Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Mohamed Khider – BISKRA
Faculty of Exact Sciences
Computer Science Department

Order No: PFE_/GLSD/M2/2024

Thesis

Presented to obtain the diploma of academic Master in

Computer Science

Option : **Software Engineering and Distributed Systems**

Deep learning approach for temperature prediction

By:
SAMAH GHERBIA

Defended the June th 2025, in front of the jury composed of:

Bennoui Hammadi	Prof	President
Hmidi Zohra	MCB	Supervisor
Djaber Khaled	MAA	Examiner

University Year 2024/2025

Acknowledgements

First and foremost, I am profoundly grateful to **Allah Almighty** for illuminating my path throughout this journey. To him belongs all praise, in beginnings and in ends.

I extend my deepest gratitude to my supervisor **Dr.Hmidi Zohra** for her invaluable guidance, continuous support, and constructive feedback. Her mentorship has been a cornerstone of my academic growth.

My sincere thanks go to **Samar Guessom** for her significant academic assistance; for clarifying complex concepts. and to **Kidous Wiaam** for offering encouragement during critical moments, which has contributed greatly to the success of this research. I am truly fortunate to have journeyed alongside such dedicated and insightful peers.

I would also like to sincerely thank **Dr. Kerdoudi Mohamed** for his valuable insights on the conception of this work. His expertise and perspective significantly enriched the foundation of this research.

Finally, I wish to express heartfelt appreciation to **my family**, whose support has been a constant source of strength. Their sacrifices have not gone unnoticed and will always be remembered with deep gratitude.

To each of you, I owe a debt of thanks that extends far beyond these words.

Dedication

This thesis is dedicated:

To ***Abir***, the one whose friendship feels like a sanctuary. More than a friend, more than family, you are the heartbeat that reminds me I am never truly alone. This accomplishment, this milestone, belongs to you as much as it does to me, for without your love and support, it would have been impossible. Thank you for being my forever confidante, the safe place I return to, and the closest soul to my own.

To the one whose absence became the loudest presence in my life: **my mother**. This work, like everything else I will ever accomplish, is for your soul. Thank you for raising me so lovingly, so well, until you no longer could.

Your high school daughter is now a graduate.

To **my father**, thank you for being a pillar when life tried to shake me, for being both mother and father when I needed more than one hand to hold. Thank you for allowing me the safety to grow, to dream, and now to make you proud.

To **Sara, Mira, and Meriem**, no words could ever match what your love has done for me. You stood by me, and never let me forget I was never alone. This success is as much yours as it is mine.

To **Djamel, Abd Errehmane, and Yusuf**, my beloved brothers and silent protectors.

To my little stars **Farah and Wiaam**

To **Wouroud**, you were my safety net when I faltered, my sanity when things felt too heavy, and my joy when I forgot to smile. This achievement carries your imprint as much as mine, and I am profoundly grateful to you for loving me in colors I never knew I needed.

To **Wiaam, Wissal, Amira, Imen, Khouloud, Ahlam, and Rawan**, you made the hard days lighter and the good days unforgettable. This journey would not have felt complete without you in it.

To **Amani**, my heart's comfort and one of the gentlest souls I've ever known. You've been there in silence and in laughter, in my scattered thoughts and my quiet fears. I'm so lucky this life gave me you and I am honored to dedicate this success to you.

To **Sirine, Camelia, Sarah, Wafa, Qamar, Romy, Maroua, Fatima, Hala, and all my internet friends**, thank you for walking beside me from afar, for believing in me when we hadn't even met.

To my beloved **students**, all six hundred of you made this year more enjoyable and this journey easier to walk.

And lastly, to **my 13 years old self**, you are living your dream, thank you for being patient. We made it, little one.

Abstract

Temperature prediction, a crucial component of weather forecasting, is of great public interest due to its role in minimizing losses, injuries, and fatalities. While traditional methods like numerical weather prediction (NWP) have been employed, advancements in artificial intelligence have significantly improved the accuracy of daily temperature forecasts.

Unlike traditional methods, deep learning models are well-suited for capturing temporal dependencies and nonlinear patterns in weather data. This study aimed to improve temperature predictions in Algiers by comparing Long Short-Term Memory (LSTM) and Gate Recurrent Unit (GRU) models, assessing their effectiveness in the context of climate variability and sustainable adaptation.

Using a dataset of daily temperatures in Algiers from 1995-2020, we evaluated the performance of LSTM and GRU models for multivariate time series prediction. Both models showed strong predictive capabilities, with the GRU model outperforming LSTM in terms of testing and training time.

Our findings suggest that recurrent neural networks (LSTM and GRU) are effective for forecasting weather phenomena and could complement traditional forecasting methods, offering more timely and accurate temperature predictions based on past and current weather data.

Key words: weather prediction, temperature, models, deep learning, GRU, LSTM, time series.

Résumé

La prédiction de la température, qui constitue un élément principal des prévisions météorologiques, suscite un grand intérêt auprès du public en raison de son importance pour réduire les pertes, les blessures et les décès. Bien que les méthodes traditionnelles comme la prévision numérique du temps (NWP) soient utilisées, les avancées en intelligence artificielle ont permis d'améliorer significativement la précision des prévisions quotidiennes de température.

Contrairement aux méthodes classiques, les modèles d'apprentissage profond excellent dans la capture des dépendances temporelles et des motifs non linéaires dans les données météorologiques. Cette étude visait à améliorer la prédiction des températures à Alger en comparant les modèles Long Short-Term Memory (LSTM) et Gate Recurrent Unit (GRU), afin d'évaluer leur efficacité dans le contexte de la variabilité climatique et pour contribuer à une adaptation durable.

À partir d'un ensemble de données comprenant les températures quotidiennes d'Alger de 1995 à 2020, les performances des modèles LSTM et GRU ont été évaluées pour la prédiction d'une série temporelle multivariée. Les deux modèles ont montré une forte capacité prédictive, le modèle GRU offrant une meilleure performance en temps d'entraînement et de test que le LSTM.

Les résultats expérimentaux confirment que les réseaux de neurones récurrents (LSTM et GRU) constituent une approche viable pour la prévision météorologique, pouvant offrir une alternative ou un complément aux méthodes traditionnelles, avec un potentiel d'amélioration de la rapidité et de la précision des prévisions de température basées sur les données passées et actuelles.

Mots-clés: prévision météorologique, température, modèles, apprentissage profond, GRU, LSTM, séries temporelles.

Contents

List of Figures

List of Tables

General Introduction

Context

Weather and climate have a tremendous impact on our daily lives; public health; economic activity; ecosystems; and governance on a global scale. Weather is defined as the conditions of the atmosphere at a certain place at a given moment in time (including temperature, humidity, wind, precipitation, etc) while climate is determined by a number of atmospheric conditions over a longer time period (typically decades). Climate and weather can therefore be inherently different but also intimately related: climate is the statistical baseline of weather observations. For example, available estimates for what seasonal temperatures to expect are derived from past climate data.

More recently, scientists have acknowledged an increase in both the frequency and intensity of extreme weather events. This increase in extreme weather events has created an awareness and desire for accurate and timely predictions [Sto16]; the most commonplace method of achieving this is through numerical weather prediction (NWP), which is a mathematical method for forecasting the future state of the atmosphere. This is done by simulating the effects of many highly complex physical equations that describe the dynamics and thermodynamics of the atmosphere. The equations are applied over a spatial grid (either global or regional) at high-resolution, and evaluated using high-performance computers. NWP models use current altmospheric observations—such as temperature, pressure, and wind speed—as input conditions, then predict, for each time step, the change in those variables, NWP models are complex mathematical applications, but they are the basis of modern day weather forecasting and are the primary methods of choice by meteorological agencies across the globe [PK19].

Problematic and Motivation

Timely and accurate weather predictions is a challenge faced around the world and Numerical Weather Prediction has been the principle technology; however,

it is hindered by uncertainty from the atmosphere, as well as costly computation associated with computing advances. Such advances are limited by additional computing costs, errors related to initial conditions, biases from numerical models, issues related to parameterization, and very large costs associated with increasing spatial or temporal resolution. As the field of meteorology continues to develop, there emerges an opportunity and need for effective and efficient solutions. Advances in deep learning, machine learning, and data analytic techniques are leading to effective and efficient solutions to weather forecasting, and possibly creating a paradigm shift to the meteorological industry.

Objective

The aim of this thesis is to improve the accuracy of weather predictions with the use of deep learning models. The climate change impacts weather predicting, making it remain a difficult endeavor. This research will assist in creating a predictor of daily temperatures, enhancing predictability and timing.

Manuscript organization

This manuscript is organized into three chapters as follows:

- **Chapter 1: Fundamental Concepts:** This chapter explores the fundamental concepts of artificial intelligence, machine learning, and deep learning, highlighting their differences and complementarities of deep learning approaches in weather prediction tasks.
- **Chapter 2: Weather prediction approach:** The second chapter presents the challenges of weather prediction and describes the design and architecture of the proposed deep learning approach. It also includes a discussion of related work in AI-based weather prediction methods.
- **Chapter 3: Implementation and Results:** In the last chapter we explained the tools and frameworks we used to implement our model. Additionally, the results we collected and evaluation of the performance of the model.
- **Conclusion:** The manuscript concludes with a general summary and future perspectives for improvements.

Chapter 1

Fundamental Concepts

1.1 Introduction

This chapter is an in-depth introduction to the theoretical basis of artificial intelligence, machine learning, deep learning, emphasizing their relationships and progression. It discusses how their techniques become critical for tackling many different real-world problems. This chapter introduces the methods of evaluation that will be used to assess their appropriateness to sequential, textual and image data overall and in weather prediction domain.

1.2 Artificial Intelligence

Artificial intelligence (AI) is a computer science field that focuses on developing computers that can perform human-like processes like learning, reasoning, and self-correction. It involves studying programming techniques to improve computer use. [BJ23]

1.2.1 Historical Background of AI

a) **Foundations (1940s-1950s)**

- Alan Turing proposed the Turing Test for machine intelligence[Col+21].
- The Dartmouth Conference (1956) coined the term "Artificial Intelligence."[Col+21].

b) **Early Developments (1960s-1970s)**

- ELIZA (1966) was an early chatbot mimicking a psychotherapist[Ber23].
- Paul Werbos introduced the concept of backpropagation for training neural networks[Wer90].

c) **AI Winter (1980s)**

- Expert systems in 1986 were a breakthrough in AI, but failed to fulfill tasks and led to AI companies going bankrupt[Ltd23].

d) **Machine Learning (ML) and Achievements(1990s)**

- International Business Machines(IBM) developed a chess-playing machine called Deep Blue in mid-1990[CHH02].

e) **AI Renaissance (2000s-Present)**

- Big Data and Deep Learning: Advancements in AI capabilities, driven by the rise of social media and the Internet of Things, have enabled organizations to collect vast amounts of data and develop deep learning algorithms.
- AlphaGo (2016): DeepMind's AlphaGo system defeated world-champion Go player Lee Sedol[Sil+16].

1.2.2 Applications of AI

The ongoing development of AI has expanded its applications in a number of domains, showcasing its adaptability and versatility. Here are several examples[Val23]:

1. **Finance:**

   - Fraud Detection: AI systems analyze transaction patterns to prevent fraudulent activities.
   - Customer Service: AI-powered chatbots handle customer inquiries.

2. **Transportation:**

   - Autonomous Vehicles: AI powers self-driving cars and trucks, enhancing road safety.
   - Traffic Management: AI analyzes traffic patterns and optimizes traffic flow.
   - Predictive Maintenance: AI predicts maintenance needs for vehicles and infrastructure.

3. **Education:**

   - Personalized Learning: AI provides customized learning experiences.
   - Administrative Tasks: AI automates grading and scheduling.
   - Learning Analytics: AI analyzes student data to identify at-risk students.

4. **Healthcare:**

   - Medical Imaging: AI algorithms can detect abnormalities more accurately than human radiologists.

