7.511	56.47	4.044
rosedew	46.70	3.267	56.48	14.47	43.55	3.703	**38.70**	3.501	46.67	12.63
soho farm	48.25	8.578	58.21	14.51	45.71	4.525	**44.18**	4.915	47.27	2.756
somersal solar farm	42.69	4.319	50.86	11.04	43.77	7.817	**36.93**	6.261	77.44	50.12

Table 7: MAE values per plant at time T_0

	GBT		GG		LG27		LG8		LL	
plant	RMSE	std	RMSE	std	RMSE	std	RMSE	std	RMSE	std
asfordby a	105.5	10.81	105.7	14.97	82.00	13.28	**76.83**	4.644	84.81	5.658
asfordby b	100.6	8.931	105.7	14.97	83.00	8.304	80.25	7.838	**72.82**	3.049
ashby	105.1	14.79	103.3	14.28	81.25	11.41	**78.41**	5.924	82.11	6.009
bake solar farm	100.5	9.410	116.0	24.02	97.50	12.56	93.98	4.534	**92.44**	8.860
box road	**72.80**	16.34	113.2	19.75	88.19	6.033	82.19	4.303	76.65	13.80
caegarw	99.04	8.247	110.5	22.95	90.54	11.20	**81.15**	4.904	83.83	12.15
caldecote	103.6	6.424	121.1	20.23	90.04	8.627	81.92	7.091	**79.06**	1.677
clapham	103.4	2.895	116.1	16.53	87.95	5.825	**77.68**	7.076	85.34	6.564
combermere farm	98.17	11.91	101.0	21.67	83.04	9.686	74.59	9.913	**74.55**	7.856
crumlin	106.3	23.77	98.42	16.57	84.25	9.569	**73.35**	7.948	79.42	5.368
far dane's	102.5	3.520	104.8	15.03	85.81	9.207	**79.09**	5.537	81.38	8.479
grange farm	108.4	15.49	112.6	16.29	91.19	13.69	84.87	9.073	**72.24**	12.43
kelly green	90.86	5.202	110.7	27.88	91.94	4.297	**82.41**	10.06	98.79	6.365
kirton	107.0	5.865	116.3	17.45	91.10	5.310	**85.28**	5.174	91.89	3.644
lains farm	95.77	8.407	111.7	22.03	88.41	6.806	**78.28**	9.502	85.35	8.325
magazine	99.58	11.12	102.3	15.8	83.50	5.773	**74.15**	6.997	102.3	10.01
moor	110.2	1.904	106.2	20.84	91.41	15.66	**83.36**	6.853	108.6	35.96
moss electrical	**71.53**	10.36	98.56	16.6	95.14	12.56	76.85	8.869	85.79	6.951
nailstone	119.0	5.455	110.0	17.23	86.22	11.06	**81.94**	5.923	111.7	23.84
newnham	107.8	7.323	121.3	23.43	106.1	7.528	**100.5**	12.03	117.3	15.84
roberts wall solar farm	103.3	8.655	124.6	26.02	104.6	7.632	**95.5**	13.55	103.2	6.732
rosedew	98.07	7.067	110.2	22.71	86.51	5.000	**76.58**	5.239	85.97	17.43
soho farm	102.2	13.54	114.2	23.81	90.62	6.807	**85.87**	8.035	89.15	3.490
somersal solar farm	87.23	8.067	96.90	16.61	82.83	12.56	**70.09**	9.784	160.8	102.7

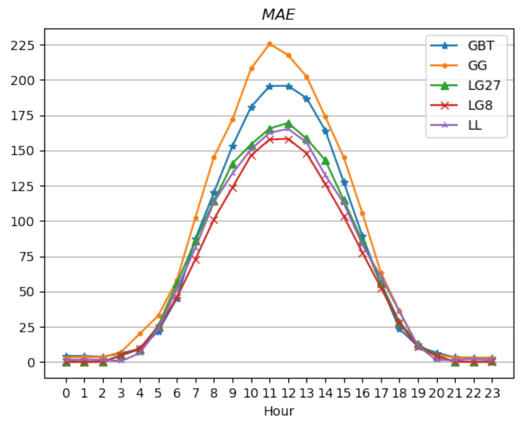
Table 8: RMSE values per plant at time T_0

