



UNIVERSITÀ
DEGLI STUDI
FIRENZE

PHD PROGRAM IN SMART COMPUTING
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)

Modeling Similarity: Theory and Applications of Siamese Networks

Filippo Costanti

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

Modeling Similarity: Theory and Applications of Siamese Networks

Filippo Costanti

Advisor:

Prof. Monica Bianchini

Head of the PhD Program:

Prof. Stefano Berretti

Evaluation Committee:

Prof. Filippo Geraci, *Istituto Informatica Telematica, CNR*

Prof. Linda Brodo, *University of Sassari*

Abstract

Siamese networks have a broad range of applications, spanning from computer vision tasks to speech recognition and graph-structured data processing. Their strength is in learning the similarity between inputs, allowing the construction of an embedding space that effectively captures relationships and patterns within the data. This makes the exploration and modeling of Siamese embedding spaces crucial to improving their performance across different domains.

This thesis contributes to the research on Siamese networks by presenting how they can be used to devise a soft estimation of the Graph-Edit Distance, which allows to solve the NP-complete graph isomorphism problem in an affordable time. Then, three distinct applications of Siamese networks are presented, related to two prediction and one classification tasks. In all of these applications, the embedding space is constructed based on the similarity target, without relying on the classical Siamese loss functions, such as contrastive or triplet losses. These examples showcase how Siamese networks can build powerful embedding spaces even without traditional loss functions, underlining their flexibility and effectiveness in solving real-world tasks across different fields.

Contents

Contents	1
1 Introduction	3
2 Siamese networks	7
2.1 The Siamese model	7
2.2 Neural Networks for Structured Data	11
3 NeuraGED: A GNN estimation of the Graph–Edit Distance	17
3.1 Introduction	17
3.2 Materials and Methods	20
3.3 Experimental Setup	23
3.4 Results and Discussion	24
3.5 Conclusion	26
4 LSTM–based Siamese Networks for Fault Detection in Meteorological Time Series Data	27
4.1 Introduction	27
4.2 Materials and Methods	29
4.3 Results	33
4.4 Discussion	36
4.5 Conclusions	37
5 Bioinformatics approach to investigate structural and non–structural proteins in human coronaviruses	39
5.1 Introduction	39
5.2 Materials and Methods	42
5.3 Results	46
5.4 Discussion	53
5.5 Conclusion	56
6 Other works	59

6.1	A Deep Learning Approach to Analyze NMR Spectra of SH-SY5Y Cells for Alzheimer's Disease Diagnosis	59
6.2	A Deep Learning approach for oocytes segmentation and analysis . .	62
7	Conclusions and Future Perspectives	65
A	Publications	69
	Bibliography	73

Chapter 1

Introduction

As the digital landscape evolves, deep learning has become a critical force in pushing the boundaries of what artificial intelligence can achieve. The introduction of Large Language Models (LLMs) such as ChatGPT, Llama, BERT, and others, has revolutionized the concept and role of artificial intelligence in everyday life. Its impact, although seemingly in its early stages, appears so significant that it requires a reconsideration of many legal and social regulations. It even prompts a reevaluation of society's concept of human intelligence. Despite this, the growing need for execution speed and process automation, along with the increasing availability of complex data, requires new deep learning techniques and the extension of existing ones. Among the various deep learning techniques and models, this thesis focuses on the application of a specific class of models known as Siamese networks, which have been successfully utilized in areas such as image similarity matching, face verification, text similarity, and anomaly detection. This research explores the potential of these models in real-world tasks, including similarity prediction and fault detection.

Siamese neural networks were introduced for the first time by Bromley *et al.* [1] to address the problem of signature verification. These networks typically consist of two identical branches that process input pairs in the same manner through various types of weight-sharing mechanisms. Siamese networks are powerful architectures in deep learning, used across a wide range of domains for supervised, self-supervised, or unsupervised learning tasks. The key strength of Siamese networks lies in their ability to focus on learning the similarity between inputs, allowing them to create an embedding space that effectively captures the relationships between the input data. As a result, the representations of input data are based on their degree of similarity or dissimilarity, which is reflected in their respective distances within the embedding space. This feature provides several significant advantages in the context of deep learning applications. Firstly, when addressing real-world challenges, the use of deep learning models can be hindered by the limited availability of labeled

data. By leveraging shared weights, Siamese networks reduce the number of model parameters, thereby minimizing the risk of overfitting and promoting better generalization, particularly in scenarios with limited data. Conversely, in cases where large amounts of data are available, Siamese networks can be utilized to generate pseudo-labels by leveraging the inherent similarities between data points. This capability makes them particularly suitable for unsupervised or semi-supervised learning applications.

Additionally, Siamese networks can leverage pairwise comparisons to generate a much larger set of training data from a small dataset. For example, a dataset of size n can generate $n(n - 1)/2$ unique pairs for training, significantly expanding the training set and allowing the network to learn from a broader range of comparisons. This approach is particularly valuable for tasks like face recognition or image verification, where each class may only have a few labeled examples. By making effective use of these pairwise comparisons, the network can still learn to distinguish between inputs, despite the limited amount of labeled data. Such a property of Siamese networks is closely aligned with the principles of one-shot learning, where the objective is to identify new classes based on just one or a few examples. Siamese networks are particularly well-suited for this task because they are trained to generalize based on relationships between samples rather than relying on memorizing specific features of each class. As a result, the network can be trained on a set of known classes and subsequently generalize to entirely new classes with minimal additional data, making this approach particularly valuable in scenarios where collecting large labeled datasets is infeasible.

While Siamese networks are capable of constructing embedding spaces, it can be challenging to ensure that the learned space is well-structured, particularly when dealing with complex or noisy data. To address this, Siamese networks are typically designed to use loss functions such as contrastive or triplet loss. These losses operate on the principle of mapping similar examples into neighboring regions of the embedding space, ensuring that the space reflects the original similarity of the data. In contrast, using other loss functions, such as *MSE* or *cross-entropy*, may not guarantee this property, potentially limiting the network's ability to capture the true similarity relationships in the data.

This thesis presents the following contributions to the research on Siamese networks.

1. It is demonstrated that Siamese networks can be used to estimate the Graph-Edit Distance, which allows facing the NP-complete graph isomorphism problem in an accessible time [2]. Indeed, this first study focuses on determining the degree of similarity between structured data, such as graphs. In this case, the network is used to embed graphs and calculate their distance, allowing the supervised learning of graph distance between inputs. Molecular

graph generators can therefore use this distance estimation as a powerful non-permutation-invariant reconstruction loss.

2. Siamese networks are applied to fault detection, specifically predicting faults in time series data from meteorological sensors [3]. The proposed workflow compares faulty versions of the series with reconstructed versions that represent normal behavior, simulating a real-world scenario of incoming data and determining the probability of a fault in real-time.
3. A bioinformatics application is presented in which a Siamese network is employed to predict the BLAST-like distance between the primary structures of homologous proteins [4]. The results of these tasks are promising and highlight the strength of Siamese networks in constructing an embedding space without relying on traditional loss functions. Instead, they utilize top-performing networks that map pairs of embeddings to specific outputs depending on the task at hand.

The rest of the thesis is organized as follows. Chapter 2 introduces the Siamese model and briefly describes other networks used in this thesis, also providing a brief literature review. Chapter 3 discusses the application of Siamese networks for predicting graph similarity, while Chapter 4 presents the fault detection application on time series data. In Chapter 5 can be found the BLAST-like similarity prediction of HCoV proteins. Chapter 6 briefly sketches two other research lines developed during the PhD period. Finally, conclusions and future perspectives are collected in Chapter 7.

Chapter 2

Siamese networks

This chapter introduces the Siamese model and briefly describes other networks used in this thesis — detailing their structure, applications, and advantages —, also providing a brief literature review. The networks discussed are designed to handle structured data types such as graphs and time series. Time series can be considered a specific form of rooted tree, and thus a directed graph, where each element in the series represents a node with only a single descendant.

2.1 The Siamese model

Siamese networks are employed in a wide range of applications. In the field of computer vision [5], they are particularly well-suited for content-based image retrieval [6] and when training data is scarce. In visual tracking, Siamese networks are used to compare the target object with candidate regions across video frames, estimating the object’s location and size in real-time applications, including autonomous driving, virtual reality, and smart traffic [7, 8, 9, 10, 7, 11]. Furthermore, in image matching tasks, Siamese networks excel at comparing the similarity of objects in input images. Image matching finds broad application in areas such as remote sensing [12], 3D indoor scene reconstruction [13], and astronomical research [14]. Siamese networks are also frequently applied to tasks involving graph-structured data. They are particularly effective in computing similarities between graphs [15], which enhances performance across various tasks such as image retrieval [16], social network analysis [17], and semantic code clone detection [18]. Additionally, they are utilized in natural language processing, for tasks like sentence similarity and textual entailment, where they measure the semantic similarity between sentences and phrases, with applications in machine translation, question answering, and information retrieval [19, 20, 21]. Finally, in text-based image retrieval, Siamese networks help connect textual descriptions to corresponding images or audio, facilitating more efficient search and retrieval processes [22, 23].

General structure

Siamese core architecture usually consists of three main components, as can be seen in Figure 2.1: i) an embedding layer that realizes a feature mapping, ii) a merge layer, or simply a function that compares the embedded samples, and iii) a decision-making layer, usually a specific task neural network.

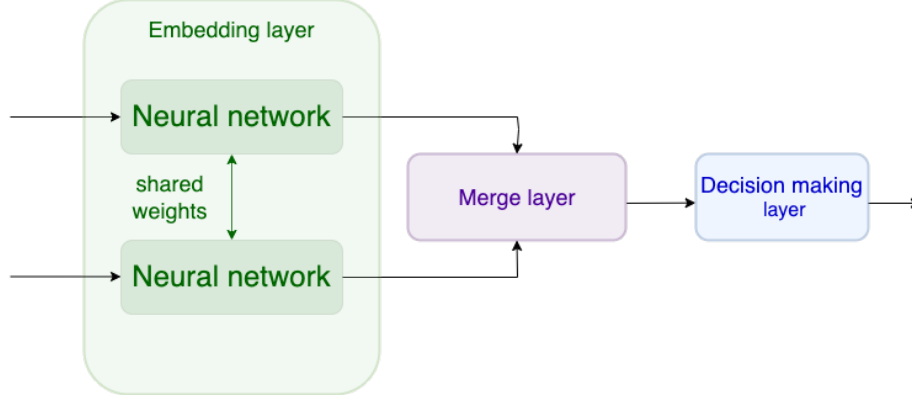


Figure 2.1: Structure of the Siamese network.

The embedding layer is made by two branches composed of two identical neural blocks, each receiving a separate input. These twin blocks share the same parameters, allowing them to process each input in an equivalent manner, producing their embeddings, as described in Equation 2.1:

$$E_i = f(x_i), E_j = f(x_j) \quad (2.1)$$

The embeddings E_i and E_j corresponding to the input data x_i and x_j , respectively, are obtained by the same neural function f . The shared weights of the blocks make them particularly suitable for comparison tasks, as both inputs are processed with the same transformations, ensuring that any differences in the output are due to the inputs themselves and not to discrepancies in processing. The twin neural blocks can be networks of any type, including Time-Delay Neural Networks (TDNNs [24]) as in Bromley *et al.*, Convolutional Neural Networks (CNNs [25]), any kind of Recurrent Neural Networks (RNNs[26]), Graph Neural Networks (GNNs [27]), depending on the nature of the data to be processed.

The two embeddings are then compared by the merge layer. Such a comparison can be carried out using either a neural function or directly a distance metric, such as the Euclidean distance (Equation 2.2) or the cosine similarity (Equation 2.3), to quantify the similarity or dissimilarity between the inputs:

$$d_{euc}(E_i, E_j) = ||E_i, E_j||_2 \quad (2.2)$$

$$d_{cos}(E_i, E_j) = \frac{E_i \cdot E_j}{||E_i|| \cdot ||E_j||} \quad (2.3)$$

While the Euclidean distance enhances the comparison of vector magnitudes by bringing representations of similar data closer together during the training phase, the cosine distance is independent of the magnitude of the embeddings and instead focuses solely on the angle between them. In certain cases, the merge layer can be omitted as shown in Chapter 3 and Chapter 4.

The decision-making layer is usually a MultiLayer Perceptron (MLP) that implements the output function for a specific task, either classification or regression.

There are several variants of Siamese networks that utilize three or more branches instead of the traditional two. These architectures process n -tuples of data simultaneously. In the case of a three-branch setting, the network takes a triplet of data as input, where one sample acts as the anchor with a known class [28]. The network calculates the distance between the embeddings of the two remaining samples, referred to as the positive and negative samples, relative to the anchor (Figure 2.2). In this context, the anchor serves as the reference sample, with the positive and negative patterns being similar or different to the anchor, respectively. Triplet network is mainly tailored to classification tasks with similarity comparison.

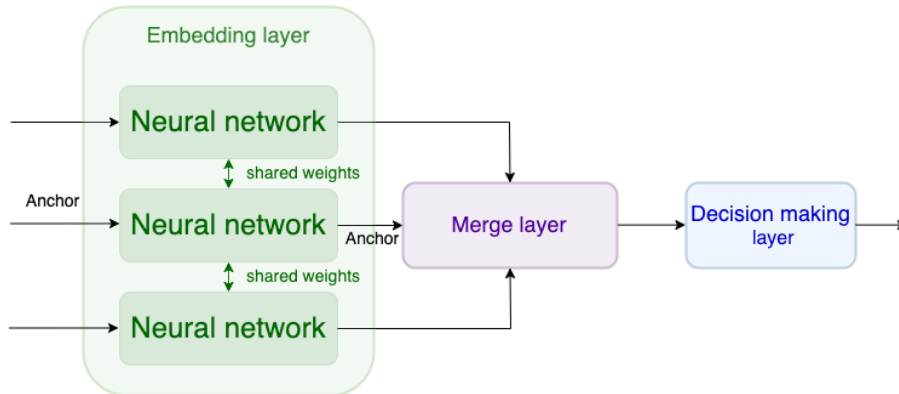


Figure 2.2: Generic structure of the three branches Siamese network.

The model is trained by minimizing the distance between the anchor and the positive sample, ensuring that positive samples are represented by embeddings closer to the anchor compared to the negative ones.

Siamese losses

There are various loss functions that can be employed to train a Siamese neural network, and in principle, any task-specific loss can be applied. For instance, binary or categorical *cross-entropy* is commonly used in classification tasks, Mean Square Error (*MSE*) is utilized for prediction tasks, and reconstruction loss is employed in

generative settings. Additionally, mixed versions of these losses can be combined to address multiple objectives. Specific losses can also be added to train particular parts of the network, such as link prediction and entropy loss in hierarchical graph pooling, as seen in DiffPool [29]. In similarity tasks, the primary loss function used in Siamese networks is the contrastive loss, which is often regarded as the native loss for such architectures. Contrastive loss (Equation 2.4) is defined as :

$$\mathcal{L}_{contrastive} = y_{ij} \cdot \frac{1}{2} \cdot \mathcal{D}(E_i, E_j)^2 + (1 - y_{ij}) \cdot \frac{1}{2} \cdot \max(0, m - \mathcal{D}(E_i, E_j))^2 \quad (2.4)$$

Here, y_{ij} is the label for the pair (i, j) , where $y_{ij} = 1$ indicates that i and j are similar, and $y_{ij} = 0$ otherwise, $\mathcal{D}$ represents one of the distance metrics computed by the merge layer, such as the Euclidean distance ($d_{euc}(\cdot)$) or the cosine similarity ($d_{cos}(\cdot)$). The parameter m denotes a margin that specifies the minimum separation between dissimilar pairs. This loss function encourages the network to minimize the distance between similar pairs while maximizing the separation between dissimilar pairs. For similar couples ($y_{ij} = 1$), the loss drives the network to reduce the distance $\mathcal{D}$ between their embeddings. Conversely, for dissimilar pairs, the loss enforces that the distance between their embeddings be at least as large as the margin m , thus separating dissimilar pairs in the embedding space.

In the triplet variant of the Siamese network, the input consists of triplets rather than pairs. The corresponding loss function, known as triplet loss, aims to learn an embedding space where the distance between the anchor and the positive sample is smaller than the distance between the anchor and the negative sample by at least a margin m . The triplet loss is formally defined in Equation 2.5 below:

$$\mathcal{L}_{triplet} = \max(0, \mathcal{D}(E_a, E_p) - \mathcal{D}(E_a, E_n) + m) \quad (2.5)$$

where $E_a = f(x_a)$ is the embedding of the anchor sample x_a , $E_p = f(x_p)$ and $E_n = f(x_n)$ are the embeddings of positive (x_p) and negative (x_n) samples, respectively. The triplet loss encourages the distance between the anchor and the positive sample to be smaller than the distance between the anchor and the negative one by, at least, the margin m , i.e. until $\mathcal{D}(E_a, E_p) > \mathcal{D}(E_a, E_n) - m$. If the difference between the distances to the positive and negative samples relative to the anchor is already greater than m (i.e., $\mathcal{D}(E_a, E_n) - \mathcal{D}(E_a, E_p) > m$), the loss becomes zero, and no further adjustment is required.

The primary limitation of these loss functions becomes evident when the dataset lacks a clear similarity structure, that is, when it is not feasible to partition the data into similar and dissimilar samples or assign them to specific classes composed of similar instances. Consequently, this thesis investigates the potential of incorporating non-Siamese losses into the Siamese architecture to address such scenarios for both classification and regression tasks.

2.2 Neural Networks for Structured Data

The composition of the two branches that constitute the Siamese architecture depends on the data to be processed. This thesis focuses exclusively on networks designed to process structured data — especially sequences and graphs —, which are typical of most real-world problems. Therefore, in the rest of this chapter, are briefly characterized structured data and *ad hoc* architectures to process them, namely Recurrent Neural Networks (RNNs) and Graph Neural Networks (GNNs).

Structured Data

Structured data have a hybrid nature, both symbolic and sub-symbolic, and cannot be represented independently from the links between some basic constitutive entities [30]. Symbolic information (collected in feature vectors) is used to label each base entity (a node). Instead, the sub-symbolic information is carried by edges between nodes, which represent relationships, symmetrical or not, such as inclusion, adjacency, presence of a chemical bond, etc. Edges can also be labeled, in order to characterize the relationship they describe. Both entities and their relationships can be homogeneous throughout the structure or not. In a way that may seem counter-intuitive, The section begins by describing the most complex structured data, that is by giving the definition of a graph, since sequences can be considered as very simple graphs, just as the networks that are normally used to process them (recurrent networks) can be considered a special case of the GNN model.

A *graph* is a non-linear, non-Euclidean data structure defined as a pair $G = (V, E)$, where V represents the set of nodes and $E \subseteq V \times V$ is the set of edges. Nodes describe entities, while edges stand for the relationships between them. Nodes and edges can be associated with values or vectors of values describing their features — $x_v \in \mathbb{R}^{d_v}$, $\forall v \in V$, and $e_{u,v} \in \mathbb{R}^{d_e}$, $\forall (u, v) \in E$, respectively. Attributes which are attached to each node or edge/arc are called labels. In general, it is assumed that a consistent labeling space is used for all of the nodes and edges constituting the graph. Graphs may also be heterogeneous, meaning that different entities can be collected within a single graph, where relationships of different nature coexist. The *stacked* versions of all the node and edge feature vectors constitute, respectively, the node feature matrix $\mathbb{X}_V \in \mathbb{R}^{|V| \times d_v}$ and the edge feature matrix $\mathbb{X}_E \in \mathbb{R}^{|E| \times d_e}$. Given a node $v \in V$, the set of its one-hop neighborhood is indicated as $\mathcal{N}(v)$.

Trees can be considered as a special type of directed graph, where each node has a unique parent, while there is no limit to the number of its descendants — therefore, a tree in which each node has at most m children is referred to as an m -ary tree. Furthermore, trees are (commonly) rooted, i.e. there is a node from which all the other nodes can be reached, giving rise to a structure that easily represents a

hierarchical arrangement of the basic entities that it collects. Finally, *sequences* are 1-ary trees, as the (labeled) entities follow one after the other (sometimes in a timely manner).

Recurrent Neural Networks

Recurrent Neural Networks (RNNs [26]) are a class of artificial neural networks designed to recognize patterns in sequential data. Unlike traditional feedforward networks, RNNs have connections that form directed cycles, allowing them to retain information from previous inputs and use it in the current processing step, realizing a sort of "memory". This capability makes RNNs particularly powerful for tasks where context or temporal dependencies are critical, such as time series forecasting, Natural Language Processing (NLP), speech recognition, and video analysis. The core feature of RNNs is their hidden states, which store information about previous inputs. As the network processes data at each time step, it updates its hidden state using feedback connections, thus influencing future outputs based on past data.

Formally, given the vector state $h_{t-1} \in \mathbb{R}^n$, at time $t - 1$, and the network input $x_t \in \mathbb{R}^d$, the state $h_t \in \mathbb{R}^n$, at t , is given by:

$$h_t = f_\theta(h_{t-1}, x_t) \quad (2.6)$$

where the learnable function f_θ remains the same for all time steps t_i , and the input sequence is $S = (x_0, x_1, \dots, x_s)$, with $x_i \in \mathbb{R}^d, i = 0, \dots, s$. f_θ is a neural network, usually an MLP. Equation 2.6 can be rewritten as:

$$h_t = g(W_\theta \cdot h_{t-1} + W_\phi \cdot x_t + b) \quad (2.7)$$

where $W_\theta \in \mathbb{R}^{n \times n}$, $W_\phi \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, and g is an activation function.

