Master's Degree Thesis

Towards Sustainable AI

Monitoring and Analysis of Carbon Emissions in Machine Learning Algorithms

Supervisors

Prof. MICHELA MEO

Co-Supervisors

Prof. GRETA VALLERO

Candidate

AURORA MARTINY

December 2023

Summary

The integration of carbon emissions as a metric in machine learning is a relatively new concept. Nowadays, a predominant focus in research lies in achieving high-performance levels without taking computational efficiency into account. This neglect could be attributed to the lack of familiarity with existing approaches to evaluate energy consumption in this domain.
This thesis delves into the realm of sustainable artificial intelligence, with a specific focus on deep learning algorithms. The primary goal is to identify key factors contributing to environmental challenges within this context. Employing an empirical approach, the study investigates the emissions patterns of various algorithms when applied to different real-time-series datasets. Furthermore, the study explores how manipulating training hyperparameters, model architecture design, and problem formulation can impact energy consumption without severely compromising performance. Using a Python library to assess carbon emissions and leveraging models designed for time-series data, such as Long Short Term Memory, alternative configurations are proposed within the specific case study.
Overall, this research aims to provide insights into utilizing different hyperparameters and configurations in deep learning algorithms to foster a more environmentally conscious artificial intelligence ecosystem.

Table of Contents

List of Tables

List of Figures

Acronyms

AI	Artificial Intelligence
BS	Base Station
CNN	Convolutional Neural Network
CO_2	Carbon Dioxide
DL	Deep Learning
GHG	Greenhouse Gas Emissions
GPU	Graphics Processing Units
IT	Information Technology
LSTM	Long Short-Term Memory

MAE	Mean Absolute Error
ML	Machine Learning
NLP	Natural Language Processing
NREL	National Renewable Energy Laboratory
NN	Neural Networks
RNN	Recurrent Neural Networks

Chapter 1

Introduction

The rationale behind assessing and mitigating the environmental impact of Artificial Intelligence (AI) systems arises from the exponential growth in their usage. In fact, the rapid and expansive growth of AI has led to big changes across various sectors, reshaping how we live and work. The term “AI” encompasses a spectrum of technologies designed to simulate human intelligence, enabling systems to learn, analyze, and adapt to complex tasks. The recent explosion of AI is indeed largely attributed to advancements in technology and the unprecedented availability of data [1]. The exponential growth in computing power, particularly with the advent of GPUs (Graphics Processing Units) and specialized hardware for AI tasks, has significantly accelerated the training and deployment of complex AI models [2], [3]. Moreover, the proliferation of digital data in various forms, including text, images, videos, and sensor data, has provided the fuel necessary for training AI algorithms. This abundance of labeled datasets has been also encouraged by the rise of data-sharing platforms [4].
However, this accelerated development in AI technologies has cast a spotlight on energy consumption and its environmental implications. The necessity to craft AI systems that not only offer remarkable performance but also curtail their carbon footprint has never been more crucial. Achieving this demands a thorough exploration into energy-efficient Machine Learning (ML) algorithms [5]. This exploration forms the basis of this thesis, which aims to advance sustainability in AI through monitoring and analyzing carbon emissions from ML algorithms.

ML is closely related to the broader field of AI due to their overlapping capabilities such as learning and decision making. In early research on computational intelligence, ML was born as a subset within the domain of AI in the late 1970s [6]. At this stage, the ML goal of developing systems that could learn and make judgments in an automated fashion was seen as one of the main objectives under

the AI umbrella.
However, as ML progressed through empirical studies on algorithms derived from data rather than symbolic representation, it started to differentiate and specialize compared to logic-based AI approaches [7]. By the 1980s, ML began to establish itself as a distinct discipline focusing on creating systems that use statistical inference to learn from examples, as opposed to expert systems designed using logic-based rule sets [8]. Nowadays, ML has evolved into a distinct subject with its theoretical foundations and tools, although it remains tied to AI through shared automation objectives. ML represents how subfields can emerge within a broader research domain and mature at different paces. ML is universally defined as a broad set of algorithms and statistical techniques that enable computer systems to automatically improve tasks through experience, without needing to be explicitly programmed [9]. At a high level, ML algorithms build mathematical models upon exposing large amounts of data. ML's data-centric paradigm now complements symbol-based AI methods with the overarching goal of developing intelligent systems.

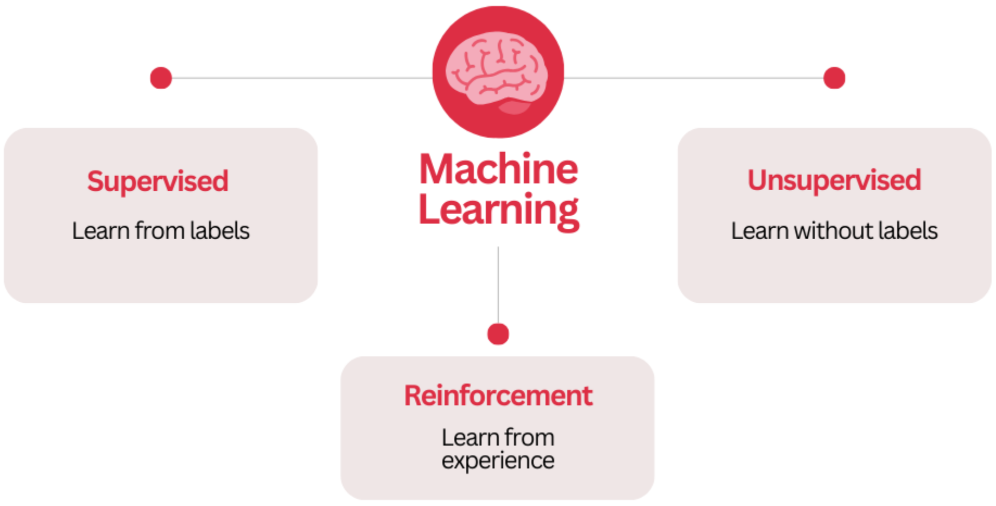


Figure 1.1: Machine Learning Types.

ML, the field centered around enabling systems to learn and improve from experience or data ([1, 3]), encompasses diverse techniques leading to different learning paradigms, as shown in Fig 1.1. Among all, supervised learning involves training models on labeled datasets, where each input data is associated not only with features but also with a corresponding output or target label [10]. In this way, the model learns to map inputs to outputs based on the provided examples [1]. This learning paradigm is of two main types:

- Classification: In classification tasks, the model learns to assign inputs to predefined categories or classes [3]. For instance, in an image classification task, the model is trained on images labeled with specific objects or classes (like "cat" or "dog"). During the training phase, it learns how to identify and classify new, unseen images into these categories based on learned patterns from the labeled data.

- Regression: Regression tasks involve predicting a continuous numerical value or quantity based on input features, also called predictors [1], as shown in Fig. 1.2. For example, in predicting house prices, the model learns from historical housing data with attributes such as square footage, number of bedrooms, and location to estimate the selling price of a house [11].

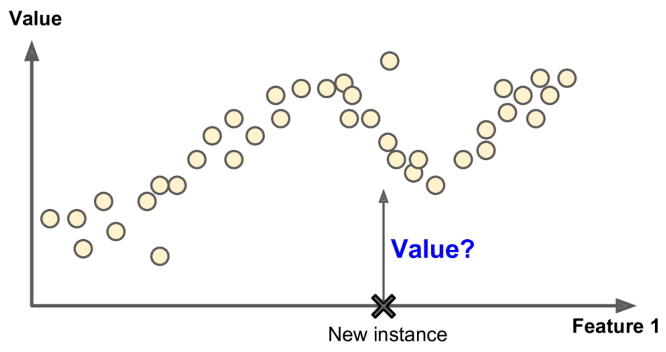


Figure 1.2: Prediction of a target numeric value based on a single feature [9].

Unsupervised learning, on the other hand, delves into uncovering hidden structures or patterns within unlabeled data [12]. Thus, the algorithm has to extract the meaning of data without the labels' help [1]. Key techniques in unsupervised learning include:

- Clustering: Clustering algorithms group similar data points together based on their features, but without prior knowledge of categories [9]. These algorithms identify clusters or groups within the data, where data points within the same cluster are more similar to each other than to those in other clusters. For instance, the k-means algorithm defines the similarity based on the distance between points [1].

Reinforcement learning introduces a dynamic aspect, where agents (i.e. the learning methods) learn optimal decision-making through interactions with their environment [9]. These agents improve their policies by navigating through trial-and-error experiences, aiming to maximize rewards in a given environment [13]. This paradigm finds application in various domains, from robotics to game-playing algorithms like AlphaGo, where the system learns to make strategic moves through repeated gameplay [3].

Representation learning emerges as a pivotal subset of methods within ML, allowing systems to automatically extract meaningful features from raw data [14]. These features are essential in tasks such as classification or detection. Overall, these diverse techniques address distinct aspects of learning: supervised for labeled data, unsupervised for hidden patterns, reinforcement for dynamic decision-making, and representation for feature extraction.

Deep Learning (DL), a subset of ML, stands out as a pivotal technology driving the evolution of AI. It harnesses artificial neural networks with multiple layers, allowing for representation learning through hierarchies of abstraction [14]. These deep neural networks can progressively derive higher-level representations by composing simple transformations on lower-level inputs [15]. During the years, the development of robust programming platforms such as TensorFlow and PyTorch and the advancements in hardware capabilities have facilitated the creation of more sophisticated deep models [16]. A recent study [2] examines how DL has been more prevalent in papers presented at the prestigious AI conference ACL[1] between 2010 and 2020: these days, deep networks serve as the backbone for all publications.

Nowadays, model training requires significant computational time and power. This is due to the increasing models' complexity, given by factors such as the number of parameters. In fact, the easiest technique to get better performances is to increase the size of the model when there is access to large-scale data sets [2]. As the training duration grows, the required computational power increases accordingly [17]. Additionally, if the model performs continuous learning, the computing cost may increase even further [18]. In this context, the realm of ML has largely centered around the pursuit of highly accurate models without giving due consideration to energy consumption as a significant factor [19]. Nevertheless, ML algorithms require vast amounts of computational resources for model training on large datasets. This data-intensive paradigm of learning also means that ML models demand high power both for training iteratively over vast datasets as well as inference during deployment at scale [20].

In classical computer science fields, algorithmic progress is rigorously tracked based on asymptotic analysis of cost scaling with problem size. For example, quicksort [21] has clearly more efficient $O(n \log n)$ runtime compared to $O(n^2)$ sorting algorithms. However, DL tasks such as seeking approximate solutions, defining problem difficulty and assessing progress pose unique challenges compared to optimal problems. Improvements are reported primarily in accuracy metrics rather than cost-scaling,

[1]https://2023.aclweb.org/

ignoring the computational expenses required to attain new state-of-the-art results [22]. In DL the primary objective has been to develop more profound and precise models without computational constraints [23]. These models entail substantial computing requirements (typically in GigaFlops) and demand extensive memory (often in millions of parameters or weights) [17]. During the training phase, these algorithms necessitate substantial computing power as they grapple with large volumes of data. Recent findings [24, 25] have highlighted that large language models alone, such as GPT-3 [26], can account for emissions totaling more than 500 tonnes of CO_2eq – that represents a power consumption of almost 1,300 MWh of electricity – for the entire process (including equipment manufacturing and energy-based operational consumption). Such models are further utilized multiple times during deployment.
These results are directly correlated with energy consumption, and thus, with the production of electricity. Since electricity generation currently stands as the primary sector emitting fossil fuel CO_2 [27], it is essential to incorporate metrics that measure the energy consumption of algorithms alongside existing benchmarks. Already in 2010, approximately 35% of human-caused greenhouse gas emissions (GHG) originated from energy production [28]. Only by evaluating model performance jointly with computational efficiency, the DL field can hope to align developmental priorities with constraints imposed by hardware limits and climate change realities.
Thus, incorporating resource utilization into evaluations of DL performance is an essential step in moving the discipline away from an emphasis only on performance metrics. This aligns with the classical concept of algorithmic efficiency, which prioritizes optimal performance in both time and space complexities when solving computational problems. Approaching DL development with the same considerations of efficiency rather than isolated improvements in accuracy will encourage the design of algorithms that achieve state-of-the-art results while minimizing the expenditure of computational resources like energy and memory [29]. Defining efficiency metrics for learning algorithms based on their joint time, space and energy scaling would help steer progress toward more sustainable ML techniques capable of both high-quality outputs and efficient usage of processing capabilities. This in turn is vital for maintaining the momentum of the field within the constraints of finite hardware availability and the need to avoid environmental damages from rapidly growing computational demands.

For these reasons, the concept of sustainable AI has emerged. It refers to the development and application of AI systems that are environmentally conscious, economically viable, and socially beneficial. It represents a concerted effort to

instigate transformation across every stage of the life cycle of AI products. This encompasses the inception of ideas, the training phase, adjustments, implementation, and overarching governance, all geared towards enhancing ecological sustainability and promoting social equity. Sustainable AI goes beyond the scope of AI applications; rather, it encapsulates the entire socio-technical framework of AI [30].
Energy-efficient AI, an integral component of sustainable AI, focuses on reducing the energy consumption and carbon footprint of AI systems and infrastructure. Recognizing the environmental implications of AI technologies, there exists a growing necessity to evaluate and comprehend the energy consumption and associated carbon emissions of these systems.
Fischer et al. [31] proposed an approach to assess the efficiency of any ML experiment by considering it as a composite entity comprising a configuration and environment. The configuration involves the specifics related to the task at hand, encompassing aspects such as the type of task (inference, training, robustness testing), the dataset employed, the model used, and all associated hyperparameters. On the other hand, the environment pertains to the hardware and software utilized during the execution of the experiment. Patterson et al. [26] suggested

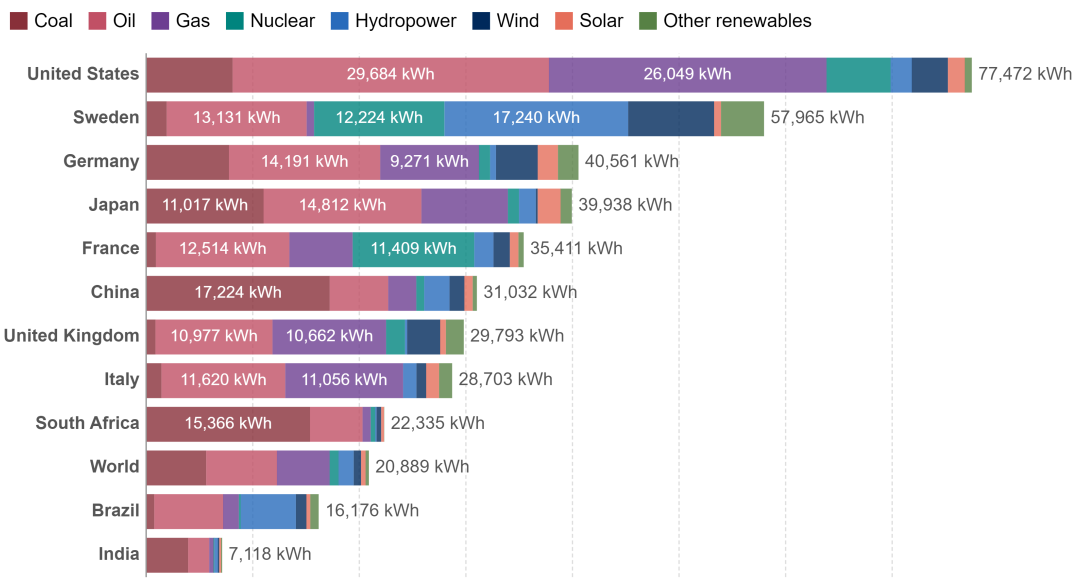


Figure 1.3: Overview of the energy mix by country in 2022 [32].

that accounting for carbon emissions within this framework could be achieved by incorporating the local energy mix as a part of the environment. The concept of an energy mix delineates the distribution of available production from energy resources to fulfill the energy requirements within a specific geographic area [32], as depicted in Fig. 1.3. Primary energy source encompasses fossil fuels such as oil, natural gas,

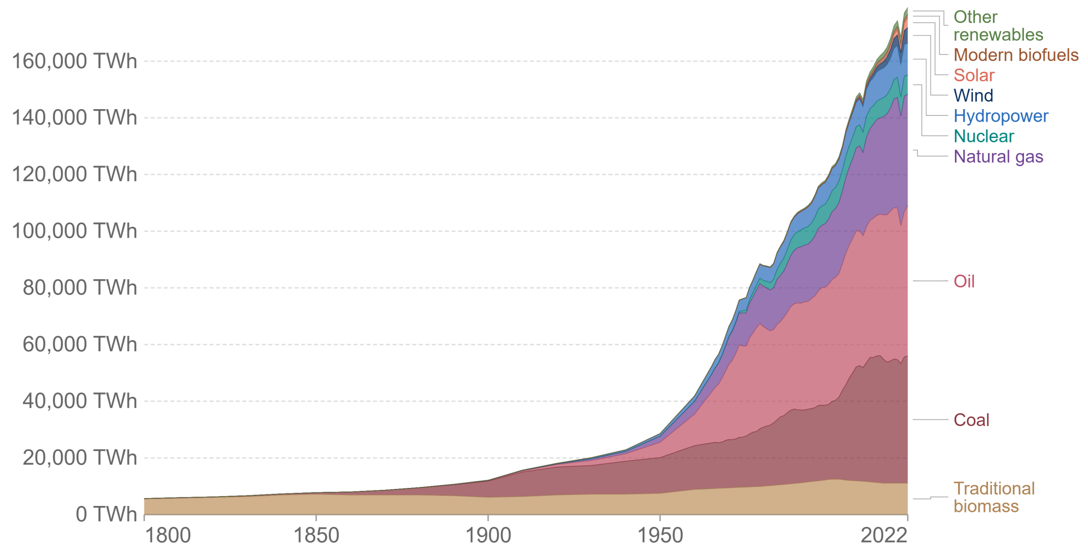


Figure 1.4: Primary energy use worldwide starting in 1800 [32].

and coal, alongside nuclear energy, waste utilization, and a diverse array of renewable energy sources, including biomass, wind, geothermal, water, and solar power. Fig. 1.4 shows the global primary energy use from 1800 to 2022. This adjustment recognizes the impact of the energy sources used during the experiment execution on carbon emissions. The properties utilized to gauge the efficiency of a task are termed metrics, such as accuracy, model size, and power draw [33]. These metrics are specific to the experiment configuration. By conducting experiments with varying configurations under a fixed task and environment, researchers can compare their efficiency. Additionally, exploring how a particular model performs across different environments can also offer insights into its adaptability and efficiency. However, it is important to note that certain configurations and environments might not be feasible due to practical constraints. For instance, the choice of a dataset might impose the usage only of certain models, and, at the same time, specific models could necessitate particular software or hardware for execution.
This integration of the local energy mix within the environmental parameters acknowledges its influence on the overall environmental footprint of the experiment.

1.1 Research Objectives and Novel Contributions

Since the computational demands of AI algorithms, particularly ML models, contribute significantly to energy consumption and subsequently to carbon emissions, formulating strategies to minimize environmental harm is essential. This thesis delves into the realm of sustainable AI, particularly focusing on the crucial aspect of monitoring and analyzing the carbon emissions produced by specific ML algorithms.

The overarching goal of this thesis is to analyze a critical domain within the realm of ML model training, specifically focusing on time-series datasets derived from network data. Understanding how parameters impact carbon emissions during both the training phases and potentially in the subsequent inference phase is crucial. Obtaining a comprehensive overview of the parameters to monitor serves as an example of initiating such an analysis in a similar context.
The novel contributions of this study are the demonstration of the following statements:

- There usually exists a compromise between carbon emissions and the performance of a model.
- The first step in evaluating carbon emissions should be to use already available open-source resources.
- Specific training hyperparameters should be considered when the analyses focus on energy efficiency.
- Adjustments in a model's architecture can impact its carbon emissions production during both training and test phases.
- Key considerations about the possible consumption should be made when initially formulating the problem.
- Each setting can have a different impact in terms of emissions during the training and inference phase.

By delving into these questions, this study aims to shed light on the factors influencing carbon emissions mainly during the training of ML models, offering insights into the manipulation of model architecture for reduced energy consumption. Moreover, it endeavors to scrutinize the implications of the initial problem setup. This exploration lays the foundation for understanding and potentially mitigating the environmental impact of ML processes on a broader scale.

1.2 Structure of the Thesis

The thesis is organized to provide a comprehensive analysis of the carbon emissions and environmental impact of DL models.
The Literature Review, delves into existing research, examining the environmental repercussions of DL. Particularly, the focus is on specific models such as Recurrent Neural Networks (RNNs), with a detailed exploration of Long Short-Term Memory

(LSTM). Additionally, this section critically evaluates current approaches geared towards enhancing energy efficiency in these models.
Following this, the study narrows its focus to the Problem Statement and Dataset Description, where the specific problem addressed is detailed. The thesis employs two primary datasets for analysis: the traffic data sourced from Italian Mobile Network Operators and the PVWatts energy estimate data in Turin, providing the empirical foundation for the research.
The Methodology section lays out the approach employed to analyze the environmental impact. It encompasses various strategies, such as CO_2 monitoring, manipulation of training parameters (including Epochs and Nodes Ablation), architectural model design through Layers Ablation, and a meticulous process for data aggregation.
The subsequent segment, Experimental Results, details the environment used for experimentation and offers a comprehensive presentation of the findings derived from the conducted analyses.
Finally, the Conclusions and Future Works section synthesizes the insights obtained from the research, underlining their implications and potential directions for future research in the field of energy efficiency in DL models.

This structured approach enables a systematic exploration and detailed investigation into the environmental impact and carbon emissions within the realm of DL models, aiming to contribute substantially to the discourse on sustainable AI and computational efficiency.

Chapter 2

Literature Review

The field of AI research relies on a dynamic and evolving landscape, marked by significant progress in recent years and a wide range of applications across various domains. Complex and large-scale DL models have played a crucial role in this growth. However, this trend is driven by the strong focus of the AI community on obtaining a system's accuracy – or similar metrics – as great as possible, often at the expense of energy efficiency considerations [19]. As AI systems become integrated into our daily lives and industrial processes, parallel concerns about the economic, environmental and social costs have gained importance. In particular, despite the undeniable potential of AI, there is a growing realization that the energy consumption and carbon footprint associated with ML algorithms represent challenges to a sustainable future [20] [17]. In fact, these improvements in accuracy are conditioned upon the availability of huge computational power, leading to an equivalent energy consumption. Hence, these models are costly during both the training and development phases. Financially, this comes from hardware costs and the electricity or cloud computing time needed. Environmentally, the carbon footprint associated with powering modern tensor processing hardware adds to the overall expense [5].

This literature review follows an extensive analysis of the critical interaction between AI and environmental sustainability. It begins with an outline of the broader AI usage, which contributes to several sectors, such as medical, financial and transportation. Specifically, the focus is on AI research that relies on DL methods. Subsequently, the focus narrows to the energy consumption patterns of these algorithms, thereby providing the starting point for understanding the core issues addressed in this thesis. By evaluating existing methodologies, this review aims to provide a holistic understanding of the interplay between DL and energy consumption, setting the stage for a deeper investigation into efficient solutions.

2.1 Carbon Emissions and Global Warming

Over the past 50 years, the demand for resources and energy has incredibly increased due to the growing global population, which has resulted in a significant rise in carbon emissions [34], as shown in Fig. 2.1. Since 2000, the amount of carbon dioxide (CO_2) in the atmosphere has increased globally by around 20 parts per million (ppm) per decade, which is up to ten times faster than the average rate of continuous CO_2 increase during the previous 800,000 years [35]. This exponential acceleration underscores the unprecedented impact of recent human activities on our planet's carbon balance and climate dynamics. Thus, it is evident that keeping global warming to no more than a 1.5°C rise above pre-industrial levels necessitates collective efforts [36]. In particular, the response to this global challenge materialized in the form of the Paris Agreement, a pivotal framework outlined under the United Nations Framework Convention on Climate Change[1] (UNFCCC) in 2015 [37].

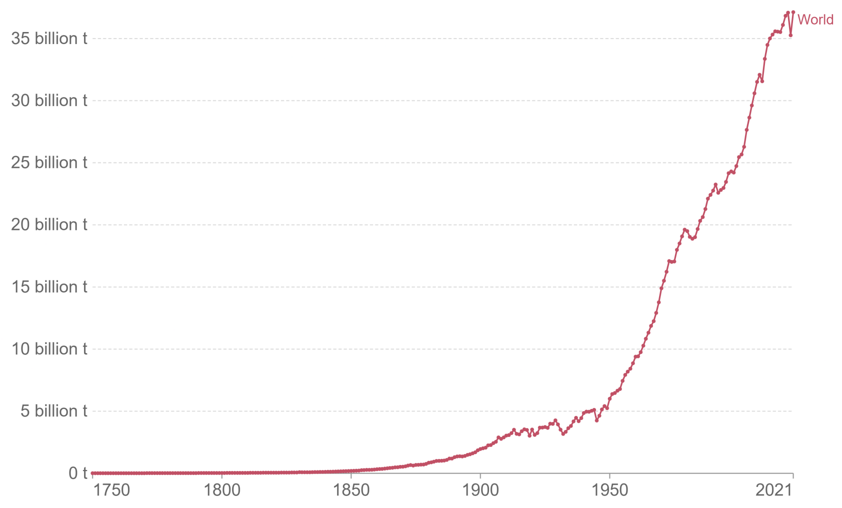


Figure 2.1: The increase in worldwide emissions from the middle of the 18th century to 2021 [34].