	GBT		GG		LG27		LG8		LL	
plant	MASE	std	MASE	std	MASE	std	MASE	std	MASE	std
asfordby a	0.7872	-	0.8408	-	0.6107	-	**0.5877**	-	0.6810	-
asfordby b	0.7538	-	0.8386	-	0.6241	-	0.5954	-	**0.5418**	-
ashby	0.7577	-	0.8012	-	0.5923	-	**0.5730**	-	0.6376	-
bake solar farm	0.6596	-	0.8115	-	0.6322	-	**0.6004**	-	0.6315	-
box road	0.7130	-	0.8323	-	0.6071	-	**0.5737**	-	0.5812	-
caegarw	0.7080	-	0.8745	-	0.6594	-	**0.6003**	-	0.7044	-
caldecote	0.7501	-	0.9079	-	0.6437	-	**0.5882**	-	0.6562	-
clapham	0.7184	-	0.8456	-	0.5948	-	**0.5466**	-	0.6052	-
combermere farm	0.7367	-	0.8537	-	0.6520	-	0.5986	-	**0.5908**	-
crumlin	0.8019	-	0.8170	-	0.6546	-	**0.5808**	-	0.6232	-
far dane's	0.7759	-	0.8368	-	0.6508	-	0.6009	-	**0.5891**	-
grange farm	0.8692	-	0.9114	-	0.6885	-	**0.6442**	-	0.6675	-
kelly green	0.6324	-	0.8650	-	0.6517	-	**0.6181**	-	0.7348	-
kirton	0.7778	-	0.9031	-	0.6425	-	**0.6385**	-	0.6712	-
lains farm	0.7109	-	0.8592	-	0.6241	-	0.5808	-	**0.5181**	-
magazine	0.7332	-	0.8029	-	0.6086	-	**0.5607**	-	0.6847	-
moor	0.7463	-	0.7842	-	0.6025	-	**0.5769**	-	0.6892	-
moss electrical	**0.6973**	-	1.0320	-	0.9377	-	0.7627	-	0.7670	-
nailstone	0.8756	-	0.8407	-	0.6256	-	**0.5815**	-	0.6647	-
newnham	0.6549	-	0.8070	-	0.6464	-	**0.6022**	-	0.6698	-
roberts wall solar farm	0.6132	-	0.8333	-	0.6430	-	**0.6221**	-	0.6775	-
rosedew	0.6883	-	0.8810	-	0.6522	-	**0.5869**	-	0.7077	-
soho farm	0.7355	-	0.8371	-	0.6314	-	**0.6241**	-	0.6792	-
somersal solar farm	0.6938	-	0.8434	-	0.6744	-	**0.6102**	-	0.7085	-

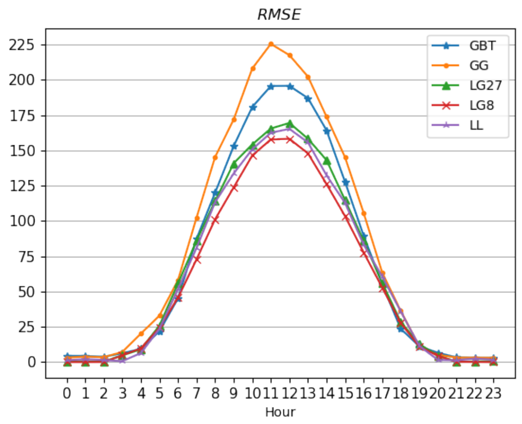
Table 9: MASE values per plant at time T_0