- Predictive Analytics: AI models predict patient outcomes and treatment success.
- Personalized Medicine: AI tailors treatments based on individual patients' genetic and health history.

5. **Weather prediction[Muk+23]:**

- Forecasting Models: By using deep learning models, AI improves the precision of weather forecasts.
- Climate Analysis: AI assists in the analysis of past weather data to identify patterns in the climate.
- Disaster Prediction: AI helps early warning systems by forecasting extreme weather occurrences.

1.3 Machine learning

Machine learning, a subset of AI, allows machines to learn from data, make predictions, classify information, and identify patterns without explicit programming.[Ngu+24]

1.3.1 Types of Machine Learning

Machine Learning uses various algorithms to solve data problems. The chosen algorithm depends on the problem, variables, and model. Commonly used algorithms in ML are:

1.3.1.1 Supervised Learning

Supervised learning is a machine learning method that maps input to output using example pairs, requiring external assistance and dividing input datasets into train and test sets, using algorithms like regression and classification.[Mah+20]

1.3.1.2 Unsupervised Learning

Unsupervised learning is an algorithm that identifies natural relationships and groupings within data without focusing on outcomes[Bi+19]. It shares similarities with statistical approaches, such as clustering algorithms, which group objects based on similar data characteristics.

1.3.1.3 Reinforcement Learning

Reinforcement learning (RL) is a machine learning technique that uses interactions with the environment to learn how to behave optimally. It aims to map states to actions to maximize expected rewards, with rewards given at final and intermediate states. The agent seeks experience about states, actions, transitions, and rewards to learn optimal behavior. Unlike other machine learning techniques, RL follows a trial and error process, with delayed rewards and a balance between exploration and exploitation[ME22]. It is one of three basic paradigms, alongside supervised and unsupervised learning.

In Figure 1.1, The schematic represents three main types of Machine Learning: Unsupervised Learning (Left Side), Supervised Learning (Middle), and Reinforcement Learning (Right Side). Supervised Learning uses color-coded data points for classification, Unsupervised Learning uses clusters of data points for pattern identification, and Reinforcement Learning involves agents receiving feedback and adjusting their actions accordingly.

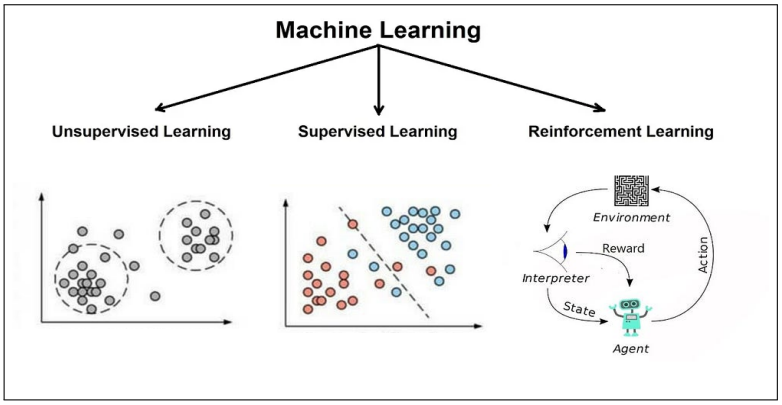

Figure 1.1: Three basic paradigms of machine learning

1.3.2 ML Techniques in Weather Prediction

Complex patterns in meteorological data are identified through the use of machine learning models such as Decision Trees, Support Vector Machines, which are employed to predict weather.

1.3.2.1 Decision Trees

Decision trees are used in data mining, a method that utilizes various data analysis tools to identify patterns, enabling precise predictions, to examine data

and predict outcomes. Each branch represents a choice between alternatives, with each leaf node representing a decision. They are built using algorithms like CART(Classification And Regression Trees) that generates binary trees with two branches, while multiway trees have more[Pet09].

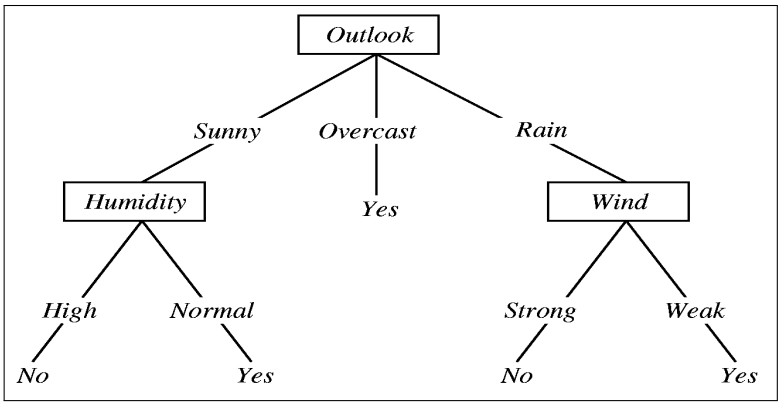

Figure 1.2: Decision tree for an activity based on the weather

In figure 1.2 the decision tree illustrated is a binary classification model using Outlook, Humidity, and Wind features, indicating favorable or unfavorable conditions for an activity outdoors based on the weather.

1.3.2.2 Support Vector Machines (SVMs)

Support Vector Machines are supervised learning methods used for classification and regression, developed by Boser, Guyon, and Vapnik in 1992. They use machine learning theory to maximize predictive accuracy and avoid overfitting to data[Jak06].

SVMs can be used to analyze time series data of daily maximum temperatures at a location, predicting the next day's temperature based on the previous n days. Classifying weather conditions using historical meteorological data by identifying the optimal boundary(hyperplane) in a multidimensional space to separate different data points. The model optimizes the hyperplane's position by maximizing the distance between support vectors and the hyperplane, enhancing its robustness and generalization ability[RS09]. SVMs are also used in other weather prediction tasks such as forecasting rain, forecasting temperature, classification of weather patterns, and more.

1.4 Deep learning

1.4.1 Definition

Deep learning is a development of machine learning that uses complex algorithms to mimic human cognitive processes, creating deep neural networks that make inferences from data analysis. This architecture minimizes human interaction and uses multiple linked layers with non-linearity, improving their ability to recognize intricate patterns. In contrast, classical machine learning uses human-designed features to form models and extract features step-by-step.[Man+24]

1.4.2 Artificial Neural Network (ANN)

An artificial neural network is a nonlinear model that mimics a human brain network and has applications in science, math, technology, health, and neurology, among other domains. ANNs are made up of a network of connected nodes that define an object through input, hidden, and output layers. There are several nodes in each layer, connected by borders, and each edge has a value that changes over the training process. Even while ANNs are quite adaptable, training them might take a lot of time. They are perfect for complex structures, particularly when the causes of drought are unknown. ANNs develop a useful input-output numerical method for future forecasts by learning from past experiences. An ANN model must be trained and tested using the necessary data and output values.[KHA21]

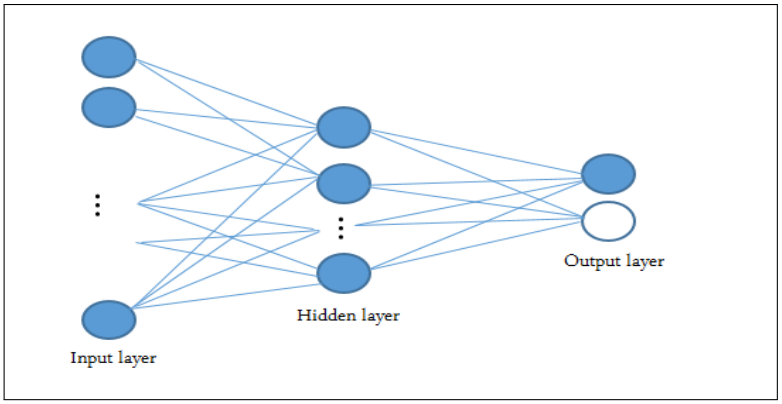

Figure 1.3: ANN architecture

1.5 DL techniques for Image-Based Data

Digital imaging technologies have generated vast amounts of visual data, leading to the development of advanced compute-based methods like CNNs, Pretrained Models, ..etc for image analysis. These methods enhance classification, segmentation, detection, and enhancement in visually complex recognition problems.

1.5.1 Convolutional Neural Network (CNN)

Convolutional Neural Network, often referred to as ConvNet or CNN, is a widely used algorithm in deep learning (DL) for automatically identifying relevant features without human supervision. It has applications in computer vision, speech processing, and face recognition. CNNs are inspired by neurons in human and animal brains, providing equivalent representations, sparse interactions, and parameter sharing. They use shared weights and local connections to make full use of 2D input-data structures like image signals, simplifying the training process and speeding up the network[Alz+21].

1.5.1.1 Architecture Overview:

CNN architectures consist of numerous convolution layers, pooling layers, and fully connected layers.

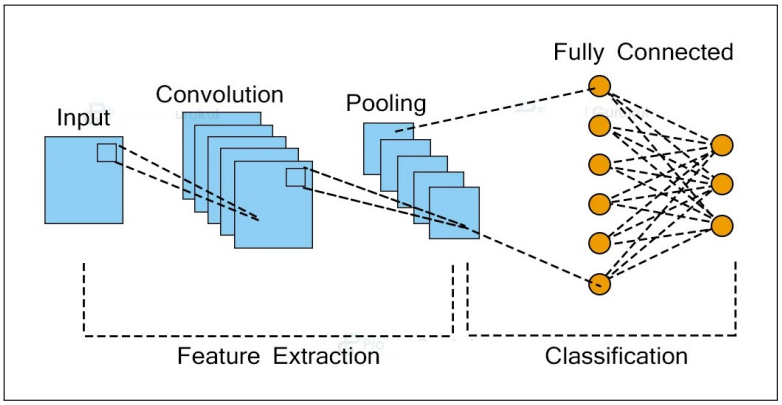

Figure 1.4: CNN architecture

Figure 1.4 illustrates the architecture of a CNN. The network consists of three main stages: input layer, feature extraction, and classification. The input layer is an image, which is processed through convolutional layers that apply filters to extract features like edges, textures, and patterns. The feature maps are then

pooled to reduce computation and prevent overfitting. The final output is a set of neurons representing a specific class.

1.5.2 Transfer learning and Pretrained Models

Transfer learning is a method that aims to transfer knowledge from different source tasks to a target task. It requires choosing a tractable way to transfer knowledge. In other hand, pre-trained models are neural networks pre-trained on large sets of data, which can be transferred to downstream tasks[You+21], the pre-training methods like feature transfer and parameter transfer are commonly used.Feature transfer pre-trains good feature representations, enhancing model performance on the target task. Parameter transfer fine-tunes model parameters that pre-encode knowledge on target task data. These methods form the foundation for Pre-Training Models (PTM), such as word embeddings and pre-trained CNNs[Han+21].

