# DEEP LEARNING-BASED ALGORITHM FOR PREDICTING COVID-19 PATIENT MORTALITY



ZHEXIN TU

UNIVERSITI KEBANGSAAN MALAYSIA

# DEEP LEARNING-BASED ALGORITHM FOR PREDICTING COVID-19 PATIENT MORTALITY

ZHEXIN TU



PROJECT SUBMITTED IN PARTIAL FULFILMENT FOR THE DEGREE OF MASTER OF DATA SCIENCE

FACULTY OF INFORMATION SCIENCE AND TECHNOLOGY
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2024

# ALGORITMA BERASASKAN PEMBELAJARAN DALAM UNTUK MEMPREDIKSI KEMATIAN PESAKIT COVID-19

ZHEXIN TU



PROJEK YANG DIKEMUKAKAN UNTUK MEMENUHI SEBAHAGIAN
DARIPADA SYARAT MEMPEROLEH IJAZAH
SARJANA SAINS DATA

FAKULTI TEKNOLOGI DAN SAINS MAKLUMAT
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2024

**DECLARATION**

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged.

22 June 2024

ZHEXIN TU
P126480

**ACKNOWLEDGEMENT**

Above all, I am deeply grateful to my parents for their unwavering support and blessings, which have endowed me with patience and sustained my well-being throughout this master's research journey.

I am very fortunate to have Associate Professor Dr. Zalinda Othman as a research supervisor.

## ABSTRAK

Pertubuhan Kesihatan Sedunia (WHO) telah secara rasmi mengisytiharkan COVID-19 sebagai pandemik global, mendorong usaha terkoordinasi di seluruh dunia untuk mengurangkan penyebaran virus ini. Secara global, ratusan juta orang telah dijangkiti virus ini, mengakibatkan berjuta-juta kematian. Pandemik COVID-19 secara signifikan mempengaruhi kesihatan global, ekonomi, perjalanan, dan rantaian bekalan. Namun, simptom yang dialami oleh individu yang dijangkiti berbeza-beza bergantung kepada status kesihatan individu tersebut, dengan simptom termasuk demam, sesak nafas, dan batuk kering, yang boleh membawa maut dalam kes-kes yang teruk. Disebabkan oleh masalah kecekapan pengenalan yang rendah dan prestasi yang tidak stabil dalam pengenalan sampel set data COVID-19, kertas ini mencadangkan algoritma ramalan 1D-CNN-LSTM untuk kematian pesakit COVID-19 dengan menggunakan kaedah ramalan pembelajaran mendalam tradisional seperti model 1D-CNN, model LSTM dan model GRU, digabungkan dengan model ramalan pembelajaran mendalam. ACC, SEN, SPE dan F1 bagi keempat-empat model algoritma 1D-CNN, LSTM, GRU dan 1D-CNN-LSTM dibandingkan. Keputusan menunjukkan bahawa: Algoritma yang kami reka, gabungan 1D-CNN dan LSTM, mempamerkan ketepatan pengkelasan yang luar biasa sebanyak 98.96%, dicapai dengan menetapkan kedalaman rangkaian pada 128 lapisan dan mengulangi proses latihan sebanyak 150 kali. Kepekaannya (SEN), kekhususannya (SPE), dan skor F1 masing-masing mencapai 95.85%, 96.43%, dan 97.05%. Selain itu, dengan pengesahan silang 5-lipat, model ini menunjukkan kadar ketepatan yang kukuh sebanyak 97%, menunjukkan kapasiti generalisasi yang kuat. Ini mengatasi pendekatan ramalan penyelidik lain dengan ketara. Implikasi hasil ini adalah penting bagi badan kerajaan dan pentadbir kesihatan awam semasa mereka merancang dan menyesuaikan taktik untuk pencegahan wabak. Penemuan yang diperoleh di sini boleh menjadi penting, membekalkan pihak berkepentingan dengan maklumat penting tentang trajektori dan corak prospektif penyakit berjangkit, dengan itu meningkatkan kesedaran dan pengukuhan kesihatan awam.

## ABSTRACT

The World Health Organization（WHO）has officially designated COVID-19 as a global pandemic, prompting coordinated efforts worldwide to mitigate the spread of the virus. Globally, hundreds of millions of people have been infected with the virus, resulting in millions of deaths. The COVID-19 pandemic is significantly affecting global healthcare, economies, travel, and the supply chain. However, the symptoms of infected people vary depending on the individual's health status, with symptoms including fever, dyspnea, and dry cough, which can lead to death in severe cases. Due to the problems of low recognition efficiency and unstable performance in the sample recognition of COVID-19 data sets, this paper proposed a 1D-CNN-LSTM prediction algorithm for COVID-19 patient death by using traditional deep learning prediction methods such as 1D-CNN model, LSTM model and GRU model, combined with deep learning prediction model. ACC, SEN, SPE and F1 of the four algorithm models 1D-CNN, LSTM, GRU and 1D-CNN-LSTM are compared. The results show that: The algorithm we've devised, a synthesis of 1D-CNN and LSTM, exhibits a remarkable classification precision of 98.96%, achieved by setting the network depth at 128 layers and iterating the training process 150 times. Its sensitivity (SEN), specificity (SPE), and F1 score hit 95.85%, 96.43%, and 97.05%, respectively. Moreover, with 5-fold cross-validation, the model demonstrated a solid accuracy rate of 97%, indicating a potent capacity for generalization. This outstrips the predictive approaches of other investigators notably. The implications of these outcomes are substantive for both governmental bodies and public health administrators as they craft and fine-tune tactics for epidemic prevention. The insights garnered here could be pivotal, furnishing these stakeholders with essential information on the trajectory and prospective patterns of infectious diseases, thereby enhancing awareness and fortification of public health.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF ILLUSTRATIONS

## LIST OF ABBREVIATIONS

| | |
|---|---|
| CNN | Convolutional Neural Network |
| DL | Deep Learning |
| ANN | Artificial Neural Network |
| RNN | Recurrent Neural Network |
| 1D-CNN | One-Dimensional Convolutional Neural Networks |
| LSTM | Long Short-Term Memory |
| GRU | Gated Recurrent Units |
| F1 | F1 measure |
| SEN | Sensitivity |
| SPE | Specificity |
| ACC | Accuracy |
| DL | Deep Learning |
| TP | True Positives |
| FP | False Positives |

# CHAPTER I

# INTRODUCTION

## 1.1 RESEARCH BACKGROUND

Coronaviruses are members of a viral family known for triggering respiratory illnesses, which can turn fatal in some cases, with SARS and MERS being quintessential instances. Some specific types of these viruses can infect animals, and in very rare cases, these viruses can transmit from animals to humans and start to spread among the human population. Research on novel coronaviruses shows that they can rapidly transmit among humans after jumping from animal hosts and cause significant disruption to the human health system (McConghy et al. 2020).

As 2019 ended, the first cases of a severe respiratory illness caused by a previously unidentified coronavirus (SARS-CoV-2) emerged in Wuhan. Subsequently, this condition was designated as COVID-19, an abbreviation for Corona Virus Disease 2019, by the World Health Organization. Present statistics suggest that the gestation period for this infectious illness typically varies between 6 to 14 days. Initial symptoms often encompass a dry cough, fever, shortness of breath, and severe tiredness. Some patients may also experience respiratory and digestive issues such as nasal congestion and diarrhea, which in extreme cases can be fatal. Recent studies show that outbreaks of the novel coronavirus can greatly strain the medical systems of affected areas in a short time. When hospital systems exceed their carrying capacity, the mortality rate may significantly increase (Li et al. 2005).

In the early stages of COVID-19' s spread, humanity knew very little about SARS-CoV-2 and was unable to take effective countermeasures, leading to a rapid increase in infections globally. The proliferation of COVID-19 has substantially influenced international healthcare, economic stability, travel industries, and the availa-

bility of commodities. As confirmed cases of COVID-19 surged globally, the WHO announced that the coronavirus outbreak exhibited all the hallmarks of a pandemic. As understanding and research of the virus continued to deepen, governments worldwide gradually implemented various interventions, such as maintaining social distancing, wearing masks, and implementing regional lockdowns, effectively controlling the spread of the epidemic. According to data released by the WHO on November 7, 2021, the worldwide tally for COVID-19 infections has climbed to an estimated figure of around 249.5 million, with over 3.14 million deaths. To end the pandemic as soon as possible, global healthcare workers have put great effort into controlling the virus's spread, vaccine development, and treatment methods. Several COVID-19 vaccines have been developed globally. However, according to the latest research, even with a high vaccination rate, a large-scale infection can still occur when vaccine efficacy is low. Thus, the development of effective COVID-19 vaccines is urgent (Wang et al. 2005; Palmer et al. 2020). Predicting the mortality risk of COVID-19 patients is crucial for vaccine development and application, helping to minimize the death rate.



Forecasting the potential fatality of individuals with COVID-19 is a crucial element in vaccine development and implementation, playing a significant role in reducing the mortality rate. Accurate predictions can optimize the allocation of medical system resources to cope with the pandemic's development, and concentrate medical, economic, and human resources in areas severely affected by the outbreak.

The rapid development of artificial intelligence has ushered in a new climax in the global information technology revolution. With the continuous improvement of computers' ability to analyze data, AI algorithms have become more accurate and widely applied in areas such as disease mortality risk prediction, assisted diagnosis, image analysis, and health management. The swift progress of artificial intelligence, particularly the significant advances in machine learning algorithms as its core technology, has shown exceptional capabilities in weather forecasting, risk analysis, and voice recognition. Deep learning in machine learning, especially the representation learning methods based on neural networks, has become a hot research topic since achieving breakthrough success in an image classification contest in 2012 (Krizhevsky et al. 2014). Advancements in deep learning have markedly propelled the fields of

natural language processing, speech recognition, image classification, and disease prediction forward. Particularly in today's rapidly developing medical informatics, Deep learning has surfaced as an exceptionally promising field of research for forecasting the mortality of COVID-19 patients, with its accuracy and predictive performance far surpassing traditional statistical methods and machine learning technologies.

Therefore, this research explores the influence various advanced deep learning models have on predicting the fatality rates associated with COVID-19 patients.

## 1.2 RESEARCH PROBLEM

The intricate and ever-changing characteristics of the novel coronavirus render forecasting the death rates among COVID-19 patients a complex task. Existing models often fail to predict COVID-19 patient deaths effectively, resulting in low reliability of outcomes. Research suggests that conventional statistical frameworks and elementary machine learning techniques may fall short in their capacity to manage sequential datasets and assimilate evolving data trends. (Zheng et al. 2020). Deep learning models, especially LSTM and GRU, have demonstrated potential in processing and learning sequential data in other medical applications (Hochreiter et al. 1997; Cho et al. 2014).



However, these models have not been sufficiently applied in predicting COVID-19 patient mortality. Additionally, the methodology introduced by Du et al. (2023) that integrates 1D-CNN with LSTM to extract features from intricate datasets remains largely underexplored, particularly in predicting fatalities among COVID-19 patients. This study aims to develop and test a model that combines 1D-CNN with LSTM for predicting the mortality of COVID-19 patients. It also assesses how this model stacks up against other forecasting methods created by researchers.

## 1.3 RESEARCH QUESTIONS

The main problem is how to apply deep learning algorithms to accurately predict COVID-19 death. Solving the challenge of deep learning to predict COVID-19 patient mortality requires us to address several delicate issues. It includes:

1. What is the most important feature for predicting COVID-19 mortality? How to

make feature selection?

2. How to perform feature engineering to improve model performance? For example, is it necessary to smooth, normalize, or standardize COVID-19 dataset?

3. What's the ideal way to integrate time series and geospatial data into models effectively?

4. How to choose appropriate loss functions, optimization algorithms, and hyperparameters for model training to improve predictive performance?

5. How to construct effective evaluation indicators to accurately evaluate the predictive power of the model?

6. How to perform cross validation and external validation of the model to ensure its stability and robustness on different datasets?

7. How to compare the proposed model with other predictive methods to demonstrate its superiority?



### 1.4 RESEARCH OBJECTIVES

The main goal of this study is to develop a forecasting model for COVID-19 based on deep learning techniques. We aim to evaluate its forecasting proficiency with the goal of bolstering the precision of our predictions. To accomplish this primary aim, the following targets have been set:

1. Conduct an analysis of the dataset to identify suitable features for the network model inputs. The chosen features undergo preprocessing to enhance the training efficiency of the model.

2. Develop four models for prediction based on commonly used deep learning prediction algorithms, namely 1D-CNN、LSTM、GRU、1D-CNN-LSTM.

3. Adjust the hyperparameter Settings of the four models to enhance the predictions of the four models and ensure the best results on the evaluation data set.

4. Analyze and compare four algorithms, select the best one for predicting the mortality rate of COVID-19 patient, and compare it with other research methods to demonstrate the superiority of this study's algorithm.

## 1.5 RESEARCH SCOPE

This study aims to develop a sophisticated deep-learning strategy with the goal of enhancing the precision of death predictions among individuals afflicted with COVID-19. To this end, commonly used deep learning algorithms including 1D-CNN, LSTM, GRU have been employed, and a hybrid model combining 1D-CNN with LSTM has been proposed for predicting patient mortality. Each model's effectiveness will be gauged through its performance on the test set, employing targeted hyperparameter tuning strategies to finetune and enhance each model's predictive accuracy to its fullest potential. The effectiveness of the models will be measured against specific benchmarks and pitted against alternative forecasting methods to highlight the proposed model's superior performance.