	GBT		GG		LG27		LG8		LL	
plant	R^2	std	R^2	std	R^2	std	R^2	std	R^2	std
asfordby a	0.8014	0.03482	0.7425	0.09792	0.8686	0.06247	**0.8908**	0.02412	0.8889	0.02064
asfordby b	0.7931	0.06627	0.7425	0.09792	0.8640	0.04283	0.8782	0.01829	**0.8801**	0.02393
ashby	0.6777	0.07870	0.7483	0.09131	0.8573	0.04708	**0.8794**	0.03021	0.8570	0.01456
bake solar farm	0.8264	0.02894	0.7198	0.08446	0.8167	0.05519	0.8434	0.02590	**0.8463**	0.01448
box road	0.7387	0.02698	0.7255	0.09812	0.8624	0.01026	0.8837	0.01366	**0.9003**	0.02645
caegarw	0.8013	0.09452	0.7266	0.10700	0.8555	0.02999	**0.8841**	0.01848	0.8709	0.01853
caldecote	0.8053	0.05422	0.6986	0.10180	0.8433	0.05014	0.8683	0.04290	**0.9036**	0.00322
clapham	0.8119	0.02868	0.7221	0.09838	0.8637	0.02871	**0.8842**	0.02139	0.8414	0.04713
combermere farm	0.8198	0.03847	0.7444	0.10390	0.8600	0.03366	**0.8865**	0.02401	0.8724	0.04335
crumlin	0.6331	0.10410	0.7593	0.09471	0.8612	0.01673	**0.8873**	0.03054	0.8866	0.01498
far dane's	0.7913	0.04876	0.7453	0.09289	0.8542	0.02251	**0.8733**	0.02610	0.8586	0.01081
grange farm	0.7883	0.05415	0.7264	0.08947	0.8226	0.05821	0.8637	0.02630	**0.8861**	0.02040
kelly green	0.8503	0.02387	0.7241	0.10160	0.8182	0.06283	**0.8595**	0.02717	0.7976	0.07440
kirton	0.7821	0.07779	0.7201	0.09585	0.8613	0.01452	**0.8745**	0.01629	0.8528	0.01182
lains farm	0.8170	0.04696	0.7348	0.09814	0.8459	0.06066	**0.8863**	0.02994	0.8858	0.02182
magazine	0.7928	0.05676	0.7467	0.09459	0.8622	0.02312	**0.8874**	0.02139	0.8311	0.01258
moor	0.7817	0.05957	0.7304	0.10180	0.8396	0.03350	**0.8639**	0.03892	0.8150	0.08407
moss electrical	**0.8863**	0.02794	0.7357	0.09120	0.7935	0.05007	0.8561	0.04843	0.8349	0.04629
nailstone	0.6434	0.12760	0.7342	0.09207	0.8639	0.01886	**0.8803**	0.01863	0.8094	0.06530
newnham	0.7961	0.03330	0.6953	0.09286	0.7891	0.04056	**0.8184**	0.04150	0.7350	0.02119
roberts wall solar farm	0.8177	0.05828	0.6834	0.09590	0.7885	0.07633	**0.8376**	0.03811	0.8128	0.06349
rosedew	0.8271	0.02360	0.7330	0.09713	0.8573	0.03217	**0.8856**	0.03290	0.8592	0.07420
soho farm	0.7459	0.04653	0.7299	0.09642	0.8567	0.02031	**0.8757**	0.03201	0.8630	0.03081
somersal solar farm	0.8494	0.01041	0.7628	0.09477	0.8699	0.02038	**0.8991**	0.01724	0.2783	0.63980

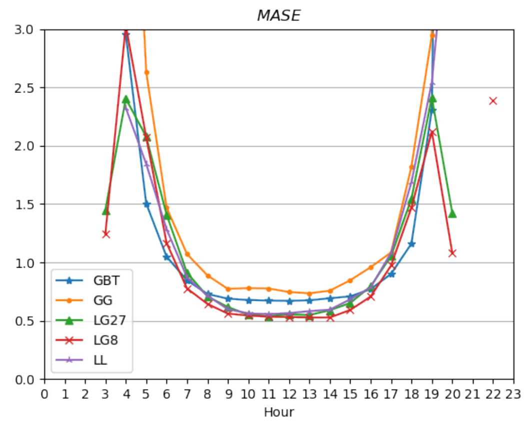
Table 10: R^2 values per plant at time T_0



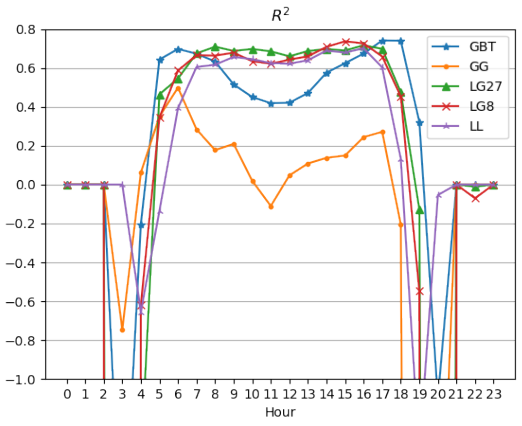
(a) MAE - Lower is better



(b) RMSE - Lower is better



(c) MASE - Lower is better



(d) R^2 - Higher is better

Figure 14: Distribution of MAE, RMSE, MASE and R^2 per hour at T_0

These anomalies have a few likely caused. Firstly as these are scaled metrics, relatively small absolute errors can have a larger apparent effect as the true irradiance approaches 0. Secondly, as shown in Figure 15b, the amount of relative noise at these hours is much higher making it harder for the model to learn.