RNNs are trained using a variant of Backpropagation called Backpropagation Through Time (BPTT [26]). BPTT unrolls the network over the entire sequence of inputs and computes gradients based on the sequence's time steps. However, BPTT faces notable challenges, such as vanishing or exploding gradients, which make training deep RNNs difficult. These issues arise because, as gradients are propagated back over many time steps, they can either shrink to near-zero values (vanishing gradient), hindering the network's ability to learn long-term dependencies, or grow exponentially large (exploding gradient), making the training process unstable.

Despite these challenges, RNNs provide a powerful framework for learning from sequential data. Their advanced variants, such as Long Short-Term Memory (LSTM [31]) and Gated Recurrent Units (GRU [32]), were developed to address these gradient issues, allowing them to maintain long-term dependencies effectively.

Long Short-Term Memory

To address the challenge of long-term dependencies, LSTMs introduce internal gating mechanisms that selectively retain useful information from the memory. These gates include the forget gate, input gate, state candidate gate and output gate, that are composed by neural networks, which output are combined to obtain the cell state vector $c_t \in \mathbb{R}^n$ at each time step. The LSTM gate is depicted in Figure 2.3.

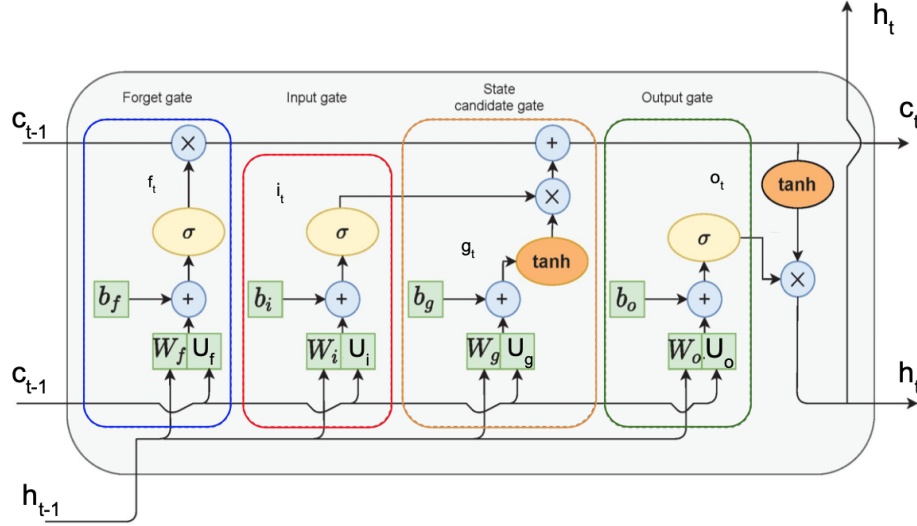


Figure 2.3: LSTM structure and gates.

The forget gate f_t regulates which parts of the previous memory need to be discarded:

$$f_t = \sigma(W_f \cdot h_{t-1} + U_f \cdot x_t + b_f) \quad (2.8)$$

The input gate i_t controls how much of the new information from the current input should be added to the cell state:

$$i_t = \sigma(W_i \cdot h_{t-1} + U_i \cdot x_t + b_i) \quad (2.9)$$

Finally, the output gate o_t determines how much of the cell state should influence the output:

$$o_t = \sigma(W_o \cdot h_{t-1} + U_o \cdot x_t + b_o) \quad (2.10)$$

In Equations 2.8, 2.9 and 2.10, f_t, i_t and o_t are learnable functions, $W_f, W_i, W_o \in \mathbb{R}^{n \times n}$, $U_f, U_i, U_o \in \mathbb{R}^{n \times d}$, $b_f, b_i, b_o \in \mathbb{R}^n$ are the network parameters, and σ is the *sigmoid* activation function. Initializing $c_0, h_0 = 0$, the cell state vector c_t and the internal state h are updated following Equations 2.11 and 2.12 below:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_g \cdot h_{t-1} + U_g \cdot x_t + b_g) \quad (2.11)$$

$$h_t = o_t \odot \tanh(c_t) \quad (2.12)$$

where $W_g \in \mathbb{R}^{n \times n}$, $U_g \in \mathbb{R}^{n \times d}$, $b_g \in \mathbb{R}^n$ and $\odot$ is the element-wise product.

Graph Neural Networks

Graph Neural Networks (GNNs) are a well-established class of machine learning models designed for processing graph-structured data.

The foundations of GNNs were laid in 2005, in [33], with a formal mathematical formulation presented in 2009 [27]. The original GNNs, also known as Recurrent Graph Neural Networks (RecGNNs), were designed to mirror the topology of the input graph, allowing for message passing between neighboring nodes, to generate an output based on the selected data. The core idea in RecGNNs is the propagation of information across the graph through recurrent updates, where nodes iteratively exchange information with their neighbors and refine their representations over several steps of neighbor aggregation. This process allows the same network to be reused at each step, reducing the number of training parameters. However, this recurrent structure introduces the risk of vanishing or exploding gradients, which can make training difficult and lead to unstable learning dynamics especially for large graphs.

Since their introduction, other types of message passing mechanisms have been formulated. Spectral Graph Neural Networks (SpecGNNs [34]) are a class of GNNs that utilize principles from spectral graph theory, operating primarily in the Fourier domain by applying transformations to the graph Laplacian. The fundamental concept of SpecGNNs is to perform graph convolution as a multiplication in the frequency domain, leveraging the eigenvectors and eigenvalues of the graph Laplacian. These models capture global graph properties, making them powerful for tasks where understanding the entire graph structure is critical. However, the computational burden of calculating the eigendecomposition of the graph Laplacian severely limits their scalability to large graphs.

Convolutional Graph Neural Networks (ConvGNNs [35]) are a type of GNN specifically designed to perform convolution-like operations on graph-structured data. They extend the concept of convolution from traditional grid-based data, such as images, to graphs, which can exhibit varying sizes and connectivity patterns. Similar to traditional GNNs, ConvGNNs operate by exchanging information (messages) between nodes in a graph. Each node aggregates information from its neighbors and updates its own representation.

In general, given a graph G , a ConvGNN represents the initial information of its nodes by storing it in internal node state vectors $h_v^{(0)}$, $\forall v \in V$. The node state vectors are then updated using the following state update function (Equation 2.13):

$$h_u^k = \gamma_{\Omega_k}^{(k)} \left(h_v^{(k-1)}, \bigoplus_{u \in \mathcal{N}(v)}^{(k)} \left(\phi_{\Theta_k} \left(h_v^{(k-1)}, h_u^{(k-1)}, e_{v,u} \right) \right) \right) \quad (2.13)$$

Here, $\bigoplus^{(k)}$ denotes a differentiable, permutation-invariant function (e.g., sum, multiplication, mean) that combines messages from the neighbors of node v , while $\phi_{\Theta_k}^{(k)}$ is a differentiable function, referred to as the message function, representing the message passed from node $u \in \mathcal{N}(v)$ along edge $e_{v,u}$. The differentiable function $\gamma_{\Omega_k}^{(k)}$, known as the state update function, updates the node’s state. Θ_k and Ω_k are learnable parameters.

ConvGNNs typically consist of multiple layers, each implementing a message-passing step that aggregates information from a node’s first-order, second-order neighbors, and so on. Each differentiable module within the network can theoretically employ different message, aggregation, and update functions. Intuitively, a ConvGNN updates node representations by aggregating messages received from neighboring nodes, where the depth k indicates that each layer captures information from up to k -hop neighbors. In other words, by initializing the node states as the feature vector of each node, $h_u^{(0)} = x_u, \forall u \in V$, after k message-passing layers, each node’s state $h^{(k)}$ will incorporate both the topology and feature information from its k -hop neighbors, effectively embedding the surrounding graph structure and node attributes within its representation. This enables the network to represent both local and increasingly global graph structures as the information propagates through multiple layers, thereby capturing more comprehensive features across the graph. However, when using a very deep ConvGNN, with many layers, node representations tend to converge to similar values, irrespective of their initial features or the underlying graph structure. This issue, known as over-smoothing, causes a loss of crucial and distinctive information, reducing the model’s ability to differentiate between nodes effectively. For the remainder of this thesis, ConvGNNs will be referred to as GNNs.

Three types of tasks can be faced by means of GNNs: node-focused, edge-focused, and graph-focused problems. Node-focused tasks involve scenarios where each node, or a subset of nodes, in the graph is associated with a target value. The goal is to produce an output for each targeted node, which can be used for classification, regression, or clustering purposes. For example, when a macro-molecule is represented as a graph, localizing a particular compound within the molecule is a node-focused task, as the output is specific to individual nodes. Graph-focused tasks involve predicting or classifying properties of entire graphs, rather than individual nodes or edges, making them ideal for tasks like molecular property prediction, where the goal is to predict an entire molecule’s properties based on its structure as a graph. Finally, edge-focused tasks concentrate on predicting or classifying

relationships between node pairs, which is useful in applications like link prediction, where the objective is to determine whether two nodes in a graph should be connected.

Chapter 3

NeuraGED: A GNN estimation of the Graph–Edit Distance

This first application of Siamese networks regards the possibility of comparing structured data such graphs. Comparing graphs, finding a distance or a soft similarity score or a soft isomorphism degree between them, could be difficult because of their complexity and heterogeneity. The difficulty of the graph isomorphism related problem stems from the fact that checking for isomorphism requires exploring the space of all possible vertex mappings between the two graphs, which can be computationally expensive, particularly for large graphs. The problem becomes more challenging with regular graphs or graphs with symmetries, as they have multiple equivalent mappings, making it hard to determine isomorphism efficiently. Graph distance could be seen as a degree of similarity between two graphs and therefore a soft estimation of their isomorphism. Finding a light and efficient way to measure distances between graphs is a crucial problem in tasks like protein–protein interaction, drug discovery and graph generation. Especially in graph–generation tasks, it is crucial to explore reconstruction losses that provide an accurate measure of the similarity between the reconstructed and the original encoded graphs. This chapter introduces a model, named NeuraGED, designed to effectively approximate the Graph–Edit Distance (GED), enabling its use as a neural reconstruction loss in graph generation tasks.

3.1 Introduction

The problem of generating graph data involves training a generative model that can produce new graph instances resembling those observed in a given dataset. Unlike other structured data, such as images or text, graphs exhibit unique challenges, due to their complexity and heterogeneity. This makes graph generation a difficult and crucial task in many domains, including social network analysis, rec-

ommendation systems, biological network modeling, and design of new chemical compounds. The main issue when dealing with graphs is, of course, their discrete nature: graphs or their encodings must be brought into a continuous domain, because discrete spaces hinder the possibility of calculating gradients and therefore applying Backpropagation-like algorithms to train generative models. Traditional generative approaches [36] struggle to capture intricate topological patterns, community structures, and attribute distributions present in complex graphs. By using neural networks to operate directly on graph-structured data, GNNs [27] can capture hierarchical relationships and local dependencies present in graphs, enabling effective generative modeling. However, generating realistic graph data poses several key challenges. First, the generated graphs must possess statistical properties and structural characteristics similar to those of the input dataset, including the degree distribution, average path length, and clustering coefficient. Additionally, the generated graphs should preserve the semantic meaning and functional properties encoded in the node and edge attributes, if available. Several approaches have been proposed to address these challenges, including sequential graph generation based either on recurrent models [37, 38, 39] or reinforcement learning [40, 41], Variational Graph Autoencoders (VGAEs [42, 43, 44]), Graph Generative Adversarial Networks (GGANs [45, 46]), and diffusion models for graph generation [47, 48]. Variational graph autoencoders learn a low-dimensional latent space representation of graphs, allowing for efficient sampling and interpolation between graph instances. Graph generative adversarial networks employ an adversarial training mechanism to generate realistic graphs by competing against a discriminator network. Sequential models iteratively generate nodes and edges conditioned on previously generated components, capturing the sequential nature of graph generation.

One of the main problems in training generative models is the choice of an adequate loss function. For one-shot models, such as variational autoencoders, the most commonly used loss functions is the Kullback–Leibler (KL) divergence [49] combined with a reconstruction loss, such as the mean squared error (*MSE*). *MSE* can compute the distance between two matrices but, in general, it does not guarantee to calculate an adequate topological distance for two adjacency matrices representing the connectivity of two graphs. In fact, there are many possible adjacency matrices for every single graph, one for each enumeration of the nodes. This means that comparing different adjacency matrices of the same graph can lead to inadequate values of the reconstruction error. Two main functions are designed for solving this problem: Graph Edit Distance (GED [50]) and Sub-graph Edit Distance (SED [51]). Unlike the SED, which violates the properties of identity and symmetry, the GED is considered a distance metric [52, 53] invariant to node order permutations. GED has an important application in inexact graph matching, especially useful in fault-tolerant pattern recognition [54]. Additionally, it has interesting applications

in bioinformatics: for instance, it can be used to measure the distance between two molecular graphs [55, 56], a particular kind of labeled graph in which the node labels indicate the type of each atom and the edge labels indicate the type of each chemical bond. Moreover, having a model which computes the similarities or dissimilarities between molecular graphs could be functional also for implementing an efficient reconstruction loss for deep generative models. Unfortunately, the tasks of computing GED and SED have been demonstrated to be NP-hard [57]. Recently, some works have employed GNNs to estimate GED. In [52] a Siamese structure with Graph Isomorphism Networks (GINs) is used to predict GED and SED, with multiple MLPs for performing features encoding, decoding, and distance of embeddings. More complex is the structure of the solution reported in [58]: here, graphs are processed by two Graph Convolutional Networks (GCNs [59]) to produce node-focused embeddings, which are subsequently processed by a graphs global attention network to obtain graph-focused embeddings, and by a sub-module to obtain a node wise attention embedding. These embeddings are then concatenated and processed by an MLP to predict a GED-like similarity score. Despite achieving good performance, those approaches cannot be used in a generation task, since the number of nodes in the generated graphs is constrained by the dataset used to train the model (*e.g.* LINUX [60], AIDS¹ and IMDB [61] datasets), making the computation impossible for new datasets. Approximation of GED and SED could also be useful in many application fields, such as social sciences or bioinformatics [62], for example to locate similar subgraphs within larger graphs — to find similar communities in social networks [63] — or for matching similar and complementary substructures in drug discovery [64, 65, 66] and for protein-protein interaction prediction [67].

In this chapter, a procedure to be used in graph generative frameworks is proposed, which consists in training a neural network to calculate a topological soft estimation of the GED using a simpler architecture compared with the literature, setting up the task as a supervised regression problem on a synthetic graph dataset generated for this purpose. The model consists of a GNN-based Siamese network, specifically, constituted by two identical subnetworks sharing the same parameters and architecture each taking a different input. The key aspect of the Siamese model lies in its ability to learn a similarity function, thus representing the ideal choice for the computation of the GED. The remaining section are organized as follows. In section 3.2, the GED learning problem and the dataset generation are described, together with the model structure; in Section 3.3, the experimental set-up is reported while the obtained results are discussed in Section 3.4. Finally, Section 3.5 collects some conclusions and depicts future perspectives.

¹<https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data>

3.2 Materials and Methods

Problem definition

Let $G = (V, E, X_V, X_E)$ be a generic undirected, labeled graph, s.t. $G \in \mathcal{G}$, where $\mathcal{G}$ is a set of graphs. As defined in 2.2, V and E are the set of nodes and edges respectively. The matrices $X_V \in \mathbb{R}^{n_n \times d_v}$ and $X_E \in \mathbb{R}^{n_e \times d_e}$ denote the node and edge features matrices, where n_n and n_e correspond to the number of nodes and edges, respectively.

The GED between two graphs G_1 and G_2 in $\mathcal{G}$, denoted as $\text{GED}(G_1, G_2)$, is defined as the minimum number of edit operations which transform G_1 in G_2 , where an edit operation on the graph G_1 can be either a single insertion or a single deletion of a node or an edge, or even a node relabeling. The GED between two identical graphs, *i.e.* between two isomorphic graphs with the same node and edge features, is therefore defined as zero.

In this study, *NeuraGED* — a GNN-based approximation of the graph edit distance — is proposed. The model is trained on a dataset $\mathcal{L}_{\mathcal{G}} = \{\langle G_1, G_2, y \rangle : y = \text{GED}(G_1, G_2) \in \mathbb{R} \mid G_1, G_2 \in \mathcal{G}\}$ of tuples, where $y \in \mathbb{R}$ is the ground-truth value for $\text{GED}(G_1, G_2)$, with $y(G_1, G_2) = y(G_2, G_1)$.

Dataset generation

The primary challenge associated with the GED calculation is its computational complexity. It has been proven that this problem belongs to the NP-hard class of complexity [57]: the NP-hardness of GED implies that computing the exact distance between two graphs is intractable for large or complex graphs, since it requires a prohibitive amount of time and resources. In fact, many algorithms have been developed for the computation of GED, employing both exact and approximate methods. Exact algorithms, such as exhaustive search, calculate the precise GED between two graphs by exploring all possible sequences of editing operations. This results in a complexity of $O((n_1 + m_1)!(n_2 + m_2)!)$, where n_1 and n_2 are the number of nodes in the first and second graph, respectively, and m_1 and m_2 are their corresponding number of edges. Other exact methods, such as dynamic programming and branch-and-bound, are characterized by exponential complexity. In contrast, approximate approaches, such as bipartite matching, have a polynomial complexity ($O(n^3)$).

Hence, in order to obtain a soft estimation of this distance, the generation of a specific dataset, in which the target distance for each couple of samples is known, is proposed.

The ZINC dataset [68], a common and widely used dataset in drug discovery and molecular modeling, was utilized as a reference in this study [69]. ZINC contains information on the molecular structures, properties, and features of various

chemical compounds, comprising molecular graphs with a maximum of 38 atoms, described by a set of 27 features. The generated dataset is composed of ordered triplets $\langle G_1, G_2, y \rangle$, where $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ describe two simple undirected and node-labeled graphs, *i.e.* undirected graphs which do not allow for parallel edges nor self-loops, while y represents the GED between G_1 and G_2 . In order to simulate the insertion and deletion edit operations, and to guarantee a general balance of the dataset, the first half of the data was constructed so that G_2 is generated as an extension of G_1 , *i.e.* with only the addition operation allowed in the generation procedure, while the second half of the dataset is built by exchanging the position of G_2 and G_1 inside the ordered triplets, so that only the deletion operation is considered. Let $\mathbf{G}$ be the set of undirected simple graphs and let $\mathbb{B}^{n \times m}$ be the set of all $n \times m$ binary matrices. The structure of the graphs is then constructed through a random generation of their adjacency matrices by means of a function $f: \mathbb{R} \rightarrow \mathbf{G}$, which generates a squared matrix $\mathbf{A} = [a_{ij}] \in \mathbb{B}^{k_1 \times k_1}$, given a number of nodes $k_1 = |V_1|$, sampled from a specific normal distribution. Moreover, since $G_1, G_2 \in \mathbf{G}$, the generated adjacency matrices are binary symmetric matrices with all zeros on the diagonal. In this context, isolated nodes are not allowed: if there are one or more nodes with no incident edges (*i.e.* null rows and therefore null columns in $\mathbf{A}_1$), random non-diagonal elements are set to 1. Given the two adjacency matrices $\mathbf{A}_1 \in \mathbb{B}^{k_1 \times k_1}$ and $\mathbf{A}_2 \in \mathbb{B}^{k_2 \times k_2}$, with $k_1 \leq k_2$, the topological estimation of the GED is computed as defined in Eq. (3.1):

$$y = n + \sum_{\substack{k_1 \leq i \leq k_2-1 \\ 0 \leq j \leq k_1-1}} a_{i,j} + \frac{1}{2} \sum_{\substack{k_1 \leq i \leq k_2-1 \\ k_1 \leq j \leq k_2-1}} a_{i,j}. \quad (3.1)$$

The two summations in Eq. (3.1) count the number of elements equal to 1 in the two submatrices that can be extracted from $\mathbf{A}_2$ if only the rows associated to the added vertices are considered, hence the rows not belonging to $\mathbf{A}_1$. Fig. 3.2 shows the GED estimation from a graphical point of view: the sum of the second submatrix, colored in red, is divided by two since undirected graphs are considered and because of the symmetrical nature of $\mathbf{A}_2$, otherwise the corresponding edges would be counted twice.

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Figure 3.1: Adjacency matrix $\mathbf{A}_1$

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.2: Adjacency matrix $\mathbf{A}_2$

In order to measure the performance of the proposed model with different initialization methods for node labels, and to generate the final samples $\langle G_1, G_2, y \rangle$, the associated feature matrices $\mathbf{X}_1 \in \mathbb{R}^{k_1 \times d_v}$ and $\mathbf{X}_2 \in \mathbb{R}^{k_2 \times d_v}$ are generated as well, together with the target y and the maximum diameter D for each pair. On the one hand, if a generic graph G is connected, then its diameter $\text{diam}(G)$ is computed. On the other hand, if G is not connected, its diameter would be infinite, therefore the relative sample is discarded. Hence, four possible situations can occur:

- Both G_1, G_2 are disconnected, then $D = 0$;
- G_1 is connected and G_2 is disconnected, then $D = \text{diam}(G_1)$;
- G_1 is disconnected and G_2 is connected, then $D = \text{diam}(G_2)$;
- Both G_1, G_2 are connected, then $D = \max\{\text{diam}(G_1), \text{diam}(G_2)\}$.

NeuraGED Model

NeuraGED is a Siamese model that utilizes twin GNNs as its embedding layers, without a merge layer, and employs an MLP as the decision–making layer, which outputs the GED between two input graphs. In particular, Siamese networks for graphs [70, 71] approximate well the embedding function $f_1 : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}^h \times \mathbb{R}^h$, where $\mathcal{G}$ is the set of graphs and h is the dimension of the embedding space.