The burning of non-renewable fossil fuels – coal, oil, and natural gas – remains a primary contributor to the escalating levels of CO_2 in the Earth's atmosphere. This surge in emissions stands as a consequence of unsustainable human activities that perpetuate the combustion of finite resources formed over millions of years, unable to replenish at the rate of consumption [38]. In addition, widespread deforestation

[1]`https://unfccc.int/`

has further exacerbated the release of carbon emissions, disrupting the delicate balance of the planet's ecosystems. The repercussions of these actions are profound, as the surge in GHG has intensified the process of global warming, triggering a cascade of environmental changes that pose a significant threat to our planet's future. The impact is twofold: a surge in emissions directly linked to energy consumption and the collateral damage from depleting natural ecosystems, both of which pose dire threats to the planet's future.

To comprehend the direct influence of human actions on the entire ecosystem, it is crucial to begin by understanding one of the most widely recognized consequences: global warming. When solar radiation reaches the Earth's surface, some of it is absorbed and re-emitted as infrared radiation. GHG, including CO_2, methane (CH_4), and nitrous oxide (N_2O), trap this heat within the atmosphere, preventing it from escaping into space. This process maintains the Earth's habitable temperature, allowing life to thrive. However, the excessive accumulation of GHG, primarily derived from human activities, has intensified the greenhouse effect, resulting in a rapid increase in global temperatures. Thus, the correlation between carbon emissions and global warming is inseparably tied to the GHG effect, the natural mechanism regulating the Earth's temperature. Undeniably, carbon dioxide emissions are the foremost driver of global climate change [34].

The surge in carbon emissions has disrupted the delicate balance of the Earth's climate systems, leading to a myriad of consequences, not only global warming. These include a surge in the frequency and intensity of extreme weather events such as hurricanes, droughts, and heat waves, as confirmed also by the European Commission[2]. Moreover, the rise in global temperatures has accelerated the melting of polar ice caps and glaciers, contributing to the alarming rise in sea levels. All these shifts in weather patterns have been led by the disruption of the delicate equilibrium of ecosystems and this, in turn, is causing the extinction of numerous plant and animal species. The repercussions of global warming and associated weather changes are far-reaching, affecting not only the environment but also human health, agriculture, and the economy. Recent records from the National Center for Environmental Information in 2023 (as of November 8) underscore the staggering toll of 25 confirmed weather and climate disaster events in the United States, each causing losses surpassing $1 billion [39]. Moreover, the impact of global warming transcends geographical boundaries, affecting both developed and developing nations. The disproportionate burden of climate change is often borne by marginalized communities and vulnerable populations, exacerbating social inequalities and economic disparities. In regions prone to extreme weather events,

[2]`https://climate.ec.europa.eu/climate-change/consequences-climate-change_en`

such as coastal areas and small island nations, the rise in sea levels poses an imminent threat to livelihoods and infrastructure. Furthermore, the disruption of agricultural patterns and water resources jeopardizes food security and exacerbates the risk of famine in vulnerable regions.
The consequences of global warming extend beyond environmental and social dimensions, influencing also the economic sectors and the geopolitical stability. The increasing frequency of natural disasters places a significant strain on national economies and infrastructure, impeding long-term sustainable development. Moreover, the geopolitical ramifications of climate-induced migration and resource scarcity may exacerbate tensions and conflicts in regions already grappling with political instability.
The interconnected nature of these challenges underscores the imperative for collective action to address the root causes of global warming and mitigate its far-reaching impact. In general, the impact of human actions is profound, with the surge in greenhouse gas emissions giving rise to a myriad of environmental changes that pose a threat to the delicate balance of the planet in every aspect. The interconnectedness of these effects underscores the urgency of addressing the root causes of carbon emissions and implementing sustainable solutions to mitigate their impact.

After understanding the significance of carbon emissions in the Earth's system, it is crucial to delve into the analysis of the primary causes behind their production. The primary sources of carbon emissions stem from the combustion of fossil fuels for electricity generation, transportation, industrial processes, and residential heating [27]. The burning of coal, oil, and natural gas releases large quantities of CO_2 into the atmosphere, constituting the largest proportion of anthropogenic carbon emissions. In addition, deforestation and land-use changes contribute to the release of CO_2, as forests act as natural carbon sinks, absorbing and storing carbon from the atmosphere. The rapid deforestation of vital ecosystems, such as the Amazon rainforest, has significantly diminished the Earth's capacity to sequester carbon, exacerbating the impact of carbon emissions on global warming.
This study specifically targets carbon emissions generated in electricity generation, given its direct correlation with the implementation of algorithms. This connection lies in the computational demands associated with running algorithms. The execution of complex algorithms, especially those used in machine learning and data processing, often requires substantial computational power. This demand is frequently met by data centers and computing facilities, which, in turn, rely on electricity for their operation.

2.2 Deep Learning Models

In the realm of AI, the inception of DL often denoted as deep neural networks, marked a departure from traditional problem-solving approaches. Initially, AI excelled at tasks that, while intellectually challenging for humans, were conceptually straightforward for computers – problems articulated by a set of formal, mathematical rules [40]. However, the true challenge for AI lies in tackling tasks that are intuitive for humans like speech recognition, where formal descriptions proved elusive.
The breakthrough in addressing these challenges came with the adoption of a new paradigm: allowing computers to learn from experience and comprehend the world through a hierarchy of concepts [41]. In this paradigm, each concept is defined in relation to simpler ones, forming a hierarchical structure. By accumulating knowledge from real-world experiences, this approach eliminates the necessity for human operators to explicitly specify all the knowledge required by the computer. The hierarchy of concepts empowers the computer to grasp intricate ideas by constructing them from simpler foundational ones. Visualized as a graph illustrating the building blocks of these concepts, the structure is deep, comprising multiple layers. This characteristic led to the nomenclature "deep learning" [1].

Thus, DL involves the training of multi-layered networks of nonlinear computational units [14]. Each of these units operates by processing inputs transmitted via weighted wires, resulting in a real number output derived from the weighted sum of the input values. This output is achieved through the application of a non-linear activation function, typically uniform across all gates within the network, although the number of inputs to individual gates may vary. Each layer processes and transmits information to the next layer, allowing the network to learn and make decisions based on complex patterns in the data it is trained on [42].
Usually, during the training process, the objective is to determine a set of weights for the wires that minimizes errors. The training of deep learning networks is facilitated through techniques like stochastic gradient descent, often referred to as backpropagation within the context of neural networks. During this process, an error function is constructed, and the network's weights are adjusted using the derivative of the error function. Notably, overfitting is a significant concern in DL, given that large networks can comprise hundreds of millions of weights [43]. On the other hand, the learning algorithm is unable to acquire certain properties, including the number of layers and neurons, as well as other choices like learning rate or activation function. These properties, known as hyperparameters, need to be manually set or determined through an optimization routine and cannot be learned during the training process [44]. These networks exhibit versatility in their output, ranging from single real numbers for regression tasks, multiple numbers in

the context of multivariate regression, to probabilities across different classes for both binary and multiclass classification scenarios. This adaptability underscores the power of deep neural networks to handle complex input data and perform diverse tasks with flexibility [11].

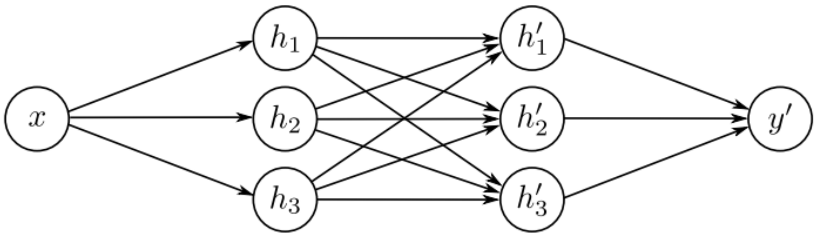


Figure 2.2: A neural network consisting of a single input, one output, and two hidden layers, with each hidden layer comprising three hidden units.

The Fig. 2.2 illustrates the architecture of a deep neural network designed with a specific configuration, featuring an input layer, two hidden layers, and an output layer. Each layer contains a defined number of units or neurons, showcasing the intricate connections that enable the network to learn and make predictions.
At the beginning of the network is the input layer. This layer represents the initial information fed into the neural network. In the context of the example, it can be imagined as the features of a dataset, where each feature corresponds to a specific aspect of the input data.
Moving into the heart of the network, there are two hidden layers, each comprising three hidden units. These hidden layers play a crucial role in learning complex patterns from the input data. The connections between units in adjacent layers are associated with weights, which are adjustable parameters learned during the training process.
At each hidden unit, the network performs a two-step process. First, it calculates the weighted sum of the inputs, considering the associated weights. This step involves multiplying each input by its corresponding weight and summing up these products. It reflects the network's ability to assign importance to different features based on their impact on the learning task. The second step involves applying a non-linear activation function to the weighted sum. This introduces non-linearity to the model, enabling the network to capture complex relationships in the data. Common activation functions include sigmoid, tanh, or ReLU (Rectified Linear Unit).
The final layer, known as the output layer, produces the network's prediction or output. The number of units in this layer depends on the nature of the task. For

example, in a binary classification task, there might be one unit with a sigmoid activation function, while a multiclass classification task could involve multiple units with softmax activation.

In traditional ML modeling, feature extraction often relies on manual efforts, requiring domain experts to identify and design relevant features for the model. However, the landscape shifts in DL, where feature extraction takes on an automated character [45]. The motivation for adopting DL, particularly in scenarios involving data such as images represented by low-level features like pixel intensities, stems from the pursuit of achieving a higher-level understanding of the data. Unlike traditional ML, DL models leverage their architecture to automatically learn and extract hierarchical representations of features from raw data, eliminating the need for explicit manual feature engineering. This shift in paradigm allows DL models to uncover complex patterns and representations in data, contributing to their efficacy in various domains. On the other hand, this translates to a strong correlation between data quantity and performance: the latter is often unsatisfactory when dealing with limited volumes of data [14].
DL is also driven by the concept of multi-task learning, which posits that an effective higher-level data representation should be applicable across various tasks [1]. For this reason, DL models often share initial levels of the network among different tasks. In a typical deep neural network architecture, layers of logical units are interconnected. In a fully connected layer, the output of each unit within the layer is linked to the input of every unit in the subsequent layer.

The ability to generate complex Neural Networks (NN), which needs the creation of larger datasets, has been one of the main causes behind the incredible advances in ML technology [4]. Representative NN today includes the Convolutional Neural Network (CNN) [46] and the Recurrent Neural Network (RNN) [1]. This thesis will concentrate on RNNs out of all the possible NNs. The choice is justified by their prevalent use within this research context and their capacity to address the specific requirements of this study.

2.2.1 Recurrent Neural Network

Human thinking does not reset every moment; instead, it relies on the ability to consolidate information acquired at different time instants, emphasizing the persistence of our thoughts. The term 'recurring' in RNNs, in fact, is attributed to their capacity to execute a consistent operation for every element within a sequence, with the resultant output contingent on preceding computations. They have the unique ability to process sequences of information one element at a time. At each temporal step, the network takes in a new input from the sequence while maintaining an internal state representation that encodes the sequence seen so

far [11]. This is achieved through the incorporation of a Hidden Layer within the network, where the key feature lies in the RNN's Hidden State – a 'memory state' retaining knowledge from previous inputs. This characteristic bears a striking resemblance to RNNs, which, just like our cognitive processes, have the capability to integrate information across different time instances. Thus, they represent a significant advancement in NN architecture, offering a dynamic solution for handling sequential data.
Fig. 2.3 shows the difference between an RNN and a feed-forward neural network:

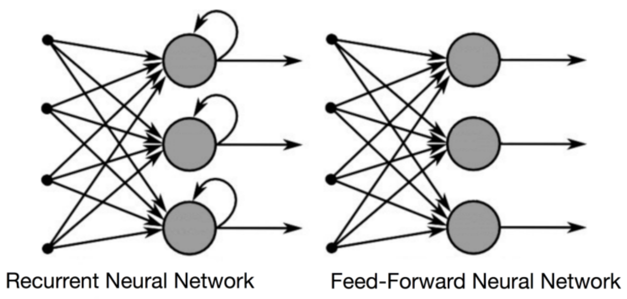


Figure 2.3: Difference in information flow between an RNN and a feed-forward neural network.

while traditional NNs maintain a unidirectional flow of activations from input to output, RNNs introduce a unique feature by incorporating backward connections. Feed-forward networks lack the capacity to retain previous input information, limiting their predictive abilities. Since these networks solely process the current input, they lack temporal sequencing. They have no inherent capability to recall any prior events, except for what has been learned during their training.

Essentially, an RNN in its most elementary form, comprises a neuron that accepts input, generates an output, and cycles that output back into itself – as depicted in Fig. 2.4.

In this way, it has the ability to consider the context of previous inputs, which gives the notion of dynamic change over time. The functioning of the network starts when it processes the input X_0 from the sequence, generating an output denoted as L_0. Subsequently, L_0, along with X_1, collectively serves as the input for the ensuing step. This process continues iteratively, with each L_t derived from the previous step and X_{t+1} forming the input for the subsequent step. This sequential progression allows the network to maintain context throughout the training process. The expression representing the current state can be defined as:

$$L_i = f(L_{i-1}, X_i) \tag{2.1}$$

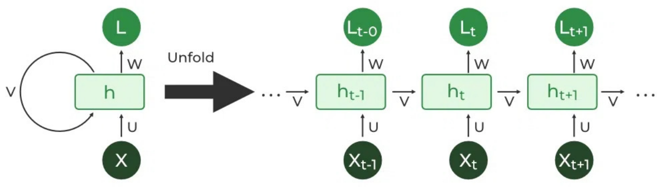


Figure 2.4: Basic instance of a RNN.

In this formula, L_i is the current state, which is a function of the preceding state $L_i - 1$ and the input X_i.
A feed-forward neural network, similar to various deep learning algorithms, employs a weight matrix for its inputs to generate an output. However, in contrast to this, RNNs assign weights to both the present and preceding inputs. Moreover, a recurrent neural network adjusts the weights for both gradient descent and backpropagation across time.
Incorporating the activation function in the formula 2.1, the current hidden state L_i can be expressed as:

$$L_i = \tanh(W_{hh} \cdot L_{i-1} + W_{xh} \cdot X_i) \tag{2.2}$$

Here, W represents the weight, L signifies the single hidden vector, W_{hh} denotes the weight at the previous hidden state, W_{xh} is the weight associated with the current input state, and tanh represents the activation function. This activation function introduces non-linearity and effectively compresses the activations into the range of [-1, 1].

Since RNNs exhibit the ability to remember and utilize preceding information, where inputs and outputs are not independent, they emerge as the solution to predict subsequent words in a sentence or sequence. Moreover, the utilization of shared parameters across all inputs or hidden layers decreases complexity and the number of parameters remains constant even as the number of time steps in the sequence expands [3].
On the other hand, as data progresses through an RNN, certain information is progressively eroded at each time step. Over time, the RNN's state gradually loses any trace of the initial inputs, functioning essentially as a short-term memory network. This issue is primarily associated with the unstable gradient problem, where the gradient diminishes as it traverses backward through layers, significantly slowing down learning in earlier layers. In RNNs, this issue is exacerbated since

the gradients not only flow backward through layers but also extend backward through time. This prolonged network operation increases the instability of the gradient, making it extremely intricate for effective learning [15]. To mitigate this issue, long-term memory cells have been introduced to address the limitations of conventional RNNs. These newer cell architectures have demonstrated such effectiveness that the base cells are no longer in use.

Similar to many other deep learning algorithms, RNNs are relatively old. The history of RNNs starts in the 1980s, but it is only in recent years, driven by increased computational power and abundant data resources, that we have recognized their remarkable potential. Furthermore, the introduction of Long Short-Term Memory (LSTM) networks in the 1990s has pushed RNNs to the forefront of deep learning. In particular, RNNs' intrinsic memory enables precise prediction and understanding of sequential data in various domains, including time series, speech, text, financial data, audio, video, weather, and more. Their unique ability to understand temporal dynamics surpasses the spatial content focus of other algorithms. In the words of Lex Fridman from MIT, RNNs shine when 'the temporal dynamics that connect the data are more important than the spatial content of each individual frame'.

2.2.2 Long Short-Term Memory

LSTM [47] is a particular type of RNN, specifically designed to handle problems involving sequential data, particularly those with long-term dependencies. Unlike traditional RNNs, LSTMs excel in remembering important information over extended time periods, solving the vanishing gradient problem often encountered in RNNs. The problem of vanishing gradients refers to the issue where, during training, the gradients calculated for adjusting the weights of neural network layers become exceedingly small as they backpropagate through the network during training. When gradients become very small, during the training process, the network's weights are updated very slowly or not at all, causing the network to learn slowly or not effectively.

The core of an LSTM model is the "cell state", which acts as a memory unit preserving information through time. This component allows LSTMs to capture dependencies between elements in a sequence, which is essential in understanding the context of the data. It is visually depicted by the horizontal line in Fig. 2.5, employing σ (Sigmoid) and tanh (Hyperbolic Tangent) layers. In the figure, h_{t-1} represents the output from the preceding LSTM unit, while h_t is the current output. Similarly, C_{t-1} denotes the memory from the last cell unit, and C_t is the newly updated memory. Finally, x_t is the input vector to the LSTM unit. The cell state runs is analogous to a conveyor belt: carrying and preserving information without alteration. The architecture is made up of specialized "gates" – the forget gate f_t,

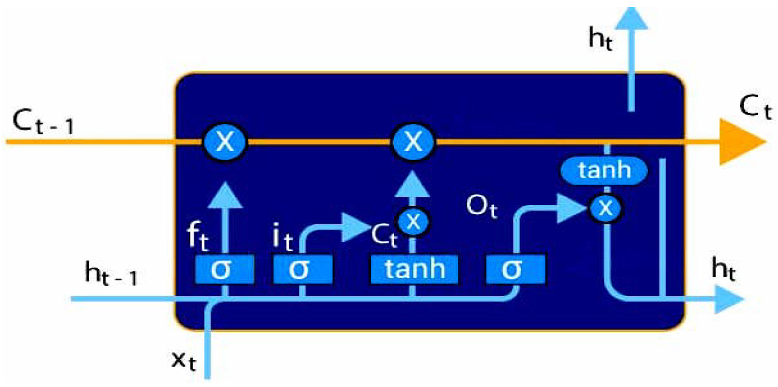


Figure 2.5: The memory element within the LSTM architecture.

input gate, and output gate. Each gate has its specific function in regulating the flow of information within the cell state.
The forget gate decides what past information to discard, taking into account the current input data x_t and the previous hidden state h_{t-1}. It filters out less relevant information, allowing the LSTM to prioritize essential elements.
The input gate processes new input data x_t and decides what new information is essential to remember, creating a "new memory update vector" C_t. This vector is then integrated into the cell state, modifying it according to the relevance of the new information.
Lastly, the output gate generates the new hidden state h_t by combining the updated cell state, the current input data, and the previous hidden state.
This way, the LSTM network effectively manages the long-term memory and short-term working memory, enabling it to process sequences by remembering and selectively using information, by employing basic addition or multiplication. For this reason, it offers an enhancement over the conventional RNN method and can be used for sequence prediction tasks, language modeling, and time-series analysis where understanding dependencies between elements over time is crucial. Furthermore, these gating mechanisms help maintain the steepness of the gradients, preventing them from diminishing significantly as they backpropagate through the network. By doing so, LSTMs effectively address the vanishing gradient problem, ensuring that the training remains effective and shorter and that the network maintains high accuracy in learning temporal dependencies over long sequences.

2.3 Carbon Emissions in Deep Learning

In the rapidly evolving landscape of DL and computational systems, understanding the intricate interplay between energy consumption and environmental impact is fundamental. This chapter embarks on an insightful exploration of energy modeling and environmental impact assessment within DL frameworks. Through a series of comprehensive subsections, this chapter delves into the theoretical foundations of energy models, investigates the relationship between energy consumption and carbon emissions in executing algorithms, and the tools used to calculate carbon emissions. This chapter seeks to analyze and understand why DL has an environmental impact and how it can be measured.

2.3.1 Theoretical Energy Models

In today's digital world, understanding how our computers use energy holds significant importance. This chapter is all about explaining two big concepts: energy and power consumption. These notions will be explored to demonstrate how programs consume energy during operation.

Understanding how ML tasks impact energy utilization is critical for reducing carbon emissions. This description investigates theoretical energy models as a first step toward measuring and comprehending the energy footprint of computing operations. These models offer insights into estimating energy from a theoretical standpoint, independent of programming languages or hardware platforms.

Defining Energy Consumption: Energy consumption represents the total power consumed over a specific duration [48]. It is expressed as the integral of power with respect to time:

$$E = \int_0^T P(t)\, dt \tag{2.3}$$

where: E = Energy [**J**]
Correspondingly, power (P_{avg}) is delineated as the rate at which energy is utilized within a given timeframe, calculated as the total energy consumed (E) divided by the duration (T):

$$P_{\text{avg}} = \frac{E}{T} \tag{2.4}$$

where: P = Power [**W**]
Static and Dynamic Power: Within computing systems, power is categorized into static (leakage) and dynamic components. Static power refers to consumption during inactivity, while dynamic power involves the energy dissipated by the circuit during active operations [48]. Dynamic power (P_{dynamic}) is influenced by factors such as activity factor (α), voltage (V_{dd}), capacitance (C), and clock frequency (f):

$$P_{\text{dynamic}} = \alpha \cdot C \cdot V_{\text{dd}}^2 \cdot f \tag{2.5}$$

where α will be zero if the circuit is off [49].
Dynamic power directly affects how much energy the processor uses.

Program Energy Consumption: The energy consumed by a program is a product of instructions count (IC), clock per instruction (CPI), and energy per clock cycle (EPC):

$$E = IC \cdot CPI \cdot EPC \tag{2.6}$$

The EPC defines the energy expended per instruction and exhibits a direct correlation with $C \cdot V_{\mathrm{dd}}^2$. Diverse instructions possess distinct CPI values, signifying varied energy usage. For instance, memory accesses, incur significantly higher energy consumption than floating-point operations due to the multiple cycles and access delays associated with memory operations [50].

2.3.2 Energy Consumption and CO_2 relationship

The exploration of the intricate relationship between energy consumption and carbon emissions in local computing systems is fundamental to understanding the environmental impact of algorithm execution. As depicted in Fig. 2.6, this section aims to unravel the complex interplay between the energy sources fuelling computational processes and the resulting carbon emissions. It delves into the energy flow within local computation, elucidating the process where traditional fossil fuels like coal, oil, and natural gas undergo combustion in power plants to generate electricity. This combustion emits GHG, including carbon dioxide, contributing significantly to the carbon footprint.

The electrical grid, functioning as a distribution network for electrical power, relies on power plants as primary sources. Fossil fuel power plants play a crucial role by converting carbon-based fuels into energy through combustion, and heating water to drive turbines that generate electricity. This is the process that emits GHG, into the atmosphere. However, the landscape of energy sources extends beyond these fossil fuels. The electrical grid harnesses power from diverse renewable or low-carbon sources like hydroelectric and solar power, contributing to a varied energy mix.
It is important to note that different power plants use varying fuel sources, resulting in a combination of energy types within a geographical region. This mix (called energy mix) might include a combination of fossil fuel power plants (coal-fired, oil-burning, natural gas) alongside renewable energy plants [32]. The choice of fuel significantly impacts the environmental footprint of a power plant. Coal-fired plants, for instance, exhibit higher CO_2 emissions per unit of power produced compared to oil-burning and natural gas plants [51].

