aiirmap

A Framework for Machine Learning Enhanced Design

User's Manual

Robert F. H. Hunter

August 2025, v1

Contents

1 Overview

aiirmap is available as a github repo: https://github.com/rfhhunter/aiirmap

aiirmap is a codebase which interfaces simulation softwares with Python databasing, machine learning, and optimization capabilities. It was originally designed to couple optoelectronic device simulations in Synopsys Sentaurus TCAD [1] with principle component analysis [2, 3] dimensionality reduction. It is written to be easily extendable to new simulation softwares and machine learning methods. To ease adoption, aiirmap is built upon standard Python packages; such as pandas [4, 5], scikit-learn [6], tensorflow [7], scipy [8], numpy [9], and matplotlib [10].

The basic functionality of aiirmap is as a databasing tool. Large sets of experiments can be easily defined using pandas DataFrame based *DataBases*. They are sent to and collected from the simulation softwares, manipulated (split, merged, sorted, thresholded, cleaned, etc.), subjected to machine learning, and analyzed with user-defined runfiles. This highly versatile run paradigm facilitates comprehensive design of experiments and problem space exploration. It also empowers machine learning approaches which hinge upon large datasets and flexible, reliable data pipeline control. *DataBases* and machine learning instances are saved for later use with csv and pickle format, respectively.

The remainder of this document is organized as follows: Section 2 provides an operational overview including install and setup (Section 2.1), an outline of the project structure (Section 2.2), the basic aiirmap objects (Section 2.3), and an outline of the runfile paradigm (Section 2.4). Section 3 describes how aiirmap interacts with simulation softwares such as Synopsys Sentaurus TCAD (Section 3.1). Section 4 discusses implementation of machine learning algorithms in the aiirmap code. Section 5 gives instructions on how to include new simulation softwares and machine learning functionalities. Section 6 provides some anecdotal notes on running aiirmap. Finally, in Section 7 is a list of publications which employ aiirmap and, in Section 8, persons and entities who contributed to the development of aiirmap are gratefully acknowledged.

Issues with the code or documentation can be reported in the github repository.

2 Running aiirmap

This section overviews how to install and setup aiirmap (Section 2.1). It then outlines the project structure (Section 2.2) and custom objects (Section 2.3) to give context to the run paradigm (Section 2.4), which is built around user-defined "runfiles".

2.1 Install and Setup

"Installation" of aiirmap is straightforward. Retrieve the aiirmap project from the github repo (https://github.com/rfhhunter/aiirmap) and clone it to your machine. aiirmap uses Python 3. The specific version is restricted only by the requirements of the packages that it uses. It has been verified on Python 3.9, 3.11, 3.12, and 3.13. The remainder of the aiirmap installation is to install any Python packages it includes, if you do not have them already. The majority of these are listed at the top of the *aiirmapCommon.py* file. No specific versioning of the included Python packages is required. aiirmap operation is verified with up-to-date packages at the time of the writing of this document (Aug. 2025).

Setup of aiirmap is likewise very easy. All aiirmap requires is a folder with three subfolders to save output files. This folder is referred to as the filing folder. It can be located anywhere which is read-write accessible while running the codebase. In aiirmap its location is defined in *config.py* using the variable *filingdir*. Filing folder subfolders are used to organize the output files. Defined immediately below *filingdir* in *config.py* they include a folder for databases and machine learning instance pickles (*dbdir*), copies of simulation files (*sfdir*), and plots (*pldir*). No other setup is required to run aiirmap. The remaining variables in *config.py* set the defaults for other aspects of the codebase, as described below.

2.2 Project Structure

This section gives a very brief overview of the project structure, primarily to give context to the runfiles paradigm. For more information on the functions and code architecture please refer to the extensive comments in the code itself.

The main folder of aiirmap contains the files with configuration variables (*config.py*), an import file (*aiirmapCommon.py*), bash terminal wrappers for a few of the functions (*pyDBCollector.py, pyProjectCollector.py, grid-runner.py*), and the files for creating Sentaurus command files (*py<2/3>prepper.py*).

Several subfolders exist. The *utilities* folder contains the heart of aiirmap functionality. Functions are the split topically into different files with relevant names. The *runfiles* folder is the location where users can drop their control scripts. A couple example runfiles are included in the aiirmap repository. The *user_manual* folder contains the latex files for this document. And the *archive* offers a location to drop old files for safekeeping.