GNNs have been demonstrated to be universal approximators on graphs [72]. Moreover, it has been proved that a GNN with a sufficient number of message passing iterations and an injective readout function is as powerful as the first–order Weisfeiler–Lehman test. This holds true both for recurrent GNNs and for some convolutional GNN models [73]. As a consequence, the choice of GINs — which were purposefully designed for maximizing the isomorphism recognition capability — seems to be natural. GINs calculate message–passing iterations in stacked convolutional layers which do not share the weights, according to Equation (3.2):

$$h_v^{(k)} = \text{MLP}^{(k)} \left(\left(1 + \epsilon^{(k)}\right) h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right), \quad (3.2)$$

where $h_v^{(k)}$ is the state of the node v at layer k , $\epsilon^{(k)}$ is a learnable parameter at layer k , and $\mathcal{N}(v)$ is the one–hop neighborhood of node v . Once the two graphs G_1 and G_2 are fed in input to the Siamese GIN, after k message–passing iterations (carried out by the same number of layers), the state of every node represents a natural node embedding (k is a hyperparameter of the network). The two graph embeddings are obtained using a global pooling with max, average or sum aggregation mode.

After the graph embeddings E_1 and E_2 are computed, they are concatenated and fed in input to a MLP which implements the function $f_2 : \mathbb{R}^h \times \mathbb{R}^h \rightarrow \mathbb{R}$, such that $f_2(E_1, E_2)$ approximates $\text{GED}(G_1, G_2)$, given that MLPs are universal approximators [74]. Setting the activation function of the last layer of the MLP with a *ReLU* guarantees the non negativity of f_2 . Moreover, given that GNNs are universal approximators on graphs [75, 76], and it can be concluded that f_2 maps the two embeddings in the GED value of the correspondent pair of graphs [77]. Figure 3.3 shows the structure of the NeuraGED model.

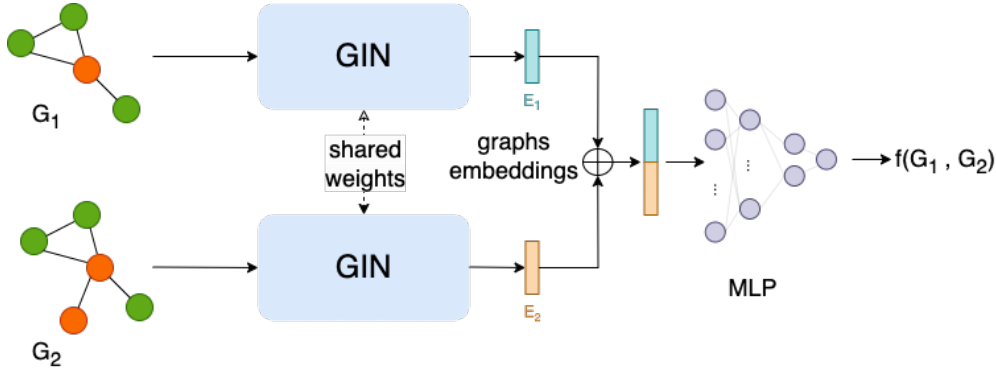


Figure 3.3: Architecture of NeuraGED

3.3 Experimental Setup

Experiments have been performed on a Intel(R) Core(TM) i7-9800X CPU with 3.80GHz equipped with 64 GB RAM. The dataset consists of 120.000 pairs of graphs, 70% of which compose the training set, 10% the validation set, and 20% the test set. Once the number of message passing layers has been fixed to 3, equal to the maximum diameter of connected graphs in the dataset, the first step of the training procedure consists in a random search of the network hyperparameters. For the GIN state updating function, a MLP with only one hidden layer has been chosen. The number of neurons in the hidden layer depends on the node state dimension, varying between $2h$ and $3h$, h being the node state dimension. The activation function of the MLP module inside the GIN layers is fixed to *SeLU* and the global pooling method is always *sum*. *SeLU* is an activation function designed to ensure data auto-normalization without requiring additional batch normalization layers, thereby stabilizing the training of GNNs that may include multiple message-passing layers.

Since the MLP implementing the distance function f_2 takes in input the concatenated embeddings of the two graphs, its layout depends on the node state dimension h . While the input layer is always of dimension $2h$ and the output layer is always composed of one unit, the number of hidden layers change within 2 and 3 and their

units within $2 \cdot h, 10 \cdot h$, for 2 layers, and $2h, 10h, 10h$, for 3 layers. The activation function used is *LeakyReLU* for all layers except the final one, which utilizes *ReLU*. The selection of *LeakyReLU* as the internal activation function for the MLP is driven by its computational simplicity, robustness, and its ability to prevent the dead neuron problem. Moreover, the final layer employs *ReLU* to guarantee a non-negative output suitable for approximating the natural numbers—typically greater than 1—required for GED.

In Table 3.1 are reported the combinations of node state dimension, GIN hidden layers, pooling type, and number of hidden layer neurons for the MLP.

Table 3.1: Experimental setup for NeuraGED. The table summarizes the parameters used in the experiments. As it can be seen, values from the second and fourth columns depend on the values of the first column, *i.e.* GIN hidden layer and MLP hidden layer sizes are multiples of the node state dimension. The best setup is reported in bold.

State dimension	GIN hidden layer	Pooling type	MLP hidden layers
64	128	sum	[128 , 640]
64	192	sum	[128 , 640]
128	256	sum	[256 , 1280]
128	384	sum	[256 , 1280]
128	256	sum	[256 , 1280, 1280]
128	384	sum	[256 , 1280, 1280]

The training of the model is carried out using *MSE* as a loss function and the *Adam* optimizer, with a learning rate of 10^{-6} , a batch size of 512, and a number of epochs equal to 1000, with early stopping based on the validation loss.

To measure the performance of the network configuration with each chosen set of parameters, a 5-fold *cross validation* has been applied, with patience for *early stopping* equal to 15. Each training inside the cross validation has been performed changing the initialization of feature matrices of the graphs.

3.4 Results and Discussion

The used metric is the MSE which measures the distance between target GED value and the one predicted by the network as in 3.3:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y - \hat{y})^2 \quad (3.3)$$

where y represents the value predicted by the network, $\hat{y}$ denotes the corresponding target, and n is the total number of samples.

Experiments show that the best model is the one having node state dimension of 128, GIN hidden layer size of 256, *sum* global pooling and MLP with two hidden layers of 256 and 1280 neurons.

In Table 3.2, the best results of the cross-validation are reported.

Table 3.2: Comparison of results obtained with different feature matrix initialization methods. For each method, the average *MSE* on the test set and its standard deviation are reported.

Feature matrix	Average test <i>MSE</i>	Standard deviation
ones	0.1343	0.0754
random uniform	1.9211	0.2822
random normal	14.4601	2.5295

As it can be appreciated from the results shown in Table 3.2, NeuraGED can identify with a low error a broad variety of graph structures. This behavior can be obtained with different methods of feature matrix initialization, indicating that the model is capable of filtering out (at least to a certain degree) this unrelated information, focusing only on the topology. However, the predictions obtained with

Table 3.3: Comparison of predicted values of the network and target, for a selection of couples in test set.

Number of nodes in the first graph	Number of nodes in the second graph	Target	Predicted target
21	28	98.0	97.61
23	24	12.0	11.77
25	34	146.0	145.36
22	26	53.0	53.15
20	24	55.0	55.09
26	30	62.0	62.14
22	24	28.0	28.17
19	26	93.0	92.76
27	31	62.0	62.32
18	36	269.0	267.88

feature matrices sampled from a distribution in the interval $[0, 1)$, referred in Table 3.2 to as *random uniform*, or a standard normal distribution, referred in Table 3.2 to as *random normal*, are not as accurate as those obtained in the case of feature matrices initialized with all the entries equal to 1. This indicates that the features introduce some noise in the GED estimation process, even if the GED is purely topological. Additionally, it is possible to observe that the sparsity of the random normal feature

matrices can hinder the learning process, but this could be avoided by shifting the mean of the normal distribution. Table 3.3 shows a selection of examples, randomly chosen from the test dataset compared to the target.

3.5 Conclusion

In this chapter, NeuraGED, a Siamese method for computing a soft–estimation of the Graph Edit Distance (GED) based on siamese Graph Neural Networks (GNNs) was presented. An exact computation of the GED is unfeasible because it constitutes a NP–hard problem and cannot be carried out for large graphs. The proposed model demonstrates the effectiveness of GNN–based Siamese models and provides a reliable and computationally affordable estimation of the GED, with very low error rates. This method can be useful for several tasks on graph data that require an evaluation of a topological distance between two structures. For instance, it could be used to compute the distance between two molecular graphs, even when dealing with large molecules. This is potentially very useful when building graph generators, as the proposed method could be used for calculating a reliable, fast, informative, and permutation–invariant reconstruction loss. Such loss could substantially improve the training procedure of many graph generative models, which typically rely on less informative and non–permutation–invariant reconstruction loss functions.

Regarding future research, it would be interesting to add a term accounting for the dissimilarities between the feature vectors to the target estimation equation. For instance, this term could be based on the euclidean distance between features. Of course, another possible development to pursue could be the addition of edge labels, consequently obtaining an estimation of the distances between the edge feature vectors. These latter two steps would extend the proposed computation to labeled graphs, making the distance much more informative in this domain. Moreover, it could be functional to add a normalized precomputed supervision, for instance by applying a logarithmic formulation.

Chapter 4

LSTM-based Siamese Networks for Fault Detection in Meteorological Time Series Data

This chapter focuses on the use of Siamese networks for fault detection. It presents an application of Siamese networks in a real-world fault detection scenario, where the data consists of time series collected from meteorological sensors at the farm "La Selva" of Istituto Tecnico Agrario B. Ricasoli of Siena. Given the sequential nature of the time series data, the embedding modules in the two branches of the Siamese network are implemented using LSTMs. The proposed pipeline simulates the scenario where data is continuously received from sensors and the network must detect faults by providing a probability score of such an event occurring. In this context, and with this type of data, Siamese networks prove to be effective in comparing reconstructed and faulty time series.

4.1 Introduction

Fault recognition and anomaly detection involve the identification of irregularities within the data, which deviate from their normal behavior, thus signaling the presence of errors, outliers or unexpected trends. Indeed, faults and anomalies can significantly degrade the quality of datasets, leading to biased models, inaccurate predictions, and failures to generalize to new data [78]. Therefore, a pre-processing activity aimed at identifying inconsistent data is crucial to guarantee their quality and to ensure robustness and reliability of neural network models.

Apart from traditional techniques, such as statistical methods and time-series analysis, supervised and unsupervised machine learning models and deep learning networks can be trained to recognize unexpected data patterns that deviate from the normal distribution [79][80]. In fact, in recent years, Artificial Intelligence

(AI)-based tools have gained momentum in the elaboration of huge amounts of data, thanks to their ability to process and extract meaningful features from complex datasets. More specifically, examples of Machine Learning (ML)-based and Deep Learning-based fault detection methods have been developed, for both temperature and humidity sensors. Methodologies in these works range from the usage of Multilayer Perceptron (MLP) networks [81] to the combination of Principal Component Analysis (PCA) and bidirectional Long-Short Term Memory (Bi-LSTM) networks [82] and complex systems combining AutoEncoder networks (AE), Support Vector Machines (SVMs) and Fuzzy logic [83] to perform fault detection and diagnosis. Current techniques achieve variable performance, with fault detection accuracies ranging from 80% to 100% depending on the type of sensor and sensor fault that has to be detected.

In this chapter, a fault detection approach is applied to a set of meteorological data experimentally collected, specifically temperature and humidity time series, during field tests of several months in a rural environment. To this aim, the low-cost low-power sensor network presented in [84] was employed, specifically tailored for monitoring the microclimate of an agricultural area. Since the number of detected faults in the original dataset was limited, a synthetic data generation approach was applied to mimic the most probable faults, thus augmenting the size and enriching the variety of the original dataset. Hence, the automatic recognition of faults and anomalies has a twofold advantage: it may help to improve the quality of the experimental dataset, by signaling erroneous data and preventing them from influencing future trainings; and it allows to early detect faults in an automatic way, thus avoiding the manual detection which can be time-consuming and less reliable in case of large datasets.

The proposed approach is based on a three-step pipeline: 1) first, a Long Short-Term Memory (LSTM) model is trained to predict the next value in the time series; 2) then, a new dataset made of predicted series and data with synthetic faults is built; 3) finally, a Siamese network with LSTMs as twin branches is trained in a supervised framework to binary classify sequences as containing faults or not, thus giving an alert as soon as a new fault is detected and providing as output a probability value of the presence of a fault.

The cooperation between sensor networks and ML approaches is nowadays a trend topic with great potential for enhancing the sensing capabilities of the monitoring systems and for enabling data-driven decision-making [85]. LSTMs, thanks to their recurrent structure and their ability to retain solely relevant features in temporal data processing, are widely used for predicting time series of meteorological data [86][87] and generally for analyzing any sequential data. Siamese networks, on the other hand, are capable of comparing two inputs to find similarities and can be applied to a variety of problems, such as real-time object tracking [88], quickly

calculating a BLAST-like score for homologous protein alignment [4] and fault detection [89].

The chapter is organized as follows. In the next section the monitoring system architecture is presented together with the dataset collection process; moreover, the generation procedure of anomalous time series data and the whole processing pipeline are described. In Section 4.3, the obtained experimental results are shown and subsequently discussed in Section 4.4. Finally, Section 4.5 reports conclusions and future perspectives.

4.2 Materials and Methods

This section, presents an overview on the main elements, both hardware and software, composing the proposed experimental setup.

Monitoring System Architecture

A LoRaWAN (long range wide area network)–enabled sensor network was established, composed of a LoRaWAN gateway and five sensing devices, specifically designed to monitor temperature ($^{\circ}\text{C}$), relative humidity (%), solar radiation intensity (W/m^2), leaf wetness (scalar value between 0 and 13), and amount of liquid precipitation (mm) with low-cost sensors. The SHT3x sensor by Sensirion was employed for the humidity measurement, performed with a capacitive sensing approach, and the temperature measurement. It is a CMOS (Complementary Metal–Oxide–Semiconductor) sensor embedding on the same chip both the sensing elements and the analog and digital conditioning electronics and it allows to reach measurement accuracy of 0.2°C and 2% for temperature and relative humidity respectively. Concerning the solar intensity, the NORPS–12 light dependent resistor by Advanced Photonix was calibrated in laboratory and used as low-cost alternative to expensive and power-consuming devices as pyranometers, admitting an average error around 15% in the solar intensity estimation [84]. Finally, a commercial resistive sensor based on interdigitated electrodes was mounted to measure the leaf wetness, while the WH–SP–RG tipping bucket rain gauge by Misol was used to estimate with resolution of 0.3 mm the level of liquid precipitation. Each unit was programmed to transmit radio packets at five-minute intervals to the central gateway, which redirected them to a LoRaWAN server, where the sensors readings were identified through the timestamp and the device id number. Field tests started in July 2023 at the farm "La Selva" of Istituto Tecnico Agrario B. Ricasoli of Siena, Italy, and are ongoing to this day. The nodes were deployed in distinct vineyards to establish a dense deployment for the study of the microclimate of a limited agricultural area ($500\text{ m} \times 300\text{ m}$). The meteorological dataset was practically populated with

data from 22 sensors (*i.e.*, two rain buckets and five sensors for temperature and humidity, leaf wetness and solar radiation) belonging to 5 distinct prototypes, with an average of 10 transmissions per hour for about 8 months (from July 2023 to February 2024 included). However, the presented study focuses only on the temperature and relative humidity time series, measured in °C and % respectively. These two sensors were chosen because they are the only ones present on all sensor nodes. A picture of the prototype equipped with the different sensors is reported in Fig. 4.1.

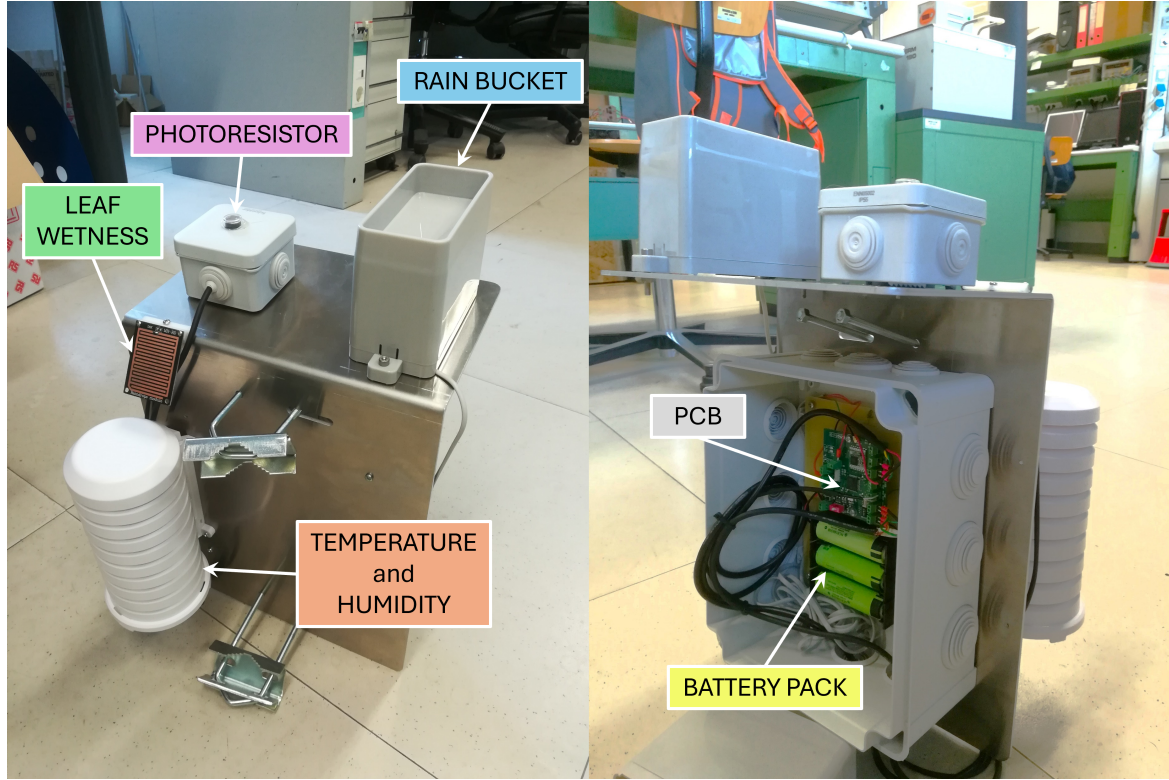


Figure 4.1: Front and back pictures of the designed prototype.

Dataset Construction

In order to guarantee a supervised and balanced dataset to feed the model, the raw time series of temperature and humidity were pre-processed. The primary issue was the presence of unlabeled missing values. Indeed, when the nodes or the gateway fail to transmit packages to the server, data is not recorded, resulting in significant gaps in the temporal dimension of the series which negatively impact the subsequent training phase. To counteract this issue, the series were filled with constant values for each missing time interval exceeding a certain δ : for example for lack of information of n hours, corresponding to $(60 \cdot n) / 5$ missing samples, $(60 \cdot n) / \delta$ time steps with zero value were injected. In the presented case, δ was set to 30 minutes.

Afterwards, the time series were split and the first half of each series was used to build the dataset for training the LSTM predictor. Therefore, each pattern in the training set consists of a time series of length w , having the next value in the series as target. The remaining portion of the time series was used to generate a second dataset for training the Siamese model, which includes two parallel sequences entering the twin branches of the network. The first sequences reproduced the scenario where fault-free data of length w were gathered, followed by the arrival of new data from the sensors. At each step, the LSTM predictor computed the newly trusted value, thereby reconstructing an approximate version of the original time series. Instead, the second half sequences were processed so that, at each step, synthetic faults, based on the original data, were added. In this way, the Siamese model was able to compare, at each time instant, a faulty version of the data, *i.e.*, the trusted data with generated faults, with a reconstruction of the measured data. A pair of sequences of length w , reconstructed and noisy, was labeled as a fault if at least 30% of the second sequence consisted of inserted errors. Such approach enabled the network to detect faults with a certain level of tolerance, triggering errors only in the case of persistent anomalies.

Both temperature and humidity measurements were normalized in the (0,1) range with respect to the minimum and maximum values recorded by the sensors during the measurement campaign.

Synthetic Generation of Faults and Anomalies

Synthetic data refers to artificially generated data useful for augmenting the size or variety of existing datasets. Synthetic data generation can be helpful in training deep learning models (in which a huge number of parameters must be adjusted), increasing their performance and generalization capabilities. This technique has been applied to a variety of problems coming from different fields, ranging from Internet of Things and Smart Grid development [90] [91] to sensor data simulation for healthcare applications [92] [93]. In the present work, the generation of fictitious data simulating the presence of anomalies of the sensors is accomplished to enrich the original dataset, where the number of detected faults was small and not usable to evaluate the performance of the trained model. Generally speaking, the main challenge is to guarantee a realistic representation of the real-world complexity [94] that, in the specific application, is translated in considering the most likely causes of failure of the system and mathematically modelling them with a data-driven approach. In this way, the data generation allowed to create a new dataset of labelled fault data, where there was control over the types and severity of the introduced faults.

Data-driven techniques were adopted to perturb normal data introducing anomalies as outliers and non-predictable trends. The following types of anomalies were

considered:

- hard sensor fault,
- unsteady electrical contact,
- sensor drift,
- degradation of the solar radiation shield protecting the temperature and humidity sensor.

The hard sensor faults were modeled as readings of the sensor stuck to the saturation values, while the false electric contact was generated as consecutive random spikes.