## 1.6 THESIS ORGANIZATION

Leveraging the power of advanced deep learning algorithms, this study centers around employing 1D-CNN, GRU, LSTM, and Optimal combination of 1D-CNN-LSTM predicts death in patients with COVID-19. Additionally, the project aims at fine-tuning these cutting-edge predictive models to elevate the precision of the outcomes. The structure of this study is as follows: Chapter I, provides preliminary insight into the background of the study's subject matter, leading up to a clear delineation of the issue at hand, and concludes by elucidating the objectives of our ongoing research; it leads into the study's content and the organization of the article's chapters, and finally, it introduces the scope of the research.

Chapter II primarily offers a comprehensive review of studies on identifying COVID-19 patients. The article commences with a comprehensive overview of COVID-19 testing, followed by an introduction to the virus itself and an update on the latest developments in diagnostic methods for the disease. Next, it introduces COVID-19 patient mortality prediction and its definition, presents deep learning and related algorithms, and then explores the most recent progress in research on utilizing deep

learning for predicting COVID-19 patient mortality. Finally, it discusses the limitations found in the literature and proposes innovative solutions.

Chapter III is about methodology. This chapter will delve into the research basis of COVID-19 patient mortality prediction, including study structure, study design, data set introduction and related processing, basic introduction to deep learning, principles of common algorithms like CNN、GRU、LSTM, and performance evaluation criteria for prediction algorithms. The establishment of these theoretical and technical frameworks will provide a solid theoretical and practical basis for subsequent algorithm optimization, model validation and clinical application, and enhances the effectiveness and precision of responses to epidemics.

Chapter IV explores the application of 1D-CNN, LSTM, and GRU models, comparing their efficacy. A novel model based on 1D-CNN and LSTM is proposed for predicting death in COVID-19 patients. Initially, 1D-CNN is utilized to extract feature information from the data. Subsequently, LSTM is employed for data analysis and interpretation, facilitating mortality prediction through model learning and inference. The models were developed and assessed using the same dataset of COVID-19 patients, with subsequent analysis of the results. Finally, the proposed approach was evaluated alongside other predictive methods to validate its effectiveness.



Finally, Chapter V summarizes the main research findings and the innovations of this study, discusses the potential extensions of the research findings, and looks forward to the application prospects of this research.

# CHAPTER II

# LITERATURE REVIEW

## 2.1 INTRODUCTION

On the last day of 2019, Wuhan, China, became ground zero for the emergence of SARS-CoV-2, the virus responsible for COVID-19. Fast forward to March 2020, and the WHO was sounding the alarm, officially labeling it a worldwide pandemic. The relentless spread of COVID-19 has since resulted in a staggering tally of over 100 million people infected globally, with the death toll surpassing 3 million. (Salman et al. 2020). Consequently, it is essential to identify effective ways to rapidly pinpoint populations most likely to be infected by the virus. The impact of COVID-19 varies greatly from person to person; it ranges from mild to severe symptoms in most cases, with fever, a persistent cough, and lethargy being the most prevalent. Nonetheless, a smaller percentage of those afflicted may undergo extreme manifestations which can include discomfort or constriction in the chest, impairment of speech or bodily movement, or in grave circumstances, it may culminate in fatality (Chen et al. 2020).



Despite the lack of specific treatments or vaccines for COVID-19, many clinical trials are evaluating potential treatments. Even in the absence of vaccines or specific treatments, infection can be prevented through measures such as frequent hand washing, self-isolation, and wearing masks, which effectively protect against COVID-19. Several studies have presented varied laboratory findings from the early stages of the COVID-19 outbreak. While most COVID-19 patients have mild conditions, clinical outcomes vary significantly among different patients (Li et al. 2020; Huang et al. 2020). Therefore, predicting which patients are more likely to develop severe illness or die is crucial.

Predicting COVID-19 patient mortality is a significant branch of COVID-19 testing research. Mortality prediction for COVID-19 patients primarily employs statisti-

cal methods, machine learning, or deep learning to estimate the likelihood of death for someone infected with the virus. Over the past few years, several predictive algorithms have been proposed. However, only a relatively small portion of these have focused on COVID-19 mortality prediction. This chapter provides an overview of the research in the field of COVID-19 patient testing. It begins with an overview of COVID-19 testing, including an introduction to COVID-19 and the progress in research on testing methods. Next, it discusses the prediction of COVID-19 patient mortality and its definitions. and introduces deep learning and related algorithms. It then reviews the current progress in research on deep learning-based mortality prediction for COVID-19 patients. Finally, it discusses limitations found in the current literature and proposes innovative methods to address.

## 2.2 COVID-19 PREDICTION

Now COVID-19 has become a global pandemic, and according to the latest reports from the World Health Organization, the virus has widely spread in multiple countries (Covid et al. 2020). Typical symptoms include fever and cough, but patients may also experience fatigue, headaches, and shortness of breath. These symptoms are not specific to COVID-19, so a definitive diagnosis requires specialized testing (Akçay et al. 2020； Chen et al. 2020).



According to research by McCongahy et al. (2020), medical systems can come under tremendous pressure within just a few weeks of an outbreak. Once the capacity of hospitals is surpassed, the mortality rate may surge dramatically. Consequently, the scientific community is intensifying research efforts to accurately predict the mortality risk of COVID-19 patients, making this research direction crucial.

## 2.3 PROGRESS IN RESEARCH ON COVID-19 PREDICTION METHODS

Starting with basic qualitative forecasts for COVID-19 patients based on empirical insights and hands-on experience, we have progressed to the utilization of models for predicting disease outcomes. This evolution in COVID-19 prediction methods has significantly contributed to advancements in the medical field. We will now examine the current state of these prediction techniques from three distinct perspectives. Each offering a comprehensive analysis of research conducted in their respective domains.

### 2.3.1 Statistical-based Prediction Methods for COVID-19

Knight et al. (2020) proposed a generalized additive model that incorporates continuous smooth predictors (penalized thin plate splines) and categorical predictors as linear components. They then linearly smoothed these continuous predictors and determined the optimal cutoff values. Ultimately, the model was precisely configured using the using the LASSO logistic regression. This COVID-19 prediction model produced an effective 4C mortality score with excellent predictive performance.

Zhou et al. (2020) conducted an in-depth exploration of various factors associated with the risk of in-hospital mortality among patients. The research team not only compared the electronic health records (EHRs) of survivors and non-survivors, but also extracted comprehensive data covering demographics, clinical information, treatment plans, and laboratory tests (particularly serial samples for viral RNA detection). By employing univariate and multivariate logistic regression methods, they successfully constructed a predictive model and achieved significant predictive outcomes.

Sorlini et al. (2020) compared the mortality of residents infected and not infected with COVID-19 using descriptive statistics and Cox proportional hazards models, categorized by risk factors, with good classification results.



### 2.3.2 Machine Learning-based Prediction Models for COVID-19

In an early work, Albahri et al.（2020）designed a machine learning (ML) technique for identifying the genomes of COVID-19. They developed a decision tree approach and utilized publicly accessible COVID-19 data containing Sarbecovirus and Betacorona virus. for performance evaluation. The proposed model, based on experimental studies, achieved a 100% accuracy rate, addressing classification accuracy, predictive precision, and other evaluation metrics.

Shaban et al. (2020) proposed an improved K-Nearest Neighbor (K-NN) model, referred to as the Enhanced K-Nearest Neighbor (EKNN) model, and developed a detection strategy for COVID-19 patients using EKNN. The results demonstrated that this strategy provided faster, and more accurate outcomes compared to other techniques available at the time.

### 2.3.3 Deep Learning-Based Prediction Models COVID-19

In a prior study, Zhang et al. (2021) pioneered a technique for segmenting medical images specific to COVID-19 related lung CT scans. By amalgamating a cutting-edge high-density GAN data set and multi-layer attention mechanism principles from U-Net, they created a tool that is proficient at affirming the presence of COVID-19 in patients.

Later, Kuvvetli et al. (2021) used three different ANN models to predict daily death tolls and COVID-19 case numbers. To forecast the severity of COVID-19, a severity prediction model was constructed using logistic regression and ANN.

Liang et al. (2020) developed a DL model which predicted the probability of serious disease progression in COVID-19 patients, using the medical data collected at the time of hospital check-in. Through the development of a web-based calculator for assessing incoming patients, the model pinpoints individuals at heightened risk of severe symptoms. This identification ensures that the most vulnerable patients receive prompt and tailored treatment. Consequently, it helps in the optimal distribution of medical resources.



## 2.4 COVID-19 PATIENT MORTALITY PREDICTION

This section provides the definition of COVID-19 patient mortality prediction, and then reviews and discusses the methods for predicting mortality in COVID-19 patients.

### 2.4.1 The Definition of "COVID-19 Patient Mortality Prediction"

The definition of "COVID-19 patient mortality prediction" refers to the use of data analysis techniques to assess the probability of death in patients diagnosed with COVID-19. This process involves a comprehensive analysis of information from various aspects, including the patient's clinical symptoms, laboratory test results, underlying health conditions, and sociodemographic characteristics. By applying statistical methods, machine learning, or deep learning approaches, researchers can build predictive models aimed at estimating the risk of death for patients infected with the coronavirus.

## 2.5 DEEP LEARNING

The foundational idea of Artificial Neural Networks (ANN) was introduced in 1943 as a mathematical model for emulating artificial neurons (Tan 2017; Tang 2018; Khoei et al. 2023). This concept laid the groundwork for the development of DL, which was formally brought into the forefront by 2006. It is based on multi-layered ANN models and has demonstrated powerful learning capabilities, offering great hope for solving many problems in fields such as anomaly detection, disease diagnosis, and image recognition (LeCun et al. 2015).

As a result, early studies on ANNs focused primarily on networks with a single hidden layer and utilized backpropagation for training purposes, as observed by Rumelhart et al. (1986). These networks with just one hidden layer are involved in shallow learning. Deep learning, which involves multiple hidden layers, was first implemented on computers in 2006(Wickramasinghe et al. 2006).

DL models can be classified into four distinct types, each characterized by the type of feedback they employ: supervised models, unsupervised models, reinforcement learning models, and hybrid models. Among these, supervised deep learning models stand out as a primary category, as they utilize labeled training datasets for training. These models assess accuracy using functions and loss functions, adjusting weights to effectively minimize errors. Key model types within the supervised deep learning category include Transformer models, CNN models, and RNN models (Hatcher et al. 2018). Figure 2.1 illustrates the primary DL categories and examples of models within each category.

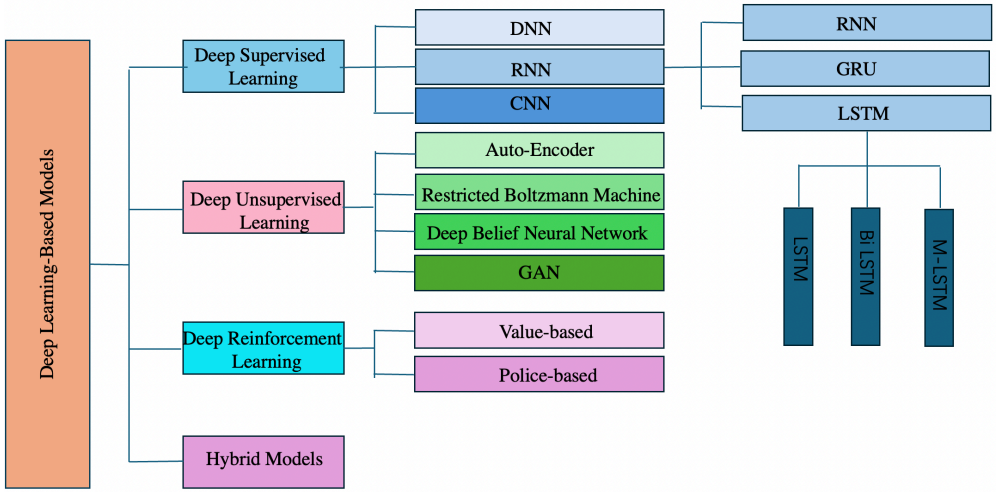



Figure 2.1 Schematic review of deep learning models

## 2.6 COMMON DEEP LEARNING MODELS

There are various DL model designs, each with its own advantages for different types of data and tasks. Here are some common deep learning models: CNN、RNN、Transformer.

### 2.6.1 CNN

CNN, a cornerstone in the realm of deep learning, adeptly captures essential spatial attributes and semantic links among data segments through the execution of convolution operations on multi-dimensional data. Designed to handle data that come in multiple array formats, CNNs are particularly skilled at processing color images, which consist of three 2D arrays depicting the intensity of pixels across three color channels. Data modalities that embody multi-array structures vary widely, ranging from 1D arrays used for signals and sequences like language, to 2D arrays for images or audio spectrograms, and extending to 3D arrays utilized in videos or volumetric imagery (Mikolov et al. 2011). The CNN model usually adopts a streamlined structure, as shown in Figure 2.2. In this structure, the feature maps are composed of k filters distributed across different channels. Furthermore, a pooling mechanism serves to shrink the dimensions of the feature map, whereas the convolutional strata create feature maps adept at encapsulating and detecting attributes through the application of filters on the input. After the convolutional layers, there is typically one or more fully connected layers, which connect to all neurons of the preceding layer. CNN often utilize pooling layers to

analyze latent patterns, achieving scale transformation and weight sharing, thereby reducing storage requirements, and optimizing the capture of semantic relevance (Tugrul et al. 2022).

