Travail de fin d'études et stage[BR]- Travail de fin d'études : Integrating short-term dispatch constraints in a system dynamics energy planning model[BR]- Stage d'insertion professionnelle

Auteur : Vidal Montesinos, Carla
Promoteur(s) : Quoilin, Sylvain
Faculté : Faculté des Sciences appliquées
Diplôme : Cours supplémentaires destinés aux étudiants d'échange (Erasmus, ...)
Année académique : 2021-2022
URI/URL : http://hdl.handle.net/2268.2/16566

UNIVERSITY OF LIÈGE

FACULTY OF APPLIED SCIENCES

MASTER'S THESIS

Integrating short-term dispatch constraints in a system dynamics energy planning model

Author:

Carla Vidal Montesinos

Supervisor:

Sylvain Quoilin

Master's thesis carried out to obtain
the degree of Master of Science in Engineering

Academic year 2021-2022

KEYWORDS

Abbreviation	Description
BATs	Batteries
BEVs	Batteries Electric Vehicles
CF	Capacity Factor
CHP	Combined heat and power
ENS	Energy not served
HDAM	Conventional hydro dam
HPHS	Pumped hydro storage
LL	Lost of Load
LP	Linear programming
MILP	Mixed integer linear programming
MTS	Mid-term scheduling
P2H	Power to heat units
RES	Renewable energy sources
VOLL	Value of Lost Load
VRES	Variable renewable energy sources

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Sylvain Quoilin, for his support and guidance during the development of this thesis. I would also like to express my gratitude to my family, who have encouraged and supported me from the beginning.

ABSTRACT

Electricity generation from renewable energy sources (RESs) is becoming increasingly important for achieving EU emissions reduction targets. However, integrating high-level RESs into the existing grid is a challenging task due to their fluctuating nature. Thus, power systems need to become increasingly flexible in order to deal with unpredictable renewable energy generation.

The aim of this work is to contribute to the development of relevant models for the simulation of the European power system and its short-term prospects. The study focuses on the simulation of high renewable energy shares along with various flexibility options (generation, transmission expansion and storage). The simulations are then used to train machine learning models based on Artificial Neural Networks (ANNs). Results indicate that ANNs can predict the main power system constraints and outcomes with good accuracy.

The generated models can be integrated into a more general system-dynamics model, thus improving the representation of the power system operation and constraints.

CONTENTS

1. INTRODUCTION

One of the most researched challenges of the energy system focuses on the interactions between centralized and decentralized power generation, as well as between different energy sectors. As a result of this research, new concepts have emerged such as "Smart Energy Systems" or "Integrated Energy Systems".

Smart Energy Systems are able to integrate electricity, heating, cooling, gas and transport sectors. This requires coordination between many sectors, such as electricity grids, district heating and cooling grids, gas grids and different fuel infrastructures. However, modelling these aspects in an integrated way is challenging. The high temporal and technical level of detail required by energy system models may not be compatible with long-term energy planning involving all relevant energy sectors [1]. Current solutions involve the use of hybrid models which, based on the level of linking between models, are classified as soft-linking and hard-linking.

A soft or hard linking means joining two or more codes in a coherent way to capture different details present in their frameworks, in order to increase the strength of the assessment. From a connectivity point of view, soft and hard linking differ in the existence of feedback communication (soft) or the absence of it (hard) [2].

Soft-linking runs both models iteratively, keeping their complexities fairly intact. However, its computational efficiency is low, and its convergence is not guaranteed.

Hard-linking often implies a simplified description of one of the two models. The models are integrated and solved in a simultaneous optimization. Its disadvantages include low computational tractability, and non-applicability to models with divergent formulations.
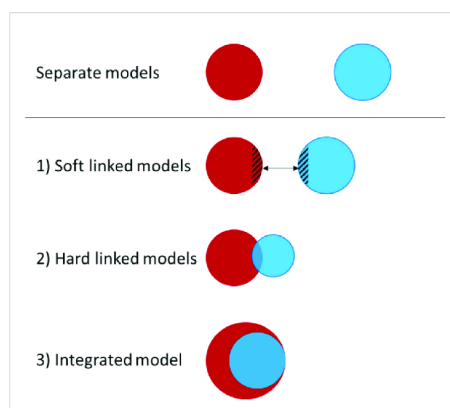
Fig. 1.1. Model linking types [3]

The current thesis formulates an alternative approach in order to couple a power system model (the EU Dispa-SET power system model) and a system dynamics model (the MEDEAS model). This method relies on the creation of a surrogate model that approx-

imates the power system optimization results. In [4], surrogate models, also known as metamodels or emulators, are defined as simplified approximations of more complex, higher order models.

1.1. Flexibility assessment: State of The Art

The integration of higher shares of variable renewable energy (VRE), such as wind and solar energy, in power systems is essential to decarbonise the power sector while still meeting the energy demand. However, the forecast uncertainty and seasonal variability of wind and solar energy are the main challenges for power systems operators and regulators [5]. Hence, a need to increase the flexibility of the system arises in order to effectively manage the high penetration of VREs.

Power system flexibility can be defined as the power system's ability to respond to both expected and unexpected changes in demand and supply [6]. Several flexibility mechanisms exist to address RES intermittency issues, such as dispatchable generation, grid flexibility, energy storage, and demand integration:

- Conventional dispatchable generation is needed to meet the electricity demand not covered by RES. This generation can also cope with a highly variable net load profile when it is dynamically flexible [7].

- Grid flexibility describes the existence of a robust transmission network that balances supply and demand across wider balancing zones, as well as cross-border interconnections that facilitate flexibility exchange between regions. It also refers to the existence of advanced controls for improving communication among system components [6].

- Energy storage systems are essentially used to modify the timing of power supply by storing electricity at low rates and discharging it at higher rates. Due to their ability to adjust their output and switch between charging and discharging modes, storage systems can handle fluctuations and unpredictability well. While pumped hydro storage continues to dominate electricity storage, future options can include compressed air energy storage, batteries, or conversion to other energy carriers (e.g. power-to-gas) [7].

- In terms of demand, it can either change in absolute terms (by increasing or decreasing given a certain elasticity) or shift over time to better match electricity supply. Examples include the operation of deferrable loads (e.g. washing machines), devices related to heating, ventilation, and air-conditioning, and the charging of electric vehicles in the future.

Insufficient flexibility may result in load shedding during periods of low VRE generation and renewable energy curtailment during periods of high VRE generation [8]. These

are often used as indicators in VRE integration analyses in order to assess a system's flexibility.

1.2. Objectives and Methodology

The goal of this master's thesis is to integrate flexibility constraints into the low time-resolution model MEDEAS. In order to do so, a surrogate model is created to predict the adequacy and flexibility of the large-scale power system Dispa-SET and then implemented into the least-detailed model MEDEAS. The flexibility mechanisms considered in this work are: dispatchable generation, grid extension and electricity storage. The EU power system case study is considered with a focus on a high share of renewables.

The surrogate model is a simple analytical model that mimics the input and output behaviour of the Dispa-SET system. For its development, computationally expensive simulations must be performed on a carefully selected set of samples [9]. First, a multi-dimensional inputs space is created by varying the following key system characteristics: flexible capacity, non-flexible capacity, storage, grid infrastructure and renewable penetration. A Latin hypercube sampling is then defined to run the Dispa-SET model over this inputs space.

For the surrogate model construction, machine learning techniques based on artificial neural network (ANN) algorithms are developed to predict the key system performance indicators, in this case curtailment and unserved energy, as a function of the system features. The generated surrogate models can approximate the behaviour of the underlying complex simulations with reasonable precision while being computationally cheaper to evaluate [9]. They can be incorporated into the MEDEAS model, thus enhancing the current flexibility integration.
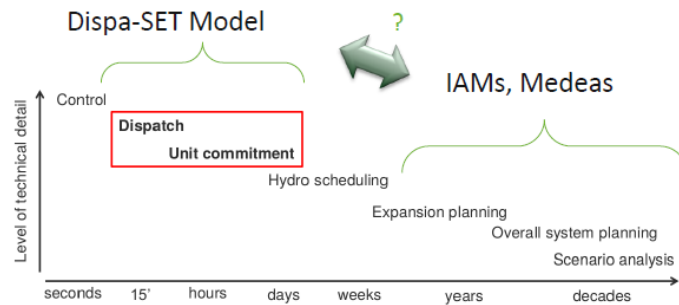
Fig. 1.2. Dispa-SET and MEDEAS model integration [1]

1.3. Contributions

This thesis has been developed using the Dispa-SET EU model [10]. Below is clarified the personal contribution to the work:

- Improvement of the Dispa-SET model by adding the function *adjust_ntc* to the repository.

- Design of experiment by running multiple variations of the model.

- Analysis and execution of the simulation from the HPC NIC5 from CÉCI [1].

- Development of a machine learning algorithm based on neural networks to train surrogate models.

Most of the scripts and data used and generated are available at the following repository: https://github.com/carlavidalm/Master_Thesis.

1.4. Thesis Outline

The present thesis is organized as follows:

- **Chapter 2**: Description of the Dispa-SET model, as well as its tools, methods and techniques.

- **Chapter 3**: Creation of the dataset for training the surrogate model using Dispa-SET.

- **Chapter 4**: Creation of a surrogate model with ANN that approximates the results of the power system optimization Dispa-SET.

- **Chapter 5**: Overview description of the MEDEAS model, how flexibility is currently integrated and how it could be improved with the ANN regression model.

- **Chapter 6**: Conclusions and future work.

---

[1] https://www.ceci-hpc.be/

2. DISPA-SET MODEL

2.1. Introduction

Dispa-SET is an open-source unit commitment and optimal dispatch model focused on the balancing and flexibility problems in European grids with a high share of VRES. The main purpose of using the Dispa-SET is to analyse large interconnected power systems with a high level of detail. Other purposes applied to this work are the analysis of VRES impacts on the power system, based on a high temporal resolution representation with several flexibility options, as well as the consideration of technical constraints in the power system.

Figure 2.1 illustrates the data flows and links within the modelling framework, which has five main components: inputs, preprocessing, simulation, and outputs.
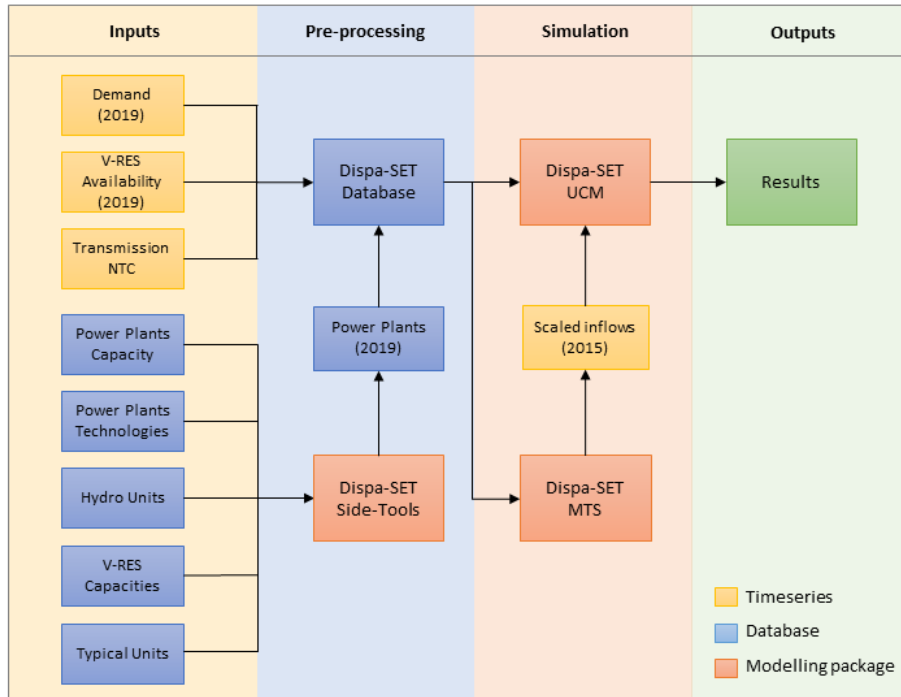
Fig. 2.1. Relational block diagram between elements in the simulation

The model is formulated as a Mixed Integer Linear Programming (MILP) or Linear Programming (LP) problem. The pre and post-processing tools are written in Python and the optimization tool is written in GAMS [11]. Input data is handled using .csv files.

The following sections are inspired by the Dispa-SET documentation [10].

2.2. Objective function

The goal of the Dispa-SET model is to minimize the total power, heating and transportation system operational costs in order to distribute the total power demand among the available generation units. Therefore, the MILP objective function is the overall generation cost over the optimization period and can be formulated as follows:

$Min_{TotalSystemCost} =$

$$
\sum_{u,i}(CostStartUp_{i,u} + CostShutDown_{i,u}) +
$$

$$
\sum_{u,i}(CostRampUp_{i,u} + CostRampDown_{i,u}) +
$$

$$
\sum_{u,i} CostFixed_u \cdot Comitted_{i,u} \cdot TimeStep +
$$

$$
\sum_{u,i} CostVariable_{i,u} \cdot Power_{i,u} \cdot TimeStep +
$$

$$
\sum_{hu,i} CostVariable_{i,u} \cdot Heat_{i,u} \cdot TimeStep +
$$

$$
\sum_{l,i} PriceTransmission_{i,l} \cdot Flow_{i,l} \cdot TimeStep +
$$

$$
\sum_{n,i} CostLoadShedding_{i,n} \cdot ShedLoad_{i,n} \cdot TimeStep +
$$

$$
\sum_{n\_th,i} CostHeatSlack_{n\_th,i} \cdot HeatSlack_{n\_th,i} \cdot TimeStep +
$$

$$
\sum_{n\_h2,i} CostH2Slack_{n\_h2,i} \cdot H2Slack_{n\_h2,i} \cdot TimeStep +
$$

$$
\sum_{chp,i} CostVariable_{chp,i} \cdot CHPPowerLossFactor_{chp,i} \cdot Heat_{chp,i} \cdot TimeStep +
$$

$$
\sum_{i,n} VOLL_{Power}(LL_{MaxPower,i,n} + LL_{MinPower,i,n}) \cdot TimeStep +
$$

$$
\sum_{i,n} 0.8 \cdot VOLL_{reserve}(LL_{2U,i,n} + LL_{2D,i,n} + LL_{3D,i,n}) \cdot TimeStep +
$$

$$
\sum_{u,i} 0.7 \cdot VOLL_{Ramp}(LL_{RampUP,u,i} + LL_{RampDown,u,i}) \cdot TimeStep +
$$

$$
\sum_{s,i} CostOfSpillage \cdot Spillage_{s,i}
$$

(2.1)

TotalSystemCost, expressed in EUR, is defined as the sum of different cost items:

- *Fixed costs*: whether the unit is on or off.