To model the sensor drift, three functions were chosen: sinusoidal, linear and logarithmic. The sinusoidal time-dependent trend with amplitude A_1 and period of 1 day was superimposed to the sensor reading T_i according to $\hat{T}_i = T_i + A_1 \cdot \sin(2\pi f t_i)$; similarly for the linear drift, $\hat{T}_i = T_i + A_2 \cdot t_i$ and the logarithmic drift, $\hat{T}_i = T_i + A_3 \cdot \log(t_i)$. Finally, the degradation of the solar radiation shield should cause an increase of the measured temperature and a decrease of the measured relative humidity depending on the radiation intensity. Just the effect on the temperature measurement was considered and modelled according to the equation $\hat{T}_i = T_i + b \cdot I_i$, where I_i is the measured solar intensity at time instant t_i in W/m^2 and b is a scale factor. Different sets of faults were generated by changing the parameters A_i and b .

Processing pipeline

The implemented pipeline is composed of three phases. In the first one, a set of time series is fed into an LSTM that computes prediction of the next time step value; in the second phase, faults are injected in the original sequences; finally, sequences with and without injected faults are compared by the Siamese network. The pipeline is summarized in Figure 4.2.

LSTM Predictor

The LSTM architecture, which effectively captures both short-term patterns and long-term dependencies, is ideally suited for modeling complex temporal relationships. Therefore, the first step of the pipeline involves training an LSTM model with a hidden linear layer of dimension h_l to compute the function given in Equation 4.1

$$\hat{s}_{t_{w+1}} = \text{LSTM}(S_w) \quad (4.1)$$

where $S_w = (s_{t_0}, s_{t_1}, \dots, s_{t_w})$ is a sequence of length w and $\hat{s}_{t_{w+1}} \in \mathbb{R}$ is the prediction of the $(w + 1)$ -th element of the series.

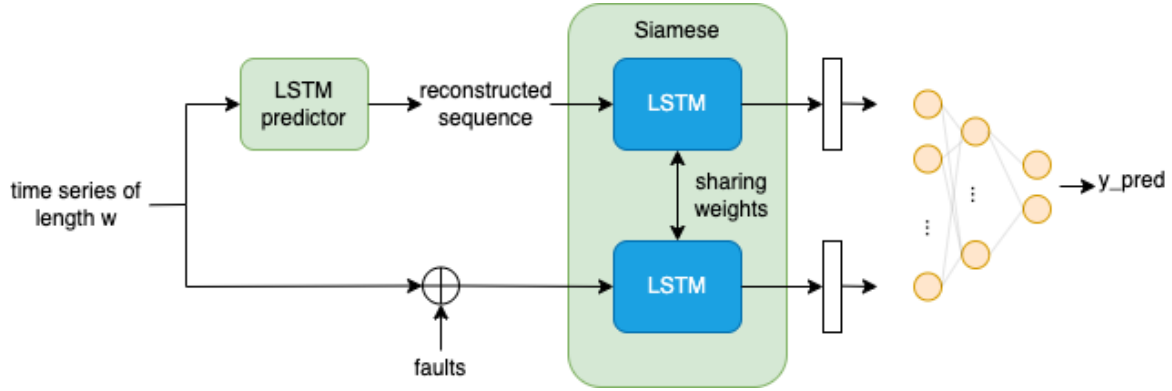


Figure 4.2: Fault detection pipeline.

Siamese Network

The proposed Siamese architecture employs two LSTMs with a hidden state of dimension h_s as the embedding blocks. Each input, consisting of a pair of sequences of length w , is embedded by the Siamese as in Equation 4.2:

$$Z^i, Z^j = f(S_w^i, S_w^j) \quad (4.2)$$

where $Z^i, Z^j \in \mathbb{R}^k$ are *signatures* of dimension k for the input pair (S_w^i, S_w^j) .

Because the merge layer typically reduces the Siamese network's output to a single number representing the distance between input samples, it is omitted in this application. Instead, all the information processed by the Siamese network is passed directly to the decision-making layer. This layer consists of an MLP that performs classification by calculating two output values representing the probabilities of a fault occurring or not, as shown in Equation 4.3:

$$\hat{y} = MLP(Z^i, Z^j). \quad (4.3)$$

$\hat{y} \in \mathbb{R}^2$ is the classification result, but given that a *Softmax* is used as the output activation function, such output can be interpreted as the probability of having a fault or not. The merge layer is omitted because a

4.3 Results

The predictor model consisted of an LSTM with a hidden linear layer of dimension $h_l = 20$. Experiments were performed with the *Adam* optimizer, based on an initial learning rate of 10^{-4} , a batch size of 32 and with early stopping on 100 epochs with patience equal to 10. Root mean square error (RMSE) was used as the loss function, which yielded excellent results for each of the five sensor nodes, both for humidity and temperature. To choose the sequence length w , a grid search was performed.

Table 4.1 displays the obtained *RMSE* from the test sets for both normalized humidity and normalized temperature with $w = 10$, corresponding to one hour-long time windows.

Table 4.1: *RMSE* calculated on the test set for the LSTM predictor.

	Node 1	Node 2	Node 3	Node 4	Node 5
Temperature (10^{-3})	1.25	0.82	1.27	1.37	1.71
Humidity (10^{-2})	9.06	7.91	7.40	8.58	8.38

Concerning the Siamese network, the hidden state dimension h_s of the two twin LSTMs and the embedding space dimension k were set to 100 and 30, respectively. The MLP consisted of an input layer of dimension $2 \cdot k = 60$, a hidden layer of 1024 units and an output layer of 2 units, with a *leakyReLU* activation function, except for the output layer in which the *Softmax* was used. For the Siamese training, the *Adam* optimizer with learning rate of 10^{-3} , batch size of 32 and early stopping criterion with patience 30 were used. The parameter selection was determined through a random search, prioritizing configurations that achieved the best results while maintaining a lower number of dimensions. This approach was adopted to obtain a simpler version of the networks without compromising performance.

The results of the classification step are reported in Tables 4.2 and 4.3 for each node: these outcomes are computed for all the types of synthetic errors (*i.e.*, hard sensor fault, unsteady electrical contact, linear, logarithmic and sinusoidal sensor drift, degradation of the solar radiation shield) and considering the same percentage of sequences for each fault. However, a binary classification is carried out, meaning that the model only recognizes the presence of a fault, without distinguishing among the fault classes. The obtained accuracies are consistent with those found in literature, which span between 80% and 90%, although the strong dependence of these results on the considered type of fault must be always accounted.

Table 4.2: Classification results of the Siamese+MLP architecture for the temperature sensors.

	Accuracy	Precision	Recall
Node 1	91.59%	88.92%	90.00%
Node 2	93.41%	91.81%	91.59%
Node 3	92.30%	84.60%	98.71%
Node 4	89.91%	87.86%	86.70%
Node 5	97.08%	93.71%	99.34%

Table 4.3: Classification results of the Siamese+MLP architecture for the humidity sensors.

	Accuracy	Precision	Recall
Node 1	90.90%	88.49%	88.80%
Node 2	97.27%	93.60%	100.00%
Node 3	96.51%	93.65%	97.96%
Node 4	90.50%	86.77%	89.97%
Node 5	96.26%	93.41%	97.54%

The parameters of synthetic faults were chosen to emulate realistic fault conditions, starting from the analysis of most commonly observed for sensor drift. Once the networks were trained, a robustness test of the models was carried out by slightly changing the fault parameters in the neighborhoods of the training value. Tables 4.4 and 4.5 show robustness results for the accuracy values of temperature and humidity time series. Tests were carried out considering one type of fault per time.

Table 4.4: Robustness results in the temperature classification. The range of the accuracy percentage on the test set, per fault type, is reported. Column ref. reports the values of accuracy for the hyper-parameters used during training.

Temperature				
	Radiation Shield		Sinusoidal Drift	
	ref. (%)	range (%)	ref. (%)	range (%)
Node 1	61.30	(60.82, 61.99)	90.38	(88.91, 96.58)
Node 2	69.70	(65.26, 72.56)	89.89	(67.96, 93.77)
Node 3	68.51	(64.92, 69.55)	80.72	(72.91, 90.85)
Node 4	66.37	(61.42, 68.15)	86.38	(74.99, 92.92)
Node 5	66.49	(62.57, 69.47)	95.28	(93.38, 97.39)
	Linear Drift		Logarithmic Drift	
	ref. (%)	range (%)	ref. (%)	range (%)
Node 1	89.84	(89.20, 94.89)	93.90	(86.64, 97.71)
Node 2	89.39	(67.60, 93.24)	98.36	(68.01, 98.62)
Node 3	80.07	(72.95, 89.27)	96.45	(71.86, 97.57)
Node 4	85.71	(74.91, 91.48)	91.30	(75.04, 94.67)
Node 5	95.07	(93.11, 96.85)	97.74	(92.91, 98.11)

Table 4.5: Robustness results in the humidity classification. The range of the accuracy percentage on the test set, per fault type, is reported. Column ref. reports the values of accuracy for the hyper-parameters used during training.

	Humidity					
	Sinusoidal Drift		Linear Drift		Logarithmic Drift	
	ref. (%)	range (%)	ref. (%)	range (%)	ref. (%)	range (%)
Node 1	91.37	(90.10, 92.43)	91.30	(90.08, 91.64)	95.20	(90.10, 95.61)
Node 2	96.28	(95.61, 96.52)	96.21	(95.65, 96.53)	98.01	(95.63, 98.45)
Node 3	95.34	(91.48, 94.89)	95.40	(91.56, 95.60)	97.79	(92.15, 97.95)
Node 4	88.33	(79.63, 91.94)	88.20	(79.76, 90.84)	96.44	(79.70, 97.51)
Node 5	95.83	(92.79, 96.66)	95.79	(92.90, 96.55)	97.71	(92.83, 98.43)

4.4 Discussion

Given that the obtained *RMSE* are in the 10^{-3} and 10^{-2} order for the temperature and humidity time series respectively, corresponding to mean reconstruction errors of about 0.5°C and 1%, the capability of the LSTM-based predictor model to effectively reconstruct both temperature and humidity signals for each of the five sensor nodes was proved. Concerning the Siamese network, results show that the proposed models perform very well in time series comparison and classification with mean accuracy of 92.86% and 94.28% for all the sensor nodes and with respect to temperature and humidity, respectively — which means a very good capability in detecting faults. Also concerning precision and recall, the models show high performance with 89.38% and 92.27% of mean precision and recall for the temperature, and 91.18% and 94.85% of mean precision and recall for the humidity. As can be observed in Tables 4.2 and 4.3, the higher values of recall indicate that the models tend to reduce the number of false negatives, which translates into a better recognition of true fault signals, given that the positive class is composed of fault time series.

The robustness tests performed on the trained models show that the networks can withstand significant perturbations of the fault parameters modelling the sensor drift. As shown in Table 4.5, the performance percentage ranges are small (about 4% in mean per model) for the humidity, except for node 4, in which higher instability can be noticed. On the other hand, robustness for temperature models is more critical. Table 4.4 shows a higher level of instability for all the nodes. Nevertheless, the percentages at the lower bound of the intervals are relative to much lower parameter values compared to the training ones. This means that, for low error values, the models fail to recognize anomalies, where low error values stands for variations of the original mean magnitude signal less than 1°C , which even an expert eye would not be able to detect as a fault. Concerning the faults due to the degradation of

the solar radiation shield, the models reached in general lower performance. This is probably due to the great variability of the measured radiation profiles depending on the experimented meteorological conditions. Indeed, the radiation measurements have very different values from sunny to cloudy days, leading to a variation more difficult to be detected especially if, during the network training, there were few cloudy days compared to those contained in the test set.

4.5 Conclusions

In this chapter, a novel approach for anomaly detection in meteorological time series was presented. A sensor network was deployed in a farm near Siena, Italy, consisting of a LoRaWAN gateway and five sensor nodes, covering five distinct vineyards. The collected physical quantities were temperature, relative humidity, leaf wetness, solar radiation and amount of liquid precipitation. The focus of this work was on the temperature and humidity time series.

The proposed method relied on an LSTM for data prediction and a Siamese network, topped with an MLP, to binary classify the presence of sequences containing faults. To this aim, a new dataset of synthetic faults was built from real data, considering the most probable causes of failure of the system (*i.e.*, missing packets, hard and soft sensor faults). Results are promising, reaching *RMSE* for the predictor loss in the 10^{-3} and 10^{-2} order for temperature and humidity respectively, which indicates a very good reconstruction of the signals (*i.e.*, mean errors of about 0.5°C and 1% for the temperature and humidity predictions). On the other hand, the accuracy in classification of faults exceeds 92% and 94% for the temperature and humidity respectively, thus proving the capability of the proposed system in the early detection of faulty data and anomalies.

In future research, the proposed approach will be extended to include other measured physical quantities, such as solar radiation intensity, leaf wetness, and liquid precipitation, with fault types tailored to each specific sensor measurement. This approach relies solely on temporal information derived from the time series and learns to identify faults based on the temporal patterns characteristic of each sensor. Additionally, a combined spatio-temporal approach could be explored by integrating time series data with spatial relationships represented in a graph structure. This would involve combining GNNs and LSTMs in a Siamese configuration to leverage both temporal patterns and spatial dependencies across sensors.

Chapter 5

Bioinformatics approach to investigate structural and non-structural proteins in human coronaviruses

In this chapter, an application of Siamese networks to bioinformatics will be discussed. The core idea is to provide methods for comparing primary structures of pairs of proteins that can approximate results of amino acid sequence alignments obtained with traditional algorithms. The work focuses on the biological analysis of human coronavirus proteins, seeking similarities between homologous proteins to explain the cross-activation of the human immune system. The main contributions of this chapter include i) training and testing a Siamese network on homologous protein datasets, and ii) testing the trained network on coronavirus proteins by comparing structural and non-structural proteins of SARS-CoV-2 with those of other human coronaviruses.

5.1 Introduction

The bat-derived Severe Acute Respiratory Syndrome Coronavirus-2 (SARS-CoV-2) causes COVID-19 disease, resulting particularly severe in elderly patients associated with risk factors [95]. It has been identified for the first time in January 2020 in patients suffering from pneumonia [95] and, in the meantime, it has emerged as a pandemic pathogen causing more than 170 million confirmed cases and more than 3.5 million deaths worldwide ¹.

SARS-CoV-2 has a positive viral RNA genome expressing open reading frames which code for structural and non-structural proteins (NSP) [96, 97, 98]. The ORF1a and ORF1ab in the genomic RNA encode for various NSPs at 5' terminal, and few

¹<https://coronavirus.jhu.edu>, assessed 31.05.21

structural proteins at the 3' terminal. The structural ones include: spike (S), nucleocapsid (N), membrane (M), and envelope (E) proteins [99]. The S protein is immunodominant [98, 100, 101] and consists of two subunits: the S1 protein, containing the receptor-binding domain (RBD), and the S2 protein responsible for cell membrane fusion [102, 103]. The N protein is also immunogenic and induces antibodies sooner than S after infection, a characteristic that makes it interesting for diagnostic assays [104]. Immunogenicity of other structural proteins has been less investigated: the relatively small M protein resulted to have immunogenic epitopes but titers generally increase later than 21 days after infection, instead responses to the E protein have been rarely detected so far. Conversely, another study detected anti-M antibodies at 10 days post onset, while detection of anti-N and anti-S increased later [105]. The translated polypeptides of ORF1ab are processed into approximately 1–15 NSPs with specific roles in the life cycle and pathogenicity of the virus [99].

Several studies have reported that some people unexposed to SARS-CoV-2 have preexisting reactivity to SARS-CoV-2 sequences [106]. The immunological mechanisms underlying this preexistent reactivity seem to be linked to previous exposure to widely circulating common cold coronaviruses [106]. Natural and experimental infection studies in humans investigated the cross-reactivity within distinct genera of coronaviruses [98]. The Coronavirinae subfamily includes four distinct genera: the alpha- and beta-CoV family, which usually infects mammals and humans, and the gamma- and delta-CoV family, which generally infects birds. From the 1960s to the present, seven coronaviruses have been documented to generate infection in humans [107]. Among such seven human coronaviruses (HCoVs), two HCoVs (229E, NL63) belong to the alpha-HCoVs, whereas the other five (OC43, SARS, HKU1, MERS and SARS-2) are beta-HCoVs [107]. The natural hosts for most of the HCoVs are bats [108], with the only exception of HCoV-OC43 and HCoV-HKU1, which originated in mice [97]. The bat-originated HCoVs include the four structural proteins mentioned above, while in mice-originated HCoVs, one more structural protein called hemagglutinin-esterase (HE) is observed. For incoming in the host cell and trigger the infection, several human proteins assume the role of viral receptors: angiotensin-converting enzyme 2 (ACE2) for SARS-CoV [109], SARS-CoV-2 [110] and HCoV-NL63 [111]; the aminopeptidase N (APN) and dipeptidyl peptidase 4 (DPP4) are entry receptors for HCoV-229E and MERS-CoV, respectively; the mice-originated beta-CoVs HCoV-OC43 and HCoV-HKU1 need 9-O-acetylated sialic acid (9-OASA) as a viral receptor [97]. The summary of these features is reported in Figure 5.1.

As the circulating common cold HCoVs (*i.e.* OC43 and HKU1) share some homologous sequences with SARS-CoV-2, HCoVs cross-reactive T-cell responses could influence the susceptibility to SARS-CoV-2 infection and the course of COVID-

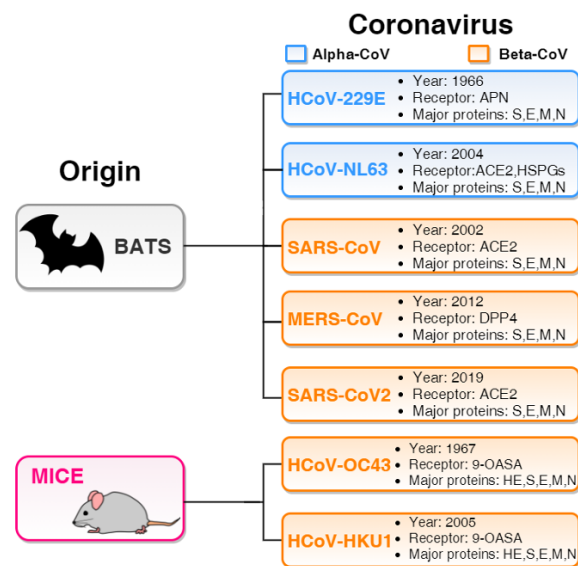


Figure 5.1: Basic features of seven HCoVs. The two blue panels list alpha-HCoVs and the five orange panels list beta-HCoVs. All the boxes include: year of identification, receptor and major proteins. Two beta-HCoVs (HCoV-OC43 and HCoV-HKU1) originate in mice and belong to hemagglutinin-esterase (HE) structural protein along with S, E, M and N. Other five HCoV originated in bats and have four major structural proteins S, E, M and N.

19 [106]. In the present study, the genome similarities among the HCoVs were better investigated. Mateus *et al.* found that the preexisting reactivity against SARS-CoV-2 comes from memory T cells and that cross-reactive T cells can specifically recognize a SARS-CoV-2 epitope as well as the homologous epitope from circulating common cold HCoVs [106]. Specifically, CD4+ T cell responses to S protein, the most important target for vaccines development, were strongly correlated with the magnitude of the anti SARS-CoV-2 IgG and IgA titers [112]. However, as reported in Grifoni *et al.* [113], not only S protein, but also M and N proteins accounted for 11%–27% of the total CD4+ response, with additional significant responses directed against nsp3, nsp4, nsp12, ORF3a and ORF8. Regarding SARS-CoV-2 CD8+ T cells responses, S and M were both recognized and significant reactivity was also noted for nsp6, ORF3a and N, which comprised nearly 50% of the total CD8+ T cell response. These findings underline the importance of determining the impacts of preexisting immune memory in COVID-19 disease severity, particularly to better and quickly investigate the emerging variants that may have an impact on the virus transmissibility and pathogenicity. Computational studies could be effective to examine the most recent Delta and Omicron variants. For instance, in a recent study [114], it has been found that the Omicron S protein has a higher affinity for human angiotensin-converting enzyme 2 (ACE2) than the Delta variant due to a significant number of mutations in the SARS-CoV-2 receptor-binding domain (RBD), indicating higher potential for transmission. In this context, future efforts are needed to investigate

the epidemiological and biological consequences of the variants. Starting with such assumption, this study focus on the investigation of the similarities among the S protein sequences of the SARS-CoV-2 (including wild type, Omicron, and Delta) and the other HCoVs, with particular attention to S proteins, to better understand the cross-reactivity of SARS-CoV-2 and other circulating HCoVs. This is an important starting point for exploring the pattern of immunodominance in COVID-19, for interpreting its pathogenesis, and for the calibration of pandemic control measures.

The main tool used to find similarities within amino-acids sequences and compute a similarity score between them is the Basic Local Alignment Search Tool (BLAST) [115]. BLAST is a heuristic algorithm that splits the query and target sequences in words of fixed length and then compares them finding most significant matches. The representation of amino acids indicates that the primary structure of proteins can be handled as sequences of vectors, making it suitable for processing with tools such as Recurrent Neural Networks (RNNs), so comparison between primary structure can be performed with Siamese networks.

Following this analysis, sequence alignments on structural proteins and NSPs among SARS-CoV-2 and all the other HCoVs listed in Figure 5.1 were performed, and then the BLAST scores were used as targets for a Siamese network model made by LSTMs [31]. Obtained results are promising, showing that neural networks can be a useful resource for scoring alignments, with lower computational demand than traditional methods, such as BLAST itself.