Considering this varied composition of power sources within the local electrical grid, the carbon emissions footprint of an individual computational device becomes contingent on these sources. Therefore, evaluating the environmental impact necessitates a thorough understanding of the energy mix powering computational

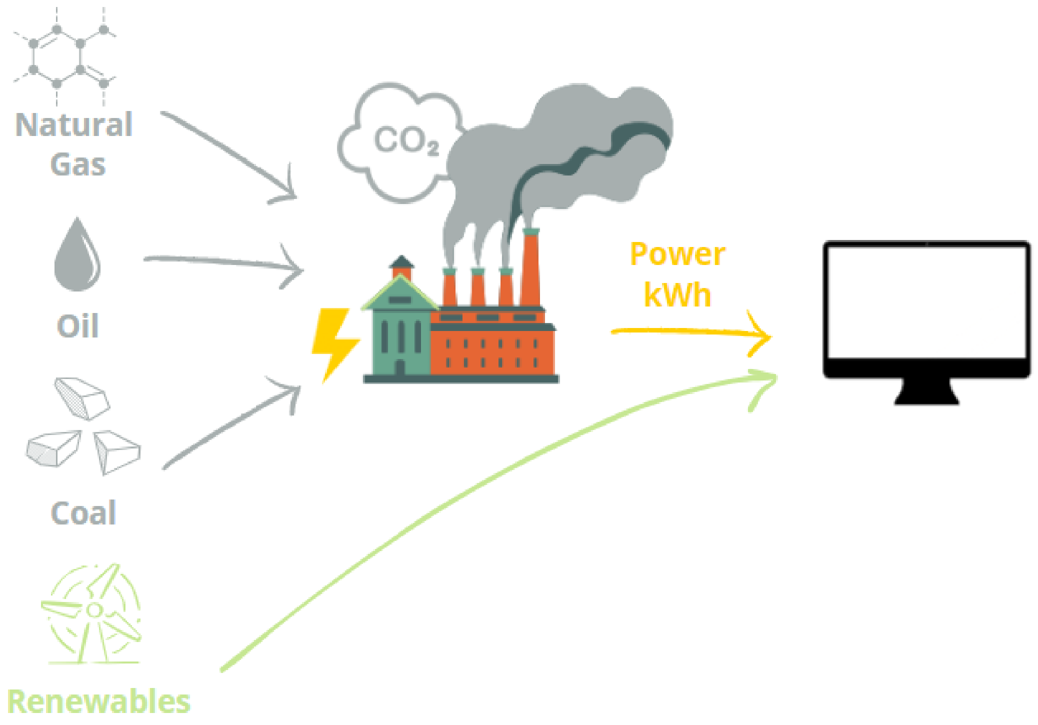


Figure 2.6: Overview of energy flow to power DL computations.

activities. Overall, the types of fuels employed by power plants within the grid directly impact the carbon emissions associated with each computing device. This underlines the criticality of comprehending and accounting for the diversity of energy sources in assessing the environmental impact of local computation.

2.3.3 Carbon Emissions Calculation Tools

In the realm of carbon emissions calculation tools, a deep exploration reveals a diverse landscape that matches different preferences and study requirements. The first crucial distinction lies in the environment within which each tool operates: whether integrated into a Python script or accessible online. A meticulously organized Table 2.1 catalogs the surveyed tools, delineating them based on their respective environments. The primary divergence stems from whether the analysis is conducted in real-time, as is the case with Python libraries, or post-execution, characteristic of online tools. Regardless of the programming language being used, online tools may be utilized without changing the code. Python libraries provide measurements of the consumption of various parts of the script but clearly can

only be used in Python programming.
Despite these differences, a unifying factor among all these tools is their ability to calculate emissions in terms of energy costs. This is paramount, considering that current scientific metrics predominantly revolve around computational time. However, this metric proves insufficient in providing a comprehensive overview due to the variability in hardware components. In fact, the same code executed on different machines can yield drastically divergent energy usage patterns. This unpredictability underscores the necessity of integrating energy cost evaluations within the framework of AI model development.
Moreover, the prevalent focus on inference and, occasionally, the training time of the final model fails to encapsulate the complete energy footprint of the AI development cycle [52]. The process of developing DL models for deployment consists of multiple stages: model selection, training, and inference. Model selection, a crucial part of the process, entails multiple iterations, diverse architectures, and extensive hyperparameter tuning, collectively contributing to substantial energy expenditures. Continuous learning paradigms exacerbate this issue by necessitating recurrent retraining, entailing extended periods of model adjustments and consequent energy consumption, aspects often sidelined in traditional evaluation metrics.
In order to fully assess the environmental impact of AI algorithms, it is necessary to shift towards a more inclusive evaluation methodology. This entails accounting for the full spectrum of energy costs, spanning not only inference and the final model training but also encompassing the cumulative energy expenses incurred during model selection, hyperparameter tuning, and continuous learning phases. Such an encompassing approach is crucial to fostering conscientious AI development that factors in both computational efficiency and environmental sustainability. In recent years, several open-source platforms have aimed to improve transparency by estimating energy usage and carbon emissions associated with DL experiments. Rather than simply measure instantaneous resource usage, these tools aim to model overall energy demands from code and dataset activity over time. This facilitates more robust comparisons of sustainability across different modeling strategies and infrastructures. The immediate and observable consequence of training and deploying a model is the release of CO_2 and other GHG resulting from heightened power consumption, specifically the dynamic consumption, during operational runtime of the equipment [53]. Among the significant tools studied are:

1. Green-Algorithms[3] (GA) [54]: employs a user-centric approach to estimate energy consumption focusing on both CPU and GPU usage. For CPU assessment, users can input their CPU model, enabling GA to access the Thermal Design Power (TDP) from its internal database. Alternatively, users can

[3]`https://www.green-algorithms.org/`

Python Libraries	**Online Tools**
CodeCarbon	Green Algorithms
Carbontracker	
Eco2AI	
experiment-impact-tracker	ML CO2 Impact
Cumulator	
energyusage	

Table 2.1: Commonly used carbon tracking tools available for online estimation afterward or at runtime in Python scripting.

manually input the TDP if available. In the absence of user-provided TDP data, GA defaults to an average approximation of 12 watts per core for power consumption estimation.
When it comes to GPU analysis, GA employs a similar model-based strategy, searching a predefined list for the corresponding TDP based on the GPU model provided. Users also have the option to input the GPU's TDP if it's not included in the database. In cases where the TDP remains unknown, GA uses an average approximation of 200 watts for GPU power consumption estimation [54].

2. ML CO2 Impact[4] [55]: focuses solely on estimating the energy consumption of GPU usage. Therefore, the tool does not provide insights into the energy consumption of the CPU during the process. It offers users the ability to input their hardware, runtime, and cloud provider details to calculate two key metrics: raw carbon emissions produced and an estimated offset figure, which is contingent upon the cloud provider's grid specifications.
This tool operates by employing a model-based methodology to approximate the TDP of the GPU based on a predefined list of GPU models and their corresponding TDP values. However, if the provided GPU model isn't included in the database, MLCO2 cannot automatically retrieve the TDP value. To

[4] `https://mlco2.github.io/impact/`

address this limitation, users are encouraged to contribute missing TDP values by submitting a pull request for database updates [55].

3. CodeCarbon [5] [56]: provides insights into the energy consumed by CPUs and GPUs during software execution. To measure CPU energy consumption, CodeCarbon relies on RAPL (Running Average Power Limit) files or on Power Gadget, specifically for INTEL CPUs, and requires root access. However, if access is not available, CodeCarbon utilizes the model of the CPU in order to find the TDP. In case the model is unknown, the tool uses a fixed value of 85W. For GPU energy consumption estimation, CodeCarbon utilizes the `pynvml` library, specifically designed for NVIDIA GPUs. Access to either the RAPL files or NVIDIA GPUs is necessary for CodeCarbon to accurately track and report energy consumption [56].

4. Carbontracker[6] [57]: specializes in tracking the energy consumption of CPU and NVIDIA GPUs during model training. It leverages RAPL files, specifically for INTEL CPUs with root access, to report CPU energy consumption. However, without access to these files, CPU measurement becomes unavailable. For NVIDIA GPUs, CarbonTracker utilizes the `pynvml` library to measure energy consumption, exclusively catering to NVIDIA GPUs and not supporting non-NVIDIA ones [57].

5. Eco2AI[7] [58]: offers a comprehensive approach to monitor energy consumption of both CPU and NVIDIA GPUs. It calculates equivalent carbon emissions by considering regional emission coefficients. For CPU energy consumption estimation, Eco2AI utilizes a model-based approach, searching a predefined list for the corresponding TDP. In cases where the TDP is unavailable, Eco2AI employs an average approximation of 100 watts. When measuring GPU energy consumption, Eco2AI relies on the `pynvml` library, specifically tailored for NVIDIA GPUs, omitting measurement capabilities for non-NVIDIA GPUs [58].

6. experiment-impact-tracker[8] (EIT) [59]: specializes in tracking CPU and NVIDIA GPU energy consumption during computational tasks. For CPU energy measurements, EIT relies on RAPL files, specifically accessible for INTEL CPUs with root access and Linux operating systems. However, it does not offer CPU energy consumption monitoring for non-INTEL CPUs or non-Linux systems.

[5]`https://codecarbon.io/`

[6]`https://github.com/lfwa/carbontracker`

[7]`https://github.com/sb-ai-lab/Eco2AI`

[8]`https://github.com/Breakend/experiment-impact-tracker`

When measuring NVIDIA GPU energy consumption, EIT uses the `nvidia-smi` command line, tailored exclusively for NVIDIA GPUs. It doesn't extend its measurement capabilities to non-NVIDIA GPUs [59].

7. Cumulator[9] [60]: measures CPU utilization by default. For CPU analysis, it utilizes a model to identify the TDP by searching a predefined list, resorting to an average of 250W in case the TDP is unknown.
   Regarding GPU assessment, Cumulator employs a similar approach, using a model to match the GPU model with its corresponding TDP. If the TDP is unavailable, it defaults to an average of 250W, based on the Nvidia GeForce GTX Titan X, a prevalent GPU model in the EPFL Machine Learning and Optimization Laboratory's IC cluster. Cumulator evaluates only one GPU or one CPU.
   Compatible with Linux, Windows, and MacOS systems, Cumulator offers also a web application feature. This web-app automatically estimates accuracy and power consumption for four distinct algorithms – Linear Regression, Random Forest, Decision Tree, and Neural Network –based on the provided dataset. Users can upload datasets and specify the target column, which will be automatically excluded from the accuracy and power consumption computations [60].

8. energyusage[10]: measures CPU power consumption by accessing the RAPL interfaces present in Intel processors. In addition to CPU power, EnergyUsage accounts for GPU power usage for systems equipped with Nvidia GPUs that support the NVIDIA-smi program. However, it's important to note that due to the specific methods used for energy measurement – utilizing the Intel RAPL interface and NVIDIA-smi – the package is compatible only with Linux kernels.

The summarized characteristics of the carbon emission calculation tools provided in Table 2.2 delineate their primary features regarding energy measurement, particularly concerning CPU and GPU assessments. Additionally, the table specifies operating system requirements where pertinent for each tool. The definition of energy measuring methodology and operating system compatibilities within these tools serve as a critical reference to users looking to include carbon emission estimates into their AI model-building process.

[9] `https://pypi.org/project/cumulator/`

[10] `https://pypi.org/project/energyusage/`

Tool	CPU Energy	CPU Model / RAPL or Power Gadget	GPU Energy	GPU Model / Power Tool	Required OS	Compatibility
CodeCarbon	Yes	Yes / RAPL and Power Gadget	Yes	No / pynvml	-	All CPUs Nvidia GPUs only
Carbontracker	Yes	No / RAPL	Yes	No / pynvml	Linux	Intel CPUs only Nvidia GPUs only
Green Algorithms	Yes	Yes / No	Yes	Yes / -	-	All CPUs All GPUs
Eco2AI	Yes	Yes / No	Yes	No / pynvml	-	All CPUs Nvidia GPUs only
EIT	Yes	No / RAPL and Power Gadget	Yes	No / nvidia-smi	-	Intel CPUs only Nvidia GPUs only
Cumulator	Yes	Yes / No	Yes	Yes / -	-	Intel CPUs only Nvidia GPUs only
ML CO2 Impact	No	-	Yes	Yes / -	-	No CPUs All GPUs
energyusage	Yes	No / RAPL	Yes	No / nvidia-smi	Linux	Intel CPUs only Nvidia GPUs only

Table 2.2: Summary of the fundamental features of each tool for measuring energy and CO_2 equivalents.

2.3.4 Deep Learning's Environmental Implications

The integration of the digital landscape within modern society presents a significant opportunity for advancing energy efficiency and monitoring carbon emissions, particularly in ML algorithms. The exponential growth and reliance on internet-enabled devices, coupled with widespread internet accessibility, have revolutionized our understanding of online and offline realms. However, this digital revolution, despite its potential for smarter energy usage and management, has posed substantial challenges in terms of energy consumption [61]. In this context, Information Technology (IT) companies play a pivotal role in steering the transition toward a more sustainable, renewable energy-driven economy, crucial for reducing GHG emissions and mitigating climate change impacts. Despite considerable strides in enhancing energy efficiency, the surge in IT-related energy demand is remarkable, encompassing the energy required not only to power the devices but also to sustain data centers, communication networks, and the manufacturing processes of these technological tools. The IT sector, primarily based in energy-intensive manufacturing hubs like China and, generally, in Asia, currently consumes a considerable amount of global electricity, as depicted in Fig. 2.7. This highlights the pressing requirement for thorough monitoring and analysis to reduce carbon emissions in ML algorithms, which is essential in managing the environmental impact of this fast-growing technological field.

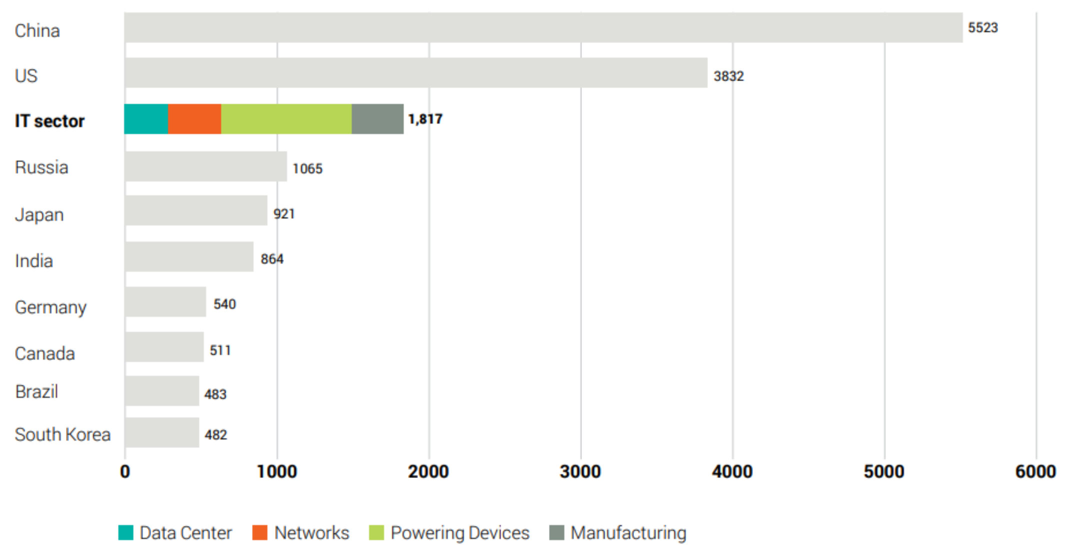


Figure 2.7: Comparison of Global Electricity Consumption in 2012 with the IT Sector's Energy Usage in billion kilowatt-hours (kWh) [62].

In the current domain of ML, DL emerged as a fundamental component, driving innovations across various applications that often demand extensive training, ongoing data monitoring, and meticulous hyperparameter optimization [17]. However, this trajectory is unsustainable from both environmental and economic perspectives. Thus, shaping the ML's energy impact associated with the extensive use of DL models plays a crucial role. As a matter of fact, DL has rapidly progressed thanks to factors such as larger datasets, advanced algorithms, and increased computational resources, leading to substantial performance improvements. The training of increasingly complex neural architectures has been made easier by the advent of powerful GPU and TPU accelerators, but this has resulted in an increasing demand for processing power, which is directly correlated with the execution time [17, 20]. The compute demand for DL surged dramatically from 2012 to 2018, escalating by 300,000 times [63]. As illustrated in another study [64], since 2012, DL models, while achieving impressive capabilities in terms of accuracy, have been associated with a training cost doubling every few months, leading to substantial environmental costs. Consequently, as the number of application domains and the complexity of DL models continue to increase, the criticality in terms of consumption of both data availability and extended training durations will be emphasized.

Overall, this growth comes at a significant environmental cost, since the significant upsurge in computational needs has led to a pronounced increase in energy consumption. In fact, these recent developments – the possibility to obtain larger amounts of data and the development of more complex model architectures – have led to two main consequences: an increase in energy-intensive data storage solutions due to the continuous need for data retrieval and transmission, data redundancy and the maintenance of cooling systems in data centers, along with a heightened need for computational power to sustain long training sessions.
Considering also that the energy sector stands as the primary source of worldwide GHG emissions, it is evident the huge environmental impact deriving from these advancements. Together, the aforementioned studies [63, 64] shed light on the prohibitive expenses linked to this trend, which align with the 'Red AI' framework – that focused on enhancing accuracy by leveraging extensive computational resources, often overlooking the associated costs or environmental implications [64]. Additionally, since a large portion of the research community cannot afford the necessary resources, the cumulative effect not only has an impact on the environment but also creates barriers to the development of AI. While hardware improvements have enabled training larger models with billions of parameters like GPT-3 [65, 66], optimizing such networks requires immense computing resources that are difficult and expensive to scale further [67]. This may suggest that the growth rate in DL's consumption of computational power may slow down [16]. Moreover, the limited supply of specialized AI chips also imposes constraints.

Despite international agreements like the UNFCCC and the Kyoto Protocol [68], GHG emissions surged at a faster rate from 2000 to 2010 compared to the preceding decade. Specifically, the annual growth of GHG emissions in the global energy supply sector increased from 1.7% per year between 1990 and 2000 to 3.1% per year from 2000 to 2010 [27]. Thus, considering the conjunction of the general problem of GHG emissions and the vast usage of DL methods, addressing the growing energy demand within this field is even more crucial. Exploring avenues to enhance energy efficiency in DL models is important in order to mitigate this global environmental impact. Additionally, raising awareness among practitioners about the energy and carbon footprint of these models could prompt proactive steps toward reducing environmental implications.
For instance, in order to continue advancing capabilities while addressing these sustainability concerns, researchers are exploring alternatives beyond the constant upscaling of computing. Different works [33, 20] have found algorithmic efficiency to be a promising avenue, showing reductions in operations needed to match past baselines. Hardware optimizations also multiply efficiency gains significantly over time. Researchers are likely to turn to problem-focused methods leveraging these algorithmic innovations rather than relying primarily on "brute force" scaling

through raw computation. Overall, a more balanced consideration of environmental costs presents an opportunity to reorient DL progress onto a sustainable path.

2.3.4.1 Continuous Training's Impact

In the landscape of AI, the notion of continuous training emerges as a critical contributor to the escalating energy demands. Unlike traditional model development processes, continuous training involves iterative refinement and constantly updating models to adapt to evolving data patterns. This perpetual learning paradigm, while enhancing predictive performance, amplifies the energy consumption of models. Understanding the disproportionate environmental impact of continuous training is paramount as it is pivotal in several application domains.

Among the domains most profoundly affected, Natural Language Processing (NLP) stands out prominently. The field has witnessed the proliferation of state-of-the-art DL models, particularly those excelling in diverse NLP tasks. The computational intensity of these models, demanding specialized hardware like GPUs or TPUs, raises accessibility concerns due to the associated financial costs ([5, 69]). The NLP community has expressed concerns regarding the escalating energy consumption and its repercussions on the environment and equitable access to research advancements. Initiatives such as dedicated conference tracks focusing on sustainable NLP methods[11] underscore the imperative to address these energy concerns [52]. This increased emphasis extends to evaluating the environmental implications and financial costs linked with both training and deploying large language models. The development and training of new models, often necessitating specialized hardware, amplify these energy costs, drawing attention to the carbon footprint and environmental toll of heightened energy consumption. The trend toward larger language models, with ever-increasing parameters and training data, presents a dual challenge in terms of environmental risks and understanding the implications of their sheer size. The proliferation of massive language models, such as BERT, GPT, and their variants, prompts a pressing need to evaluate these models based on the resources they consume and report on their associated costs. This is crucial because the environmental and financial costs of such models disproportionately affect marginalized communities, which are less likely to benefit from these advancements while being most vulnerable to adverse environmental consequences. As models continue to expand in size, so does the difficulty of comprehending the extensive training data, creating added layers of complexity in understanding their environmental impact and resource consumption. Therefore, the research community must prioritize assessing and mitigating these environmental and financial costs while developing

[11]`https://2021.eacl.org/news/green-and-sustainable-nlp`

strategies to understand and address these implications within the NLP field.

A recent study [25] investigates the carbon footprint of developing a NLP model through its multi-stage training process. The researchers found that the total CO_2 emissions from a model's training iterations could be equivalent to what would be produced by 5 cars over their usable lifetimes. It also equaled the emissions from over 300 flights between two major cities. This highlights the environmental impact of developing even a single sophisticated AI system. While the goal of AI research is often to optimize for metrics like accuracy, this work underscores the need to also consider sustainability.
Rohde et al.[70] have profiled the energy demands of tasks in computer vision, speech recognition and gaming. Models that handle more complex problems require much more intensive computations. The computing power is quantified in petaflop-days, which refers to the floating point operations performed in a day at a scale of tens of trillions. More intricate AI architectures directly translate to higher energy usage, GHG emissions and resource expenditure.

Extending our lens beyond NLP, the domain of time-series forecasting stands as another critical area with significant considerations. Here, continual training is pivotal in enhancing models' predictive abilities over time. However, the ramifications of energy-intensive training processes transcend mere model development, encompassing broader environmental and financial implications. Notably, literature lacks comprehensive studies addressing the energy consumption and emissions produced by algorithms applied in this domain, signaling a gap in understanding the environmental footprint of time-series forecasting models.

Collectively, these studies demonstrate the significance of evaluating new AI techniques based on their carbon footprint in addition to their predictive capabilities. Energy efficiency must be a priority as models continue increasing in scale and sophistication.

2.4 Existing Energy-Efficient Approaches

This section provides a summary of recent ML research on energy and power estimation during specific phases of computation (training and inference).

The primary goal of early attempts to optimize DL models was to decrease the number of parameters or weights in neural networks. Techniques like pruning, compression, and model compactness aimed at diminishing the number of weights to curtail the memory accesses, which account for a significant fraction of energy consumption, are studied in papers such as [71] and [72]. Overall, the goal was to alleviate the energy burden by minimizing memory-related overheads while preserving model performance.

Then, application-specific accelerators and predictive energy modeling were introduced by two main projects: Eyeriss [73] and NeuralPower [74], respectively.
Eyeriss emphasized energy efficiency by reconfiguring architecture, tailored for deep CNNs, by incorporating compression techniques. Moreover, it employs a row stationary (RS) dataflow to optimize data movement and reduce costly DRAM accesses [73]. In this way, Eyeriss achieved notable gains in energy efficiency while ensuring high throughput.
NeuralPower introduced layer-wise predictive models based on sparse polynomial regression. These models accurately predict energy consumption across all layers of CNNs deployed on GPU platforms. By providing insights into power and runtime at each layer, NeuralPower aids in identifying energy bottlenecks and guiding the selection of energy-efficient architectures. This predictive modeling approach, as evidenced in [74], outperformed prior models by significantly improving prediction accuracy, and enable ML researchers to make informed decisions about energy-efficient architectures.

Other studies, such as DeLight, integrate energy awareness in the design and training phases of DL models by modeling energy use concerning basic arithmetic operations and communication of shared weights among different cores. It aimed to optimize energy consumption during network training [75], promising potential reductions in overall energy consumption.

Chapter 3

Problem Statement and Dataset Description

An empirical approach centered on practical experiments formed the foundation of this study. This research employed two separate real-world datasets that were both significant and relevant in order to conduct a comparative analysis. These datasets, while distinct in their nature and characteristics, were crucial for comparing and contrasting different aspects of the research. The first dataset comprised Traffic Data sourced from an Italian Mobile Network Operator, while the second dataset involved PVWatts Energy Estimate Data from Turin. Both datasets shared a common domain – time-series data. Initially, the study delved into unraveling the intricate energy consumption patterns exhibited by prevailing DL algorithms when applied to network data. Analyzing network data posed numerous challenges owing to its real-time nature and inherent variability. This required adept handling of streaming data while simultaneously addressing the periodicity in training models to ensure relevance and accuracy. Subsequently, a similar in-depth analysis was conducted on the solar panel production data. The primary distinguishing factor between the two datasets, pivotal for this research, lay in the stability of the data. The production data derived from photovoltaic (PV) panels demonstrated a markedly more predictable trend compared to the inherently volatile nature of traffic data in real-time scenarios. This dichotomy in data stability provided a crucial contrast for understanding and delineating the nuances of energy consumption patterns when considering these distinct problems that belong to the time-series domain.

3.1 Traffic Data from Italian MNO

The traffic dataset [76] used in this study is provided by an Italian Mobile Network Operator (MNO) located in Milan and its surrounding area. This dataset provides a comprehensive perspective on the network traffic conditions prevalent throughout the region, as depicted in Fig. 3.1. The city has been divided into seven distinct zones, each chosen for its unique characteristics and activities. The hourly traffic volume has been collected from 1420 Base Stations (BSs) spanning a duration of two months during the year 2015.