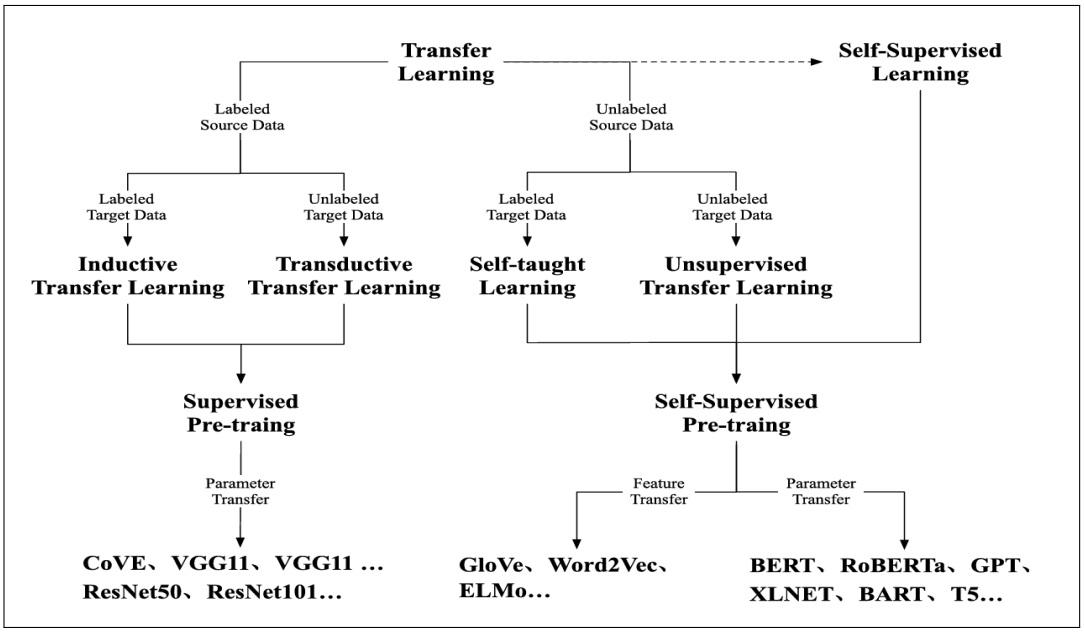

Figure 1.5: Taxonomy of pre-training methods: from transfer learning and self-supervised learning pre-trained neural models [Han+21].

The figure 1.5 illustrates four types of transfer learning based on the types of data that we have available:

1. **Inductive Transfer Learning:** Uses labeled source and labeled target data

- this is the situation in the context of supervised pre-training with models like VGG11, ResNet, etc.

2. **Transductive Transfer Learning:** Uses labeled source data and unlabeled target data - this still represents a situation in the context of supervised pre-training with labeled source data.

3. **Self-taught Learning:** Uses unlabeled source data and labeled target data - this is generally viewed as considering indices in the context of self-supervised pre-training (e.g. GloVe, Word2Vec, etc.).

4. **Unsupervised Transfer Learning:** Uses unlabeled source data and unlabeled target data - this involves using self-supervised learning methods like BERT, GPT, etc.

1.6 DL techniques for Textual and Sequential Data

Sequential and textual data are utilized in various applications. These data can be ordered and have temporal dependencies, necessitating models that can develop contextualized relationships over time or sequence. Deep learning solutions like Recurrent Neural Networks, LSTM networks, GRUs, and Transformer-based architectures have been successfully applied to identify useful relationships over time and sequence, enabling tasks like translation and predictions.

1.6.1 Recurrent Neural Network (RNN)

Recurrent neural networks are the opposite of typical neural networks, in which the output affects its own input. They remember prior knowledge, making predictions and judgments based on historical context. [Al 24] It is specifically designed to handle sequential data, such as time series or natural language.

1.6.1.2 Architecture Overview:

The figure 1.6 shows recurrence, where the neurons in the hidden layer have input both from the input layer and their previous activations. The recurrence in the example is temporal as it is the flow from the last time step to the current time step and allows the network to remember patterns over time.

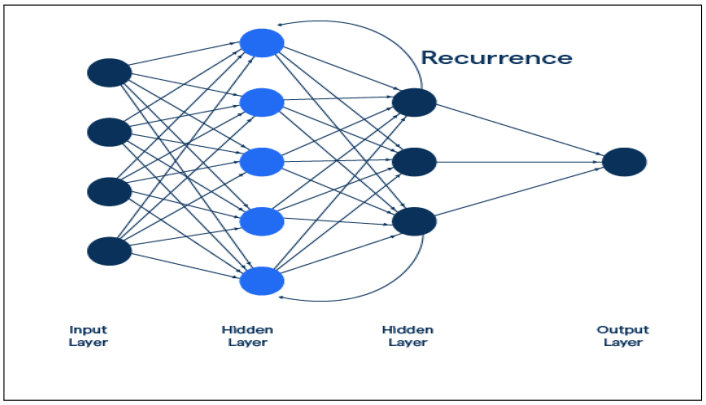

Figure 1.6: RNN architecture

1.6.1.3 Limitations:

Due to gradients multiplying repeatedly, RNNs cannot accurately model long-term dependencies at the long time scale. A formal description of this problem is provided in Bengio et al.[BSF94].

1.6.2 Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) architecture is an extension of traditional Recurrent Neural Networks (RNNs) designed to capture long-term dependencies in sequential data. It is an approach consisting of recurrently connected subnets called memory blocks. These blocks are differentiable versions of digital computer memory chips, containing self-connected memory cells and three multiplicative units: input, output, and forget gates. The LSTM architecture provides continuous analogues of write, read, and reset operations[Gra12].

1.6.2.1 Architecture and working mechanism:

There are three kinds of gates in architecture; the input gate, forget gate, and output gate. Each gate is accompanied by the cell state.

The forget gate is concerned with determining what information should be forgotten; the input gate is concerned with determining which values from the input would result in the cell state changing; and the output gate is concerned with determining what to output given the cell state. Due to their structure, LSTM units can learn when they should forget or remember information over longer periods of time, which is what makes LSTMs appropriate for situations where there are long-range temporal dependencies.

The figure 1.7 depicts a stacked LSTM network for time series prediction,

consisting of two layers: Layer 1 processes inputs of shape (T, 16) where, T is the number of time steps and 16 is the number of features at each time step, an outputs (T, L1), and Layer 2 produces outputs (T, L2), combined by an output layer for prediction.

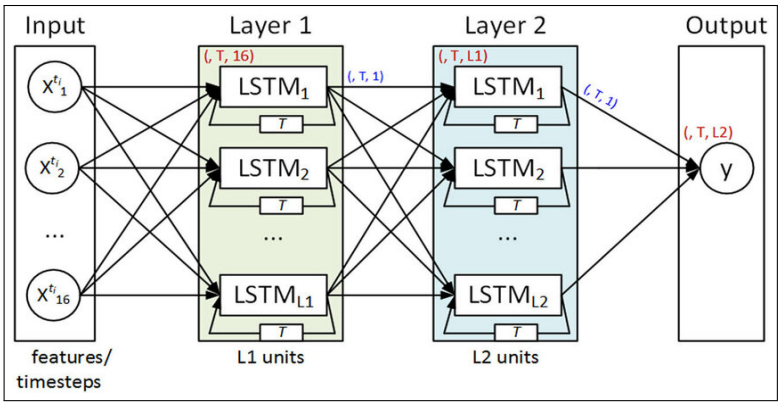

Figure 1.7: LSTM network for time series prediction

1.6.2.2 Applications:

Long Short-Term Memory networks (LSTMs) have been applied to a broad variety of tasks in automatic processing of sequences including **speech recognition**, **language modeling**, **machine translation**, and **weather prediction**[Has+19]. What makes LSTMs so powerful in sequence-to-sequence modeling is their ability to model long sequences[SVL14].

1.6.3 Gated Recurrent Units (GRU)

The Gated Recurrent Unit is a simplified version of the LSTM, combining forget and input gates into a single "update gate" and an additional "reset gate". GRUs simplify the LSTM structure while maintaining the ability to model long-term dependencies.[Ran16].

1.6.3.1 Architecture:

The figure 1.8 illustrates the architecture of a Gated Recurrent Unit (GRU) model. The GRU is implemented with two gates: the reset gate $r(t)$ and the update gate $z(t)$. The reset gate determines how much of the previous hidden state $h(t-1)$ to forget when producing the candidate hidden state $\tilde{h}(t)$, which is constructed from the input $x(t)$ and the gated version of the previous hidden

state. The update gate determines how much to update the unit activation through interpolation between the previous hidden state and candidate hidden state. The output hidden state $h(t)$ is a weighted sum of $h(t-1)$ and $\tilde{h}(t)$, depending on $z(t)$. Depending on $z(t)$, the model may retain all hidden state information in the form of $h(t-1)$, or it may retain the candidate $\tilde{h}(t)$, or some combination of them. This means that the GRU can retain or forget long ranges of input over long sequences of data when outputs may be based on that historical information. While the GRU does not have cycles like LSTM, it has shown the ability to learn temporal dependencies well while using fewer parameters.

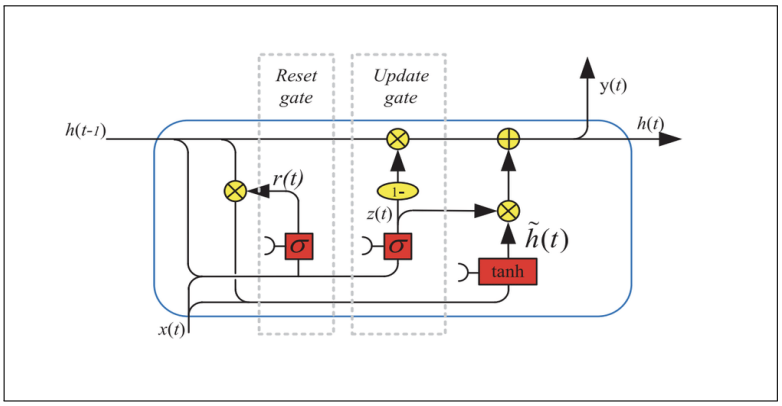

Figure 1.8: GRU model's architecture

1.6.3.2 Applications:

GRUs are widely utilized in time dependent applications like **real-time language modeling** [Lee+18], **weather prediction** [DYH22], and **sensor based activity recognition** [Pan+22].

1.6.4 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a subfield of Artificial Intelligence that enables computers to understand, interpret, and generate human language[CPS13]. It focuses on enabling computers to understand and generate human language, which is a system of rules or symbols. This technology caters to users who cannot learn machine-specific languages.

1.6.5.1 Natural language processing techniques

Natural language processing involves four phases.

1. **Morphological and Lexical Analysis:** The lexicon of a language refers to its vocabulary, including words and expressions, and morphology involves analyzing, identifying, and describing the structure of words[CC20]. The lexicon of a language refers to its vocabulary, including words and expressions, and morphology involves analyzing, identifying, and describing the structure of words.

2. **Syntactic Analysis:** The analysis of words in a sentence helps depict its grammatical structure, transforming them into related structures, such as "the boy go to the university," which would be rejected by an English syntactic analyzer.

3. **Semantic and discourse analysis:** Semantic Analysis abstracts dictionary meaning from context, assigning meaning to syntactic structures. It maps syntactic structures to task domain objects. Discourse Analysis senses depend on preceding sentences and subsequent sentences, like "it" in "he wanted it."

4. **Pragmatic Analysis:** Pragmatic analysis involves abstracting language use in situations requiring world knowledge, reinterpreting words to understand their meaning, such as "open the door?" as a request.

1.6.5 Transformers and Large Language Models (LLMs)

LLMs are models that are trained to predict words from large internet text datasets, containing billions of parameters optimized at great computational, energy, and financial expense. Word prediction is a crucial part of human language processing and has long been shown to provide a learning signal for linguistic structures and semantic categories.[PH22] Large language models have shown competencies that were once thought beyond the reach of neural networks.

1.6.6.1 Transformers

A transformer is a deep learning model that efficiently handles challenging natural language processing issues by processing sequence data using an attention strategy. It was first introduced in 2017 and, because of its parallel processing capabilities, it takes the place of the conventional recurrent neural network design in machine translation tasks[Liu+24].

1.6.6.2 Attention-based methods

Recent research has proposed Attention-based intrusion detection system techniques, such as combining CNN with an Attention mechanism to develop a system that achieves higher accuracy levels, requires a smaller dataset for training, and reduces training time. There are various methods namely:

a) **Self-attention:** Unlike recurrent networks or convolutional networks, which process data in order or focus on local patterns, self-attention is the significant innovation of the transformer architecture, it enables each word or token in the input sequence to weigh the relevance of every other word in the sequence [Fer+25]. This allows the model to better capture contextual relationships and long-range dependencies. Self-attention enhances anomaly detection by concentrating on the most crucial data parts, thereby enhancing the efficiency of anomaly identification.

b) **Multi-head attention:** Multi head Attention enhances self-attention, improving anomaly detection by providing multiple perspectives on data [Fer+25]. The model uses self-attention to consider token relationships, while multi-head attention allows it to focus on different positions and representation subspaces. This is achieved by using h different attention heads, each with its own set of learned projections, instead of computing attention once.

c) **Local attention:** Local attention in high-speed networks focuses on critical data streams for quick anomaly detection, enhancing computational efficiency and reducing computational cost by restricting focus to a specific set of input positions[Khe24].

d) **Cross attention:** Cross Attention is a technique used in encoder-decoder architectures, where encoder outputs are queries and key-value pairs come from the decoder. It is used to analyze data sources like logs and network flows, improving detection accuracy by targeting specific parts of the encoder's output [Nav+23].