2.3 Project Objects

There are two main aiirmap objects; *DataBases* and *aiirMappings*. These objects are substantiated as Python classes located in the files *utilities/databasing.py* and *utilities/ml/machine_learning.py*, respectively. This section briefly overviews their intended operation and attributes. For further details please refer to comments in the py files.

2.3.1 DataBases

aiirmap *DataBases* are how sets of simulations are realized in the code. They are defined in the file *utilities/databasing.py* and this is also where all the functions for their creation and manipulation are located. The aiirmap *DataBase* is based upon the pandas DataFrame allowing large datasets to be easily and quickly defined and manipulated (cleaned, split, combined, sorted, thresholded, etc.). aiirmap *DataBases* are saved as csv files in the database filing folder for later use. Their attributes are as follows, provided in the order they appear in the csv file:

- **grid** [boolean]:
  Whether or not the *DataBase* includes only simulation inputs. This is determined by whether or not the *experiment-outputs* column exists in the *DataBase.dataframe* (see below for more information). *DataBases* with only reference and/or simulation input columns are referred to as "Grids". The first line in a *DataBase* csv is the comment "#AiirMap DataBase". The next line will contain the text "GRID FILE" if the *DataBase* is a Grid or will be empty if not.

- **dbfilename** [string ending in .csv]:
  The identifier for the dataset; its filename. One line in the csv file.

- **dbFile** [path string]:
  The path to the dataset. Note that this does *not* need to match its actual location on your machine, when a *DataBase* is loaded the accurate *dbFile* will be used in the code (and will be recorded if the *DataBase* is saved). One line in the csv file.

- **lineage** [list with format of [[datetime string, action description str], ...]]:
  A recording of past manipulations of the *DataBase*. This is saved in the csv file using multiple lines; one line for each entry in the list with a header line reading "Lineage:". Lineage inheritance is also implemented. For *DataBases* created through merging of other *DataBases* a few lines of the lineage of these ancestors is recorded in the new database csv file and distinguished by »» and «« (the size of the inherited lineage can be set in *config.py*).

- **dataframe** [pandas DataFrame]:
  The main attribute for the *DataBase*. This is where your dataset data goes. Simulation inputs and outputs are given by the columns and are typically indexed with text (for instance with the parameter and variable names used in a Synopsys Sentaurus TCAD model). Different experiments (simulation instances) are given by the rows and are indexed with integers. The *DataBase.dataframe* has two special columns which act to separate the dataframe into three sections; reference, inputs, and outputs. These two columns are defined as *experiment-inputs* and *experiment-outputs* (this default can be changed or added to by editing *<in/out>putStartStrs* in *config.py*). They are placed at the left-side of each section resulting in a column structure which looks like; *db_idx*|reference columns|*experiment-inputs*|input parameter columns|*experiment-outputs*|output variable columns (where *db_idx* is the dataframe index). Note that *db_idx* begins at 0, while *experiment-<in/out>puts* begin at 1 (are the experiment number rather than its index).

2.3.2 aiirMappings

aiirMappings are the objects which codify instances of machine learning application when data is transformed. They include the inputs and outputs to the machine learning algorithm as well as the mapping between them. Information on which algorithm has been employed and how (its hyperparameters) are also recorded here. The *aiirMapping* class is defined in the file *utilities/ml/machine_learning.py*. *aiirMappings* are saved using the pickle (.pkl) binary file format. The attributes of the *aiirMapping* object, in the order given in the code, are as follows:

- **mapType** [(a specific) string]:
  The label for the machine learning algorithm which was used. The labels for the implemented algorithm are *pca* and *ae*. *pca* is the label for principle component analysis, a linear dimensionality reduction technique [2, 3]. *ae* is the label for nonlinear dimensionality reduction using autoencoder architecture neural nets [11].
- **mapName** [string]:
  Descriptor for the *aiirMapping* instance. As opposed to *DataBase.dbfilename* this string should not end in .pkl.
- **mapFolder** [string path]:
  The path to the folder containing the *aiirMapping* pickle file. Note that this does *not* need to match the actual location of the pickle file in the code, when the *aiirMapping* is loaded then the accurate location is used in the code and will be written to file if the *aiirMapping* is saved.
- **settings** [dictionary]:
  These are the hyperparameters for the given machine learning algorithm. Each algorithm has different hyperparameter options and thus different *settings* dictionary structure. For the *aiirMapping.settings* structures specific to each algorithm see the comments at the top of the *aiirMapping* class definition.
- **ml** [machine learning object]:
  Also located in *utilities/ml/machine_learning.py* is the definition of classes for each type of machine learning algorithm (so far only *DimensionalityReduction*). These objects package the actual machine learning objects created by the scikit-learn or tensorflow (or other package) functions for later access.
- **filteredInputDB** [DataBase]:
  This is the aiirmap *DataBase* with the input data for the machine learning algorithm. Filtering refers to the fact that the experiments in this dataset have been cleaned and thresholded, it contains only the data which is actually being used as input to the algorithm. This *DataBase* contains all columns present in the original *DataBase* for this filtered data.
- **filteredInputGrid** [(Grid) DataBase]:
  Very similar to the previous attribute but now the reference and outputs columns have been removed. Input columns which are extraneous to the machine learning training have also been removed in a process called column cleaning (see the function *dbDRCleanCols* in *utilities/databasing.py* for more information). This is the training data (only).
- **results** [dict]:
  The outputs from the machine learning algorithm. Each algorithm will have different outputs according to type and intended operation. For the *aiirMapping.results* structures specific to each algorithm see the comments in the *aiirMapping* class definition.