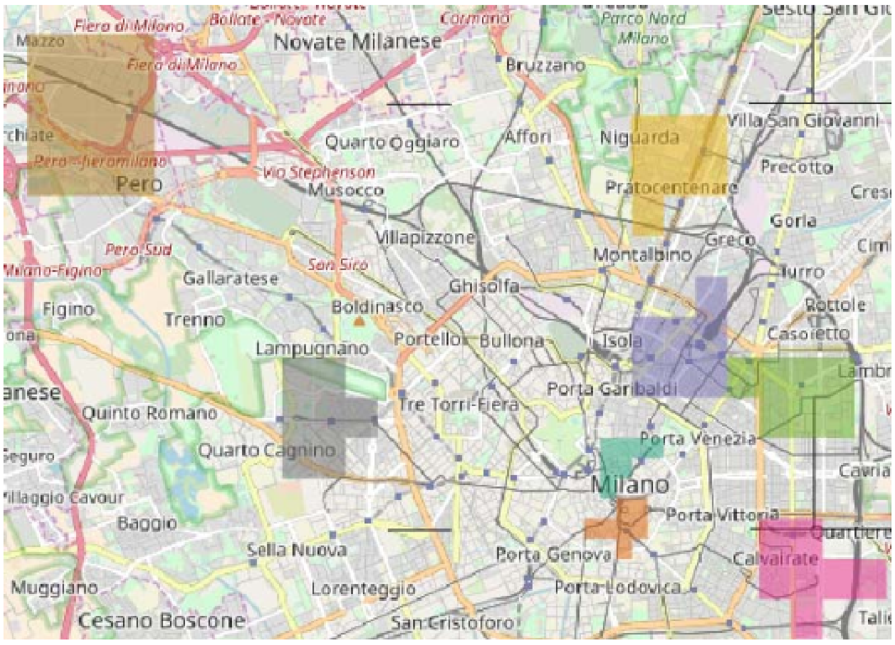


Figure 3.1: Examined regions with traffic volumes.

Zone	**Color in the map**
Business	dark green
Residential	yellow
Train station	purple
San Siro	grey
Politecnico di Milano	light green
Industrial	magenta
Rho Fiere	brown

Table 3.1: Zone and corresponding map colors.

Table 3.1 displays the considered zones and their corresponding colors as represented on the map. Each zone is associated with a specific color for visual identification on the map. The zones serve as microcosms, representing the diverse areas found within an urban environment. For instance, the Politecnico di Milano (Polimi) area, marked by the light green square in the figure, denotes an area frequented by students, experiencing heightened activity levels during specific times of the day. In contrast, other zones represent a mix of business districts, residential streets, train station areas, soccer stadiums, university campuses, industrial sectors, and exhibition venues, each exhibiting its own traffic patterns and behaviors.

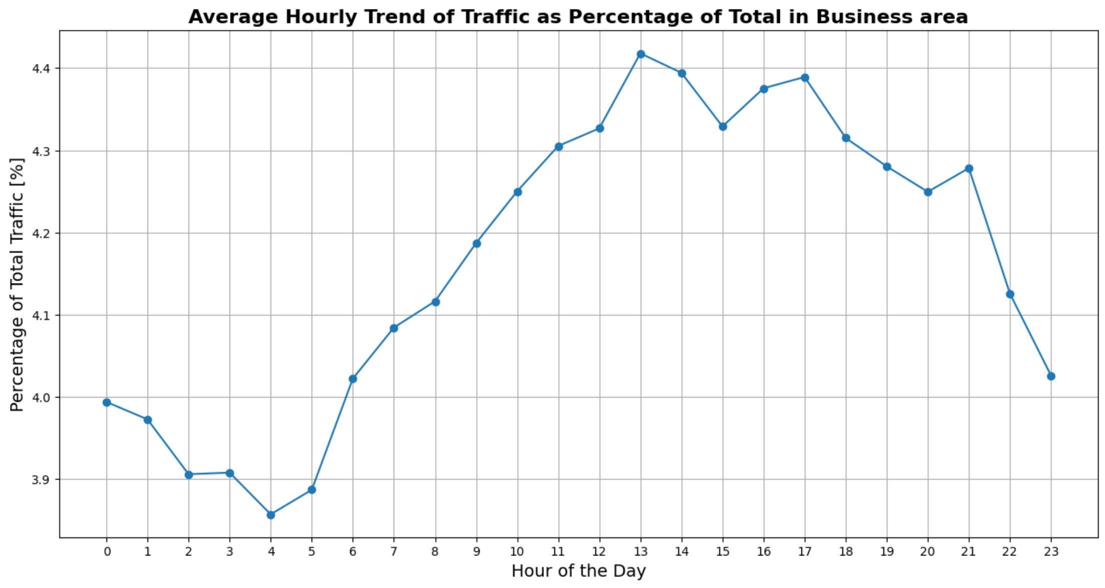


Figure 3.2: Average hourly distribution of traffic as a percentage of daily total. Specifically considering the business area.

For example, the business district (dark green) witnesses traffic peaks during core business hours, while the residential zone (yellow) observes increased traffic in the evening. Similarly, the train station area (purple) reflects high activity, primarily coinciding with the start and end of typical working hours. The San Siro neighborhood (grey), home to the soccer stadium, presents sporadic and fluctuating traffic volumes depending on event schedules.
The visualization in Fig. 3.2 depicts the proportional distribution of traffic across individual hours, considering the entirety of a day within the business area. Each data point represents the percentage of traffic observed during a specific hour relative to the total traffic recorded throughout the day. This analysis offers insight into the hourly traffic patterns within the business area: the data illustrates a

consistent increase in traffic volume from 8 am, reaching its peak around 1 pm, followed by a gradual decline. This pattern corresponds with typical working hours, depicting heightened activity during the morning and early afternoon, gradually tapering off later in the day.

Within each zone, the presence of a macro BS along with 6 micro BSs are considered. Both these types of stations are components of cellular networks, each serving distinct coverage areas and functions within the network infrastructure:

- Macro base stations are large cell towers strategically positioned to cover larger geographic areas (≤ 35 `km`), such as neighborhoods, towns, or urban areas [77]. They provide wide-area coverage and are usually installed at higher elevations to maximize coverage range. These base stations handle high-capacity data and voice traffic, serving a large number of mobile devices within their coverage area.
- Micro base stations are smaller in size and cover more localized areas compared to macro stations. They are deployed in areas with high user density or where additional capacity is needed. Micro base stations have lower power and cover shorter distances compared to macros – ranging from a few meters to one or two kilometers [77].

They might be used concurrently in the same geographic area and they are both integral to maintaining an efficient and reliable cellular network. Indeed, they work together to provide seamless connectivity across different scales and user demands. This configuration with one macro and more micro Bss, ensures that the service area is effectively covered by one macro cell that overlaps with smaller cells, thus enabling comprehensive network coverage across various zones.

The dataset organizes each BS's data into a format that contains several metrics for analysis that vary over time. Each entry in the dataset is associated with a timestamp, that indicates the temporal sequence in Unix timestamp format, acting as the temporal reference point for the recorded data. The level of granularity and extensive temporal coverage (2 months) within the dataset enables multifaceted analyses and comprehensive evaluations of user behavior and traffic dynamics across the varied BS. Since the original data was stated in KBytes, the total network traffic is determined using the information retrieved from the dataset and then multiplied by 8000. The problem's target is represented by this entire volume.

The dataset provided comprises network traffic measurements recorded every 15 minutes. The aim is to forecast the network traffic at specific time intervals by utilizing the information from previous time steps, such as in [76]. Thus, predicting the total network traffic f of BS b at hour t of day d, leveraging historical traffic data.

Let $\mathbf{N_{b,d,t}}$ symbolize the total network traffic at BS b during hour t on day d. The primary aim is to forecast $\mathbf{N_{b,d,t}}$ by leveraging the network traffic data from the preceding k time steps. Given that the dataset is sampled every 15 minutes, the objective is to predict the initial 4 values within an hour $(t, t+1, t+2, t+3)$ based on the k preceding traffic measurements.

Let $\{\mathbf{N_{b,d,t-i}}\}_{i=1}^{k}$ denote the sequence of total network traffic samples at BS b for the past k time steps. The prediction model aims to estimate $\mathbf{\hat{N}_{b,d,t}}$, the predicted total network traffic at hour t based on the previous k observations:

$$\mathbf{\hat{N}_{b,d,t}} = f(\mathbf{N_{b,d,t-1}}, \mathbf{N_{b,d,t-2}}, \ldots, \mathbf{N_{b,d,t-k}}) \tag{3.1}$$

Here, f represents the predictive model, which could be an ML algorithm or a specific mathematical function. The function f is trained on historical data to learn the relationship between past traffic observations and the subsequent value. The primary objective is to minimize a loss function that quantifies the dissimilarity between the predicted and actual total network traffic values. In general terms, the objective is formulated as follows:

$$\underset{f}{\text{minimize}}\ \mathcal{L}(\mathbf{N_{b,d,t}}, \mathbf{\hat{N}_{b,d,t}}) \tag{3.2}$$

The specific Mean Absolute Error (MAE) formulation used to quantify this dissimilarity is given by:

$$\mathcal{L}_{MAE}(\mathbf{N_{b,d,t}}, \mathbf{\hat{N}_{b,d,t}}) = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{N_{b,d,t+i}} - \mathbf{\hat{N}_{b,d,t+i}}| \tag{3.3}$$

This equation calculates the average absolute differences between the actual and predicted values over n prediction steps, providing a measure of the model's accuracy in forecasting total network traffic at each time step.

3.2 PVWatts Energy Estimate Data in Turin

The second dataset utilized in this study originates from the PVWatts Calculator, an innovative tool[1] developed by the National Renewable Energy Laboratory[2] (NREL) to assist individuals in evaluating and planning solar panel installations. This tool enables users to input site-specific data, providing energy-production estimates

[1]`https://pvwatts.nrel.gov/`

[2]`https://www.nrel.gov/`

crucial for determining the most suitable size, placement, and configuration of solar systems. NREL, a government-owned research facility based in Golden, CO and funded by the US Department of Energy, specializes in renewable energy and energy efficiency research, boasting over two decades of leadership in the sector. The PVWatts Calculator was conceived as part of NREL's collaboration with the Environmental Protection Agency within the RE-Powering America's Land initiative [78].

The dataset plays a pivotal role in our research by offering crucial insights into estimating solar energy production, with a specific focus on the Turin area. Its significance lies in its ability to provide detailed and region-specific information that aids in accurately predicting solar energy output. The data is based on realistic solar irradiation patterns, representing the Typical Meteorological Year (TMY) in the area with hourly granularity. A TMY refers to a standard collection of weather data encompassing hourly values throughout a year at a specific geographical spot [79]. These datasets are curated from long-term records, usually spanning a decade or more. The selection process involves picking data for each month from the year that best represents the typical weather patterns for that specific month. For example, data for January might originate from 2013, while February's data could be sourced from 2020, and so forth. PVWatts utilizes weather information derived from the NREL National Solar Radiation Database[3] (NSRDB) where accessible, supplemented by data gathered from various other sources to cover regions beyond its availability.

In particular, the data incorporated into this study originates from the INTL TORINO-CASELLE weather source, located approximately 8.3 miles from Turin and marked by a latitude of 45.18° N and a longitude of 7.65° E. The training set is 580 KB in size and contains 8,760 samples – the equivalent of one year of data. The specifications used in the PVWatts Calculator for Turin include:

- System size: 1 kW direct current (DC). Therefore, the capacity of the system is 1 kWp since the kWp (kilowatts-peak) refers to the maximum power output the system can generate under standard test conditions.
- Module type: standard. Thus, it utilizes crystalline silicon cells.
- Array type: fixed with an open rack design. This setting assumes a static placement of the PV modules without any tracking mechanisms that adjust with the sun's movement throughout the day.
- Estimated system losses: 14.08%, which accounts for performance losses

[3]`https://nsrdb.nrel.gov/data-sets/tmy`

expected in a real system.

- Array tilt: 20°. It represents the inclination angle of the photovoltaic modules concerning the horizontal plane. In the context of a fixed array, this angle serves as the standard or default position.
- Array azimuth: 180°. It refers to the angle measured clockwise from true north, indicating the direction in which the array is oriented. In the case of a south-facing array, an azimuth angle of 180° is the default setting.
- Inverter efficiency: 96%. It represents the standard nominal efficiency for converting DC (direct current) to AC (alternating current).

The decisions guiding the specifications for the dataset collection were rooted in the necessity of establishing a generic dataset for calculating the emissions of an algorithm applied within the domain of time-series solar energy prediction. Each specification was intentionally kept general to generate a baseline dataset fitting the overarching objectives of this research. For instance, opting for an open rack design reflects a deliberate choice aimed at accommodating diverse installation environments. This configuration, permitting unrestricted airflow around the PV modules, suits generalized applications where the precise installation location, whether ground-mounted or rooftop, remains unspecified. It's a versatile setup designed to facilitate efficient air circulation around the modules, aligning with the broader scope of this research's objectives in time-series prediction while ensuring adaptability across various deployment scenarios.

The problem's target and features were meticulously selected to capture key variables influencing solar energy generation within the Turin area. The selected features can be broadly categorized into three main groups, each representing a distinct aspect of the solar energy generation environment.
These categories include:

1. **Solar Radiation:**
   - Beam Irradiance (W/m^2)
   - Diffuse Irradiance (W/m^2)
   - Plane of Array Irradiance (W/m^2)
2. **Temperature Conditions:**
   - Ambient Temperature ($°C$)
   - Cell Temperature ($°C$)

3. **Wind Dynamics:**
    - Wind Speed (m/s)

Additionally, the temporal aspect is represented by a general feature:

- **Temporal Feature:**
    - Month

This categorization provides a comprehensive view, distinguishing between solar radiation, temperature conditions, wind dynamics, and the temporal influence of the month, all crucial aspects in understanding the patterns of solar energy generation in the Turin area. The target variable, "AC System Output (W)", was the focal point, representing the actual power output generated by the system under these varying meteorological conditions. This selection aimed to capture and predict the system's real-world performance, essential for evaluating the algorithm's predictive accuracy in forecasting solar energy production.

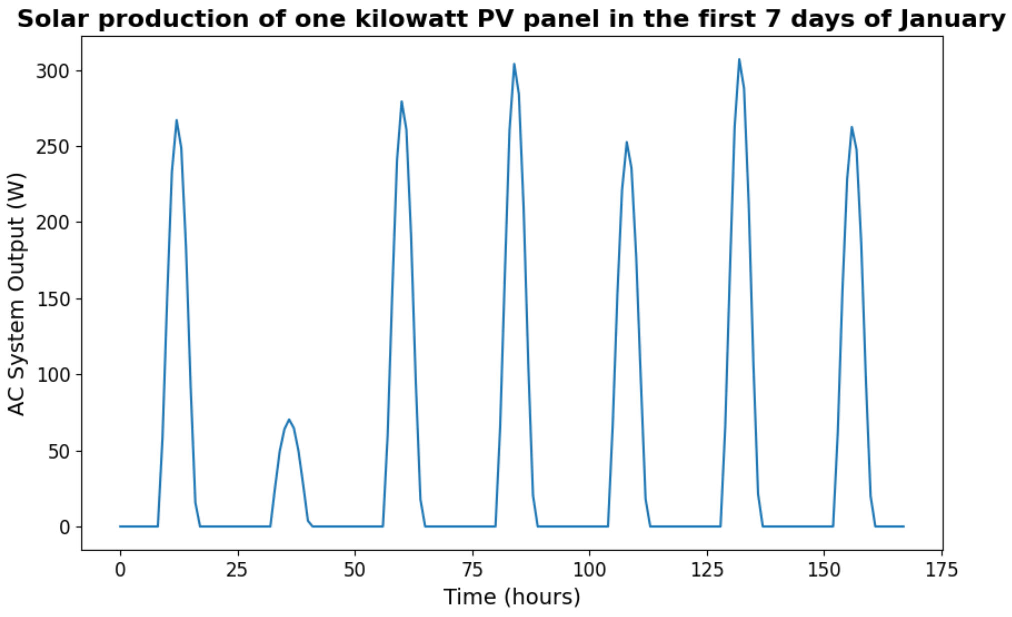


Figure 3.3: AC System Output in Watt obtained from the first week of January of the PV panel production data.

In examining the dataset, an illustrative Fig 3.3, was extracted to showcase the actual AC System outputs obtained from the initial week of January. This plot directly represents the values retrieved from the dataset, offering a tangible representation of real-time production estimated with the tool. As can be seen from this plot, the maximum output is approximately 300 W, which is a consistent observation throughout most days. However, specific days exhibit notably lower output levels, as exemplified on the second day in the figure, indicating potential fluctuations attributed to varying weather conditions. Naturally, the graph demonstrates a complete absence of production during nighttime hours, aligning with the expected absence of solar energy generation during these periods. This picture highlights the impact of weather dynamics on solar energy output by providing a clear understanding of the daily changes and the order size of peaks.

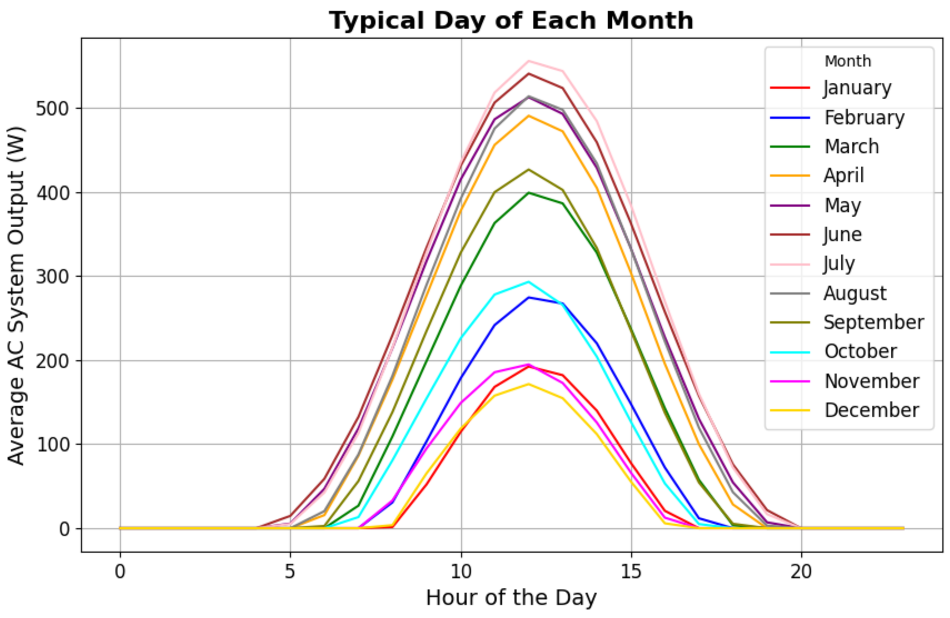


Figure 3.4: Variation in PV production across months: average production days per month highlighting shifts in active hours and maximum output between seasons.

In order to get a full understanding of the dataset, another plot was generated showing the typical days for each month, as reported in Fig. 3.4. Each typical day was derived by averaging the production values across all days within a specific month. This representation aimed to elucidate how PV production varies concerning distinct periods, considering both the duration of production hours (active hours) and the maximum output during the day. Notably, this depiction unveiled significant fluctuations in production characteristics. The plot highlighted substantial shifts in production characteristics, varying from about 200 W to 500 W, more than doubling between seasons. Such fluctuations are expected due to the substantial variation in solar intensity between winter and summer seasons. Still comparing seasons, from summer to winter, also the active hours decreased by almost half. This reduction demonstrates the seasonal impact on the output: production occurs solely during daylight hours.

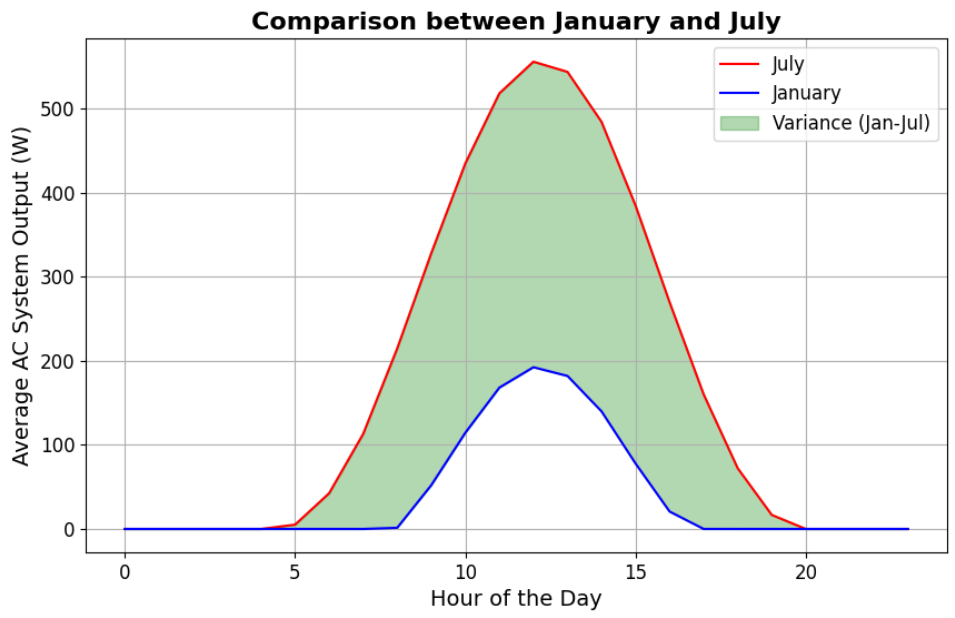


Figure 3.5: Contrasting PV production in summer (July) and winter (January) months by highlighting stark differences in solar output and active hours, representing seasonal variations.

To underscore these differences, another figure zoomed in on average days in July and January, as depicted in Fig. 3.5. By highlighting these two distinct months, typically representative of summer and winter, respectively, the plot accentuates the stark difference in solar energy generation between these seasons. Furthermore, to emphasize this disparity, the area between the curves representing July and January was colored (green), aiding in visually highlighting the substantial differences in solar energy production between these contrasting seasons.

The problem entails predicting the energy production at time t, leveraging the given features as inputs. Let $\mathbf{X}$ represent the matrix of features, including “Month”, “Day”, “Beam Irradiance”, “Diffuse Irradiance”, “Ambient Temperature”, “Wind Speed”, “Plane of Array Irradiance”, and “Cell Temperature”. The target variable, denoted as $\mathbf{Y}$, is the “AC System Output”.

The parameter sl determines the sliding windows’ size, determining the historical data used for prediction. This means that in order to predict the energy output at time t, the model utilizes the feature values from $t - sl$ to $t - 1$.

Overall, given a dataset with N samples and M features, where $\mathbf{X} \in \mathbb{R}^{N\times M}$ represents the feature matrix and $\mathbf{Y} \in \mathbb{R}^{N}$ denotes the target vector, the goal can be represented with a predictive model $f(\cdot)$ trained to estimate the energy output:

$$\hat{y}_t = f(\mathbf{X_{t-sl:t-1}}) \tag{3.4}$$

This formulation adopts a sequential approach, employing past data within the sliding window to forecast the energy output at the current time t. The objective is to minimize the discrepancy between the predicted values $\hat{\mathbf{Y}}$ and the actual energy production values $\mathbf{Y}$ over the dataset by defining a loss function $\mathcal{L}$:

$$\underset{f}{\text{minimize}}\ \mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) \tag{3.5}$$

This optimization process aims to train the model $f(\cdot)$ to accurately predict the energy production given historical feature data. The specific loss function could be the MAE, Mean Squared Error (MSE), etc.
To comprehensively assess the model's performance and facilitate result comparison, the MAE is used as the evaluation metric. The MAE calculates the average absolute differences between the predicted $\hat{\mathbf{Y}}$ and actual $\mathbf{Y}$ values over n data points, enabling a robust estimation of the model's predictive accuracy. The specific formulation is:

$$\mathcal{L}_{MAE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{Y}_i - \hat{\mathbf{Y}}_i| \tag{3.6}$$

In the context of forecasting network traffic $N_{b,t}$ at a specific base station b, the problem is univariate, focusing on predicting a single variable (traffic volume) over future time steps based on its past values. This univariate approach simplifies the prediction task, considering only one target variable without incorporating relationships with other variables.
On the other hand, forecasting the energy production of a PV panel involves a multivariate problem. It encompasses predicting the energy output considering various influencing factors such as irradiance, temperature, wind speed, and others. This multivariate nature involves analyzing and forecasting the relationship between multiple input variables (features) and the target variable (energy production), capturing the complex interplay between these factors to predict the panel's performance accurately.

Chapter 4

Methodology

This study aims to monitor the carbon emissions of NNs and evaluate how emissions trends relate to accuracy variations. A critical first step is to establish a methodology to reliably calculate the carbon footprint of model development. When quantifying emissions, several factors must be considered, including the hardware platform, training hyperparameters, and network architecture.
To monitor NN carbon emissions, this work utilizes CodeCarbon[1] – an open-source tool for calculating energy and emissions from code execution (refer to Section 2.3.3). CodeCarbon samples the power draw of hardware during training and calculates total energy based on high-frequency measurements. Standard emissions factors then convert energy values into carbon equivalents.