1.6.6.3 Transformer architectures

There are two main parts to the transformer architecture: the encoder and the decoder.

1. ***Encoder:*** The Transformer model's encoder module uses a multi-head attention mechanism and feed-forward neural network to capture dependencies in input sequences, extracting features through stacking layers and passing the final result to the decoder module for decoding [Liu+24].

2. ***Decoder:*** The Transformer model's decoder module uses multiple layers, multi-head attention mechanisms, and a feed-forward neural network. It also has an encoder-decoder attention mechanism for computation [Liu+24]. Masks ensure current time step information is focused, preventing future leakage and maintaining causality.

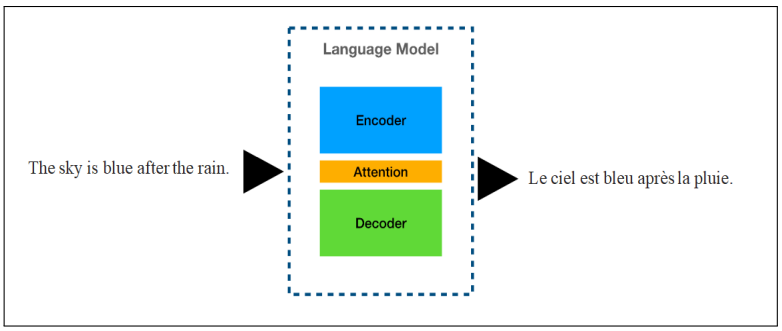

Figure 1.9: Bahdanau attention between the input to the encoder and the decoder output in an example of English to French translation[Lug24].

The transformer architecture is particularly effective for sequence-to-sequence tasks like machine translation, where input and output sequences are transformed. (See Figure 1.9)

a. Encoder-decoder Architecture

The Encoder-decoder architecture of LLMs is a modification of the Transformer Encoder-decoder architecture, consisting of two main components: the Encoder and the Decoder. The encoder encodes the input sequence using multiple transformer's multi-head self-attention layers, while the decoder generates the target sequence using cross-attention. This architecture is the foundation for LLMs like T5 and flan-T5.[Nav+23] (Figure 1.10)

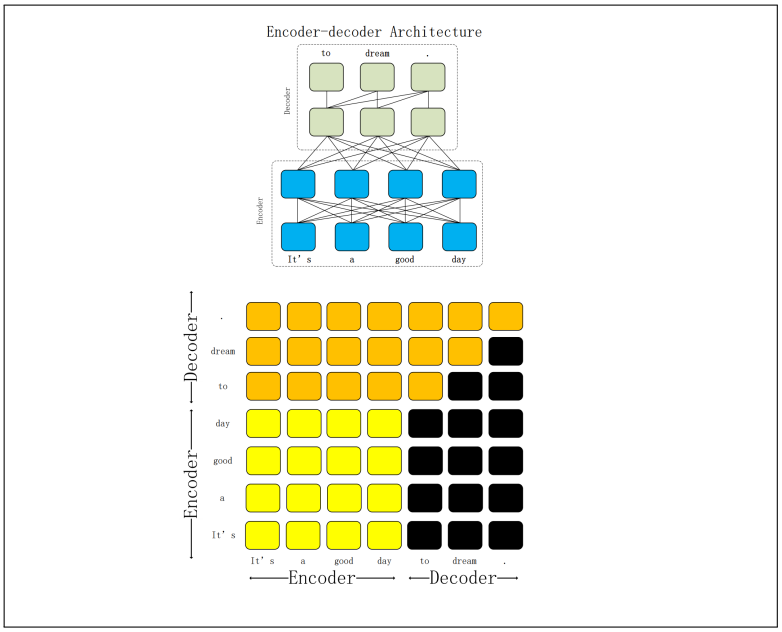

Figure 1.10: Encoder-decoder architecture is shown in the diagram, along with the range of tokens that each input token can attend to. [Liu+24].

b. Decoder-only Architecture

Decoder-only transformers, like the GPT series, are used for tasks like language modeling and text generation, focusing on sequence generation based on prior context. These models consist of decoder layers with masked multi-head self-attention, ensuring predictions depend only on preceding tokens [Liu+24]. They are ideal for tasks like autocomplete and text continuation. LLMs with a Decoder-only architecture use the decoder component of the traditional Transformer architecture, focusing solely on the decoding process. This architecture has been applied to various language generation tasks, demonstrating its effectiveness without an explicit encoding phase. It can be classified into Causal Decoder and Prefix Decoder architectures [Fer+25].

1. ***Causal Decoder:*** The causal decoder architecture uses a unidirectional attention mask to ensure input tokens attend to past and itself, processing both input and output tokens in the same manner [Zha+23].

2. ***Prefix Decoder:*** A non-causal decoder, also known as a bidirectional attention calculation, does not rely solely on past information [Nav+23].

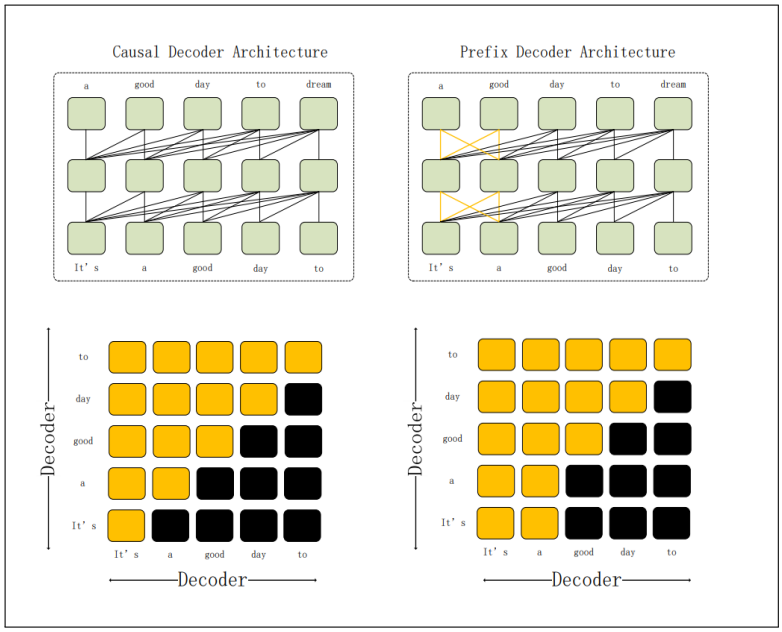

Figure 1.11: Visual representation of a decoder-only architectures. [Liu+24].

1.7 DL Applications in Weather Prediction

Deep learning algorithms can learn complicated, non-linear relationships from large datasets, which is why they are being used for weather prediction[Fan+21]. In the context of weather prediction the common used approaches are LSTM, GRU and Transformers. for that the table 1.1 below will address the differences between them based on many factors:

Table 1.1: Comparison of LSTM, GRU, and Transformer Architectures

Aspect	LSTM	GRU	Transformer
Architecture	Uses memory cells, input, forget, and output gates	Combines input and forget gates into update and reset gates	Based on self-attention and feed-forward layers
Sequence Processing	Sequential (step-by-step)	Sequential (simpler structure)	Fully parallelized across sequence
Training Speed	Slower due to gate complexity	Faster (fewer parameters)	Fast due to parallel computation
Memory Efficiency	Moderate (more parameters)	High (fewer parameters)	Demands more memory due to attention weights
Long-Term Dependencies	Good via memory cells	Good but less explicit	Excellent, captures global dependencies
Data Requirement	Performs well on medium datasets	Suitable for small to medium datasets	Requires large datasets for optimal performance
Suitability for Weather Forecasting	Suitable for time-dependent modeling	Effective and efficient for weather data	High accuracy with spatial-temporal data
Interpretability	Limited interpretability	Similar to LSTM	better via attention visualization
Example Use Cases	Time series, speech recognition	Weather prediction, sensor networks	NLP, weather modeling with attention

1.8 Machine learning and Deep learning

Table 1.2 shows the comparison between machine learning and deep learning in various aspects, focusing on weather prediction.

Table 1.2: Comparison of ML and DL in Weather Prediction

Feature	**Machine Learning**	**Deep Learning**
Model Complexity	Simpler models, often linear or tree-based	Complex models with multiple layers (CNN, LSTM, .. etc)
Data Requirements	Can work with smaller datasets but may require manual feature engineering	Requires large datasets; can automatically extract features from raw data
Accuracy and Performance	Generally less accurate than DL for complex weather patterns	Can provide high accuracy, especially for short-to-medium range predictions
Applications	Suitable for simpler forecasting tasks or when data is limited	Ideal for complex tasks like predicting atmospheric states, precipitation, and temperature
Advantages	Faster development and deployment, easier to interpret	Can handle large datasets and complex patterns,improving forecast accuracy

1.9 Conclusion

This chapter provided an overview of the foundational concepts, highlighting their potential to improve weather prediction. The findings offer a foundation for further research into AI's use in weather prediction, potentially leading to more accurate and beneficial predictions. In the second chapter, we will focus on the design and conception of our proposed prediction models, detailing the data preprocessing steps and architecture choices.

Chapter 2

Weather prediction approach

2.1 Introduction

This chapter describes the literature review and outlines the design and conceptualization of the proposed temperature prediction approach, which documents the data source, data preprocessing steps and explains the deep learning models used and the evaluation strategies.

2.2 Related work

The literature review includes several methods and approaches to weather prediction. In this section, we will present and discuss these related works we aligned with our project.

The importance of predicting daily temperature has led researchers to explore and build models using an array of different research methodologies. Traditionally, daily temperature prediction has relied on NWP, Numerical Weather Prediction[BTB15], and ARIMA, AutoRegressive Integrated Moving Average, which is a statistically based model that simplifies univariate time series analysis of non-seasonal aspects of time series data [Box+76]. On the other hand, we have SARIMA, Seasonal AutoRegressive Integrated Moving Average, which is based on ARIMA with the capability to identify both the seasonal and non-seasonal components present in the data and is therefore a more robust way to analyze seasonal dataHyndman and Athanasopoulos [HA18].

However, with the improved forecast resolution, they are more costly as well as time-consuming. For that, the rapid advancement of artificial intelligence technology has resulted in a commitment to implementing machine learning methods for weather forecasting, given that machine learning methods can model complex nonlinear correlations and enhance prediction accuracy significantly.

Forinstance, Hossain et al. [Hos+15] suggested that a deep neural network with Stacked Denoising Auto-Encoders (SDAE) outperforms a standard multilayer feedforward network in time series prediction tasks. The study also suggested that predicting air temperature from historical data can be improved by incorporating related weather variables.

Zaytar and El Amrani [ZE16] developed a weather forecasting model using a stacked LSTM network. The model's performance was evaluated using RNN and LSTM, proving their self-learning capabilities and superiority for time series prediction.

Jaharabi et al. [Jah+23] and Xu et al. [Xu+24] are the closest to our study, Jaharabi et al. has built a model which can predict the temperature of future years

in an accurate manner based on machine learning and deep learning algorithms in which they used LSTM for the purpose of turning existing data into a tool for future prediction, they also applied ARIMA and SARIMA. In the other hand, Xu et al. investigated the ability of six traditional machine learning models based upon performance criteria for predicting daily temperatures to improve forecasting accuracy, the stacking based multi-model fusion method utilized the ensemble of predictors, as well as utilized a recurrent neural network (RNN).