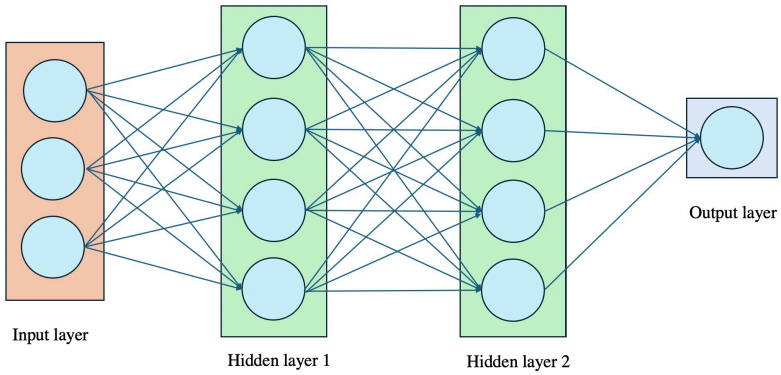



Figure 2.2 CNN architecture diagram

CNN have a wide range of applications across various fields due to their ability to effectively process and interpret the spatial and hierarchical patterns in data. Below, we'll discuss some key areas where CNN are prominently used:

**a. The Main Applications of CNN**

**i. The Applications of CNN in the Image Field**

In the 2014 ILSVRC-2012 image classification competition, the CNN model AlexNet proposed by Krizhevsky et al. (2014), achieved outstanding results, demonstrating a top-5 error rate of only 15.3%, in contrast to the second-ranked entry with an error rate of 26.2%. This represented nearly a halving of the error rate in image classification. During the same year, Branson et al. (2014) developed a novel approach for detecting parts and extracting CNN features across multiple areas of attitude standardization. Leveraging part annotation information, they acquired precise attitude standardization space. Furthermore, they devised a model that integrates pose standardization extraction from the low-level feature layer with unaligned image features from the high-level feature layer, thereby enhancing accuracy in image classification.

Bharati et al. (2020) employed a composite CNN framework to analyze a dataset of chest X-rays for detecting pulmonary diseases. In a related study, Dong et al. (2017) used CNNs to analyze over 16,000 patient X-ray images. This approach highlights how using CNNs can potentially accelerate progress in deep learning research. Rajkomar et

al. (2017) used a GoogLeNet CNN, training and testing with a dataset expanded from 1,850 chest X-rays to over 150,000 images. These images were reorganized into lateral and frontal views, achieving an accuracy rate of approximately 100% in their assessments.

Xiao et al. (2015) integrated visual focus mechanisms into CNN for nuanced categorization tasks. Their methodology encompasses a trio of attention strategies: initial attention guides the system to suggest potential patch candidates, while top-down attention at the object scale weeds out the pertinent segments associated with a specific item. At a more granular level, top-down attention zooms in on distinguishing features of the item. By synthesizing these selective attention processes, they cultivated specialized networks tailored to pinpoint objects or their components within the foreground, thereby isolating defining characteristics.

**ii. The Applications of CNN in the Vegetation Remote Sensing Field**

The utilization of CNN in the field of vegetation remote sensing primarily centers around deploying CNN architectures to examine and decipher satellite or aerial images. This is done to monitor, categorize, and evaluate various aspects of vegetation. Key applications include:

Nevavuori et al. (2019) developed a region-specific deep learning model for predicting crop yields, focusing on a unique profile of temperature and photoperiod. Terliksiz and Altýlar. (2019) underscored the paramount importance of judiciously choosing the most suitable data frame when employing three-dimensional Convolutional Neural Networks (3D-CNNs) for the purpose of predicting crop yields. The scholars meticulously delineated the impact that the data frame selection process has on the accuracy and efficacy of predictive models in vegetation remote sensing.

Yang et al. (2019) examined the transferability of CNN models over time for estimating rice grain yields. They assessed the effectiveness of a CNN that had been trained on one or several phenological stages in predicting unencountered phenological phases.

### iii. The Applications of CNN in the Smart Homes Field

CNN plays a vital role in advancing home automation and smart home technologies. The goal of smart homes is to increase convenience, security, energy efficiency, and overall comfort through technological integration. CNN are central to these systems as they analyze and interpret environmental sensory data. Here is a general overview of how CNN are implemented in the smart home sector:

Gochoo et al. (2019) used Aruba data with a non-invasive CNN activity recognition model to monitor a single elderly woman living alone over eight months, effectively categorizing her daily activities such as eating, moving from bed to toilet, meal preparation, and sleeping.

Arifoglu and Bouchachia (2019）assessed how different types and structures of convolutions (1D Convolution, 2D Convolution, and a combination of 2D CNN with LSTM) perform in identifying unusual behavior in individuals with dementia.

Bianchi et al. (2019) used a CNN model to categorize personalized human activities in a smart home environment, effectively classifying activities of the elderly.



## 2.6.2 RNN

RNN are a class of neural networks designed to handle sequential data, capturing temporal dynamic behavior which is essential for tasks involving sequences such as time series prediction, natural language processing, and speech recognition. The CNN model usually adopts a streamlined structure, as shown in Figure 2.3. Unlike feedforward neural networks, RNN has loops allowing information to persist, effectively enabling them to use their internal state (memory) to process sequences of inputs. Common issues like gradient vanishing and explosion are addressed by advanced variations such as LSTM and GRU.

RNN is particularly well-suited for handling sequence data, making them highly effective in applications that require the analysis and prediction of sequential and time-dependent data. Below are some of the key areas where RNNs are commonly applied:

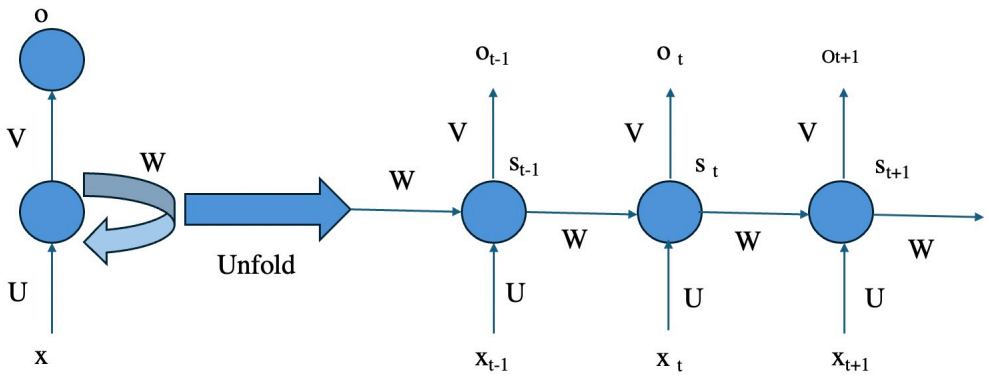



Figure 2.3 RNN architecture diagram

**a. The Main Applications of RNN**

**i. The Applications of RNN in the Electric Motor Fault Diagnosis Field**

RNN are utilized in the electric motor fault diagnosis sector because of their proficiency in handling sequential and time-series data. This capability makes them exceptionally adept at examining the dynamic behaviors and temporal patterns that are typical in motor operations. Here are specific applications of RNN in diagnosing faults in electric motors:

The motor defect detection method adopted by Luo et al. (2018) is based on the LSTM model, which leverages data from the prior sample, this method forecasts the three-phase current value of the next sample in real-time, thus facilitating ongoing monitoring of the motor. The outcomes showed that this approach successfully enabled accurate detection.

Zhang et al. (2017) employed LSTM as a classification tool to specifically identify three typical failures in wind turbine bearings. The results show that LSTM can show excellent stability and accuracy in fault diagnosis even if the fault characteristics vary very little.

Zhao et al. (2017) utilized empirical mode decomposition in combination with LSTM for the monitoring and prediction of rotating machinery. In comparison to SVRM, the result is that this model is more effective in handling parameter selection and demonstrated higher accuracy.

### ii. The Applications of RNN in the Microsystems Field

RNN has been widely used in the field of Microsystems because of its excellent performance in processing sequential and time series data. The following details the application of RNN in Microsystems:

Nguyen et al. (2018) developed a neural network model consisting of four LSTM units, by training an RNN on sampled sequences. This developed RNN model accurately and efficiently predicts the RX voltage, as demonstrated by highly accurate voltage prediction results.

### 2.6.3 Transformer

Transformer is a deep learning architecture that revolutionizes sequential tasks such as NLP and time series analysis. The architecture was first described by Vaswani et al. (2017) in their landmark paper attention is "Attention is All You Need". Transformer uses a method known as self-attention to evaluate the significance of each piece of provided information. Unlike RNN, transformer does not process data sequentially but in parallel, which greatly improves the efficiency of training. In addition, because it does not rely on cyclic connections, the common gradient disappearance problem of RNN is effectively avoided. The Transformer model usually adopts a streamlined structure, as shown in Figure 2.4



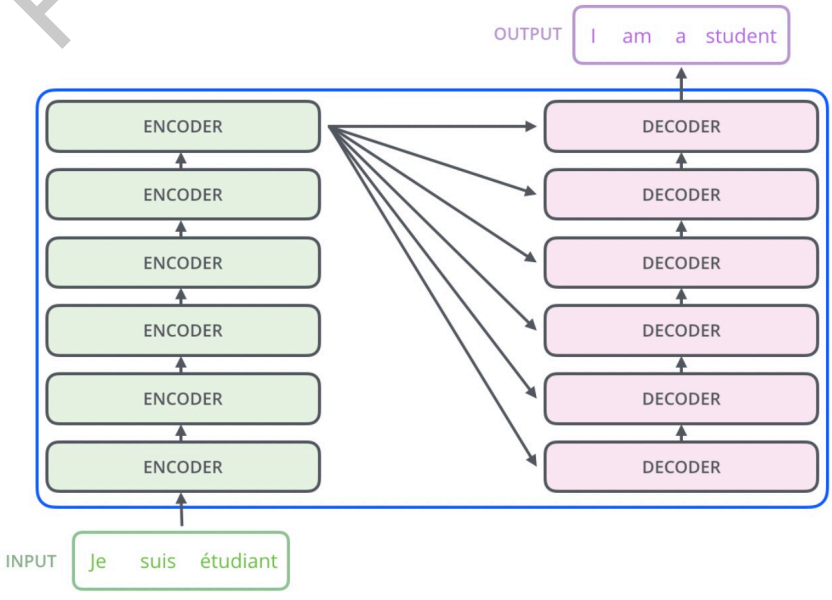



Figure 2.4 Transformer architecture diagram

### a. The Main Applications of Transformer

#### i. The Applications of Transformer in the Medical Image Processing

The Transformer model is deployed to dissect, comprehend, and manage medical imagery. Leveraging its prowess in sequential data management, this method is adept at pulling, tweaking, and scrutinizing visual data extracted from an array of medical imaging techniques, including X-rays, MRI scans, CT scans, among others.

Valanarasu et al. (2021) proposed a novel medical Transform, which incorporates an axial attention model with the gating mechanism. To effectively train the model on medical images, they devised a training strategy that integrates partial and global approaches, resulting in favorable experimental outcomes. In the same year, Ji et al. (2021) utilized MCTrans to construct a system compatible with network structures such as UNet, showcasing the exceptional properties of the model.

Xie et al. (2021) introduced the DeTrans model in their 2021 study, skillfully integrating a CNN with a transformer – dubbed CoTr – to enhance the precision of 3D medical image segmentation.

Yang et al. (2022) introduced the Sinogram Inner Structure Transformer (SIST), an advanced LDCT denoising network designed to minimize medical image noise by leveraging the inherent structure within the sinogram domain.



#### ii. The Applications of Transformer in the Bioinformatics Processing

Transformer has taken the bioinformatics world by storm, thanks to their sophisticated self-attention capabilities. Here's a glimpse into how these powerful tools are being utilized within the realm of bioinformatics:

Chen et al. (2021a) introduced an innovative hybrid approach that combines the transformer model, known for its prowess in handling sequential data, with the well-established U-Net architecture, a staple in the field of medical imaging segmentation. This novel integration marks a significant deviation from traditionally employed techniques in medical imaging segmentation, heralding a new era of precision and efficiency in processing medical images.

Tao et al. (2020) performed a comprehensive analysis employing the Genomic Impact Transformer to foresee numerous cancer phenotypes through the examination of somatic genomic variations. Their investigation delved deeply into the intricate links between genetic mutations and the emergence of different cancer types. By leveraging the transformer, the researchers uncovered patterns and connections that conventional techniques may have missed. The advanced analytical prowess of the Genomic Impact Transformer was pivotal in pinpointing essential genomic changes tied to specific cancer phenotypes. This innovation holds substantial promise for improving the precision of cancer diagnoses and crafting more effective treatment plans.

Ma et al. (2021) used the heterogeneous graph converter model to elucidate the complexity of single-cell biological networks that are unique to different cell types. In this way, they provide a sophisticated approach to understanding the unique interactions and regulatory mechanisms that control cell function. The model allows researchers to capture the complex and diverse relationships in cell networks, facilitating a more nuanced analysis of cell type-specific behavior.



Cao et al. (2021) introduced an advanced high-throughput protein function annotation tool based on a transformer architecture. This innovative tool not only demonstrated excellent accuracy in predictions, but also demonstrated a high degree of generalization across diverse protein datasets. By employing the technology of the transformer model, the annotator can effectively capture the complex relationships and patterns inherent in protein sequences. The researchers conducted extensive experiments to validate their approach, consistently achieving better performance metrics than existing methods. This advance holds great promise for accelerating the annotation process in proteomics, thereby contributing to deep biological insights, and advancing related fields such as drug discovery and molecular biology.

**iii. The Applications of Transformer in the Text Processing**

The applications of Transformer in various tasks related to text data include, but are not limited to, language translation, text summarization, sentiment analysis, and topic classification. Transformers are renowned for their ability to handle long-range dependencies within text, making them superior in understanding context and nuances in language compared to earlier models such as RNN and LSTM.

The groundbreaking BERT model, unveiled by Devlin et al. (2018) in their influential 2018 publication, continues to be a pivotal force in the realm of natural language processing (NLP). Representing a paradigm shift, BERT has revolutionized the approach to textual applications, facilitating a deeper and more sophisticated level of machine comprehension of human speech.

Raffle et al. (2019) unveiled a captivating study that delved into the realm of natural language processing. They introduced the innovative Transformer 5 model, a testament to the ingenuity of transfer learning. This model revolutionized the field by standardizing diverse text-based linguistic challenges into a singular text-to-text framework.