2.4 Runfiles

aiirmap provides the functions and objects for a versatile simulation flow control. It is up to the user how to they want to use this functionality and, indeed, the exploratory nature of design and machine learning enhanced design means that this use will evolve. "Runfiles" are simply the user scripts written to access aiirmap functionality. Users may also directly use the Python terminal if they prefer. All aiirmap capabilities are imported by including *aiirmapCommon.py* at the start of your script (*from aiirmapCommon import **).

Typical runfile operations are listed below. These are just examples offered to spark the user's imagination. Any desired functionality can be included in runfiles and organized as the user sees fit. Runfile functions may also be loaded as modules into other runfiles. General useful functionality can be ported into the aiirmap utilities and shared if the user wishes.

Potential runfile operations: Creating a new *DataBase* and filling it with data to send as input to the simulation software. Creating a new *DataBase* from one or more other *DataBases*. Loading a *DataBase* and running analysis or machine learning and potentially outputting a new *DataBase* for further simulation. Comparing performance across *DataBases*. Designing specific investigations. Creating sequential simulation run operations. etc.

3 Simulation Software Interactions

This section outlines how aiirmap interacts with the simulation softwares which have been implemented in its environment. As mentioned, aiirmap has been designed in a manner which allows it to be extended to include other simulation softwares. The procedure for adding new softwares into the code is outlined in Section 5 below. Users may add subsections here outlining their software interface.

3.1 Synopsys Sentaurus TCAD Interaction

Synopsys Sentaurus TCAD is an finite element analysis optoelectronics device simulator [1]. aiirmap was originally written to interface Python with Sentaurus to provide databasing and machine learning capabilities. Sentaurus interacts with Sentaurus in two ways; collection of simulation results and (batch-capable) run control.

Collection of results is instantiated in the Sentaurus project flow using the Sentaurus Python tool. aiirmap includes a script (*py2prepper.py*) which writes the command file for this tool during preprocessing to allow maximum flexibility in the strict Sentaurus operation flow. One creates the Python tool in their project and then includes the following line in their project tooldb file: '*set WB_tool(gsub,prologue) { exec python $wdir/path/to/py2prepper.py $wdir }*' (do not include the quotation marks). Due to the order of operations in Sentaurus, the collector tool first writes the input parameter values and the nodes associated with each experiment into a csv file in the Sentaurus folder, this occurs during preprocessing. The rest of the operations occur when the project is finished or when requested by the aiirmap user (using *pyProjectCollector.py*, *pyDBCollector.py*, or other scripts which run the functions for which those files are bash wrappers). At that point, the output variable values are collected from the Sentaurus gvars.dat file, matched with the input parameters using the node numbers in the collector csv file, and together saved to an aiirmap *DataBase* (see *utilities/sReadAndInteract.py* for more information). aiirmap also writes a second csv in the Sentaurus project folder which records the execution times of the nodes. Pertinent Sentaurus files, as defined in *config.py*, are then copied to a project subfolder in the simulation files filing folder.

It can be noted that the aiirmap Sentaurus collection protocol allows for use of the Sentaurus gopt optimization tool which creates child projects for the optimization of each design. These are tracked and linked to their parent algorithmically in the collection code.

The second functionality, Sentaurus run control, was written for two reasons; 1) the ease at which one can define experiments, including large numbers of experiments, in a Python environment, as compared to the Sentaurus GUI, and 2) to enable large dataset runs using a batching functionality which was required to overcome memory issues inherent to the version of Sentaurus which was being used. The Sentaurus batch run control is instantiated through protocols which directly write the Sentaurus project's gtree.dat file which codifies the state of the Sentaurus workbench. The *grid-runner.py* wrapper file runs an automatic procedure which takes an input database, splits it into easy to handle subset databases, which it runs sequentially, collecting and collating the results as it goes.

