1 2 9 0

UNIVERSIDADE Đ COIMBRA

Filipa Graça Carvalho

Deep Modelling for Anticancer Drug Response Prediction with Therapeutic Data

Dissertation in the context of the Master in Biomedical Engineering, Specialization in Clinical Informatics and Bioinformatics advised by Prof. Dr. Joel P. Arrais and Prof. Dr. Maryam Abbasi and presented to the Faculty of Sciences and Technology.

September 2022

Faculty of Sciences and Technology

Department of Informatics Engineering

Deep Modelling for Anticancer Drug Response Prediction with Therapeutic Data

Filipa Graça Carvalho

Dissertation in the context of the Master in Biomedical Engineering, Specialization in Clinical Informatics and Bioinformatics advised by Prof. Dr. Joel P. Arrais and Prof. Dr. Maryam Abbasi and presented to the Faculty of Sciences and Technology.

September 2022

1 2 9 0

UNIVERSIDADE Ð COIMBRA

This work was developed in collaboration with:

Center for Informatics and Systems of the University of Coimbra

.CISUC

Centre for Informatics and Systems of the University of Coimbra

Acknowledgments

Esta dissertação marca o fim de uma jornada de cinco anos na Universidade de Coimbra que não seria possível unicamente com esforço individual. Assim, deixo o meu agradecimento a todas as pessoas que, de diversas formas, ensinaram e apoiaram, contribuindo para a concretização deste trabalho.

Gostaria de começar por agradecer aos Professores Joel P. Arrais e Maryam Abbasi por me terem dado a oportunidade de desenvolver este trabalho, por toda a confiança e acompanhamento ao longo dos últimos meses. Todos os desafios propostos, questões colocadas e discussões foram essenciais não só para o desenvolvimento deste trabalho, mas também do meu crescimento pessoal. Obrigada pela compreensão e dedicação.

Um palavra de agradecimento à professora Bernardete Ribeiro pela disponibilidade e incentivo. Ainda, agradeço ao Tiago O. Pereira e ao Tiago C. Pereira por todas as contribuições e conselhos imprescindíveis que auxiliaram na melhoria deste trabalho. A todos os restantes colegas do LARN, pelos inúmeros debates e sugestões, um grande obrigada.

Aos meus pais por todo o esforço e apoio para que eu pudesse ter a melhor experiência de estudar em Coimbra. Obrigada por acreditarem sempre em mim e por nunca me deixarem desistir. Sem vocês este percurso não seria possível e sei que vou sempre contar com o vosso amor. Obrigada para todo o sempre.

Ao meu irmão, avós e restante família que sempre me apoiaram e de uma maneira ou de outra tornaram este caminho mais simples com apenas uma palavra de incentivo, obrigada.

Às minhas amigas de Coimbra que estiveram lá desde o primeiro dia, que juntas partilhámos alegrias e também as mágoas durante estes cinco anos. Ficará para sempre na minha memória todos os momentos, trabalhos infindáveis e noitadas nesta cidade que se tornou um bocadinho nossa. A todos os outros amigos que o curso nos fez questão de juntar, obrigada pela companhia, pelas conversas e por todas as partilhas.

À minha colega de casa que se tornou uma grande amiga, que lidou comigo todos os dias, nos bons e maus momentos, um eterno obrigada pela paciência, carinho e apoio.

Às amigas que Coimbra não me apresentou mas fez questão de juntar em diversos encontros, obrigada por estarem sempre presentes e pela constante motivação.

Por fim, um obrigada a ti, Coimbra, por esta aventura retratada em cinco anos memoráveis.

Obrigada.

Financing

This research has been funded by the Portuguese Research Agency FCT, through D4 - Deep Drug Discovery and Deployment (CENTRO-01-0145-FEDER029266). This work is funded by national funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the project CISUC - UID/CEC/00326/2020 and by European Social Fund, through the Regional Operational Program Centro 2020.

