FLEXIBILITY EXPLOITATION FOR RESIDENTIAL, TERTIARY AND INDUSTRIAL BUILDINGS

FLEXBIT PROJECT

CETPartnership Joint Call 2023

Deliverable 2.2 Advanced Forecasting Tools

Lead Author: Electrum/Mateusz Witkowski https://orcid.org/0009-0005-4439-1458
Contributed authors:

Pio Alessandro Lombardi orcid: https://orcid.org/0000-0003-4598-9759
Bartłomiej Arendarski: https://orcid.org/0000-0002-3145-916X
Tomasz Sikorski orcid: https://orcid.org/0000-0002-4423-7216
Gaetano Zizzo orcid: https://orcid.org/0000-0003-4413-4855
Francesco Saverio Cannizzaro orcid: https://orcid.org/0009-0004-1528-1682
Krystian Chudzik orcid: https://orcid.org/0000-0003-3833-0972
Szymon Smagowski orcid: https://orcid.org/0009-0003-1965-8099
Vishnu Suresh orcid: https://orcid.org/0000-0003-2891-9206
Fachrizal Aksan orcid https://orcid.org/0000-0001-6357-2637
Status: 25.05.2026

DOI: **10.5281/zenodo.20506602**

This research was funded by CETPartnership, the Clean Energy Transition Partnership under the 2023 joint call for research proposals, co-funded by the European Commission (GA N°101069750) and with the funding organizations detailed on https://cetpartnership.eu/funding-agencies-and-call-modules.

Co-funded by the European Union

Supported by:

Federal Ministry for Economic Affairs and Climate Action

on the basis of a decision by the German Bundestag

Ministero delle Imprese e del Made in Italy

GSRI GENERAL SECRETARIAT FOR RESEARCH AND INNOVATION

FlexBIT Consortium

Table of Contents

1 Introduction

This deliverable presents the advanced forecasting tools developed within the FlexBIT project, focusing on ultra-short-term and short-term prediction of photovoltaic (PV) generation and electricity demand. The forecasting module constitutes a key component of the FlexBIT digital platform, providing predictive capabilities that support real-time control, optimization, and flexibility management across residential, tertiary, and industrial energy systems.

The scope of this deliverable covers data originating from multiple sources, including platform operational data, IoT and SCADA/EMACS measurements, as well as external data streams such as weather services and market signals. The document describes the data sources, feature engineering techniques, forecasting methodologies, model selection strategies, and validation procedures applied across different demonstrators within the project.

The forecasting tools are designed to operate under the specific requirements of the FlexBIT platform, where high temporal resolution, low latency, and robustness to variability are essential. In particular, the models support rolling predictions at minute-level granularity, enabling their integration into the real-time advisory and control loop of the platform.

This deliverable builds upon the system architecture and control concepts defined in Deliverable D2.1, providing the predictive layer that feeds optimization and decision-making modules. It is closely linked with other activities within Work Package 2, including data integration, flexibility identification, and the development of control algorithms.

By combining advanced machine learning techniques with scalable data processing pipelines, the forecasting module contributes to the overall objectives of FlexBIT, enabling improved utilization of renewable energy sources, enhanced flexibility exploitation, and more efficient operation of distributed energy systems.

2 Objectives and Functional Requirements

2.1 Forecasting Objectives

The forecasting module within FlexBIT serves as the primary predictive layer feeding the optimization and control infrastructure described in D2.1. Rather than pursuing forecasting as an end in itself, the module is designed around concrete operational needs of the platform — each forecasting horizon and target variable is motivated by a specific downstream use case.
The principal objectives are:

- **Short-term prediction for real-time operational decisions — both solar forecasting pipelines produce rolling 1-minute updates with a 15-minute ahead horizon, directly supporting storage dispatch decisions, demand response activation, and real-time self-consumption optimization within the fast advisory loop.**
- **Intra-day updates for scheduling — forecasts are refreshed continuously throughout the day, allowing the optimization engine to revise dispatch schedules as weather conditions and load patterns evolve, rather than committing to a fixed plan.**
- **Uncertainty-aware forecasting — given the inherent variability of solar generation and the diversity of load profiles across demonstrators, the forecasting framework is designed to account for prediction uncertainty, supporting more robust optimization decisions rather than relying on point forecasts alone.**

2.2 Target Forecast Variables

The forecasting module predicts two distinct target variables — PV generation and load demand — each tied to a specific role in the FlexBIT advisory loop.
For PV generation, the target is the active power output of the photovoltaic installation, expressed in kW. This is the quantity recorded by the EMACS SCADA system at the Alu-Frost demonstrator and stored in the operational database. The Stanford SKIPP'D benchmark uses the same target variable, which enables direct comparison against the reference baseline.

Load demand forecasting also targets active power, expressed in kW, but from the consumption side of the point of connection. This variable represents the electrical demand observed at the demonstrator site and is used to anticipate short-term changes in local energy consumption.

2.3 Time Horizons and Resolutions

The temporal design of the forecasting module is driven by the operational requirements of the FlexBIT advisory loop, where predictions must be both timely and sufficiently granular to support real-time dispatch decisions.
For solar/PV generation forecasting, both pipelines operate on 1-minute input resolution and issue new predictions every minute on a rolling basis:

- **Architecture B (feature-based) — produces a multi-output forecast simultaneously covering every minute within the 1–15 minute ahead window, giving the optimization layer a complete short-term generation profile at each update step.**
- **Architecture A (image-based) — produces forecast targeting the 1-15 minutes ahead of window horizon, updated every minute as new sky imagery becomes available.**

Both pipelines are expected to deliver predictions within a few seconds of receiving new input data, ensuring compatibility with the 1-minute rolling update cycle.

For load demand forecasting, the implemented time-series pipeline also operates on 1-minute input resolution and supports rolling short-term prediction updates. In contrast to the PV image-based setup, load forecasting is based exclusively on numerical measurements from the point of connection, using recent lagged demand values together with cyclical timestamp features.

The load forecasting workflow covers two forecast configurations: a 1-minute ahead model for immediate demand tracking and a 15-minute ahead model for short-term advisory and optimization use. Both configurations are updated as new measurements become available and are designed to provide lightweight CPU-based inference suitable for operational deployment. The 15-minute ahead demand forecast is used together with the corresponding PV generation forecast to estimate the near-term site power balance, while the 1-minute ahead forecast provides a higher-frequency view of rapid demand fluctuations.

3 Data Sources and Input Features

3.1 Data Sources

The forecasting framework developed within the FlexBIT project relies on a heterogeneous set of data sources reflecting both the distributed architecture of the platform and the diversity of demonstrators, including residential, tertiary, and industrial environments. The input data combines real-time operational measurements, external environmental information, and contextual signals, enabling the forecasting models to capture both physical system dynamics and user-driven consumption patterns.

The primary data sources consist of on-site measurements collected via SCADA systems (including EMACS) and IoT devices. These data streams include photovoltaic (PV) power output, electrical load consumption, voltage and current measurements, and state-of-charge indicators for storage systems. Signals related to electric vehicle (EV) charging and discharging are also included where available.

External weather data is incorporated to capture the environmental drivers of renewable generation. This includes solar irradiance (SWD/GHI), cloud cover, temperature, wind speed, and humidity. In selected demonstrators, image-based data such as fisheye sky camera images are used to support ultra-short-term solar nowcasting.

Additional contextual and operational data are also considered, including:

- **time-related features (hour of day, day of week, seasonal indicators),**
- **occupancy patterns (where available),**
- **industrial process schedules,**
- **external signals such as electricity prices or demand response activations (depending on the pilot).**

This combination ensures that both exogenous drivers and system-level dynamics are represented in the forecasting process.

3.2 Data Characteristics

The datasets used in FlexBIT forecasting are characterized by high temporal resolution and strong variability, reflecting real operating conditions of distributed energy systems. Typically, measurements are available at a 1-minute resolution, which enables ultra-short-term forecasting but also introduces challenges related to data volume and noise.

The data exhibits strong temporal dependencies, including autocorrelation and recurring daily or seasonal patterns. At the same time, it is highly non-stationary, particularly for PV generation, where rapid fluctuations are caused by cloud movement and changing weather conditions.

Another important aspect is the heterogeneity across sites. Differences in building types, installed technologies, climate conditions, and user behavior result in diverse data distributions across demonstrators. In addition, real-world data quality issues must be considered, including:

- **missing values,**
- **sensor noise,**
- **communication delays.**

These characteristics necessitate robust preprocessing and flexible modelling approaches capable of adapting to different environments.

3.3 Feature Engineering

Feature engineering plays a crucial role in improving forecasting performance by enabling models to capture relevant temporal, physical, and statistical patterns in the data.

For time-series-based models, lagged observations of PV power and load are used to represent short-term temporal dependencies through a sliding window approach. Time-based features are incorporated to reflect periodic behavior, including hourly, daily, and seasonal cycles, often encoded using cyclical transformations.
Weather-related features are included to capture the influence of external conditions on system behavior. In addition, statistical indicators such as moving averages, rolling standard deviations, and ramp rates are used to describe short-term dynamics and variability.
In the case of image-based forecasting pipelines, additional features are extracted from visual data, such as cloud motion vectors and sky condition indicators.
Depending on the model type, features are represented differently:

- **sequential models (e.g., LSTM, GRU) operate directly on ordered time-series sequences,**
- **tabular models (e.g., CatBoost, linear regression) rely on engineered feature vectors.**

This flexible representation ensures compatibility with the different modelling approaches described in Section 4.