- *Variable costs*: based on the power output of the units.

- *Start-up and shut-down costs*.

- *Ramp-up and ramp-down costs.*

- *Shed load costs*: due to necessary load shedding.

- *Transmission costs*: determined by the flow in the transmission lines.

- *Loss of load costs*: power exceeding the demand or not matching it, ramping and reserve.

- *Spillage costs*: due to spillage in storage.

2.3. Demand balance

The main constraint to be met is the power supply–demand balance, for each period and each zone, in the day-ahead market. Thus, the sum of the power generated by all the units present in the node (including the power generated by the storage units) and the power injected from neighbouring nodes is equal to the load in that node, plus the power consumed for heat generation through P2H units and power consumed for energy storage, minus the shed load.

$$
\sum_u (Power_{u,i} \cdot Location_{u,n}) + \sum_l (Flow_{u,i} \cdot LineNode_{l,n})
$$
$$
= Demand_{DA,n,i} + Demand_{Flex,n,i} + \sum_s (StorageInput_{s,i} \cdot Location_{s,n}) +
$$
$$
\sum_{p2h} (PowerConsumption_{p2h,i} \cdot Location_{p2h,i}) - ShedLoad_{n,i} - LL_{MaxPower_{n,i}} + LL_{MinPower_{n,i}}
$$
$$
(2.2)
$$

2.4. Rolling Horizon

The mathematical problem could theoretically be solved for a whole year divided into one-hour time steps. However, when attempting to solve the model with realistically sized data sets, it would likely to become extremely computationally demanding. The problem is therefore divided into smaller optimization problems that are run recursively throughout the year.

An example of this is shown in Fig 2.2, in which both the optimization horizon and the look-ahead (or overlap) period are one day. The initial values of the day j optimisation are the final values of the previous day's optimisation. A look-ahead period is modelled and then discarded in order to avoid issues that may occur at the end of the optimization, such as emptying the hydro reservoirs or starting low-cost but non-flexible power plants. In this example, the optimization is performed for 48 hours, but only the first 24 hours are conserved.
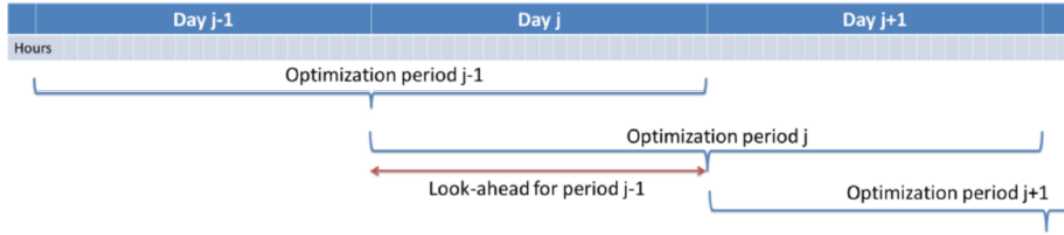
Fig. 2.2. Principle of the rolling horizon optimisation [10]

The optimisation horizon and overlap values can be set in the Dispa-SET configuration file. In this thesis, the Horizon length is set to 4 days (96 hours) and the overlap to 1 day (24 hours).

2.5. Mid-Term Scheduling (MTS)

Because emptying the storage has a zero marginal cost, a non-constrained optimization tends to leave the storage completely empty at the end of the optimisation horizon. For that reason, a minimum storage level must be imposed at the last hour of each optimisation horizon, which is typically a few days. Thus, the minimum storage level at the last hour is an exogenous input. It can be obtained from a long-term scheduling optimization, which is called Midterm Hydro-Thermal Scheduling (MTS).

MTS module represents a simplified version of the linear programming formulation used to pre-allocate reservoir levels of large storage units which are then used as guidance curves. It is especially relevant in systems with high shares of HDAM and HPHS.

The formulation used in this thesis is called Regional-MTS, in which MTS is run for the selected zones simultaneously. MTS is achieved by relaxing the integer variables and removing the following constraints, transforming the MILP problem into a LP formulation:

- Parameters and variables linked to the thermal sector (e.g. CHP units, thermal storage and heating demands)

- Parameters and variables linked to power plant cycling (e.g. Start-up and shut-down time, ramping rates, minimum up and down time, etc)

- Costs associated to the above-mentioned constraints

2.6. Dispa-SET EU model

The present thesis is based the Dispa-SET EU model, a case study of the EU. The following sections describe the EU model inputs used in the simulations.

2.6.1. Zones

All EU countries except Cyprus, Malta, and Luxembourg are covered in this thesis together with Norway, Switzerland, and the United Kingdom. Table 2.1 presents an overview of countries and their country codes (ISO 3166).

Code	Country	Code	Country
AT	Austria	IE	Ireland
BE	Belgium	IT	Italy
BG	Bulgaria	LT	Lithuania
CH	Switzerland	LV	Latvia
CZ	Czech Republic	NL	Netherlands
DE	Germany	NO	Norway
DK	Denmark	PL	Poland
EE	Estonia	PT	Portugal
EL	Greece	RO	Romania
ES	Spain	SE	Sweden
FI	Finland	SI	Slovenia
FR	France	SK	Slovakia
HR	Croatia	UK	United Kingdom
HU	Hungary		

Table 2.1. Overview of countries and ISO Alpha 2 country codes

2.6.2. Technologies

The technologies considered in this study are listed in Table 2.2. Three columns are used to classify technologies. The VRES column indicates the variable renewable technologies, the Storage column indicates the technologies that can store energy and the Flexibility column indicates whether a technology is flexible, semi-flexible, or inflexible.

Variable renewable energies are considered inflexible because they are non-dispatchable and depend on the intermittency of their resources.

On the other hand, reservoir-based hydroelectric facilities are considered semi-flexible, as they have room for manoeuvre thanks to their reservoir storage. Steam turbines are also classified as semi-flexible since they depend on the fuel used, for example, natural gas would be more flexible than nuclear energy.

Note that only technologies related to electricity are considered. Heating and CHP units are outside the scope of this thesis.

Technology	Description	VRES	Storage	Flexibility
COMC	Combined cycle	N	N	F
GTUR	Gas turbine	N	N	F
ICEN	Internal combustion engine	N	N	F
STUR	Steam turbine	N	N	S
HDAM	Conventional hydro dam	N	Y	S
HROR	Hydro run-of-river	Y	N	I
HPHS	Pumped hydro storage	N	Y	S
WTOF	Offshore wind turbine	Y	N	I
WTON	Onshore wind turbine	Y	N	I
PHOT	Solar photovoltaic	Y	N	I
BATS	Stationary batteries	N	Y	F

Table 2.2. Overview of technologies

2.6.3. Fuel types and prices

Dispa-SET only considers a limited number of fuel types, which are summarized in Table 2.3.

Fuel	Description
BIO	Biofuels
GAS	Gas
HRD	Coal
LIG	Lignite
NUC	Nuclear energy
OIL	Petroleum
PEA	Peat Moss
GEO	Geothermal steam
SUN	Solar energy
WAT	Hydro energy
WIN	Wind energy
WST	Energy from waste
OTH	Other fuels and energy carriers

Table 2.3. Overview of fuel types

Different fuels can be used to power a given technology, e.g. steam turbines can be powered by almost any type of fuel. In Dispa-SET, each power plant unit must be defined with the pair of values (Technology, Fuel).

Table 2.4 shows a summary of the fuel prices considered.

	Price
Nuclear	3
Black coal	10
Gas	80
Fuel-Oil	65
Biomass	10.08
Lignite	7.23
Peat	9.36

Table 2.4. Overview of fuel prices[EUR/MWh]

2.6.4. Power Plants database

In order to improve computational efficiency, some original units are clustered into larger ones during pre-processing. As a result, the number of continuous and binary variables is reduced and the simulation accuracy can, in some cases, be performed without significant loss [10].

To generate the database of clustered power plants units, the *get_capacities.py* script of the DispaSET SideTools is run. The power plants database has the following fields for all units:

Description	Field Name	Units
Unit Name	Unit	-
Power Capacity (for the clustered unit)	PowerCapacity	MW
N° of original units clustered	Nunits	-
Zone	Zone	-
Technology	Technology	-
Fuel	Fuel	-
Efficiency	Efficiency	%
Effciency at minimum load	MinEfficiency	%
Minimum up time	MinUpTime	h
Minimum down time	MinDownTime	h
Ramp up rate	RampUpRate	%/min
Ramp down rate	RampDownRate	%/min
Start up cost per unit	StartUpCost_pu	EUR
No load cost per unit	NoLoadCost_pu	EUR/h
Ramping cost	RampingCost	EUR/MWh
% of minimum nominal capacity	PartLoadMin	%
Start up time	StartUpTime	h
CO_2 intensity	CO2Intensity	t_{CO_2}/MWh

Table 2.5. Common fields for clustered units

For BATS units, ramp up and down rates are set to 1 and all other parameters, except for efficiency, are set to 0.

For storage units, there are some parameters that need to be defined. For other equipment, these can be left blank.

Description	Field Name	Units
Storage capacity	STOCapacity	MWh
Self-discharge rate	STOSelfDischarge	%/d
Maximum charging power	STOMaxChargingPower	MW
Charging efficiency	STOChargingEfficiency	%

Table 2.6. Specific fields for clustered storage units

The discharge efficiency of a storage unit should be assigned to the common field Efficiency. Likewise, the common field PowerCapacity is the nominal power in discharge mode.

For BATS individual units, the number of hours that can discharge at its power capacity before exhausting its energy capacity (Storage Capacity/Power Capacity) is fixed to 4.

In this thesis, the LP formulation for the optimization problem is used (i.e. without the binary variables). In this case, the Ramp up and down rates are updated as follows:

$$RampUpRate = PartLoadMin \cdot \frac{1}{max\,(1; MinDownTime)} +$$
$$\left(1 - PartLoadMin \cdot \frac{1}{min\left(\frac{1}{60}; RampUpRate\right)} \right)$$

$$RampDownRate = PartLoadMin \cdot \frac{1}{max\,(1; MinUpTime)} +$$
$$\left(1 - PartLoadMin \cdot \frac{1}{min\left(\frac{1}{60}; RampDownRate\right)} \right)$$

2.6.5. Electricity Demand

Electricity demand time series of year 2019 is given per zone. It is assumed to be inelastic to the price signal.

2.6.6. Renewable generation

The availability factor (AF) for RES is defined as the percentage of the nominal power that can be generated each hour. It is provided as an hourly non-dimensional time series.

This variable renewable generation is either fed to the grid or curtailed. The availability factor for non-renewable technologies is 1.

2.6.7. Net Transfer Capacities (NTC)

The capacities of electricity transmission between countries are given as hourly time series and assumed constant for the whole year. They are based on the maximum of reported historical capacities of 2019.

2.6.8. Input Prices

	Price
Price of CO2	25
Price of Unserved Heat	84.21
Load Shedding Cost	1000
Price of Transmission	0
Price of Unserved H2	75
Curtailment Cost	20

Table 2.7. Overview of other prices [EUR/MWh]

3. MULTIDIMENSIONAL INPUT SPACE FOR THE SIMULATIONS

3.1. Introduction

The purpose of this chapter is to create the dataset for training the surrogate model. A selection of the sample points within the design space is performed using a technique known as design of experiments (DoE). At these selected points, a computationally expensive simulation is run, and the responses are recorded. In Chapter 4, these input/output data are used to approximate the behaviour of the complex simulations by using a surrogate model [9].

3.2. Case study

As mentioned before, the EU power system is considered in Dispa-SET in order to create and validate the surrogate model. The period for each simulation is one year, specifically, 2019.

To generate the dataset, power plant units are classified into five types: flexible units, slow units, storage units, PV units and wind units.

The classification of conventional units into flexible and non-flexible units is based on certain conditions. IRENA [6] defines flexible units as "units that can ramp up and down quickly, have a low minimum operating level and fast start-up and shutdown times."

Units	Fuel	Condition
$Flex_{units}$	GAS, HRD, OIL, BIO, LIG, PEA, NUC, GEO	$PartLoadMin < 0.5$ and $TimeUpMin < 5$ and $RampUpRate > 0.01$
$Slow_{units}$		$PartLoadMin \geq 0.5$ or $TimeUpMin \geq 5$ or $RampUpRate \leq 0.01$

Table 3.1. Flexible and slow units classification

In order to assess renewable penetration, onshore wind turbines and photovoltaic units are considered among renewable sources, since wind and solar energies currently have the highest potential [12]. The European Commission [13] states that, by 2030, the share of wind and solar energy in electricity production capacity should double from 33% today to 67%. By then, solar energy would also be the largest source of electricity in the EU, while wind would account for 31% of installed capacity.

Regarding storage units, as the potential for more hydro units is rather limited in the EU, only BATS units are considered.

Units	Fuel	Technology
$Storage_{units}$	OTH	BATS
PV_{units}	SUN	PHOT
$Wind_{units}$	WIN	WTON

Table 3.2. Storage, PV and wind units

In addition, three parameters of interest are calculated: the annual average Availability Factor (AF) of onshore turbines and solar panels and the peak load.