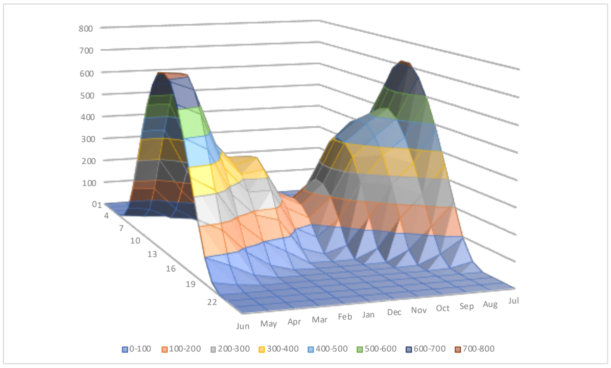
5.1.3 Monthly Performance

Looking at the data aggregated by month similar trends to the daily aggregate present themselves (Figure **??**). In absolute terms, the LG8 performs the best throughout the year, in some cases outperforming the GBT by $\approx 20\%$. However when looking at the scaled errors it would again appear to be less clear.

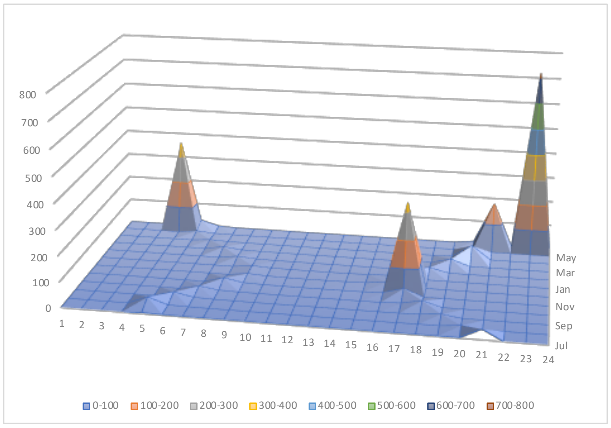
The same explanation for the models poor performance at dusk and dawn can be used to understand the the trends from summer to winter. Simply put, during winter there are less daylight hours and those with daylight receive much less irradiance than in summer (Figure 15a).

5.2 Forecast performance

As outlined in Section 4.2.3 the industry model does not have any architectural form of forecasting. An estimate of its forecasting accuracy is calculated by introducing a lag between features and labels. This means that a singe "lag" can be used to forecast for all hours of the day.

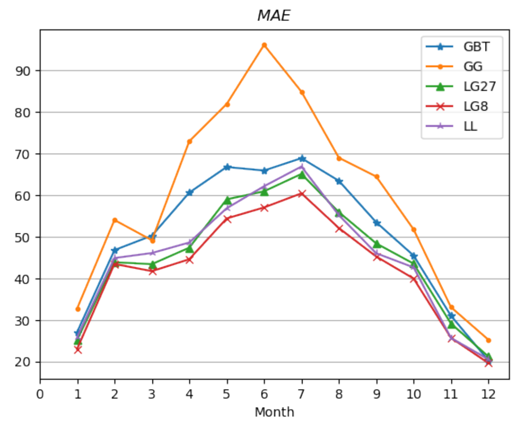


(a) Average irradiance throughout the day and year

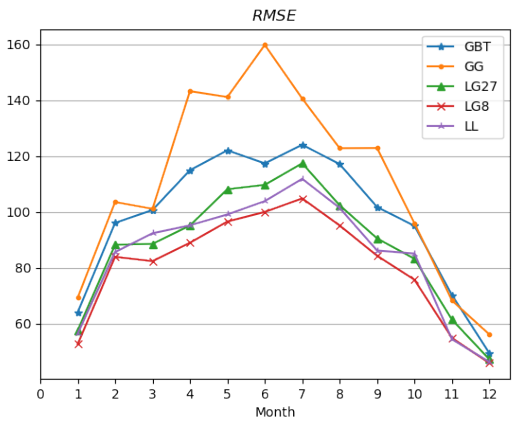


(b) Scaled irradiance ($\frac{min-max}{avg}$)

