IBEX Framework

Technical Documentation

Jan Biały

2026

Technical documentation for the IBEX Framework.

CHAPTER

ONE

API REFERENCE

1.1 TensorCreator

class IBEX_Module.TensorCreator.**TensorCreator**(*config: str*)

Bases: object

Class responsible for constructing structured PyTorch tensors from raw IBEX measurement data files.

The class traverses the IBEX raw data directory tree, loads event-level data for individual energy channels, applies basic preprocessing steps (including optional hexadecimal value translation), and combines the data into channel-specific tensors saved in `.pt` format.

Tensor creation can be performed either for continuous orbital ranges or for datasets split around the perigee change of the IBEX spacecraft.

init_channel_tensors(*perigee_change: bool = False*) → None

Initialize multithreaded tensor creation for all energy channels.

The method spawns a separate worker thread for each energy channel and generates channel-specific tensors according to the configured instruction file and directory boundaries.

Parameters

perigee_change (*bool, optional*) – If True, tensor creation is performed across subfolders spanning the perigee change of the IBEX spacecraft. If False, data are processed within a continuous directory range.

Raises

ValueError – If an unsupported instruction file is specified.

Return type

None

1.2 TensorAnalyzer

class IBEX_Module.TensorAnalyzer.**ChannelAnalyzer**(*config: str = 'MainConfig.yml'*)

Bases: object

Class responsible for preprocessing IBEX ENA data based on high-quality acquisition time intervals specified in instruction files (HiCullGoodTimes.txt and LoGoodTimes.txt).

The class provides three main data preprocessing modes:

1. Extraction of raw ENA events satisfying quality criteria and saving them as PyTorch tensors.
2. Calculation of good data sums (total number of ENA detections per valid time interval).
3. Aggregation of ENA detection features by computing physically meaningful statistical descriptors (means, circular means, and standard deviations).

The preprocessing is performed independently for each IBEX-Hi and IBEX-Lo energy channel.

Notes

Some methods of this class are marked as deprecated and preserved only as legacy artefacts from earlier versions of the analysis pipeline. They are not used in the current implementation but remain available for reference and reproducibility of previous results.

---

analyze(*method: str*, *file_path_prefix: str*, *file_path_suffix: str*) → None

Deprecated since version 1.1: Deprecated end-to-end analysis routine from an earlier version of the ChannelAnalyzer class. This method is retained for archival and reference purposes only and is not used in the current data processing pipeline.

Perform correlation analysis between all energy channels and generate correlation matrices and heatmaps.

Parameters

- **method** (*str*) – Correlation method identifier (`'pearson'`, `'weighted_pearson'`, `'spearman'`, or `'MI'`).
- **file_path_prefix** (*str*) – Prefix of input file paths.
- **file_path_suffix** (*str*) – Suffix of input file paths.

Return type
None

init_analyzer_tensors(*hi_name: str = 'hi_hex_channel'*, *lo_name: str = 'lo_hex_channel'*, *option: str = 'save_good_data_sums'*) → None

Initialize multithreaded preprocessing of all IBEX energy channels.

Depending on the selected option, the method performs one of the supported preprocessing procedures independently for each channel using a thread pool.

Parameters

- **hi_name** (*str, optional*) – Filename prefix for IBEX-Hi channel tensor files.
- **lo_name** (*str, optional*) – Filename prefix for IBEX-Lo channel tensor files.
- **option** (*str, optional*) – Type of preprocessing to perform. Supported values are:
  - `"save_raw_good_data"` — extract and save raw filtered ENA events.
  - `"save_good_data_sums"` — compute and save ENA detection sums.
  - `"aggregate_all_physical_features"` — aggregate ENA features for global analysis.

Raises
Exception – If an unsupported option value is provided.

load_data(*file_path: str*) → ndarray

Deprecated since version 1.1: Deprecated I/O helper method retained from a previous version of the class. File loading is no longer handled by this component.

Load numerical data from a text file.

Parameters
file_path (*str*) – Path to the input file.

Returns
Loaded data array, or an empty array in case of an error.

Return type
np.ndarray

mutual_information(*x: ndarray*, *y: ndarray*) → float

Deprecated since version 1.1: Deprecated legacy method retained for reference purposes. Mutual information analysis is no longer part of the current processing workflow.

Estimate mutual information between two one-dimensional arrays.

Parameters

- **x** (*np.ndarray*) – First input array.
- **y** (*np.ndarray*) – Second input array.

Returns
Estimated mutual information value.

Return type
float

pearson_correlation(*x: ndarray*, *y: ndarray*) → float

Deprecated since version 1.1: This method is deprecated and preserved only as a legacy implementation from a previous version of the ChannelAnalyzer class. It is no longer used in the current analysis pipeline.

Compute the Pearson correlation coefficient between two one-dimensional arrays.