3.4 Data Preprocessing

Before being used for model training and inference, all data undergoes a consistent preprocessing pipeline to ensure reliability and consistency.

The preprocessing process includes data cleaning, where invalid or corrupted measurements are removed, and missing values are handled using interpolation or imputation techniques. Data normalization or standardization is applied to ensure compatibility with machine learning algorithms.

Time alignment is performed to synchronize multiple data sources, ensuring that measurements from SCADA systems, weather services, and other inputs are temporally consistent. Outlier detection methods are used to filter unrealistic spikes or drops in the data.

For time-series models, windowing techniques are applied to construct input sequences based on sliding windows. Finally, the dataset is split into training and testing subsets using a chronological approach, ensuring that future data is not used during model training.

These steps ensure that the input data is clean, consistent, and suitable for real-time forecasting applications within the FlexBIT platform.

3.5 Data Governance and Privacy

Data management within the FlexBIT framework follows strict governance and privacy principles aligned with EU regulations and the distributed architecture of the platform.

The system ensures compliance with GDPR requirements, including the protection and anonymization of user-related data where necessary. Secure data transmission mechanisms are implemented to protect communication between devices and platform components, and access to data is controlled through role-based policies.

The platform supports decentralized data handling, allowing for edge-based processing and reducing reliance on centralized storage of raw data. Clear data ownership and sharing policies are defined among project partners, ensuring controlled data exchange within energy communities.

Interoperability is ensured through standardized data formats and APIs, enabling seamless integration with SCADA systems and energy management platforms.

Overall, this approach ensures that forecasting services are secure, scalable, and compliant, while supporting the distributed and heterogeneous nature of the FlexBIT system.

3.6 Image data acquisition and processing pipeline

3.6.1 Image-based forecasting architecture for ultra-short-term PV nowcasting

Overview and Purpose

The image-based forecasting pipeline extends the FlexBIT advanced forecasting framework by introducing a dedicated sky-image acquisition, storage, preprocessing, training, and inference architecture. Its purpose is to support ultra-short-term PV generation forecasting, especially for the 1–15 minute operational horizon required by the FlexBIT advisory and control loop.

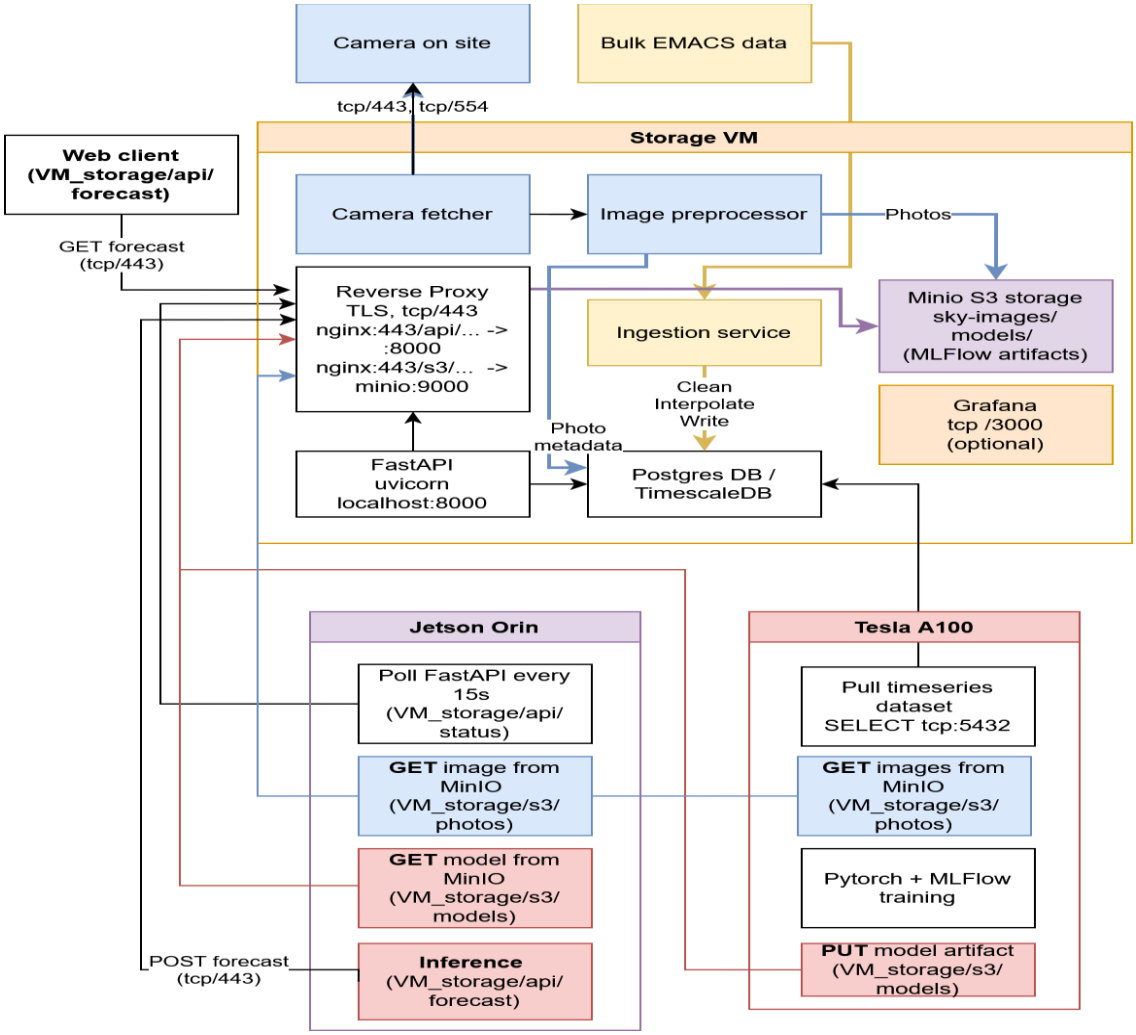

Figure 1 image-based forecasting pipeline

3.6.2 Data Acquisition and Preprocessing

The architecture is centred around the Storage VM, which acts as the main integration point for image data, time-series data, metadata, APIs, and model artifacts. A camera installed on site continuously provides sky images through secure network communication. The Camera Fetcher component retrieves images from the on-site camera using TCP/443 or TCP/554 and passes them to the Image Preprocessor.
The preprocessing stage prepares images for downstream forecasting tasks by standardizing image quality, extracting metadata, and making the data suitable for machine learning workflows.

3.6.3 Data Storage and Integration

Processed images are stored in MinIO S3 storage under dedicated paths for photos and model artifacts. MinIO provides an S3-compatible object storage layer used both by the training environment and the edge inference environment.
Image metadata and relevant time-series information are written to PostgreSQL / TimescaleDB, where they are aligned with EMACS/SCADA measurements and other operational signals. The ingestion service is responsible for cleaning, interpolating, and writing structured data to the database.
The Bulk EMACS data stream provides historical and operational time-series measurements, which are used together with image-derived information to support model training and validation. This enables the forecasting framework to combine visual sky conditions with measured PV generation and environmental variables.

3.6.4 Model Training Environment

The Tesla A100 environment is used for computationally intensive model training. It pulls time-series datasets from TimescaleDB and retrieves historical sky images from MinIO.
Trained PyTorch and MLflow models are then stored back in MinIO as versioned model artifacts. This supports reproducibility, model lifecycle management, and future comparison of model versions.

3.6.5 Edge Inference Layer

The Jetson Orin device represents the edge inference layer. It periodically checks the FastAPI service for inference status, retrieves the latest images and model artifacts from MinIO, and performs local inference.
The resulting PV forecast is sent back to the Storage VM through the API endpoint. This enables decentralized, low-latency execution of forecasting models close to the operational environment.

3.6.6 API and Communication Layer

External access is handled through a Reverse Proxy based on Nginx with TLS termination. The proxy exposes selected API and S3 routes over TCP/443, ensuring secure communication between the web client, edge device, training environment, storage services, and forecasting API.
The FastAPI service provides the operational interface for forecast requests, inference status, and metadata exchange.

3.6.7 Contribution to FlexBIT Objectives

The architecture supports the FlexBIT objectives by enabling:

- **continuous acquisition of high-resolution sky image data,**

- **integration of image data with EMACS/SCADA time-series measurements,**
- **centralized model training on GPU infrastructure,**
- **edge deployment of trained models on Jetson Orin,**
- **secure API-based communication through the Storage VM,**
- **storage of datasets and model artifacts in an S3-compatible environment,**
- **operational use of forecasts in the FlexBIT advisory and control loop.**

4 Time – Series Forecasting Methodology

This section presents the time-series-based forecasting framework developed within the FlexBIT project for photovoltaic (PV) power and load demand prediction at ultra-short-term horizons. The methodology is designed to support both 1-minute-ahead and 15-minute-ahead forecasting tasks, which are essential for real-time control and short-term operational planning in distributed energy systems.

The proposed approach relies exclusively on historical time-series measurements of PV power and related variables, in contrast to parallel efforts within the project that utilize image-based inputs such as fisheye sky images. While these approaches differ in input modality, they target the same forecasting objectives and are complementary within the overall FlexBIT architecture.