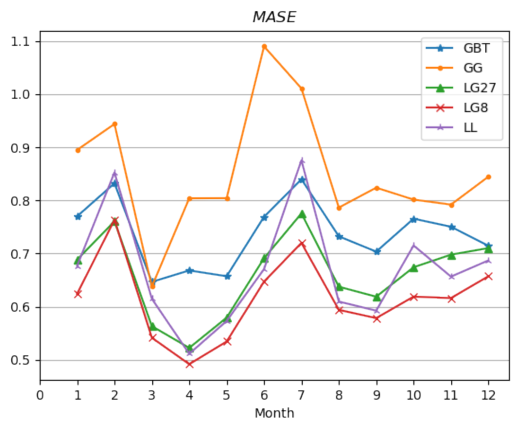
Figure 15: Average hourly irradiance throughout the year and hourly irradiance range scaled by the monthly average



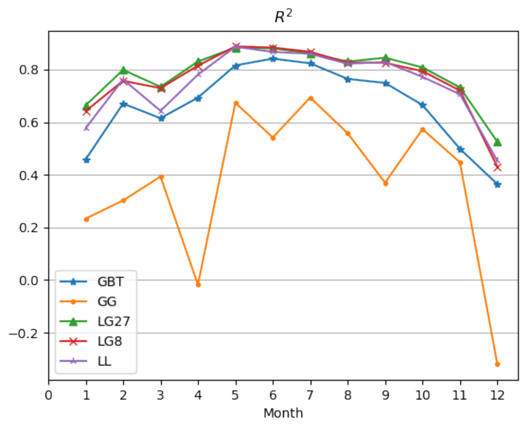
(a) MAE - Lower is better



(b) RMSE - Lower is better

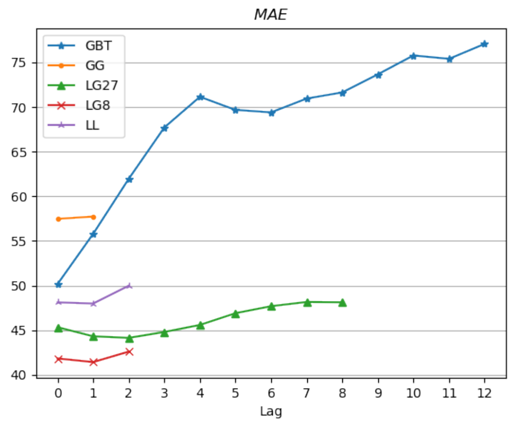


(c) MASE - Lower is better

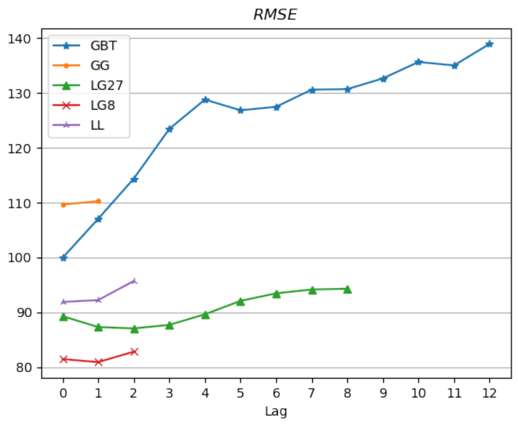


(d) R^2 - Higher is better

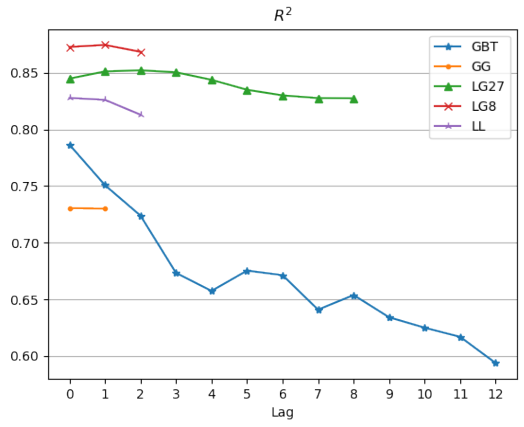
Figure 16: Distribution of MAE, RMSE, MASE and R^2 per month at T_0