Parameters

- **x** (*np.ndarray*) – First input array.
- **y** (*np.ndarray*) – Second input array.

Returns
Pearson correlation coefficient.

Return type
float

spearman_correlation(*x*, *y*) → float

Deprecated since version 1.1: Legacy method retained for backward compatibility with older analysis scripts. Not used in the current version of the pipeline.

Compute the Spearman rank correlation coefficient.

Parameters

- **x** (*np.ndarray*) – First input array.
- **y** (*np.ndarray*) – Second input array.

Returns
Spearman rank correlation coefficient.

Return type
float

weighted_pearson_correlation(*x: ndarray*, *y: ndarray*, *time_durations_x: ndarray*, *time_durations_y: ndarray*) → float

Deprecated since version 1.1: Deprecated weighted correlation method retained as a historical artefact from an earlier analysis approach. It is not used in the current workflow.

Compute a weighted Pearson correlation coefficient using time durations as weights.

Parameters

- **x** (*np.ndarray*) – First input array.
- **y** (*np.ndarray*) – Second input array.
- **time_durations_x** (*np.ndarray*) – Weights associated with the first array.
- **time_durations_y** (*np.ndarray*) – Weights associated with the second array.

Returns
Weighted Pearson correlation coefficient.

Return type
float

1.3 FileMerger

class IBEX_Module.FileMerger.**FileMerger**(*config: str*)

Bases: object

Class responsible for merging preprocessed IBEX data files into unified, channel-specific output files.

The class operates on data batches generated in earlier stages of the analysis pipeline and combines them into single merged datasets. This approach enables processing large data volumes while limiting system resource usage, in particular RAM memory consumption.

---

merge_files(*channel_num: int = 15*) → dict | None

Merge preprocessed IBEX data files into channel-specific merged datasets.

For each energy channel, the method loads partial data files defined in the configuration, concatenates them vertically, and saves the merged result to the output directory.

Parameters
channel_num (*int, optional*) – Upper bound of channel indices to process. Channels are iterated from 1 to `channel_num - 1`. Default is 15 (corresponding to 14 channels).

Returns
Dictionary containing `return_code` and `error_message` in case of an error. Returns `None` if all channels are merged successfully.

Return type
dict or None

write_log(*return_code: int*, *error_message: str*) → None

Log an error message and return code to the execution terminal.

Parameters

- **return_code** (*int*) – Numeric code describing the error type.
- **error_message** (*str*) – Description of the encountered error.

Return type
None

write_message(*message: str*) → None

Log a status or informational message to the execution terminal.

Parameters
message (*str*) – Informational message to be logged.

Return type
None

1.4 IBEX_NN

class IBEX_Module.IBEX_NN.**RateAutoencoder**(*input_dim: int = 14*, *latent_dim: int = 4*)

Bases: `Module`

Autoencoder model for ENA detection rate vectors.

The model compresses 14-dimensional per-channel ENA rate vectors into a low-dimensional latent representation and reconstructs the original rates from it.

---

forward(*x: Tensor*) → tuple[Tensor, Tensor]

Forward pass through the autoencoder.

Parameters
x (*torch.Tensor*) – Input tensor of shape `(batch_size, input_dim)`.

Returns

- **x_hat** (*torch.Tensor*) – Reconstructed input tensor.
- **z** (*torch.Tensor*) – Latent representation.

IBEX_Module.IBEX_NN.**aggregate_events_to_bins**(*event_data: dict[int, Tensor]*, *time_bin: float = 600.0*) → tuple[ndarray, ndarray]

Aggregate raw ENA events into fixed-duration time bins.

For each time interval, the function computes:

- ENA detection rates for all 14 energy channels,
- auxiliary physical and geometrical conditions derived from a reference channel.

Time bins with missing data in any channel are discarded.

Parameters

- **event_data** (*dict[int, torch.Tensor]*) – Dictionary mapping channel indices to event-level tensors.
- **time_bin** (*float, optional*) – Width of the time bin in seconds. Default is 600 s.

Returns

- **rates** (*np.ndarray*) – Array of shape `(N_bins, 14)` containing ENA detection rates for all energy channels.
- **conditions** (*np.ndarray*) – Array of shape `(N_bins, N_conditions)` containing aggregated physical and geometrical features per time bin.

IBEX_Module.IBEX_NN.**init_dataset_preparation_pipeline**(*base_dir: str*, *out_dir: str*, *regimes: list[str] = ['BeforePerigeeChange', 'AfterPerigeeChange']*) → None

Execute the full dataset preparation pipeline for multiple mission regimes.

For each regime, the function loads raw event data, aggregates it into time bins, and saves the resulting datasets to disk.

Parameters

- **base_dir** (*str*) – Base directory containing regime-specific input data.
- **out_dir** (*str*) – Output directory for aggregated datasets.
- **regimes** (*list of str, optional*) – List of regime identifiers to process.