A unified modelling framework is adopted for both forecasting horizons, ensuring methodological consistency while allowing flexibility in model adaptation. The workflow consists of data preprocessing, feature construction, model development, and performance evaluation, and includes both baseline and advanced machine learning models to systematically assess accuracy complexity trade-offs.

4.1 Methodological Approach

The forecasting framework is formulated as a supervised learning problem based on time-series data, where the objective is to predict future photovoltaic (PV) power output using historical observations. The methodology is designed to support multiple forecasting horizons within a unified framework, specifically 1-minute-ahead and 15-minute-ahead predictions.

Let P_t denote the PV power at time step t. The forecasting task is defined as:

$$\widehat{P_{t+h}} = f(P_t, P_{t-1}, P_{t-2}) \quad \textbf{(1)}$$

where $h \in \{1,15\}$ represents the forecasting horizon. The same modeling pipeline is applied for both horizons, with the prediction target adjusted accordingly.
The input to the models consists of lagged observations of the PV power signal, constructed using a sliding window approach as described in Section 3. These inputs capture short-term temporal dependencies that are critical for ultra-short-term forecasting.

The forecasting process is implemented as a structured and repeatable modeling pipeline applied consistently across all considered models and forecasting horizons.
The workflow consists of the following stages:

- **Input Structuring: Preprocessed and feature-engineered data (as described in Section 3) are organized into model-specific input formats. Depending on the model type, this includes either sequential representations preserving temporal order or tabular representations based on lagged features.**
- **Model Training: Each forecasting model is trained using historical data within a supervised learning framework. The models learn a mapping from input features to future PV power values corresponding to the selected forecasting horizon h.**
- **Prediction Generation: Once trained, the models are used to generate predictions on unseen data. Forecasts are produced independently for each time step without recursive multi-step propagation, ensuring stability for ultra-short-term horizons.**

- **Model Evaluation: Predictions are compared against observed values on a hold-out test dataset. A consistent evaluation protocol is applied across all models to ensure comparability of results.**

This pipeline is applied identically to both 1-minute-ahead and 15-minute-ahead forecasting tasks, with the forecasting horizon chosen as necessary depending on the task.

4.2 PV Generation Forecasting

Alongside the image-based CNN–LSTM pipeline described in Chapter 5, a feature-only track addresses the same PV generation target without relying on sky images. It runs on the SKIPP'D data using the stratified train/validation/test split and the engineered feature set introduced in Chapter 5, keeping only the numerical inputs — lagged PV values, rolling statistics, sun elevation and azimuth, clear-sky index, and short-term variability. The motivation is twofold: to serve sites without an all-sky camera through a light model that runs on a CPU, and to measure how much of the 15-minute accuracy comes from the numbers alone rather than the images. Three regressors are trained on these features, each at the 1-minute and 15-minute horizons, following the same data split and evaluation protocol as the image pipeline.
Two of the three are gradient-boosted decision trees. XGBoost grows its trees in sequence, with every new tree correcting the errors left by the ones before it. LightGBM follows the same idea but bins the features first and grows each tree leaf by leaf, which makes it noticeably faster on minute-resolution data. Random Forest takes the opposite route, training many trees independently on bootstrapped samples and averaging them, trading the boosting mechanism for variance reduction and serving as a non-boosted reference. Running all three on identical features shows whether the accuracy rests on boosting in particular or holds across tree-based models more broadly. The data preparation, training procedure, and results for this track are reported together with the image model in Chapter 5.

4.3 Load Demand Forecasting

4.3.1 Baseline Models

Baseline models are employed to establish reference performance levels for ultra-short-term load demand forecasting at both time horizons. These models are characterized by low computational complexity and strong interpretability, making them essential benchmarks for evaluating the added value of more advanced machine learning approaches. Given the high temporal resolution and strong autocorrelation of load demand at short horizons, baseline models often provide competitive performance and therefore serve as critical points of comparison.

- **Persistence Model**

The persistence model represents the simplest forecasting approach, assuming that future load demand remains equal to the most recent observed value. The forecast is defined as:

$$\hat{P}_{t+h} = P_t \tag{2}$$

where $P_{t+h}\hat{}$ is the predicted PV power at time t, and P_t is the observed value at time t.This model is particularly effective for ultra-short-term horizons, where PV power changes are often gradual under stable weather conditions. Due to its simplicity and strong performance in short-term forecasting, persistence serves as a fundamental benchmark.

- **Linear Regression**

Linear regression models the relationship between future load demand and past observations as a linear combination of lagged inputs. The model is expressed as:

$$\widehat{P_{t+h}} = \beta_0 + \sum_{i=1}^{n} \beta_i P_{t-i} \quad \textbf{(3)}$$

where:

- **β0 is the intercept term,**
- **βi are the regression coefficients,**
- **n is the number of lagged inputs.**

The model parameters are estimated by minimizing the Mean Squared Error (MSE) between predicted and observed values:

$$\min_{\beta} \sum_{t} \left(P_{t+h} - \widehat{P_{t+h}}\right)^2 \quad \textbf{(4)}$$

Linear regression provides a simple yet effective way to capture short-term temporal dependencies and serves as a baseline for assessing nonlinear models.

- **Ridge Regression**

Ridge regression extends the linear regression model by introducing L2 regularization to prevent overfitting and improve generalization. The objective function is defined as:

$$\min_{\beta} \sum_{t} \left(P_{t+h} - \widehat{P_{t+h}}\right)^2 + \lambda \sum_{i=1}^{n} \beta_i^2 \quad \textbf{(5)}$$

where:

- **λ ≥ 0 is the regularization parameter controlling the penalty on coefficient magnitude.**

The inclusion of the regularization term discourages large coefficients, leading to more stable predictions, particularly when input features are correlated.

4.3.2 Advanced Forecasting Models

Advanced forecasting models are introduced to capture nonlinear relationships and complex temporal dependencies that may not be adequately represented by baseline approaches. These models extend the modeling capability beyond linear assumptions and are particularly relevant for handling rapid fluctuations in load demand caused by changing occupancy and industrial activity.

The selected models represent different modeling paradigms, including sequential deep learning and nonlinear machine learning methods, enabling a comprehensive assessment of forecasting performance.

- **Long Short-Term Memory (LSTM)**

The Long Short-Term Memory (LSTM) model is implemented as a sequential neural network architecture designed to capture short-term temporal dependencies in load demand data. The model operates on input sequences constructed from lagged observations, with each sample represented as a time series window of fixed length.

As illustrated on Figure 2 LSTM sequential model architecture**, the input to the model consists of a univariate time series with shape (SEQ_LEN,1), where SEQ_LEN denotes the number of past time**

steps used for prediction. This sequential input is processed by an LSTM layer containing 32 hidden units, which extracts temporal features by maintaining an internal memory state.

The output of the LSTM layer is a feature vector of dimension 32, which is then passed to a fully connected dense layer with a single neuron. This final layer maps the learned representation to a scalar output corresponding to the predicted PV power at the forecasting horizon *t+h*.

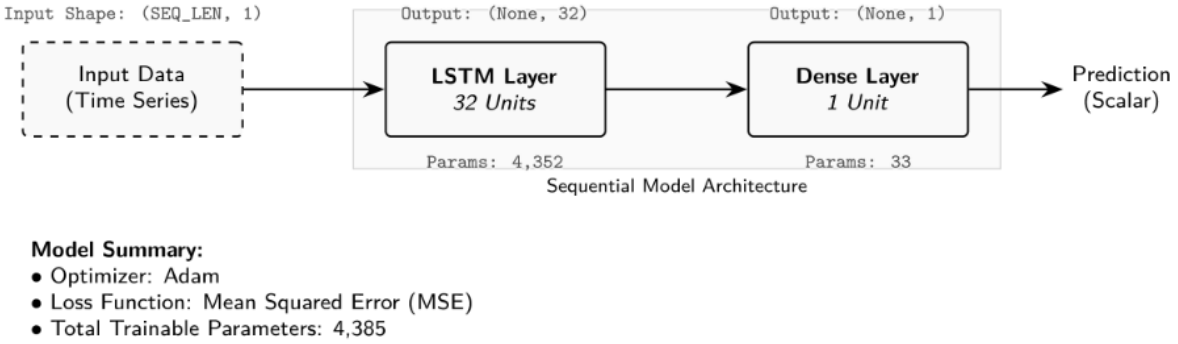

Figure 2 LSTM sequential model architecture

The model can be expressed as:

$$\widehat{P_{t+h}} = f_{\text{LSTM}}(P_{t-1}, P_{t-2}, \dots, P_{t-n}) \quad \textbf{(6)}$$

where $f_{LSTM}(\cdot)$ represents the nonlinear transformation learned by the network.

The model is trained using the Adam optimization algorithm with Mean Squared Error (MSE) as the loss function. The total number of trainable parameters in the network is 4,385, reflecting a relatively lightweight architecture suitable for high-frequency forecasting tasks.

This configuration balances model complexity and computational efficiency, making it appropriate for ultra-short-term forecasting applications within real-time energy management systems.