2.3 Methodology

To develop a reliable and effective deep learning-based model for predicting future temperature, we designed and built two models: a LSTM (Long Short-Term Memory) model and a GRU (Gated Recurrent Unit) model. Both models were trained and adjusted on pre-processed, normalized time-series data, and evaluated using common performance metrics.

We took advantage of LSTM and GRU networks specifically because of their ability to account for long-term dependencies and patterns in sequential data.

To build an accurate model we have planned our work in several steps shown on figure 2.1.

2.3.1 Data Collection

The model is trained on the same dataset as Jaharabi et al. [Jah+23] and Xu et al. [Xu+24]. The dataset was provided by University of Dayton and is available on the data science and machine learning platform Kaggle under the name "Daily temperature of major cities" and can be found at:
(www.kaggle.com/datasets/sudalairajkumar/daily-temperature-of-major-cities/data).
It consists of historical weather data records of the daily temperature in Fahrenheit of 167 major cities throughout the world, including Algiers, from January 1, 1995 to May 2020. It has 8 columns with around 4,423 cities, consisting of Region, Country, State, city, as well as three columns for Month, Day, and Year.

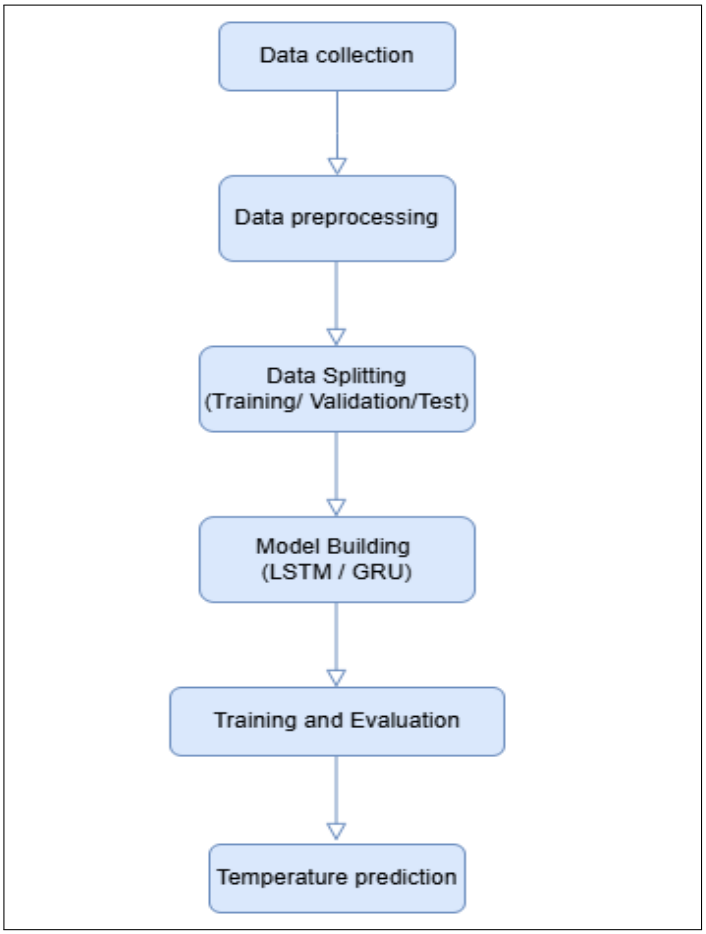

Figure 2.1: Flowchart of the temperature prediction approches.

Region	Country	State	City	Month	Day	Year	AvgTemperature
Africa	Algeria	–	Algiers	1	1	1995	64.2
Africa	Algeria	–	Algiers	1	2	1995	49.4
Africa	Algeria	–	Algiers	1	3	1995	48.8
Africa	Algeria	–	Algiers	1	4	1995	46.4
Africa	Algeria	–	Algiers	1	5	1995	47.9
Africa	Algeria	–	Algiers	1	6	1995	48.7
Africa	Algeria	–	Algiers	1	7	1995	48.9
Africa	Algeria	–	Algiers	1	8	1995	49.1
Africa	Algeria	–	Algiers	1	9	1995	49.0
Africa	Algeria	–	Algiers	1	10	1995	51.9

Table 2.1: Raw Sample of Daily Average Temperature Data

2.3.2 Data preprocessing

To enhance the model's generalizability, various data preprocessing operations are needed. These include inverting missing values, dropping unnecessary columns, and data splitting.

1. **Data cleaning:** The data contains missing rows with a value of -99 in average temperature, which are inverted using the previous row.

2. **Feature Engineering:** A new Date column was created from year, month, and day. This will help with indices and sorting according to time. The data was filtered for Algiers only and the Year, month, day, state, and region columns were then dropped from the dataset as they are redundant.

   In order to include time related patterns into the model, several date based features were extracted from the Date column: Day of Year, Month, and Day of Week, these features will capture seasonal, monthly, and weekly patterns in the data Cyclic Encoding because features like day, month, and week contain cyclic patterns (e.g., December is followed by January), sine and cosine transformations were utilized to represent these time elements, in order to maintain natural continuity and cyclic repetition

3. **Normalization:** The temperature values had to be normalised between [0, 1] with the MinMaxScaler from to promote rapid convergence and efficiency of model training.

4. **Data splitting:** the training set consists of 80% of the data, the validation set consists of 15% of the training data, and the testing set consists of the last 20% of the data, in chronological order. This time-based split is important, as it allows for the model to be trained in the past and to be tested in the future. This assure that the model is valid for future use.

Country	City	AvgTemperature	Date	day_of_year	month	day_of_week	sin_day	cos_day	sin_month	cos_month	sin_dow	cos_dow
Algeria	Algiers	64.2	1995-01-01	1	1	6	0.017213	0.999852	0.5	0.866025	-0.781831	0.623490
Algeria	Algiers	49.4	1995-01-02	2	1	0	0.034422	0.999407	0.5	0.866025	0.000000	1.000000
Algeria	Algiers	48.8	1995-01-03	3	1	1	0.051620	0.998667	0.5	0.866025	0.781831	0.623490
Algeria	Algiers	46.4	1995-01-04	4	1	2	0.068802	0.997630	0.5	0.866025	0.974928	-0.222521
Algeria	Algiers	47.9	1995-01-05	5	1	3	0.085965	0.996298	0.5	0.866025	0.433884	-0.900969

Table 2.2: Sample of the final preprocessed dataset with cyclic features

2.3.3 Models Building

Following dataset preparation and modeling pipeline setup, we will now describe the design of the two recurrent neural network architectures, LSTM and GRU, chosen due to their success in modeling temporal dependencies in sequential weather data.

2.3.3.1 LSTM model

As it is illustrated in Figure 2.2. The proposed LSTM model uses a multivariate input sequence of 160 timesteps and 7 features: : `AvgTemperature`, `sin_day`, `cos_day`, `sin_month`, `cos_month`, `sin_dow`, and `cos_dow`.

- `AvgTemperature` represents the average recorded temperature for each day.
- `sin_day` and `cos_day` represent the **day of the year** (from 1 to 365) encoded using sine and cosine transformations to preserve its *cyclical nature*, ensuring that day 1 and day 365 are considered close in time.
- `sin_month` and `cos_month` encode the **month of the year**, enabling the model to learn seasonal patterns without introducing discontinuities.
- `sin_dow` and `cos_dow` encode the **day of the week** (from Monday to Sunday), allowing the model to detect recurring weekly trends.

This combination of raw temperature data and cyclically encoded temporal features helps the model learn relationships influenced by both short and long-term time structures, such as daily, weekly, and yearly patterns.

The first layer of the model is a 1D Convolutional layer (Conv1D, with 32 filters). This layer detects local temporal patterns such as short-term fluctuations or trends in temperature over nearby days. The 32 filters mean the model learns 32 different local patterns from the input sequence, helping to improve feature extraction.

The next component is the first LSTM layer (LSTM 1) with 128 units. A larger number of units is chosen here to allow the model to capture rich and complex long-term dependencies from the high-dimensional input sequence. This wide representation helps in learning diverse temporal dynamics over extended time windows.

Following the second layer a dropout layer with a dropout rate of 0.1 follows, helping to prevent overfitting by randomly deactivating a portion of neurons during training.

The next layer is the second LSTM layer (LSTM 2) with 32 units. It reduces the dimensionality of the sequence representation while focusing on more abstract, higher level temporal features. The decrease in units reflects the model's transition from general temporal understanding to more distilled representations.

The fifth layer is the third LSTM layer (LSTM 3) with 16 units, the smaller number of units ensures that only the most relevant temporal patterns are retained for the final prediction.

Finally, the sixth and last layer is a fully connected (Dense) layer with a single neuron. This layer performs regression by mapping the learned representation to a scalar value that represents the predicted temperature for the next day.

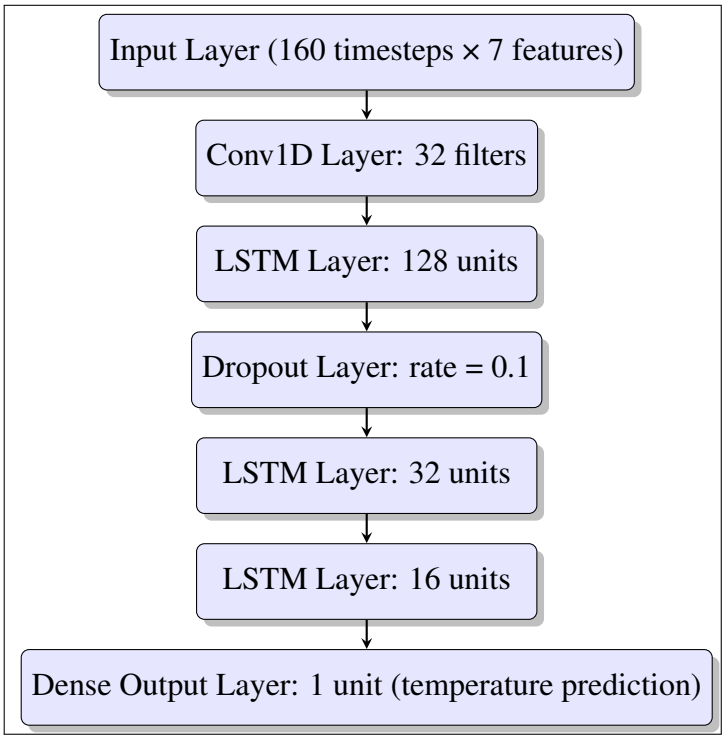

Figure 2.2: LSTM-Based Approch for Temperature Prediction

2.3.4 GRU Model

The GRU-based architecture illustrated in figure 2.3 also processes a multivariate input sequence that is comprised of 160 timesteps and 7 features, the same as the LSTM model inputs.

Similar to the LSTM model, the first layer is a 1D convolutional layer (Conv1D, with 32 filters), which facilitates identifying short-term temporal dependencies using 32 filters. This layer efficiently detects local patterns, such as short-lived trends or fluctuations that occur within a small time window of the input sequence.

Following the convolutional layer, the model includes a GRU layer (GRU 1) with 64 units, allowing the GRU layer to capture long-range dependencies in the input sequence. The GRU unit is designed to handle temporal relationships, much like the LSTM. However, unlike LSTM, the GRU uses a simpler gating mechanism that combines the forget and input gates into a single update gate. This makes GRU units computationally more efficient than LSTMs while still being capable of learning complex temporal patterns and retaining long-range dependencies in the data. The GRU helps the model maintain information across time steps, which is crucial for predicting future trends, especially for time series data like temperature predictions.

After the first GRU layer, a dropout layer is added with a rate of 0.3 to prevent overfitting. This layer randomly deactivates a fraction of the neurons during training, promoting more robust learning and helping the model generalize better to unseen data. The inclusion of dropout ensures that the model doesn't rely too heavily on any particular neuron, thereby enhancing its ability to handle new, unseen examples effectively.