	Value	Units
AF_{wton}	0.2604	%
AF_{pv}	0.1313	%
$PeakLoad$	440929.4125	MW

Table 3.3. Parameters

3.3. Design parameters

It is first necessary to define the input parameters for the simulations, which include six unitless variables:

Capacity ratio [%]:

$$Capacity ratio = \frac{PowerCap_{flexunits} + PowerCap_{slowunits} + PowerCap_{storageunits}}{PeakLoad} \quad (3.1)$$

Share flexibility [%]:

$$Share_{flex} = \frac{PowerCapacity_{flexunits}}{PowerCapacity_{flexunits} + PowerCapacity_{slowunits}} \quad (3.2)$$

Share storage [%]:

$$Share_{storage} = \frac{PowerCapacity_{storageunits}}{PeakLoad} \quad (3.3)$$

Share wind [%]:

$$Share_{wind} = \frac{PowerCapacity_{windunits}}{PeakLoad} \cdot AF_{wton} \quad (3.4)$$

Share pv [%]:

$$Share_{pv} = \frac{PowerCapacity_{pvunits}}{PeakLoad} \cdot AF_{pv} \quad (3.5)$$

Rntc [%]: The Net Transfer Capacity Ratio is a parameter that takes into account the effect of the capacity grid. It is calculated as follows:

Each NTC between two countries (c and x) can be estimated as the mean of its value for each hour of the horizon.

$$NTC_{c \to x} = \sum_h \frac{NTC_{c \to x,h}}{N_h} \qquad (3.6)$$

The NTC Ratio for each country can be defined as the sum of all the Net Transfer Capacities of this country to others divided by its maximum load.

$$NTC_c = \frac{\sum_x NTC_{c \to x}}{Maxload_c} \qquad (3.7)$$

Each country is then weighted by its contribution according to its maximum load.

$$NTC_{c,weigthed} = NTC_c \cdot \frac{MaxLoad_c}{\sum_c MaxLoad_c} \qquad (3.8)$$

Finally, the NTC Ratio is calculated as the sum of the NTC Ratios of all the countries:

$$R_{NTC} = \sum_c NTC_{c,weighted} \qquad (3.9)$$

The obtained values for one-year simulation are shown in Table 3.4:

Parameter	Value
Capacity ratio	1.658
Share flexibility	0.417
Share storage	0.497
Share wind	0.106
Share pv	0.035
Rntc	0.282

Table 3.4. Input parameters

3.4. Design of experiments

Design of experiments (DoE) refers to the different methods of locating sample points in a design space. The sample points should provide a good representation of the entire design space, so that the surrogate model can accurately predict complex simulation responses based on them [9]. Among all DOE methods, the Latin Hypercube Sampling (LHS) is applied, which generates random samples uniformly distributed in a sample space.

The parameters provided in Table 3.4 are varied, combined, and an optimization is run for each selected combination. A Latin hypercube sampling technique is used to avoid intractable computational time and cover the input space of the optimization model. A total amount of 2718 input parameter combinations are generated. The variation ranges of each input are detailed in Table 3.5.

Parameter	Range
Capacity ratio	0.5-1.8
Share flexibility	0.01-0.99
Share storage	0-0.5
Share wind	0-0.5
Share pv	0.2-0.5
Rntc	0-0.7

Table 3.5. Input parameters ranges

For each simulation, the input parameters are used to adjust, i.e. increase or decrease, the installed and storage capacity of the units as well as the Net Transfer Capacities. For this purpose, the following functions are created in the Dispa-SET repository:

Flexible Capacity: The function *adjust_flexibility* modifies the power capacities of flexible and slow units in order to achieve the imposed flexibility ratio $Share_{flex}$.

First, the *current_total_cap* is calculated as the sum of the power capacities of flexible and slow units for the whole system. Thus, the target of flexible capacity can be defined as:

$$target\ flexible\ capacity = current\_total\_cap \cdot Share_{flex} \qquad (3.10)$$

The flexible capacity remained to reach the desired $Share_{flex}$ can be calculated as the difference between the *target flexible capacity* and power capacities of flexible units:

$$\delta = target\ flexible\ capacity - current\_flex\_cap = current\_flex\_cap \cdot (Share_{flex} - 1) \qquad (3.11)$$

A dataframe is then created with the input zones (countries) as rows, and the following columns: *flex* ($PC_{flexunits}$), *slow* ($PC_{slowunits}$) , *total* ($PC_{flex+slow}$) and *ratio* ($Share_{flex}$ current ratio).

The dataframe is sorted based on the *ratio* column in descending order. Then, a new column *cum_sum* is added as the cumulative sum of the *total* column.

zones	*flex*	*slow*	*total*	*ratio*	*cum_sum*
1					
...					
n					

Table 3.6. Dataframe created in *adjust_function*

Finally, an algorithm loops through all the countries, by adding or subtracting the remaining capacity value to the current ones, to ensure that the final $Share_{flex}$ is the desired one. It works as follows:

if $\delta > 0$ **then**:

    remain = δ

    for $z \in zones$ **do**

        $weight = \frac{total_z}{current\_total\_cap - cum\_sum_z + total_z}$

        $added\_cap = min(weight \cdot remain, total_z - flex_z)$

        $new\_flex\_cap_z = flex_z + added\_cap$

        $new\_slow\_cap_z = slow_z - added\_cap$

    end for

else if $\delta < 0$ **then**:

    remain = $-\delta$

    for $z \in zones$ **do**

        $weight = \frac{total_z}{current\_total\_cap - cum\_sum_z + total_z}$

        $removed\_cap = min(weight \cdot remain, flex_z)$

        $new\_flex\_cap_z = flex_z - removed\_cap$

        $new\_slow\_cap_z = slow_z + removed\_cap$

    end for

else

    $new\_flex\_cap_z = flex_z$

    $new\_slow\_cap_z = slow_z$

end if

Power Capacity: The function *adjust_capacity* modifies the installed capacities of the units in the following table:

Units	New Power Capacity [MW]
$Storage_{units}$	$PeakLoad \cdot Share_{storage}$
$Wind_{units}$	$\frac{PeakLoad \cdot Cap\_ratio \cdot Share_{wind}}{AF_{wton}}$
PV_{units}	$\frac{PeakLoad \cdot Cap\_ratio \cdot Share_{pv}}{AF_{pv}}$

Table 3.7. New power capacities for wind, pv and storage units

Grid Capacity: The function *adjust_ntc* modifies the grid capacity by multiplying the Rntc value to the actual NTC interconnection lines values between all countries:

$$New\ Grid\ Capacity\ [MW] = NTC_{c \to x} \Big\downarrow \Big(\overrightarrow{NTC_{c \to x,h}}^{\ hours} \Big) \cdot R_{NTC}$$

A folder is created for each simulation, whose name its combination of parameters: *cap_ratio - flex - sto - wind - pv - rntc*. Inside are the required files to run the simulation in GAMS.

3.5. Output evaluation

The simulations are run on the HPC NIC5 cluster of the CÉCI (*Consortium des Équipements de Calcul Intensif*) hosted at the University of Liège. Parallel jobs are submitted, where 4 cores with 8000 MB each solve each simulation in GAMS at the same time. CPU time per simulation ranges from 6 to 22 hours.

Finally, the simulations are read and the unfeasible ones are discarded. The remaining ones are analysed and the parameters of interest in table 3.8 are extracted.

Parameter	Unit	Parameter	Unit
Cost	€/MWh	Shedding	TWh
Congestion	h	LostLoad	TWh
PeakLoad	MW	CF gas	%
MaxCurtailment	MW	CF nuc	%
MaxLoadShedding	MW	CF wat	%
Demand	TWh	CF win	%
NetImports	TWh	CF sun	%
Curtailment	TWh		

Table 3.8. Output parameters

Figure 3.1 presents an example of Curtailment on a dispatch plot from the first week of July in Germany. It shows that a large amount of electricity is exported due to high wind power generation. However, some of this energy has to be curtailed, coloured in red.
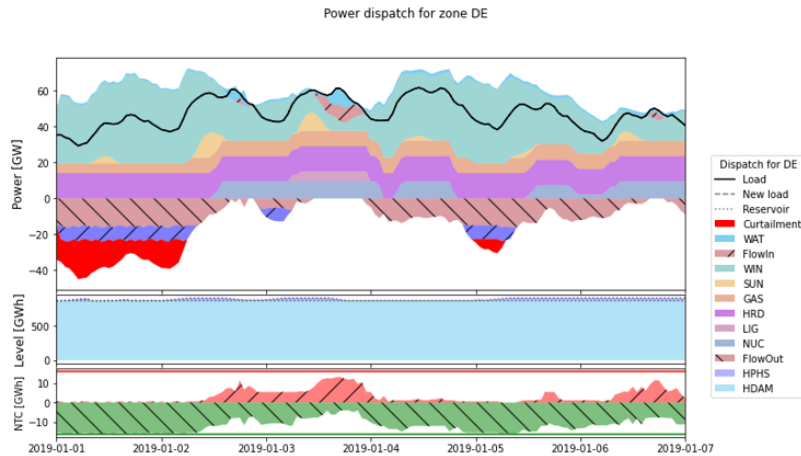

Fig. 3.1. Power dispatch plot for a week in July in Denmark

As mentioned before, the most common indicators to assess system flexibility are curtailment and load shedding. Disa-SET considers the possibility of voluntary load shedding as a result of contractual agreements between generators and consumers [10]. However,

it also considers some variables, called lost loads, that represent the capacity that cannot be provided by the system when the minimum or maximum power, reserve or ramp constraints are reached. Lost loads are an expensive last resort of the system that is used when no other options are available. The LostLoad parameter in Table 3.8 represents the sum of all lost loads variables.

It is therefore necessary to replace the load shedding indicator with the *ENS (Energy not served)* indicator, defined as the sum of load shedding and LostLoad, in order to evaluate the lack of flexibility during periods of low VRE generation.

Since the main goal of this thesis is to assess the flexibility of the system, Curtailment and Energy Not Served (ENS) are the two parameters to be predicted based on the input parameters.

3.6. Dataframe generation

The six input parameters used to generate simulations are combined with those from the Table 3.8 in order to generate the final DataFrame in .csv form.

The following Table 3.9 shows the parameters of the Dataframe which vary from each simulation.

Parameter	Unit	Parameter	Unit
Cost	€/MWh	CF nuc	%
Congestion	h	CF wat	%
PeakLoad	MW	CF win	%
MaxCurtailment	MW	CF sun	%
MaxLoadShedding	MW	Capacity ratio	%
Demand	TWh	Share flex	%
NetImports	TWh	Share sto	%
Curtailment	TWh	Share Wind	%
ENS	TWh	Share PV	%
CF gas	%	Rntc	%

Table 3.9. Dataframe parameters

The final simulation results can be found in the generated *dispaset_results.csv* of the author's repository. The scripts and data used for the Dispa-SET EU simulations are available at: https://github.com/MPavicevic/DispaSET-SideTools/tree/Matijs/dispaset_sidetools.

4. SURROGATE MODEL CREATION AND VALIDATION

4.1. Introduction

This chapter describes the creation of a surrogate model using machine learning (ML) algorithms. This model predicts the adequacy and flexibility of the EU-wide Dispa-SET power system model, making it possible to couple it with the MEDEAS system dynamics model.

4.2. Machine learning regression methods

The Artificial Neural Network (ANN) algorithm is used for fast parameter estimation of curtailment and Energy not served (ENS). Although neural networks are well known techniques for classification problems, they can also be applied to regression problems. Regression ANNs predict an output variable as a function of the inputs. The input features (independent variables) can be categorical or numeric types while the output must be a numeric variable. Since the goal is to predict two output values, two neural networks are developed in regression mode.

The input dataset used is the one obtained in Table 3.9 in Chapter 3. Each neural network has the following structure, with X being the inputs and Y being the target:

i	X						y
	Capacity ratio	Share flex	Share sto	Share wind	Share PV	Rntc	ENS
1							
...							
n							

Table 4.1. Dataframe for ENS

i	X						y
	Capacity ratio	Share flex	Share sto	Share wind	Share PV	Rntc	Curtailment
1							
...							
n							

Table 4.2. Dataframe for Curtailment

4.3. Neural Network architecture

Artificial Neural Networks (ANNs) are biologically inspired computational networks composed of layers of nodes. Each node, or artificial neuron, is connected to another and has an associated weight and threshold (or bias).

Among the different types of ANNs, this thesis focuses on multilayer perceptrons (MLPs), based on a supervised learning algorithm. MLP architecture is a layered feed-forward neural network, where neurons are arranged in consecutive layers, and the information flows unidirectionally, from the input layer to the output layer, through the hidden layer(s). Nodes of one layer are fully connected to all nodes of the adjacent layer.
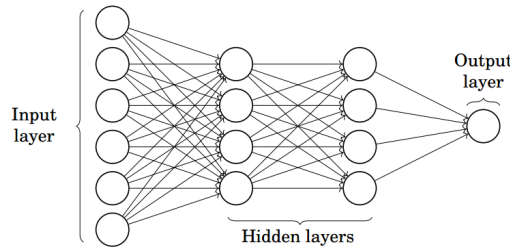
Fig. 4.1. Artificial Neural Network [14]

4.3.1. Neurons

Neurons are the atomic units of a neural network and are arranged into layers. They consist of two functions that work together: a linear and an activation function.

Linear function
The output of the linear function is the sum of the inputs, each multiplied by a coefficient (or weight) plus the bias term.

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{4.1}$$

Where x_i are the inputs, w_i are the weights and b is the bias term.

Activation function
The activation function is responsible for introducing a non-linearity between neurons. It squashes (or limits) the amplitude of output signal into a finite value. For simplicity, nodes from the same layer use the same activation function. There are many activation functions, but the following are considered:

Rectified Linear (ReLU)

$$f(z) = max(z, 0) \tag{4.2}$$

Hyperbolic Tangent (Tanh)

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{4.3}$$

Figure 4.2 shows an example of a neuron with $(x_1, ..., x_m)$ inputs and their corresponding weights $(w_1, ..., w_m)$, a bias (b) and the activation function applied to the weighted sum of the inputs.
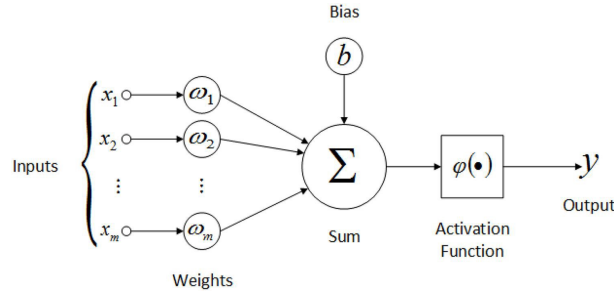

Fig. 4.2. Artificial Neuron [15]

4.3.2. Loss and cost function

The loss function is defined as the correction of the fit for a single observation within the training set. The cost function (J) measures how well the model fits the training data as the average loss over the entire training set.

For regression tasks, the most common cost functions are the Mean Squared Error (MSE) and the Mean Absolute Error (MAE).

The MSE computes the mean of squares of errors between labels and predictions while the MAE computes the mean of absolute difference between labels and predictions.

$$MSE = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N} \qquad (4.4)$$

$$MAE = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{N} \qquad (4.5)$$

Where N is the number of training samples in a set, $\hat{y}$ is the predicted value and y is the actual target value.

The MAE is used as the loss function.

4.3.3. Forward propagation

It is the process that calculates and stores intermediate variables (including the target value) using the features present in a single observation in order from the input layer to the output layer. For the first training example, biases are typically initialised to 0 and weights to values from a normal distribution N(0,1).

4.3.4. Back propagation

Once forward propagation is completed, the network predictions for each data point ($\hat{y}$) are obtained. As the target value of each data point is also known (y), the network error can be estimated using the cost function.