Liu et al. (2019) introduced the RoBERTa model, an enhanced iteration of BERT specifically optimized for more robust pre-training. This model was created to overcome the shortcomings in text processing that were present in the initial BERT model, which largely stemmed from considerable undertraining.



## 2.7 ADVANCEMENTS IN DEEP LEARNING-BASED ALGORITHMS FOR COVID-19 PATIENT MORTALITY PREDICTION

Deep learning has been widely used in several fields such as image recognition and natural language processing and has shown great potential in processing complex medical data. By applying deep learning techniques, researchers can develop predictive models that help physicians better assess patient conditions, optimize treatment plans, and ultimately improve the timeliness and accuracy of clinical interventions. Previous research has shown that deep learning has unique advantages in predicting medical outcomes. Deep learning plays an irreplaceable role in predicting death of COVID-19 patients. Currently, the dominant deep learning methods for predicting the mortality of COVID-19 patients include CNN、RNN (Kamalov et al. 2022).

### 2.7.1 CNN For Predicting COVID-19

CNN has demonstrated a remarkable ability to explore the multi-layered nature of data, especially in predicting the risk of death in COVID-19 patients, and its effectiveness has been clearly demonstrated. Specifically, a typical CNN architecture for processing COVID-19 patient data is shown in the figure. It involves inputting COVID-19 case

data collected over some time into CNN. It learns the rules contained in the sequence data through multiple one-dimensional sliding Windows, to train an efficient network. After the data is processed by the convolution layer, it is transformed into tiled one-dimensional dense vectors, which are used as the basis for building predictive models.

Now, CNN has been widely used in several research fields related to COVID-19 prediction and has achieved positive results. For example, Hindawi et al. (2021) used a deep learning model to analyze patients' clinical data to predict the mortality risk of COVID-19 patients, showing higher predictive accuracy than traditional statistical methods. Additionally, Ozturk et al. (2020) utilized DarkCovidNet to analyze chest X-ray images of COVID-19 patients, successfully predicting the severity of the condition, and the accuracy rate of the model reached 87.02 %. Hemdan et al. (2020) developed a model named COVIDX-Net, integrating data from over 50 patients to predict the mortality risk of COVID-19 patients. The study highlighted the predictive value of age, underlying diseases, and certain laboratory indicators such as white blood cell count and liver function tests.



However, some comparative studies have found that CNN does not deliver the best predictive outcomes, falling short of models such as LSTM. Dairi et al. (2021) compared the performance of LSTM and CNN in predicting death in patients with COVID-19. The results show that the prediction effect of CNN is inferior to that of LSTM.

### 2.7.2 RNN For Predicting COVID-19

RNN is a specially designed type of neural network created to handle sequential data, effectively grasping the temporal patterns in the sequences, a crucial characteristic for tasks involving sequences, such as predicting COVID-19 patient mortality. The typical architecture of an RNN-based COVID-19 prediction model commences at the input layer, which is responsible for receiving previously observed data over a defined period (5-7 days). The input data undergoes processing by multiple RNN units before being forwarded to a dense layer. Ultimately, the activation output of this dense layer is utilized for predicting the number of COVID-19 cases. Figure 2.1 outlines various extensions to the fundamental RNN architecture, encompassing subgroups such as regular RNN, GRU, and LSTM.

**a. Plain RNN For Predicting**

RNN shows its professional performance in the analysis of time series data, and it has a good adaptability to short-term trend prediction. However, in the novel coronavirus pneumonia (COVID-19) prediction task, the traditional RNN model faces application limitations. In literature, RNN was included in a comparative study, and its performance was not found to be superior (Zeroual et al. 2020). It is worth noting that even for comparative analysis purposes, it is not reasonable to ignore the study of RNN. Although the application of the standard RNN model is limited, its derived model has been applied in the prediction of COVID-19 outbreak.

For example, Alassafi et al. (2021) used recurrent neural networks (RNN) and long short-term memory (LSTM) networks to predict the likely number of COVID-19 deaths. The results show that the prediction accuracy of RNN model and LSTM model is 98.58% and 93.45% respectively. By comparison, the prediction effect of RNN model is better than that of LSTM model. Niu et al. (2021) developed an RNN that integrated both spatial and temporal data, achieving superior performance compared to other models such as GRU and SEIR.



**b. LSTM For Predicting**

As an improved type of RNN, LSTM is designed to overcome the problem of gradient disappearance or explosion that may be encountered in the training process of traditional neural networks. This design idea was first proposed by scholars such as Hochreiter et al. (1997). In addition to the standard LSTM, several derivative versions of LSTM, such as BiLSTM, M-LSTM and ConvLSTM, have been developed to meet the special needs of COVID-19 epidemic prediction (Absar et al. 2022). Figure 2.1 clearly shows the extended versions of these LSTM.

In the application of COVID-19 outbreak prediction, LSTM-based models have become a widely adopted mainstream approach. Some studies have used it as the main forecasting model, and positive results have been achieved in practice (Dairi et al.2021; Devaraj et al. 2021).

Aldhyani et al. (2021) have achieved a significant breakthrough with their BiLSTM model in accurately predicting COVID-19 patient fatalities in Gulf nations.

Furthermore, this advanced LSTM model has demonstrated superior performance compared to the standard LSTM, yielding more accurate forecasting results.

Ayoobi et al. (2021) conducted a comparative analysis of time-series predictions for newly reported COVID-19 mortality rates, employing models such as BiLSTM, LSTM, and ConvLSTM. The results revealed that both the BiLSTM and ConvLSTM models demonstrated lower prediction errors in comparison to the traditional LSTM model.

However, it is important to note that LSTM and its derivative versions do not guarantee the best predictions in all cases. Studies have shown that LSTM and its variants do not perform as well as other deep learning models in some situations (Kapoor et al. 2020; Nabi et al. 2021).

**c. GRU For Predicting**

GRU is an enhanced version of the foundational RNN, equipped with a so-called forget gate, a concept first introduced by Cho et al. (2014). Designed to tackle the challenging problem of vanishing and exploding gradients that the standard RNN can encounter, the GRU's architecture proves to be particularly effective for forecasting the mortality of individuals with COVID-19, it's pivotal to employ reliable predictive measures. It adeptly marries the unpretentious nature of the basic RNN with the sophisticated gradient moderation found in the LSTM.

Shahid et al. (2020) utilized both the LSTM and GRU models in predicting the mortality rate of COVID-19 patients and found that the GRU model outperformed the LSTM.

On the other hand, in several comparative studies, it did not achieve optimal performance (Zeroual et al. 2020; Nabi et al. 2021).

## 2.8 THIS PAPER RELIES ON THE DATASET OF PREVIOUS RESEARCH

The COVID-19 dataset published by Meir Nizri has become an important source for research into the transmission, diagnosis, and prediction of the disease(COVID-19 Dataset. 2020). The dataset includes information on the demographic, medical-related and disease-related characteristics of COVID-19 patients, providing a rich and valuable

data base for researchers. Several key studies and their academic contributions are outlined below:

Unal and Dudak (2020) used a variety of machine learning classification methods to carry out research, and the research results showed that SVM algorithm performed well in the classification task under investigation, and its classification accuracy reached 100% of the ideal standard.

Ayah et al. (2021) used multiple algorithms such as NB, CART and KNN to conduct in-depth analysis of Covid-19 data sets. The aim is to build a mortality prediction framework based on the health data of COVID-19 patients and other relevant factors, to predict the life prognosis of patients, that is, the probability of death or survival. The results of the study validated the significant validity of the established model in predicting death and death of Covid-19 patients based on their health status.

Ahmed et al. (2022) adopted a variety of machine learning methods, including LR, DT, LR, SVM and KNN, and conducted an in-depth discussion on the possibility of chronic diseases increasing the severity of COVID-19 patients' diseases. The research results show that the above algorithm has shown high efficiency in accurately classifying COVID-19 cases according to the chronic disease symptoms of patients, with the classification accuracy of 88 %, 88 %, 87 %, 86 % and 88 % in order, thus verifying the application value of machine learning technology in this dataset.

Ramdan et al. (2024) utilized K-Medoid cluster analysis to conduct a comprehensive examination of the clinical characteristics of COVID-19 infected patients. The findings from the study indicate that non-obese individuals have a higher likelihood of infection based on their obesity status distribution. Further investigation revealed that among obese patients, coexisting pneumonia and hypertension tended to exacerbate the disease, whereas in the non-obese group, hypertension played a more significant role in disease progression. The presence of both obesity and cardiovascular disease demonstrated further implications for gender and pneumonia's influence on the disease course. Therefore, obesity and its related complications significantly contribute to the development of novel coronavirus pneumonia as an influential factor that cannot be overlooked.

Extensive research on this dataset reveals its application prospects in many fields, which provides important data support for the related prediction of COVID-19 patients and makes significant contributions to the research of this paper.

## 2.9 DISCUSSION

Deep learning technology has played a crucial role in predicting death in patients with COVID-19, but it still faces several challenges in practical application. The previous research results show that there are some problems in the construction of prediction model, such as accuracy, prediction precision and high complexity of data integration. Although CNN overcomes the problems of data complexity and multi-modal data integration, it is difficult to process time series data. LSTM and GRU have advantages in the processing of time series data, which can enhance the model's learning of time-dependent features and improve the accuracy of prediction, but they are unable to handle multidimensional data well. In view of this, there is an urgent need to develop new deep learning models, which not only improve the comprehensive performance of predicting the risk of death in COVID-19 patients, but also show higher effectiveness in actual clinical application scenarios. The new model is designed to meet the needs of big data processing and provide solid technical support for dealing with similar public health events that may occur in the future.

## 2.10 CHAPTER SUMMARY

This chapter presents a comprehensive survey of studies related to forecasting the mortality of COVID-19 patients by leveraging the capabilities of deep learning. It reviews recent work related to the issues and objectives mentioned in the previous chapter. It begins with an overview of COVID-19 and its current detection status. Next, it outlines the prediction of COVID-19 patient mortality, describes the deep learning methods used for this prediction, and introduces CNN, RNN, and DNN algorithms. Then, it describes the current state of research in deep learning-based COVID-19 patients mortality prediction. Finally, it discusses the limitations found in the literature and proposes innovative methods to address this issue.

# CHAPTER III

# METHODOLOGY

## 3.1 INTRODUCTION

This chapter provides methodological details of this paper. The aim of the COVID-19 Patient Mortality Prediction study is to identify and predict the likely risk of death of people infected with COVID-19, which is essential for timely medical intervention and resource allocation. This chapter will discuss in depth the research basis of COVID-19 death prediction, including data sets, basic introduction to deep learning, principles of common algorithms, and performance evaluation criteria of related prediction algorithms. The establishment of these theoretical and technical frameworks will provide a solid theoretical and practical foundation for subsequent algorithm optimization, model validation and clinical application, and help improve the accuracy of epidemic response.



The second section of the chapter clarifies in depth the level of detail of the study structure, and clearly points out the prediction of COVID-19 patient death that is intended to be solved by deep learning methods. At the same time, through logical reasoning and clear research path diagram, the internal relations and dependencies among process nodes such as problem definition, data preparation, algorithm selection and performance evaluation are revealed to improve the accuracy of the prediction model. In addition, the role of cross-validation and model regularization strategies in preventing overfitting and improving model generalization ability is also discussed.

The third section of this chapter shows the construction process of different deep learning models in terms of research design. This section examines in detail the potential effects of different network architectures, activation function selection, and hyperparameter configuration on the prediction results, emphasizing that it is critical to

consider the balance between model generalization and computational efficiency during the model design phase.

In section 4, 5 and 6 of this chapter, the author comprehensively considers the characteristics of multi-source, dynamic and timeliness of data, and explains the technical means of data collection, pre-processing, and enhancement in detail. The problems of missing value processing, outlier detection and sample imbalance are comprehensively considered, and adopting suitable data engineering strategies enhances the quality of the input data, and data segmentation is performed to create a robust data foundation that facilitates the formation of superior predictive models.

In Section 7, the basic knowledge of deep learning is systematically reviewed, and some mainstream algorithms applied in this research are introduced, such as CNN, (RNN, and the recently emerging attention mechanism and Transformer network. The advantages and limitations of these algorithms in processing time series data, image data and sequence data are introduced respectively, which provides reference for selecting suitable algorithms in the future.

In section 8, the structure and algorithm theory of CNN, LSTM and GRU are introduced in detail.

In section 9, the performance evaluation criteria are discussed in detail, including common evaluation indexes such as ACC, SEN, F1, etc., and an evaluation method considering the confidence and uncertainty of the model prediction results is also proposed.

In summary, Section 10 summarizes the main contents of the methodology chapter, emphasizes the important impact of deep learning on the algorithm prediction, and clarifies the key role of the quality of data preprocessing on the model performance. Through these detailed method description and theoretical analysis, this chapter has laid a solid theoretical and technical foundation for the specific application, effect evaluation and result discussion of the model in the following chapters.

## 3.2 RESEARCH STRUCTURE

It is crucial to carefully consider task requirements, data characteristics, and available

resources when designing and optimizing four models. A robust approach starts with a relatively minimalist model architecture, followed by gradually introducing more complexity based on the model's performance during the training process. This strategy not only helps to effectively utilize resources, but also prevents the model from overfitting training data, thereby improving its generalization ability on unseen data. Monitoring the changes in the loss function during the training and validation stages, as well as tracking relevant evaluation indicators, are key to measuring the effectiveness of model improvement and making timely adjustments.

Hyperparameter tuning is an indispensable part of improving model performance, where cross validation can provide reliable estimates of model generalization ability. It should be emphasized that suitable parameters like Hidden Size and Num Layers largely hinge on the traits of the dataset, including sequence length and input feature dimensions. Long sequences or high-dimensional feature spaces may require larger model capacity, while limited computing resources may prompt us to seek more efficient model design.