Through this methodology, the study aims to identify how training hyperparameters and architectural design patterns differentially impact accuracy and emissions levels. A series of controlled experiments will vary the number of epochs, the size of both training and test sets, the network depth and width, and other factors. Analyzing their effects can provide insight into optimizing networks' sustainability without significantly sacrificing predictive capabilities. Repeated monitoring of emissions improved model versions will quantify sustainability gains from various techniques. The overall goal of this research is to formulate an approach for quantifying carbon emissions from DL-based solutions and correlating trends to maintain high accuracy through careful hyperparameters and design choices. It particularly focuses on the proper handling of timeseries data with diverse characteristics.

[1]`https://codecarbon.io/`

4.1 Monitoring Carbon Emissions

As already explained in Section 2, anthropogenic climate change poses a serious global threat due to human-induced alterations to the carbon cycle through GHG emissions such as CO_2 [38]. According to the IPCC, excess CO_2 in the atmosphere absorbs and re-emits longwave radiation, increasing surface temperatures and shifting climatic trends. In many countries, energy production is a primary contributor to national CO_2 emissions [80]. In the DL scenario, as in many other domains, the global goal of lowering carbon emissions conflicts with the huge quantity of energy required to develop complicated and well performing models [26].

To explore sustainable AI, this research employs an empirical approach based on experiments. Firstly, to monitor and reduce carbon emissions in DL algorithms, it is essential to understand the energy consumption patterns during the various phases: the training and testing processes.
By finely instrumenting the training code, it is possible to attribute energy consumption to certain elements such as architectural components and hyperparameters by tracking emissions at the algorithmic level. This granular profiling can reveal the most energy-intensive phases of model training. With this insight, it becomes possible to optimize algorithm configurations and hyperparameter choices to reduce carbon footprint without sacrificing performance.

Repeating the monitoring of optimized training versions allows quantifying emissions savings from different techniques aimed at decreasing carbon emissions during computationally intensive phases. Over time, this data-driven process helps establish best practices for developing carbon-efficient DL and guide technical advances towards sustainability. With an enhanced understanding of algorithmic energy consumption patterns and their drivers, the field can work to reduce anthropogenic warming impacts through continuous optimization at all levels, from hardware to hyperparameter choices.

4.1.1 CodeCarbon

CodeCarbon is a Python package that is designed to estimate the CO_2 emissions from code execution. It considers the computing resources used, whether on cloud infrastructure or personal devices and it calculates energy usage by accessing the RAPL files or by searching in a list the TDP associated with the CPU model [56]. Power usage is also incorporated for machines with Nvidia GPUs supporting the NVIDIA System Management Interface. Consumption from the CPU package domain(s) and any GPU power comprise the total measurement.

To determine emissions, the tool uses the GeoJS API[2] to get the user's location and corresponding energy mix to calculate a CO_2 intensity in kg per kWh. If the location is unknown, the tool defaults to world averages. For the US, state-level eGRID[3] data directly provides emissions rates. Internationally, the tool reverse-engineers fuel-specific CO_2 formulas from eGRID to apply each country's energy mix. This consistency improves accuracy for electricity domains. In order to reflect also the energy lost to heat during usage, power supply efficiency is taken into consideration. Users can specify efficiency if known, otherwise, the tool defaults to the minimum 80% efficiency certified by the 80-Plus program.
The eGRID data provides state-level energy production and carbon emissions for fuels like coal, oil, and natural gas. CodeCarbon's goal is to calculate kg of CO_2 emitted per MWh for each fuel. It converts emissions values from metric *tons* to kg and divides by energy production values to determine emission intensities, using the calculations:

$$\text{metric tons } CO_2 \times 1{,}000 = \text{kg } CO_2 \tag{4.1}$$

$$\text{Emissions} = \frac{\text{kg } CO_2}{\text{MWh}} \tag{4.2}$$

This allows for measuring international grid carbon footprints based on reliable energy production values.

4.2 LSTM-Based Prediction

In addressing both network traffic prediction and PV panel energy forecasting, the LSTM architecture emerges as a suitable choice because of its ability to handle sequential data, making it particularly useful for timeseries forecasting tasks. Despite the distinct characteristics of the problems – one being univariate, focused on network-related predictions, and the other multivariate, concerning photovoltaic panel energy forecasts – the shared temporal nature of the data aligns seamlessly with the LSTM's strengths. Its inherent ability to retain long-term dependencies and model sequential information proves beneficial for capturing patterns and relationships present in timeseries datasets, contributing significantly to the accurate prediction of future values in both scenarios. This capability aligns well with the characteristics of both datasets, where temporal patterns and dependencies significantly impact future observations.

[2] `https://www.geojs.io/`

[3] `https://www.epa.gov/egrid`

In PV panel forecasts, solar energy production is influenced by daily and seasonal patterns. The amount of sunlight received, which directly impacts energy generation, varies throughout the day and across seasons. Therefore, understanding these temporal patterns, such as the diurnal variations or seasonal changes in solar radiation, becomes crucial for accurate predictions.
Similarly, in network traffic predictions, the behavior of network usage exhibits temporal dependencies. The volume of traffic passing through a network at any given time is influenced by recurring patterns, such as peak usage hours during the day, fluctuations based on weekdays versus weekends, or periodic events that trigger increased traffic (like promotions, holidays, or scheduled updates). These patterns create dependencies where the current state of the network traffic is closely linked to its previous states, making past observations crucial for predicting future network utilization accurately.
In both cases, these dependencies arise from the inherent characteristics of time series data, where each observation is not only influenced by its immediate past but also by its historical trends. Capturing and understanding these temporal patterns is vital for predictive models to extrapolate future behavior accurately. Therefore, the adaptability of LSTM models, with their ability to capture and learn from temporal dependencies, is well-suited for handling such datasets.

Specifically, the Adam [81] optimizer is chosen in order to leverage its efficiency in handling extensive NNs, crucial for fast convergence across diverse datasets. In terms of parameter initialization, the Glorot uniform initializer (Xavier initialization) was utilized for kernels, ensuring stability, while biases were set to a constant value of zero. The Glorot uniform initializer[4,5] computes the bounds of the uniform distribution for initializing weights based on the number of input and output neurons in a given layer. By maintaining a controlled variance in weights, it aids in stabilizing the learning process, allowing gradients to propagate effectively through the network during backpropagation. This initialization method aims to tackle the vanishing or exploding gradient problem often encountered during training.
The inclusion of a Dense layer with a linear activation function for the final output aligns with the regression nature of the task, ensuring continuous predictions. Employing the MAE as the loss function offers a robust measure of prediction error, well-suited for regression problems. Additionally, incorporating accuracy as a metric provides a holistic evaluation of the model's performance. Each architectural decision was made with the intent to optimize adaptability, convergence speed, stability, and evaluation accuracy in the predictions, considering the forthcoming experiment specifications.

[4]`https://pytorch.org/docs/stable/nn.init.html`

[5]`https://keras.io/api/layers/initializers/`

4.2.1 Python libraries comparison

Following the initial model setup, a critical analysis centered around the LSTM architecture is conducted. This exploration aims to compare the environmental impact, specifically in terms of emissions, arising from the same architecture developed using two distinct Python libraries: Keras[6] and PyTorch[7]. The underlying motivation for this comparative study is the critical necessity to make sure that the library selection used to build the model does not substantially influence or bias the results of any further analysis or conclusions that are reached. By establishing this comparative study, the purpose was to determine whether any observed disparities or trends in emissions were primarily attributed to the inherent design or configuration differences between the libraries rather than the fundamental architectural choices within the model. This comparative approach was adopted as a crucial step toward ensuring the robustness and reliability of the subsequent analyses and findings related to energy consumption and emissions.

4.3 Handling Negative Predictions in PV Panel Dataset

In the context of the PV Panel dataset, a post-processing step is introduced to enhance the practicality and reliability of the predictions generated by the LSTM network. The nature of energy production outputs from photovoltaic panels makes negative predictions unrealistic and impractical. To address this issue, a manual adjustment is implemented during post-processing, where any negative values predicted by the model are rectified to zero. This corrective step aligns the predictions with the physical constraints of the domain, ensuring that the model's outputs remain within the bounds of feasibility. By explicitly setting negative predictions to zero, the post-processing technique enhances the interpretability and applicability of the model's results in the context of solar energy production.

4.4 Manipulating Training Hyperparameters

In the initial phase of this methodology, the strategic manipulation of training hyperparameters emerged as the easiest, but effective, approach. Hyperparameters are important configuration settings influencing the learning process of the model, both in terms of time and performance.

[6]`https://keras.io/`

[7]`https://pytorch.org/`

Firstly, the focus revolves around elucidating the impact of hyperparameters on energy consumption and, consequently, the resulting carbon emissions. By recognizing that certain hyperparameters exhibit a strong correlation with the duration of model training, this methodology targeted these variables as potential levers to influence overall consumption. In fact, this study is anchored in the premise that a direct correlation exists between carbon emissions and computational time, an association inherently linked with energy consumption. This rationale is deeply rooted in the understanding that changes in the computational demands directly impact the amount of energy used (energy is dependent on both power and time, see Section 2.3.1), a relationship that also affects the carbon emissions left behind in the environment (explained to Section 2.3.2). Therefore, in order to test the validity of these hypotheses, hyperparameters were carefully adjusted. By changing hyperparameters, such as the training set size, the study aimed to identify trends in energy use and carbon emissions.

Secondly, the investigation involves a comparative analysis between carbon emissions and the performance output of the models. The goal is to identify particular scenarios in which the increase in carbon emissions was disproportionately greater than the improvements in performance indicators. By conducting this comparison, the study sought to identify critical junctures in the model's development where the pursuit of better performance significantly amplified carbon emissions. This approach aimed to highlight trade-offs in the model's advancement, emphasizing situations where optimizing for superior performance led to an unsustainable ecological impact. Overall, this comparison will guide the formulation of strategies that balance performance gains with environmental considerations more effectively.

The whole investigation seeks to decode the intricate interplays of different factors in order to facilitate the optimization of DL models not just in terms of performance metrics but also in tandem with carbon emissions reduction.

4.4.1 Epochs and Input Sequence Length Ablation

This subsection delves into the intricate relationship between training hyperparameters, specifically epochs and input sequence length, and the emissions of model training. The number of epochs, representing how many times the learning algorithm will run through the whole training dataset, directly influences the convergence and refinement of the model. By definition, a single epoch gives every sample in the training dataset the ability to change its internal model parameters (weights). A higher number of epochs often leads to increased model accuracy but comes at the cost of longer training times.
Conversely, the input sequence length, depicting the temporal context provided to

the model, significantly impacts the network's ability to capture long-range dependencies within the data. Modifying this parameter affects the amount of historical information fed into the model, subsequently affecting the model's capability to learn complex patterns. The investigation's focus first centers on adjusting the input sequence length by initially altering the training set size while keeping the test set size constant. Subsequently, variations in the test set size have been introduced, consequently impacting the training set size as well. In the first scenario, the primary emphasis is on adjusting the amount of historical data available for the model to learn from. By varying the training set size, the model's exposure to historical patterns is manipulated, influencing its ability to comprehend complex temporal dependencies. Conversely, in the second scenario, modifications are introduced to the test set size, consequently impacting the training set size. Here, the focus extends beyond solely adjusting the historical data available for training. It involves altering the partitioning between training and testing datasets, which affects the model's understanding of unseen data. This shift can provide insights into how variations in the testing dataset influence the generalization ability of the model, consequently influencing the temporal aspects of model training and its corresponding emissions.

These hyperparameters have been scrutinized, emphasizing their intrinsic relationship with the duration required for model training, and therefore, their emissions. This correlation is carried on through a meticulous exploration of varying these hyperparameters across a spectrum of values. By delving into a broad range of parameter values, the objective is to extract a clear trend highlighting how changes in epochs and input sequence length directly influence the temporal aspect of model training, and thus, the associated carbon emissions.

4.5 Architectural Model Design

The second principle guiding our analysis was centered on the implementation of sustainable model architecture as an additional strategy for reducing carbon emissions. This revolved around the modification of critical components influencing the structure of the LSTM model. Specifically, the focus is on the manipulation of two key elements: the number of layers within the network architecture and the quantity of nodes within each layer. These architectural elements play a pivotal role in determining the model's complexity and capacity to capture intricate temporal patterns. By altering the number of layers, we concurrently explore how the depth of the network influences its capacity to understand temporal dependencies across various hierarchical levels, examining its impact on both model performance and the corresponding trend in carbon emissions. Additionally, adjusting the nodes within each layer sought to explore the impact of local feature extraction and abstraction within the temporal context. A larger number of nodes within a layer can potentially enable the model to detect more nuanced and intricate patterns within the sequential data. This is due to the increased capacity of the network to process and extract diverse features at a more granular level, allowing for finer distinctions and more detailed representations of the underlying data. This exploration was undertaken to comprehend the trade-offs between model complexity and computational efficiency in terms of energy consumption.

By designing models that are computationally efficient and require fewer resources, we can significantly lower carbon emissions. It seems reasonable to analyze layer size and quantity variations and find optimal configurations that balance performance and sustainability.

4.5.1 Layers and Nodes Ablation

In the context of optimizing for reduced carbon emissions, a critical aspect under examination involves investigating the impacts of altering the number of layers and nodes within the LSTM architecture. These hyperparameters were selected based on their fundamental role in shaping the model's complexity and capability to capture intricate temporal patterns, which directly relate to the energy consumption of the system. The number of layers determines the depth of the network, influencing its ability to understand temporal dependencies at multiple hierarchical levels. Similarly, the quantity of nodes within each layer directly affects the model's capacity for local feature extraction and abstraction within the temporal context. By investigating these specific hyperparameters, the objective is to discern their individual effects on model performance and corresponding trends in energy consumption. The analysis conducted is strategically designed to discern the individual impacts of varying the number of layers and nodes within the LSTM architecture. In fact, to accurately attribute the influence of each parameter, the investigation was performed separately for these two hyperparameters. In one scenario, while exploring the impact of altering the number of layers, the number of nodes within each layer was kept constant. Conversely, when examining the effects of adjusting the number of nodes, the number of layers remained fixed. This segregation allowed for a focused evaluation, aiming to isolate and understand the direct energy consumption.
Overall, this exploration aims to identify optimal configurations that strike a balance between model efficiency, performance, and sustainability, ultimately contributing to a reduction in carbon emissions.

4.6 Tailored Problem Formulations

The preprocessing and structuring of data play a pivotal role in shaping the efficiency and effectiveness of predictive models, particularly in the realm of minimizing computational resources for training and keeping baseline performances. This section delves into the preprocessing strategies tailored for distinct datasets, where the techniques adopted are tailored to the specific nuances of each problem, exemplifying the importance of data structuring in achieving optimal modeling outcomes.

Network Data Problem: Data Aggregation
In addressing the network data-related issue, where the emphasis lies on univariate predictions for individual base stations within specific zones, the approach begins by acknowledging the initial formulation of the problem. Originally, the problem was defined by considering data acquired from each base station separately, resulting in the creation of a distinct model for each station. While this approach yielded high accuracy, it posed potential inefficiencies in terms of emissions due to the creation of numerous models. Recognizing this, the solution incorporates a strategic shift towards data aggregation. The concept of data aggregation involves combining information retrieved from multiple base stations within the same zone before constructing the LSTM networks. This aggregation aims to merge the data from various stations within a specific area, effectively simplifying the input dataset for the LSTM model. The underlying objective is to condense the number of models to be built, streamlining computational resources, and redirecting focus toward zone-specific patterns rather than station-level intricacies. This shift in approach is motivated by the aspiration to enhance the efficiency of the models in terms of emissions while maintaining a reasonable level of accuracy.

PV Panel Problem: Feature Selection and Historical Data Consideration
In the case of the PV panel problem, characterized by multivariate predictions based on various available features, the emphasis shifts toward feature selection. Here, the forecast relies on multiple input features, and the objective is to identify and select the most relevant ones for prediction. The strategy involves a meticulous examination and selection of features deemed crucial in determining solar energy output. Feature selection is supposed to be a valuable strategy for reducing emissions produced during the training of the model. By focusing on a subset of the most relevant features, the model is provided with a streamlined set of information, potentially leading to more efficient training processes and consequently lower emissions. Additionally, the impact of feature selection on accuracy is a crucial consideration. While reducing the number of features, there is even the possibility that the model's performance might be enhanced. This is because the selected features are anticipated to capture the essential patterns and characteristics

influencing solar energy output, potentially resulting in a more refined and accurate predictive model. Thus, feature selection not only offers a pathway for emission reduction but also holds the potential to improve the overall accuracy of the predictive model.
Specifically, this feature selection process is conducted within the framework of the three distinct categories into which the features are subdivided: solar radiation, temperature conditions, and wind dynamics. By isolating and incorporating only the most pertinent features from these categories, this approach aims to refine the predictive model, concentrating computational resources on the most influential factors governing energy production. This strategic selection process ensures that the predictive model focuses specifically on the key aspects within each category, contributing to a more nuanced and accurate representation of the intricate interplay between solar radiation, temperature conditions, and wind dynamics in determining solar energy generation.

In the context of the PV panel dataset, where data stability is notably consistent, a second modification in the problem formulation involves manipulating the sliding window configuration. This approach serves as an alternative study with respect to the initial feature selection strategy. The objective here is to scrutinize the relationship between accuracy and emissions by varying the historical data considered by the model. By adjusting the sliding window, the amount of historical information provided to the model is modified, potentially influencing its understanding of temporal patterns.

In both scenarios, the adjustment in the problem formulation aids in simplifying the model structure, focusing computational resources on the most critical aspects of the prediction tasks, and ultimately contributing to a more emissions-efficient modeling approach.

Chapter 5

Experimental results

5.1 Experimental Setup

The LSTM neural network architecture is used for modeling both the network dataset and the PV panel dataset. The experimentation initially involves assessing the impact of utilizing different programming environments for model implementation, specifically between Keras and PyTorch. Subsequently, Keras is selected for all experiments, but it is worth noting that both Keras and PyTorch exhibited equal applicability for the task. The computations pertaining to carbon emissions are carried out using the CodeCarbon python library, enabling accurate quantification of emissions resulting from the computational processes involved in model training and inference. Italy is selected as specific location when running the tracker.
Every experiment and computation is performed on a personal computer equipped with an 11th generation Intel Core i7-1165G7 CPU operating at 2.80GHz constant consumption mode, with 16 GB of available memory. Contrarily, the GPU is not present and therefore not tracked. All the specifications are reported in Table 5.1.

Component	**Model**	**TDP**
CPU	11th Gen Intel Core i7-1165G7 @ 2.80GHz	28 W
GPU	Not Found	-

Table 5.1: CPU and GPU specifications.

The following experimental configurations are exclusively dependent on the two distinct datasets: network data and PV panel data. Each dataset represents a unique problem domain, thus necessitating specific setups and baseline settings for the LSTM models employed in these experiments.

Network Data Configuration
Regarding the network data, the baseline LSTM structure, drawn from a referenced paper [76], involves deploying an LSTM model for each base station. Each model consists of 5 LSTM layers and a final Dense layer. The predictions made for the forthcoming 4 samples, which encapsulate an hour's worth of data, are based on insights drawn from the 10 preceding samples – considering the definition given in Section 3.1 this means $k = 10$. These 10 samples cover a temporal span of 2.5 hours, acting as the input window that the model utilizes to forecast the subsequent hour's data. The testing period is fixed at 14 days ($14 \cdot 24 \cdot 4 = 1344$ rows, considering that each dataset row represents 15 minutes), while the training period spanned 46 days (corresponding to 4416 rows). The number of epochs is set to 500 as an additional training hyperparameter, and 64 nodes per layer are also taken into consideration for the architectural design. Lastly, 0.001 is chosen as the Adam optimizer's learning rate.

In the following experiments, the pre-established baseline configuration is systematically changed in order to understand the effect of changing hyperparameters and design characteristics.
In the pursuit of manipulating training hyperparameters, a series of experiments unfolds with distinct variations:

- Epochs Variation: A range of epochs [50, 100, 250, 500, 1000] constitutes the focal point of the initial experiment. The objective here is to observe the model's performance and emissions concerning different epoch counts while keeping the other specifications constant in each trial, adhering to the baseline's settings (for instance the training and testing days are steadfastly set to 46 and 14, respectively).
- Testing Days Variation: A range of testing days [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] becomes the subject of the second experiment. This analysis delves into varying the testing duration while adapting the training days accordingly to encompass the entire dataset. The adjustment in testing days is paired with a corresponding change in training days to maintain the entire dataset's inclusion in the training process. This approach ensures that the model considers the entirety of the available data, enabling a comprehensive assessment of its adaptability and accuracy across varied temporal scopes. All other specifications remain constant, aligned with the baseline setup.
- Training Days Variation: In the third experiment, the focus lies on a range of training durations [7, 14, 21, 28, 35, 42, 46]. This analysis examines how the model's learning capacity and stability evolve across varied training durations while maintaining a consistent testing period, as set in the baseline. Notably, when using less than 46 days for training, the model operates on a subset of

the available dataset, producing a temporal gap between training and test data. The testing duration remains unaltered, ensuring a fixed evaluation period across all training durations, allowing a focused exploration of the model's learning behavior and performance stability. All other specifications align with the settings defined in the baseline.

Other experiments are taken into consideration in the context of modifying the architectural model design:

- Layers Number Variation: The number of layers undergoes variation across a spectrum of values [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] in the first experiment. This exploration aims to discern the impact of different layer counts on model accuracy and emissions.
- Node Count Variation: The second experiment concentrates on varying the number of nodes within each layer, spanning values of [16, 32, 64, 128, 256]. Here, a single layer is maintained to delve deeper into the specific influence of node count variation on the model's performance and emissions.

In the final experiment, the approach shifts from creating an individual LSTM model for each base station to consolidating data from multiple stations within the same zone. This consolidation aims to generalize the problem by designing a single model for each zone, reducing the model's specificity. The experimentation involves also modifying the architecture by reducing the number of layers from 6 to 1 and adjusting the training epochs from 500 to 100. The decision to alter the architecture by reducing the layer count and adjusting the training epochs stems from the findings derived from earlier experiments and a careful evaluation of model performance and emission outcomes. This change in approach permits a broader analysis by shifting the focus from specific base stations to zones, potentially allowing for a reduction of computational resources.

PV Panels Configuration
Regarding the PV panels dataset, the baseline settings are determined through a grid search methodology. The test set size is 25% of the whole dataset. Given that the dataset spans 365 days and that each sample lasts for one hour, the test set comprises 25% of the dataset ($25 \cdot 8760 \div 100 = 2.190$ samples), with the remaining samples (6570 samples) making up the training set. A 24-hour sliding window is established by selecting a sequence length of 24-time steps while considering hourly measurements. The number of epochs is set to 250. Design-wise, in the LSTM architecture, the number of layers is fixed at 3, and the number of nodes at 256. Similarly to the previous dataset, the learning rate for the Adam optimizer remains constant at 0.001.

During the experimental phase, these baseline settings are varied to understand their impact. The following tests are conducted in order to modify the training hyperparameters:

- Epochs Variation: The number of epochs is experimented within the range [10, 50, 100, 150, 200, 250]. Like the network dataset scenario, all other configurations remain consistent with the established baseline settings to isolate the specific influence of the epochs' variation on model performance and emissions.

- Testing Size Variation: The size is varied as a percentage of the total dataset, exploring [5%, 10%, 20%, 25%, 30%, 40%]. As the test set size changes, the training set size is derived as the difference between the total dataset and the test set size.

- Training Size Variation: In the third experiment, while maintaining a fixed test set size at 25%, the training set size is explored across [25%, 35%, 45%, 55%, 65%, 75%] to understand its influence on model performance and emissions.

Furthermore, a number of trials are carried out to investigate the complex aspects of architectural design using LSTM models, within the framework of the PV panels dataset:

- Layers Number Variation: The architecture is investigated by changing the depth of the LSTM network. The number of layers is systematically adjusted, spanning a range [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], while keeping all other model configurations consistent with the baseline setup.

- Node Count Variation: The number of nodes is varied across the range [32, 64, 128, 256, 512, 1024], focusing on a single LSTM layer for this experiment. This experiment concentrates on exploring the effect of node density within a single layer.

The investigation into the impact of historical data is executed by manipulating the sliding window configuration across a range of values: [2, 8, 16, 24, 48, 72, 96]. This deliberate variation in the sliding window alters the temporal context provided to the model, influencing the depth of historical data considered during predictions. Each value within this range corresponds to a distinct temporal span, allowing for a comprehensive evaluation of how different amounts of historical information affect both the model's predictive accuracy and its associated carbon emissions.