For the neural network to learn, the network error signal must be propagated backwards through the network layers from the last to the first, updating the weights and biases of the network as the signal travels. Mathematically, the cost function (J) must be minimized by fine-tuning the weights and biases. This process is known as *Gradient Descent algorithm*.

This method calculates the gradient of the cost function with respect to all the parameters to update within the network ($\frac{\partial J}{\partial \theta}$). Gradients of each variable can be calculated using the chain rule and gradients of above layers.

Once the gradient of the variable is known, the Gradient Descent algorithm iteratively updates its value using the gradient at the current position (direction of the steepest descent), then scales it (by a learning rate α) and subtracts the obtained value from the current position (makes a step). This process can be expressed as:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\partial J}{\partial \theta_t} \tag{4.6}$$

Where θ is the variable to be modified during the optimization of the model which can be weights or biases.

The Gradient Descent repeats this update until the cost function converges. Note that, in this case, the model update is performed only after all training examples are evaluated, since the cost function sums the error for each point within training set. This is known as **Batch Gradient Descent** as it has only one batch of the size of the training set.
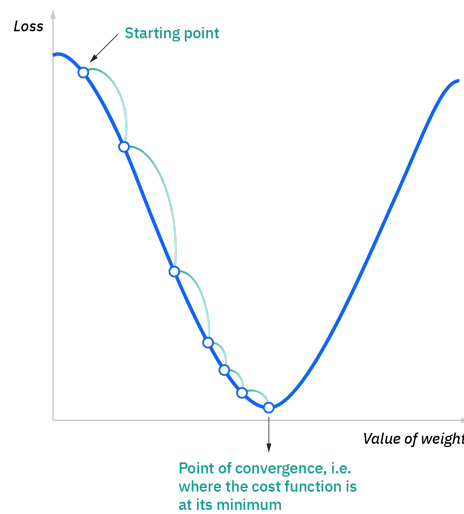

Fig. 4.3. Gradient Descent algorithm [16]

There are two other types of Gradient Descent based on the batch size:

Stochastic gradient descent (SGD): The batch size is of one training sample, meaning that it updates each training example's parameters one at a time.

Mini-batch gradient descent: It combines both batch and stochastic gradient descent. It splits the training dataset into small batches and updates each batch separately. The batch size is between one and the size of the training set.

4.3.5. Batch size and number of epochs

These two hyperparameters must be specified in the algorithm. The batch size is the number of samples processed before the model update while the number of epochs is the number of complete passes through the entire training dataset.

4.3.6. Learning rate

The learning rate α is an important hyperparameter which scales the gradient and thus controls the step size. It has a strong influence on performance and its value is usually between 0.0 and 1.0.

High learning rates result in larger step sizes, but there is a risk that the model converges too quickly to a suboptimal solution. On the other hand, low learning rates have the advantage of higher accuracy, but compromise overall efficiency as they are computationally expensive.
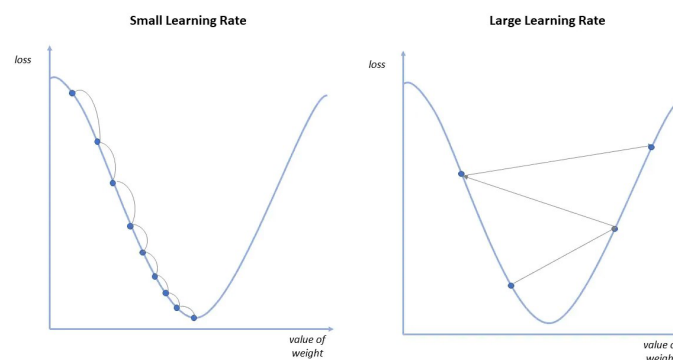

Fig. 4.4. High and low learning rates [16]

4.3.7. Other optimization algorithms

Gradient descent can run into certain problems during training that can slow down the learning process or, in the worst case, even prevent the optimal weights from being found. These are saddle points and local minima, that is, when the cost function becomes flat

at this point, and its gradient approaches zero. Gradients close to zero do not improve weight parameters and impede the entire learning process. [17]

Three optimization algorithms are used to solve these problems:

Gradient descent with Momentum

Momentum is a method that speeds up gradient descent in the relevant direction thus leading to faster converging. It can be defined as the moving average over the past gradients and then use it to update the weights of the network. This could be written as follows:

$$v_t = \beta \cdot v_{t-1} + (1 - \beta) \cdot \frac{\partial J}{\partial \theta_t}$$
$$\theta_{t+1} = \theta_t - \alpha \cdot v_t \tag{4.7}$$

The momentum term comes from an analogy to physics. One can think of the derivative terms ($\frac{\partial J}{\partial \theta_t}$) as providing acceleration to a ball rolling downhill and the momentum terms (v_{t-1}) as the velocity. Finally, the momentum parameter (β) can be considered as a friction that "slows down" the velocity. The most common value for this hyperparameter is 0.9, which is the value employed.

The saddle points and local minima become less critical for gradients when using the momentum term, since the step size toward the global minimum now depends not only on the slope of the cost function (α), but also on the cumulative velocity over time (v_t).

RMSProp Algorithm

Through the multidimensional space represented by the network's weights, RMProp algorithm can accelerate gradient descent in some directions, and dampen oscillations in others.

$$d\theta_t = \frac{\partial J}{\partial \theta_t}$$
$$v_t = \beta \cdot v_{t-1} + (1 - \beta) \cdot d\theta_t^2 \tag{4.8}$$
$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{d\theta_t}{\sqrt{v_t} + \epsilon}$$

Adam optimizer

Adam (derived from *adaptive moment estimation*) is one of the best optimization algorithms. It combines the best properties of Momentum and RMSProp algorithms.

$$d\theta_t = \frac{\partial J}{\partial \theta_t}$$
$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot d\theta_t$$
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot d\theta_t^2 \tag{4.9}$$
$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{m_t}{\sqrt{v_t} + \epsilon}$$

4.4. Frameworks

TensorFlow is an open-source deep learning library developed by Google used to build deep neural networks. A computation graph (or graph) is the basic unit of computation in TensorFlow. It is a series of TensorFlow operations arranged into a graph of nodes. The nodes of the graph represent the mathematical operations covered so far, while the edges between these nodes represent the multidimensional data arrays (or tensors). Given a loss function and a neural network defined as a graph, TensorFlow can compute gradients for the network and optimize the graph by minimizing the loss function.

Keras is a high-level API written in Python, running on top of TensorFlow. Designed to enable fast experimentation, Keras allows to quickly iterate through model architecture in an easier way.

Scikit-learn is an open source data analysis library that offers a selection of efficient tools for machine learning and statistical modeling.

4.5. Training, validation and test datasets

In order to avoid overfitting, the data is shuffled and then split into three sets, each with its respective percentage in parenthesis:

Training set (70%) It is used to train the network.

Validation set (10%) It is used to find the best hyperparameters and to measure overfitting. After every epoch, a prediction is made on the validation set to monitor the overfitting and to know when the training process is completed.

Test set (20%) Once training is complete, the test set is used to measure the model performance on a dataset that the network has not seen.

4.6. Data pre-processing

DataFrame inputs cover different ranges. Some ML algorithms are highly sensitive to these features and therefore the variable with the larger scale totally dominates when trying to predict the trend. These include machine learning algorithms that use gradient descent as an optimization technique or those that calculate distances between data. Thus, feature scaling needs to be performed on the input variables.

However, scaling output variables must also be considered if there are big differences in terms of scale. A target variable with a large range of values may result in large error gradient values, causing weight values to change dramatically, and thus making the learning process unstable. By reducing the scale of the target variable, the size of the gradient used to update the weights is reduced, resulting in a more stable model and training process.

As shown in Table 4.3, the magnitude of dataset range for each target variable is quite large, requiring scaling the output variables.

Target	Unit	Mean	Standard deviation	Min	Max
Curtailment	TWh	1351.0738	1066.5681	0.0228	5572.3302
ENS	TWh	53.0120	29.9609	0.0035	157.2003

Table 4.3. Statistical description of target values

The most commonly used feature scaling techniques are Min-Max Normalization (MMN) and Z-Score normalization or standardization (ZSN). MMN rescales the dataset so that each value ranges between 0 and 1 while ZSN rescales the dataset to have zero mean and unit variance.

Normalization is appropriate when the data does not follow a Gaussian distribution. As neural networks do not assume any data distribution, this pre-processing transformation is used.

$$X_{new} = \frac{X_i - min(X)}{max(X) - min(X)} \tag{4.10}$$

Preprocessing.MinMaxScaler() function in the sklearn library allows to scale each feature between 0 and 1.

4.7. Structure of MLP

Keras uses an instance of a model object to contain a neural network. The neural network is defined using a *Sequential model* which consists of a linear stack of layer objects, each with an input and output tensor.

Input layer shape
The shape of the input matrix is (Number of data observations x Number of features). It is possible to define the number of observations in a dataset using *None* as a placeholder, which means that the dimension can take any integer value.

Hidden layer(s) shape
The number of hidden layers is an hyperparameter to tune, as well as the nodes within these layers. The shape of the first hidden layer would be (Number of features x Number of neurons of the HL).

Output layer shape
A single neuron comprises the output layer that, by using the inputs from the last hidden layer, predicts a single output value for each observation ($\hat{y}$).

4.8. Bias and variance

Bias-variance trade-off is a fundamental principle for understanding the generalization of traditional predictive learning models.

- **Bias error**. It refers to the error introduced by the model due to its erroneous assumptions.

- **Variance error**. It refers to the error that is introduced by the sensitivity to fluctuations in the training set.

The bias-variance trade-off predicts that as model complexity increases, bias decreases and variance increases, leading to a U-shaped test error curve. However, recent empirical studies with over-parameterized neural networks show that the test error curve does not conform to the classical U-shape: the test error decreases as networks become wider. There is evidence that both bias and variance decrease as the number of parameters increase in common classification and regression settings [18].

As a result, the trade-off between bias and variance can be overcome in deep neural networks so that bias and variance can be manipulated independently. An overview of how bias and variance errors can be controlled in a deep neural network is given below.

- **High bias**: High error rate when predicting on the training set, i.e. the model is not fitting the data well. To reduce the bias, the network architecture may need to be changed by adding layers, neurons, or both.

- **High variance**: Low bias error means that a network fits the training data well. However, if the validation error exceeds the test error, the network is *overfitting*. Adding data and regularization to the network are the best ways to reduce variance. Among the most common regularization techniques, such as L2 regularization, batch normalization or dropout, the latter is used.

4.9. Regularization techniques: Dropout

Dropout is a regularization technique in which randomly selected neurons are "dropped-out" during training. Dropping a unit out means temporarily removing it from the network, as well as its incoming and outgoing connections. The presence of other hidden units becomes unreliable as a result of dropout, preventing co-adaptation [19].

Applying dropout is equivalent to sample an exponential number of different "thinned" networks during training. A thinned network consists of all the units that weren't removed. The probability p of retaining a unit during training is an hyperparameter to be considered.

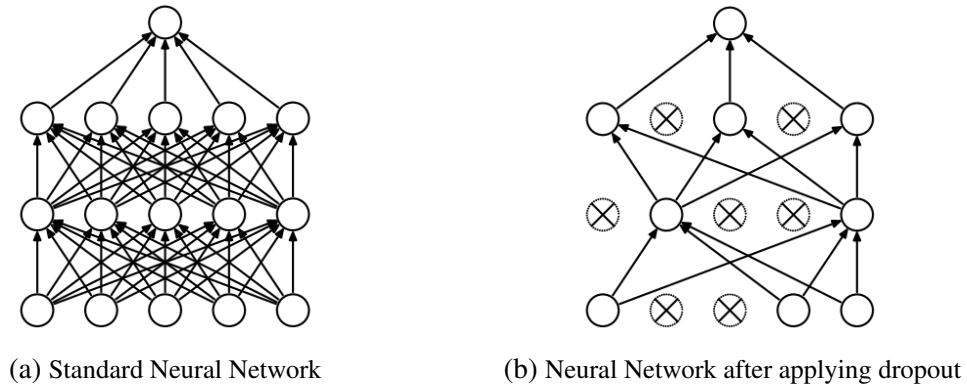(a) Standard Neural Network   (b) Neural Network after applying dropout

Fig. 4.5. Dropout technique [19]

At test time, the outgoing weights of that unit are scaled down by multiplying them by p. This single "unthinned" network with smaller weights approximates the effect of averaging the predictions of the "thinned" networks.

Dropout can be implemented on any hidden layer(s) or the input layer. In this study, dropout is implemented on all hidden layers by adding a Dropout layer in the *Sequential model* after each hidden layer. The Dropout layer takes a *dropout rate* as an argument, which is defined as the fraction of the input units to drop (1-p). This is an hyperparameter to be tuned.

4.10. Hyperparameter tuning

Deep Neural Network (DNN) models learn the values of their parameters, such as connection weights and bias, from training data. Model learning or training is the process of determining their values from the data. However, there are a few high-level parameters called hyperparameters whose values cannot be learned from the data and have to be defined beforehand.

Hyperparameters control both the training process and the topology of an ML model, which can have a significant impact on the ML program's performance. There are two types of hyperparameters:

- **Model hyperparameters**: These influence model selection, such as the number and width of hidden layers.

- **Algorithm hyperparameters**: These influence the quality and speed of the learning algorithm, such as the learning rate or the batch size.

The Keras Tuner library is used to find the optimal set of hyperparameters, which is called hyperparameter tuning or hypertuning.

4.10.1. Model building function

A *build_model* function must first be written, which takes an hp argument to define the hyperparameters and returns a compiled Keras model. The hp object is an instance of the Keras Tuner HyperParameters class and specifies the range of hyperparameter values by three methods: *hp.Choice, hp.Int* and *hp.Float*.

The hyperparameters selected for the ANN model and their ranges are shown in Table 4.4.

Hyperparameter	Name	Type	Interval/step or Values
Network	Number of hidden layers	Int	[1-2]/1
	Learning rate	Float	[0.0001-0.01]/log
	Optimizer	Choice	SDG,rmsprop,adam
Hidden Layer	Number of units	Int	[32-512]/32
	Dropout rate	Int	[0.5-0.8]/0.1
	Activation function	Choice	relu, tanh

Table 4.4. Selected hyperparameters

The *build_model* function builds one of the models from the search space using the HyperParameters object.