(a) MAE - Lower is better



(b) RMSE - Lower is better



(c) R^2 - Higher is better

Figure 17: Lag error over time

In order to compare to the models forecast to the lagged GBT, their output steps are converted into the equivalent "lags". The output "step" of the models corresponds to a one hour offset from a three hour interval. As such $\text{STEP} \bmod 3 = 0$ maps to hours $[0, 3, 6, 9, 12, 15, 18, 21]$; accordingly $\text{STEP} \bmod 3 = 1$ maps to $[1, 4, 7, 10, 13, 16, 19, 22]$ and $\text{STEP} \bmod 3 = 2$ maps to $[2, 5, 8, 11, 14, 17, 20, 23]$. Steps are converted into lags by aggregating groups of three steps together.

As can be clearly seen from the graph in Figure 17, unlike any of the other models, as the lag between weather features and irradiance increases the performance of the GBT rapidly deteriorates. It is also worth noting that when comparing unadjusted steps, even the GG eventually outperforms the GBT.In other words, if T_0 was 3 o'clock then the GGs prediction for $5, 6, 7, 8$ would be comparable or better than those of a lagged GBT.

5.3 Other Considerations

This section will cover aspects of the model not directly related to performance. A key part of this work is to prepose models with practical applications. In addition to performance, implementation considerations must be made when choosing to productionize a model. This section will discuss the various models practicality.

5.3.1 Training Time

As shown in Table 11, for a single plant the GBT is by far the fastest model to train taking only a few minutes. The Local Local models per plant training time is more comparable that of a single fold of the Local Global and Global Global models. It is clear that the models with a global element have an advantage over the purely local models when per fold train time is considered. This lead will only increase as the number of plants does. If the number of plants where to double, one would expect to see total train time of GBT and LL to double whist the GG models time to train would be unchanged. The expected behaviour of the LG is unknown, a training epoch would be twice as long but it may need fewer epochs. The worst case is that training time would double however one would expect to see better than liner scaling.

Model	Per Plant	Per Fold
GBT	5 - 10 mins	2 - 3 hours
GG	-	1 - 2 hours
LG(8|27)	-	1 - 2 hours
LL	45 - 60 mins	18 - 24 hours

Table 11: Approximate training time for each model

5.3.2 Datasets

Whilst the same number of training examples exist for each plant, the training dataset file-size is vastly different depending on the model. The GBTs dataset is by far the smallest, consisting of 22 floats per instance. The training set used for both the LL and LG models where the same. As can be seen in Table 12 they are comparable in size to that of GGs. This is purely coincidental, as GGs size is only effected by the resolution of the images used and, to a lesser extent the world representation. The Local dataset will grow linearly with the number of plants.

Model	File Size
GBT	8MB
GG	3.8GB
LG27	4.2GB
LL / LG8	3.6GB ($\approx$ 150MB per plant)

Table 12: Dataset sizes for each model

It is also worth noting that changing the size of either the forecast horizon or the recurrent window will effect the dataset size. This can be seen when comparing the file-size of the data used to train LG8 vs LG27.

Conclusion

Overall the project was a success. The main goals of the project was to provide an alternative approaches that outperformed the industry model. In that way, the project achieve what it set out to do. Both of the LG models constantly outperform the industry models in all respects in addition to including a built in forecast.

The GG model, unfortunately did not perform as well as hoped. However it still presents an interesting approach for future work as it overcomes many of the shortcomings of the LG, specifically the liner increase in imputation time and training data size.

6.1 Future work

There is large scope for future work on many aspects of the project. Obvious next steps include, model optimisation and hyper parameter tuning. It is probable that further performance gains over the industry model could easily be obtained with simple parameter tuning. Additionally, it could also be worth while exploring alternative weather providers. Sources such as the European Space Agency or Met Office may provide access to richer datasets. Other areas for performance gains, especially in regard to forecasts is the relationship between image resolution and geo-special area.

Longer term expansion of the project includes making use of the irradiance predictions to predict power output of solar farms. This, would in turn, provide the ability to optimise the energy use, e.g. charging a battery vs transmitting to the grid.