Specifically, for LSTM and GRU, the management and gating mechanisms of hidden states make them adept at capturing long-term dependencies, adjusting the size and number of hidden layers of these models, which directly affects their memory ability and learning of complex sequence patterns. 1D-CNN captures features on sequence data through local connections and pooling operations, and the selection of the number, size, and convolution layers of its filters is directly related to the diversity and depth of the model's recognition features.

In summary, the progression and enhancement of models such as 1D-CNN, GRU, LSTM, or 1D-CNN-LSTM consist of a revolving process of fine-tuning and perfecting, all reliant on the responses derived from trials and experiments. Based on initial settings and combined with performance indicators observed in experiments, gradually iterate the model structure and parameter configuration to achieve optimal performance. This iterative optimization strategy requires a deep understanding of the principles of the model, while flexibly utilizing various tuning techniques and tools to ensure that the model serves specific analysis or prediction tasks efficiently and accurately. The main adjustment parameters are as follows:

1. Input Size

Input Size, the feature dimension of the input tensor represents the number of features in the input data for each time step. In natural language processing, if each word is encoded as a fixed length vector (for example, a vector obtained through word embedding), then Input Size is the dimension of this vector. For example, if a 100-dimensional word embedding is used, then the Input Size is 100. This parameter determines the input complexity that the model can handle and is the primary consideration when designing LSTM and GRU models.

2. Hidden Size

The hidden size specifies the feature dimension of the LSTM unit's output, which is also referred to as the hidden state's dimension. It represents the internal state size of the model, influencing the complexity of the representations the model can learn. A larger Hidden Size can increase the model's expressive power, but it also means more computing resources and training time. The hidden state is transmitted at each time step of the time series, storing historical information about past inputs. Choosing the appropriate Hidden Size is crucial for model performance and usually requires experimental tuning.



3. Num Layers

Num Layers denotes the count of LSTM layers piled within a model, with each layer consisting of LSTM units. Multilayer LSTMs are adept at grasping more intricate levels of features, thereby facilitating the capture of more abstract, high-level information. Theoretically, adding layers can amplify the model's capability to express complex patterns, yet it simultaneously escalates the model’s complexity and the intricacies involved in training. This increase might precipitate problems such as overfitting or the disappearance of gradients. Thus, determining the optimal layer count involves striking a delicate balance between the model's complexity and the dataset's demands.

4. Bias

The Bias term is a Boolean parameter that determines whether bias is included in the LSTM unit. The existence of bias can increase the flexibility of the model and help it

better fit the data. In general, the default setting is True, which includes bias terms.

5. Batch First

Batch First is a Boolean parameter that determines the dimensional arrangement of the input and output tensors. If set to True, the batch size of the input and output will become the first dimension, with a shape of [batch_size], seq_len, feature]； Otherwise, the batch dimension is the second dimension and the shape is [seq_len], batch_size, feature]。 Set this parameter based on the convenience of data processing and the requirements of the framework.

6. Dropout

Dropout serves as a regularization strategy designed to curb overfitting. Within the LSTM framework, the dropout rate delineates the proportion of dropout applied across each LSTM output layer, barring the final one.

7. Regularization Parameters

The weight of regularization terms (such as L2 regularization): used to control the complexity of the model and prevent overfitting.

8. Optimizer Parameters

Learning rate: Control the step size of the optimizer to update parameters, which requires tuning to ensure convergence speed and performance.

9. Loss Function

Choose the appropriate loss function based on the task type, such as mean squared error (MSE) for regression tasks and cross entropy for classification tasks.

10. Training batch size and number of training iterations

a. Batch Size

Regulate the number of samples processed in each iteration, influencing both the stability and pace of gradient descent.

b. Training Iterations

Determine the number of rounds in the training process, usually using cross validation or early stopping methods to determine the optimal number of iterations.

## 3.3 RESEARCH DESIGN

### 3.3.1 1D-CNN Model Design

1D-CNN is a specialized deep learning framework crafted for the analysis of linear data sequences. It finds extensive applications in various domains, including speech recognition, time series forecasting, and the examination of bioinformatics sequences. Key considerations when architecting a 1D-CNN model include the following parameters and procedural elements:

1. Input Shape

The input of 1D-CNN is one-dimensional sequence data, whose shape is usually batch_size, sequence_length, channels). Among them, sequence_length represents the length of the sequence (for example, the number of sampling points for audio segments, the number of observation points for time series data), channels refer to the number of channels in the data. For single channel signals (such as pure audio, single variable time series), channels=1; for multi-channel signals (such as multi-sensor data, multiple frequency bands of audio), channels >1.

2. Convolutional Layer

a. Filters/Output Channels

The number of filters in each convolutional layer determines the depth of the output feature map, which is the type of feature learned by the model. Increasing the number of filters can enhance the model's expressive power, but it also increases computational complexity.

b. Kernel Size

The width of the filter's sliding on the sequence determines the range of local features that the model can capture. Common choices include 3, 5, 7, etc. A larger filter size can

capture a wider range of features but may lose some detailed information.

c. Stride

The stride with which the filter advances influences both the resolution of the resulting feature map and the model's receptive field. Increasing the stride can decrease computational load, but it risks missing detailed information in the sequence.

d. Padding

Adding zeros at the edges of the sequence to maintain the size of the output feature map or to control the degree of reduction. "SAME" padding makes the output size the same as the input (under certain conditions), while "VALID" padding does not add additional zeros, allowing the output size to decrease.

3. Pooling Layer

Pooling types mean that commonly used one-dimensional pooling types include Max Pooling and Average Pooling. Through the process of pooling, the dimensional scope of feature maps is condensed, computational intricacy is lessened, and the preservation of pivotal information is ensured.

Pooling size and step size: Like filter size and step size, it determines the range of pooling operations and the size of output feature maps.

4. Fully Connected Layer

After the convolutional and pooling layers, one or more fully connected layers are usually added to integrate global features and make final classification or regression predictions. The number of neurons in the fully connected layer can be adjusted based on the complexity of the task and the expected output.

### 3.3.2 LSTM Model Design

The LSTM model proficiently handles data sequences, and tuning a multitude of parameters is crucial to enhance its performance. The construction of the LSTM model discussed herein involved specific design and parameter optimization strategies. It includes:

1. Characteristics of Input Data

When designing an LSTM model, the first thing to consider is the characteristics of the input data, including sequence length and feature dimension. The death prediction model studied in this article is independent of time sequence. The time step is set to 1, and the input data is in the form of data volume, time step, and feature number.

2. Parameters of LSTM Layer

The LSTM layer is the core part of the LSTM model, which includes the dimensions of hidden states (units) and the selection of activation functions. The size of the hidden state dictates the neuron count in an LSTM unit, a decision often influenced by the task's complexity and data attributes. Typically, the ReLU activation function is favored to enhance the model's capacity for handling non-linearities.

3. Dropout Parameter

To mitigate the risk of overfitting, incorporating a Dropout layer after the LSTM layer proves effective. By fine-tuning the Dropout rate, one can regulate the likelihood of neuron exclusion at every temporal increment. Usually, a suitable value can be selected between 0.1 and 0.5, and then optimized through cross validation.



4. Optimizer Parameters

The selection of optimizers and the setting of learning rates have a significant impact on the training and performance of the model. Using the Adam optimizer, the learning rate is usually set from 0.001 and then optimized through experiments and cross validation.

5. Loss Function

The choice of loss function directly affects the training effectiveness and performance of the model. For this study, the mean squared error loss function is chosen.

6. Training Batch Size and Number of Training Iterations

Batch size and training iteration times are also important parameters that need to be adjusted. The batch size affects the stability and speed of gradient descent. The number

of training iterations depends on the complexity of the data and the performance requirements of the model, and the optimal number of iterations can be determined through cross validation.

### 3.3.3 GRU MODEL DESIGN

The GRU model is a variant of RNN, proposed by Cho et al. (2014), aimed at solving the problems of gradient vanishing and exploding that traditional RNNs face when processing long sequence data. The GRU model, through a streamlined structural design, has lower computational complexity and faster training speed compared to the LSTM model in certain tasks, while still effectively capturing long-term dependencies in the sequence. When designing a GRU model, the following parameters and considerations are crucial:

1. Input Shape

The GRU model receives one-dimensional or three-dimensional input data, and for sequential data, its shape is usually batch_size, sequence_length, input_dim)。 Among them, sequence_length is the length of the sequence, and input_dim is the feature dimension of each time step.

2. Hidden Layer Size

This is the dimension of the internal state of the GRU unit, which also determines the complexity of the representations that the model can learn. A larger hidden size means that the model can learn more complex features, but it also requires more computing resources and training time. Selecting the right hidden size necessitates striking a balance between the model's complexity and the dataset's specific features.

3. Number of Layers

Each additional layer theoretically enhances the model's expressive power. However, it could also lead to potential overfitting and vanishing gradients. The selection of layers should be contingent upon the complexity of the given task.

4. Dropout

To prevent overfitting, Layers that essentially drop out can be incorporated following

each GRU's hidden layer to randomly "eliminate" a fraction of the neuron's output. The Dropout ratio is usually between 0 and 0.5.

### 3.3.4 Research Model Choice Reasons

This paper adopts 1D-CNN, LSTM, GRU, 1D-CNN-LSTM models. These models work well with time series data and multi-dimensional features, and they were chosen for the following reasons:

**a. 1D-CNN**

1D-CNN has an excellent role in time series data processing. Due to its powerful ability to capture local information, 1D-CNN has shown significant advantages in the local feature extraction and pattern recognition of physiological data of COVID-19 patients, such as patient clinical indicators such as body temperature and blood oxygen saturation, which evolve over time. Through efficient convolutional operations, 1D-CNN was able to effectively identify key patterns associated with the risk of death in COVID-19 patients. Chaddad and Tanougast (2022) used 1D-CNN to predict the survival of COVID-19 patients, and the accuracy of the model ranged from 83.39% to 84.47%, far exceeding the performance of other models in existing studies. In addition, 1D-CNN has higher parallel computing efficiency when dealing with large data sets. 1D-CNN was chosen for the experiment.



**b. LSTM and GRU**

With its unique complex gating mechanism, LSTM demonstrates a distinct advantage in addressing and predicting long-term dependency issues in time series data. In the context of clinical data for patients infected with the novel coronavirus, LSTM can effectively capture the intricate patterns of patients' health evolution over time and extrapolate disease progression trends, enabling accurate predictions of patient mortality. GRU is a simplified version of LSTM with similar performance but more computationally efficient. GRU can achieve similar results to LSTM in a shorter training time and is especially suitable for application scenarios with limited computing resources. Fernandes et al. (2022) utilized the LSTM model to forecast COVID-19 patient mortality rate, achieving an outstanding result with a coefficient of determination ($R^2$) as high as 0.9656. This underscores its superior predictive capability

compared to other composite evaluation models and significantly enhances prediction accuracy. Therefore, this study aims to use LSTM and GRU models to prospectively predict and analyze the risk of death in patients with novel coronavirus pneumonia.

**c. 1D-CNN-LSTM**

The 1D-CNN-LSTM hybrid model combines the advantages of spatio-temporal series mining, which can not only deal with high-dimensional features efficiently, but also grasp the long-term dependence of time series. Among them, 1D-CNN is responsible for extracting key local features from the time series, while LSTM focuses on mining long-term dynamic relationships within the time series. By combining the characteristics of these two structures, the model performs better in solving complex spatiotemporal data analysis problems. Yu et al. (2021) applied the combined model of CNN and LSTM to process the data of COVID-19 patients in their study and confirmed that the model can effectively improve the accuracy and robustness of the prediction of patient death risk when integrating different types of data information. In view of this, this study intends to adopt the 1D-CNN-LSTM model for experimental investigation.

### 3.3.5 1D-CNN-LSTM Model Design

The 1D-CNN-LSTM model is a hybrid model that combines 1D-CNN and LSTM, aiming to efficiently process one-dimensional sequence data. The design of the hybrid model is based on references from other literature（Yu et al. 2021). This combination leverages the advantages of 1D-CNN in capturing local features and LSTM's ability to handle long-term dependencies. The structure of the 1D-CNN-LSTM model is shown in Figure 3.1. Table 3.1 shows the parameters of each layer for the four models.