5.2 Materials and Methods

The NCBI SARS-CoV-2 database (MM1) provides many useful COVID-related resources and data, among which the SARS-CoV-2 and other human coronavirus sequences were used in this study. In particular, the built dataset comprises sequences of two alpha-HCoVs, 229E and NL63, and five beta-HCoVs: OC43, HKU1, MERS-CoV, SARS-CoV, and SARS-CoV-2. Alignments were performed with BLAST² (MM2), both on the complete genome of the viruses and more specific protein-coding regions. The first experiment started with the multiple sequence alignment of the whole genomes of the SARS-CoV-2 and the other coronaviruses in the dataset. Then, each coronavirus genome was independently aligned with the genome of SARS-CoV-2, evaluating the percentage sequence identity and homology. In the second experiment, which was also carried out with BLAST, the percentage sequence identity and homology of subsequences coding for specific structural and non-structural proteins were evaluated. The complete list of proteins of SARS-CoV-2 is available

²https://blast.ncbi.nlm.nih.gov/Blast.cgiPAGE_TYPE=BlastSearch&BLAST_SPEC=Betacoronavirus

on the NCBI SARS-CoV-2 database³, and the used BLASTn and BLASTp parameters are shown in Table 5.1. Alignments were carried out for each of the proteins (genes) of SARS-CoV-2: ORF1ab polyprotein (ORF1ab), ORF1a polyprotein (ORF1a), spike glycoprotein (S), ORF3a protein (ORF3a), envelope protein (E), membrane glycoprotein (M), ORF6 protein (ORF6), ORF7a protein (ORF7a), ORF7b protein (ORF7b), ORF8 protein (ORF8), nucleocapsid protein (N), and ORF10 protein (ORF10). Finally, a multiple sequence alignment, among all considered alpha- and beta-HCoV-S, was performed based on CLUSTAL to obtain a percent identity matrix and a similarity phylogram tree, constructed by using the neighbor-joining method.

Table 5.1: BLASTn and BLASTp paramters.

BLASTn		BLASTp	
Word size	28	Word size	3
Gap cost	Linear	Gap cost	Existence: 11 Extension: 1
Match/mismatch score	1,-2	Matrix	BLOSUM62
		Compositional adjustment	Conditional score matrix adjustment

Following traditional experimentation, machine learning (ML) experiments were conducted to replicate the same alignments using an artificial intelligence technique. To efficiently process the sequences, LSTM networks [31] were employed. Specifically, a Siamese LSTM network was constructed to perform pairwise alignments of protein sequences. This paradigm was utilized to estimate the similarity score between two aligned sequences, calculated as the complement of their distance.

Dataset

To train and validate the neural network, a dataset of examples was created based on protein clusters from the NCBI [116] database⁴. Each cluster contains similar proteins with a certain degree of homology, allowing the construction of a relatively homogeneous and balanced dataset of distances between pairs of examples as shown in Figure 5.2. Proteins were sampled from the following NCBI protein clusters: CHL002, CHL005, CHL008, CHL009, CHL204, CHL206, PLN4514, PLN4516, PLN4517, PLN4518, PLN4519, PLN0009, PLN0010, PLN0012, PLN0068, PLN0083, PLN0084, PLN0089, and PLN0091.

Since 58% (35 out of 60) of COVID-19 proteins have amino acid lengths between 100 and 550, the proteins in these clusters were filtered by length, retaining only

³<https://www.ncbi.nlm.nih.gov/Structure/SARS-CoV-2.html>

⁴<https://www.ncbi.nlm.nih.gov/proteinclusters>

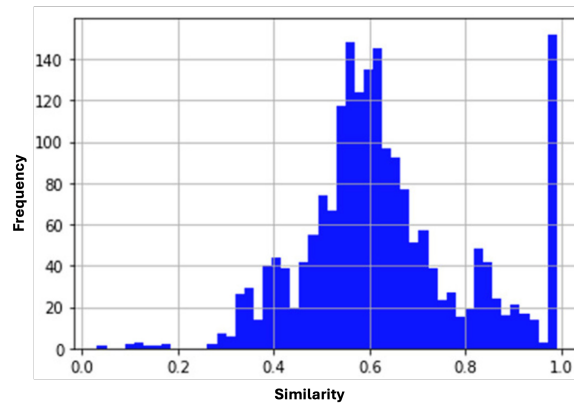


Figure 5.2: Distribution of similarity between pairs of examples in the data set. The number of pairs is proportional to similarity.

those that matched this range. This strategy allowed the construction of a Siamese network optimized for comparing COVID-19 proteins. After filtering, a dataset of 109 proteins with varying lengths was obtained, as shown in Figure 5.3.

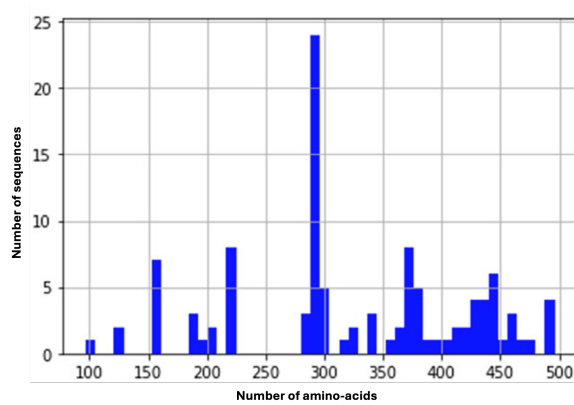


Figure 5.3: Distribution of protein sequence length in the data set.

These 109 proteins were then aligned using BLASTp, with the same parameters as the previous experiments, resulting in a dataset of 1978 pairwise alignments having as targets values from Equation 5.1:

$$y_{ij} = 1 - (c_{ij} * s_{ij}) \quad (5.1)$$

where c_{ij} is the fraction alignment cover, and s_{ij} the identity score of proteins i and j , obtaining by BLAST alignments. The alignment was performed exclusively between proteins belonging to the same cluster.

The dataset of 1978 protein pairs was split using a pseudo-random generator into a training set (comprising 80% of the examples, or 1,582 alignments) and a validation set (comprising 20% of the examples, or 396 alignments). The validation

set was used to optimize hyperparameters and provide an initial assessment of the model's performance. The final evaluation was conducted on a test set containing COVID-19 sequences, corresponding to the sequences from the earlier experiments.

Network structure

The network structure, depicted in Figure 5.4, is similar to that described in the previous chapters. In this case, the embedding blocks of the Siamese network consist of twin LSTMs with hidden linear layer of dimension h , which process the one-hot representation of the input sequences.

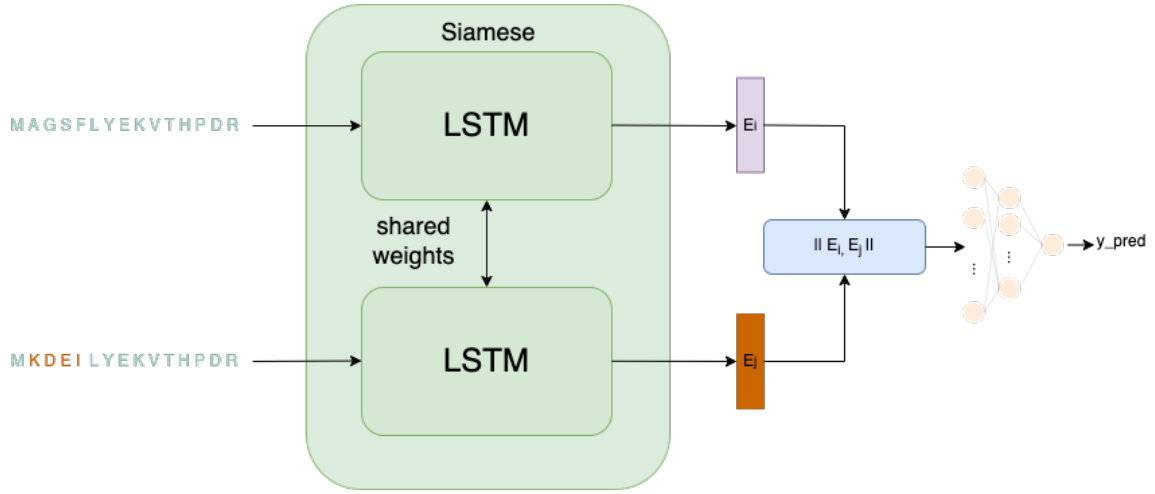


Figure 5.4: Network structure.

The input sequences are padded to a length of l_{pad} . The sequences embeddings E_i and E_j are obtained following 5.2 and 5.3:

$$E_i = LSTM(S_i) = LSTM(a_{i_0}, a_{i_1}, \dots, a_{i_{l_{pad}}}) \quad (5.2)$$

$$E_j = LSTM(S_j) = LSTM(a_{j_0}, a_{j_1}, \dots, a_{j_{l_{pad}}}) \quad (5.3)$$

where S_i and S_j are sequences of l_{pad} one-hot vector. The Euclidean distance is used as the merge layer in the architecture, followed by a MLP as the decision layer, which outputs the similarity between the sequences:

$$\hat{y}_{ij} = MLP(d_{euc}(E_i, E_j)) \quad (5.4)$$

Excluding l_{pad} , which is fixed at 550, representing the maximum amino acid sequence length in the dataset, all other network parameters were optimized through

a grid search. Specifically, the hidden dimension of the LSTMs, and consequently the embedding dimension, h , was set to 32. This value is sufficient to represent sequences as long as those in the dataset, and higher dimensions do not contribute to improved performance. The MLP architecture consists of a single hidden layer with 100 neurons, activated using *ReLU*, followed by batch normalization and a dropout rate of 0.2. The output layer contains a single neuron activated by *sigmoid*, so that the output falls within the range $[0, 1]$ and could represent the BLAST similarity score. *MSE* is employed as the loss function in this architecture.

5.3 Results

The first part of the results focuses on the similarities identified using BLAST, followed by the performance evaluation of the Siamese network. The alignment of SARS-CoV-2 and the other HCoVs was obtained by BLAST. As expected, the higher sequence identity for the SARS-CoV-2 complete genome resulted with SARS-CoV (Identity 82%, Query cover 89%, and Gaps 1%), followed by HKU1 (Identity 72% but very low Query cover 5%) and by NL_63 (Identity 72% but very low Query cover 2%). To obtain a whole picture of the similarities among all the considered viral strains, by performing a multiple sequence alignment (MSA), a percent identity matrix for all the sequences was generated (Figure 5.5), and a similarity phylogram tree, confirming other phylogenetic analyses [112, 117, 118, 119, 120, 121], was created using the neighbor-joining algorithm (Figure 5.6).

Each of the most closely related pairs of sequences is aligned to each other. Next, each new alignment is analyzed to build a sequence profile. Finally, alignment profiles are aligned to each other or to other sequences until a full alignment is built. Again, the higher sequence identity for the SARS-CoV-2 complete genome appeared to be with SARS-CoV (identity 80.04%), followed by MERS (56.93%) and the two beta-HCoVs, HCoV-C43 and HCoV-HKU1 (50.49% and 51.59%, respectively). The alpha-CoVs, HCoV-229E and HCoV-NL_63, had lower identity values with respect to MERS, SARS-CoV, and SARS-CoV-2, despite which they showed 68.31% of identity each other. HCoV-OC43 and HCoV-HKU1 showed an identity equal to 73.38% of each other, followed by HCoV-229E and HCoV-NL63 (about 53%) and the high-CFR HCoVs (MERS, SARS-CoV, and SARS-CoV-2) with values around 50%. Figure 5.6 shows the phylogram of the HCoVs. A phylogram is a scaled phylogenetic tree in which the branch lengths are proportional to the amount of evolutionary divergence. Specifically, the branch length is determined by the number of nucleotide substitutions that have occurred after branching. If the branch lengths are proportional to the number of character changes, it is possible to notice the high similarity between the alpha-CoVs 229E and NL_63 (number of mismatches about 3400) and between the beta-CoVs with low-CFR [112] HKU1 and OC43 (number

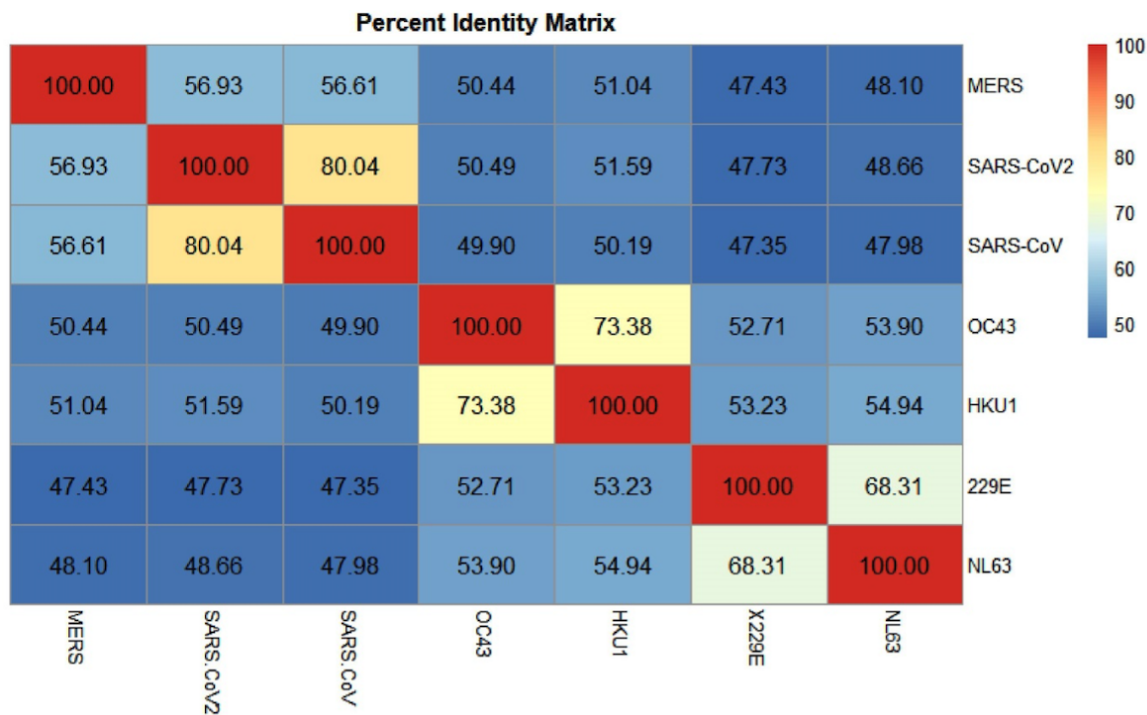


Figure 5.5: Percent Identity Matrix. In panel A, the Percent Identity Matrix is reported, which shows the percentage of identities among the HCoVs.

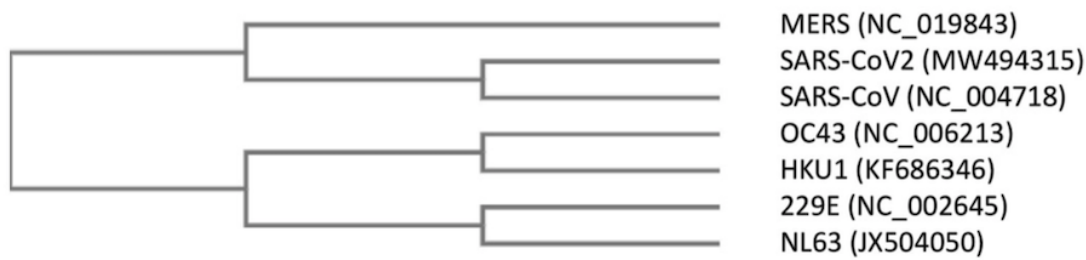


Figure 5.6: Similarity phylogram tree. The branch length is proportional to the number of nucleotide substitutions, *i.e.*, to the number of evolutionary events that took place after the branching point.

of mismatches about 2900). Phylogenetically, SARS-CoV and SARS-CoV-2 share a most recent common ancestor within the subgenus Sarbecovirus (number of mismatches about 4400) and are relatively distant to MERS-CoV (belonging to the subgenus Merbecovirus) in the genus Betacoronavirus [119]. In the case of T cell responses, no obvious pattern of antigen specificity was observed based on SARS-CoV-2 genome organization [113]. Starting from this assumption, this study focus on the exploration of the similarity among different HCoV proteins, involved in CD4+ and CD8+ T cell responses. As reported in the study by Grifoni *et al.* [113], the structural proteins S, M, and N are involved in CD4+ T cell response. Moreover, it is not negligible, and the additional significant response is directed against NSPs, such as nsp3, nsp4, nsp12, ORF3a, and ORF8 [113]. Regarding SARS-CoV-2 CD8+

T cell responses, S and M were both recognized and significant reactivity was also observed for nsp6, ORF3a, and N [113]. These data indicate that optimal vaccine T cell response to SARS-CoV-2 might benefit from additional classes of epitopes, such as the ones derived from the above-listed structural and non-structural proteins. This is the reason why the amino acid sequence alignment between the mentioned SARS-CoV-2 proteins and the other HCoVs (summarized in Table 5.2) were performed with the aim of understanding their similarities in all HCoVs as a starting point for further epitope studies. In addition, preliminary identification of epitopes in SARS-CoV-2 was carried out using the Immune Epitope Database (IEDB)⁵. Experimental data on epitopes tested in the major number of T cell assays have been selected in the context of infectious diseases such as SARS-CoV-2 and then an alignment of the retrieved epitopes with homologous regions in other coronaviruses was performed. The obtained results are listed in Table 5.2.

Thus, to evaluate the potential for cross-reactivity, the structural and non-structural protein sequence homology were compared among SARS-CoV-2 and the other HCoVs: MERS-CoV, SARS-CoV, HCoV-OC43, and HCoV-HKU1 and the alpha-CoVs HCoV-229E and HCoV-HKU1-NL63.

⁵<https://www.iedb.org/>

Table 5.2: Protein sequence alignment results. Amino acid sequence alignment between the mentioned SARS-CoV-2 proteins and the other HCoV-229E. “—” in the column “Epitopes” indicates that no epitope has been reported in IEDB. NSP ORF3a, ORF7a and ORF8 refers to epitopes LLYDANYFL, QLRARSVSPKLFIRQEEVQELY and IRVGARKSAPLIEL respectively. SP Spike (S), Nucleocapside (N) and Membrane (M) refers to QYIKWPWYIW, SPRWYFYLY and KEITVATSRTLSTYYK epitopes respectively.

Protein SARS- CoV-2	HCoV	Accession	% Query cover	E-value	% Identity	% Gaps	Epitopes
Nsp3	229E	AGT21366	66	$5 \cdot 10^{-55}$	28	14	-
	NL63	AFD98833	49	10^{-48}	28	12	-
	OC43	YP_009924321	74	10^{-152}	30	6	-
	HKU1	YP_009944271	75	10^{-142}	29	6	-
	MERS	YP_009047231	95	0	31	9	-
	SARS-CoV	NP_828862	100	0	76	1	-
Nsp4	229E	AGT21366	92	$3 \cdot 10^{-46}$	27	8	-
	NL63	AFD98833	96	$2 \cdot 10^{-53}$	29	10	-
	OC43	YP_009924322	95	$5 \cdot 10^{-142}$	43	2	-
	HKU1	YP_459935	99	$6 \cdot 10^{-137}$	42	2	-
	MERS	YP_009047232	98	$3 \cdot 10^{-142}$	40	1	-
	SARS-CoV	NP_904322	100	0	80	0	-
Nsp6	229E	AGT21366	85	10^{-28}	31	7	-
	NL63	AFD98833	80	$6 \cdot 10^{-28}$	31	3	-
	OC43	YP_009924324	100	$5 \cdot 10^{-44}$	32	4	-
	HKU1	YP_009944274	100	$1 \cdot 10^{-44}$	30	2	-
	MERS	YP_009047218	100	$6 \cdot 10^{-57}$	56	4	-
	SARS-CoV	YP_009944371	98	0	88	0	-
Nsp12	229E	ALJ99946	30	$5 \cdot 10^{-129}$	64	0	-
	NL63	AIW52827	99	0	59	0	-
	OC43	AIW52827	99	0	66	0	-
	HKU1	AXT92527	99	0	67	0	-
	MERS	ATU80202	13	$3 \cdot 10^{-60}$	69	0	-
	SARS-CoV	-	100	0	96	0	-

Continued on next page

Table 5.2: Protein sequence alignment results. Amino acid sequence alignment between the mentioned SARS-CoV-2 proteins and the other HCoV-229E. “—” in the column “Epitopes” indicates that no epitope has been reported in IEDB. NSP ORF3a, ORF7a and ORF8 refers to epitopes LLYDANYFL, QLRARSVSPKLFIRQEEVQELY and IRVGARKSAPLIEL respectively. SP Spike (S), Nucleocapside (N) and Membrane (M) refers to QYIKWPWYIW, SPRWYFYLY and KEITVATSRTLSSYYK epitopes respectively. (Continued)

Protein SARS-CoV-2	HCoV	Accession	% Query cover	E-value	% Identity	% Gaps	Epitopes
ORF3a	229E	-	-	-	-	-	-
	NL63	AFO70498	-	-	-	-	PLTARGRVA
	OC43	-	-	-	-	-	-
	HKU1	AXT92527	-	-	-	-	MKYHPNTVD
	MERS	AKQ21056	-	-	-	-	PLYVPE
	SARS-CoV	AAP41038	100	$8 \cdot 10^{-156}$	72	0	LLYDANYFV
ORF7a	229E	-	-	-	-	-	-
	NL63	-	-	-	-	-	-
	OC43	-	-	-	-	-	-
	HKU1	-	-	-	-	-	-
	MERS	-	-	-	-	-	-
	SARS-CoV	AAP41038	100	$8 \cdot 10^{-156}$	72	0	-
ORF8	229E	-	-	-	-	-	-
	NL63	-	-	-	-	-	-
	OC43	-	-	-	-	-	-
	HKU1	AZS52623	-	-	-	-	VPV-HMP VHPMVMP
	MERS	AVV62534	-	-	-	-	LEQ-DQK LTSLSEL
	SARS-CoV	AAP41043	31	$\cdot 10^{-3}$	30	15	YEG-NSP FHPLADN
S	229E	CAA71056	57	10^{-111}	31	12	TYIK
	NL63	QEG59362	60	$2 \cdot 10^{-104}$	31	13	n.a.
	OC43	AIX10763	89	$4 \cdot 10^{-142}$	38	5	YYVKW PWYVW
	HKU1	AGW27863	70	$9 \cdot 10^{-141}$	35	6	MYVKW PWYVW
	MERS	QBM11748	78	10^{-177}	35	7	YYNKW PWYIW