- **Gated Recurrent Unit (GRU)**

The Gated Recurrent Unit (GRU) model is employed as a computationally efficient alternative to the (LSTM) architecture for modelling temporal dependencies in photovoltaic (PV) power data. Similar to the LSTM, the GRU processes sequential input constructed from lagged observations of the PV power signal.

The input to the model consists of a univariate time series with shape (SEQ_LEN,1), where SEQ_LEN denotes the number of past time steps used for prediction. The sequential input is processed by a GRU layer with 32 hidden units, which captures temporal relationships using a simplified gating mechanism that combines memory and update operations.

The output of the GRU layer is a feature vector of dimension 32, which is subsequently passed to a dense layer with a single neuron to produce the final scalar prediction corresponding to the PV power at the forecasting horizon *t+h*.

The model can be expressed as:

$$\widehat{P_{t+h}} = f_{\mathbf{GRU}}(P_{t-1}, P_{t-2}, \dots, P_{t-n}) \quad \textbf{(7)}$$

The GRU architecture used in this study contains a total of 3,393 trainable parameters, which is lower than the corresponding LSTM model due to its reduced gating structure. This results in improved computational efficiency while maintaining the ability to model short-term temporal dependencies. The model is trained using the Adam optimizer with MSE as the loss function, ensuring consistency with the training setup used for other neural network models.

- **CatBoost Regressor**

To complement the sequence-based deep learning models, a gradient boosting approach is employed using the CatBoost regression framework. Unlike recurrent neural networks, CatBoost operates on tabular representations of the data, where temporal dependencies are captured through lagged input features.

As illustrated on Figure 3 Cat Boost regression framework, the input to the model consists of a feature vector x_t constructed from lagged PV power observations, i.e.,

$$x_t = [P_{t-1}, P_{t-2}, \dots, P_{t-n}] \quad \textbf{(8)}$$

This feature vector is processed by an ensemble of decision trees built sequentially through a boosting procedure. At each iteration k, a new tree is trained to reduce the residual errors of the previous ensemble, progressively improving the model's predictive performance. The overall model can be expressed as:

$$\widehat{P_{t+h}} = \sum_{k=1}^{K} f_k(x_t) \quad \textbf{(9)}$$

where:

- **$f_k(\cdot)$ represents the k-th decision tree,**
- **K is the total number of boosting iterations.**

In this study, the CatBoost model is configured with 300 boosting iterations, a tree depth of 6, and a learning rate of 0.05. The model is trained using the Root Mean Square Error (RMSE) as both the loss function and evaluation metric.

The boosting process enables the model to capture nonlinear relationships between lagged inputs and future PV power values without explicitly modelling temporal sequences. This makes CatBoost particularly effective for handling complex interactions in high-frequency time-series data while maintaining computational efficiency.

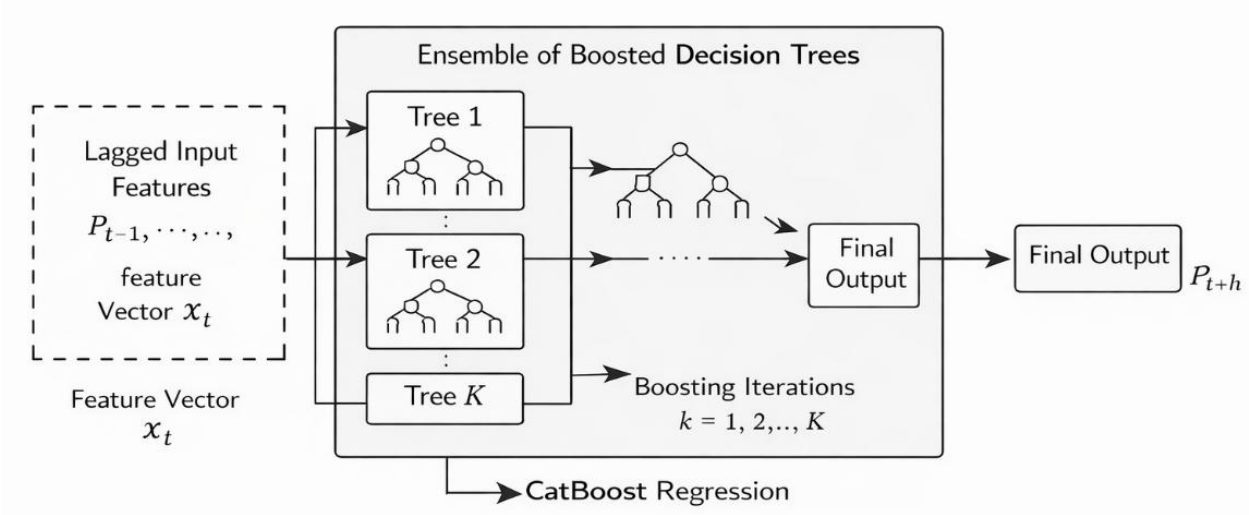

Figure 3 Cat Boost regression framework

- **Histogram Gradient Boosting Regressor**

As an additional tree-based benchmark, a Histogram Gradient Boosting Regressor is used. This model follows the same boosting principle as other gradient-boosted decision tree methods, but first discretizes continuous input features into histogram bins, which makes training efficient on tabular time-series data. The model operates on the same lagged and time-encoded feature vectors as the CatBoost regressor, allowing nonlinear relationships between recent power/load observations and the forecast target to be captured without explicitly modelling the sequence structure. In this study, it serves as a lightweight CPU-based comparison model for evaluating whether the observed improvements are specific to CatBoost or are representative of boosted tree methods more generally.

4.3.3 Model Selection Rationale

The selection of forecasting models is guided by the need to balance predictive performance, computational efficiency, and deployment feasibility within the FlexBIT platform. Given the diversity of application scenarios and the requirement for real-time operation, a heterogeneous set of models is considered, spanning from simple baselines to more advanced machine learning approaches.
Baseline models, including persistence and linear regression, are selected to provide reference performance levels and to establish the effectiveness of more complex methods. These models are computationally lightweight and highly interpretable, making them suitable for rapid evaluation and benchmarking in ultra-short-term forecasting tasks.

Advanced models are chosen to capture nonlinear relationships and temporal dependencies inherent in the forecasting target. Recurrent neural networks such as LSTM and GRU are included due to their ability to model sequential patterns and short-term temporal dynamics. In parallel, the CatBoost regressor is incorporated as a powerful nonlinear model operating on tabular data, offering strong performance without requiring explicit sequence modelling.

A key aspect of the model selection is the consideration of deployment constraints associated with the FlexBIT architecture. In particular, forecasting modules are expected to operate in a decentralized manner, potentially on edge devices such as embedded computing platforms (e.g., NVIDIA Jetson). This imposes limitations on computational resources, memory usage, and latency.

To address these constraints, the selected models cover a spectrum of complexity:

- **Lightweight models: (persistence, linear regression) enable fast inference and low resource consumption,**
- **Moderate-complexity models: (GRU, CatBoost) provide improved predictive capability with manageable computational cost,**
- **Higher-capacity models: (LSTM) allow for more expressive temporal modeling when computational resources permit.**

This layered selection ensures flexibility in deployment and facilitates future integration with distributed or federated learning frameworks, where models may be trained or fine-tuned locally on edge devices while contributing to a global forecasting system.

Overall, the chosen set of models supports a trade-off between accuracy, interpretability, and scalability, aligning with the operational requirements and architectural vision of the FlexBIT platform.

4.3.4 Training strategy

The training strategy is designed to ensure a fair and consistent evaluation of all forecasting models while maintaining alignment with the temporal nature of the data and the operational requirements of the FlexBIT platform.

- **Data Splitting and Temporal Consistency**

The dataset is divided into training and testing subsets using a chronological split, where the first 80% of the data is used for training and the remaining 20% is reserved for testing. This approach preserves the temporal ordering of the time series and prevents information leakage from future observations into the training process. The same data split is applied across all models to ensure comparability of results.

- **Model-Specific Training Configurations**

Different model classes are trained using configurations appropriate to their structure while maintaining overall consistency in the experimental setup.

- **Linear models (Linear Regression, Ridge Regression) are trained using standard least-squares optimization, with Ridge Regression including an $L2$ regularization term to improve generalization.**
- **Recurrent neural networks (LSTM, GRU) are trained using the Adam optimizer with Mean Squared Error (MSE) as the loss function. A fixed number of training epochs is used, along with batch-based optimization to ensure stable convergence.**
- **CatBoost is trained using gradient boosting with Root Mean Square Error (RMSE) as the loss function, following the configuration described in Section 4.3.**

- **Consistency Across Models**

To ensure a fair comparison, all models are trained using the same input features and data preprocessing pipeline (Section 3). Identical training and testing datasets are used and the same forecasting horizons (h=1 and h=15) are utilized. This guarantees that differences in performance are attributable to model characteristics rather than inconsistencies in data handling.

- **Training Efficiency and Deployment Considerations**

The training configurations are selected to balance model performance with computational efficiency. Given the potential deployment of forecasting models on distributed or edge computing platforms within the FlexBIT architecture, training procedures are designed to remain scalable and adaptable. In particular, the use of relatively lightweight architectures and controlled training settings enables future extension toward decentralized or federated learning scenarios, where models may be trained or updated locally on resource-constrained devices.

The adopted training strategy ensures methodological consistency, reproducibility, and alignment with real-world deployment constraints. By maintaining uniform training conditions across models while respecting their structural differences, the framework supports a reliable and interpretable comparison of forecasting approaches.