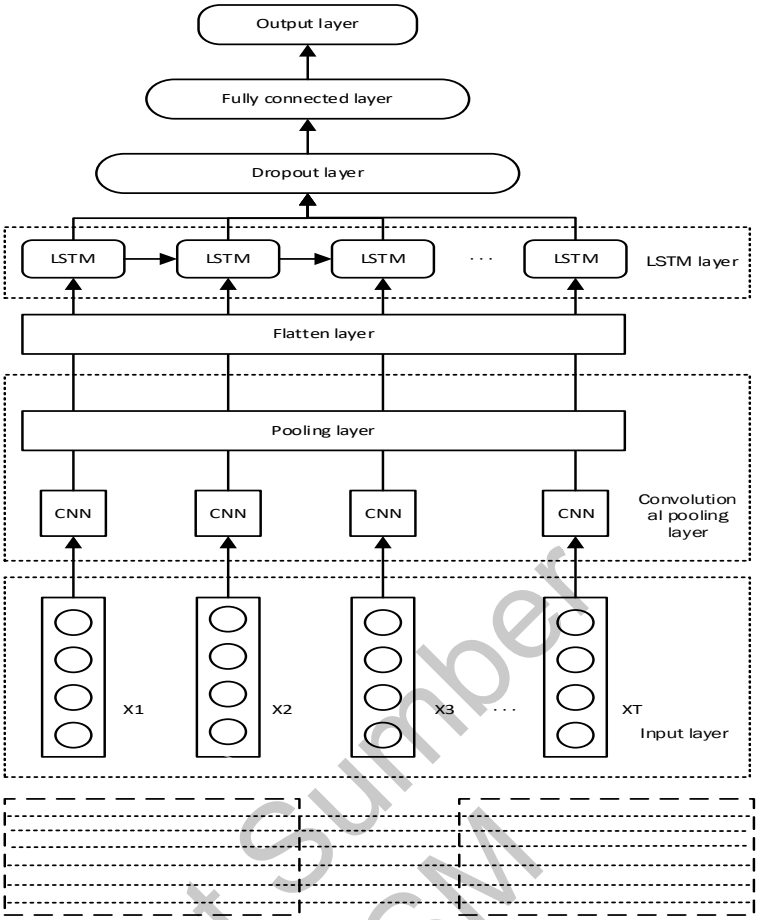



Figure 3.1 1D-CNN-LSTM model structure

Table 3.1 Parameter settings for each layer of the model

| Model | 1D-CNN | LSTM | GRU | 1D-CNN-LSTM |
|---|---|---|---|---|
| Input Layer | Features: 21<br>Sequence length: 1 | Features: 21<br>Sequence length: 1 | Features: 21<br>Sequence length: 1 | Features: 21<br>Sequence length: 1 |
| Convolutional Layer | Number of units: 128<br>Convolutional kernel size: 3 | —— | —— | Number of units: 128<br>Convolutional kernel size: 3 |
| Pooling Layer | Maximum Pooling<br>Step size: 2 | —— | —— | Maximum Pooling<br>Step size: 2 |
| Flatten Layer | —— | —— | —— | Need |

to be continued ...

... continued.

| | | | | |
|---|---|---|---|---|
| LSTM Layer | —— | Number of units: 128 | —— | Number of units: 128 |
| GRU Layer | —— | —— | Number of units: 128 | —— |
| Dropout Layer | Ratio: 50% | Ratio: 50% | Ratio: 50% | Ratio: 50% |
| Fully Connected Layer | Number of units: 100 Activation function: ReLU | Number of units: 100 Activation function: ReLU | Number of units: 100 Activation function: ReLU | Number of units: 100 Activation function: ReLU |
| Output Layer | Number of units: 2 Activation function: Softmax | Number of units: 2 Activation function: Softmax | Number of units: 2 Activation function: Softmax | Number of units: 2 Activation function: Softmax |



The following is a detailed explanation of each model design and parameter selection:

1. Input Layer

To ensure consistent variable control, four models were utilized in this study. The input features and sequence length used in these models were standardized at 21 features and a sequence length of 1. This approach guarantees that all 21 features in the input COVID-19 dataset are accounted for. Additionally, as each record in the dataset represents individual patient data, a sequence length of 1 was chosen for modeling. Consequently, the four models make independent predictions at each time point without considering historical data.

2. Convolutional Layer

Cross-validation experiments demonstrate that in both 1D-CNN and 1D-CNN-LSTM models, optimal model performance is achieved when utilizing 128 units and 3 convolution kernels in the convolution layer. In contrast, the LSTM and GRU models

lack a convolutional layer. This parameter setting does not impact the LSTM model or the GRU model, and it is beneficial for controlling variables.

3. Pooling Layer

The maximum pooling layer is used to preserve salient features, which means preserving the most critical indicator values when working with covid-19 datasets. After cross-verification, the results show that the model effect can be maximized when the step size is 2. In contrast, the LSTM and GRU models lack a pooling layer. This parameter setting does not impact the LSTM model or the GRU model, and it is beneficial for controlling variables.

4. Flatten Layer

Only the 1D-CNN-LSTM model contains a flat layer that converts the multidimensional data output from the convolutional layer into one-dimensional data for easy input into subsequent LSTM layers.

5. LSTM Layer and GRU Layer

The LSTM model and the 1D-CNN-LSTM model contain LSTM layer, each configured with 128 units, which show excellent performance.

The GRU model contains a GRU layer, configured with 128 units. GRU, as a variant of LSTM, can achieve similar performance while reducing computational complexity.

6. Dropout Layer

During this study, the author established a strategy of fixing the Dropout ratio at 50%, which is designed to both prevent model overfitting and enhance the performance of the four models. The choice of 50% Dropout ratio is based on years of practice and rules of thumb in the deep learning field. When Hinton et al. (2014) introduced the Dropout method, they pointed out that for many application scenarios, a ratio of 50% Dropout can show excellent performance.

7. Fully Connected Layer

This study opted for a configuration comprising 100 neuron nodes and employed the ReLU activation function. Through preliminary experiments, the study compared and analyzed the impact of varying numbers of neuron nodes (e.g., 64, 100, 128) on model performance. The results indicated that the model with 100 neuron nodes exhibited superior performance on the verification set. Furthermore, the utilization of ReLU activation function addressed gradient vanishing issues and enhanced model training efficiency. Additionally, due to its ability to realize non-linear expressions, the ReLU activation function demonstrated exceptional performance when processing the covid-19 dataset selected in this study.

8. Output Layer

Predicting the risk of mortality in COVID-19 patients entails a binary classification task, necessitating an output layer with two neural units corresponding to the prediction outcomes of "survival" and "death" respectively. The Softmax activation function was selected to transform the original values of the model's output layer into a probability distribution, where each cell's output values represent the likelihood that the patient belongs to the corresponding category. The use of Softmax ensures that the sum of probabilities for all outputs is 1, providing a clear interpretation of probabilities and facilitating analysis and understanding of predicted outputs from the four models. Furthermore, Softmax performs effectively across diverse classification tasks and efficiently adjusts model parameters to optimize performance. Consequently, the output layer employs two neural units paired with Softmax activation functions.



### 3.3.6 Hyperparameter Optimization

After setting the parameters in the previous section, ensure that the above parameters have the same impact on the four model algorithms. In this study, hyperparameter optimization was performed by changing the network layer and the number of training epochs.

**a. Network Layers**

Augmenting the educational process of four unique models is successfully accomplished through the expansion of network layers, enabling the assimilation of

increasingly intricate features and patterns. However, an excessive number of layers could lead to overfitting. Strategically adjusting the quantity and sequence of convolutional, pooling, LSTM, and GRU layers permits the nuanced extraction of COVID-19 patient characteristics, thereby elevating the predictive precision of each model.

**b. Training Epochs**

Regarding training epochs, expanding the number of iterations enables the model to learn more thoroughly from the training data, leading to better convergence. Yet, too many epochs risk causing overfitting. Selecting an optimal number of training epochs ensures the model performs best on the COVID-19 validation set, which in turn, bolsters its generalizability.

This investigation seeks to enhance the efficiency in both training and prediction of the quartet of models by adjusting the counts of network layers and the duration of training epochs. Identifying the most effective model configuration enhances the accuracy of predicting COVID-19 patient mortality. The best training epochs and network configurations are then chosen based on performance on the validation set, guarding against models that perform well on training data but falter with new data.



## 3.4 DATASET INTRODUCTION

The dataset used in this article mainly comes from Kaggle (https://www.kaggle.com/datasets/meirnizri/covid19-dataset) . The COVID-19 dataset is a rich resource for researchers and data scientists, providing multiple aspects of COVID-19 related data for analysis and modeling. We can see a statistical description of the COVID-19 dataset in Figure 3.2. In the Boolean features, 1 means "yes" and 2 means "no". values as 97, 98 and 99 are missing data.

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| USMER | 1048575.0 | 1.632194 | 0.482208 | 1.0 | 1.0 | 2.0 | 2.0 | 2.0 |
| MEDICAL_UNIT | 1048575.0 | 8.980565 | 3.723278 | 1.0 | 4.0 | 12.0 | 12.0 | 13.0 |
| SEX | 1048575.0 | 1.499259 | 0.500000 | 1.0 | 1.0 | 1.0 | 2.0 | 2.0 |
| PATIENT_TYPE | 1048575.0 | 1.190765 | 0.392904 | 1.0 | 1.0 | 1.0 | 1.0 | 2.0 |
| INTUBED | 1048575.0 | 79.522875 | 36.868886 | 1.0 | 97.0 | 97.0 | 97.0 | 99.0 |
| PNEUMONIA | 1048575.0 | 3.346831 | 11.912881 | 1.0 | 2.0 | 2.0 | 2.0 | 99.0 |
| AGE | 1048575.0 | 41.794102 | 16.907389 | 0.0 | 30.0 | 40.0 | 53.0 | 121.0 |
| PREGNANT | 1048575.0 | 49.765585 | 47.510733 | 1.0 | 2.0 | 97.0 | 97.0 | 98.0 |
| DIABETES | 1048575.0 | 2.186404 | 5.424242 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| COPD | 1048575.0 | 2.260569 | 5.132258 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| ASTHMA | 1048575.0 | 2.242626 | 5.114089 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| INMSUPR | 1048575.0 | 2.298132 | 5.462843 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| HIPERTENSION | 1048575.0 | 2.128989 | 5.236397 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| OTHER_DISEASE | 1048575.0 | 2.435143 | 6.646676 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| CARDIOVASCULAR | 1048575.0 | 2.261810 | 5.194850 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| OBESITY | 1048575.0 | 2.125176 | 5.175445 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| RENAL_CHRONIC | 1048575.0 | 2.257180 | 5.135354 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| TOBACCO | 1048575.0 | 2.214333 | 5.323097 | 1.0 | 2.0 | 2.0 | 2.0 | 98.0 |
| CLASIFFICATION_FINAL | 1048575.0 | 5.305653 | 1.881165 | 1.0 | 3.0 | 6.0 | 7.0 | 7.0 |
| ICU | 1048575.0 | 79.553974 | 36.823073 | 1.0 | 97.0 | 97.0 | 97.0 | 99.0 |

Figure 3.2 Statistical description of COVID-19 dataset

Table 3.2 Description of the meaning of each feature in the COVID-19 dataset

| **Column names** | **Meaning** |
|---|---|
| USMER | The code representing the medical unit |
| MEDICAL_UNIT | Indicate specific medical institutions |
| SEX | Indicate the patient's gender |
| PATIENT_TYPE | Indicate patient type |
| DATE_DIED | Indicate the patient's date of death |
| INTUBED | Indicate whether to intubate or not |
| PNEUMONIA | Indicate whether you have pneumonia |
| AGE | Indicate the patient's age |
| PREGNANT | Indicate whether pregnant or not |
| DIABETES | Indicate whether you have diabetes |
| COPD | Indicate whether one has chronic obstructive pulmonary disease |
| ASTHMA | Indicate whether one has asthma |
| INMSUPR | Indicate whether immunosuppression is present |
| HIPERTENSION | Indicate whether one has hypertension |
| OTHER_DISEASE | Indicate whether there are other illnesses present |
| CARDIOVASCULAR | Indicate whether one has cardiovascular disease |
| OBESITY | Indicate obesity or not |
| TOBACCO | Indicate whether tobacco is used or not |

to be comtinued …

… continued.

| | |
|---|---|
| RENAL_CHRONIC | Indicate whether the patient has chronic kidney disease |
| CLASIFFICATION_FINAL | Indicate the final disease classification |
| ICU | Indicate whether to enter the intensive care unit |

The original dataset contained data on 1,048,575 COVID-19 patients and 21 unique features, The specific meaning of each feature is shown in Table 3.2. These 21 features can be divided into four categories:

1. Demographic Feature

The demographic features include SEX and AGE.

2. Disease-related Feature

The disease-related features include CLASIFFICATION_FINAL, PATIENT_TYPE, PNEUMONIA, PREGNANT, DIABETES, COPD, ASTHMA, INMSUPR, HIPERTENSION, CARDIOVASCULAR, RENAL_CHRONIC, OTHER_DISEASE, OBESITY and TOBACCO.

3. Medical Related Feature

The medical related features include USMER, MEDICAL_UNIT, INTUBED and ICU.

4. Outcome Related Feature

The outcome related features include DATE_DIED.