Continued on next page

Table 5.2: Protein sequence alignment results. Amino acid sequence alignment between the mentioned SARS-CoV-2 proteins and the other HCoVs. “—” in the column “Epitopes” indicates that no epitope has been reported in IEDB. NSP ORF3a, ORF7a and ORF8 refers to epitopes LLYDANYFL, QLRARSVSPKLFIRQEEVQELY and IRVGARKSAPLIEL respectively. SP Spike (S), Nucleocapside (N) and Membrane (M) refers to QYIKWPWYIW, SPRWYFYLY and KEITVATSRTLSSYYK epitopes respectively. (Continued)

Protein SARS- CoV-2	HCoV	Accession	% Query cover	E-value	% Identity	% Gaps	Epitopes
	SARS-CoV	AAR86775	100	0	76	2	QYIKW PWYVW
N	229E	AGW80953	65	$6 \cdot 10^{-24}$	29	12	SPKLHFYYL
	NL63	ABK63972	13	$9 \cdot 10^{-14}$	31	1	PPKVHIFYYP
	OC43	YP_009555245	74	$6 \cdot 10^{-55}$	38	14	LPRWYFYLY
	HKU1	AKQ21062	-	-	-	-	APRWYFYTY
	MERS	QBM11755	91	$5 \cdot 10^{-53}$	48	6	APRWYFYTY
	SARS-CoV	AYV99827	100	0	91	0	SPRWYFYLY
M	229E	QNT54758	99	$2 \cdot 10^{-25}$	31	3	EYMTVAVP STTHYS
	NL63	AFV53151	90	$2 \cdot 10^{36}$	31	4	KYVIVATP STTIVCD
	OC43	AAA45462	95	$4 \cdot 10^{-55}$	41	0	AYMTVAK VTHLCTYK
	HKU1	YP_173241	93	$3 \cdot 10^{-51}$	36	0	VYVTVAKV QVLCTYK
	MERS	AHX00737	90	$2 \cdot 10^{-61}$	43	0	NEVTVAKP NVLIALK
	SARS-CoV	ACZ72273	100	$3 \cdot 10^{-154}$	90	0	KEITVATS RTLSSYYK

The sequence alignments revealed a higher identity percentage of S, N, and M proteins between SARS-CoV-2 and SARS-CoV, followed by MERS for N and M proteins, and HCoV-OC43 for S protein, although the identity is only about 38%. Among the common cold HCoVs, HCoV-OC43 was found to have the most similar sequence for SARS-CoV-2 structural proteins. For ORF3a and ORF8, similarity was found only between SARS-CoV-2 and SARS-CoV, whereas no similarity was retrieved for ORF7a. As far as it concerns nsp3, nsp4, and nsp6, the amino acid sequence alignment showed higher identity for SARS-CoV, followed by MERS and HCoV-OC43. In the case of nsp12, the results for HCoV-HKU1 and HCoV-OC43 were quite comparable with high Query cover (99%) and Identity (67 and 66%, respectively). Similar to what has been observed with the genome alignment, the alpha-CoVs shared

the smallest sequence identity and sequence coverage with SARS-CoV-2 structural and non-structural proteins. This multiple alignment investigation could also be extended to the “variant of concern” (VOC) for SARS-CoV-2 (such as Delta and Omicron), which refers to viral variants with mutations in their spike protein receptor-binding domain (RBD) that dramatically improve binding affinity in the RBD–hACE2 complex, while also causing fast dissemination in human populations [114]. The S



Figure 5.7: Percent identity matrix among the S proteins of HCoVs.

protein mediates the attachment of the virus to host cell–surface receptors and the fusion between virus and cell membranes. It is also the principal target of neutralizing antibodies generated following infection by SARS-CoV-2. Consequently, mutations that affect the antigenicity of the S protein are of particular importance. Thus, the of this work is on the S protein in order to better investigate the similarities/ differences in protein sequences among different HCoVs, including SARS-CoV-2 S and the most recent variants Omicron and Delta. A lower sequence identity between alpha-CoVs and SARS-CoV-2 can be confirmed. Moreover, concerning variants Delta_QWK65230.1 and Omicron_7QO9, the latter shared a smaller S protein sequence identity than the former (99.37% and 93.02%, respectively) (Figure 5.7). This trend holds true also for the higher sequence identity of Delta variants with the other HCoVs in comparison to Omicron, with the only exception constituted by the alpha-CoVs. Such smaller sequence identity between Omicron and SARS-CoV-2 S proteins could lead to significant changes, especially in the RBD region, that might contribute to high binding specificity with hACE2, which in turn may result in a

higher transmission rate and considerable impact on pathogenesis when compared to the Delta variant.

The distance estimation experiments with Siamese LSTMs also showed very promising results. After selecting the best hyperparameters with a grid search over the validation set, the best model (described in Section 3) obtained an error of 0.48% on the validation set itself. Evaluating the same model on the COVID-19 test set confirmed the quality of the model on data it had not been adapted to, showing an error rate of 8%, as reported in Table 5.3.

Table 5.3: Error percentage of the Siamese network on training, validation, and COVID proteins sets.

Set	Error %
Training	0.4
Validation	0.48
Coronavirus	8

The results of the network are as close to the BLAST score as the proteins represent near homologous conditions. The training set proteins contain highly conserved pairs. Conversely, the score of protein pairs which are far homologous is hardly calculated by the network. In Table 5.4 presents a selection of the alignment scores. However, by choosing a larger training set containing more pairs of proteins with a lower degree of homology, it is possible to approximate the more accurate BLAST score.

Table 5.4: Error percentage of the Siamese network on training, validation, and COVID proteins sets.

Protein SARS-CoV-2	HCoV	Blast distance	Siamese distance
Nsp4	OC43	0.6	0.65
	MERS	0.2	0.1
	SARS-CoV	0.14	0.22
ORF3a	SARS-CoV	0.28	0.07
N	MERS	0.56	0.4
	SARS-CoV	0.09	0.09

5.4 Discussion

Grifoni *et al.* [113] suggested that a COVID-19 vaccine consisting only of SARS-CoV-2 S protein would be able to elicit SARS-CoV-2-specific CD4+ T cell responses.

However, the data also indicate that there are many potential CD4+ T cell targets in SARS-CoV-2, and the inclusion of additional SARS-CoV-2 structural antigens such as M and N would better mimic the natural SARS-CoV-2-specific CD4+ T cell response. Concerning SARS-CoV-2 CD8+ T cell responses, the S protein was an important target. SARS-CoV-2 M was strongly recognized, and significant reactivity was observed for other antigens (nsp6, ORF3a, and N), which comprised nearly 50% of the total CD8+ T cell response. Starting from this assumption, the first step was performing a sequence alignment to better understand the complete genome similarity among the HCoVs and then attention was focused on each protein involved in T cell response. Through sequence alignment, the potential cross-reactive immunity to HCoVs of both structural and non-structural proteins was investigated.

Structural proteins

- **S protein:** it is a type I transmembrane N-linked glycosylated protein, with molecular mass of 150–200 kilo Dalton (kDa) consisting of 1,273 amino acids and displays a varying degree of conservation across the Coronaviridae family [122]. As reported in Table 2, the sequence identity of S protein in SARS-CoV-2 to other beta-HCoVs goes from 35% (for HKU1 and MERS) to 76% for SARS-CoV, while it is lower (31%) for alpha-HCoVs, with a higher number of gaps found. Among the common cold HCoVs, characterized from low-CFR, the HCoV-OC43 was found to have the most similar S sequence to SARS-CoV-2, despite their identity is only of 38%. Delta_QWK65230.1 and Omicron_7QO9 shared a sequence identity of, respectively, 99.37 and 93.02% with SARS-CoV-2. These results suggested that the Omicron variant, characterized by multiple mutations in the S protein, could reduce antibody neutralization and vaccine protection from infection.
- **N protein:** it is complexed in the structural organization of the nucleocapsid comprising three highly conserved domains [122]. SARS-CoV-2 is most similar to SARS-CoV, harboring sequence homology of 91% in N protein, followed by MERS-CoV (48%). Among the common cold beta-HCoVs, HCoV-OC43 shares the higher percentage of Query cover and gaps (respectively 74 and 14%). Differently, between the alpha-HCoVs, 229E resulted to have 65% of query cover and 29% of identity, whereas NL63 showed, respectively, 13 and 47%. No alignment is observed for HCoV-HKU1.
- **M protein:** it is an O-linked glycoprotein of around 25–30kDa and is the most abundant among various structural proteins [122]. It facilitates the molecular assembly of virus particles and may be involved during pathogenesis. The M protein identity in SARS-CoV was 90%, followed by MERS and HCoV-OC43.

Regarding alpha-HCoVs, HCoV-229E and HCoV-NL63 share, respectively, 99 and 90% of query cover. Conversely, the sequence identity is lower.

Non Structural proteins

- **Nsp3:** it is the largest protein encoded by the CoV genome, and the average molecular mass is around 200 kD, which is very important in the replication/-transcription complex [123]. SARS-CoV and MERS were found to have high query cover and 76 and 31% of sequence identity, respectively. Among common cold beta-HCoVs, OC43 and HKU1 have a query cover-up than 70%, but only 30 and 29% of sequence identity. The alpha-HCovs 229E and NL63 showed lower values of query cover and identity.
- **Nsp4:** it has an essential role in replication and in the assembly of the replicative structures [99]. As reported in Table 5.2, nsp4 was found to have the highest identity values in SARS-CoV (80%) and in HCoV-OC43 (43%) among the beta-HCoVs with a low percentage of gaps too. In alpha-HCoVs, the sequence identity of NL63 is 2% and in 229E is 27%, with a query cover value more than 90%.
- **Nsp6:** it generates autophagosomes from the endoplasmic reticulum and is involved in autophagy [99]. Nsp6 showed high query cover values. The sequence identity was found to be 88% in SARS-CoV, followed by MERS, OC43, and HKU1 (56%, 32%, and 30%). The query cover value is 100% in OC43, HKU1, and MERS and 98% in SARS-CoV.
- **Nsp12:** it is the RNA-dependent RNA polymerase, the central component of coronaviral replication and transcription machinery [99]. The nsp12 query cover is around 99% in the beta-HCoVs, with the only exception of MERS. As far as it concerns the identity, in SARS-CoV, it is 96%, while in the other beta-HCoVs, it goes from 66% (HCoV-OC43) to 69% (HCoV-MERS). In the alpha-HCoVs, the sequence identity is lower.
- **ORF3a, ORF7a, and ORF8:** ORF3a plays a fundamental role in virus replication and release, ORF7a is a viral structural protein, and ORF8 is an accessory protein correlated with the ability of the virus to spread [124]. These ORFs are specific for SARS-CoV and do not show significant homology to proteins of other HCoVs.

The proposed systematic MSA analysis of all structural and non-structural proteins allowed a better investigation of sequence similarities, and it is a starting point for

a deeper epitope analysis. The percentage identity of all the proteins is summarized above, in particular the sequence alignments revealed a higher identity percentage of S, N, and M proteins between SARS-CoV-2 and SARS-CoV, followed by MERS, HCoV-OC43, and HCoV-HKU1. In nsp3, nsp4, and nsp6, amino acid sequence alignment showed a higher identity for SARS-CoV, followed by MERS and HCoV-OC43, whereas, concerning ORF3a and ORF8, the similarity was found only between SARS-CoV-2 and SARS-CoV, and no identity was found for ORF7a. In several studies, it has already been shown that the vast majority of N protein epitopes are characterized by low rates of evolutions. For the S protein, epitopes are preferentially located in regions that are predicted to be ordered and well-conserved and may help develop long-lasting, broad-spectrum SARS-CoV-2 vaccines [125, 126]. Owing to the proposed alignment experiments carried out with Siamese LSTM neural networks, it was possible to show that such an alternative technique provides very low error rates and could be a starting point for further epitope investigation. This approach also became effective for the potential emerging SARS-CoV-2 variant investigation. Moreover, it could be a model applicable not only for SARS-CoV-2 but also for other biological queries.

5.5 Conclusion

A preexisting immune memory due to exposure to common cold HCoVs seems to have a significant impact on the COVID-19 disease severity, thus suggesting the fundamental role on SARS-CoV-2 cross-reactivity of the protein sequence similarities with different circulating coronaviruses. As expected, the sequence alignments revealed a higher identity percentage of S, N, and M proteins between SARS-CoV-2 and SARS-CoV, followed by MERS. Among the common cold HCoVs, HCoV-OC43 was found to have the most similar structural proteins to SARS-CoV-2 involved in T cell responses, followed by HCoV-HKU1. Regarding NSPs, nsp3, nsp4, and nsp6, amino acid sequence alignment showed a higher identity for SARS-CoV, followed by MERS and HCoV-OC43, with the only exception of nsp12, which was found to have a higher query cover and identity in the cases of HCoV-HKU1 and HCoV-OC43. Concerning ORF3a and ORF8, the similarity was found only between SARS-CoV-2 and SARS-CoV, whereas no identity was found for ORF7a. Similar to what was observed with genome alignments, the alpha-CoVs shared smaller sequence identity and sequence coverage with SARS-CoV-2 in structural and non-structural proteins. In conclusion, the results of this work are as follows:

1. The similarity of amino acid sequences across various HCoVs was systematically examined as a foundation for further studies on cross-reactive T-cell recognition between circulating common cold HCoVs and SARS-CoV-2. This

analysis included MSA of both structural and non-structural proteins. Given the involvement of structural proteins and NSPs in CD4+ and CD8+ T-cell responses, a detailed assessment of sequence identity (%) could be valuable for identifying cross-reactive epitopes. Consequently, a deeper investigation into cross-reactive T-cell immunity to SARS-CoV-2 holds significant potential implications for differential COVID-19 clinical outcomes and could impact the effectiveness of COVID-19 vaccines.

2. Finally, alignment experiments carried out with the Siamese LSTM neural networks showed that artificial intelligence techniques show very low error rates with respect to the ground truth provided by BLAST. This indicates that such models are competitive with the traditional methods for pair alignment. All the considerations made in the literature concerning the BLAST alignment scores hold true for presented score. It is to be noted, though, that such a model can be repurposed, according to the needs of the user, by training it on supervisions from any available alignment method.

Chapter 6

Other works

This chapter presents some research, conducted during the doctoral period, not included in the main stream of the thesis, but related to its contents. The contributions presented here have in common the use of deep learning techniques for the processing of biomedical data. Indeed, Section 6.1 outlines a research which is part of a preliminary study aimed at the early detection of Alzheimer's disease symptoms, where deep learning techniques are utilized to extract information from NMR spectra of metabolites from SH-SY5Y diseased cells. The addressed task involves developing a new pipeline for analyzing NMR spectra where, even with a limited number of data, informative features can be extracted for classifying healthy vs diseased cells. Instead, Section 6.2 focuses on the application of deep networks in computer vision, for the segmentation of oocyte images, to support operators in their selection and to improve the success probability of Medical Assisted Procreation (MAP).

6.1 A Deep Learning Approach to Analyze NMR Spectra of SH-SY5Y Cells for Alzheimer's Disease Diagnosis

Neurodegenerative Diseases (NDs) are severe neurological disorders affecting millions of people worldwide with no effective treatments so far. NDs are characterized by a complex network of pathological events leading to oxidative stress, mitochondrial dysfunction, metal ion dyshomeostasis, protein misfolding, and neuroinflammation [127]. All these factors are strongly interdependent and, although playing a key role in neurodegeneration etiology, their implications as causes or consequences have not yet been clarified. Alzheimer's Disease (AD) accounts for 60–80% of dementia cases which are treated with few drugs available to relieve the early symptoms. The lack of treatment has led the research to evaluate as many molecules as possible, in order to discover active ingredients [128, 129]. These preliminary screenings were carried out using in vitro neuronal models. The most useful model for such experiments is the SH-SY5Y cell line, derived from human neuroblastoma

[130, 131]. Previous studies have shown a comparison between different types of neuronal models demonstrating the many advantages offered by SH-SY5Y cells: they are human-derived, do not require animal testing, they are very proliferative, and their cost of cultivation is low [132].

AD is characterized by a slow progression, making early diagnosis difficult. Moreover, the small memory problems that characterize the early stages of AD are often misidentified as a natural consequence of aging. For these reasons, in the last years, the scientific community has put a lot of effort in the discovery of reliable biomarkers for early AD diagnosis. In this scenario, NDs need to be fully investigated combining multidisciplinary approaches for the characterization of the corresponding metabolic features. NMR metabolomics allows to identify the entire metabolite spectrum in many biological fluids and cellular extract/medium, obtaining a metabolic fingerprint (metabolic profile) that characterizes the healthy or diseased state [133]. NMR spectroscopy, compared to other techniques, has the advantage of being rapid and not requiring a particular treatment of the samples [134].

Data produced by spectroscopic investigations are diversified and are usually combined with multivariate statistical analysis [135]. Indeed, one of the main problems of NMR-related tasks is the huge number of features per spectrum to be analyzed, while the available spectra often amount to small numbers. As a consequence, both dimensionality reduction and data augmentation are normally needed to automatically classify spectral data. The main technique used for the former task is Principal Component Analysis (PCA), that allows a linear dimensionality reduction. While widely used, PCA may not be appropriate for complex and highly non-linear data like NMR spectra. Recent applications of Deep Learning (DL) methods to NMR data opened a new reliable path to guarantee an information-conservative feature selection [136, 137, 138, 139].

Application of DL models to the early detection and automated classification of AD has recently gained considerable attention, as rapid progress in neuroimaging techniques has generated large-scale multimodal data [140, 141]. Based on brain NMR images, for instance, the gray to white matter ratio can be evaluated [142], while some forms of degeneration can be revealed, to highlight the premature brain aging typical of AD. In fact, in the brain of AD patients, the deposition of the amyloid protein and the death of neurons, located in the gray matter, are observed. However, radiological examinations also show damage to the white matter, that part of the brain which is instead mainly made up of myelin, a substance that surrounds the neurons, facilitating their communication. In addition to deep learning techniques applied to neuroimaging, models have been implemented which are capable of integrating flexible combinations of routinely collected clinical information, including demographics, medical history, neuropsychological testing, and functional assessments [143]. Finally, attention modules, able to produce saliency maps, can

give rise to interpretable methods for cerebral images, showing that the detected disease-specific patterns track distinct patterns of degenerative changes throughout the brain and correspond closely with the presence of neuropathological lesions on autopsy. For the non-invasive prediction of Alzheimer's disease, DL techniques have also been applied to retinal images [144]. In fact, a reduction in the density of the capillary network around the center of the macula has been observed in AD patients. To early diagnose AD, retinal images were captured in different modalities (optical coherence tomography, optical coherence tomography-angiography, ultra-widefield retinal photography, and retinal autofluorescence) and integrated with patient data [145].

The present study addresses the problem of early diagnosis of Alzheimer's disease from a new perspective. A DL approach, alternative to PCA and based on Convolutional AutoEncoders (CAEs), is proposed. A targeted procedure for NMR spectral data compression is also presented, in order to deal with the scarcity of data. Apart from the novelty inherent in the application of DL methodologies to this problem, the results of the downstream classification phase, needed for discriminating healthy and diseased cells, are very promising while, as far as is known in the current literature, this is the first time that such an automatic protocol has been used for the analysis of NMR spectra of SH-SY5Y cells.

In summary, the main contributions of this work consist in:

1. the definition of an entire pipeline to process the NMR spectral data of the SH-SY5Y cell line (see Figure 6.1) which includes
 - a *biologically-informed* preprocessing,
 - a DL approach based on CAEs, alternative to PCA, to automatically compress NMR spectra, represented as vectors of intensity values;
2. a set of comparative experiments — showing the superiority of the proposed DL compression method w.r.t. PCA — to classify healthy/diseased cells for early diagnosis of AD.

Based on the proposed pipeline, a high dimensionality reduction in the spectral signal (up to more than 300 times) is obtained, maintaining informative features (guaranteed by a reconstruction error always smaller than 5%). Indeed, after data compression, the classification task has been carried out by a SVM with a RBF kernel. A grid search approach was used in order to find the best kernel parameters. For performance evaluations, due to the very small number of available examples (94), a leave-one-out technique was implemented. Each experiment consisted of 94 training/test runs, each of which reserved a single pattern for testing, while the other 93 were used in the training phase. Although the experimental results, based on the small dataset available, are preliminary, they are nonetheless promising —

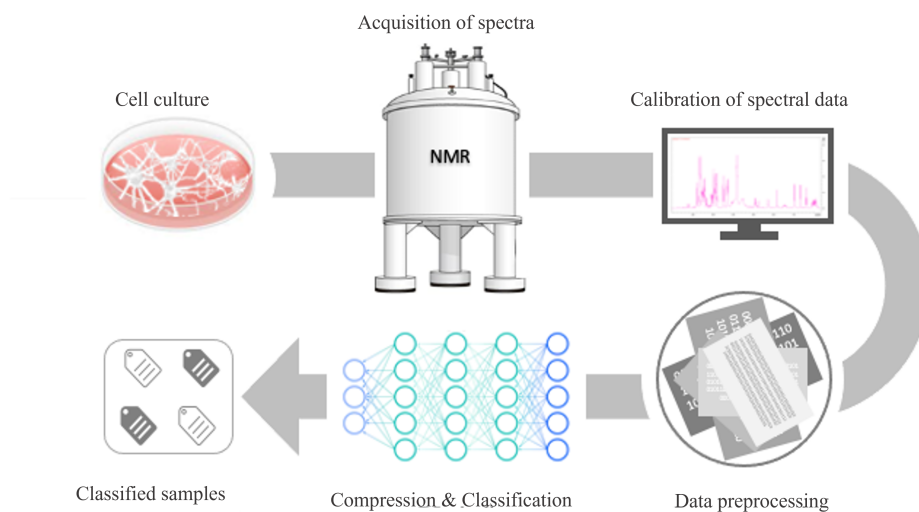


Figure 6.1: Processing workflow: From wet lab experiments to healthy/diseased cell classification.

reaching a classification accuracy over 76%, a precision of 74% and a recall (capacity in recognizing healthy cells) over 94% — and show the suitability of compressed data to be used in a downstream classification task to distinguish between healthy and diseased neuronal cells, in early diagnosis of neurodegenerative pathologies. The significantly reduced dimension of embedded spectra, obtained with almost no information loss, would allow to drastically reduce both the computational cost and memory storage, opening the door to the intensive use of automatic processing techniques for NMR data (not only related to the metabolic cell profile).