Further details on Sentaurus interaction are found in the comments in the code (specifically, see *py2prepper.py* and the wrappers in the main aiirmap folder).

4 Machine Learning Algorithm Implementations

This section outlines the different machine learning algorithms which have been implemented in the aiirmap framework. Users may add subsections here for added functionality. Comments on adding new machine learning functionality are provided in Section 5.

Implementation of machine learning follows two methods depending upon the class of machine learning algorithm. First, for machine learning algorithms which act upon a dataset adding information or providing analysis but not necessarily transforming the data, a simple file with functions may be used. A number of **clustering** algorithms have been implemented this way. These include DBSCAN [12], HDBSCAN [13], KMeans [14], and Agglomerative Hierarchical [15]. This functionality is located in *utilities/ml/clustering.py*. Functions apply clustering to a *DataBase*, compute centroids and simple statistics, and auto-split *DataBases* according to cluster. Please see the code for more information. The second machine learning implementation is used for algorithms which transform the data. The included example being principle component analysis [2, 3] and autoencoder-based [11] **dimensionality reduction**. These implementations use the *aiirMapping* object described in Section 2.3. A brief outline of aiirmap dimensionality reduction is below.

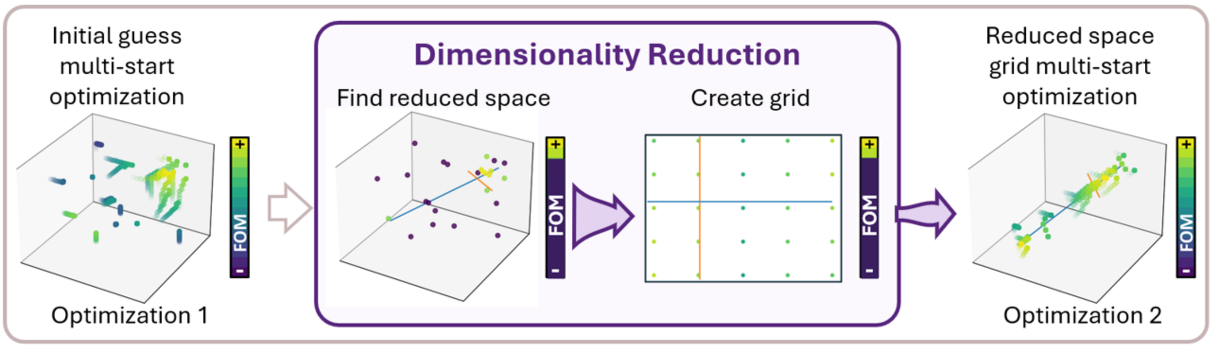

Figure 1: aiirmap dimensionality reduction enhanced design schema. FOM indicates a chosen figure of merit. Figure sourced from [16].

4.1 Dimensionality Reduction

aiirmap dimensionality reduction was designed to execute the schema shown in Figure 1 [16]. Top designs from a first stage of multi-start optimization are used to train the dimensionality reduction algorithm which identifies a latent high-performance design subspace of reduced dimensionality. A grid of points are generated to survey this subspace and are projected back out into the full design space and used as start points for a second stage of optimization. More information on this method and its advantages can be found in [16].

The dimensionality reduction implementation is located in *utilities/ml/machine_learning.py*. This functionality, including the *aiirMapping* object, reflect the schema shown in Figure 1. Functions take a *DataBase* clean and threshold it, apply dimensionality reduction, and then use the reduced space to generate a grid of designs which is projected back into the original design space. This is completed via the *applyDR* function. The exploratory nature of machine learning application led to the further development of so-called "DR investigations". These are sets of multiple dimensionality reduction (DR) runs with differing parameters which are applied to the same input data and saved together in the form of a pickle and pandas dataframe csv. See the code, specifically *investigateDR*, for more info. A wrapper which automatically detects whether a single or multiple/investigation dimensionality reduction is being run is found in *runNewDR*. Some functions for analyzing the reduced space and the reconstruction of data (when datapoints are projected into and then back out of the reduced space) are also provided.