5.2 Results

5.2.1 Different Environments

The initial analysis focuses on the experimental environment by comparing the two primary Python libraries for developing neural networks. This comparative study is exclusively conducted for the Business zone of the network traffic data, operating under the assumption that consistent outputs are achievable regardless of the specific dataset employed. Within this scope, the implementation of the LSTM network is done using both Keras and PyTorch frameworks and following the baseline settings. To provide a comprehensive perspective, a detailed Table 5.2 offers insights into emissions and durations during both the training and testing phases. This focused examination within a singular zone aims to establish a robust comparison between Keras and PyTorch in terms of their impact on emissions when implementing the LSTM network. From the results, it is evident that the training phase is notably longer, leading to higher emissions when employing the PyTorch library compared to Keras. Conversely, during the testing phase, the trend reverses. Since the model is evaluated at each epoch, the divergence in training times between the two libraries might stem from variations in metric computation, which is integrated into the training process itself. This difference in metric implementation could result in significantly different training durations. On the other hand, during the testing phase, where only prediction occurs, the significant differences in timing could be attributed to the standard functions used by each library to execute predictions. Consequently, for all subsequent analyses, Keras is consistently utilized. The focus remains on emissions during the training phase, where this library demonstrates significantly lower energy consumption.

Python Library	**Phase**	**Emissions (gCO_2eq)**	**Duration (seconds)**
Keras	Training	0.2961	254.62
	Testing	0.0187	15.85
PyTorch	Training	4.5206	3858.17
	Testing	0.0002	0.05

Table 5.2: Comparative analysis of accuracy and emissions during training and testing phases using Keras and PyTorch libraries, focusing on the Business zone within the network traffic dataset.

5.2.2 Baseline Results

The baseline configuration yields diverse outcomes across various metrics for both network traffic data and PV panel data. A comprehensive table showcases various critical measures, including accuracy, emissions, and duration during both the training and testing phases.

Specifically, for the network data, each experiment's result is calculated across all the zones studied, as shown in Table 5.3. This detailed examination under the baseline settings provides a holistic view of the model's performance and the environmental impact. The initial observation reveals a direct correlation between duration and the associated emissions. An examination of the outcomes concerning duration and emissions during the training and testing phases highlights their parallel growth. As the duration extends, a synchronous increase in emissions is notable: this is emphasized by the fact that the training phase's duration is approximately ten times longer than the testing phase. This discrepancy in time directly influences emissions, with the prolonged training duration significantly contributing to a noticeable rise in emitted carbon. Furthermore, the evaluation of accuracy, computed for both the training and testing phases, indicates a consistent trend. The observed accuracy values hover around 0.3 for all zones in both the training and testing phases. This suggests a balanced model performance without significant signs of overfitting or underfitting across the examined zones.

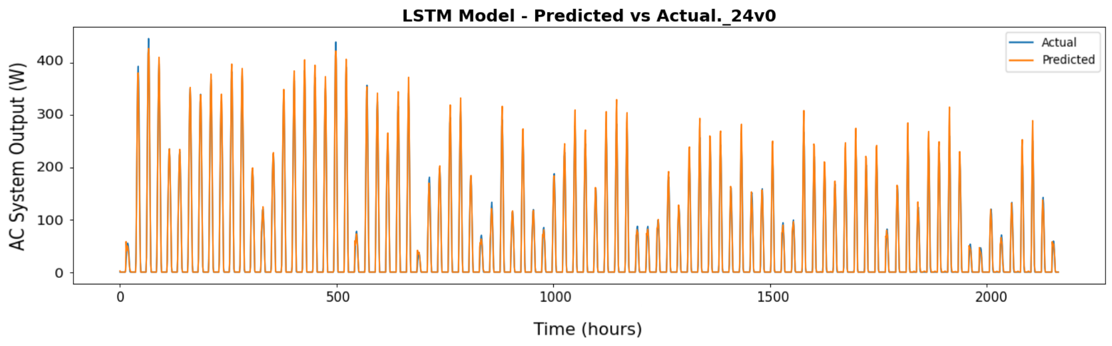


Figure 5.1: Graph depicting the comparative analysis between predicted and actual PV panel system outputs, showcasing the model's performance in forecasting photovoltaic energy generation against the ground truth measurements.

The outcomes for the PV panel dataset are showcased in another detailed Table 5.4, presenting the metrics acquired during the baseline configuration experiments. Additionally, to provide a visual understanding of the prediction accuracy, Fig. 5.1 displays the actual and predicted values. This representation visually demonstrates how closely the predicted trend aligns with the actual data. Furthermore, a

Zone	**Phase**	**Accuracy**	**Emissions (gCO_2eq)**	**Duration (seconds)**
Business	Training	0.304	0.296	254.6
	Testing	0.291	0.019	15.9
Industrial	Training	0.318	0.274	235.3
	Testing	0.300	0.013	11.0
San Siro	Training	0.309	0.262	225.1
	Testing	0.293	0.012	10.5
Rho Fiere	Training	0.312	0.276	236.9
	Testing	0.300	0.013	11.0
Residential	Training	0.317	0.291	250.6
	Testing	0.306	0.019	15.7
Train Station	Training	0.310	0.294	252.6
	Testing	0.300	0.018	15.1
Polimi	Training	0.312	0.418	359.2
	Testing	0.297	0.025	21.0

Table 5.3: Accuracy, emissions, and duration during both training and testing phases across all zones under the baseline setup, concerning the network traffic data.

zoomed-in figure specifically focuses on the initial two days of the dataset (Fig. 5.2), offering a more precise view of the predictions' alignment with the true trend. This remarkable precision in prediction is attributable to the dataset's stability,

Phase	**Accuracy**	**Emissions (gCO_2eq)**	**Duration (seconds)**
Training	0.481	8.427	2314.6
Testing	0.615	0.002	1.8

Table 5.4: Accuracy, emissions, and duration during both training and testing phases, considering the PV panel data and the baseline settings.

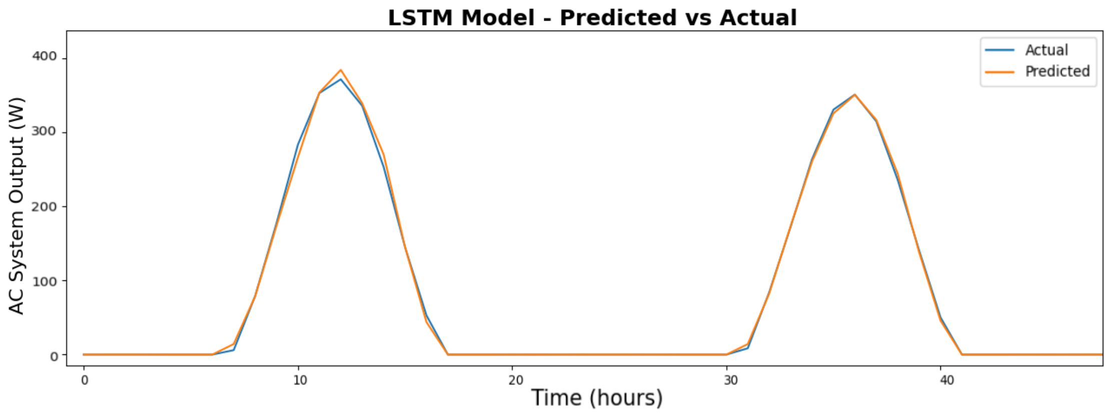


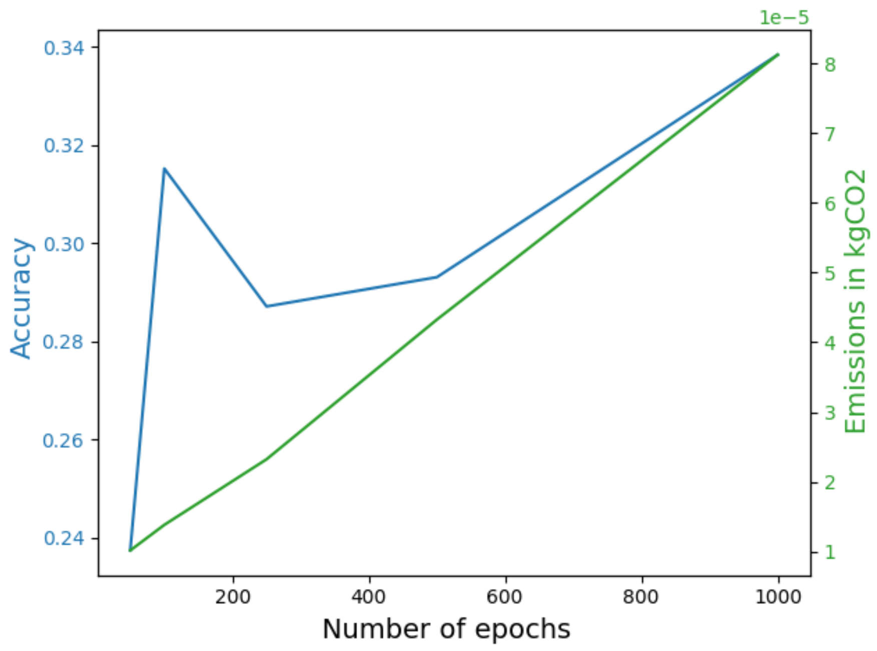
Figure 5.2: Close-up view detailing the initial two days' predictions and actual values of the PV panels system output, offering a more detailed insight into the model's performance within this specific timeframe.

facilitating a more straightforward prediction process due to its consistency.

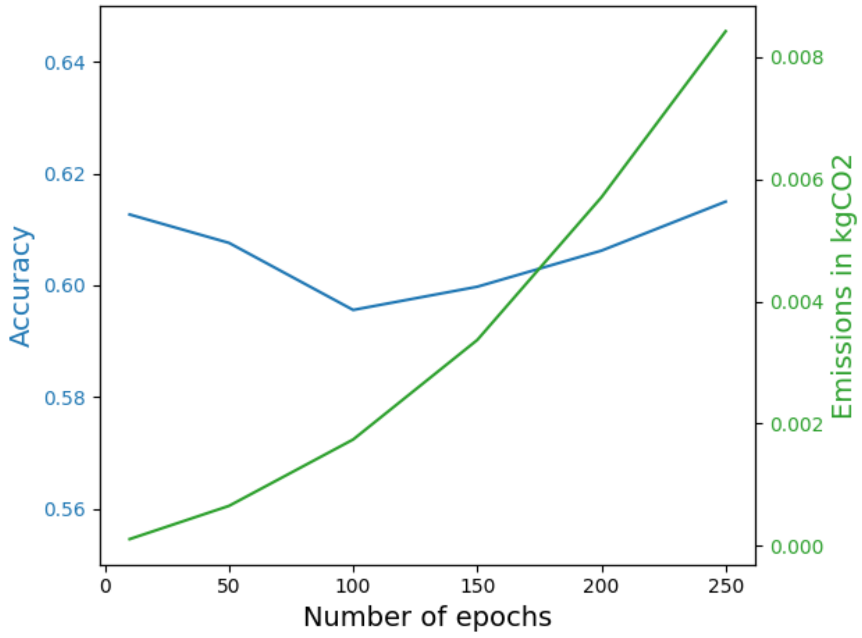
5.2.3 Manipulating Training Hyperparameters

This section delves into a comprehensive exploration of the impacts of varying training hyperparameters on performance and emissions, with a primary focus on the number of epochs, test set size, and train set size. This multifaceted analysis aims to unravel the intricate interplay between these critical factors and their influence on two distinct datasets: the PV panel dataset and the network traffic dataset, with a specific emphasis on the Train Station zone for the latter.

The initial segment of this investigation centers around the manipulation of the number of epochs. To discern the nuanced effects on model performance, both test accuracy and training emissions are meticulously scrutinized. A detailed visual representation of the outcomes is encapsulated in Fig. 5.3, where the x-axis delineates the total number of performed epochs.



(a) Accuracy and Emissions Variation with Epochs for Train Station Zone in Network Traffic Dataset

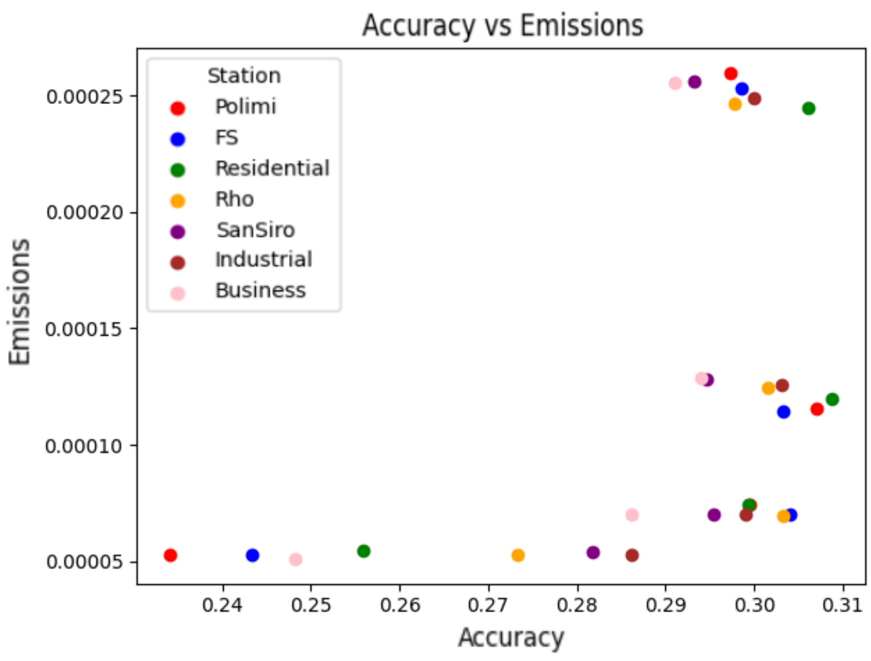


(b) Accuracy and Emissions Variation with Epochs for PV Panel Dataset.

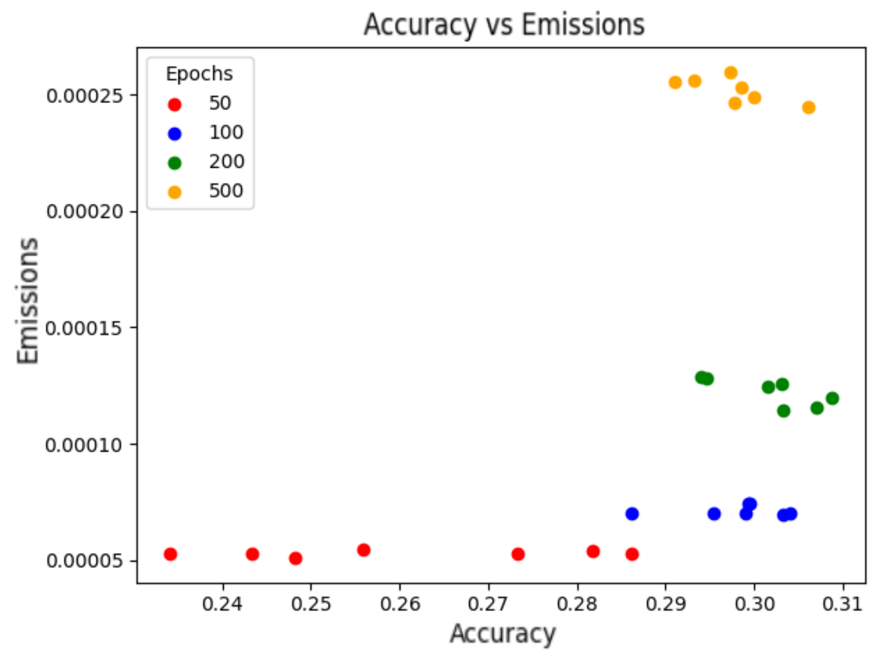
Figure 5.3: Comparison of Accuracy and Emissions Across Different Datasets Varying with Number of Epochs.

Fig. 5.3a encapsulates the findings for the network traffic data, focusing solely on the Train Station zone for the sake of simplicity, while Fig. 5.3b encapsulates the results for the PV panel dataset. This visual comparison serves as a foundational overview, setting the stage for a deeper exploration of the observed distinctions between the two datasets. In both datasets, emissions demonstrate a linear increase with the rising number of epochs. However, the behavior of accuracy varies notably. In the PV panel dataset, accuracy remains almost constant, whereas in the network traffic data, it consistently improves, although with occasional oscillations. Consequently, establishing a balance between accuracy and emissions proves challenging in the latter case. On the other hand, a significant finding is revealed in the PV panel dataset. The difference in accuracy varying the number of epochs is considerably smaller compared to the difference in emissions. This suggests that extended model training leads to higher emissions, without enhancing the performance. This implies that reducing the number of epochs could result in substantial emission reduction with only a slight compromise in accuracy.
In summary, while emissions show a linear growth with increasing epochs in both cases, accuracy exhibits diverse patterns. The network traffic data lacks a discernible balance between accuracy and emissions, whereas for the PV panel dataset, reducing epochs could significantly cut emissions with minimal impact on accuracy.

After comparing the results obtained for the two distinct datasets by varying the number of epochs and analyzing the relationship between accuracy and emissions, a different type of visualization is proposed in order to visually find different model behavioral patterns with respect to the number of epochs. In this alternative visualization, depicted in Fig. 5.4, emissions are plotted on the y-axis, while accuracy is plotted on the x-axis. Each specific number of total epochs tested during experiments is represented by plotting the results obtained for each base station. Two different figures are created for this purpose: Fig. 5.4a and Fig. 5.4b. Fig. 5.4a, employs color differentiation according to zone names, directing attention towards scrutinizing the enhancement within each specific zone rather than the dataset as a whole. On the other hand, in Fig. 5.4b the points are colored based on the number of epochs, allowing for a clear observation that even though accuracy continues to increase, the absolute improvement is significantly smaller compared to the growth in emissions. Hence, even when considering this dataset, it appears plausible to use a reduced number of epochs compared to the one used by the baseline method, i.e., 500 epochs. Moreover, by combining the insights coming from both visualizations, deeper considerations can be made. Firstly, the experiments using a lower number of epochs, i.e., 50 epochs, reveal a higher variance in accuracy between the zones, while for a total number of epochs equal to or greater than 100, the variance noticeably decreases.

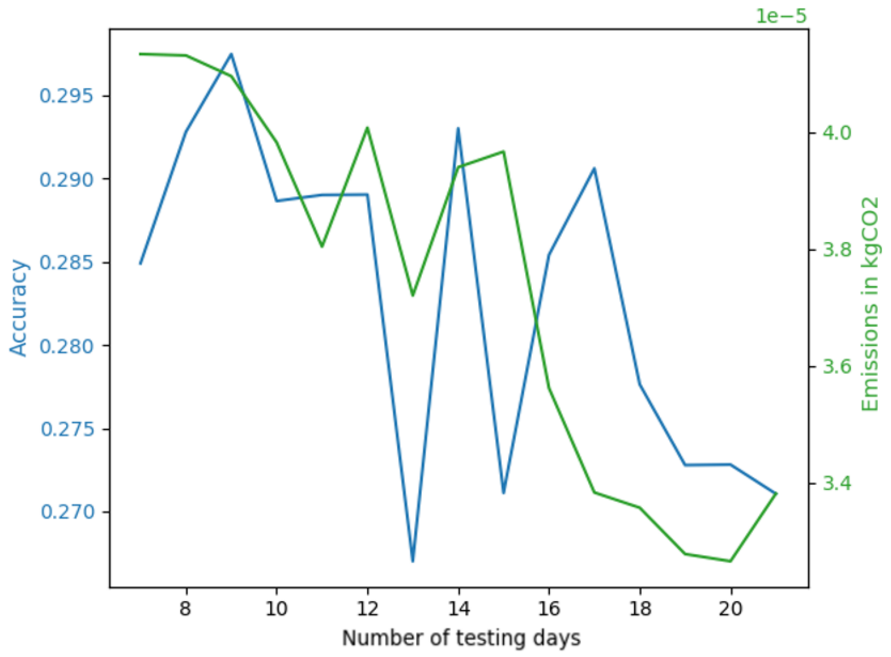


(a) Number of epochs variation by aggregating per zone.

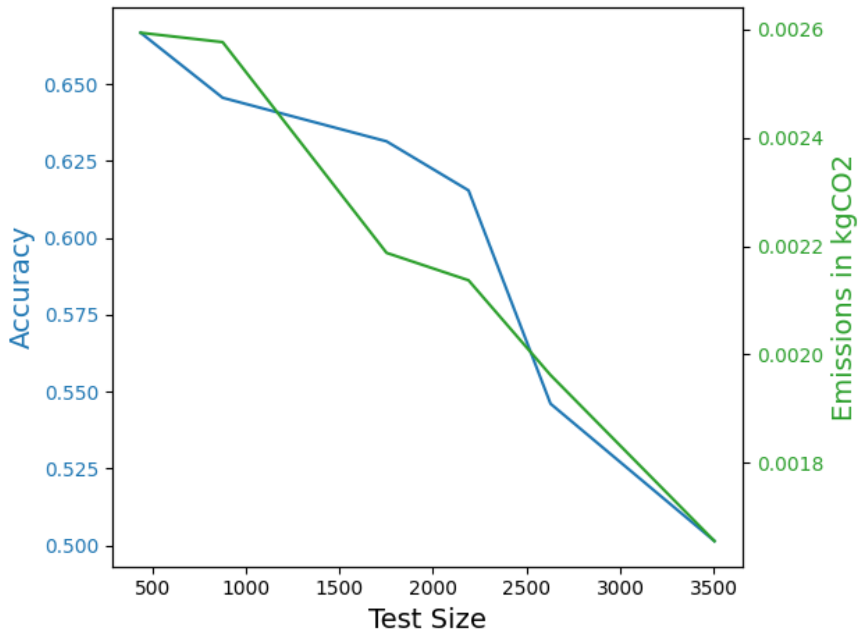


(b) Number of epochs variation by aggregating per epochs.

Figure 5.4: Comparison of accuracy and emissions variations for different epoch aggregations.



(a) Accuracy and Emissions Variation with Number of Testing Days for Train Station Zone in Network Traffic Dataset

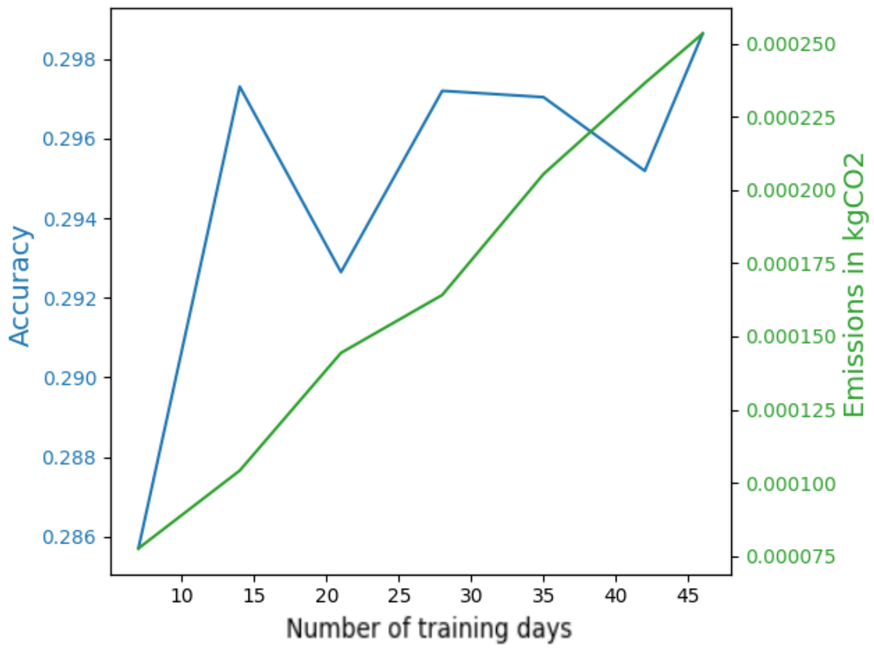


(b) Accuracy and Emissions Variation with Test Size for PV Panel Dataset.

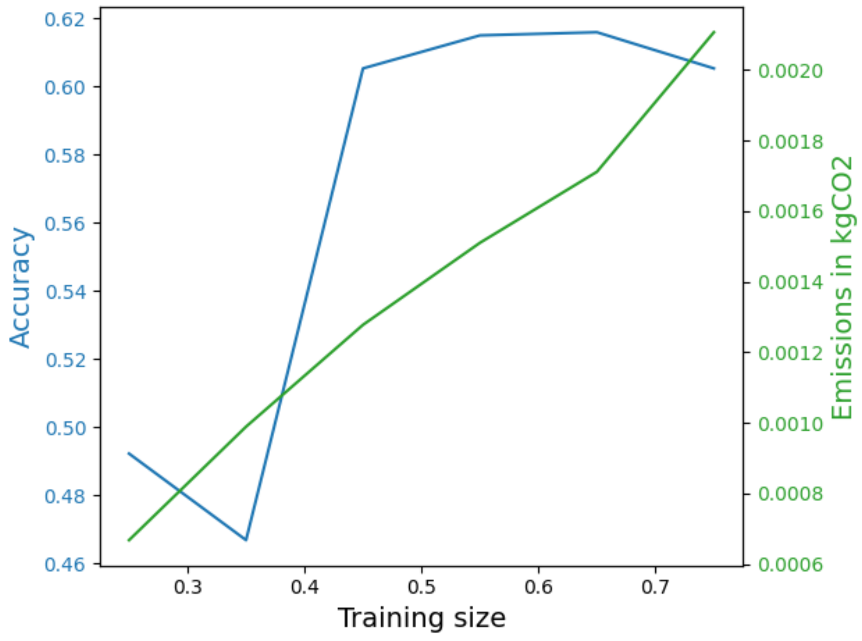
Figure 5.5: Comparison of Accuracy and Emissions Across Different Datasets Varying with Size of Test Set.