6.2 A Deep Learning approach for oocytes segmentation and analysis

Medical Assisted Procreation (MAP) has seen an increasing demand over the years: in Italy, MAP demand registered a 10% increase between 2013 and 2019 [146]; an increase in terms of thousands of MAP cycles was registered in Europe (75k to 900k) and in Asia (60k to 550k) between 1993 and 2017 [147].

This work focuses on second level in-vitro fertilization (IVF), a set of pharmacological therapies, surgical and laboratory procedures which consists in placing spermatozoa and oocytes on a plate with an ideal culture medium. If fertilization is successful, the embryo is obtained, which can be transferred back to the uterus for gestation. There are two different IVF techniques, FIVET and ICSI. In FIVET (Fertilization In Vitro with Embryo Transfer), once the oocytes have been taken, they are placed on a plate with the spermatozoa, to facilitate fertilization; in ICSI (IntraCytoplasmic Sperm Injection), which is used in case of spermatozoa with low motility,

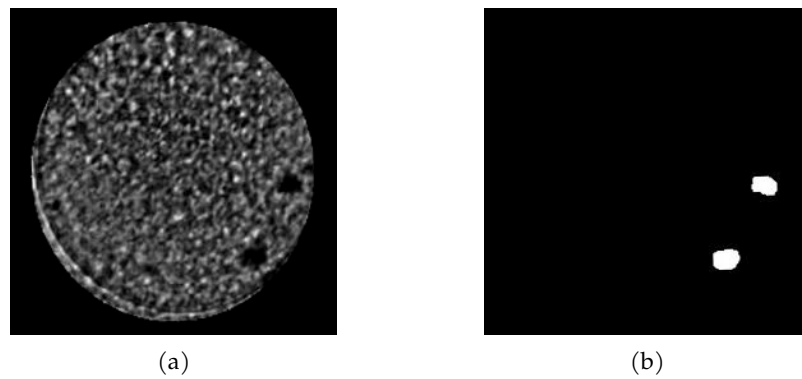


Figure 6.2: Sample image from the training set in (a); the corresponding mask in (b)

spermatozoa are directly injected into the oocyte. It is worth noting that the success of both FIVET and ICSI depends, beyond other genetic factors, on the capability of the human operator to perform a good selection of embryos.

The present work aims at showing how a semantic segmentation approach based on deep learning can be used to support the operators, to improve the success probability of the ICSI procedure. Actually, the quality of an oocyte is low if it shows abnormalities, such as smooth endoplasmic reticulum (SER), vacuoles, granular cytoplasm and inclusions. Among the others, the detection of inclusions is an important step in determining the quality of the oocytes.

With the aim of both inferring the presence of inclusions and locating them, a semantic segmentation approach was used. The main reasons to use deep learning as a support tool for human operators can be summarized as follows.

- **Explainability:** image semantic segmentation can be described as a pixel level image classification, which provides the capability to localize the object of interest into the image. This makes the human operator interpretation of the system output easier.
- **Consistency:** while the interpretation by the human expert can be significantly variable, experiments performed by an AI system can be repeated until a reliable result is obtained, so that operator errors can be prevented.

The dataset was initially composed by 26, 40x inverted images captured by using a Leica Micromanipulator with Nikon camera, provided with respective masks highlighting inclusions (Figure 6.2). Successfully training a deep segmentation network typically requires a large amount of supervised data. However, data scarcity is a common challenge in deep learning for biomedical applications due to privacy concerns and the inherent difficulty of obtaining annotated data. Therefore, data augmentation techniques were used, based on deep learning-driven image generation, to mitigate the lack of a large annotated dataset. In particular, following the procedure employed in [148] and [149], an image-to-image translation approach

was used, to transform semantic label maps in fake oocyte images, that can be used to enlarge the small training set. The label maps were created by randomly placing from one to five circular spots with variable diameter (from 2 to 7 pixels) within the images, which closely approximates the true distribution of inclusions in oocytes, according to expert opinion. Finally, a Segmentation Multiscale Attention Network (SMANet) [150] was applied to perform feature extraction and produce a pixel-wise prediction. The generated images showed to be effective for network training, guaranteeing an accuracy of 0.9988% and a MeanIoU (Intersection over Union) of 0.722%.

Actually, it is worth noting that synthetic images can be effectively used for data augmentation, since better results were achieved by training on synthetic images rather than using real data, demonstrating the high quality of the generated patterns. Moreover, the results were further improved by fine-tuning the model — pre-trained on synthetic data — on real images.

Chapter 7

Conclusions and Future Perspectives

This thesis focuses on the application of Siamese networks across various fields within computer science. After providing a detailed overview of Siamese networks and other relevant neural architectures in Chapter 2, the emphasis shifts to exploring the feasibility of solving tasks without relying on traditional Siamese-specific loss functions, such as contrastive or triplet loss. Such loss functions require training the network based on a dataset with similarity or dissimilarity targets, which is not always feasible. Training with other types of loss functions has the advantage of enabling training even in the absence of similarity targets.

The examples provided throughout the work demonstrate the effectiveness of Siamese networks in tasks where the comparison between input data is crucial. The networks successfully create an *ad hoc* embedding space where input data are projected and further analyzed by subsequent layers, referred to in this thesis as the decision layer. This approach highlights the versatility and power of Siamese networks in generating useful embeddings for comparison-based tasks.

In Chapter 3, a soft estimation of the GED, called NeuraGED was introduced, demonstrating the effectiveness of leveraging GNN-based Siamese models to deliver a reliable, computationally efficient estimation of GED, with very low error rates. This approach can be applied to various tasks that require evaluating topological distances, or a degree of soft isomorphism, between graph structures. For instance, it could be used to compute the distance between two molecular graphs, even when dealing with large molecules. This is potentially very useful when building graph generators, as the proposed method could be used for calculating a reliable, fast, informative, and permutation-invariant reconstruction loss. Such loss could substantially improve the training procedure of many graph generative models, which typically rely on less informative and non-permutation-invariant reconstruction loss functions.

The NeuraGED architecture does not use a merge layer for comparing graph embeddings. Instead, an MLP is responsible for performing the computation of the

GED. The network is trained using the *MSE* loss function, which allows the model to closely approximate the target GED values. This suggests either that the Siamese network has successfully constructed a high-quality embedding space, or that the MLP is effectively compensating for any imperfections in the embeddings. In this case, the MLP functions as both a distance metric and a decision-making layer, simultaneously fulfilling the roles typically handled by a merge layer and a decision layer.

A similar task is presented in Chapter 5, focusing on predicting distances, or similarity degrees, between protein sequences. In this case, an LSTM-based Siamese network is employed to estimate the genetic distance between primary protein structures, using a BLAST-like score as the ground truth. This approach demonstrates that such models can be competitive with traditional pair alignment methods like BLAST. However, the model can be repurposed by retraining on supervisions from any available alignment method, providing flexibility depending on the user's needs. All the standard considerations for BLAST alignment scores remain valid for this model's predictions.

The network architecture used for estimating the protein similarity is similar to the one discussed in Chapter 3, with LSTMs replacing GNNs. However, in contrast to the previous model, this design includes a merge layer that calculates the Euclidean distance between sequence embeddings. The combination of this merge layer with the MLP in the decision layer demonstrates that, even in this case, the Siamese network effectively projects inputs into a well-structured embedding space that maintains the original target distances. These distances are further refined by the MLP in the decision layer, ensuring that the network preserves and enhances the relationship between the input sequences and their target distances.

Chapter 4 focuses on a fault detection task, framed as a classification problem. LSTM-based Siamese networks are employed to compare well-reconstructed signals from temperature and humidity sensors, experimentally collected from a farm near Siena, with both faulty and non-faulty versions of the same original signals. The reconstruction of signals is carried out using pre-trained LSTM models. Notably, this architecture does not include a merge layer. Instead, the decision-making layer is an MLP, with the final layer consisting of two units activated by a *Softmax* function, producing the output as a probability of the presence of a fault. The training loss used is the categorical *cross-entropy*, enabling the network to effectively structure the embedding space based on the class labels.

The use of non-Siamese loss functions, in combination with a well-designed decision layer, allows Siamese networks to construct meaningful embedding spaces. However, further research is necessary to fully understand the potential and limitations of Siamese models and to design new architectures. For example, future work could involve integrating both Siamese and non-Siamese losses, such as combining

contrastive loss with *cross-entropy* for classification tasks. This approach would involve modifying the network by adding a merge layer and creating branches where embeddings are analyzed by the decision layer. This hybrid method would allow the embedding space to be modeled both by the relationships between classes and by the principle of similarity, where similar inputs are placed closer in the embedding space.

Furthermore, a promising future direction for Siamese networks lies in their application to generative tasks. In generative tasks, particularly in molecular generation, it is crucial to have an embedding space that effectively captures similarities and dissimilarities between inputs. By mapping similar examples close together and dissimilar ones further away, the network facilitates the generation of new objects, such as molecules, that resemble a specific target or fall within a certain desired distribution. Creating an *ad hoc* embedding space grounded in similarity principles enables a more accurate definition of target similarity. This precision enhances the generation of new molecules or objects that fit the required characteristics. Moreover, this method can be particularly valuable when generating new structures directly in the latent space, allowing for more accurate and efficient creation processes that align with the defined target features. This future exploration could unlock significant advancements in fields like drug discovery.

Another future research direction involves the design of novel hierarchical Siamese networks based on GNNs. Comparing large graphs poses challenges even for Siamese architectures, so enhancing GED estimation could benefit from integrating graph pooling techniques, such as DiffPool or MincutPool, within *stacked* Siamese layers. This approach would allow better performances for the applications on large graphs incorporating relational information from locally pooled groups of nodes, thereby enriching the embedding with hierarchical information.

Appendix A

Publications

Journal papers

1. Bacconi Sara, **Costanti Filippo**, Bianchini Monica, Pancino Niccolò, Bongini Pietro, “NeuraGED: A GNN estimation for Graph–Edit Distance”, *Procedia Computer Science*, Elsevier, pages: 4186–4193, 2024.
Candidate’s contributions: idea of NeuraGED and design of the network, writing of the paper.
2. **Costanti Filippo**, Kola Arian, Scarselli Franco, Valensin Daniela, and Bianchini Monica, “A deep learning approach to analyze NMR spectra of SH-SY5Y cells for Alzheimer’s disease diagnosis”, *Mathematics*, pages: 2664, 2023.
Candidate’s contributions: ideation and design of the pipeline, design of the network, in silico experiment, writing of the paper.
3. Cicaloni Vittoria, **Costanti Filippo**, Pasqui Arianna, Bianchini Monica, Niccolai Neri, and Bongini Pietro, “A bioinformatics approach to investigate structural and non-structural proteins in human coronaviruses”, *Frontiers in Genetics*, pages: 1303. 2022.
Candidate’s contributions: design of the network, BLAST and Siamese experiments, writing of the paper.

Peer reviewed conference papers

1. **Costanti Filippo**, Cappelli Irene, Fort Ada, Ceroni Giuseppe Elia, Bianchini Monica, “LSTM–based Siamese Networks for Fault Detection in Meteorological Time Series Data”, *IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*, 2024.
Candidate’s contributions: ideation and design of the pipeline, design of the network, experiment, writing of the paper.

2. Andreini Paolo, Pancino Niccolò, **Costanti Filippo**, Eusepi Gabriele, Corradini Barbara Toniella, "A Deep Learning approach for oocytes segmentation and analysis", *ESANN 2022 proceedings European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages:1, 2022.

Candidate's contributions: experiments, writing of the paper.

Ringraziamenti

Voglio scrivere questi ringraziamenti in italiano, lingua solare, poetica e cardiaca per eccellenza, sia perché è la mia lingua natale e riesco ad esprimermi con tutti i colori di cui ho bisogno, sia perché di per se stessa riesce meglio a trasmettere il calore della gratitudine.

Il primo ringraziamento è per la Prof.ssa Monica Bianchini, mio supervisore. Oltre all'aiuto che mi ha dato, lasciandomi sempre libero, le sono grato perché ha sempre creduto in me sin dal primo esame della triennale. Non da meno è la gratitudine che voglio esprimere al Prof. Franco Scarselli, uomo di viva intelligenza e di buon cuore, i confronti con il quale mi hanno insegnato come ragionare nell'ambito del deep learning. Ringrazio i miei colleghi per i fruttuosi confronti e collaborazioni.

Ringrazio anche il Prof. Filippo Geraci e la Prof.ssa Linda Brodo per aver gentilmente dedicato del tempo nella revisione della presente tesi e la regione Toscana per aver finanziato la borsa di studio Pegaso.

Infine, ma non per importanza e presenza, ringrazio la mia famiglia ed i miei amici. Potrei scrivere intere pagine per ciascuno di loro elencando i motivi per cui sono grato, ma credo che non bastino per esprimere tale sentimento. Dedico soltanto ulteriori due parole per ringraziare Nicoletta Lonzi e Nicola Costanti, questo traguardo è in parte anche il loro. Voglio ringraziare anche Daniela Valensin, sempre pronta a darmi una mano e Arian Kola, compagno di viaggio nella ricerca e amico sincero. Un grazie particolare lo dedico ad Alessandro Benassai, perché l'idea di proseguire il mio percorso accademico è sgorgata nella mia mente dopo una stimolante conversazione con lui.

Bibliography

- [1] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a " siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.
- [2] Sara Bacconi, Filippo Costanti, Monica Bianchini, Niccolò Pancino, and Pietro Bongini. Neuraged: A gnn estimation for graph–edit distance. *Knowledge-Based Intelligent Information and Engineering Systems: 28th International Conference, KES 2003, Elsevier,, 2024.*
- [3] Filippo Costanti, Irene Cappelli, Ada Fort, Giuseppe Elia Ceroni, and Monica Bianchini. Lstm–based siamese networks for fault detection in meteorological time series data. *IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*, 2024.
- [4] Vittoria Cicaloni, Filippo Costanti, Arianna Pasqui, Monica Bianchini, Neri Niccolai, and Pietro Bongini. A bioinformatics approach to investigate structural and non-structural proteins in human coronaviruses. *Frontiers in Genetics*, 13:891418, 2022.
- [5] Abhilash Nandy, Sushovan Haldar, Subhashis Banerjee, and Sushmita Mitra. A survey on applications of siamese neural networks in computer vision. In *2020 International Conference for Emerging Technology (INCET)*, pages 1–5. IEEE, 2020.
- [6] Alberto Rossi, Matin Hosseinzadeh, Monica Bianchini, Franco Scarselli, and Henkjan Huisman. Multi-modal siamese network for diagnostically similar lesion retrieval in prostate mri. *IEEE Transactions on Medical Imaging*, 40(3):986–995, 2021.
- [7] Weihong Ren, Xinchao Wang, Jiandong Tian, Yandong Tang, and Antoni B Chan. Tracking-by-counting: Using network flows on crowd density maps for tracking multiple targets. *IEEE Transactions on Image Processing*, 30:1439–1452, 2020.

- [8] Xiufen Ye, Yue Sun, and Chuanlong Li. Fcn and siamese network for small target tracking in forward-looking sonar images. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–6. IEEE, 2018.
- [9] Yi Zhou, Tian Wang, Ronghua Hu, Hang Su, Yi Liu, Xiaoming Liu, Jidong Suo, and Hichem Snoussi. Multiple kernelized correlation filters (mkcf) for extended object tracking using x-band marine radar data. *IEEE Transactions on Signal Processing*, 67(14):3676–3688, 2019.
- [10] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6668–6677, 2020.
- [11] Ao Li, Lei Luo, and Shu Tang. Real-time tracking of vehicles with siamese network and backward prediction. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.
- [12] Hao Zhu, Licheng Jiao, Wenping Ma, Fang Liu, and Wei Zhao. A novel neural network for remote sensing image matching. *IEEE transactions on neural networks and learning systems*, 30(9):2853–2865, 2019.
- [13] Zhao wei, Lian xiaofeng, Liu zaiwen, and Mao shitao. Panoramic image matching based on graph cuts algorithm. In *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pages 1874–1876. IEEE, 2012.
- [14] Zhezhe Zhou, Meng Zhao, Ying Han, and Shengyong Chen. A pyramid image matching algorithm for chang’e-2 ccd images of the moon. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pages 1741–1746. IEEE, 2018.
- [15] Hao-Chun Yang and Chi-Chun Lee. A siamese content-attentive graph convolutional network for personality recognition using physiology. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4362–4366. IEEE, 2020.
- [16] Yun Gu, Khushi Vyas, Mali Shen, Jie Yang, and Guang-Zhong Yang. Deep graph-based multimodal feature embedding for endomicroscopy image retrieval. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):481–492, 2020.
- [17] Xiang Ling, Lingfei Wu, Saizhuo Wang, Tengfei Ma, Fangli Xu, Alex X Liu, Chunming Wu, and Shouling Ji. Multilevel graph matching networks for deep graph similarity learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(2):799–813, 2021.

- [18] Nikita Mehrotra, Navdha Agarwal, Piyush Gupta, Saket Anand, David Lo, and Rahul Purandare. Modeling functional similarity in source code with graph-based siamese networks. *IEEE Transactions on Software Engineering*, 48(10):3771–3789, 2021.
- [19] Zhengguang Li, Hongfei Lin, Wei Zheng, Michael M Tadesse, Zhihao Yang, and Jian Wang. Interactive self-attentive siamese network for biomedical sentence similarity. *IEEE access*, 8:84093–84104, 2020.
- [20] Wei Bao, Wugedele Bao, Jinhua Du, Yuanyuan Yang, and Xiaobing Zhao. Attentive siamese lstm network for semantic textual similarity measure. In *2018 international conference on asian language processing (ialp)*, pages 312–317. IEEE, 2018.
- [21] Saurabhchand Bhati, Laureano Moro Velazquez, Jesús Villalba, and Najim Dehak. Lstm siamese network for parkinson’s disease detection from speech. In *2019 ieee global conference on signal and information processing (globalsip)*, pages 1–5. IEEE, 2019.
- [22] Yichi Zhang, Bryan Pardo, and Zhiyao Duan. Siamese style convolutional neural networks for sound search by vocal imitation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(2):429–441, 2018.
- [23] Ariyo Oluwasanmi, Muhammad Umar Aftab, Eatedal Alabdulkreem, Bulbula Kumeda, Edward Y Baagyere, and Zhiquang Qin. Captionnet: Automatic end-to-end siamese difference captioning model with attention. *IEEE access*, 7:106773–106783, 2019.
- [24] Hampshire and Waibel. A novel objective function for improved phoneme recognition using time delay neural networks. In *International 1989 Joint Conference on Neural Networks*, pages 235–241. IEEE, 1989.
- [25] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [26] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.
- [27] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

- [28] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *Similarity-based pattern recognition: third international workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*, pages 84–92. Springer, 2015.
- [29] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.
- [30] Monica Bianchini and Giovanna Maria Dimitri. Deep learning techniques for biomedical data processing. *Intelligent Decision Technologies*, 17(1):211–228, 2023.
- [31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [32] Kyunghyun Cho. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [33] Franco Scarselli, Sweah Liang Yong, Marco Gori, Markus Hagenbuchner, Ah Chung Tsoi, and Marco Maggini. Graph neural networks for ranking web pages. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 666–672. IEEE, 2005.
- [34] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering, 2017.
- [35] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.
- [36] Seung-Hwan Lim, Sangkeun Matt Lee, Sarah Powers, Mallikarjun Shankar, and Neena Imam. Survey of approaches to generate realistic synthetic graphs. *Oak ridge national laboratory*, 2015.
- [37] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.
- [38] Pietro Bongini, Monica Bianchini, and Franco Scarselli. Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450:242–252, 2021.
- [39] Pietro Bongini. Graph neural networks for drug discovery: An integrated decision support pipeline. In *2023 IEEE International Conference on Metrology for*

- eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRaine)*, pages 218–223. IEEE, 2023.
- [40] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.
- [41] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.
- [42] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.
- [43] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4–7, 2018, Proceedings, Part I* 27, pages 412–422. Springer, 2018.
- [44] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.
- [45] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [46] Dongbo Ma, Da Yuan, Maojun Huang, and Ling Dong. Vgc-gan: a multi-graph convolution adversarial network for stock price prediction. *Expert Systems with Applications*, 236:121204, 2024.
- [47] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021.
- [48] Hanqun Cao, Cheng Tan, Zhangyang Gao, Yilun Xu, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. A survey on generative diffusion models. *IEEE Transactions on Knowledge and Data Engineering*, 2024.
- [49] Jonathon Shlens. Notes on kullback-leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*, 2014.
- [50] Horst Bunke and Gudrun Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.