5 Sky Image Based Forecasting Methodology

In this section, the algorithm is described as implementing a deep learning pipeline using the PyTorch framework to perform short-term forecasting over a 1–15-minute time horizon based on sky images and PV measurements system. In this study, we performed several key stages (see Figure 4) to forecast PV power output based on historical sky images and PV power data:

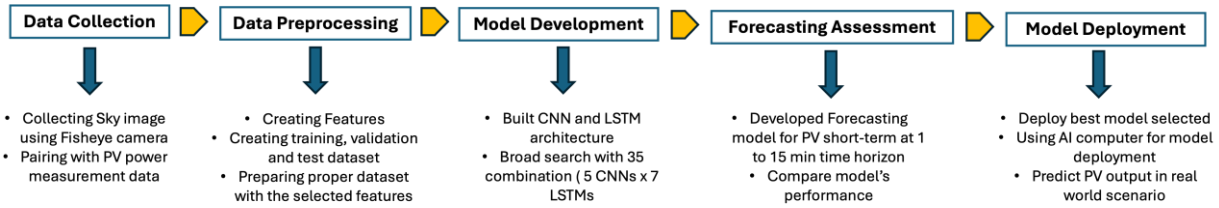

Figure 4 Sky Image Based Forecasting Methodology

5.1 Data Collection

Accurate and reliable data are essential for developing advanced forecasting models. In the context of short-term PV power forecasting based on sky images, synchronized acquisition of sky images and PV power output measurements is required, as these two variables constitute the primary inputs of the proposed framework. Nevertheless, several important challenges remain, including determining appropriate methods for capturing sky images and establishing an effective relationship between sky-image data and the corresponding PV power output.

According to a previous study presented by Nie et al., sky images captured using ground-based fisheye cameras contain rich information regarding sky conditions and cloud movement patterns. However, extracting meaningful patterns from these images and utilizing them effectively for prediction remains a challenging task. Despite these challenges, the study demonstrated promising forecasting performance. The proposed approach involved capturing sky videos or sky images using cameras installed on top of buildings and pairing them with corresponding PV power generation data. Based on this reference, we also adopted a similar approach to this study. However, before implementing the proposed system in a real-world environment, we aim to improve the performance of our custom model beyond the existing benchmark models. Therefore, we utilized the publicly available open-source dataset provided in the previous study for model development and evaluation.

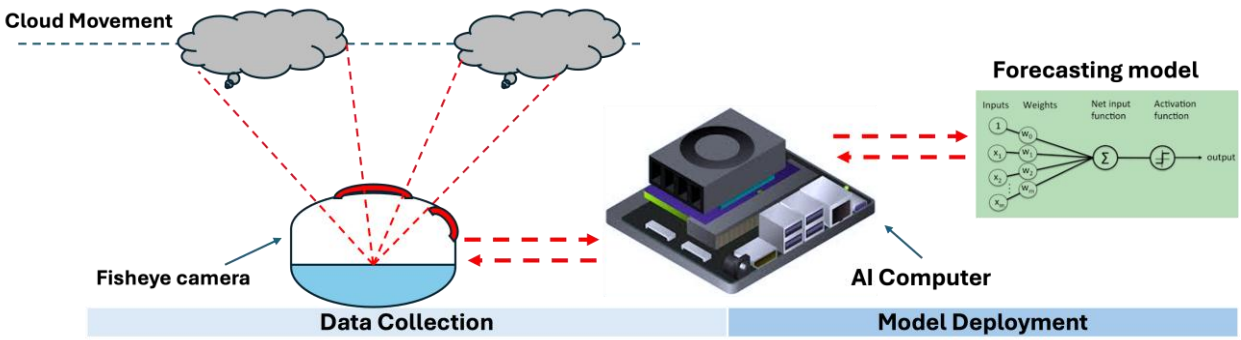

Figure 5 Forecasting process with Fisheye camera

For model development in this study, we employed processed data known as SKIPP'D - a sky images and photovoltaic power generation dataset for short–term solar forecasting. This open-source dataset contains 3 years of processed sky images with dimension 64x64 pixels and concurrent PV power output data with 1 minute interval. All data was stored in Python NumPy array format.
For model deployment in real-world applications, we used sky image cameras and jetson nano AI computers to collect sky image condition and pair with PV power measurement data (see Figure 5).

5.1.1 Planned Deployment Hardware and Operational Pipeline

The real-world deployment of the image-based forecasting system within the FlexBIT project is based on a distributed edge–cloud architecture designed for continuous acquisition, processing, and inference of sky-image data in operational industrial environments.
The deployment infrastructure consists of three primary hardware layers:

- **on-site sky image acquisition devices,**
- **centralized storage and orchestration infrastructure,**
- **edge inference and GPU training environments.**

5.1.1.1 Sky Image Acquisition Layer

The image acquisition subsystem is based on dedicated 360° all-sky optical cameras installed directly at the demonstrator sites near the photovoltaic installations. The currently evaluated hardware platforms include:

- **Ubiquiti UVC-AI-360 (see Figure 6 Ubiquiti UVC-AI-360).**
- **Mobotix Q26 fisheye cameras (see Figure 7 Mobotix Q26).**

Figure 6 Ubiquiti UVC-AI-360

Figure 7 Mobotix Q26

These cameras continuously capture hemispheric sky images at 1-minute intervals, providing high temporal resolution required for ultra-short-term PV nowcasting.
Image streams are retrieved by the Camera Fetcher service operating within the Storage VM environment. The Camera Fetcher communicates with the cameras using secure network protocols:

- **HTTPS (TCP/443),**
- **RTSP (TCP/554).**

The Camera Fetcher is responsible for:

- **periodic image retrieval,**
- **camera communication management,**
- **metadata synchronization,**
- **forwarding images to the preprocessing pipeline,**
- **handling image acquisition reliability and buffering.**

This service constitutes the entry point of the operational image-processing pipeline described in Section 3.6.

5.1.1.2 Central Storage and Integration Layer

Figure 8 GPU RTX A6000

The Storage VM acts as the central integration and orchestration layer between cameras, databases, AI services, and inference nodes. The infrastructure is currently deployed on an RTX A6000 (see Figure 8 GPU RTX A6000) Pro workstation operating inside Electrum's internal network.
The Storage VM hosts:

- **FastAPI services,**
- **MinIO S3-compatible object storage,**
- **PostgreSQL / TimescaleDB,**
- **image preprocessing services,**

- telemetry ingestion components,
- reverse proxy and TLS communication services.

Processed images are stored inside MinIO object storage, while metadata and synchronized telemetry are written into PostgreSQL/TimescaleDB.
The platform integrates image data with:

- SCADA/EMACS measurements,
- PV generation data,
- meteorological signals,
- irradiance measurements,
- operational telemetry from demonstrators.

5.1.1.3 GPU Training Environment

Figure 9 NVIDIA Tesla A100

The AI training environment is based on an NVIDIA Tesla A100 (see Figure 9 NVIDIA Tesla A100) server used for computationally intensive model training and validation tasks.
The training node:

- retrieves historical image datasets from MinIO,
- pulls synchronized time-series measurements from PostgreSQL,
- trains CNN-LSTM forecasting architectures using PyTorch,
- stores trained models back into MinIO as versioned ML artifacts.

This environment supports:

- large-scale experimentation,
- hyperparameter optimization,
- model reproducibility,
- future model lifecycle management using MLflow.

5.1.1.4 Edge Inference Layer – Jetson Orin

Figure 10 NVIDIA Jetson Orin

Operational inference is planned to be executed on NVIDIA Jetson Orin (see Figure 10) embedded AI devices deployed at the edge layer near the demonstrator infrastructure.

The Jetson Orin platform was selected due to:

- **GPU acceleration capabilities,**
- **low-latency inference performance,**
- **low power consumption,**
- **suitability for industrial edge AI deployments.**

The inference node periodically:

- **polls the FastAPI service,**
- **downloads the latest sky image,**
- **retrieves the active model artifact from MinIO,**
- **performs local inference,**
- **sends forecast results back to the Storage VM.**

This architecture enables decentralized execution of forecasting services and minimizes communication latency between data acquisition and operational forecasting.
The generated forecasts are subsequently integrated into:

- **FlexBIT advisory services,**
- **flexibility management algorithms,**
- **storage dispatch optimization,**
- **real-time energy management workflows.**

The planned deployment architecture therefore establishes a scalable and production-oriented operational pipeline connecting:

- **sky image acquisition,**
- **AI training infrastructure,**
- **edge inference devices,**
- **SCADA/EMACS operational telemetry,**
- **and real-time forecasting services within the FlexBIT platform.**

5.2 Data Preprocessing

For model development, data preprocessing is required to prepare clean and ready data which suitable for model's architecture. In this study we used sky images and PV power data from an HDF5 file based on SKIPP'D dataset along with their corresponding timestamps from NumPy file. The dataset in this study is designed into training, validation, and testing sets during the model development process.