This observation led to another important finding: the difference in accuracy improvement between 50 and 100 total epochs varies significantly depending on the zone being considered. This indicates that the same hyperparameter settings may need to be adjusted differently across different zones, providing valuable insights for further investigation. Expanding the exploration of hyperparameter values relevance, an examination of the impact of test set size and train set size becomes crucial. In the initial analysis, focusing on the test size, notable oscillations are observed in both accuracy and emissions when considering the traffic dataset, specifically within the Train Station zone – as shown in Fig. 5.5a. Despite the frequent fluctuations, there is a decremental trend in both accuracy and emissions. Anyway, it is hard to confirm this behavior since the ranges within which these metrics vary are remarkably narrow. Conversely, when scrutinizing the PV panel dataset, a distinct pattern emerges, as represented in Fig. 5.5b. In this case, both emissions and accuracy exhibit a nearly linear decrease in correlation with the size of the test set. This trend is rationalized by the reciprocal relationship between the test and train sizes: as the test set size expands, the train set size contracts proportionately, contributing to the overall decline observed in both accuracy and emissions.

In the subsequent investigation, the focus shifts to the impact of varying training set sizes while maintaining a fixed test set size. These results are reported in Fig. 5.6a and Fig. 5.6b. The findings from both datasets reveal a consistent trend where both accuracy and emissions exhibit an upward trajectory as the training size increases, aligning with the initial hypothesis. This reaffirms and accentuates the results observed in the previous test size ablation, shedding light on the heightened significance of this relationship. The increased emphasis may be attributed to the creation of a temporal gap between the training and test phases when reducing the training size. Indeed, the temporal link within the data emerges as a pivotal aspect, especially in the context of time series analysis. In the realm of time series, the chronological order of data points is fundamental for capturing patterns and trends inherent in temporal sequences. When varying the training size and introducing a potential temporal gap between the training and test phases, the intricate temporal connections within the data may be compromised. The essence of time series analysis lies in comprehending how past events influence future occurrences. In the context of machine learning models, maintaining a robust temporal link during training becomes imperative to enable the model to glean meaningful insights from historical data and generalize effectively to unseen future instances. The observed increase in both accuracy and emissions with an expanding training size underscores the significance of preserving this temporal continuity, highlighting the nuanced relationship between the temporal structure of data and the performance of machine learning models, particularly in time series scenarios.



(a) Accuracy and Emissions Variation with Number of Training Days for Train Station Zone in Network Traffic Dataset



(b) Accuracy and Emissions Variation with Train Size for PV Panel Dataset.

Figure 5.6: Comparison of Accuracy and Emissions Across Different Datasets Varying with Size of Train Set.

This temporal gap potentially disrupts the coherent evolution of the model, accentuating the importance of maintaining both a sufficient training size and a temporal continuity of the data used during these critical phases.

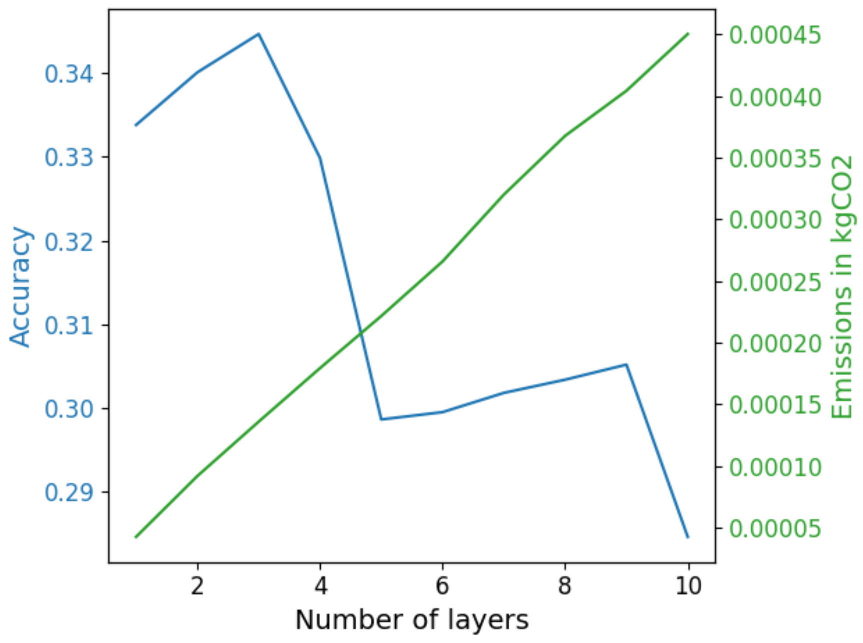
5.2.4 Architectural Model Design

The exploration of architectural model design involves modifying two crucial aspects: the number of LSTM layers and the number of nodes within a single LSTM-layer network, impacting both the network traffic and PV panel datasets. Assessing training emissions and test accuracy provides insights into the model's behavior under these variations.

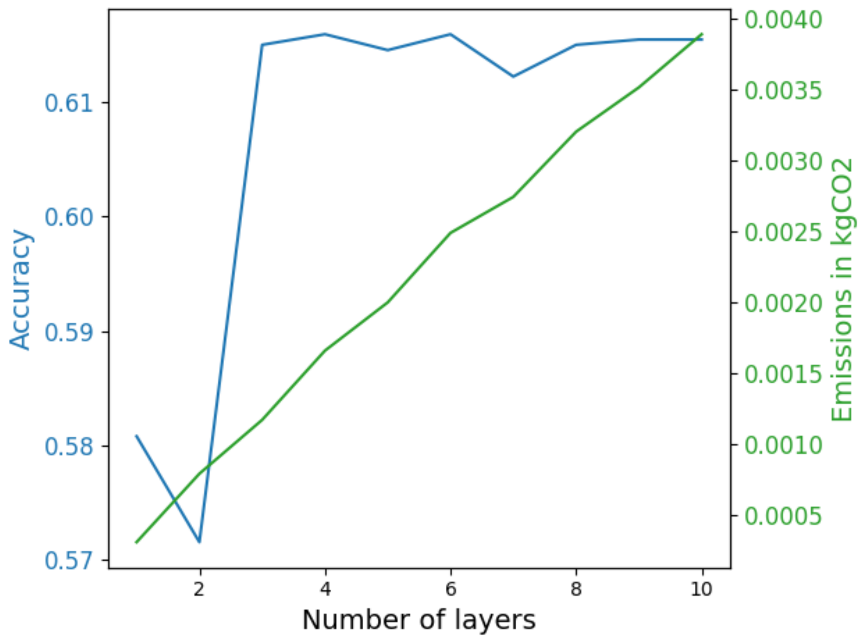
Initially, by varying the number of layers, visual representations are generated for both datasets to allow for direct comparisons, as shown in Fig. 5.7. These figures present accuracy and emissions on the y-axis against the number of layers on the x-axis. Interestingly, a distinctive pattern emerges: while emissions exhibit a consistent linear growth with the number of layers for both datasets, the behavior of accuracy diverges notably. Focusing on the network dataset, outcomes for the Train Station zone are exclusively detailed. As can be seen in Fig. 5.7a, the accuracy of this zone's base stations exhibits a non-linear trend that clearly declines as the number of layers rises over five. On the other hand, accuracy in the PV panel dataset shows an increasing trend up to three layers, at which point it plateaus, as shown in Fig. 5.7b.

This behavior in accuracy might be attributed to the complexity and granularity of the datasets. In the network dataset, the increment in layers might introduce architectural over-complexity, leading to an intricate model that struggles to generalize well to unseen data, resulting in decreased accuracy. A surplus of layers seems to affect the model's ability to generalize, which means that it can potentially lead to either overfitting or an intricate model unable to adapt to diverse data patterns. On the other hand, in the case of the PV panel dataset, the model may initially benefit from additional layers by capturing more intricate patterns, but then a plateau in the accuracy trend is reached. This suggests that further increasing layers may not significantly contribute to improving accuracy due to model saturation. Therefore, the initial rise in accuracy followed by a plateau indicates that the model has reached its capacity to learn from the data.

In conclusion, when aiming to strike a balance between accuracy and emissions, opting for a restrained number of layers appears advantageous in both datasets. In fact, choosing a reduced number of layers is beneficial as it potentially reduces computational demands, consequently curbing emissions. For the network dataset, the recommendation is to opt for the smallest feasible number of layers.

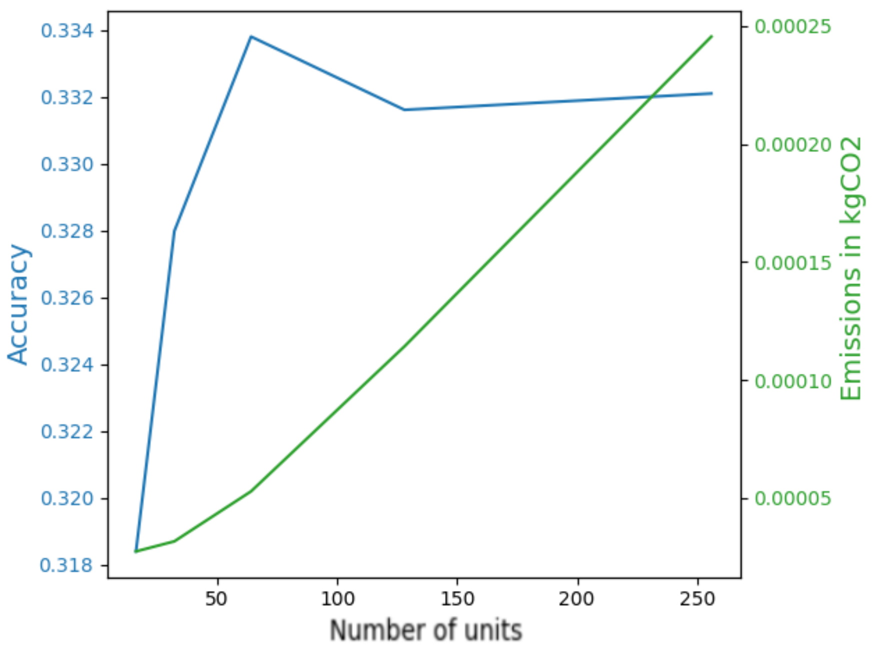


(a) Accuracy and Emissions Variation with Number of Layers for Train Station Zone in Network Traffic Dataset

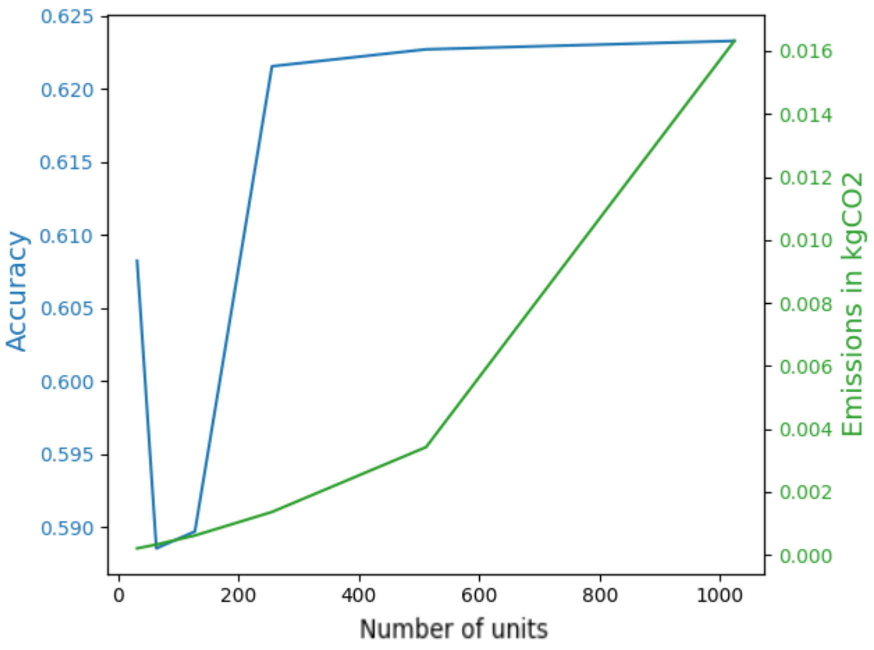


(b) Accuracy and Emissions Variation with Number of Layers for PV Panel Dataset.

Figure 5.7: Comparison of Accuracy and Emissions Across Different Datasets Varying with Number of LSTM Layers.



(a) Accuracy and Emissions Variation with Number of Nodes for Train Station Zone in Network Traffic Dataset



(b) Accuracy and Emissions Variation with Number of Nodes for PV Panel Dataset.

Figure 5.8: Comparison of Accuracy and Emissions Across Different Datasets Varying with Number of Nodes, when Considering only One LSTM Layer.

Conversely, in the case of the PV panel dataset, selecting the initial value that corresponds to the onset of the plateau proves advantageous, steering towards a configuration where additional layers do not significantly contribute to accuracy but may contribute to emissions.

Continuing with the analysis of architectural model design, another explored aspect is varying the number of nodes in a single-layer LSTM model for both datasets. Visualizing the outcomes through Fig. 5.8 reveals interesting trends. Across both datasets, increasing the number of nodes within the LSTM model initially boosts accuracy, showing a positive correlation. However, this improvement in accuracy plateaus after a certain amount of LSTM units, while the environmental cost – measured in terms of emissions – continues to rise steadily. This illustrates a trade-off in which additional accuracy improvements come with a disproportionately larger environmental cost. The marginal improvements in accuracy beyond a certain threshold do not justify the exponential increase in emissions. Hence, pursuing maximum accuracy without an equivalent rise in emissions is not justified. If the increase in emissions does not correspond to substantial accuracy improvements, it is unreasonable to sacrifice environmental impact for marginal gains in accuracy.

5.2.5 Tailored Problem Formulations

Targeted exploration of the problem formulations tailored to reduce emissions of the models across the two datasets is then taken into consideration. Since each dataset has its own characteristics, distinct strategies are employed. In the first case (regarding the network dataset), a comprehensive approach involves the aggregation of data originating from various sources, aiming to reduce the number of built models. Conversely, in the second case, related to PV panel dataset, two different analyses are conducted, proposing techniques concentrated on diverse characteristics of the problem. This includes both a meticulous investigation into feature selection, highlighting the critical variables that significantly impact model outcomes, and a nuanced exploration of sliding window variations, designed to adapt the model to evolving temporal patterns.

Network Data Problem: Data Aggregation
In order to mitigate potential environmental impact, a strategic aggregation technique is implemented for the network dataset. This approach aims to merge data from various base stations, thereby reducing the overall number of models. Indeed, instead of creating a model for each individual base station, a single model is crafted for each zone. This not only significantly decreases training time but also carries the hypothesis of reducing associated emissions. Building on insights from prior experiments, adjustments are made to both training hyperparameters and the architectural model design. These settings are applied uniformly to both

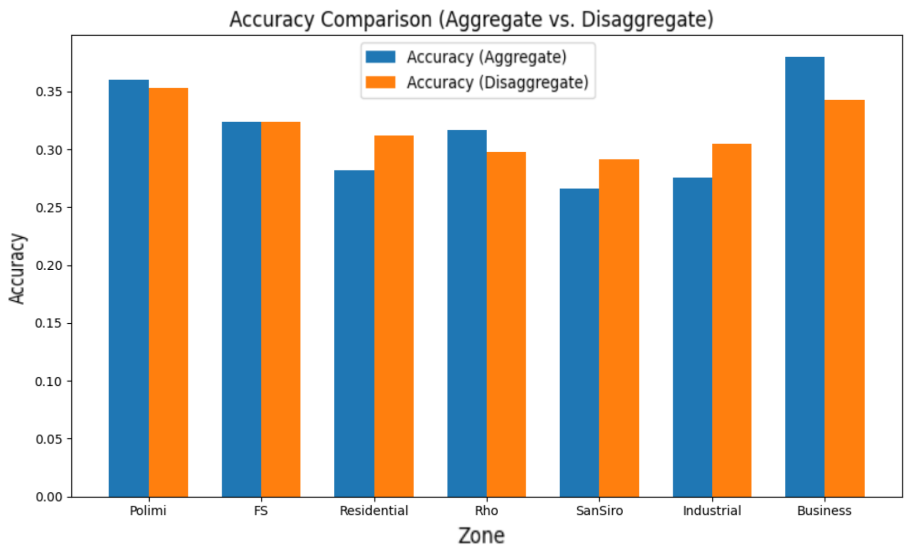
the aggregated and non-aggregated configurations to ensure comparable results. Notably, the number of epochs is curtailed to 100 from the previous 500, as the marginal gain in accuracy doesn't justify the heightened emissions. Additionally, the number of layers is streamlined from 4 to 1, given that, in the corresponding earlier analysis, increasing layers not only fails to improve accuracy but actually hampers it, while emissions exhibit a linear growth. After modifying the values for the number of epochs and layers, the metrics obtained for the modified setup are comprehensively presented in Table 5.5.

This table includes crucial performance indicators such as accuracy, emissions, and duration for both the training and testing phases. To facilitate a more nuanced understanding of the gains achieved with this modified configuration, any reduction in terms of emissions is reported in green within brackets, highlighting positive advancements. Conversely, in emissions instances where a loss is observed, it is denoted in red within brackets. Moreover, the color scheme is employed also to highlight gains and losses in the case of accuracy, but clearly with opposite meanings: the color green signifies an increase, while the color red denotes a decrease. This color-coded approach aims to accentuate the impact of improvements, particularly in terms of emissions, providing a visual cue for the reader to discern the magnitude and direction of the changes in the two considered metrics. The presented results unequivocally demonstrate the effectiveness of the modified configuration in substantially lowering emissions across all zones, thanks to the reduction in both epochs and layers. While there is a slight dip in accuracy, it is crucial to emphasize that this decline is relatively minor in magnitude. This nuanced trade-off underscores the success of the streamlined configuration in prioritizing environmental sustainability without compromising accuracy to a significant extent.

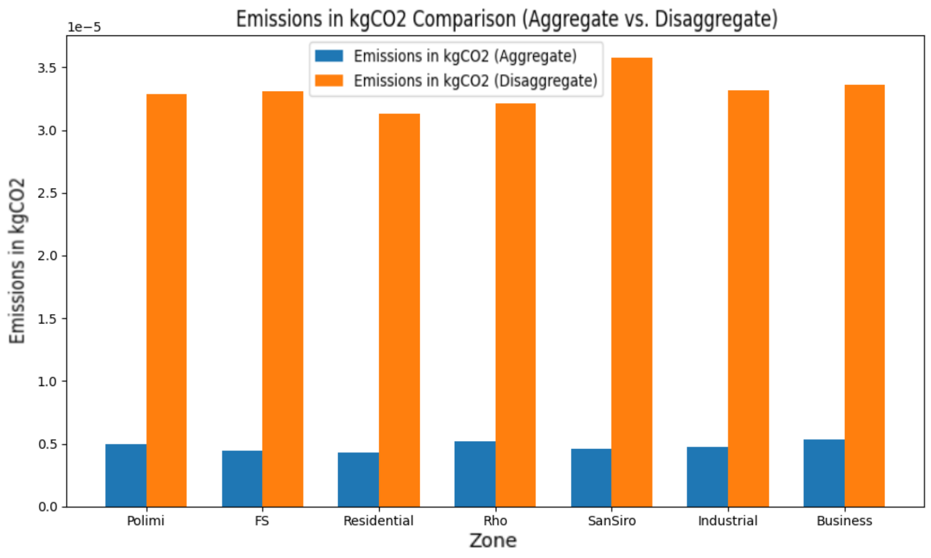
After showcasing the favorable outcomes resulting from adjustments in the number of epochs and layers, two histograms presented in Fig. 5.9 encapsulate the comparison between aggregated and non-aggregated values, showcasing the stark differences in accuracy and emissions. While accuracy, shown in Fig. 5.9a remains consistently comparable across all zones, emissions undergo a drastic reduction, exceeding a sixfold decrease in the aggregated approach, underscoring the efficacy of this tailored aggregation strategy, as depicted in Fig. 5.9b. The effectiveness is derived not only from maintaining accuracy but also from substantial mitigation of the environmental impact achieved through the significant reduction of emissions.

Zone	**Phase**	**Accuracy**	**Emissions (gCO_2eq)**	**Duration (seconds)**
Business	Training	0.246 (-0.058)	0.021 (-0.275)	18.8
	Testing	0.282 (-0.009)	0.005 (-0.014)	4.3
Industrial	Training	0.259 (-0.059)	0.021 (-0.253)	19.2
	Testing	0.278 (-0.022)	0.006 (-0.007)	4.8
San Siro	Training	0.248 (-0.061)	0.021 (-0.241)	19.0
	Testing	0.267 (-0.026)	0.006 (-0.006)	5.3
Rho Fiere	Training	0.244 (-0.068)	0.021 (-0.255)	19.9
	Testing	0.257 (-0.043)	0.005 (-0.008)	4.3
Residential	Training	0.256 (-0.061)	0.023 (-0.268)	20.4
	Testing	0.279 (-0.027)	0.005 (-0.014)	4.2
Train Station	Training	0.250 (-0.060)	0.022 (-0.272)	20.0
	Testing	0.273 (-0.027)	0.005 (-0.013)	4.5
Polimi	Training	0.258 (-0.054)	0.022 (-0.396)	20.2
	Testing	0.292 (-0.005)	0.005 (-0.020)	4.5

Table 5.5: Accuracy, emissions, and duration during both training and testing phases across all zones under the modified setup, concerning the network traffic data. Discrepancies from the original baseline are emphasized in green if they represent improvements and in red if they indicate deterioration.



(a) Accuracy comparison between aggregated and separated base station configuration.



(b) Emissions comparison between aggregated and separated base station configuration.

Figure 5.9: Comparison between aggregated and non-aggregated values for accuracy (Fig. 5.9a) and emissions (Fig. 5.9b). The histograms illustrate the stark differences in accuracy and the drastic reduction in emissions achieved through the tailored aggregation strategy.

PV Panel Problem: Feature Selection and Historical Data Consideration
Following the thorough exploration of training hyperparameters and architectural model design, modified settings – with respect to those of the baseline – have been adopted for the subsequent analyses. Specifically, the number of training epochs has been reduced from 250 to 50, driven by the observation that the difference in accuracies between these configurations is negligible, while the divergence in emissions is noteworthy.

The outcomes of these adjustments are presented in Table 5.6, employing a consistent color-coded visual representation, mirroring the approach used in Table 5.5 for the modified baseline metrics related to the network dataset. This impersonal presentation enhances the clarity of comparative impacts on both accuracy and emissions across varied configurations, aiding in the objective interpretation of the results. The outcomes underly the impact of the number of epochs specifically on the emissions during the training phase, while there is no impact on the inference phase since the structure of the model is not affected by this hyperparameter.

In the subsequent analysis, we introduce the results of the feature selection strategy after adopting the modified value of epochs for all the following configurations to ensure comparability. The feature selection strategy involves removing, at each experiment, one specific category of features (solar radiation, temperature conditions, and wind dynamics). The results are reported in Table 5.7, where the values of training emissions and test accuracy for all the different trials are compared.

Phase	**Accuracy**	**Emissions (gCO_2eq)**	**Duration (seconds)**
Training	0.476 (-0.005)	0.569 (-7.858)	477.4
Testing	0.612 (-0.003)	0.005 (+0.003)	3.9

Table 5.6: Accuracy, emissions, and duration during both training and testing phases, considering the modified settings with respect to the baseline of PV panel data. Discrepancies from the original baseline are emphasized in green if they represent improvements and in red if they indicate deterioration.

The results obtained from feature selection provide valuable insights into the trade-offs between model performance and environmental impact. Notably, when wind dynamics-related features are excluded, the model demonstrates even superior accuracy than that obtained with the baseline. This outcome aligns with the concept that reducing the number of features allows the model to focus on the

Configuration	**Accuracy**	**Emissions (gCO_2eq)**
Baseline	0.612	0.559
w/o Solar Radiation	0.604	0.521
w/o Temperature Conditions	0.610	0.542
w/o Wind Dynamics	0.614	0.542

Table 5.7: Comparison between values of training emissions and test accuracy of the feature selection strategy after adopting the modified number of epochs (50). Each experiment involves the removal of one specific category of features – solar radiation, temperature conditions, or wind dynamics.

most influential factors, resulting in a more refined and accurate prediction. On the other hand, the observed reduction in emissions across all three experiments underscores the environmental advantages of feature selection. By deliberately choosing a subset of relevant features, the model is trained with less data, leading to lower emissions during the training process. This reduction in emissions is a direct consequence of the streamlined input data, reinforcing the idea that a more focused model not only may enhance accuracy but also contributes to a more environmentally sustainable machine learning approach.