## 3.5 DATA PREPROCESSING

### 3.5.1 Filling in Missing Values and Prediction Label Conversion

To better fill the missing values and deal with the missing values of 97,98,99 in each feature, and convert the date_dead label to died, and express the specific date as 1 representing death, and 9999-99-99 as 0 representing alive, representing survival. This study carried out the code operation as shown in the figure 3.3, and the specific results are shown in the figure 3.4.

```
df['renal_chronic'] =
imputer.fit_transform(df.renal_chronic.values.reshape(-1, 1))
df['tobacco'] = imputer.fit_transform(df.tobacco.values.reshape(-
1, 1))
df['pneumonia'] =
imputer.fit_transform(df.pneumonia.values.reshape(-1, 1))
# and the age values the median
imputeAge = SimpleImputer(missing_values=np.nan,
strategy='median')
df['age'] = imputeAge.fit_transform(df.age.values.reshape(-1, 1))
# now, the others columns have more null values, let's examine
them
df.intubed.value_counts(dropna=False)
df.loc[df.intubed.isnull(), 'patient_type'].value_counts()
# in patient_type the value 1 means that the patient returned at
home
# so mostly null values means that the patient returned home

df.loc[df.intubed.isnull(), 'intubed'] = 2
df[df.pregnant.isnull()].head()
# the null values means that the patient are male

df['pregnant'] =
imputer.fit_transform(df.pregnant.values.reshape(-1, 1))
df.icu.value_counts(dropna=False)
df.loc[df.icu.isnull(), 'patient_type'].value_counts()
# just like the 'intubed' column, the patient treated the virus
at home

df['icu'] = imputer.fit_transform(df.icu.values.reshape(-1, 1))
df.isnull().sum()
pd.options.display.float_format = '{:.2f}'.format

df.describe()
```

Figure 3.3 The code of filling in missing values and prediction label conversion

| | usmer | medical_unit | sex | patient_type | intubed | pneumonia | age | pregnant | diabetes | copd |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 |
| mean | 1.63 | 8.98 | 1.50 | 1.19 | 1.97 | 1.84 | 41.78 | 0.99 | 1.87 | 1.98 |
| std | 0.48 | 3.72 | 0.50 | 0.39 | 0.18 | 0.41 | 16.88 | 1.00 | 0.34 | 0.16 |
| min | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 25% | 1.00 | 4.00 | 1.00 | 1.00 | 2.00 | 2.00 | 30.00 | 0.00 | 2.00 | 2.00 |
| 50% | 2.00 | 12.00 | 1.00 | 1.00 | 2.00 | 2.00 | 40.00 | 0.00 | 2.00 | 2.00 |
| 75% | 2.00 | 12.00 | 2.00 | 1.00 | 2.00 | 2.00 | 53.00 | 2.00 | 2.00 | 2.00 |
| max | 2.00 | 13.00 | 2.00 | 2.00 | 2.00 | 2.00 | 121.00 | 2.00 | 2.00 | 2.00 |

| inmsupr | hipertension | other_disease | cardiovascular | obesity | renal_chronic | tobacco | clasiffication_final | icu | died |
|---|---|---|---|---|---|---|---|---|---|
| 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 | 1048575.00 |
| 1.98 | 1.84 | 1.96 | 1.97 | 1.84 | 1.98 | 1.91 | 5.31 | 0.35 | 0.07 |
| 0.16 | 0.38 | 0.21 | 0.18 | 0.37 | 0.17 | 0.29 | 1.88 | 0.75 | 0.26 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 3.00 | 0.00 | 0.00 |
| 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 6.00 | 0.00 | 0.00 |
| 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 7.00 | 0.00 | 0.00 |
| 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | 7.00 | 2.00 | 1.00 |



Figure 3.4 The result of filling in missing values and prediction label conversion

### 3.5.2 Feature Selection

The results of thermal map analysis show that there is a significant correlation between the features, and each feature has an important impact on the prediction output of the model. Based on this finding, we decided to keep all features without feature culling when building the final predictive model, as shown in Figure 3.5.

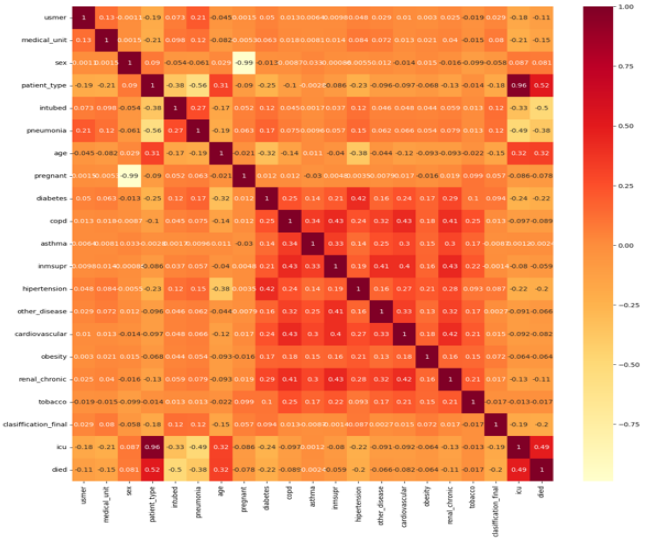



Figure 3.5 Correlation heatmap of different COVID-19 patient characteristics with mortality risk

## 3.6 DATA SEPARATION

Data segmentation is the process of dividing the entire data set into training sets, validation sets, and test sets so that different data sets can be used for model training and evaluation. This process is a critical step in machine learning and statistical modeling, as it can help evaluate a model's ability to generalize to unknown data, i.e. its predictive performance for new data.

### a. Training Set

In the realm of model training and parameter optimization, the training set typically commands the lion's share of the dataset, often hovering around the 80 %.

### b. Test Set

The test set is crucial for assessing a model's performance and its ability to generalize. This set consists of data that remains unseen by the model throughout the training process, and it is employed to mimic how the model would perform in real-world applications. Generally, the test set comprises about 20% of the total dataset, making up the balance of the data after training.

**c. Validation Sets**

By evaluating the effectiveness of the model on the validation data set, hyperparameter tuning can be performed, overfitting phenomenon can be avoided in the experiment, and subsequent training strategy optimization can be guided, and the best model can be selected among many models.

In this paper, the COVID-19 patient data set was divided into a training set and a test set according to the ratio of 80% and 20%. Then the training set after the first segmentation is generated to generate the actual training set those accounts for 64% of the original data set and the verification set those accounts for 16%. The main advantage of secondary segmentation is that it can control the proportion of training set, verification set, and test set after segmentation more flexibly and precisely, which helps to ensure the stratified sampling effect of each step and helps to adjust the segmentation proportion and optimize the experiment in subsequent experiments. This is especially the case when there is a severe imbalance in the categories of the COVID-19 patient data set. In the secondary segmentation, the segmentation step is more detailed to ensure that stratified sampling can be carried out in each step to avoid uneven distribution of categories of training set, verification set or test set, thus affecting the training and evaluation results of the model.



**d. Split Code**

**i. First split**

Split data into training set (80%) and test set (20%):

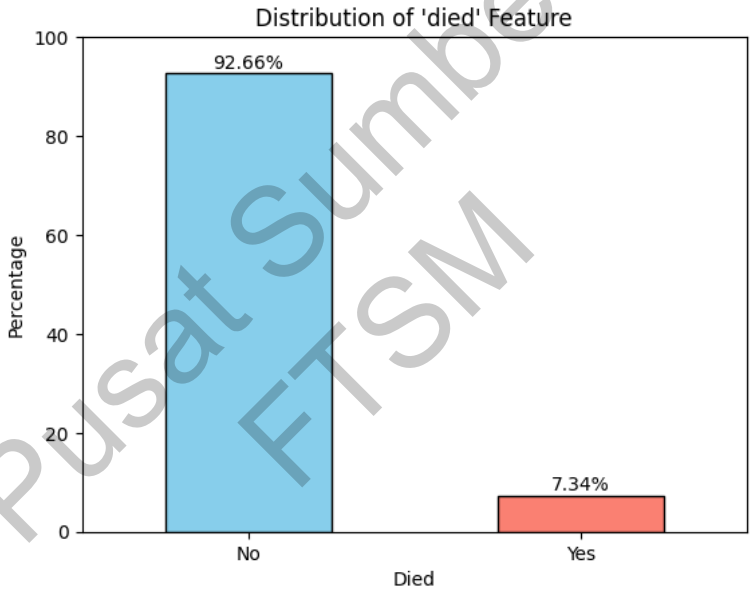
X_train_initial, X_test, y_train_initial, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

**ii. Second split**

Split training set into training (80%) and validation (20%) :

X_train_final, X_val, y_train_final, y_val = train_test_split(X_train_initial, y_train_initial, test_size=0.2, stratify=y_train_initial, random_state=42)

Figure 3.6 shows the probability distribution of the binary classification of the "died" feature. The figure indicates that 92.66% of COVID-19 patients survived, while 7.34% unfortunately passed away, reflecting a severe imbalance in the data distribution. This skewed data phenomenon could cause the model to overfit the majority class (i.e., the surviving cases) during training, negatively impacting the model's generalization performance. To mitigate the issues brought by class imbalance, the SMOTE resampling technique can be applied to synthesize minority class samples in the training data, achieving a balance between classes. This strategy helps the model to fully learn the characteristics of the minority class, thereby enhancing the performance and robustness of various algorithms, including 1D-CNN, LSTM, GRU, and 1D-CNN-LSTM.






3.6 Data distribution of the feature "died"

## 3.7 OVERVIEW OF DEEP LEARNING

Deep learning, as a major branch of artificial intelligence technology, can be traced back to the imitation of biological neural networks in the 1940s. However, it was not until recent decades that deep learning truly entered its golden age with the explosion of big data, the leap in computing power, and the innovation of algorithms. The rise of this field has not only revolutionized the basic paradigm of machine learning, but also greatly broadened the possibility of AI application in many industries, from autonomous

vehicle, image recognition, speech recognition, natural language processing to medical diagnosis, financial risk control and other fields.

The reason why deep learning can achieve such widespread applications and significant performance improvements is partly due to its ability to automatically learn and construct multi-level abstract representations from raw data. This process simulates the way the human brain understands complex information, gradually building from low-level features (such as edges and textures) to high-level concepts (such as object categories). Through progressive nonlinear transformations, deep networks can capture the complex structure of data and solve previously difficult problems such as fuzzy classification and pattern recognition.

Among them, CNN effectively reduces the number of parameters and improve computational efficiency through local weight sharing and pooling operations, especially when dealing with tasks with translation invariance (such as objects in images maintaining their characteristics regardless of their position changes). RNN and their variants, such as LSTM and GRU, solve the limitations of traditional neural networks in processing sequence data by introducing memory units, making modeling of time series data possible, such as achieving long-term dependency on context in natural language comprehension and generation tasks.



In addition, the progress of deep learning also benefits from the innovation of optimization algorithms such as SGD, Adam, and regularization strategies such as Dropout and batch normalization, which improve the training speed and generalization ability of models.

In summary, deep learning is not only a technological revolution, but also a profound transformation of the way artificial intelligence understands the world. Its continuous exploration and practice are constantly driving the new frontier of intelligent technology.

## 3.8 COMMON DEEP LEARNING ALOGRITHOMS

### 3.8.1 CNN

At its core, CNN comprises an intricate set of layers, including numerous convolutional layers, activation layers, pooling layers, and a final set of fully connected layers. The ingenious use of weight sharing and sparse connectivity in CNN, not only streamlines the parameter count and simplifies the training process but also cements their status as one of the most esteemed and prevalently utilized network types—a field that has seen considerable advancements. Within these convolutional layers, feature extraction unfolds as convolution kernels operate on the input, with the kernel count dictating the depth of the extraction process. The subsequent activation layer takes this output and infuses nonlinearity into the model's behavior. Pooling layers serve to compress the model's dimensions, thus paring down parameters further, where techniques like Max pooling and Average pooling are commonplace. Culminating in the fully connected layers, this processed information is transformed into a linear format to streamline the output of classification outcomes. This typical CNN architecture is depicted in Figure 3.7.



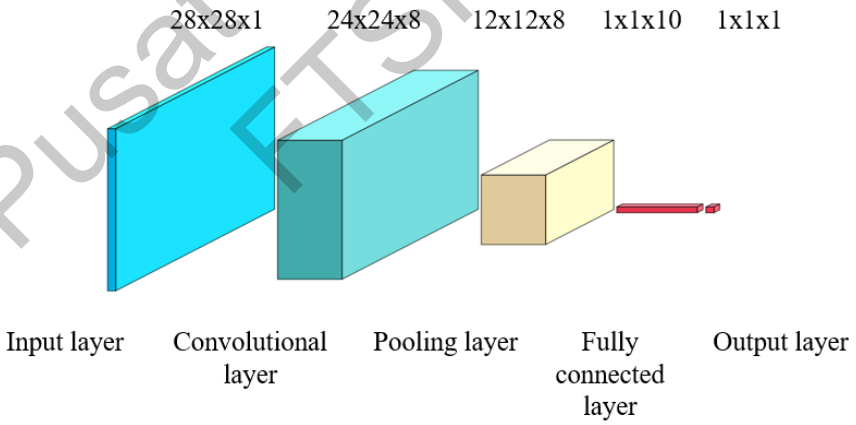



Figure 3.7 CNN structure diagram

Convolutional layers are one of the core components of convolutional neural networks, and their main function is to extract features from images or other forms of data. This task is accomplished by applying convolution operations to the input data using convolutional kernels, effectively transforming the input into more abstract representations known as feature maps. Convolution operation involves sliding the convolution kernel in the input data and performing calculations on each window to

generate new output data. In this process, the convolution kernel is a small two-dimensional tensor, usually in the shape of a square matrix, whose size is defined by the width and height of the convolution kernel, and a feature map is generated through this operation. Each element within the convolution kernel is a weight, which serves as a learning parameter and controls the output of the convolution operation. For each local region of the input data, the convolution kernel performs a multiplication operation with the elements of that region one by one, and the result is added to obtain a scalar value, which is the output of the convolution operation. The convolutional kernel initiates its journey from the upper left corner of the input data, progressively moving rightward and then downward, each time encompassing a small section to create an output value. After traversing all local regions of the input data, the convolution kernel completes the entire convolution process. After the convolution operation, an activation function, such as ReLU function, is usually added to enhance the nonlinear expression ability of the model. The convolution operation is shown in Figure 3.8.



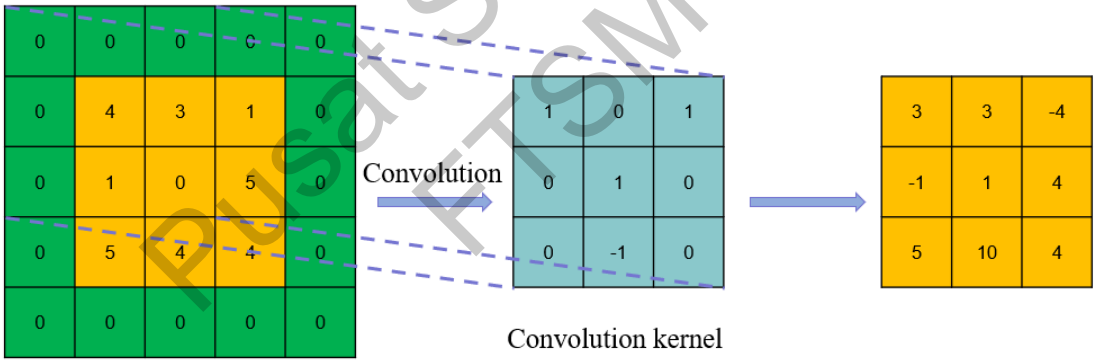



Figure 3.8 Convolution operation

The output calculation formula of the convolutional layer is shown in equation (3.1):

$$C_n=\sigma(W_n \otimes X+b_n) \tag{3.1}$$

In the given formula, $C_n$ stands for the result produced by the convolutional layer, while X signifies the CNN model's input. The activation function is denoted by σ, and $W_n$ refers to the weight matrix associated with the nth convolutional kernel in the existing layer. Furthermore, $b_n$ indicates the bias linked to the nth convolutional

kernel within that layer. The symbol ⊗ represents the convolutional operations undertaken, and N is the count of convolutional kernels involved.