The second GRU layer has 32 units (GRU 2) and processes the sequential representation from the first GRU layer. This layer compresses the learned knowledge into a more abstract and distilled representation of the input sequence. The output is a simplified, high level abstraction of the sequence, capturing only the key temporal features needed for accurate predictions.

The model concludes with a dense layer (Dense) consisting of a single neuron. This linear layer takes the compressed representation from the GRU layers and maps it to a scalar output, which is the predicted temperature value for the next day.

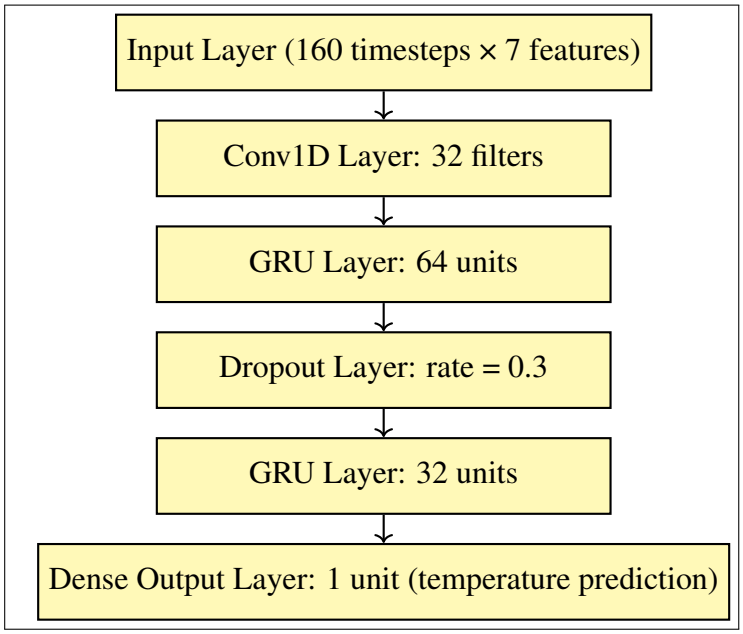

Figure 2.3: GRU-Based Architecture for Temperature Prediction

2.3.5 Evaluation

The evaluation and comparison steps of the model for the test set assess the generalization capacity of trained models applied to unseen data is undertaken after the models are trained. Several statistical measures are used to measure generalization performance or the models' ability to fit the data including the Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE) and coefficient of determination (R-squared) and are calculated using the following Equations:

a) **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Absolute Error (MAE) computes the average size of absolute values of errors in relation to predicted versus actual values.

b) **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mean Squared Error (MSE) measures the average of errors between predicted and actual values, highlighting larger deviations in predictor performance related to value-based deviations.

c) **Root Mean Squared Error (RMSE):**

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

The Root Mean Square Error (RMSE) is the square root of the average difference between predicted and actual values.

d) **Coefficient of Determination (R^2):**

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

The R^2 is a measure of the model's prediction accuracy on a scale of 0 to 1.

e) **Mean Absolute Percentage Error (MAPE):**

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Mean Absolute Percentage Error (MAPE) is the average absolute percentage difference between predicted and actual values.

```python
# Inverse transform only temperature
y_pred_inv = scaler.inverse_transform(y_pred_full)[:, 0]
y_test_inv = scaler.inverse_transform(y_test_full)[:, 0]

# Compute metrics
mse = mean_squared_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
r2 = r2_score(y_test_inv, y_pred_inv)
mape = np.mean(np.abs((y_test_inv - y_pred_inv) / y_test_inv)) * 100
accuracy = 100 - mape
```

Figure 2.4: Evaluation metrics calculations.

2.4 Conclusion

Within this chapter, a detailed overview of the design of the system was provided along with the reasoning behind each choice made in relation to the various components, from how datasets are handled to model architecture. A clear and thorough framework, understanding and selecting model choice in line with the problem, will serve the conception phase and the overall purpose of being able to capture the temporal dependencies involved in the weather data. In the following chapter, we will begin to move from the design to the implementation phase in terms of the proposed architecture to put the models to the test and assess and verify their performance through real-world results.

Chapter 3

Implementation and Results

3.1 Introduction

This chapter describes how the predictive models which were described in the second chapter are implemented and a detailed evaluation of their quality and performance with respect to several evaluation metrics. We start by describing the environment and development tools and how the models were created, trained, and tested. We place special emphasis on the GRU and LSTM architectures, which are trained with the time series data. We evaluated the models based on several error metrics, and their training process was monitored by loss curves and the prediction values were compared to actual values so that we could assess their accuracy. In the final section, we present comparisons to the models we have proposed and a comparative analysis with existing model, offering insights into the strengths and effectiveness of the approach.

3.2 Environment and development tools

The process discussed in the previous chapter was executed using a variety of languages, programming environments, and tools commonly utilized in deep learning projects. This section will explain the basic techniques for developing, building, and assembling the model.

3.2.1 Programming languages and frameworks

This section will delve into the software utilized in our project, including programming languages and libraries.

Python language

Figure 3.1: Python logo

Python, an open-source programming language, was created by Guido van Rossum in 1991. It is a versatile object-oriented language with a portable type

system, dynamic, extensible, free, simple syntax, and code shorter than C or Java. Python is multi-thread, scalable, and easy to read and write[Jav+19].

Tensorflow

Figure 3.2: Tensorflow logo

TensorFlow is a Google-developed open-source machine learning framework, offering a comprehensive ecosystem of tools, libraries, and community resources for researchers and developers to build and deploy effective models, particularly for deep neural network inference, making it a valuable tool in the field[Aba+19].

Numpy

Figure 3.3: Numpy logo

NumPy is a library for the Python programming language; it also adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to manipulate data, numerically compute and analyze data. Jim Hugunin developed Numpy, and it was incorporated by Travis Oliphant in 2005[Har+20].

Matplotlib

Matplotlib is a Python library that allows for the creation of static, animated, and interactive data visualizations, including line, bar, histogram, and scatter plots. It

Figure 3.4: Matplotlib logo

is commonly used alongside libraries like NumPy and pandas for data manipulation and analysis[Hun07].

3.2.2 Development Tools

The project utilized a diverse range of development tools and environments, with the main tools and environments being listed in this section.

Anaconda Environment

Figure 3.5: Anaconda logo

The Anaconda Environment feature enables the construction of isolated Python project environments. Each project environment has its own version of Python and set of packages associated with it; this is helpful in managing dependencies and preventing project contamination during development[Ana20].

Jupyter

Jupyter is an open-source community that provides interactive notebooks for the purpose of creating and sharing documents that have live code, equations, visualizations, and narrative text. It supports Python, R, and Julia and is widely used

Figure 3.6: Jupyter logo

in data science, scientific computing, and machine learning because it integrates code execution with rich, descriptive output[Klu+16].

3.2.3 Hardware Configuration

The experiments and model training were conducted on my personal laptop with the following hardware specifications:

- **Device Name:** DESKTOP-SPR59OH.
- **Processor:** Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz (up to 2.40GHz).
- **Installed RAM:** 8 GB.
- **System Type:** 64-bit operating system, x64-based processor.
- **Touch and Pen Support:** Not available.
- **Operating System:** Windows 10.

3.3 Implementation

A detailed breakdown of the models construction and training process provided below:

3.3.1 LSTM Implementation

The implementation is in line with the conceptual design and was confirmed using `model.summary()` shown in figure 3.7 to confirm the output shapes at each layer

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 156, 32)           192

 lstm (LSTM)                 (None, 156, 128)          82432

 dropout (Dropout)           (None, 156, 128)          0

 lstm_1 (LSTM)               (None, 156, 32)           20608

 lstm_2 (LSTM)               (None, 16)                3136

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 106,385
Trainable params: 106,385
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.7: model summary

The model underwent training and testing using an 80/20 split, with 15% of the training set reserved for validation.

The training configuration was designed to ensure effective learning and generalization of the model. The key training parameters and callbacks are summarized below:

- **Loss Function**: *Mean Squared Error (MSE)* was used, consistent with the regression nature of the temperature prediction task.
- **Optimizer**: *Adam* optimizer was used for its adaptive learning rate and efficiency on time series data.
- **Batch Size**: A batch size of 32 was maintained, balancing convergence stability and computational efficiency.
- **Epochs**: Initially set to 100, with actual epochs governed by *EarlyStopping* to avoid overfitting.
- **Callbacks**:
  - **ModelCheckpoint**: Saved the best-performing model based on validation loss.
  - **EarlyStopping**: Halted training if no validation improvement was observed over 10 consecutive epochs and restored the best weights.
  - **ReduceLROnPlateau**: Reduced the learning rate when validation loss plateaued, allowing finer adjustments in later epochs.

3.3.2 GRU Implementation

The GRU implementation followed the same preprocessing pipeline, data splitting, and training configuration used for the LSTM model. As in the LSTM experiment, the dataset was divided using an 80/20 split, with 15% of the training set held out for validation.

The only change between the two implementations lies in the model architecture itself, replacing LSTM layers with GRU layers. This adjustment was made to evaluate the performance of Gated Recurrent Units, which are known for their simpler gating mechanism and faster training time compared to LSTMs, while still capturing long-range dependencies in sequential data.

The model's architecture was confirmed using the `model.summary()` command and is illustrated in Figure 3.8, which shows the output shapes and layer configurations.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 156, 32)           1152

 gru (GRU)                   (None, 156, 64)           18816

 dropout (Dropout)           (None, 156, 64)           0

 gru_1 (GRU)                 (None, 32)                9408

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 29,409
Trainable params: 29,409
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.8: GRU model summary

The training setup was identical to the LSTM model to ensure a fair comparison between the two architectures.

3.4 Results

At the conclusion of the implementation and training stages, we are able to present the performance for each of the implemented architectures, starting with LSTM's results and then GRU's.

3.4.1 LSTM results

In order to further evaluate the findings of the LSTM model, we show the training and validation loss curves, then select representations of the actual vs the predicted temperature values, followed by a detailed analysis of the model's performance metrics.

3.4.1.1 Training and validation loss analysis

The training and validation loss curves are illustrated in Figure 3.9, which represents the model's mean squared error over 100 epochs.

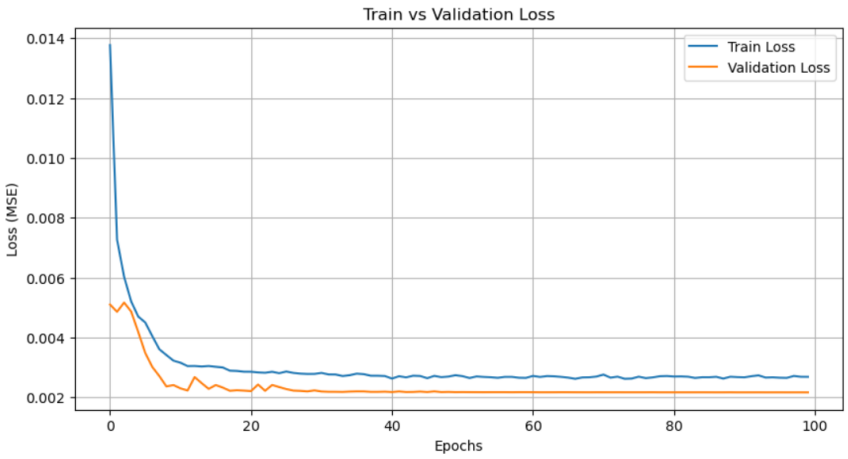

Figure 3.9: Training and validation loss during model training

As shown in Figure 3.9, the training loss consistently decreases, indicating that the model is learning effectively from the data. The validation loss initially follows the training loss, suggesting a good generalization. After a certain point, the validation loss stabilizes, which can be attributed to the early stopping mechanism preventing overfitting.

This convergence pattern confirms that the model training process was stable and successful.

3.4.1.2 Predictions based on LSTM

Figure 3.10 illustrates the comparison between the actual and predicted temperature values over time for Algiers for the test data.