4.10.2. Tuners

The Keras Tuner library provides the Tuner class to manage the hyperparameter search process. Tuner subclasses include RandomSearch, Hyperband, and BayesianOptimization for widely used tuning algorithms. By using these algorithms, good hyperparameter settings can be found in fewer trials and without running all possible combinations (i.e. Grid Search).

Random Search
This tuner randomly picks different combinations from all possible hyperparameter combinations in order to find a good fit. Compared to Grid Search, Random Search allows exploring more hyperparameter space in less time.

Hyperband
This tuner focuses on speeding up Random Search through adaptive resource allocation and early stopping. The algorithm trains a large number of random models for a few epochs (less than the maximum) and carries forward the best performing models to the next round. Iteratively, the number of candidate models decreases and their resources (epochs) increase. Lastly, the final candidates are fully trained and evaluated. Algorithm input parameters to set are:

- *max_epochs*: The maximum number of epochs to allocate to a single model. It

should be slightly higher than the largest model's expected convergence epochs.

- *factor*: Factor that reduces the number of models and increases the number of epochs. It is set to the default value, 3.

The number of models to train is calculated as: $1 + \log_{factor} max\_epochs$ rounded up.

During training the early stopping technique is used by calling the callback object *keras. callbacks.EarlyStopping (monitor='val_loss', patience=5)* from the Keras tuner library. The training stops if the *'val_loss'* doesn't improve in 5 epochs.

Bayesian

In contrast to random search and hyperband, Bayesian optimization takes previous evaluations into account when choosing the next set of hyperparameters to evaluate.

Initially, the algorithm randomly chooses some hyperparameter combinations and then, based on their performances, it uses Bayes theorem to select the next best candidate. The last step is repeated iteratively until the tuner reaches optimal hyperparameters or exhausts all trials allowed.

Figure 4.6 shows how some different search algorithms mentioned fill in the hyperparameter space. The colour gradient of the points indicates the order of the search, from black to yellow to orange.
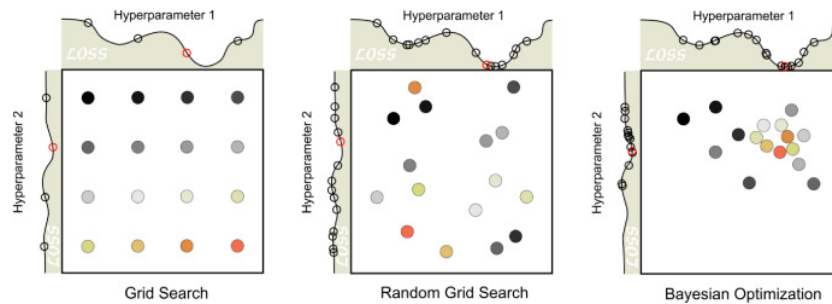

Fig. 4.6. Comparison of hyperparameter search methods [20]

The most important arguments for both Random Search and Bayesian tuners are:

- *max_trials*: Number of trials (hyperparameter combinations) to execute during the search. It is set to 16.

- *executions_per_trial*: Number of times to execute each trial. Given the stochastic nature of the Gradient Descent optimization algorithms, results may vary when rerun the same model. Multiple executions per trial reduce variance and enable a more accurate assessment of the model's performance. It is set to 3.

To instantiate any of the tuners, two common arguments must be passed: *hypermodel* and *objective*. For the first one, the *build_model* function is set, which returns a model

instance. As for the second one, the *'val_loss'* string is passed since the tuner needs to minimize the validation loss. All three tuners are used and their selected optimal hyperparameters are compared.

4.10.3. Hyperparameter tuning process

The method *tuner.search()* is called, which performs a search for best hyperparameter configurations. For each trial, the tuner trains the model on the training set obtaining its optimized weights (parameters) and it is then evaluated in the validation set. Below is a small scheme of what happens during hyperparameter optimization.
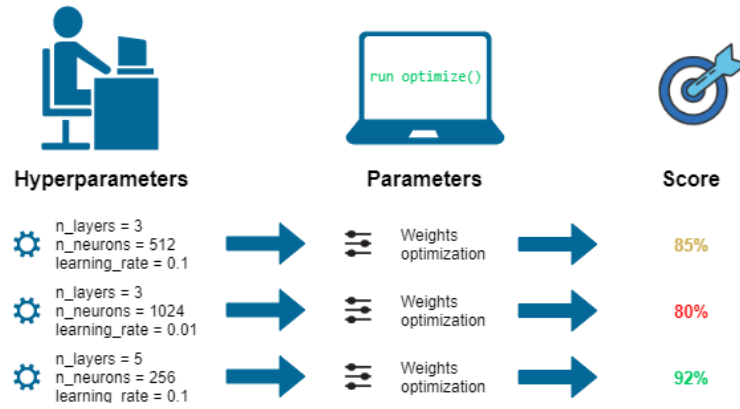

Fig. 4.7. Hyperparameter optimization [21]

However, two training hyperparameters, the batch size and the number of epochs, must be set before training. The *batch size* is set to 32, the default value, which generally gives good results [22]. For the *number of epochs*, it can be initialized to a large number, such as 300, and visualize through the learning curves at what epoch the validation loss value stabilizes. By calling *keras.callbacks.Tensorboard* during the search, the learning curves of the trials can be visualised through tensorboard. In order to determine the *epochs* parameter, 6 trials are run.
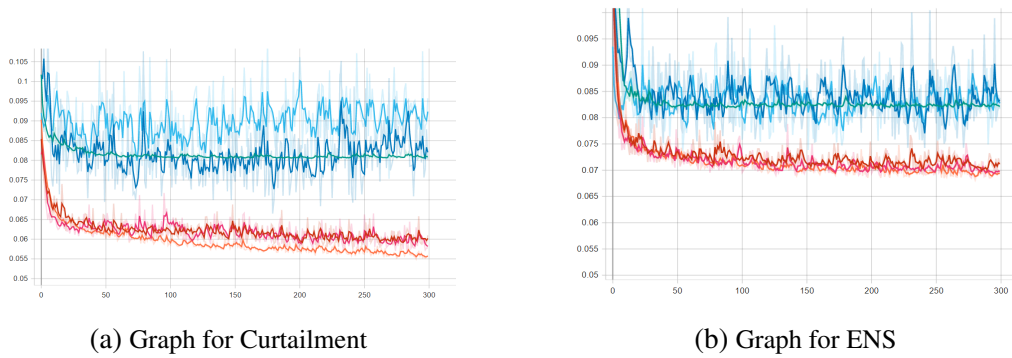

(a) Graph for Curtailment

(b) Graph for ENS

Fig. 4.8. *Validation loss* curves

The plots in Figure 4.8 shows that the value of *'val_loss'* for the best trials stabilizes

approximately at *epoch* 150 for both Curtailment and ENS. For this reason, this is the value employed for the searching.

Once the searching is completed, the *tuner.get_best_models (num_models=1)* method returns the best model based on the tuner's objective. The model is loaded at its best epoch evaluated on the validation set.

At the same time, the *tuner.get_best_hyperparameters (num_trials=1)* returns the hyperparameters of the best model. The following tables 4.5 and 4.6 break down the hyperparameter results for each optimizer:

Hyperparameter	Random Search	Hyperband	Bayesian
Number of hidden layers	1	1	1
Learning rate	0.00176	0.00287	0.00079
Optimizer	adam	sgd	adam
Units Hidden Layer 1	480	448	512
Dropout Hidden Layer 1	0.6	0.6	0.5
Activation Hidden Layer 1	relu	relu	relu

Table 4.5. Optimal hyperparameters for Curtailment

Hyperparameter	Random Search	Hyperband	Bayesian
Number of hidden layers	1	1	1
Learning rate	0.00176	0.00101	0.00170
Optimizer	adam	adam	adam
Units Hidden Layer 1	480	288	512
Dropout Hidden Layer 1	0.6	0.7	0.5
Activation Hidden Layer 1	relu	relu	relu

Table 4.6. Optimal hyperparameters for ENS

4.11. Validation

For best performance, it is recommended to retrain the model on the entire dataset (both validation and training set). In order to do so, the best model should be reinstantiated with the best hyperparameters and then trained. Theoretically, more training data will make the final model more generalizable to new data.

The retrained model is saved at its best epoch in a checkpoint file so that it can be loaded later. This is done by calling the callback *keras.callbacks.ModelCheckpoint()* when training using *model.fit()*.

Finally, the final model performance is measured using the test set. Figure 4.9 shows a scheme of the hyperparameter tuning process followed.
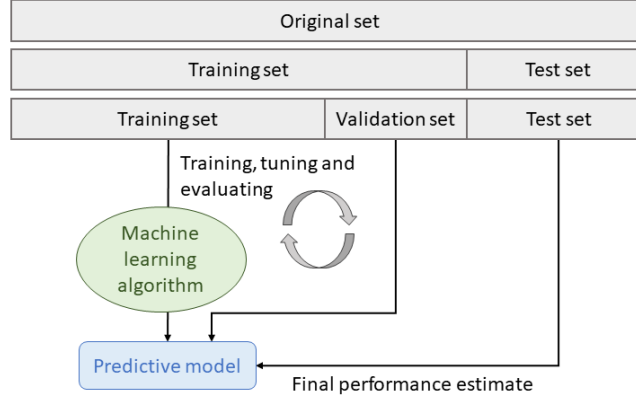
Fig. 4.9. Hyperparameter tuning process. Figure inspired by [23]

Since the target values were scaled during pre-processing, predictions should be scaled back. This is achieved by using the *inverse_transform* function from scikit-learn library.

Four performance metrics are used to evaluate the model prediction: MAE, MSE, RMSE, and R^2.

Root Mean Squared Error (RMSE): Square root of mean squared error.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{4.11}$$

Coefficient of determination (R^2) : Statistical measure that defines the percentage of variance in the dependent variable (target) that can be explained by the independent variables (features) in a regression model. In other words, it measures the goodness of fit of a model, i.e. how well the regression model fits the data.

It is calculated by dividing the sum of squares of the residuals (SSQ) of the regression model by the total sum of squares (SST) of the mean model and subtracting 1. Values range from 0 to 1.

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}^2)} \tag{4.12}$$

Metrics are calculated using both normalized and rescaled predictions. The following tables 4.7 and 4.8 show the metric values obtained for both scales:

	Scaled Test data				Non-scaled Test data			
	MAE	MSE	RMSE	R2	MAE	MSE	RMSE	R2
Random Search	0.04136	0.01107	0.10523	0.69735	230.454	343838.661	586.378	0.69735
Hyperband	0.04336	0.01362	0.11671	0.62772	241.616	422941.666	650.340	0.62772
Bayesian	0.03886	0.01064	0.10314	0.70924	216.527	330327.794	574.742	0.70924

Table 4.7. Statistical metrics for Curtailment

	Scaled Test data				Non-scaled Test data			
	MAE	MSE	RMSE	R2	MAE	MSE	RMSE	R2
Random Search	0.05700	0.01552	0.12458	0.56441	8.959	383.518	19.584	0.56441
Hyperband	0.05740	0.01558	0.12482	0.56275	9.024	384.984	19.621	0.56275
Bayesian	0.05632	0.01573	0.12544	0.55840	8.853	388.813	19.718	0.55840

Table 4.8. Statistical metrics for ENS

As a result of the increased range of the scale, the MAE, MSE, and RMSE errors in the original scale are larger than in the normalized values. Nevertheless, the results show good metrics for both Curtailment and ENS.

Bayesian optimization returns the lowest error metrics and the highest R^2 coefficient for Curtailment. As for ENS, bayesian optimization provides the lowest MAE error while random search optimizer provides the highest R^2. However, it does not necessarily mean that the bayesian tuner should always be used. Tables 4.5 and 4.6 show some discrepancies in hyperparameters selected by each of the tuners.

For Curtailment (Table 4.5), the number of hidden layers and the activation function of the first layer are the same across all three tuners. For ENS (Table 4.4), they also agree on the optimizer. As a result, the hyperparameters that are common to all three tuners have a greater effect on the loss function than the others.

Figures 4.10 and 4.11 illustrate the results obtained through Bayesian optimization.

Figure 4.10 illustrates the loss curves during testing, with the testing curve being lower than the training curve. This occurs due to the dropout regularization mechanism. At training time, the network does not operate at full capacity, resulting in a high training loss. At test time, the dropout is turned off, resulting in a lower test loss. Moreover, as the model changes over time, the loss in the first batches of an epoch is usually larger than in later batches. The testing loss for an epoch is therefore calculated using the model at the end of the epoch, which results in a lower loss [24].
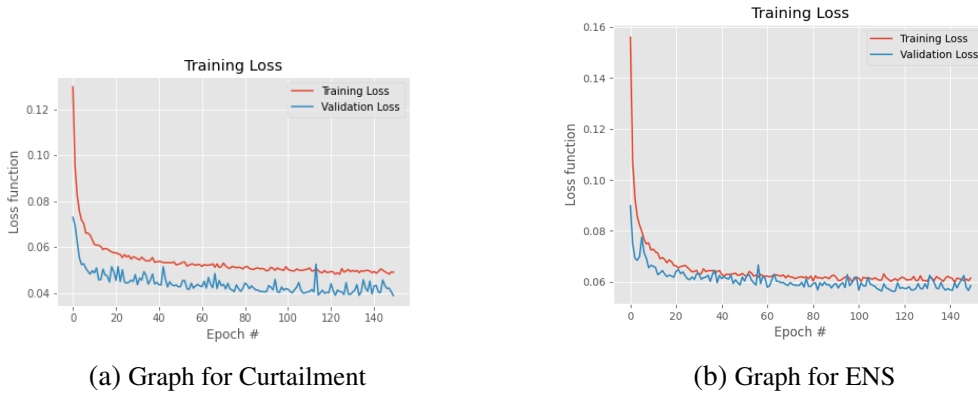

(a) Graph for Curtailment   (b) Graph for ENS

Fig. 4.10. *Testing and training loss* curves

Figure 4.11 shows scatter matrices for (a) curtailment and (b) energy not served. These

matrices plot the design variables in relation to each other and to the target value. Apparently, there is no cross-correlation between the features, which means that the Latin Hypercube technique is effective. However, scatter plots between target values and variables reveal some trends.

Curtailment (a) generally increases with *share_pv*, *share_wind* and *capacity ratio* variables. On the other hand, Energy not served (b) shows a decreasing trend with decreasing *share_sto* and *rNTC*.