Once data is loaded, we create a function to build a table linking each record to its properties: date, month, hour, and PV value. After that we compute two metrics for each record. The first is the Clear Sky Index (KT), which measures how cloudy it is (see equation 10) — calculated by dividing the actual PV power by the theoretical maximum for that month and hour (defined as the 95th percentile of historical output). From the KT perspective, a value of 1.0 indicates clear sky conditions. A KT value of approximately 0.5 corresponds to moderate cloud cover, while values below 0.3 indicate heavy cloud conditions.

$$KT = \frac{actual\ power}{theoretical\ max} \tag{10}$$

The second is instantaneous variance (see equation 11), defined as the squared difference between a record PV value and the previous record value, giving a measure of how rapidly conditions are changing from moment to moment. The low variance indicates stable conditions, in contrast to high variance indicating rapidly changing conditions of cloud.

$$variance = (pv[t] - pv[t-1])^2 \tag{11}$$

These two metrics are then averaged per calendar day to produce daily statistics. Next, we aim for a 15% validation set. At this stage, we select validation days in two equal parts: the first part is chosen to ensure a consistent distribution of KT values (meaning it covers all sky conditions evenly), while the second part is selected for an even distribution of variance values (ensuring both stable and volatile days are represented equally). This stratified selection employs equal-count bins and utilizes a common random number generator. The remaining days are assigned to the training set.

Once the split is established, we analyze the distribution to compare the representation of KT and variance in the training and validation datasets. Subsequently, it compiles all the information into output files, creating distinct NumPy arrays for images, PV values, and timestamps corresponding to each of the training, validation, and test sets. Additionally, a metadata file is generated to document the parameters of the split, relevant statistics, and the specific days included in each validation group.

5.3 Model development

In this research, we conduct extensive architecture exploration for a two-stage system designed for solar power forecasting. The objective is to identify the optimal combination of a Convolutional Neural Network (CNN) and a Long Short-Term Memory network (LSTM) that can forecast solar PV output 1-15 minutes ahead. The process starts by setting a random seed across Python, NumPy, and PyTorch to guarantee that all results can be reproduced, followed by checking for the availability of a GPU and configuring it accordingly. Following that, we load the pre-split training, validation, and test datasets (including images, PV values, and timestamps) that were generated during the earlier data preprocessing phase. For every timestamp, we calculate six features based on time: sine and cosine transformations of the hour and day of the year (to represent cyclical patterns), as well as the sun's elevation and azimuth angles derived from the Astropy astronomy library. All PV values are then standardized using a Robustscaler, which is resistant to outliers.

At this stage, the search space consists of five CNN architectures that vary in size: tiny, small, medium, large, and deep. Each of these architectures is constructed from stacked convolutional layers, incorporating batch normalization, ReLU activations, and optional max-pooling, followed by a global average pooling layer and a compact fully connected layer (see Figure 11 Model development image). Each configuration has been meticulously adjusted for learning rate and dropout level according to its architecture size. On the LSTM side, seven different configurations are established (see table 1), differing in hidden size (ranging from 32 to 256 units), layer count (from 1 to 3), and whether the network operates in a bidirectional manner. Once the CNN is trained, its feature extractor layer is used to compress every image in the full dataset into a compact feature vector. These vectors are saved to disk, so they do not need to be recomputed. In Stage 2, the saved CNN features are combined with the corresponding PV reading at each timestep and fed as a sequence into the LSTM. The LSTM sees a window of 10 consecutive timesteps and is trained to predict the PV value 1-15 steps ahead.

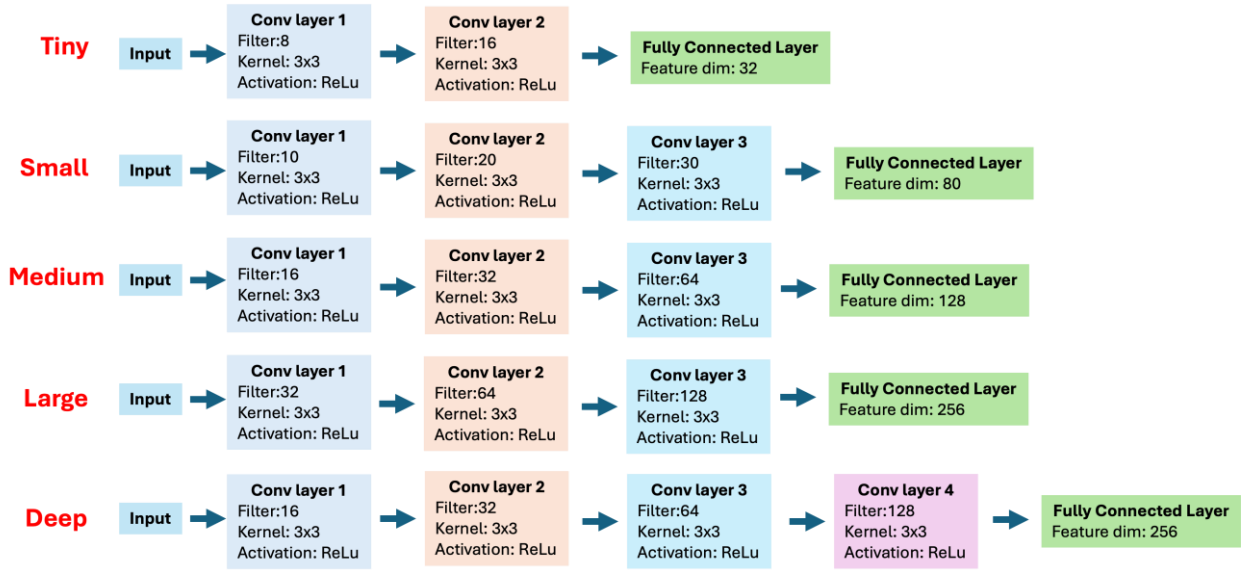

Figure 11 Model development image

After all process, there are 35 combinations (5 CNNs × 7 LSTMs) have been tested, this stage ranks every run by validation R^2. The process concludes a full summary CSV containing, for each run, the CNN and LSTM architecture details, parameter counts, best and final train/validation losses, an overfit ratio (the ratio of final validation loss to final training loss), and the R^2, RMSE, and MAE scores on both the validation and test sets.

Table 1 Overview of Tested CNN–LSTM Architectures

Name	Hidden size	layer	Learning rate	dropout	direction
Tiny_1L	32	1	0.001	0.2	unidirectional
Small_1L	64	1	0.001	0.2	unidirectional
Small_2L	64	2	0.001	0.3	unidirectional
Med_2L	128	2	0.001	0.3	unidirectional
Med_bidir	128	2	0.001	0.3	bidirectional
Large_2L	256	2	0.0008	0.3	unidirectional
Deep_3L	128	3	0.0008	0.35	unidirectional

5.4 Forecasting Assessment

In this study, we compare model performance under two different scenarios for short-term PV power forecasting. In the first scenario, the forecasting model is developed using sky images in the original fisheye format. In the second scenario, we evaluate the model performance using a different sky image representation, namely the unwrapped version of the fisheye images. For this purpose, we used the Defisheye library to correct fisheye distortion and transform the images into a normal perspective view centered on the sky region (see Figure 12 Fisheye format VS normal Perspective).

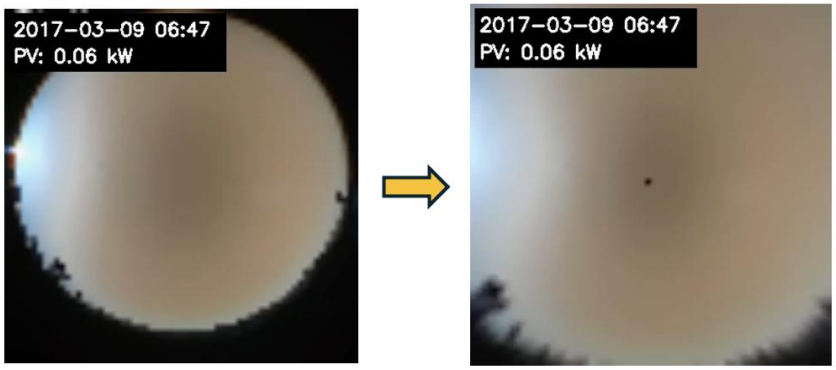

Figure 12 Fisheye format VS normal Perspective

The data preprocessing steps and model development workflow remain identical in both scenarios, with the only difference being in the input image format. From this experiment, two sets of results are obtained. The first section presents the model performance using fisheye sky images, while the second section compares this result. We are comparing forecast horizons of 15 minutes ahead.

This study indicates that the use of unwrapped image perspectives as model input does not significantly improve forecasting performance, while introducing additional preprocessing requirements to convert fisheye images into normal-perspective representations. Therefore, this approach was not pursued further, and the fisheye image format was retained for subsequent model development.

A comparative analysis of models using different image input formats revealed that forecasting performance varied under different sky conditions when fisheye and unwrapped image formats were applied. These differences motivated a deeper investigation to determine the most suitable input format for specific sky conditions.

Two variants of the CNN–LSTM model were trained using the same data-splitting strategy with two image formats: raw fisheye images and unwrapped panoramic views. Both variants were additionally supplemented with sun elevation and azimuth angles derived from geolocation and timestamp information. The results showed that the fisheye and unwrapped variants achieved nearly identical validation performance, indicating that the unwrapping process provided no measurable benefit at the selected image resolution.