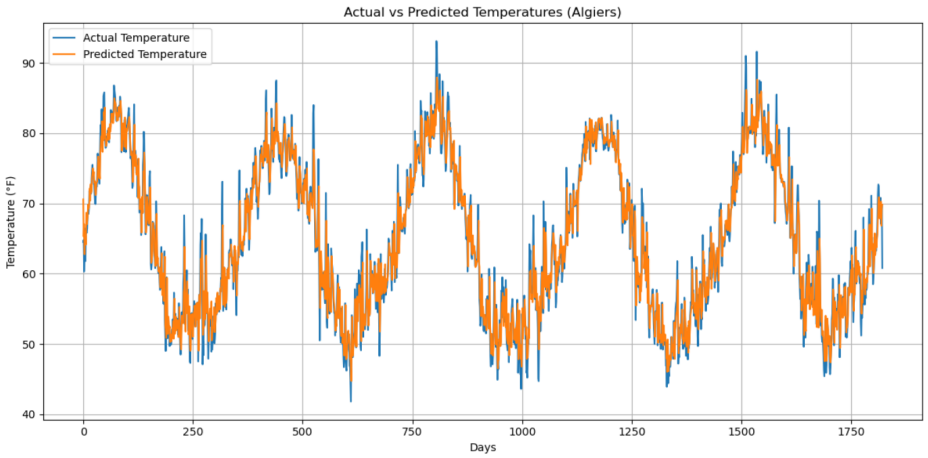

Figure 3.10: Actual vs. Predicted Temperature on Test Data

As shown in the figure, the predicted temperature values in orange closely follow the actual values in blue, indicating that the model has effectively captured the temporal patterns in the data.

Table 3.4.2 displays the predicted values through the LSTM model and the actual values from the last days of the test set for comparison purposes. The actual values are close to the predicted values, as in the case of the date 2020-05-07, highlighted in green, which is the closest to actual with the error of 1.45.

The results show that the LSTM model is capable of predicting the outcomes.

Date	**Actual Temperature (°F)**	**Predicted Temperature (°F)**
2020-05-05	72.5	69.39
2020-05-06	68.9	70.51
2020-05-07	68.9	67.45
2020-05-08	69.8	68.20
2020-05-09	70.8	69.01

Table 3.1: Comparison of Actual vs Predicted Temperatures.

3.4.1.3 Evaluation metrics results

Table 3.4.1 summarizes the model's performance on the test dataset.

Metric	Value
MAE	2.081
MSE	8.013
RMSE	2.831
R-squared (R^2)	0.932
MAPE	3.333%
Accuracy	96.667%

Table 3.2: Performance metrics on the test set

The **MAE** of 2.081 indicates that, on average, the model's predictions differ from the true temperatures by a distance of 2.08°F, which is low error for daily temperature predictions.

Recognized as one of the strongest metrics of error, **RMSE** had a value of 2.831 and indicated that the distribution of prediction errors did not have a large amount for larger outliers typically found in time series data.

A strong indicator of the model performance was the model's (R^2) value of 0.932 since this indicates approximately 93.2% of the variance in the actual temperature values is then explained in the model's predictions. This indicates a strong fit between the data and model predictions.

Complementing this was the value of **Mean Absolute Percentage Error-MAPE** at 3.333% which indicates that the model's predictions are, on average, 3.33% of the true values on average, while the calculated **accuracy**of 96.667% indicates the model was able to predict accurate predictions. Taken together, these results indicate that the LSTM architecture is able to learn complex temporal structure and fluctuations in the temperature data and does perform well when seeing unseen data.

3.4.2 GRU Results

We will evaluate the GRU model the way we did with the LSTM model.

3.4.2.1 Training and validation loss analysis

The GRU model's training and validation loss are shown in Figure 3.11, illustrating the model's mean squared error over 100 epochs.

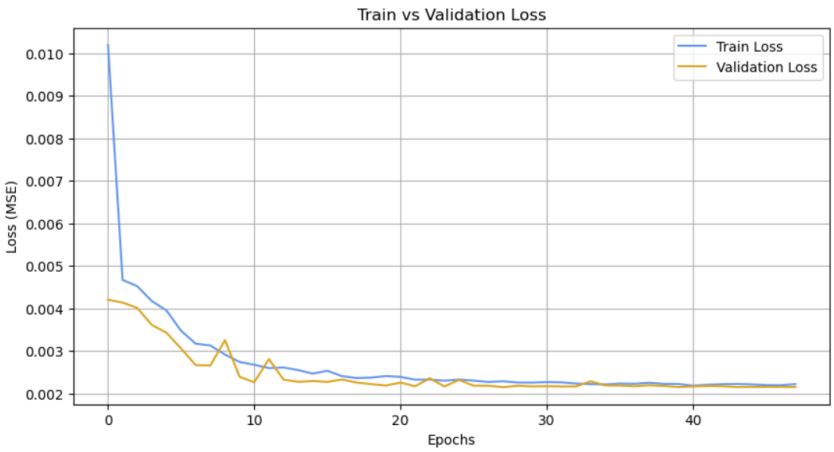

Figure 3.11: Training and validation loss during GRU model training

The training loss is shown in Figure 3.11 As can be observed, the training loss monotonically decreased over the epochs. This indicates consistent learning by the model from the training data. The validation loss became stable after showing a similar trend as the training loss and reinforces the initial model design to use early stopping in order to avoid overfitting. These results suggest that GRU model training is stable and successful overall.

3.4.2.2 Predictions based on GRU

Figure 3.12 compares the predicted and actual temperature values over time for Algiers using the test data.

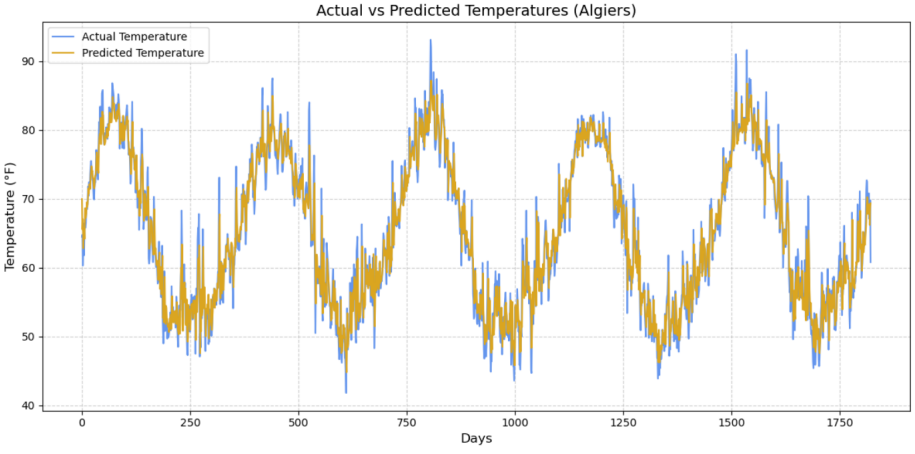

Figure 3.12: Actual vs. Predicted Temperature on Test Data (GRU)

The predicted temperatures, shown in orange, closely follow the actual values in blue, indicating that the GRU model successfully captured the time-based patterns present in the data.

Table 3.1 presents a comparison between the actual and predicted temperature values from May 5 to May 9, 2020. The predictions were generated using the GRU model and the results indicate a strong alignment between the actual and predicted values. Among them, the prediction for the date 2020-05-09, highlighted in green, demonstrates the closest match to the actual temperature with a 1.44 error.

This observation supports the effectiveness of the GRU model in capturing temporal dependencies and producing reliable temperature predictions.

Date	**Actual Temperature (°F)**	**Predicted Temperature (°F)**
2020-05-05	72.5	69.24
2020-05-06	68.9	70.44
2020-05-07	68.9	67.46
2020-05-08	69.8	68.47
2020-05-09	70.8	69.36

Table 3.3: Actual vs. Predicted Temperatures

3.4.2.3 Evaluation metrics results

The evaluation metrics for the GRU model's performance on the test dataset are summarized in Table 3.4.

Metric	**Value**
MAE	2.072
MSE	7.972
RMSE	2.824
R-squared (R^2)	0.933
MAPE	3.322%
Accuracy	96.678%

Table 3.4: Performance metrics on the test set using GRU

The **Mean Absolute Error (MAE)** of 2.072 indicates that, on average, the predicted temperatures differ from the actual values by approximately 2.07°F, reflecting a high level of precision. The **Root Mean Squared Error (RMSE)** of 2.824 supports this by showing that the prediction errors are generally small and not heavily affected by large outliers.

The model also achieved a strong **R-squared** (R^2) value of 0.933, suggesting that 93.3% of the variance in the actual temperature values is explained by the predictions, demonstrating a robust fit.

The **Mean Absolute Percentage Error (MAPE)** of 3.322% confirms that the predictions are, on average, within 3.32% of the true values. Complementing this, the **accuracy** of 96.678% underscores the model's effectiveness in delivering accurate and reliable temperature forecasts.

These results collectively indicate that the GRU model performs well in capturing temporal dependencies and delivering precise temperature predictions.

3.5 Discussion

Figure 3.13 presents a side by side comparison of performance metrics for both GRU and LSTM architectures.

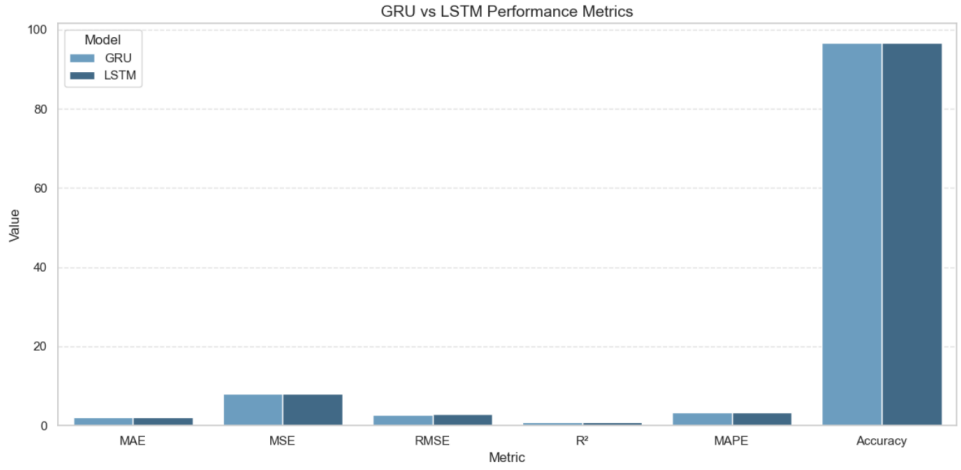

Figure 3.13: GRU vs. LSTM Performance Metrics

From the visualized data, both models exhibit highly competitive performance in predicting temperature, with only marginal differences across all metrics.

Model	**MAE**	**MSE**	**RMSE**	R^2	**MAPE**	**Accuracy**
GRU	2.072	7.972	2.824	0.933	3.322%	96.678%
LSTM	2.081	8.013	2.831	0.932	3.333%	96.667%

Table 3.5: Evaluation Metrics Comparison between GRU and LSTM

Given in the table 3.5, the GRU model slightly outperforms in terms of MAE, MSE, and RMSE. These lower error values suggest that GRU made slightly more precise predictions overall.

In terms of explained variance, GRU achieved an R^2 score of 0.933, compared to 0.932 for LSTM, indicating a marginally better ability to capture the underlying variance in temperature trends.

The MAPE values were close for both models, indicating they offer predictions within approximately 3.3% of the actual values on average.

Lastly, accuracy scores were virtually identical.

Overall, while both architectures demonstrate strong generalization and low error on the test data, GRU shows a slight edge in most metrics. Given its relatively simpler structure and faster training time, GRU may be the more efficient choice in practical scenarios without sacrificing prediction quality.