(a) Matrix plot for Curtailment
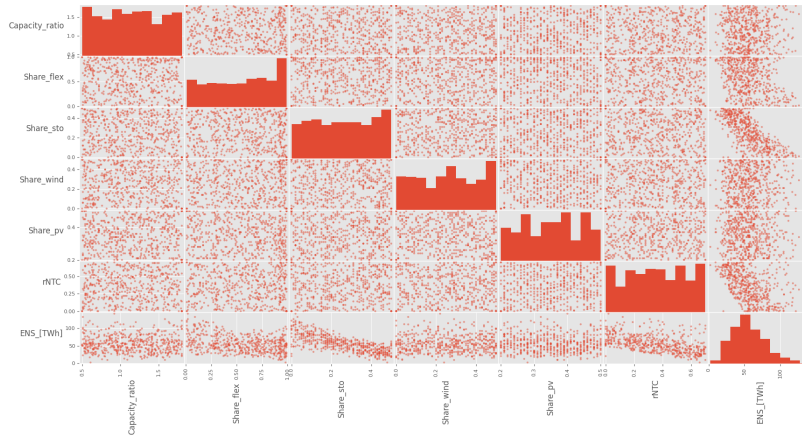


(b) Matrix plot for ENS

Fig. 4.11. Matrices plots

4.12. Regression results

Regression plots are generated by loading the best model from the checkpoint file and evaluating it in a series of test sets. Each test set is calculated by varying two input parameters, e.g. *Share flex* and *Share sto*, while keeping the others constant. Therefore, it is possible to predict target values as a function of two variables.

Figure 4.12 shows four Curtailment plots generated by combining the following input parameters: *Share wind*, *Share pv*, *Share flex*, *rNTC* and *Capacity factor*.

Figures (a) and (b) show that the curtailment increases as the *Capacity factor*, *Share wind* and *Share pv* increase, since the system flexibility remains unchanged.

Alternatively, figure (c) shows that curtailment decreases as system flexibility increases for a given level of *Share pv*. Thus, the maximum level of curtailment occurs when *Share pv* is at its maximum and *Share flex* is at its minimum. Similarly, Figure (d) illustrates that curtailment decreases with increasing flexiblity through grid extension.
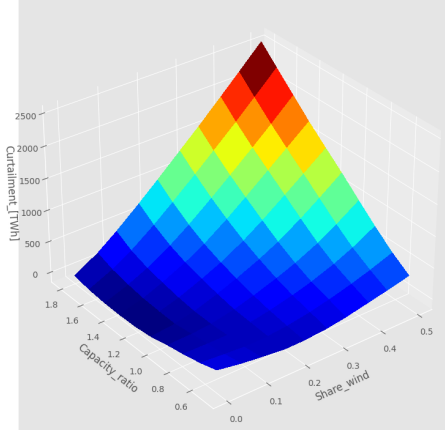
(a) Curtailment vs *Share wind* and *Share pv*
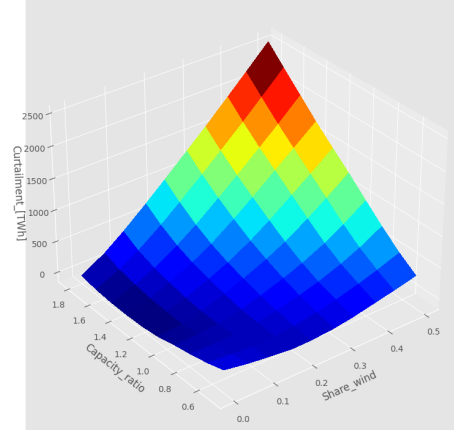
(b) Curtailment vs *Share wind* and *Capacity ratio*
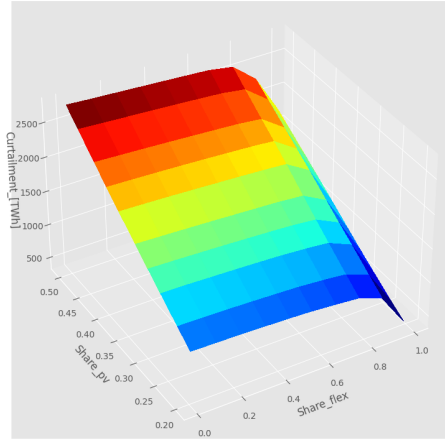
(c) Curtailment vs *Share pv* and *Share flex*
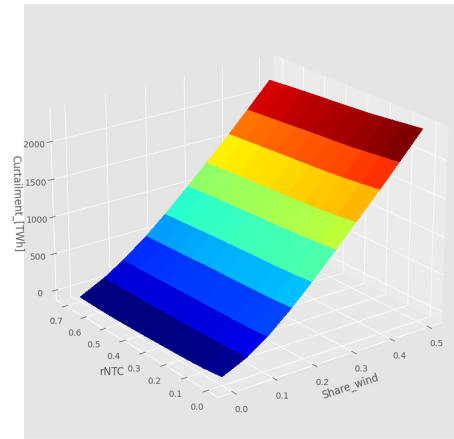
(d) Curtailment vs *Share wind* and *rNTC*

Fig. 4.12. Curtailment plots

Likewise, in Figure 4.13, four plots are generated for the ENS by combining the same input parameters as in Curtailment.

Figures (a) and (b) show that ENS tends to increase as the *Capacity factor*, *Share wind* and *Share pv* decreases.

On the other hand, Figures (c) and (d) show that as flexibility of the system increases through *rNTC* and *Share flex* parameters, ENS decrease. ENS reaches its maximum when RES generation and system flexibility are lower.
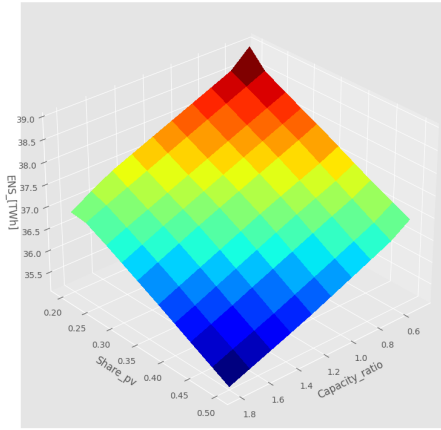
(a) ENS vs *Capacity ratio* and *Share pv*
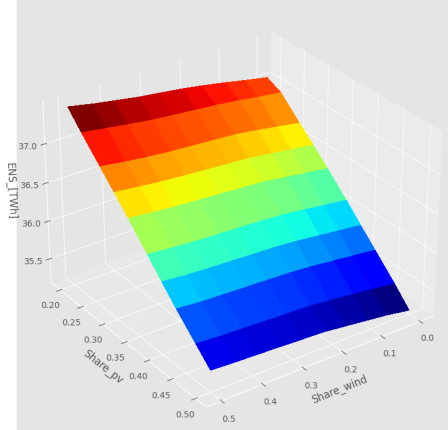
(b) ENS vs *Share pv* and *Share wind*
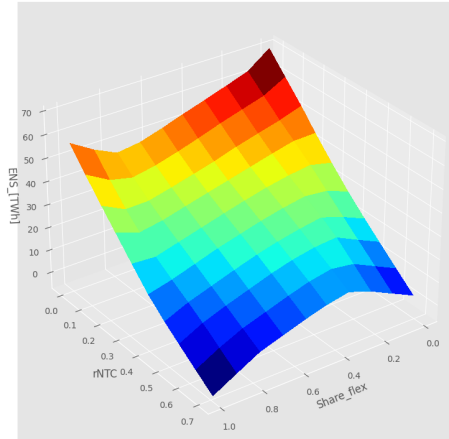
(c) ENS vs *Share flex* and *rNTC*
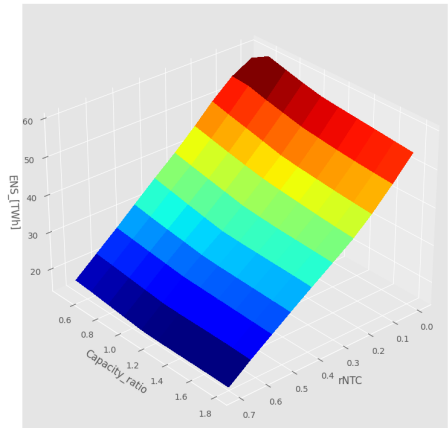
(d) ENS vs *Capacity ratio* and *rNTC*

Fig. 4.13. ENS plots

As illustrated in Figures 4.12 and 4.13, there is no linear relationship between the variables. Due to this non-linearity, neural networks appear to be a good option when predicting flexibility indicators, since a linear regression would not be able to predict the values accurately.

5. MEDEAS MODEL

5.1. Introduction

This chapter includes an overview description of the MEDEAS model and its framework. The major limitation of MEDEAS is the low temporal resolution, which prevents a proper flexibility assessment such as the computation of the excess renewable energy or the lack of peak capacity. The main solution presented is based on the integration of the surrogate model into MEDEAS. Although such integration has not been addressed in this work, this chapter evaluates how the management of curtailment and energy not served could be improved.

5.2. MEDEAS Model

Integrated environmental assessment modelling describes any type of analysis that integrates multiple disciplines and dimensions in order to gain a better understanding of human-environmental interactions, which are often highly dynamic, non-linear, and complex. Its objective is to provide an holistic knowledge of system and project alternative future climates as a useful information for policy-making [25].

Integrated Assessment Models (IAMs) or Energy-Economy-Environment (E3) Models are computer programs that connect a series mathematical representations of information from different disciplines. There is a great variety of IAMs due to the diverse approaches developed to address complex interactions and high uncertainties in the environmental and human systems.

The open-source MEDEAS modelling framework is designed with the objective of informing decision-making to achieve the transition to sustainable energy systems by addressing some of the limitations identified in current IAMs and focusing on biophysical, economic, social, and technological restrictions.

The model was built using Vensim DSS software for Windows Version 6.4E (x32) and is also available in Python open-source code.

5.3. Overview of MEDEAS modelling framework

MEDEAS models analyse the long-term strategic outcomes of human-nature interactions and are designed using System Dynamics, which facilitates integrating knowledge from different perspectives and disciplines, along with feedback from different subsystems.

Its aim is to develop medium to long-term socio-economic-environmental scenarios at

three different geographical levels: global (MEDEAS-W), European Union (MEDEAS-EU) and country-level for Austria and Bulgaria (MEDEAS-AU and MEDEAS-BGR). The simulation horizon is usually between 1995 and 2060, although for long-term strategic sustainability analyses it can be extended to 2100.

MEDEAS models are structured in seven main modules: Economy, Energy, Energy Infrastructures, Materials, Land Use, Climate Change and Social and Environmental Impacts Indicators. Figure 5.1 illustrates the main relationships between the different modules of MEDEAS models.
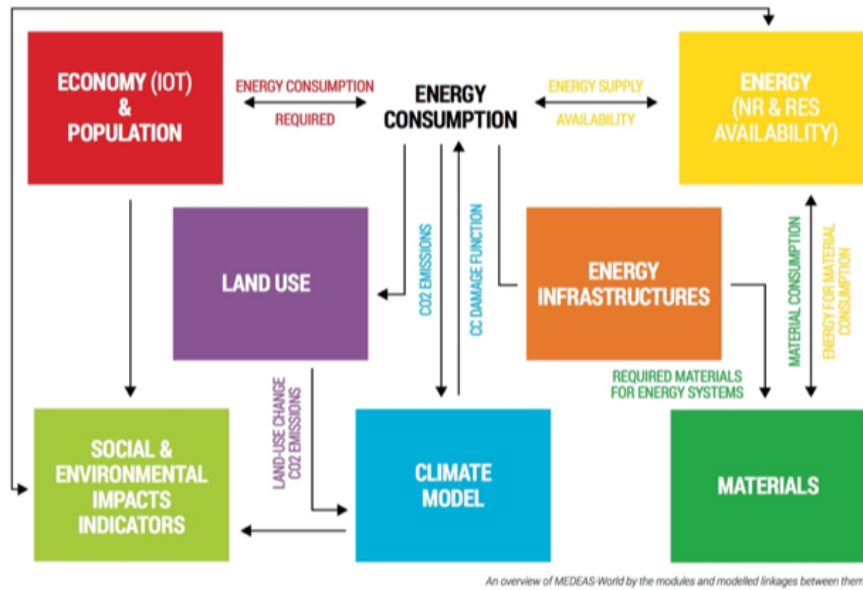
Fig. 5.1. Conceptual schematic overview of the MEDEAS models [26]

In the present thesis, the European-aggregated scale MEDEAS-EU is considered, focusing on the Energy module.

5.4. Energy Returned on (Energy) Invested (EROI)

EROI is defined as the ratio of the amount of usable energy obtained from a particular energy resource (*Energy_returned*) to the amount of energy expended to obtain that energy resource (*Energy_invested*) [12]. EROI is a key factor when evaluating the efficiency of an energy source, and RES technologies generally have lower values than conventional fossil fuels. [27], [28].

Net energy, on the other hand, is the energy remaining after accounting for the energy expended to obtain the energy resource. In other words, it is the energy available to society and it must be greater than zero for a system to be viable. A net energy approach is applied that takes into account the energy return on energy invested (*EROI*) of the individual technologies and for the entire system.

$$EROI = \frac{E_{returned}}{E_{invested}} \tag{5.1}$$

$$Net\ energy = E_{returned} \cdot \left(1 - \frac{1}{EROI}\right) \tag{5.2}$$

MEDEAS dynamically and endogenously computes the EROI from a standard approach ($EROI_{st}$) of the full energy system. Therefore, all the necessary overcapacity, storage, and networks are allocated to the entire energy system, rather than to a specific technology.

The dynamic $EROI_{st}$ of the system is defined as the ratio of the final energy delivered to society (1) and two factors: the energy required to build, operate, maintain, and dispose of the energy generation plant (2); and the energy required to manage the intermittency of RES (3) [27].

$$EROI_{st}^{system} = \frac{(1)}{(2) + (3)} \tag{5.3}$$
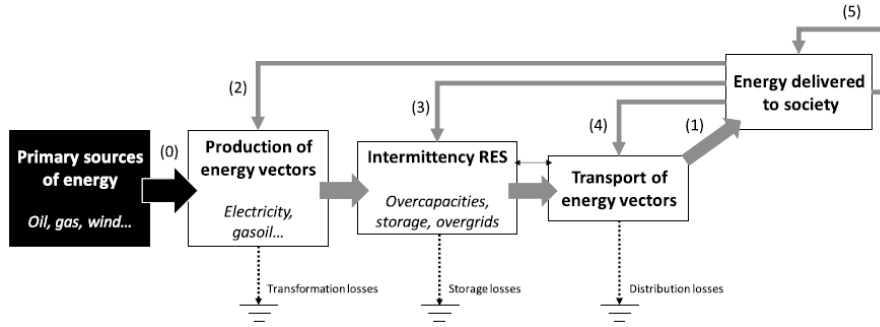
Fig. 5.2. Representation of society's energy metabolism [27]

To calculate the $EROI_{st}^{system}$, the following assumptions are taken into account [27]:

- $EROI_{st}$ for non-renewable energy sources is assumed constant over time.