Following the exploration of strategies to enhance the performance and sustainability of the PV panel dataset, a pivotal consideration is given to the implementation of sliding windows. This approach investigates the correlation between the quantity of historical data provided to the model and the delicate trade-off between accuracy and emissions. As depicted in Fig. 5.10, the accuracy curve (in blue) exhibits a fluctuating pattern, while emissions (in green) linearly rise. Regarding accuracy, the results suggest the existence of specific production trends within distinct time frames. For instance, it seems reasonable to believe that a comprehensive understanding of the PV panels' production trend requires a 24-hour window. Indeed, in correspondence with a 24-hour window, the accuracy greatly increases. Notably, the figure reveals that even with a reduced timeframe, such as 8 hours, the model achieves high accuracy. This can be attributed to the periodic nature of solar light presence within this timeframe. The same does not happen when considering a 16-hour window, since probably it does not reflect any specific behavior.

The overarching insight gleaned from the analysis is that identifying and leveraging

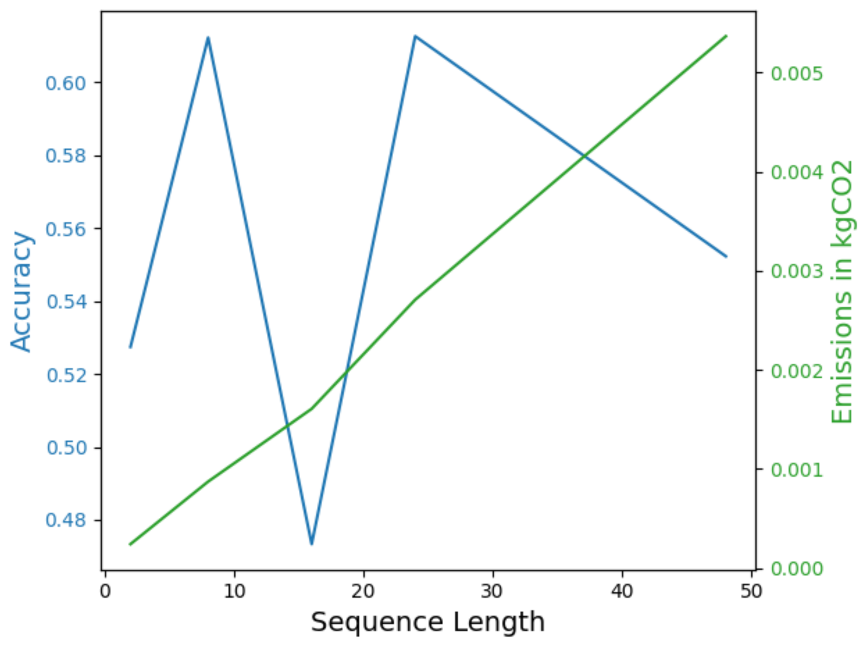


Figure 5.10: Accuracy and Emissions Variation with Sliding Window Configuration for PV Panel Dataset.

specific patterns associated with distinct time intervals can provide a means to optimize accuracy while minimizing emissions. By recognizing the periodicity of certain production trends and their correlation with time intervals, it becomes possible to pinpoint the minimum duration necessary to attain nearly maximum accuracy. This strategic alignment of temporal patterns and emission considerations opens avenues for designing more resource-efficient models without compromising predictive performance.

Chapter 6

Conclusions and Future Works

This research unfolds a nuanced understanding of the intricate interplay between DL model performance and carbon emissions, unveiling several novel insights that significantly contribute to the discourse on sustainable AI. A series of comprehensive experiments was meticulously conducted, shedding light on a delicate balance that exists when seeking high performance while simultaneously minimizing carbon emissions. The findings underscore that the pursuit of high performance should be carefully weighed against the associated increase in emissions, emphasizing the importance of a judicious trade-off.

This study emphasizes the pragmatic use of existing open-source resources for the evaluation of emissions, advocating for the adoption of standardized and reliable tools, according to the specific hardware settings that have to be monitored in each case study. Using standardized tools for assessing emissions is crucial to ensure the comparability of different research studies. This enables the research community to evaluate and compare emissions on a common scale by establishing a reliable benchmark. Overall, this consistency fosters a more robust understanding of the environmental impact of DL applications, facilitating a clearer comparison between different models, techniques, or datasets. By endorsing this approach, the research not only provides a practical guide for researchers about already available tools but also contributes to the establishment of a more comparable framework for assessing carbon emissions in DL applications.

Within the realm of training hyperparameters, this study illuminates their pivotal role in achieving energy efficiency. The correlation between these hyperparameters, the subsequent training duration, and, consequently, emissions, is accentuated. This insight offers a valuable perspective for practitioners, indicating that thoughtful consideration of training hyperparameters can influence not only the model's accuracy but also its environmental impact.

Moreover, the impact of architectural model design is also considered, revealing that adjustments can significantly affect a model's carbon emissions production. The complexity of a model emerges as a crucial factor, influencing learning time, power requirements, and ultimately, emissions. Additionally, architectural choices in model design extend their influence beyond emissions during the training phase, paralleling the significance attributed to training hyperparameters. Their impact is equally pronounced in the subsequent inference phase, thus imparting a long-term effect on emissions over an extended operational lifecycle. This comprehensive consideration of emissions, encompassing both training and inference, gains paramount importance in real-world scenarios where models are intended for continuous and prolonged deployment. The examination of emissions across these phases assumes a pivotal role in understanding and optimizing the environmental footprint of these models, ensuring sustained efficiency and eco-friendly operations throughout their practical deployment lifespan.

Importantly, this research highlights a holistic approach to problem formulation, asserting that early considerations of potential energy consumption can guide the development of more energy-efficient solutions. This aligns with the overarching theme of promoting sustainability throughout the entire ML lifecycle, from problem definition to model deployment.
By integrating carbon emissions as a metric in ML algorithms, we can foster an environment that prioritizes both performance and sustainability. With dedicated research and implementation of sustainable AI practices, we can mitigate the environmental impact of AI and contribute to a greener world.

In future investigations, expanding the scope of this research entails the inclusion of a broader spectrum of models and datasets. An interesting avenue involves replicating this exploration using hardware configurations equipped with GPUs or employing computer clusters. This expansion seeks to ascertain the generalizability of the findings across diverse hardware scenarios. Assessing whether similar trends in emissions prevail under varied computational setups will offer invaluable insights into the reproducibility and applicability of the identified emission patterns.
Another fascinating possibility for future exploration is to observe how a model affects the environment when it is used in the real world. This pragmatic approach aims to bridge the gap between theoretical analyses and practical implications, providing researchers and industries with tangible insights into the actual environmental ramifications of employing ML models. Understanding the real-time impact of model deployment offers an invaluable opportunity to gauge the ecological

footprint and ascertain the practical implications of employing these models in day-to-day operational scenarios.

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 1–3, 14, 16).

[2] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. *A Survey on Green Deep Learning*. 2021. arXiv: 2111.05193 [cs.LG] (cit. on pp. 1, 4).

[3] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. «Dive into Deep Learning». In: *CoRR* abs/2106.11342 (2021). arXiv: 2106.11342. URL: https://arxiv.org/abs/2106.11342 (cit. on pp. 1–4, 18).

[4] C. -C. Jay Kuo and Azad M. Madni. *Green Learning: Introduction, Examples and Outlook*. 2022. arXiv: 2210.00965 [cs.LG] (cit. on pp. 1, 16).

[5] Emma Strubell, Ananya Ganesh, and Andrew McCallum. «Energy and Policy Considerations for Deep Learning in NLP». In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3645–3650. DOI: 10.18653/v1/P19-1355. URL: https://aclanthology.org/P19-1355 (cit. on pp. 1, 10, 31).

[6] A. L. Samuel. «Some Studies in Machine Learning Using the Game of Checkers». In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210 (cit. on p. 1).

[7] Tom M. Mitchell. *The Need for Biases in Learning Generalizations*. Tech. rep. New Brunswick, NJ: Rutgers University, 1980. URL: https://www.cs.cmu.edu/~tom/pubs/NeedForBias_1980.pdf (cit. on p. 2).

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction, Second Edition*. 2nd ed. Springer Series in Statistics. Springer Science+Business Media, LLC, part of Springer Nature. New York, NY: Springer, 2009, pp. XXII, 745. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7. URL: https://doi.org/10.1007/978-0-387-84858-7 (cit. on p. 2).

[9] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 1st. O'Reilly Media, Inc., 2017. ISBN: 1491962291 (cit. on pp. 2, 3).

[10] Qiong Liu and Ying Wu. «Supervised Learning». In: (Jan. 2012). DOI: `10.1007/978-1-4419-1428-6_451` (cit. on p. 2).

[11] Simon J.D. Prince. *Understanding Deep Learning.* MIT Press, 2023. URL: `http://udlbook.com` (cit. on pp. 3, 15, 17).

[12] H.B. Barlow. «Unsupervised Learning». In: *Neural Computation* 1.3 (Sept. 1989), pp. 295–311. ISSN: 0899-7667. DOI: `10.1162/neco.1989.1.3.295`. eprint: `https://direct.mit.edu/neco/article-pdf/1/3/295/811863/neco.1989.1.3.295.pdf`. URL: `https://doi.org/10.1162/neco.1989.1.3.295` (cit. on p. 3).

[13] Yuxi Li. *Deep Reinforcement Learning: An Overview.* 2018. arXiv: `1701.07274 [cs.LG]` (cit. on p. 3).

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep learning». In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: `10.1038/nature14539`. URL: `https://doi.org/10.1038/nature14539` (cit. on pp. 4, 14, 16).

[15] Michael A. Nielsen. *Neural Networks and Deep Learning.* misc. 2018. URL: `http://neuralnetworksanddeeplearning.com/` (cit. on pp. 4, 19).

[16] Andrew J. Lohn and Micah Musser. «AI and Compute: How Much Longer Can Computing Power Drive Artificial Intelligence Progress?» In: *Center for Security and Emerging Technology* (Jan. 2022). DOI: `10.51593/2021CA009` (cit. on pp. 4, 30).

[17] Eva Garcia-Martin, Crefeda Rodrigues, Graham Riley, and Håkan Grahn. «Estimation of energy consumption in machine learning». In: *Journal of Parallel and Distributed Computing* 134 (Aug. 2019). DOI: `10.1016/j.jpdc.2019.07.007` (cit. on pp. 4, 5, 10, 29).

[18] Anne-Laure Ligozat, Julien Lefevre, Aurélie Bugeau, and Jacques Combaz. «Unraveling the Hidden Environmental Impacts of AI Solutions for Environment Life Cycle Assessment of AI Solutions». In: *Sustainability* 14.9 (2022). ISSN: 2071-1050. DOI: `10.3390/su14095172`. URL: `https://www.mdpi.com/2071-1050/14/9/5172` (cit. on p. 4).

[19] Stefanos Georgiou, Maria Kechagia, Tushar Sharma, Federica Sarro, and Ying Zou. «Green AI: Do Deep Learning Frameworks Have Different Costs?» In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2022, pp. 1082–1094. ISBN: 9781450392211. DOI: `10.1145/3510003.3510221`. URL: `https://doi.org/10.1145/3510003.3510221` (cit. on pp. 4, 10).

[20] Alexander E.I Brownlee, Jason Adair, Saemundur O. Haraldsson, and John Jabbo. «Exploring the Accuracy – Energy Trade-off in Machine Learning». In: *2021 IEEE/ACM International Workshop on Genetic Improvement (GI)*. 2021, pp. 11–18. DOI: `10.1109/GI52543.2021.00011` (cit. on pp. 4, 10, 29, 30).

[21] Charles AR Hoare. «Quicksort». In: *The Computer Journal* 5.1 (1962), pp. 10–16 (cit. on p. 4).

[22] Danny Hernandez and Tom B. Brown. *Measuring the Algorithmic Efficiency of Neural Networks*. 2020. arXiv: `2005.04305 [cs.LG]` (cit. on p. 5).

[23] Crefeda Rodrigues, Graham Riley, and Mikel Luján. «SyNERGY: An energy measurement and prediction framework for Convolutional Neural Networks on Jetson TX1». In: Oct. 2018 (cit. on p. 5).

[24] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. *Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model*. 2022. arXiv: `2211.02001 [cs.LG]` (cit. on p. 5).

[25] Emma Strubell, Ananya Ganesh, and Andrew McCallum. *Energy and Policy Considerations for Deep Learning in NLP*. 2019. arXiv: `1906.02243 [cs.CL]` (cit. on pp. 5, 32).

[26] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. *Carbon Emissions and Large Neural Network Training*. 2021. arXiv: `2104.10350 [cs.LG]` (cit. on pp. 5, 6, 46).

[27] T. Bruckner et al. «Chapter 7 - Energy systems». In: *Climate Change 2014: Mitigation of Climate Change. IPCC Working Group III Contribution to AR5*. Cambridge University Press, Nov. 2014. URL: `http://www.ipcc.ch/pdf/assessment-report/ar5/wg3/ipcc%5C%5fwg3%5C%5far5%5C%5fchapter7.pdf` (cit. on pp. 5, 13, 30).

[28] I.C. Change et al. *Mitigation of Climate Change. Contribution of Working Group III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. 2014, pp. 1454–147 (cit. on p. 5).

[29] Donald E. Knuth. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms.* USA: Addison Wesley Longman Publishing Co., Inc., 1997. ISBN: 0201896834 (cit. on p. 5).

[30] Aimee Wynsberghe. «Sustainable AI: AI for sustainability and the sustainability of AI». In: *AI and Ethics* 1 (Feb. 2021). DOI: 10.1007/s43681-021-00043-6 (cit. on p. 6).

[31] Raphael Fischer, Matthias Jakobs, Sascha Mücke, and Katharina Morik. «A Unified Framework for Assessing Energy Efficiency of Machine Learning». In: Jan. 2023, pp. 39–54. ISBN: 978-3-031-23617-4. DOI: 10.1007/978-3-031-23618-1_3 (cit. on p. 6).

[32] Hannah Ritchie and Pablo Rosado. «Energy Mix». In: *Our World in Data* (2020). https://ourworldindata.org/energy-mix (cit. on pp. 6, 7, 22).

[33] Raphael Fischer, Matthias Jakobs, and Katharina Morik. *Energy Efficiency Considerations for Popular AI Benchmarks.* 2023. arXiv: 2304.08359 [cs.LG] (cit. on pp. 7, 30).

[34] Hannah Ritchie, Max Roser, and Pablo Rosado. «CO_2 and Greenhouse Gas Emissions». In: *Our World in Data* (2020). https://ourworldindata.org/co2-and-greenhouse-gas-emissions (cit. on pp. 11, 12).

[35] Dieter Lüthi et al. «High-resolution carbon dioxide concentration record 650,000–800,000years before present». In: *Nature* 453.7193 (May 2008), pp. 379–382. ISSN: 1476-4687. DOI: 10.1038/nature06949. URL: https://doi.org/10.1038/nature06949 (cit. on p. 11).

[36] M.R. Allen et al. «Framing and Context». In: *Global Warming of 1.5°C. An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty.* Ed. by V. Masson-Delmotte et al. Cambridge, UK and New York, NY, USA: Cambridge University Press, 2018, pp. 49–92. DOI: 10.1017/9781009157940.003 (cit. on p. 11).

[37] UNFCCC. *UNFCCC: Report on the Structured Expert Dialogue (SED) on the 2013–2015 review.* Report FCCC/SB/2015/INF.1. 2015 (cit. on p. 11).

[38] Intergovernmental Panel on Climate Change (IPCC). «Summary for Policymakers». In: *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change.* Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press, 2013. Chap. SPM, pp. 1–30. DOI: 10.1017/CBO9781107415324.004 (cit. on pp. 11, 46).

[39] NOAA National Centers for Environmental Information (NCEI). *U.S. Billion-Dollar Weather and Climate Disasters*. https://www.ncei.noaa.gov/access/billions/. 2023. DOI: 10.25921/stkw-7w73 (cit. on p. 12).

[40] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4th. Pearson, 2021 (cit. on p. 14).

[41] M. I. Jordan and T. M. Mitchell. «Machine learning: Trends, perspectives, and prospects». In: *Science* 349.6245 (2015), pp. 255–260. DOI: 10.1126/science.aaa8415. eprint: https://www.science.org/doi/pdf/10.1126/science.aaa8415. URL: https://www.science.org/doi/abs/10.1126/science.aaa8415 (cit. on p. 14).

[42] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag New York, Inc., 2006 (cit. on p. 14).

[43] Avrim Blum, John Hopcroft, and Ravi Kannan. *Foundations of Data Science*. Cambridge: Cambridge University Press, 2020. ISBN: 9781108485067. DOI: DOI:. URL: https://www.cambridge.org/core/books/foundations-of-data-science/6A43CE830DE83BED6CC5171E62B0AA9E (cit. on p. 14).

[44] Christian Janiesch, Patrick Zschech, and Kai Heinrich. «Machine learning and deep learning». In: *Electronic Markets* 31.3 (Sept. 2021), pp. 685–695. DOI: 10.1007/s12525-021-00475-2. URL: https://doi.org/10.1007/s12525-021-00475-2 (cit. on p. 14).

[45] Iqbal H. Sarker. «Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions». In: *SN Computer Science* 2.6 (Aug. 2021), p. 420. ISSN: 2661-8907. DOI: 10.1007/s42979-021-00815-1. URL: https://doi.org/10.1007/s42979-021-00815-1 (cit. on p. 16).

[46] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 16).

[47] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 19).

[48] Michel Dubois, Murali Annavaram, and Per Stenström. *Parallel Computer Organization and Design*. Cambridge University Press, 2012 (cit. on p. 21).

[49] Neil Weste, David Harris, and A Banerjee. «CMOS VLSI Design: A Circuits and Systems Perspective». In: *11* (2005), p. 739 (cit. on p. 21).

[50] Mark Horowitz. «1.1 Computing's Energy Problem (and What We Can Do About It)». In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE. 2014, pp. 10–14 (cit. on p. 22).

[51] Monoj Kumar Mondal, Hemant Kumar Balsora, and Prachi Varshney. «Progress and trends in CO2 capture/separation technologies: A review». In: *Energy* 46.1 (2012). Energy and Exergy Modelling of Advance Energy Systems, pp. 431–441. ISSN: 0360-5442. DOI: `https://doi.org/10.1016/j.energy.2012.08.006`. URL: `https://www.sciencedirect.com/science/article/pii/S0360544212006184` (cit. on p. 22).

[52] Raghavendra Selvan, Nikhil Bhagwat, Lasse F. Wolff Anthony, Benjamin Kanding, and Erik B. Dam. «Carbon Footprint of Selecting and Training Deep Learning Models for Medical Image Analysis». In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 506–516. DOI: `10.1007/978-3-031-16443-9_49`. URL: `https://doi.org/10.1007%2F978-3-031-16443-9_49` (cit. on pp. 24, 31).

[53] Anne-Laure Ligozat and Sasha Luccioni. *A Practical Guide to Quantifying Carbon Emissions for Machine Learning Researchers and Practitioners*. Research Report ffhal-03376391f. MILA; LISN, 2021. URL: `https://hal.science/hal-03376391/document` (cit. on p. 24).

[54] Loïc Lannelongue, Jason Grealey, and Michael Inouye. *Green Algorithms: Quantifying the carbon footprint of computation*. 2020. arXiv: `2007.07610 [cs.CY]` (cit. on pp. 24, 25).

[55] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. *Quantifying the Carbon Emissions of Machine Learning*. 2019. arXiv: `1910.09700 [cs.CY]` (cit. on pp. 25, 26).

[56] Kadan Lottick, Silvia Susai, Sorelle A. Friedler, and Jonathan P. Wilson. *Energy Usage Reports: Environmental awareness as part of algorithmic accountability*. 2019. arXiv: `1911.08354 [cs.LG]` (cit. on pp. 26, 46).

[57] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. *Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models*. 2020. arXiv: `2007.03051 [cs.CY]` (cit. on p. 26).

[58] SA Budennyy et al. «Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai». In: *Doklady Mathematics*. Springer. 2023, pp. 1–11 (cit. on p. 26).

[59] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. *Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning*. 2022. arXiv: `2002.05651 [cs.CY]` (cit. on pp. 26, 27).

[60] M. J. Tristan Trebaol, Mary-Anne Hartley, and H. S. Ghadikolaei. *A tool to quantify and report the carbon footprint of machine learning computations and communication in academia and healthcare.* Infoscience EPFL: record 278189. 2020 (cit. on p. 27).

[61] Gary Cook, Jude Lee, Tamina Tsai, Ada Kong, John Deans, Brian Johnson, Elizabeth Jardim, and Brian Johnson. *Clicking Clean: Who is winning the race to build a green internet?* Tech. rep. Greenpeace, 2017 (cit. on p. 28).

[62] Peter Corcoran and Anders Andrae. «Emerging Trends in Electricity Consumption for Consumer ICT». In: (July 2013) (cit. on p. 29).

[63] D. Amodei and D. Hernandez. *AI and Compute.* `https://blog.openai.com/aiand-compute`. 2018 (cit. on pp. 29, 30).

[64] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. «Green AI». In: *Commun. ACM* 63.12 (Nov. 2020), pp. 54–63. ISSN: 0001-0782. DOI: `10.1145/3381831`. URL: `https://doi.org/10.1145/3381831` (cit. on pp. 29, 30).

[65] Tom B. Brown et al. *Language Models are Few-Shot Learners.* 2020. arXiv: `2005.14165 [cs.CL]` (cit. on p. 30).

[66] Shaden Smith et al. *Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model.* 2022. arXiv: `2201.11990 [cs.CL]` (cit. on p. 30).

[67] Arnault Pachot and Céline Patissier. *Towards Sustainable Artificial Intelligence: An Overview of Environmental Protection Uses and Issues.* 2022. arXiv: `2212.11738 [cs.AI]` (cit. on p. 30).

[68] United Nations Framework Convention on Climate Change (UNFCCC). *Kyoto Protocol to the United Nations Framework Convention on Climate Change.* Treaty Document FCCC/CP/1997/L.7/Add.1. United Nations, 1998 (cit. on p. 30).

[69] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. «On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?» In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency.* FAccT '21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610–623. ISBN: 9781450383097. DOI: `10.1145/3442188.3445922`. URL: `https://doi.org/10.1145/3442188.3445922` (cit. on p. 31).

[70] Friederike Rohde, Maike Gossen, Josephin Wagner, and Tilman Santarius. «Sustainability challenges of Artificial Intelligence and Policy Implications». In: *Ökologisches Wirtschaften - Fachzeitschrift* 36.O1 (Feb. 2021), pp. 36–40. DOI: 10.14512/OEWO360136. URL: https://oekologisches-wirtschaften.de/index.php/oew/article/view/1792 (cit. on p. 32).

[71] S. Han, H. Mao, and W.J. Dally. «Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding». In: *arXiv preprint arXiv:1510.00149* (2015) (cit. on p. 32).

[72] S. Han, J. Pool, J. Tran, and W. Dally. «Learning both weights and connections for efficient neural network». In: *Advances in Neural Information Processing Systems*. 2015, pp. 1135–1143 (cit. on p. 32).

[73] Yu-Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. «Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks». In: *IEEE Journal of Solid-State Circuits* 52.1 (2017), pp. 127–138. DOI: 10.1109/JSSC.2016.2616357 (cit. on p. 33).

[74] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, and Diana Marculescu. *NeuralPower: Predict and Deploy Energy-Efficient Convolutional Neural Networks*. 2017. arXiv: 1710.05420 [cs.LG] (cit. on p. 33).

[75] B.D. Rouhani, A. Mirhoseini, and F. Koushanfar. «Delight: Adding energy dimension to deep neural networks». In: *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM. 2016, pp. 112–117 (cit. on p. 33).

[76] Greta Vallero, Daniela Renga, Michela Meo, and Marco Ajmone Marsan. «Greener RAN Operation Through Machine Learning». In: *IEEE Transactions on Network and Service Management* 16.3 (2019), pp. 896–908. DOI: 10.1109/TNSM.2019.2923881 (cit. on pp. 35, 37, 56).

[77] ITU-R. *Framework for the radio interface(s) and radio sub-system functionality for international mobile tele-communications-2000 (IMT-2000) (Question ITU-R 39/8)*. ITU-R Recommendation M.1035. 1994 (cit. on p. 37).

[78] Nicholas A DiOrio et al. «Solar System Modeling at NREL». In: (Oct. 2018). URL: https://www.osti.gov/biblio/1477226 (cit. on p. 39).

[79] Tshewang Lhendup and Samten Lhundup. «Comparison of methodologies for generating a typical meteorological year (TMY)». In: *Energy for Sustainable Development* 11.3 (2007), pp. 5–10. ISSN: 0973-0826. DOI: https://doi.org/10.1016/S0973-0826(08)60571-2. URL: https://www.sciencedirect.com/science/article/pii/S0973082608605712 (cit. on p. 39).

[80] Intergovernmental Panel on Climate Change (IPCC). «Anthropogenic and Natural Radiative Forcing». In: *Climate Change 2013 – The Physical Science Basis: Working Group I Contribution to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2014, pp. 659–740. DOI: `10.1017/CBO9781107415324.018` (cit. on p. 46).

[81] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]` (cit. on p. 48).