In the research of Jaharabi et al. [Jah+23], they found in their study that, in the Rio de Janeiro, SARIMA was the best modeling using MAE, MSE and RMSE, better than Prophet, ARIMA and LSTM. Based on that, they used the SARIMA model on the Delhi, in which it had MAE: 2.178, MSE: 7.419 and RMSE: 2.723. In comparison, we used Algiers first where we decided to use deep learning models that were LSTM and GRU. We determined that GRU was the best model and tested it on the Delhi as well to a fair comparison. In total, we had similar results on Delhi with the GRU with a MAE: 2.067, MSE: 7.296 and RMSE: 2.701 that were slightly better than what Jaharabi et al. [Jah+23]. reported with SARIMA. This would suggest that GRU has the capability of better performance than SARIMA, and for nonlinear and complex temperature patterns across various geographic datasets.

Model	**MAE**	**MSE**	**RMSE**
SARIMA	2.178	7.419	2.724
GRU (Ours)	2.067	7.296	2.701

Table 3.6: Performance comparison of SARIMA and GRU models

3.6 Conclusion

This chapter implemented and evaluated deep learning models to predict temperatures using past weather data. The results demonstrated great predictive performance: while GRU was slightly better than LSTM for most metrics, both architectures were able to achieve very low RMSE, MAE, MSE, and MAPE. Training and

validation losses indicated the models were learning consistently, and the curves predicted by the models were closely aligned to the actual temperature values. Moreover, the GRU model was successfully extended to a second city - Delhi - to compare with SARIMA, which demonstrated the model's robustness in comparison with traditional machine learning methods. This resulted further validated the multi-input model and opens up future real-life applications and further development.

General Conclusion

This thesis sought to show how deep learning models can be used, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, to predict temperature. This study employed a well-defined deep learning pipeline involving data processing, feature engineering, model architectural design, and model evaluation, which proved that these deep learning models can produce accurate predictions of daily temperature values.

Both LSTM and GRU networks were capable of appropriately characterizing the temporal dependencies of the meteorological data. Of these models, the GRU model produced slightly improved performance. The GRU network was able to produce accuracy levels on par with LSTM networks but with a simpler architecture and lower training time. All models performances were assessed using a series of evaluation metrics including MAE, RMSE, MAPE, and R^2. Each of these performance metric categories displayed solid predictive capability of both models, supporting their use to characterize time series forecasting applications.

It should be noted that this study does not intend to undermine the predictive capability of other machine learning or deep learning methods applied to weather forecasting; LSTM and GRU were selected because they have been historically demonstrated to have capability to model temporal dependencies in episodic time series. As this study initiates research into the applications of deep learning models for further weather prediction applications, the results suggest research into additional variables, multi-location forecasts. Future work may also include considering additional architectures such as hybrid models, attention or transformer architectures, in case they can provide better accuracy and better scalability.

In summary, this research supports the utility of recurrent neural networks for current weather predictions and serves as an inexpensive and effective alternative to predictive research approaches commonly used today.

Bibliography

[Aba+19] Martín Abadi et al. *TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow. org*. 2019.

[Al 24] Ziad Al Sarrih. "MASTERARBEIT/MASTER'S THESIS". In: (2024).

[Alz+21] Laith Alzubaidi et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of big Data* 8 (2021), pp. 1–74.

[Ana20] I Anaconda. *Anaconda documentation. Anaconda software distribution*. 2020.

[BTB15] Peter Bauer, Alan Thorpe, and Gilbert Brunet. "The quiet revolution of numerical weather prediction". In: *Nature* 525.7567 (2015), pp. 47–55.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[BJ23] Belen Bermejo and Carlos Juiz. "Improving cloud/edge sustainability through artificial intelligence: A systematic review". In: *Journal of Parallel and Distributed Computing* 176 (2023), pp. 41–54. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2023.02.006. URL: https://www.sciencedirect.com/science/article/pii/S0743731523000217.

[Ber23] David M Berry. "The limits of computation: Joseph Weizenbaum and the ELIZA chatbot". In: *Weizenbaum Journal of the Digital Society* 3.3 (2023).

[Bi+19] Qifang Bi et al. "What is machine learning? A primer for the epidemiologist". In: *American journal of epidemiology* 188.12 (2019), pp. 2222–2239.

[Box+76] George E. P. Box et al. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[CHH02] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. "Deep blue". In: *Artificial intelligence* 134.1-2 (2002), pp. 57–83.

[CPS13] Abhimanyu Chopra, Abhinav Prashar, and Chandresh Sain. "Natural language processing". In: *International journal of technology enhancements and emerging engineering research* 1.4 (2013), pp. 131–134.

[CC20] KR1442 Chowdhary and KR Chowdhary. "Natural language processing". In: *Fundamentals of artificial intelligence* (2020), pp. 603–649.

[Col+21] Christopher Collins et al. "Artificial intelligence in information systems research: A systematic literature review and research agenda". In: *International Journal of Information Management* 60 (2021), p. 102383.

[DYH22] Hendri Darmawan, Mike Yuliana, and Moch Zen Samsono Hadi. "Realtime Weather Prediction System Using GRU with Daily Surface Observation Data from IoT Sensors". In: *2022 International Electronics Symposium (IES)*. IEEE. 2022, pp. 221–226.

[Fan+21] Wei Fang et al. "Survey on the Application of Deep Learning in Extreme Weather Prediction". In: *Atmosphere* 12.6 (2021). ISSN: 2073-4433. DOI: 10.3390/atmos12060661. URL: https://www.mdpi.com/2073-4433/12/6/661.

[Fer+25] Andrea Filippo Ferraris et al. "The architecture of language: Understanding the mechanics behind LLMs". In: *Cambridge Forum on AI: Law and Governance*. Vol. 1. Cambridge University Press. 2025, e11.

[Gra12] Alex Graves. "Long Short-Term Memory". In: *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–45. ISBN: 978-3-642-24797-2. DOI: 10.1007/978-3-642-24797-2_4. URL: https://doi.org/10.1007/978-3-642-24797-2_4.

[Han+21] Xu Han et al. "Pre-trained models: Past, present and future". In: *AI Open* 2 (2021), pp. 225–250. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.08.002. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000231.

[Har+20] Charles R Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (2020), pp. 357–362.

[Has+19] Irtiza Hasan et al. “Head Pose Estimation and Trajectory Forecasting”. In: (2019).

[Hos+15] Moinul Hossain et al. “Forecasting the weather of Nevada: A deep learning approach”. In: *2015 international joint conference on neural networks (IJCNN)*. IEEE. 2015, pp. 1–6.

[Hun07] John D Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in science & engineering* 9.03 (2007), pp. 90–95.

[HA18] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. 2nd ed. OTexts, 2018. URL: https://otexts.com/fpp2/.

[Jah+23] Wasiou Jaharabi et al. “Predicting temperature of major cities using machine learning and deep learning”. In: *arXiv preprint arXiv:2309.13330* (2023).

[Jak06] Vikramaditya Jakkula. “Tutorial on support vector machine (svm)”. In: *School of EECS, Washington State University* 37.2.5 (2006), p. 3.

[Jav+19] Aaquib Javed et al. “An analysis on python programming language demand and its recent trend in bangladesh”. In: *Proceedings of the 2019 8th international conference on computing and pattern recognition*. 2019, pp. 458–465.

[KHA21] Shahab Kareem, Zhala Jameel Hamad, and Shavan Askar. “An evaluation of CNN and ANN in prediction weather forecasting: A review”. In: *Sustainable Engineering and Innovation* 3.2 (2021), pp. 148–159.

[Khe24] Hamza Kheddar. “Transformers and large language models for efficient intrusion detection systems: A comprehensive survey”. In: *arXiv preprint arXiv:2408.07583* (2024).

[Klu+16] Thomas Kluyver et al. “Jupyter Notebooks–a publishing format for reproducible computational workflows”. In: *Positioning and power in academic publishing: Players, agents and agendas*. IOS press, 2016, pp. 87–90.

[Lee+18] Kyungmin Lee et al. “Accelerating recurrent neural network language model based online speech recognition system”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 5904–5908.

[Liu+24] Yiheng Liu et al. “Understanding llms: A comprehensive overview from training to inference”. In: *arXiv preprint arXiv:2401.02038* (2024).

[Ltd23] Ltd Co. *Artificial Intelligence Technology*. Jan. 2023.

[Lug24] George F Luger. “LLMs: Their Past, Promise, and Problems.” In: *International Journal of Semantic Computing* 18.3 (2024).

[Mah+20] Batta Mahesh et al. “Machine learning algorithms-a review”. In: *International Journal of Science and Research (IJSR).[Internet]* 9.1 (2020), pp. 381–386.

[Man+24] Nikoleta Manakitsa et al. “A Review of Machine Learning and Deep Learning for Object Detection, Semantic Segmentation, and Human Action Recognition in Machine and Robotic Vision”. In: *Technologies* 12.2 (2024). ISSN: 2227-7080. DOI: 10.3390/technologies12020015. URL: https://www.mdpi.com/2227-7080/12/2/15.

[ME22] Eduardo F Morales and Hugo Jair Escalante. “A brief introduction to supervised, unsupervised, and reinforcement learning”. In: *Biosignal processing and classification using computational learning and intelligence*. Elsevier, 2022, pp. 111–129.

[Muk+23] S Karthik Mukkavilli et al. “Ai foundation models for weather and climate: Applications, design, and implementation”. In: *arXiv preprint arXiv:2309.10808* (2023).

[Nav+23] Humza Naveed et al. “A comprehensive overview of large language models”. In: *arXiv preprint arXiv:2307.06435* (2023).

[Ngu+24] Andy Nguyen et al. “Enhancing student engagement through artificial intelligence (AI): Understanding the basics, opportunities, and challenges”. In: *Journal of University Teaching and Learning Practice* 21.6 (2024), pp. 1–13.

[Pan+22] Jianguo Pan et al. “GRU with dual attentions for sensor-based human activity recognition”. In: *Electronics* 11.11 (2022), p. 1797.

[Pet09] Elia Georgiana Petre. “A decision tree for weather prediction”. In: *Bul. Univ. Pet.–Gaze din Ploieşti* 61.1 (2009), pp. 77–82.

[PH22] Steven T Piantadosi and Felix Hill. “Meaning without reference in large language models”. In: *arXiv preprint arXiv:2208.02957* (2022).

[PK19] Zhaoxia Pu and Eugenia Kalnay. “Numerical weather prediction basics: Models, numerical methods, and data assimilation”. In: *Handbook of hydrometeorological ensemble forecasting* (2019), pp. 67–97.

[RS09] Y Radhika and M Shashi. “Atmospheric temperature prediction using support vector machines”. In: *International journal of computer theory and engineering* 1.1 (2009), p. 55.

[Ran16] Rajib Rana. “Gated recurrent unit (GRU) for emotion classification from noisy speech”. In: *arXiv preprint arXiv:1612.07778* (2016).

[Sil+16] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[Sto16] Peter Stott. “How climate change affects extreme weather events”. In: *Science* 352.6293 (2016), pp. 1517–1518.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems* 27 (2014).

[Val23] Athanasios Valavanidis. “Artificial intelligence (ai) applications”. In: *Department of Chemistry, National and Kapodistrian University of Athens, University Campus Zografou* 15784 (2023).

[Wer90] Paul J Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[Xu+24] Qiwei Xu et al. “Evaluating the Performance of a Stacking-Based Ensemble Model for Daily Temperature Prediction”. In: *American Journal of Environmental Science and Engineering* 8.3 (2024), pp. 79–85.

[You+21] Kaichao You et al. “Logme: Practical assessment of pre-trained models for transfer learning”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12133–12143.

[ZE16] Mohamed Akram Zaytar and Chaker El Amrani. “Sequence to sequence weather forecasting with long short-term memory recurrent neural networks”. In: *International Journal of Computer Applications* 143.11 (2016), pp. 7–11.

[Zha+23] Wayne Xin Zhao et al. “A survey of large language models”. In: *arXiv preprint arXiv:2303.18223* (2023).