Other more theoretical areas include exploration of the models internal representations used to forecast. Could a form of adversarial or multi-objective training be used to improve performance? Can they be used for other purposes, e.g wind energy forecasting? How stable is the forecast, and what effect does increasing the length of the recurrent window have? How effect of an approach is this for data imputation?

Another area of exploration is reviewing how well the models can generalise to unseen locations. If they can generalise well this approach could be adapted for use with domestic solar installations distributed throughout the country.

Reflection

This section will provide a review of the project from the authors perspective.

7.1 Project Appraisal

The scope of project was continuously evolving throughout the year. The constantly moving goal posts, despite being self inflicted, where at times difficult to deal with. However it allowed for the project to grow and change in scope to fit discoveries that where made as the project progressed.

The project was initially ran with a waterfall like approach, with tasks being systematically done moving down the GANTT chart from the project proposal. However this was a fairly ridged approach and would not work with the more dynamic nature of the project, as such I switched to a SRUM like approach for the second half of the project. This change allowed for easy adaptations to the plan as can be seen when comparing the tasks proposed to the completed work. This approach facilitated a major change to the fundamental architecture that was not realised until very late in the project. A downside to this change was that time for parameter tuning was cut.

A major difficulty throughout the project was model management. When working on a project of this scope, model management is vial. Numerous versions and variations of the same process exist. I created over 100 variations of the training data sets and 15 variations of the model architecture. It is very easy to become confused as to the version being worked on, or worse change parameters without tracking them. More than once did I have to re-run experiments due to poor version tracking. Additionally when working in note-book style environments, as was done for the majority of the project, version control can become even harder as it is commonplace to duplicate large sections of code.

Despite the constant changes and various difficulties encountered throughout the project, overall it has been great success. Numerous data processing methods and pipelines have been built to easily facilitate future work. Various automation scripts and easily configurable notebooks have been created and most importantly, a new industry beating approach has been proposed.

7.2 Self Reflection

The amount of work I proposed was ambitious, from the projects inception I had set grand goals for myself. This was due to a combination of both optimisation and naivety. As the bulk of the work began after the January exams it quickly became apparent that it would not be possible to achieve everything within the given timeframe and so the scope was changed. This was in part due to the fact that I underestimated how long it would take to run experiments.

That said, my personal time management was for the most part successful. All key dates and deadlines where met throughout the year in addition to managing coursework for other modules. However, occasionally I had a tendency to spend time exploring tangential areas to that of the main project. Whilst in the long run this supplemental work may prove useful, it had very little bearing on the completion of the project.

Overall the project was an enjoyable, stressful and rewarding experience. At times it felt like "I had bitten of more than I could chew," but in the end it was worth it. I have be able to learn many new technologies and skills as well as develop a deeper understanding of the theory behind the practical techniques used. Culminating in successfully building a model that outperformed industry. Additionally, the work I have done throughout the year has enabled me to take advantage of opportunities I never saw coming.

References

[1] Matt McGrath. Final call to save the world from 'climate catastrophe'. `https://www.bbc.co.uk/news/science-environment-45775309`, October 2018.

[2] Geraldine Ang, Dirk Röttgers, and Pralhad Burli. The empirics of enabling investment and innovation in renewable energy. *OECD Environment Working Papers*, No. 123(123), 2017.

[3] Paul Denholm, Matthew O'Connell, Gregory Brinkman, and Jennie Jorgenson. *Overgeneration from solar energy in California: a field guide to the duck chart*. National Renewable Energy Laboratory Golden, CO, 2015.

[4] Aurora Energy Research. Intermittency and the cost of integrating solar in the gb power market. `https://www.solar-trade.org.uk/wp-content/uploads/2016/10/Intermittency20Report_Lo-res_031016.pdf`, September 2016.

[5] Ian McKenzie Smith Keith Brown Edward Hughes, John Hiley. Electrical energy systems. In *Hughes Electrical and Electronic Technology*, chapter 39, pages 824–828. Pearson/Prentice Hall, 2008.

[6] Richard Conway and Sandy May. Using azure databricks, structured streaming, and deep learning pipelines to monitor 1,000+ solar farms in real time. `https://databricks.com/session/using-azure-databricks-structured-streaming-deep-learning-pipelines-to-monitor-1000-solar-farms-in-real-time`, October 2018.