6 Validation and Performance Assessment

This section presents the approach formulated for the models deployment and validation activities planned for WP3 activities. This approach constitutes preparing the demonstrator site, ensuring realistic and verifiable conditions for models deployment, establishing validation metrics and other model assessment criteria allowing for grounded and multifaceted assessment of models performance. Therefore, this section provides the groundwork for future gathering demonstrator validation results.

6.1 Validation Setup

The validation procedure is designed to assess forecasting quality under realistic operating conditions and to ensure that model performance is comparable across sites, variables, and forecasting horizons. In line with the training strategy described in Section 4.5, validation is carried out using a chronological split of the available datasets, with the oldest observations used for model development and the most recent observations retained for out-of-sample testing. This setup reflects the intended real-world use of the forecasting tools, where future system states must be predicted exclusively on the basis of past information.
For each demonstrator or pilot dataset, the validation window covers representative operating periods including regular days, rapidly changing weather conditions, and periods with high variability in demand or renewable generation. This makes it possible to verify not only average model quality, but also robustness under dynamic and operationally critical situations. Where data availability permits, the evaluation is repeated across multiple seasonal periods in order to reduce the risk of overestimating performance based on a single favorable test interval.
The adopted validation scheme combines a hold-out test approach with baseline benchmarking. All advanced models are evaluated against the persistence model and linear baselines described in Section 4.3. This allows the project to quantify the added value of more sophisticated methods and to determine whether higher model complexity is justified by measurable gains in predictive accuracy, stability, or operational usefulness.
In addition to global validation across the full dataset, pilot- and site-specific tests are performed to examine how model behavior changes depending on local characteristics such as load composition, PV system size, measurement quality, user behavior, and climatic conditions. This is particularly important in the FlexBIT context, where forecasting tools must remain transferable across residential, tertiary, and industrial settings while still being adaptable to local operating conditions.

6.2 Evaluation Metrics

Forecast quality is assessed using a set of complementary statistical indicators. Since no single metric is sufficient to characterize model behavior in all circumstances, the assessment combines error magnitude metrics, normalized indicators, and goodness-of-fit measures. This multi-metric approach enables both technical comparison between models and interpretation from an operational perspective.
The primary evaluation metrics are:
- Mean Absolute Error (MAE), which quantifies the average absolute deviation between predicted and observed values and is easy to interpret in physical units.
- Root Mean Square Error (RMSE), which penalizes larger deviations more strongly and is therefore useful for identifying models that occasionally produce large forecasting errors.
- Mean Absolute Percentage Error (MAPE), applied where the denominator remains meaningful and sufficiently far from zero, in order to express forecasting accuracy on a relative scale.
- Normalized Root Mean Square Error (nRMSE), used to compare performance across sites and variables with different ranges or nominal capacities.
- Coefficient of determination (R^2), which indicates the proportion of observed variability explained by the forecasting model.
For operational interpretation, metric values are calculated separately for each forecast variable, site, and forecasting horizon. In addition, aggregated results are produced for the complete

evaluation dataset. Where relevant, inference time and model resource requirements are also recorded, because deployment feasibility in edge or distributed environments is an important design criterion for the FlexBIT platform. For PV generation, the primary metrics are RMSE and R^2, consistent with the model ranking used during development in Section 5. MAPE is not applied to PV generation, because the night-time zeros make the relative error undefined. Aggregate accuracy alone does not reveal how a model behaves under variable cloud, so generation forecasts are additionally assessed across weather conditions. The test set is stratified using the clear-sky index (KT) and the instantaneous variance defined in Section 5 (equations 10 and 11), grouping records into clear, partly cloudy, and overcast conditions and into low- and high-variability periods. RMSE and R^2 are then reported within each group. This isolates performance under rapidly changing cloud cover, which is the hardest regime for ultra-short-term solar forecasting.

6.3 Results by Forecast Variable

6.3.1 Load forecasting

For load demand forecasting, the validation focuses on the ability of the models to capture short-term consumption fluctuations, recurring daily patterns, and abrupt changes associated with occupancy, industrial processes, or control actions. Performance is assessed separately for different user segments whenever the underlying data permit such disaggregation. In the FlexBIT use case, particular importance is attached to errors occurring during local peaks and ramp events, because these directly affect scheduling quality, congestion mitigation, and peak-shaving strategies.

6.3.2 RES generation forecasting

For renewable generation forecasting, and especially photovoltaic generation, the assessment examines both average prediction accuracy and the ability to react to fast irradiance changes caused by cloud movement. The two forecasting pathways considered in WP2 - the feature-based time-series models and the image-based solar nowcasting approach - are evaluated against the same target horizons and reference baselines wherever comparable data are available. Special attention is given to the 1-minute to 15-minute horizon, which is the most relevant for the real-time flexibility advisory loop. Results are reported with RMSE and R^2, both as aggregate values and broken down by clear-sky index and variance group, so that accuracy under stable skies can be separated from accuracy under fast-changing cloud.

6.3.3 Price forecasting

Where dynamic tariffs, market-based signals, or externally sourced price indicators are included in a demonstrator, the same validation framework is applied with appropriate adaptations to the data characteristics. In these cases, evaluation emphasizes the consistency of forecasts over operational planning horizons and the usefulness of predicted price patterns for scheduling and cost optimization, rather than only point-wise numerical accuracy.

6.3.4 Other relevant signals

Additional variables, such as state-of-charge-related indicators, flexibility activation signals, or weather-derived auxiliary variables, are validated when they are directly used by optimization or advisory modules. Their assessment follows the same general principles: chronological testing, baseline comparison where relevant, and evaluation of both statistical accuracy and practical value for downstream decision-making.

6.4 Comparative Analysis Approach

The comparative analysis is structured along four dimensions. First, baseline models are compared with advanced forecasting methods in order to determine whether nonlinear or sequential architectures provide a consistent improvement over simple reference approaches. Second, advanced models are compared against one another to identify the most appropriate trade-off between accuracy, interpretability, computational burden, and ease of deployment.
Third, results are analyzed site by site. This allows the consortium to identify whether a given model architecture generalizes well across demonstrators or whether model selection should remain context dependent. Such an analysis is essential because the statistical properties of the input data may differ substantially between residential, tertiary, and industrial environments. Fourth, the comparison is repeated for each forecasting horizon, since the relative ranking of models may change between very short-term nowcasting and longer intraday prediction tasks.
The comparative assessment is not limited to identifying the numerically best model. It also supports the practical model selection process for FlexBIT deployment. A model offering slightly lower accuracy but significantly lower inference time or reduced computational footprint may be preferable for edge execution, while more complex models may be retained for centralized or cloud-supported deployment scenarios.

6.5 Possible limitations for validation

Several limitations must be considered when interpreting the validation results. First, dataset size and quality are not uniform across all pilots. Differences in monitoring granularity, missing data, sensor calibration, and operational disturbances may influence the apparent performance of the evaluated models. Second, transferability between sites cannot be assumed without caution. Even when the same modelling framework is used, local behavior, equipment characteristics, and climate conditions may require site-specific calibration or retraining.
A further limitation concerns cold-start situations. For newly integrated assets, communities, or forecasting targets, insufficient historical data may initially constrain model quality and favor simpler approaches until an adequate learning history is accumulated. In addition, rare events - such as abrupt weather transitions, atypical industrial operation, curtailment, outages, or communication failures - may lead to temporary forecast degradation because they are underrepresented in the training data.
As additional demonstrator data become available and the forecasting services are integrated more tightly with the FlexBIT platform, the models can be re-tuned, re-validated, and further specialized for operational deployment.

7 Use in FlexBIT and Operational Relevance

7.1 Role in WP2

The advanced forecasting tools developed within this deliverable constitute a core component of Work Package 2 (WP2), acting as the predictive intelligence layer of the FlexBIT platform. Their primary role is to provide high-resolution, continuously updated forecasts of photovoltaic (PV) generation and load demand, which are directly consumed by the optimization and control modules defined in D2.1.
Within WP2, the forecasting module enables:

- **real-time situational awareness of energy production and consumption,**
- **anticipation of short-term system states at different time horizons (1-minute to day-ahead),**
- **provision of input data for flexibility identification and clustering algorithms,**
- **continuous update of system conditions in line with the FlexBIT "system-of-systems" concept.**

The forecasting layer is tightly integrated with data acquisition pipelines (IoT devices, SCADA/EMACS systems, weather APIs) and serves as a bridge between raw data streams and decision-making processes. This aligns with the FlexBIT objective of building a robust digital infrastructure for real-time energy management and predictive control

7.2 Support to Flexibility Management

Forecasting tools play a critical role in enabling and enhancing flexibility management within the FlexBIT platform. Their operational relevance stems from the fact that flexibility can only be effectively exploited if future system states are known with sufficient accuracy and temporal resolution.
The developed forecasting framework supports flexibility management in the following ways:

- **Real-time flexibility activation**
  Ultra-short-term forecasts (1–15 minutes ahead) allow the platform to trigger demand response actions, storage dispatch, and EV charging/discharging decisions within the fast control loop.
- **Optimal scheduling of resources**
  Intra-day and day-ahead forecasts enable scheduling of energy storage systems, industrial processes, and building loads, improving alignment with renewable generation.
- **Improved self-consumption and RES integration**
  Accurate PV forecasts allow maximizing local consumption of renewable energy within energy communities, reducing curtailment and grid dependency.
- **Flexibility potential identification**
  Forecasted deviations between expected generation and demand serve as inputs for AI-based algorithms that identify and quantify flexibility potential across heterogeneous assets.
- **Support for market participation**
  Reliable predictions enable participation in flexibility markets (balancing, congestion management), where forecast accuracy directly impacts economic outcomes.
- **Robust decision-making under uncertainty**
  The incorporation of uncertainty-aware forecasting improves resilience of control strategies, reducing risks associated with forecast errors.