Return type
None

IBEX_Module.IBEX_NN.**load_autoencoder_data**(*path: str | Path*) → tuple[ndarray, StandardScaler]

Load and preprocess aggregated rate data for autoencoder training.

The preprocessing includes logarithmic transformation and per-channel standardization.

Parameters
path (*str or pathlib.Path*) – Path to the directory containing `rates.npy`.

Returns

- **X** (*np.ndarray*) – Preprocessed rate matrix.

- **scaler** (*StandardScaler*) – Fitted scaler used for standardization.

IBEX_Module.IBEX_NN.**load_event_data**(*base_dir: str | Path*) → dict[int, Tensor]

Load raw IBEX event-level data for all energy channels.

The function loads preprocessed PyTorch tensors corresponding to individual IBEX-Hi (channels 1–6) and IBEX-Lo (channels 7–14) energy channels.

Parameters

base_dir (*str or pathlib.Path*) – Base directory containing `Hi_data` and `Lo_data` subdirectories.

Returns

Dictionary mapping global channel indices (1–14) to event tensors of shape `(N_events, K_features)`.

Return type

dict[int, torch.Tensor]

Raises

FileNotFoundError – If any required channel file is missing.

IBEX_Module.IBEX_NN.**make_dataloaders**(*X: ndarray*, *batch_size: int = 256*, *val_frac: float = 0.2*) → tuple[DataLoader, DataLoader]

Create training and validation DataLoaders from a NumPy array.

Parameters

- **X** (*np.ndarray*) – Input data matrix.
- **batch_size** (*int, optional*) – Batch size.
- **val_frac** (*float, optional*) – Fraction of data reserved for validation.

Returns

- **train_loader** (*DataLoader*) – DataLoader for training.
- **val_loader** (*DataLoader*) – DataLoader for validation.

IBEX_Module.IBEX_NN.**per_channel_mse**(*model: Module*, *X: ndarray*, *device: str = 'cpu'*) → ndarray

Compute per-channel reconstruction mean squared error.

Parameters

- **model** (*torch.nn.Module*) – Trained autoencoder model.
- **X** (*np.ndarray*) – Input data matrix.
- **device** (*str, optional*) – Computation device.

Returns

Array of per-channel MSE values.

Return type

np.ndarray

IBEX_Module.IBEX_NN.**plot_per_channel_mse**(*mse: ndarray*, *title: str*) → None

Plot per-channel reconstruction error.

Parameters

- **mse** (*np.ndarray*) – Per-channel MSE values.
- **title** (*str*) – Plot title.

Return type
None

IBEX_Module.IBEX_NN.**plot_training_history**(*history: dict[str, list[float]]*, *title: str*) → None

Plot training and validation loss curves.

Parameters

- **history** (*dict*) – Dictionary containing loss histories.
- **title** (*str*) – Plot title.

Return type
None

IBEX_Module.IBEX_NN.**reconstruction_mse**(*model: Module*, *X: ndarray*, *device: str = 'cpu'*) → float

Compute global reconstruction mean squared error.

Parameters

- **model** (*torch.nn.Module*) – Trained autoencoder model.
- **X** (*np.ndarray*) – Input data matrix.
- **device** (*str, optional*) – Computation device.

Returns
Global reconstruction MSE.

Return type
float

IBEX_Module.IBEX_NN.**save_aggregated_dataset**(*output_dir: str | Path*, *rates: ndarray*, *conditions: ndarray*, *time_bin: float*) → None

Save aggregated rate and condition datasets to disk.

The function stores NumPy arrays along with a JSON metadata file describing the aggregation parameters.

Parameters

- **output_dir** (*str or pathlib.Path*) – Output directory for the aggregated dataset.
- **rates** (*np.ndarray*) – Array containing per-channel ENA detection rates.
- **conditions** (*np.ndarray*) – Array containing aggregated physical and geometrical features.
- **time_bin** (*float*) – Time bin width used during aggregation (in seconds).

Return type
None

IBEX_Module.IBEX_NN.**train_autoencoder**(*model: Module*, *train_loader: DataLoader*, *val_loader: DataLoader | None = None*, *n_epochs: int = 200*, *lr: float = 0.001*, *device: str = 'cpu'*, *verbose: bool = True*) → dict[str, list[float]]

Train an autoencoder model using mean squared error loss.

Parameters

- **model** (*torch.nn.Module*) – Autoencoder model to be trained.
- **train_loader** (*DataLoader*) – DataLoader providing training samples.
- **val_loader** (*DataLoader, optional*) – DataLoader providing validation samples.
- **n_epochs** (*int, optional*) – Number of training epochs.
- **lr** (*float, optional*) – Learning rate.
- **device** (*str, optional*) – Device identifier (`'cpu'` or `'cuda'`).
- **verbose** (*bool, optional*) – If True, training progress is printed.

Returns

Dictionary containing training and validation loss histories.

Return type

dict