[7] Daniel Gómez-Lorente, Isaac Triguero, Consolación Gil, and O. Rabaza. Multi-objective evolutionary algorithms for the design of grid-connected solar tracking systems. *International Journal of Electrical Power Energy Systems*, 61:371 – 379, 2014.

[8] Massimiliano De Benedetti, Fabio Leonardi, Fabrizio Messina, Corrado Santoro, and Athanasios Vasilakos. Anomaly detection and predictive maintenance for photovoltaic systems. *Neurocomputing*, 310:59 – 68, 2018.

[9] Cyril Voyant, Gilles Notton, Soteris Kalogirou, Marie-Laure Nivet, Christophe Paoli, Fabrice Motte, and Alexis Fouilloy. Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105:569 – 582, 2017.

[10] H. Denio. Aerial solar thermography and condition monitoring of photovoltaic systems. In *2012 38th IEEE Photovoltaic Specialists Conference*, pages 000613–000618, June 2012.

[11] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake. Leveraging smart meter data to recognize home appliances. In *2012 IEEE International Conference on Pervasive Computing and Communications*, pages 190–197, March 2012.

[12] Data source - weatherbit. `https://www.weatherbit.io`.

[13] Data source - darksky. `https://darksky.net`.

[14] Data source - openweathermap. `https://openweathermap.org`.

[15] Data source - met office. `https://www.metoffice.gov.uk`.

[16] Antonios K. Alexandridis and Achilleas D. Zapranis. Wavelet neural networks: A practical guide. *Neural Networks*, 42:1 – 27, 2013.

[17] John Berkowitz Zachary Chase Lipton and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015.

[18] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. *CoRR*, abs/1608.05343, 2016.

[19] Christophe Paoli, Cyril Voyant, Marc Muselli, and Marie-Laure Nivet. Forecasting of preprocessed daily solar radiation time series using neural networks. *Solar Energy*, 84(12):2146 – 2160, 2010.

[20] Ahmad Alzahrani, Pourya Shamsi, Cihan Dagli, and Mehdi Ferdowsi. Solar irradiance forecasting using deep neural networks. *Procedia Computer Science*, 114:304 – 313, 2017.

[21] Ricardo Marquez and Carlos F.M. Coimbra. Forecasting of global and direct solar irradiance using stochastic learning methods, ground experiments and the nws database. *Solar Energy*, 85(5):746 – 756, 2011.

[22] Matthew Reno, Clifford Hansen, and Joshua Stein. Global horizontal irradiance clear sky models : implementation and analysis. 01 2014.

[23] William Holmgren, Clifford Hansen, and Mark Mikofski. pvlib python: a python package for modeling solar energy systems. *Journal of Open Source Software*, 3:884, 09 2018.

[24] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[25] CS231n convolutional neural networks for visual recognition. `http://cs231n.github.io/`.

[26] Met Office. Nowcasting. `https://www.metoffice.gov.uk/learning/making-a-forecast/hours-ahead/nowcasting`.

[27] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, 2015-January:802–810, 2015.

[28] W.-C. Woo and W.-K. Wong. Operational application of optical flow techniques to radar-based rainfall nowcasting. *Atmosphere*, 8(3), 2017.

[29] David Fleet and Yair Weiss. Optical flow estimation. In *Handbook of mathematical models in computer vision*, pages 237–257. Springer, 2006.

[30] Kelson R. T. Aires, Andre M. Santana, and Adelardo A. D. Medeiros. Optical flow using color information: Preliminary results. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 1607–1611, New York, NY, USA, 2008. ACM.

[31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[32] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[33] Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015.

[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[35] Alexandre de Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Y Bengio. Artificial neural networks applied to taxi destination prediction. 07 2015.

[36] Mikel Canizo, Isaac Triguero, Angel Conde, and Enrique Onieva. Multi-head CNN-RNN for multi-time serise anomaly detection. 2019. Submitted.

[37] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics Data Analysis*, 120:70 – 83, 2018.

[38] Christoph Bergmeir and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192 – 213, 2012. Data Mining for Software Trustworthiness.

[39] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688, 2006.

[40] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[41] Julián Luengo, Salvador García, and Francisco Herrera. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and Information Systems*, 32(1):77–108, Jul 2012.

[42] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.