The primary role of the pooling layer is to compress the input data's dimensions, diminish the network's over-sensitivity to the input, and mitigate overfitting by decreasing the parameter count. This layer generally receives data or other high-dimensional information output by the convolutional layer as input, and outputs the data after dimensionality reduction processing. The commonly used pooling operations include two methods: maximum pooling and average pooling. More specifically, the max pooling operation divides the input data into multiple disjoint regions, and then selects the maximum value from each region as the output of that region (as shown in Figure 3.9). On the contrary, the average pooling operation calculates the average value of all values in each region and takes this average value as the output, as shown in Figure 3.10.



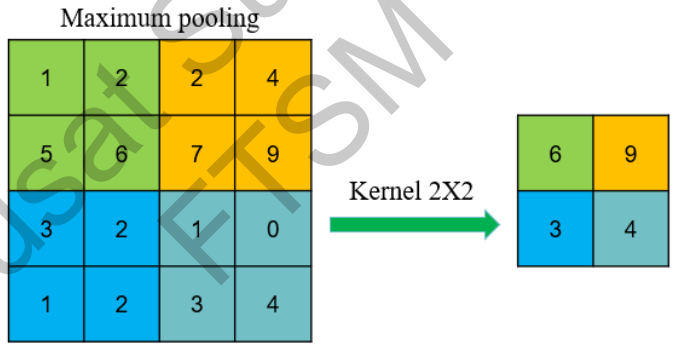



Figure 3.9 Maximum pooling operation

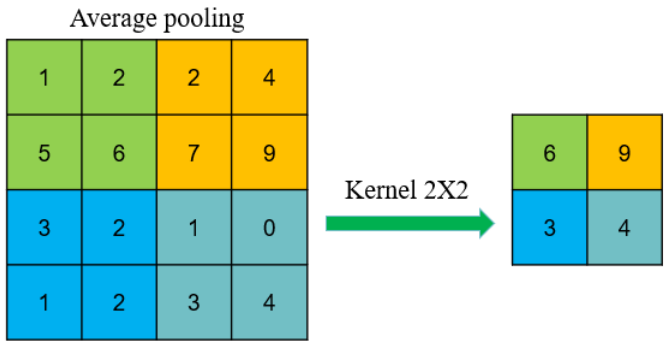



Figure 3.10 Average pooling operation

The calculation of two pooling operations is shown in equations (3.2) and (3.3):

$$P_n=MAX(C_n) \tag{3.2}$$

$$P_n=AVERAGE(C_n) \tag{3.3}$$

In the formula: $C_n$ represents the input of the pooling layer; $P_n$ represents the output of the pooling layer; MAX represents the maximum pooling function; AVERAGE represents the average pooling function.

Within the realm of deep learning, densely connected layers stand as a cornerstone, consistently employed across an array of neural network architectures. These layers function by establishing connections between every neuron in a given layer and all neurons in the preceding layer. The output generated emerges as a cumulative weighted sum of the preceding layer's neurons, subsequently transformed by the application of a non-linear activation function. Usually, the fully connected layer's output can be either classification or regression results. The fully connected layer can help the network better learn features from input data through feature extraction, data classification, and parameter tuning, thereby improving the network's performance and generalization ability. The fully connected structure is shown in Figure 3.11.



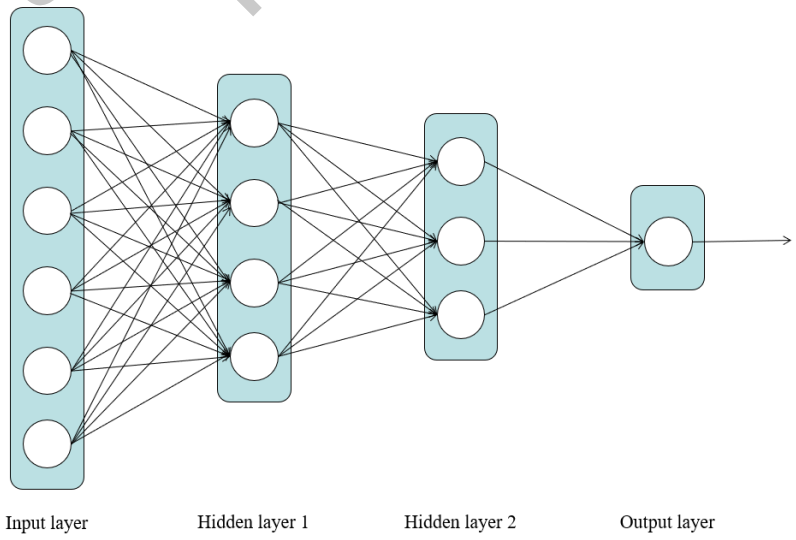



Figure 3.11 Fully connected layer structure

The formula for calculating the fully connected layer is shown in equation (3.4):

$$H_C=Sigmoid(P_nXW_n+b_n) \quad (3.4)$$

In the formula: $H_C$ represents the output of the fully connected layer; $P_n$ represents the output of the pooling layer and is also the input of the fully connected layer; $W_n$ represents the network weight coefficient; $b_n$ represents the bias coefficient.

### 3.8.2 LSTM

Hochreiter and Schmidhuber (1997) presented the LSTM, a sophisticated iteration of recurrent neural networks, specifically designed to tackle the problem of insufficient long-term memory found in conventional RNN. In an LSTM, the process of adding or removing information is managed by three types of gates: forget gates, input gates, and output gates. The forget gate decides which information from the previous unit state should be discarded. The input gate regulates the introduction of new information into the unit state, creates candidate vectors for the unit state, and updates the old unit state to the new one accordingly. The output gate controls which pieces of information are released from the present unit's state. Figure 3.12 illustrates the structure of the LSTM.



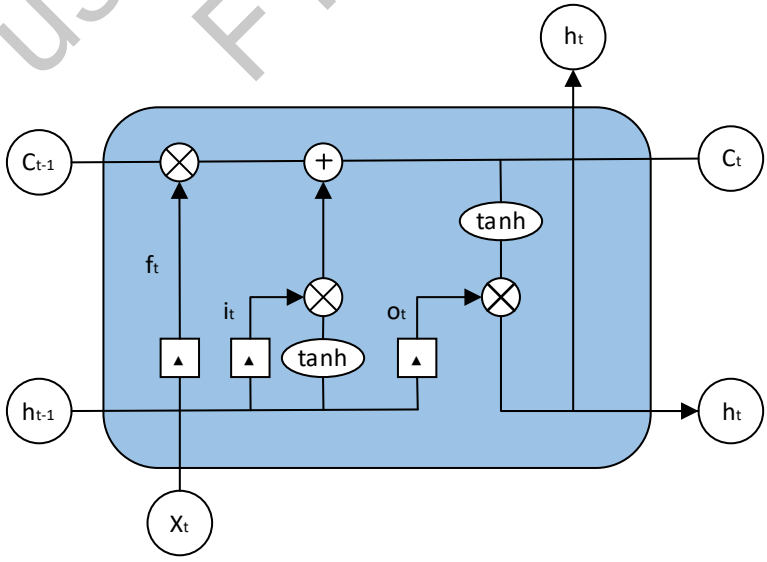



Figure 3.12 LSTM architecture diagram

**a. Forgetting Gate**

The information to be discarded from the cellular state is calculated as shown in Equation (3.5).

$$f_t=\sigma\left(w_f[h_{t-1},x_t]+b_f\right) \quad (3.5)$$

In the formula: $w_f$ represents the network weight; $b_f$ represents bias; $h_{t-1}$ represents the output of the previous moment; $x_t$ represents the input at time t; $\sigma$ represents the activation function.

**b. Input Gate**

It determines the information to be saved in the cell state. Divided into two parts, the first part is to generate new temporary cell states, and the second part is to update old cell states.

The calculation process for generating new temporary cell states is shown in equations (3.6) and (3.7):

$$i_t=\sigma(w_i\ [h_{t-1},x_t]+b_i) \quad (3.6)$$

$$\widetilde{C}_t=tanh(w_c\ [h_{t-1},x_t]+b_c) \quad (3.7)$$

In the formula: $h_{t-1}$ represents the hidden layer output at time t-1; $x_t$ represents the input at time t; $w_i$ and $w_c$ represent network weights; $b_i$ and $b_c$ represent biases; It represents the input threshold; $\widetilde{C}_t$ represents the temporary cell state.

The generation of new temporary cell states can be further divided into two steps: first, read the hidden layer output $h_{t-1}$ at time t-1 and the input xt at time t, and convert the value to between (0~1) through the sigmoid function, i.e. it. Secondly, read the hidden layer output $h_{t-1}$ at time t-1 and the input xt at time t. Through the tanh activation function, convert the value to between (-1~1), and then multiply it with it to obtain the final input information at time t.

The other part is the update of the old cell state $C_{t-1}$. The update formula is shown in equation (3.8):

$$C_t=f_t*C_{t-1}+i_t*\widetilde{C}_t \quad (3.8)$$

In the formula: $f_t*C_{t-1}$ represents selectively forgetting past information; $i_t*\tilde{C}_t$ represents selectively retaining new information; Ct represents the updated new cell state.

**c. Output Gate**

To determines which information in the cell state needs to be output, calculated as shown in equations (3.9) and (3.10):

$$o_t=\sigma(w_o[h_{t-1},x_t]+b_o) \quad (3.9)$$

$$h_t=o_t*tanh(C_t) \quad (3.10)$$

$w_o$ represents the network weight; $b_o$ represents bias; At a given time, symbolized as t, the output produced by the hidden layer is depicted as $h_{t-1}$. Simultaneously, the input at this specific period is denoted as $x_t$. The established output threshold is represented as $o_t$, whereas $h_t$ characterizes the resultant output from the hidden layer during this time. Also in this time frame, $C_t$ stands for the cell state.

This gate first reads the hidden layer output $h_{t-1}$ at time t-1 and the input $x_t$ at time t and converts the value into a range between (0~1) through the sigmoid function, which is the output threshold $o_t$. Then, the cell state $C_t$ is transformed into values between (-1~1) through the tanh function, and multiplied by the output threshold ot to obtain the final selected output information, which is the hidden layer output ht at time t.

### 3.8.3 GRU

The GRU, or Gated Recurrent Unit, is designed to deal with time series data, addressing the gradient vanishing issue found in standard RNNs. It manages information flow through learnable gates and is often considered a simplified variant of LSTM. Like LSTM, GRU utilizes gating mechanisms to regulate input, memory, and other information flows. These gates work together to decide which information is retained as the output of the GRU. The distinctive feature of these gating mechanisms is their ability to maintain long-term sequence information, preventing it from being erased over time or discarded as irrelevant to predictions. The GRU's architecture is depicted in Figure 3.13.

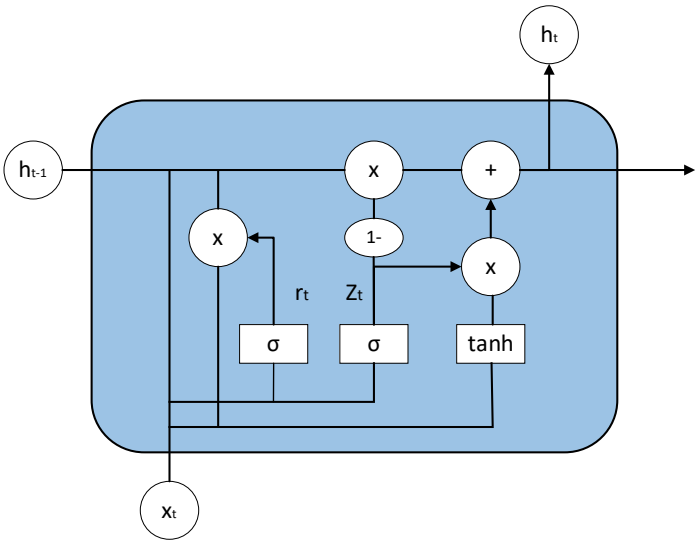



Figure 3.13 GRU structure diagram

**a. Reset Gate**

Fundamentally, the reset gate is crucial for deciding the extent of prior information that should be disregarded, and this is calculated as depicted in the Equation (3.11):

$$r_t = \sigma(w_r [h_{t-1}, x_t] + b_r) \qquad (3.11)$$

In equation (3.11), $x_t$ stands for the input received at the t time index. The output yielded by the hidden layer at index t-1 is symbolized as $h_{t-1}$. Network weight is signified by $w_r$, whereas $b_r$ is indicative of bias. Lastly, the outcome produced by the reset gate is denoted by $r_t$.

This gate will read the input $x_t$ at time t and the output $h_{t-1}$ at time t-1, first undergo a linear transformation, and then convert the value to (0-1) through the Sigmoid activation function, that is, $r_t$

**b. Update Gate**

Determine how much information from the past needs to be transmitted to the future or determine how much information from the previous and current time steps needs to continue to be transmitted. This feature is very powerful because the model can decide to copy all the information from the past to reduce the risk of gradient vanishing. The formula for its calculation is presented in the following equation (3.12):