- $EROI_{st}$ is dynamically estimated for renewable technologies for electricity generation

- Allocating technologies based on their relative EROIst buffered with energy investments to manage intermittency (RES technologies with higher EROI typically cover a greater share of energy demand)

- Overcapacities and overgrids related to the increasing penetration of variable RES technologies in the system are endogenously computed in the model.

- Additional losses due to the use of storage are modelled

42

5.5. Modelling of variability of renewable technologies

The intermittency of RES is considered in the model framework, introducing a certain level of flexibility as a function of the variable RES penetration. This is achieved by computing endogenous levels of new power grids, storage and overcapacities.

5.5.1. Grid development

The additional grids needed to integrate variable renewable electricity generation are estimated per MW of variable RES. The additional material requirements related to grid developments are then calculated, which ultimately affect the system's EROI.

5.5.2. Storage

Pumped Hydro Storage (PHS) is the main electrical storage technology in the MEDEAS model. The storage requirements are estimated using the study from [29].

Figure 5.3 shows the exponential fit to [29] data to estimate the electric storage capacity demand associated with different levels of RES variables. However, according to the literature review, a 20-35% share of variable RES may require low levels of storage and overcapacity. Consequently, the exponential fit is adjusted so that it passes through the point (20%; 0) by subtracting the current installed PHS.
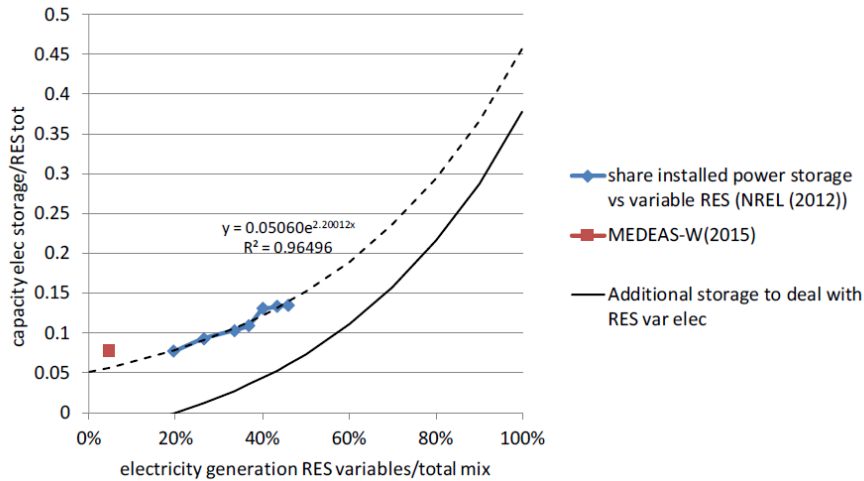

Fig. 5.3. Ratio of capacity electricity storage and total capacity of RES as a function of the electricity generation from RES variables in the total mix [27]

5.5.3. Overcapacities of dispatchable and variable RES power plants

A literature review of studies analysing the implications of RES intermittency for system overcapacity reaches one conclusion: an increasing level of overcapacity of both dispatchable and variable RES is required as the latter increase their generation share in the electricity sector. The approach followed estimates the reduction in Capacity Factor (CF i.e., ratio of the actual electrical energy produced over a period of time to the maximum electrical energy that could be produced over that period [27]) of RES plants depending on the penetration of variable RES.

Dispatchable RES power plants

The overcapacity of dispatchable RES is estimated using the study from [29]. The polynomial curve in Figure 5.4 illustrates the reduction in the CF of dispatchable RES as a function of the penetration of variable RES. The model applies the same reduction factor to both RES baseload and nuclear plants for the sake of simplicity.
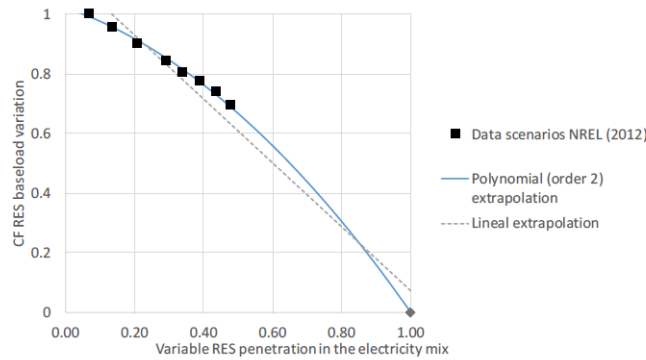
Fig. 5.4. Capacity Factor reduction of baseload power plants as a
function of variable RES penetration [27]

Variable RES power plants

The overcapacity of variable RES is estimated using the study from [7]. In this case, two main effects must be considered depending on the penetration of renewables in the electricity mix: the exponential growth of the overcapacities of RES variables and their reduction of the capacity factor.

Figure 5.5 shows the two curves introduced to the model. The first one (a) estimates the overcapacity curve based on [7] data as an exponential fit and extrapolate it until 100% variables RES penetration. The capacity factor reduction curve (b) is then calculated as a function of overcapacity, assuming that $CF = \frac{1}{1+overcapacity}$.
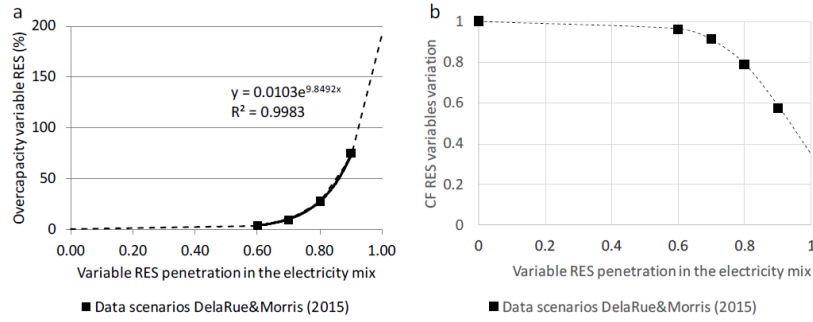
Fig. 5.5. (a) Overcapacities of RES variables (b) Capacity Factor reduction of RES power plants [27] as a function of variable RES penetration

5.6. Comparison with the ANN model

This section compares the model built from ANN in Chapter 4 with the current one integrated in MEDEAS. Figure 5.6 shows the regression plots of both approaches.

Figure (a) shows the model used in MEDEAS, as explained in subsection 5.5.3. Based on Delarue's study [7], overcapacity (i.e. curtailment) is modeled as a function of variable RES penetration in the electricity mix.

To compare the models, graph (b) is generated using the neural network model. For this purpose, the share of RES is varied at different pv/wind ratios and curtailment is then calculated. Three curves are plotted by changing the *Share wind* value of the ANN model. The blue curve is generated by setting the *Share wind* variable to 0.06 and varying the *Share pv* variable. This results in a number of different wind and pv ratios, whose sum corresponds to the Share res value. For the red and green curves, *Share wind* values of 0.28 and 0.44 are used. The dots represent the simulation value and the curve represents the ANN regression model.
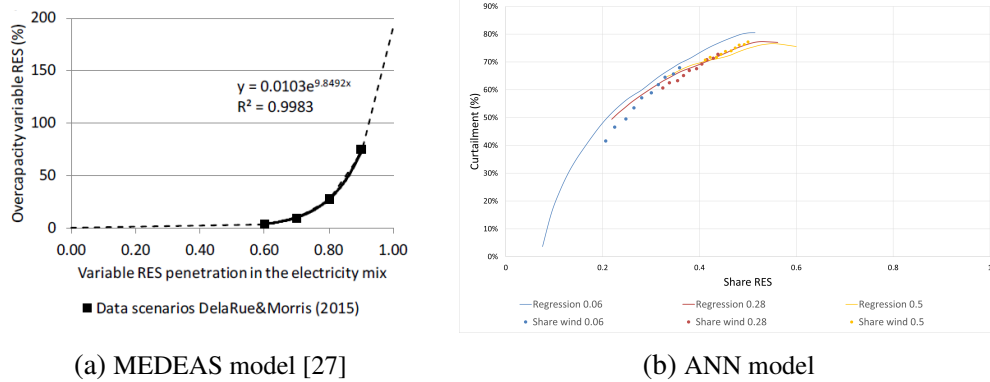
(a) MEDEAS model [27]   (b) ANN model

Fig. 5.6. Curtailment vs variable RES penetration. Curtailment (%) is calculated as Energy curtailed [TWh] / Energy produced by RES [TWh]

MEDEAS does not currently compute energy not served as a function of Share RES. However, this could be modelled with ANN regression and introduced into the model as shown in Figure 5.7.
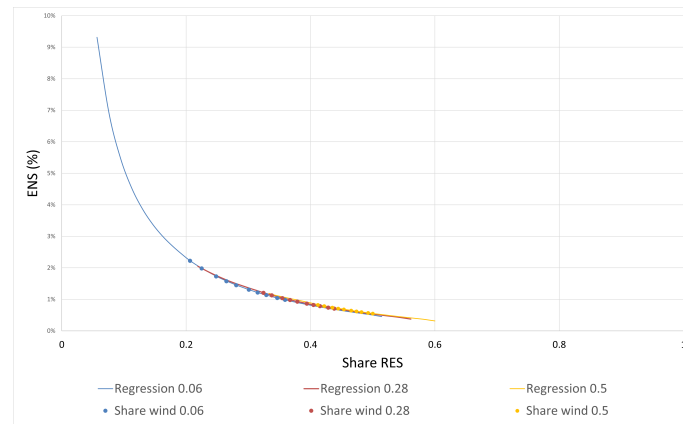
Fig. 5.7. ENS vs variable RES penetration. ENS (%) is calculated as
Energy not served [TWh] / Energy produced by RES [TWh]

The following reasons can be stated as to why the ANN regression model for Curtailment is an improvement over the MEDEAS model:

- The new model has more inputs to compute the curtailment.

- The ANN model considers an interconnected system (European network), whereas Delarue's study only considers Belgium.

- Regression ANN values are more reliable since they are based on more simulations.

6. CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

In this thesis, a new technique for coupling models based on the creation of a surrogate model was performed. The main goal was to integrate flexibility constraints into a system dynamics model (MEDEAS) by using the aforementioned technique. The Dispa-SET model was used for the analysis of VRES impacts on the power system based on a high temporal resolution representation, with several flexibility options and constraints. Dispa-SET simulations were run on a set of samples, selected by the Latin hypercube technique, and then used to train machine learning methods based on ANN. Curtailment and Energy not served, the two main flexibility indicators, were predicted using the surrogate model as a function of the system features. Thus, the generated model can approximate the behaviour of the complex simulations of Dispa-SET and be integrated into the MEDEAS model.

A dataset was developed for the creation of the surrogate model by varying the system parameters using the Dispa-SET model. 2718 simulations were run on the HPC cluster of ULiège to obtain more parameters of interest. System key parameters were defined to predict their values through a machine learning method. This was done in order to evaluate the flexibility of the energy systems. Furthermore, two artificial neural networks were built using the Tensorflow and Keras libraries. Three optimizers from the Keras Tuner library were used to find the optimal hyperparameters for each neural network. The architecture of these neural networks was defined once the optimal hyperparameters were found. Despite the slightly different architectures obtained by the three optimizers, all of them achieved good prediction accuracy. This suggested that some hyperparameters were more critical than others.

Three-dimensional plots were generated for the Curtailment and ENS parameters as a function of two input variables. Since there were no linearities in the results, neural networks proved to be an appropriate choice for regression, as they can approximate functions that are not linear.

Finally, the MEDEAS model was presented along with its modelling of RES intermittency and variability based on Delarue's work [7]. In the MEDEAS model, overcapacity was modelled as an exponential function of variable RES shares. Moreover, the surrogate model, i.e. the ANN regression model, also modelled overcapacity based on different RES shares. In this case, the pv/wind ratio could be adjusted to produce different curves, resulting in a more detailed model. In comparison to the previous model, the ANN model had more inputs, considered a whole interconnected system instead of a single country, and had a greater number of simulations, thus making its results more reliable.

Furthermore, the ENS parameter was modeled with ANN regression for future incorporation into the MEDEAS model as it is not currently computed.

Therefore, we can conclude that integrating the surrogate model into MEDEAS will improve the modelling of variable RES intermittency.

6.2. Future work

Further extensions to this work that may improve the method and results are presented below:

- The integration of the surrogate model into MEDEAS. This would imply a good knowledge of IAMs models as well as of the External Function Libraries [30] in order to create a function that integrates the ANN algorithm into MEDEAS.

- The addition of the Energy Not Served concept into MEDEAS, and the implementation of the corresponding surrogate model.

- Comparison of the surrogate model method with other coupling techniques.

- Integration of other technologies into the surrogate model such as pumped storage hydropower (HPHS) as storage options or Offshore wind turbines (WTOF) as renewable penetration.

- The use of MILP formulation for Dispa-SET model instead of the current (and faster) LP approach.

- Since more data leads to a better machine learning model, an increase in the number of simulations could improve the accuracy of the models.

BIBLIOGRAPHY

[1] S. Quoilin and C. Vidal, "Surrogate models to predict the adequacy and flexibility of large-scale power systems: Case-study with the eu power system," English, Vlorë, Albania, 2022.

[2] P.-H. Gonzalo *et al.*, "Capturing features of hourly-resolution energy models through statistical annual indicators," *Renewable Energy*, 2022. DOI: https://doi.org/10.1016/j.renene.2022.07.040. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148122010357.

[3] P. Helgesen, A. Lind, O. Ivanova, and A. Tomasgard, "Using a hybrid hard-linked model to analyze reduced climate gas emissions from transport," *Energy*, vol. 156, May 2018. DOI: 10.1016/j.energy.2018.05.005.

[4] B. Williams and S. Cremaschi, "Surrogate model selection for design space approximation and surrogatebased optimization," in *Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design*, ser. Computer Aided Chemical Engineering, S. G. Muñoz, C. D. Laird, and M. J. Realff, Eds., vol. 47, Elsevier, 2019, pp. 353–358. DOI: https://doi.org/10.1016/B978-0-12-818597-1.50056-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128185971500564.

[5] "Introduction to system integration of renewables," IEA, Paris, 2020. [Online]. Available: https://www.iea.org/reports/introduction-to-system-integration-of-renewables.