5 Adding New Simulation Software or Machine Learning Algorithms

The design philosophy of aiirmap is for it to be a general purpose framework which allows any simulation softwares which can communicate with Python to be coupled to all of the powerful databasing, optimization, machine learning, analysis, visualization, control, etc. capabilities written in Python packages. New simulation softwares can be added, as can new machine learning algorithms. This section will briefly outline the procedures for each.

Adding new simulation softwares into aiirmap will depend highly upon the context. Specifically, how the new software communicates with Python. The main functionality of the new interface is to be able to collect a correctly formatted *DataBase* csv. It is suggested that the user familiarize themselves with the *DataBase* format as described in Section 2.3 above and in *utilities/databasing.py*. Other functionality may be added, such as run control and/or storage of simulation files but is not required (although the author does suggest run control to allow two-way communication between aiirmap and your software). Interaction with Sentaurus provides all three of these functionalities and may be used as an example, see Section 3.1 and *utilities/sReadAndInteract.py* for more information. As the Sentaurus TCAD section outlines, the interaction between Sentaurus and aiirmap is a bit convoluted due to the operational paradigm of Sentaurus, new software implementations may contain many less functions than was required in *sReadAndInteract.py*. Sharing the added software functionality in the github repository is greatly appreciated! Please write a new utility file similar to *sReadAndinteract.py* with the new interface and feel free to add a section to this document outlining its use (locate it in Section 3).

Adding new machine learning algorithms into aiirmap can be done two ways. If the new algorithm use is light, exploratory, or time-constrained the user may find it sufficient to implement the new algorithm only within their personal runfiles. On the other hand, if the new algorithm use is substantial, requires standardization, or if the user has the intent to share implementation of the new algorithm involves the creation of new *aiirMapping* variants or addition of similar objects, as the user sees fit. The description of *aiirMapping* objects in Section 2.3 above and the architecture and comments in *utilities/ml/machine_learning.py* are pertinent in this case (take care to note how two different dimensionality reduction methods are included). The hope is that a large majority of machine learning algorithms will be able to fit into the *aiirMapping* form factor which would act as a standardized frontispiece for machine learning functionalities which transform the data. Sharing of added functionality in the github repository is greatly appreciated! Please include a brief overview of how to run the new algorithm as a subsection in Section 4.

6 Notes

Please be aware of the following points:

- The plotting subfolder (*pldir*) is not used by a great number of the plotting functions. Suggested practice for these functions is to screenshot or save your plots directly from the matplotlib figure window. This course of action allows the user to adjust the aspect ratio and make sure all figure components are visible for each image.
- The *amverbose* variable (found in *config.py*) is meant as a global switch to turn off or on extra terminal output. It is only partially implemented.

7 Publications

The following peer-reviewed publications have utilized the aiirmap framework, listed from newest to oldest:

- R.F.H. Hunter, "Next-Generation Multi-Junction Photovoltaic Design Paradigms and Adaptive Optics Techniques for Telecommunications Applications and the Global Energy Transition," Ph.D. Thesis, University of Ottawa, Aug. 2025.
- R.F.H. Hunter *et al.*, "Machine learning enhanced design optimization and knowledge discovery for multi-junction photonic power converters," *Scientific Reports*, Aug. 2025. doi: https://doi.org/10.1038/s41598-025-16408-4
- M. de Lafontaine *et al.*, "Figures of merit to quantify carrier collection in betavoltaics: Gain and gain efficiency," *Cell Reports Physical Science*, 102789, Aug. 2025. doi: https://doi.org/10.1016/j.xcrp.2025.102789
- G.P. Forcade *et al.*, "Multi-junction laser power converters exceeding 50% efficiency in the short wavelength infrared", *Cell Reports Physical Science*, 6(6):102610, May 2025. doi: https://doi.org/10.1016/j.xcrp.2025.102610

The following scientific conference presentations (some with proceedings) have utilized the aiirmap framework:

- M. de Lafontaine *et al.*, "Optimizing Tritium Powered GaAs p-i-n and p-n Betavoltaic Cells," *14th International Conference on Tritium Science and Technology*, Ottawa, Canada, 2025.
- P. Wilson *et al.*, "Impact of Luminescent Coupling on Multijunction InGaAs Photonic Power Converters under Current Mismatched Conditions in the C-Band," *Photonics North 2025*, Ottawa, Canada, 2025.
- P. Wilson *et al.*, "Quantifying the luminescent coupling process in C-band multi-junction photonic power converters," *Photonics West*, San Fransisco, USA, 2024.
- R.F.H. Hunter, "Using machine learning to optimize multi-junction photonic power converters," *SPIE OPTO 2024*, San Francisco, California, United States, 2024.
- K. Hinzer *et al.*, "Multi-junction photonic power converters: AI enhanced design optimization," *52nd IEEE Photovoltaic Specialists Conference (PVSC)*, Seattle, USA, 2024.
- K. Hinzer *et al.*, "C-band Multi-Junction Photonic Power Converters: AI Techniques for Optimized Designs and Role of Luminescent Coupling," *6th Optical Wireless and Fiber Power Transmission Conference (OWPT)*, Japan, 2024.
- Y. Grinberg *et al.*, "Dimensionality Reduction in Photonics Design – New Methods and Applications," *Photonics North 2024*, Vancouver, Canada, 2024.
- M. de Lafontaine *et al.*, "p-i-n Betavoltaic Cells under 63Ni Irradiation: Quantifying Carrier Collection and Power Output," *52nd IEEE Photovoltaic Specialists Conference (PVSC)*, Seattle, USA, 2024.
- G.P. Forcade *et al.*, "High-Performance Multi-Junction C-Band Photonic Power Converters: Calibrated Optoelectronic Model for Next Generation Designs," *50th IEEE Photovoltaic Specialists Conference (PVSC)*, San Juan, Puerto Rico, 2023.

Users may feel free to add their publications and presentations to the lists. These lists are not considered exhaustive.

8 Acknowledgments

aiirmap was originally developed as part of the AIIR-Power project [17, 18]. The author is grateful to the agencies which provided funding to the AIIR-Power project and to the members of the consortium for their guidance. While its design diverged significantly over time, aiirmap got its initial inspiration from the project spacemap, developed by AIIR-Power collaborators at the National Research Council of Canada. spacemap is available at https://github.com/NRC-AI4Photonics/spacemap. This work was completed as part of the author's doctoral studies, the author is grateful to the SUNLAB research group, to the University of Ottawa, to the Government of Ontario, and to the NSERC CREATE program for funding and support. The aiirmap logo was created with ChatGPT.

References

[1] Synopsys Inc. *Synopsys Sentaurus TCAD*. 2025. URL: https://www.synopsys.com/manufacturing/tcad.html.

[2] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (Nov. 1901), pp. 559–572. DOI: 10.1080/14786440109462720. (Visited on 04/21/2024).

[3] Ian T. Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 13, 2016), p. 20150202. DOI: 10.1098/rsta.2015.0202. (Visited on 04/21/2024).

[4] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010. DOI: 10.25080/Majora-92bf1922-00a.

[5] The pandas development team. *pandas-dev/pandas: Pandas, version 2.2.2*. Version latest. Apr. 2024. DOI: 10.5281/zenodo.3509134.

[6] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. DOI: 10.5555/1953048.2078195.

[7] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[8] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[9] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[10] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[11] Muhammad Al-Digeil et al. “PCA-enhanced autoencoders for nonlinear dimensionality reduction in low data regimes”. In: *Proceedings of the 36th Canadian Conference on Artificial Intelligence*. The 36th Canadian Conference on Artificial Intelligence. Montreal, Quebec: Canadian Artificial Intelligence Association, 2023. DOI: 10.21428/594757db.05a13011. (Visited on 06/05/2023).

[12] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[13] Leland McInnes, John Healy, and Steve Astels. “hdbscan: Hierarchical density based clustering”. In: *Journal of Open Source Software* 2.11 (2017), p. 205. DOI: 10.21105/joss.00205.

[14] Stuart P. Lloyd. “Least squares quantization in PCM”. In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137.

[15] J. H. Ward. “Hierarchical Grouping to Optimize an Objective Function”. In: *Journal of the American Statistical Association* 58.301 (1963), pp. 236–244. DOI: 10.1080/01621459.1963.10500845.

[16] Robert F. H. Hunter et al. “Machine learning enhanced design and knowledge discovery for multi-junction photonic power converters”. In: *Scientific Reports* (Aug 2025). DOI: https://doi.org/10.1038/s41598-025-16408-4.

[17] National Research Council of Canada. *Supporting collaborative projects between Canada and Germany in artificial intelligence and value-added manufacturing*. 2021. URL: https://nrc.canada.ca/en/stories/supporting-collaborative-projects-between-canada-germany-artificial-intelligence-value-added.

[18] Fraunhofer Institute for Solar Energy Research. *AI-assisted design and fabrication of photonic infrared power converters for energy and telecommunication*. 2021. URL: https://www.ise.fraunhofer.de/en/research-projects/aiir-power.html.