These functionalities directly support the FlexBIT vision of aggregating distributed flexibility resources and optimizing their use across residential, tertiary, and industrial sectors

7.3 Expected Impact

The deployment of advanced forecasting tools within the FlexBIT platform is expected to generate significant technical, economic, and environmental impacts:

- **Technical impact**
  - **Increased accuracy of short-term energy predictions,**
  - **Improved stability and responsiveness of control algorithms,**
  - **Enhanced interoperability with SCADA/EMS systems and digital twins,**
  - **Scalability across different demonstrators and energy community types.**
- **Operational impact**
  - **More efficient utilization of distributed energy resources (DER),**
  - **Reduction of imbalance between generation and demand,**
  - **Faster and more reliable control decisions in real-time operation,**
  - **Improved coordination between heterogeneous assets (PV, storage, EVs, loads).**
- **Economic impact**
  - **Reduction of operational costs through optimized energy use,**
  - **Increased revenues from flexibility services and energy trading,**
  - **Better investment planning due to improved predictability.**
- **Environmental impact**
  - **Higher penetration and utilization of renewable energy sources,**
  - **Reduction of CO_2 emissions through improved RES integration,**
  - **Decreased reliance on fossil-fuel-based balancing mechanisms.**
- **System-level impact**
  - **Contribution to grid stability and congestion mitigation,**
  - **Enablement of energy communities and prosumer participation,**
  - **Support for the transition towards a fully flexible, decentralized, and digitalized energy system, as envisioned by FlexBIT .**

Overall, the forecasting module is not a standalone component but a critical enabler of FlexBIT's core functionalities, directly influencing the platform's ability to exploit flexibility, optimize energy flows, and achieve its objectives of efficiency, resilience, and sustainability.

8 Delivered Code Package

The code deliverable accompanying this document is split across two repositories, one per forecasting domain: PV generation forecasting, described below, and electrical load and demand forecasting, documented in the following section.

8.1 PV Generation Forecasting Repository

The repository (stanford-benchmark-solar-forecasting - https://github.com/flexbit-initiative/stanford-bechmark-solar-forecasting) implements the methodology described in the previous sections as a self-contained Python package. It covers the full pipeline from raw data ingestion to model training, evaluation, and diagnostic analysis, and is organised so that any of the modelling approaches it implements — current or future — can be run, retrained, and compared on the same data and the same test set.

The repository targets ultra-short-term PV forecasting at one-minute and fifteen-minute horizons on the Stanford SKIPP'D dataset (rooftop PV co-located with a fish-eye sky camera, three years of one-minute records). The choice of dataset reflects two practical constraints: it is the recognised reference benchmark for sky-image-based PV nowcasting, and its fixed 20-day test set (10 sunny and 10 cloudy days) allows results to be compared directly against the published literature. The same horizons and evaluation protocol will be reused on FlexBIT demonstrator data once sufficient on-site history has been collected.

The codebase is organised in three logical layers: a data layer (raw inputs, a stratified train/validation/test split balanced across clear-sky index and PV variability, and a tabular feature store of 73 engineered variables per timestamp), an experiments layer (one self-contained folder per modelling approach), and a results layer (model checkpoints, per-prediction files, ranked leaderboards, and diagnostic plots).

Currently delivered is a two-stage CNN + LSTM image pipeline that consumes ten-minute sequences of sky images together with sun and time features. Two extensions are planned. The first is a feature-based time-series track operating exclusively on numerical inputs using gradient-boosted trees - XGBoost, LightGBM, Random Forest. The second, conditional on the outcome of the first, is a hybrid configuration combining the two pathways either through a regime-aware switch (selecting the best predictor as a function of measured short-term variability) or through output blending, to be introduced if the image-based and feature-based predictors prove to have complementary strengths.

The package is Poetry-managed, dependency-pinned, and seeded for reproducibility across NumPy, PyTorch, and CUDA. Data files and model weights are excluded from version control. A documented runbook reproduces the full pipeline from raw inputs in roughly half a day on a single A100 GPU.

8.2 Load Demand Forecasting Repository (Time Series)

The repository (https://github.com/flexbit-initiative/flexbit-research-VS) implements a time-series forecasting workflow for FlexBIT load demand data. It is organised as a lightweight research repository rather than a GPU-oriented deep-learning package and is intended for rapid experimentation on local hardware. The current implementation covers the full notebook-based pipeline from raw data ingestion to feature engineering, model training, evaluation, and diagnostic plotting for both one-minute-ahead and fifteen-minute-ahead forecasting tasks.

The repository targets ultra-short-term forecasting of electrical load and PV generation using numerical time-series measurements collected at one-minute resolution. Two forecast horizons are

considered: one minute ahead, using the raw observed one-minute records without interpolation or synthetic augmentation, and fifteen minutes ahead, using real fifteen-minute averages derived from the original one-minute measurements. This distinction was introduced to avoid leakage and artificial smoothing: final reported experiments use only observed data, and no synthetic previous-year data is generated from future values.

The codebase follows the project structure defined in the repository template. The data/ directory stores raw, interim, and processed data, with raw measurements kept separate from modelling outputs. It is kept locally. The repository is organised around the project template structure, but the final deliverables for this work package are contained in the exploration/ directory. This directory includes the cleaned notebook workflows for one-minute and fifteen-minute load forecasting and one-minute and fifteen-minute PV forecasting, together with generated diagnostic figures. The experiments/ directory remains part of the repository template but was not used for the final deliverable, since the modelling work was completed and documented directly in executable notebooks.

The implemented forecasting pipeline is shared across the load and PV notebooks. Each notebook performs chronological train/test splitting, builds lag-based and cyclical timestamp features, trains multiple models on the same test period, and reports a common set of metrics: MAE, RMSE, MAPE, nRMSE, and R2. For one-minute forecasting, the feature set is deliberately compact, using only the two most recent lagged values together with cyclical time encodings. For fifteen-minute forecasting, the tabular feature set additionally includes a small number of short-horizon lags and rolling statistics. The recurrent models use the same input variables as the tabular models but are arranged as time sequences.

The current model suite includes persistence baselines, Linear Regression, Ridge Regression, HistGradientBoosting, CatBoost, LSTM, and GRU models. The gradient-boosting and linear models provide lightweight feature-based baselines suitable for CPU execution, while the recurrent models are included as compact neural time-series references. All experiments were developed and run on a local laptop environment rather than on GPU infrastructure. The repository is Poetry-managed and dependency-pinned, but the exploratory notebooks were also executed using an available local TensorFlow/Jupyter kernel where needed. Report-ready outputs include ranked results tables, static RMSE comparison plots, and interactive Plotly visualisations for inspecting forecasts over the unseen test period.

9 Conclusion

This deliverable presented the advanced forecasting framework developed within the FlexBIT project to support real-time flexibility management, operational optimization, and intelligent energy control across residential, tertiary, and industrial environments. The proposed forecasting layer combines both time-series-based and image-based methodologies to address the operational requirements of ultra-short-term photovoltaic (PV) generation and load demand forecasting.

The work introduced a complete forecasting pipeline covering data acquisition, preprocessing, feature engineering, model training, validation, deployment architecture, and operational integration. The framework incorporates heterogeneous data sources, including SCADA/EMACS measurements, IoT telemetry, weather information, and fisheye sky images, enabling the models to capture both temporal dependencies and rapidly changing environmental conditions.

Several forecasting approaches were investigated, ranging from lightweight baseline methods such as persistence and linear regression to advanced machine learning architectures including LSTM, GRU, CatBoost, Histogram Gradient Boosting, and hybrid CNN–LSTM models. The conducted experiments demonstrated that advanced deep learning and gradient boosting methods provide improved capability for modelling nonlinear dynamics and short-term variability in PV generation data while remaining compatible with real-time operational constraints.

A dedicated image-based forecasting infrastructure was also developed, including distributed edge–cloud deployment architecture based on FastAPI services, MinIO object storage, PostgreSQL/TimescaleDB, NVIDIA Tesla A100 training infrastructure, and NVIDIA Jetson Orin edge inference devices. This architecture enables scalable deployment, decentralized inference, low-latency forecasting, and future extensibility toward federated and distributed AI workflows.

The performed assessment of fisheye and unwrapped image representations indicated that fisheye images provide comparable forecasting performance while avoiding additional preprocessing complexity. Consequently, the original fisheye representation was retained for further operational deployment and future development activities.

Overall, the developed forecasting framework establishes a robust foundation for predictive energy management within the FlexBIT platform. The proposed solutions support real-time advisory services, flexibility optimization, storage dispatch, and intelligent control strategies while remaining scalable and adaptable to heterogeneous demonstrator environments. Future work will focus on large-scale operational validation, integration with optimization modules, uncertainty-aware forecasting strategies, and further enhancement of edge AI deployment capabilities within decentralized energy systems.