[6] "Power system flexibility for the energy transition, part 1: Overview for policy makers," International Renewable Energy Agency, Abu Dhabi, 2018. [Online]. Available: https://www.irena.org/publications/2018/Nov/Power-system-flexibility-for-the-energy-transition.

[7] E. Delarue and J. Morris, "Renewables intermittency: Operational limits and implications for long-term energy system models," MIT Joint Program on the Science and Policy of Global Change, 2015. [Online]. Available: http://hdl.handle.net/1721.1/95762.

[8] E. Kaushik *et al.*, "Comprehensive overview of power system flexibility during the scenario of high penetration of renewable energy in utility grid," *Energies*, vol. 15, no. 2, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/2/516.

[9] M. N. Thombre, H. A. Preisig, and M. B. Addis, "Developing surrogate models via computer based experiments," in *12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process*

Engineering, ser. Computer Aided Chemical Engineering, K. V. Gernaey, J. K. Huusom, and R. Gani, Eds., vol. 37, Elsevier, 2015, pp. 641–646. DOI: `https://doi.org/10.1016/B978-0-444-63578-5.50102-X`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/B978044463578550102X`.

[10] S. Quoilin, I. Hidalgo Gonzalez, and A. Zucker, "Modelling future eu power systems under high shares of renewables: The dispa set 2.1 open source model," *JRC TECHNICAL REPORTS*, 2017.

[11] G. D. Corporation, *General algebraic modeling system (gams) release 24.5.6*, 2015. [Online]. Available: `https://www.gams.com/`.

[12] I. Capellán-Pérez, C. de Castro, and I. Arto, "Assessing vulnerabilities and limits in the transition to renewable energies: Land requirements under 100% solar energy scenarios," *Renewable and Sustainable Energy Reviews*, vol. 77, pp. 760–782, 2017. DOI: `https://doi.org/10.1016/j.rser.2017.03.137`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1364032117304720`.

[13] *Opening remarks by executive vice-president timmermans and commissioner simson at the press conference on the repowereu plan*, Brussels. [Online]. Available: `https://ec.europa.eu/commission/presscorner/detail/en/SPEECH_22_3184`.

[14] V. Pavlovsky. "Introduction to artificial neural networks." (2013), [Online]. Available: `https://www.vojtech.net/posts/introduction-artificial-neural-networks`.

[15] R. M. S. de Oliveira *et al.*, "A system based on artificial neural networks for automatic classification of hydro-generator stator windings partial discharges," *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, DOI: `https://doi.org/10.1590/2179-10742017v16i3854`. [Online]. Available: `https://www.scielo.br/j/jmoea/a/M7Cc4KCtC7KMtkxWVH55BVL/?lang=en#`.

[16] "Gradient descent." (2020), [Online]. Available: `https://www.ibm.com/cloud/learn/gradient-descent`.

[17] "Optimization in deep learning: Adagrad, rmsprop, adam." (2021), [Online]. Available: `https://artemoppermann.com/optimization-in-deep-learning-adagrad-rmsprop-adam/`.

[18] B. Neal *et al.*, "A modern take on the bias-variance tradeoff in neural networks," *CoRR*, vol. abs/1810.08591, 2018. arXiv: `1810.08591`. [Online]. Available: `http://arxiv.org/abs/1810.08591`.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: `http://jmlr.org/papers/v15/srivastava14a.html`.

[20] D. Passos and P. Mishra, "A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks," *Chemometrics and Intelligent Laboratory Systems*, vol. 223, p. 104 520, 2022. DOI: https://doi.org/10.1016/j.chemolab.2022.104520. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743922000314.

[21] "Practical hyperparameter optimization." (2020), [Online]. Available: https://www.kdnuggets.com/2020/02/practical-hyperparameter-optimization.html.

[22] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 437–478. DOI: 10.1007/978-3-642-35289-8_26. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_26.

[23] "The essentials of machine learning data curation." (2020), [Online]. Available: https://www.artiba.org/blog/the-essentials-of-machine-learning-data-curation.

[24] T. O'Malley *et al.*, *Kerastuner*, https://github.com/keras-team/keras-tuner, 2019.

[25] J. Weyant, "Some contributions of integrated assessment models of global climate change," *Review of Environmental Economics and Policy*, vol. 11, no. 1, pp. 115–137, 2017. DOI: 10.1093/reep/rew018. [Online]. Available: https://doi.org/10.1093/reep/rew018.

[26] "Medeas model." (2022), [Online]. Available: https://www.medeas.eu/model/medeas-model.

[27] I. Capellán-Pérez, C. de Castro, and L. J. Miguel González, "Dynamic energy return on energy investment (eroi) and material requirements in scenarios of global transition to renewable energies," *Energy Strategy Reviews*, vol. 26, p. 100 399, 2019. DOI: https://doi.org/10.1016/j.esr.2019.100399. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2211467X19300926.

[28] C. A. Hall, J. G. Lambert, and S. B. Balogh, "Eroi of different fuels and the implications for society," *Energy Policy*, vol. 64, pp. 141–152, 2014. DOI: https://doi.org/10.1016/j.enpol.2013.05.049. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0301421513003856.

[29] NREL, "Renewable electricity futures study (entire report)," National Renewable Energy Laboratory, Golden, CO, USA, 2012.

[30] Vensim, *External function libraries*. [Online]. Available: https://www.vensim.com/documentation/25720.html.

[31] "Neural networks." (2020), [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks.

[32] M. Bernico, *Deep Learning Quick Reference: Useful hacks for training and optimizing deep neural networks with TensorFlow and Keras*. 2018.

[33] "How to use data scaling improve deep learning model stability and performance." (2020), [Online]. Available: https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/.

[34] S. Lek and Y. Park, "Multilayer perceptron," in *Encyclopedia of Ecology*, S. E. Jørgensen and B. D. Fath, Eds., Oxford: Academic Press, 2008, pp. 2455–2462. DOI: https://doi.org/10.1016/B978-008045405-4.00162-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080454054001622.

[35] P. Kumar, S. Batra, and B. Raman, "Deep neural network hyper-parameter tuning through twofold genetic approach," 2021. DOI: https://doi.org/10.1007/s00500-021-05770-w. [Online]. Available: https://link.springer.com/article/10.1007/s00500-021-05770-w#citeas.

[36] Z. Yang, Y. Yu, C. You, J. Steinhardt, and Y. Ma, "Rethinking bias-variance trade-off for generalization of neural networks," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 13–18 Jul 2020, pp. 10 767–10 777. [Online]. Available: https://proceedings.mlr.press/v119/yang20j.html.

[37] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018. [Online]. Available: http://jmlr.org/papers/v18/16-558.html.

[38] "The game of increasing r-squared in a regression model." (2021), [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/the-game-of-increasing-r-squared-in-a-regression-model/.

[39] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[40] A. Krook-Riekkola, C. Berg, E. O. Ahlgren, and P. Söderholm, "Challenges in top-down and bottom-up soft-linking: Lessons from linking a swedish energy system model with a cge model," *Energy*, vol. 141, pp. 803–817, 2017. DOI: https://doi.org/10.1016/j.energy.2017.09.107. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544217316274.

APPENDIX A

Name	Description
au	All Units
chp(u)	CHP units
f	Fuel types
h	Hours
hu(au)	Heat only units
i(h)	Time step in the current optimization horizon
l	Transmission lines between nodes
mk	Markets {DA: Day-Ahead, 2U: ReserveUp, 2D: ReserveDown, Flex: Flexibility}
n	Zones within each country (currently one zone, or node, per country)
n_th	Thermal nodes
n_h2	Hydrogen nodes
p	Pollutants
p2h(au)	Power to heat units
p2h2(au)	Power to hydrogen storage technologies
s(au)	Storage Units (including hydro reservoirs)
t	Power generation technologies
th(au)	Units with thermal storage
thms(au)	Thermal storage units only
tr(t)	Renewable generation technologies
u(au)	Generation units (all units except for P2HT units)
wat(au)	Hydro technologies
z(h)	Subset of every simulated hour

Table A1. Sets

Name	Units	Description
AvailabilityFactor(au,h)	%	Availability factor
CHPPowerLossFactor(u)	%	Power loss when generating heat
CHPPowerToHeat(u)	%	Nominal power-to-heat factor
CHPMaxHeat(chp)	MW/u	Maximum heat capacity of chp plant
CHPType		
CommittedInitial(au)	n.a.	Initial committment status
CostFixed(au)	EUR/h	Fixed costs
CostRampUp(u)	EUR/MW	Ramp-up costs
CostRampDown(u)	EUR/MW	Ramp-down costs
CostShutDown(u)	EUR/u	Shut-down costs
CostStartUp(u)	EUR/u	Start-up costs
CostVariable(au,h)	EUR/MW	Variable costs
CostHeatSlack(n_th,h)	EUR/MWh	Cost of supplying heat via other means
CostH2Slack(n_h2,h)	EUR/MWh	Cost of supplying H2 by other means
CostLoadShedding(n,h)	EUR/MWh	Cost of load shedding
Curtailment(n)	n.a	Curtailment allowed or not {1 0} at node n
CostCurtailment(n,h)	EUR/MWh	Cost of VRES curtailment
Demand(mk,n,h)	MW	Demand
Efficiency(au,h)	%	Efficiency
EmissionMaximum(n,p)	tP	Emission limit
EmissionRate(au,p)	tP/MWh	P emission rate
FlowMaximum(l,h)	MW	Line limits
FlowMinimum(l,h)	MW	Minimum flow
Fuel(u,f)	n.a.	Fuel type {1 0}
HeatDemand(n_th,h)	MWh/u	Heat demand profile for chp units
H2Demand(n_h2,h)	MW	H2 rigid demand
LineNode(l,n)	n.a.	Incidence matrix {-1 +1}
LoadShedding(n,h)	MW	Load shedding capacity
Location(au,n)	n.a.	Location {1 0}
Location_th(au,n_th)	n.a.	Location {1 0}
Location_h2(au,n_h2)	n.a.	Location {1 0}
Markup(u,h)	EUR/MW	Markup
MaxCapacityPtL(n_h2)	MW	Max capacity of PtL
Nunits(au)	n.a.	Number of units inside the cluster
OutageFactor(au,h)	%	Outage Factor (100 % = full outage)
PartLoadMin(au)	%	Minimum part load
PowerCapacity(au)	MW/u	Installed capacity
PowerInitial(u)	MW/u	Power output before initial period
PowerMinStable(au)	MW/u	Minimum power output
PriceTransmission(l,h)	EUR/MWh	Transmission price
PtLDemandInput(n_h2,h)	MWh	Demand of H2 for PtL at each timestep

RampDownMaximum(u)	MW/h/u	Ramp down limit
RampShutDownMaximum(au)	MW/h/u	Shut-down ramp limit
RampStartUpMaximum(au)	MW/h/u	Start-up ramp limit
RampUpMaximum(u)	MW/h/u	Ramp up limit
Reserve(au)	n.a.	Reserve technology {1 0}
StorageChargingCapacity(au)	MW/u	Storage capacity
StorageChargingEfficiency(au)	%	Charging efficiency
StorageSelfDischarge(au)	%/day	Self-discharge of the storage units
StorageCapacity(au)	MWh/u	Storage capacity
StorageDischargeEfficiency(au)	%	Discharge efficiency
StorageOutflow(au,h)	MW/u	Storage outflows
StorageInflow(au,h)	MW/u	Storage inflows (potential energy)
StorageInitial(au)	MWh	Storage level before initial period
StorageProfile(au,h)	%	Storage level at the end of each horizon
StorageMinimum(au)	MWh	Storage minimum
Technology(au,t)	n.a.	Technology type {1 0}
TimeDownMinimum(au)	h	Minimum down time
TimeUpMinimum(au)	h	Minimum up time
TimeStep	h	Duration of a timestep of optimization
VOLL	EUR/MWh	Value of Lost Load

Table A2. Parameters

Name	Units	Description
AccumulatedOverSupply(n,h)	MWh	Accumulated oversupply due to the flex demand
CostStartUpH(u,h)	EUR	Cost of starting up
CostShutDownH(u,h)	EUR	cost of shutting down
CostRampUpH(u,h)	EUR	Ramping cost
CostRampDownH(u,h)	EUR	Ramping cost
CurtailedPower(n,h)	MW	Curtailed power at node n
CurtailedHeat(n_th,h)	MW	Curtailed heat at node n_th
CurtailedH2(n_h2,h)	MW	Curtailed hydrogen at node n_h2
Flow(l,h)	MW	Flow through lines
Power(au,h)	MW	Power output
PowerConsumption(au,h)	MW	Power consumption by P2H units
PowerMaximum(u,h)	MW	Power output
PowerMinimum(u,h)	MW	Power output
ShedLoad(n,h)	MW	Shed load
StorageInput(au,h)	MWh	Charging input for storage units
StorageLevel(au,h)	MWh	Storage level of charge
LL_MaxPower(n,h)	MW	Deficit in terms of maximum powerv
LL_RampUp(u,h)	MW	Deficit in terms of ramping up for each plant
LL_RampDown(u,h)	MW	Deficit in terms of ramping down
LL_MinPower(n,h)	MW	Power exceeding the demand
LL_2U(n,h)	MW	Deficit in reserve up
LL_3U(n,h)	MW	Deficit in reserve up - non spinning
LL_2D(n,h)	MW	Deficit in reserve down
spillage(au,h)	MWh	spillage from water reservoirs
SystemCost(h)	EUR	Hourly system cost
Reserve_2U(au,h)	MW	Spinning reserve up
Reserve_2D(au,h)	MW	Spinning reserve down
Reserve_3U(au,h)	MW	Non spinning quick start reserve up
Heat(au,h)	MW	Heat output by chp plant
HeatSlack(n_th,h)	MW	Heat satisfied by other sources
H2Slack(n_h2,h)	MW	H2 demand satisfied by other sources
WaterSlack(au)	MWh	Unsatisfied water level constraint
StorageSlack(au,h)	MWh	Unsatisfied storage level constraint
H2Output(au,h)	MWh	H2 output from H2 storage to fulfill demand
PtLDemand(n_h2,h)	MW	Demand of H2 for PtL for each n_h2 node

Table A3. Variables

Name	Units	Description
Committed(au,h)	n.a.	Unit committed at hour h {1 0} or integer
StartUp(au,h)	n.a.	Unit start up at hour h {1 0} or integer
ShutDown(au,h)	n.a.	Unit shut down at hour h {1 0} or integer

Table A4. Integer Variables