- [51] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern recognition letters*, 18(8):689–694, 1997.
- [52] Rishabh Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakravarthy, Yogish Sabharwal, and Sayan Ranu. Greed: A neural framework for learning graph distance functions. *Advances in Neural Information Processing Systems*, 35:22518–22530, 2022.
- [53] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13:113–129, 2010.
- [54] Michal Dory and Merav Parter. Fault-tolerant labeling and compact routing schemes. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 445–455, 2021.
- [55] Rashid Ibragimov, Maximilian Malek, Jiong Guo, and Jan Baumbach. Gedevo: an evolutionary graph edit distance algorithm for biological network alignment. In *German conference on bioinformatics 2013*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.
- [56] Sichen Deng and Yong Yu. Predicting structural similarity between molecules using graph neural networks. In *2022 10th International Conference on Bioinformatics and Computational Biology (ICBCB)*, pages 78–84. IEEE, 2022.
- [57] Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009.
- [58] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 384–392, 2019.
- [59] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [60] Xiaoli Wang, Xiaofeng Ding, Anthony KH Tung, Shanshan Ying, and Hai Jin. An efficient graph indexing method. In *2012 IEEE 28th International Conference on Data Engineering*, pages 210–221. IEEE, 2012.
- [61] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

- [62] Pietro Bongini, Niccolò Pancino, Franco Scarselli, and Monica Bianchini. Biognn: how graph neural networks can solve biological problems. In *Artificial Intelligence and Machine Learning for Healthcare: Vol. 1: Image and Data Analytics*, pages 211–231. Springer International Publishing Cham, 2022.
- [63] Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, et al. A comprehensive survey on community detection with deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [64] Pietro Bongini, Elisa Messori, Niccolò Pancino, and Monica Bianchini. A deep learning approach to the prediction of drug side-effects on molecular graphs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2023.
- [65] Niccolò Pancino, Yohann Perron, Pietro Bongini, and Franco Scarselli. Drug side effect prediction with deep learning molecular embedding in a graph-of-graphs domain. *Mathematics*, 10(23):4550, 2022.
- [66] Pietro Bongini, Franco Scarselli, Monica Bianchini, Giovanna Maria Dimitri, Niccolò Pancino, and Pietro Lio. Modular multi-source prediction of drug side-effects with drugnn. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20(2):1211–1220, 2022.
- [67] Niccolò Pancino, Alberto Rossi, Giorgio Ciano, Giorgia Giacomini, Simone Bonechi, Paolo Andreini, Franco Scarselli, Monica Bianchini, and Pietro Bongini. Graph neural networks for the prediction of protein-protein interfaces. In *ESANN*, pages 127–132, 2020.
- [68] John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.
- [69] Jintae Kim, Sera Park, Dongbo Min, and Wankyu Kim. Comprehensive survey of recent drug discovery using deep learning. *International Journal of Molecular Sciences*, 22(18):9983, 2021.
- [70] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. Neural network-based graph embedding for cross-platform binary code similarity detection. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 363–376, 2017.
- [71] Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, and Philip S Yu. Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery*, 35:688–725, 2021.

- [72] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102, 2008.
- [73] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- [74] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [75] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.
- [76] Pan Li and Jure Leskovec. The expressive power of graph neural networks. *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 63–98, 2022.
- [77] Francesc Serratosa. Redefining the graph edit distance. *SN Computer Science*, 2(6):438, 2021.
- [78] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.
- [79] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. Machine learning for anomaly detection: A systematic review. *Ieee Access*, 9:78658–78700, 2021.
- [80] Shaohua Qiu, Xiaopeng Cui, Zuowei Ping, Nanliang Shan, Zhong Li, Xianqiang Bao, and Xinghua Xu. Deep learning techniques in intelligent fault diagnosis and prognosis for industrial systems: A review. *Sensors*, 23(3):1305, 2023.
- [81] Kuei-Peng Lee, Bo-Huei Wu, and Shi-Lin Peng. Deep-learning-based fault detection and diagnosis of air-handling units. *Building and Environment*, 157:24–33, 2019.
- [82] Xiuying Yan, Ting Guan, Kaixing Fan, and Qing Sun. Novel double layer bilstm minor soft fault detection for sensors in air-conditioning system with kpca reducing dimensions. *Journal of Building Engineering*, 44:102950, 2021.
- [83] Sana Ullah Jan, Young Doo Lee, and In Soo Koo. A distributed sensor-fault detection and diagnosis framework using machine learning. *Information Sciences*, 547:777–796, 2021.

- [84] Irene Cappelli, Lorenzo Parri, Marco Tani, Valerio Vignoli, and Ada Fort. Pervasive monitoring in the context of precision agriculture: Using low cost ldr sensors for solar intensity measurement. *Accepted for publication on proceedings of 2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2024.
- [85] Tawseef Ayoub Shaikh, Tabasum Rasool, and Faisal Rasheed Lone. Towards leveraging the role of machine learning and artificial intelligence in precision agriculture and smart farming. *Computers and Electronics in Agriculture*, 198:107119, 2022.
- [86] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.
- [87] Mohamed Akram Zaytar and Chaker El Amrani. Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. *International Journal of Computer Applications*, 143(11):7–11, 2016.
- [88] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4834–4843, 2018.
- [89] Lincoln Moura de Oliveira, Francisco Eduardo Mendes da Silva, Gabriel Marçal da Cunha Pereira, Demercil Souza Oliveira, and Fernando Luiz Marcelo Antunes. Siamese neural network architecture for fault detection in a voltage source inverter. In *2021 Brazilian Power Electronics Conference (COBEP)*, pages 1–4. IEEE, 2021.
- [90] Jason W Anderson, Ken E Kennedy, Linh B Ngo, Andre Luckow, and Amy W Apon. Synthetic data generation for the internet of things. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 171–176. IEEE, 2014.
- [91] Chi Zhang, Sanmukh R Kuppannagari, Rajgopal Kannan, and Viktor K Prasanna. Generative adversarial network for synthetic time series data generation in smart grids. In *2018 IEEE international conference on communications, control, and computing technologies for smart grids (SmartGridComm)*, pages 1–6. IEEE, 2018.
- [92] Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. Sensegen: A deep learning architecture for synthetic sensor data generation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 188–193. IEEE, 2017.

- [93] Skyler Norgaard, Ramyar Saeedi, Keyvan Sasani, and Assefaw H Gebremedhin. Synthetic sensor data generation for health applications: A supervised deep learning approach. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1164–1167. IEEE, 2018.
- [94] Alvaro Figueira and Bruno Vaz. Survey on synthetic data generation, evaluation methods and gans. *Mathematics*, 10(15):2733, 2022.
- [95] Liangjun Chen, Weiyong Liu, Qi Zhang, Ke Xu, Guangming Ye, Weichen Wu, Ziyong Sun, Fang Liu, Kailang Wu, Bo Zhong, et al. Rna based mngs approach identifies a novel human coronavirus from two individual pneumonia cases in 2019 wuhan outbreak. *Emerging microbes & infections*, 9(1):313–319, 2020.
- [96] Xiaofen Zhong, Huanghao Yang, Zu-Feng Guo, Wan-Yee Fion Sin, Wei Chen, Junjie Xu, Ling Fu, Jie Wu, Chun-Kit Gannon Mak, Chak-Sum Samuel Cheng, et al. B-cell responses in patients who have recovered from severe acute respiratory syndrome target a dominant site in the s2 domain of the surface spike glycoprotein. *Journal of virology*, 79(6):3401–3408, 2005.
- [97] Jie Cui, Fang Li, and Zheng-Li Shi. Origin and evolution of pathogenic coronaviruses. *Nature reviews microbiology*, 17(3):181–192, 2019.
- [98] Angkana T Huang, Bernardo Garcia-Carreras, Matt DT Hitchings, Bingyi Yang, Leah C Katzelnick, Susan M Rattigan, Brooke A Borgert, Carlos A Moreno, Benjamin D Solomon, Luke Trimmer-Smith, et al. A systematic review of antibody mediated immunity to coronaviruses: kinetics, correlates of protection, and association with severity. *Nature communications*, 11(1):4704, 2020.
- [99] Resal Raj. Analysis of non-structural proteins, nsps of sars-cov-2 as targets for computational drug designing. *Biochemistry and biophysics reports*, 25:100847, 2021.
- [100] Zhiliang Cao, Lifeng Liu, Lanying Du, Chao Zhang, Shibo Jiang, Taisheng Li, and Yuxian He. Potent and persistent antibody responses against the receptor-binding domain of sars-cov spike protein in recovered patients. *Virology journal*, 7:1–6, 2010.
- [101] Yan-Jun Zhou, Rong-Hong Hua, Yun-Feng Wang, Tong-Qing An, Jin-Xia Liu, Jin-Yu Yang, Yu-Zhuo Hua, and Guang-Zhi Tong. Development of monoclonal antibodies against sars-cov and identification of antigenic epitopes.

- Sheng wu Gong Cheng xue bao= Chinese Journal of Biotechnology*, 21(2):211–215, 2005.
- [102] Himani Bisht, Anjeanette Roberts, Leatrice Vogel, Alexander Bukreyev, Peter L Collins, Brian R Murphy, Kanta Subbarao, and Bernard Moss. Severe acute respiratory syndrome coronavirus spike protein expressed by attenuated vaccinia virus protectively immunizes mice. *Proceedings of the National Academy of Sciences*, 101(17):6641–6646, 2004.
- [103] Yan Ren, Zhengfeng Zhou, Jinxiu Liu, Liang Lin, Shuting Li, Hao Wang, Ji Xia, Zhe Zhao, Jie Wen, Cuiqi Zhou, et al. A strategy for searching antigenic regions in the sars-cov spike protein. *Genomics, Proteomics and Bioinformatics*, 1(3):207–215, 2003.
- [104] Benjamin Meyer, Christian Drosten, and Marcel A Müller. Serological assays for emerging coronaviruses: challenges and pitfalls. *Virus research*, 194:175–183, 2014.
- [105] Chunyu Liu, Xiuhua Li, Chuntao Zhang, Sihong Xu, Yiming Shao, Hui Zhuang, Xiaoyan Che, Yan Qiu, Hongzhang Yin, Defu Li, et al. Establishment of a reference panel for the detection of anti-sars-cov antibodies. *Biologicals*, 35(3):203–210, 2007.
- [106] Jose Mateus, Alba Grifoni, Alison Tarke, John Sidney, Sydney I Ramirez, Jennifer M Dan, Zoe C Burger, Stephen A Rawlings, Davey M Smith, Elizabeth Phillips, et al. Selective and cross-reactive sars-cov-2 t cell epitopes in unexposed humans. *Science*, 370(6512):89–94, 2020.
- [107] Md Asif Ahsan, Yongjing Liu, Cong Feng, Ralf Hofestädt, and Ming Chen. Overcovid: an integrative web portal for sars-cov-2 bioinformatics resources. *Journal of integrative bioinformatics*, 18(1):9–17, 2021.
- [108] Susanna KP Lau, Hayes KH Luk, Antonio CP Wong, Kenneth SM Li, Longchao Zhu, Zirong He, Joshua Fung, Tony TY Chan, Kitty SC Fung, and Patrick CY Woo. Possible bat origin of severe acute respiratory syndrome coronavirus 2. *Emerging infectious diseases*, 26(7):1542, 2020.
- [109] Wenhui Li, Michael J Moore, Natalya Vasilieva, Jianhua Sui, Swee Kee Wong, Michael A Berne, Mohan Somasundaran, John L Sullivan, Katherine Luzuriaga, Thomas C Greenough, et al. Angiotensin-converting enzyme 2 is a functional receptor for the sars coronavirus. *Nature*, 426(6965):450–454, 2003.

- [110] Markus Hoffmann, Hannah Kleine-Weber, Simon Schroeder, Nadine Krüger, Tanja Herrler, Sandra Erichsen, Tobias S Schiergens, Georg Herrler, Nai-Huei Wu, Andreas Nitsche, et al. Sars-cov-2 cell entry depends on ace2 and tmprss2 and is blocked by a clinically proven protease inhibitor. *cell*, 181(2):271–280, 2020.
- [111] Heike Hofmann, Krzysztof Pyrc, Lia Van Der Hoek, Martina Geier, Ben Berkhout, and Stefan Pöhlmann. Human coronavirus nl63 employs the severe acute respiratory syndrome coronavirus receptor for cellular entry. *Proceedings of the National Academy of Sciences*, 102(22):7988–7993, 2005.
- [112] Ayal B Gussow, Noam Auslander, Guilhem Faure, Yuri I Wolf, Feng Zhang, and Eugene V Koonin. Genomic determinants of pathogenicity in sars-cov-2 and other human coronaviruses. *Proceedings of the National Academy of Sciences*, 117(26):15193–15199, 2020.
- [113] Alba Grifoni, Daniela Weiskopf, Sydney I Ramirez, Jose Mateus, Jennifer M Dan, Carolyn Rydyznski Moderbacher, Stephen A Rawlings, Aaron Sutherland, Lakshmanane Premkumar, Ramesh S Jadi, et al. Targets of t cell responses to sars-cov-2 coronavirus in humans with covid-19 disease and unexposed individuals. *Cell*, 181(7):1489–1501, 2020.
- [114] Suresh Kumar, Thiviya S Thambiraja, Kalimuthu Karuppanan, and Gunasekaran Subramaniam. Omicron and delta variant of sars-cov-2: a comparative computational study of spike protein. *Journal of medical virology*, 94(4):1641–1649, 2022.
- [115] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.
- [116] NCBI Resource Coordinators. Database resources of the national center for biotechnology information. *Nucleic acids research*, 44(Database issue):D7, 2016.
- [117] Franz X Heinz and Karin Stiasny. Profile of sars-cov-2. *Wiener klinische Wochenschrift*, 132(21):635–644, 2020.
- [118] Javier A Jaimes, Nicole M André, Joshua S Chappie, Jean K Millet, and Gary R Whittaker. Phylogenetic analysis and structural modeling of sars-cov-2 spike protein reveals an evolutionary distinct and proteolytically sensitive activation loop. *Journal of molecular biology*, 432(10):3309–3325, 2020.

- [119] Zigui Chen, Siaw S Boon, Maggie H Wang, Renee WY Chan, and Paul KS Chan. Genomic and evolutionary comparison between sars-cov-2 and other human coronaviruses. *Journal of virological methods*, 289:114032, 2021.
- [120] DingLiuJiaFung XQ Liang. To. s.(2021). *Human Coronavirus-229E,-OC43,-NL63, and-HKU1 (Coronaviridae)*. *Encycl. Virology*, 2:428–440.
- [121] Rachel T Eguia, Katharine HD Crawford, Terry Stevens-Ayers, Laurel Kelnhofer-Millevolte, Alexander L Greninger, Janet A Englund, Michael J Boeckh, and Jesse D Bloom. A human coronavirus evolves antigenically to escape antibody immunity. *PLoS pathogens*, 17(4):e1009453, 2021.
- [122] Rohitash Yadav, Jitendra Kumar Chaudhary, Neeraj Jain, Pankaj Kumar Chaudhary, Supriya Khanra, Puneet Dhamija, Ambika Sharma, Ashish Kumar, and Shailendra Handu. Role of structural and non-structural proteins and therapeutic targets of sars-cov-2 for covid-19. *Cells*, 10(4):821, 2021.
- [123] Jian Lei, Yuri Kusov, and Rolf Hilgenfeld. Nsp3 of coronaviruses: Structures and functions of a large multi-domain protein. *Antiviral research*, 149:58–74, 2018.
- [124] Filipe Pereira. Evolutionary dynamics of the sars-cov-2 orf8 accessory gene. *Infection, Genetics and Evolution*, 85:104525, 2020.
- [125] Sergio Forcelloni, Anna Benedetti, Maddalena Dilucca, and Andrea Giansanti. Identification of conserved epitopes in sars-cov-2 spike and nucleocapsid protein. *Current Genomics*, 22(7):541, 2021.
- [126] Hong-Zhi Chen, Ling-Li Tang, Xin-Ling Yu, Jie Zhou, Yun-Feng Chang, and Xiang Wu. Bioinformatics analysis of epitope-based vaccine design against the novel sars-cov-2. *Infectious diseases of poverty*, 9:1–10, 2020.
- [127] Henryk Kozlowski, Maciej Luczkowski, Maurizio Remelli, and Daniela Valensin. Copper, zinc and iron in neurodegenerative diseases (alzheimer's, parkinson's and prion diseases). *Coordination Chemistry Reviews*, 256:2129–2141, 2012.
- [128] Noa Stern, Alexandra Gacs, Enikő Tátrai, Beáta Flachner, István Hajdú, Krisztina Dobi, István Bágyi, György Dormán, Zsolt Lőrincz, Sándor Cseh, et al. Dual inhibitors of ache and bace-1 for reducing a β in alzheimer's disease: From in silico to in vivo. *International Journal of Molecular Sciences*, 23(21):13098, 2022.

- [129] P. Nagu, A. K. A. Pathan, and V. Mehta. Screening of herbal molecules for the management of alzheimer's disorder: In silico and in vitro approaches. *Applied Biological Research*, 24(3):255–272, 2022.
- [130] J. L. Biedler, S. Roffler-Tarlov, M. Schachner, and L. S. Freedman. Multiple neurotransmitter synthesis by human neuroblastoma cell lines and clones. *Cancer Research*, 11:3751–3757, 1978.
- [131] Y. Luo et al. Role of amber extract in protecting sh-sy5y cells against amyloid beta 1-42-induced neurotoxicity. *Biomedicine & Pharmacotherapy*, 2022.
- [132] M. Bell and H. Zempel. A human cell model for tau sorting and vulnerability. *Reviews in the Neurosciences*, 33(1):1–15, 2022.
- [133] K. Huang, N. Thomas, P. R. Gooley, and C. W. Armstrong. Systematic review of nmr-based metabolomics practices in human disease research. *Human Metabolites*, 2022.
- [134] G. H. R. da Silva, L. F. Mendes, F. V. de Carvalho, E. de Paula, and I. F. Duarte. Comparative metabolomics study of the impact of articaine and lidocaine on the metabolism of sh-sy5y neuronal cells. *Human Metabolites*, 2022.
- [135] D. Paris et al. Metabolic response of sh-sy5y cells to gold nanoparticles by nmr-based metabolomics analyses. *Human Metabolites*, 2016.
- [136] C. Corsaro, S. Vasi, F. Neri, A. M. Mezzasalma, G. Neri, and E. Fazio. Nmr in metabolomics: From conventional statistics to machine learning and neural network approaches. *Applied Sciences*, 12:2824, 2022.
- [137] Piotr Klukowski, Roland Riek, and Peter Güntert. Rapid protein assignments and structures from raw nmr spectra with the deep learning technique artina. *Nature Communications*, 13:6151, 2022.
- [138] Piotr Klukowski, Marek Augoff, Mateusz Zięba, Maciej Drwal, Adrian Gonczarek, and M. J. Walczak. Nmrnet: a deep learning approach to automated peak picking of protein nmr spectra. *Bioinformatics*, 34(15):2590–2597, 2018.
- [139] Jing Luo, Qiang Zeng, Kaibo Wu, and Yu Lin. Fast reconstruction of non-uniform sampling multidimensional nmr spectroscopy via a deep neural network. *Journal of Magnetic Resonance*, 317:106772, 2020.
- [140] Taeho Jo, Kwangsik Nho, and Andrew J. Saykin. Deep learning in alzheimer's disease: Diagnostic classification and prognostic prediction using neuroimaging data. *Frontiers in Aging Neuroscience*, 11:220, 2019.

- [141] Shuo Liu, Arjun V. Masurkar, Henry Rusinek, et al. Generalizable deep learning model for early alzheimer's disease detection from structural mris. *Scientific Reports*, 12:17106, 2022.
- [142] Alessandro Rossi, Giovanni Vannuccini, Pierpaolo Andreini, et al. Analysis of brain nmr images for age estimation with deep learning. *Procedia Computer Science*, 159:981–989, 2019.
- [143] Shanshan Qiu, Michael I. Miller, Pradeep S. Joshi, et al. Multimodal deep learning for alzheimer's disease dementia assessment. *Nature Communications*, 13:3404, 2022.
- [144] Christopher E. Wisely, Dan Wang, Ricardo Henao, et al. Convolutional neural network to identify symptomatic alzheimer's disease using multimodal retinal imaging. *British Journal of Ophthalmology*, 106:388–395, 2022.
- [145] Carol Y. Cheung et al. A deep learning model for detection of alzheimer's disease based on retinal photographs: a retrospective, multicentre case-control study. *The Lancet Digital Health*, 4(11):E806–E815, 2022.
- [146] Ministero della Salute. Relazione del ministro della salute al parlamento sullo stato di attuazione della legge contenente norme in materia di procreazione medicalmente assistita (legge 19 febbraio 2004, n. 40, articolo 15), 2021.
- [147] GD. Adamson, J. de Mouzon, G. Chambers, F. Zegers-Hochschild, O. Ishihara, M. Banker, S. Dyer, and M. Kupka. International committee for monitoring assisted reproductive technology: world report on assisted reproductive technology. 2017.
- [148] P. Andreini, G. Ciano, S. Bonechi, C. Graziani, V. Lachi, A. Mecocci, A. Sodi, F. Scarselli, and M. Bianchini. A two-stage GAN for high-resolution retinal image generation and segmentation. *Electronics*, 11(1):60, 2021.
- [149] G. Ciano, P. Andreini, T. Mazzierli, M. Bianchini, and F. Scarselli. A multi-stage GAN for multi-organ chest X-ray image generation and segmentation. *Mathematics*, 9(22), 2021.
- [150] S. Bonechi, M. Bianchini, F. Scarselli, and P. Andreini. Weak supervision for generating pixel-level annotations in scene text segmentation. *Pattern Recognition Letters*, 138:1–7, 2020.