“All men dream: but not equally. Those who dream by night in the dusty recesses of their minds wake up in the day to find it was vanity, but the dreamers of the day are dangerous men, for they may act their dreams with open eyes, to make it possible.”

T.E. Lawrence

Resumo

O cancro é uma das principais causas de morte em todo o mundo, urgindo a necessidade da sua deteção e tratamento. Na era da medicina de precisão, o principal objetivo é incorporar a variabilidade individual para selecionar com mais exatidão as estratégias de terapia e prevenção que se adequam a cada pessoa. No entanto, a previsão acerca da resposta a um fármaco para o tratamento do cancro continua a ser um desafio.

Abordagens bem sucedidas, incluindo aprendizagem de máquina e aprendizagem profunda, foram apresentadas para prever a sensibilidade de tumores a tratamentos anticancerígenos, contudo, devido à complexidade do problema, permanecem ainda muitos desafios em traduzir a combinação de diversos dados clínicos com dados genómicos num prognóstico adequado.

Neste trabalho, são propostos dois modelos de arquitetura de aprendizagem profunda para prever o efeito de fármacos anticancerígenos em tumores através da concentração inibitória média (IC50). Ambos os modelos podem ser divididos em duas partes: primeiramente, é realizado o pré-treino de dois *autoencoders* com dados de grande dimensão (expressão e mutação genéticas) para capturar as características fundamentais dos tumores; de seguida, este conhecimento genético é transferido para linhas celulares cancerígenas para prever o impacto das variantes genéticas num determinado fármaco. Ademais, foram introduzidas estruturas SMILES para o modelo apreender as características relevantes das moléculas através da utilização de Redes Neuronais Recorrentes e Redes Neuronais Convolucionais. Finalmente, o modelo aplica dados de sensibilidade de fármacos correlacionados com os dados genómicos e moleculares para identificar características que prevêem o valor de IC50 para cada par fármaco-linha celular.

Os resultados obtidos demonstram a capacidade de ambos os modelos em extrair representações relevantes dos dados para prever interações fármaco-recetor, isto é, o valor de IC50 que descreve a potência de uma substância em inibir um tumor. Os modelos propostos alcançaram um desempenho de erro quadrático médio de 1,08 e 1,06, superando modelos anteriormente apresentados no estado da arte.

Palavras-Chave

Aprendizagem Profunda, Redes Neuronais Recorrentes, Redes Neuronais Convolucionais, Expressão Génica e Perfis de Mutação, SMILES

Abstract

Cancer is a leading cause of death worldwide, enhancing the need for its detection and treatment. In the era of precision medicine, the main goal is to incorporate individual variability in order to choose more accurately which therapy and prevention strategies suit each person. However, drug response prediction for cancer treatment remains a challenge.

Successful approaches, including machine and deep learning, have been proposed to predict the sensitivity of tumors to specific anticancer treatments, however, due to the complexity of the problem, there remain many challenges in how to effectively translate the combination of clinical data with genomics data into prognostic.

In this work, it is proposed two deep neural network models to predict the effect of anticancer drugs in tumors through the half-maximal inhibitory concentration (IC50). The models can be seen as two-fold: first, we pre-trained two autoencoders with high-dimensional data (gene expression and mutation profiles) to capture the essential features from tumors; then, this genetic background is translated to cancer cell lines to predict the impact of the genetic variants on a given drug. Moreover, SMILES structures were introduced so that the model can apprehend relevant features regarding the drug compound using Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). Finally, the model applies drug sensitivity data correlated to the genomic and drugs data to identify features that predict the IC50 value for each pair of drug-cell line.

The obtained results demonstrate the effectiveness of both models in extracting deep representations to predict drug-target interactions, i.e., the IC50 value that portrays the potency of a substance in inhibiting a tumor. The proposed models achieved a performance of a mean squared error of 1.08 and 1.06, outperforming previous state-of-the-art models.

Keywords

Deep Learning, Recurrent Neural Networks, Convolutional Neural Networks, Gene Expression and Mutation Profiles, SMILES.

Contents

List of Tables

List of Figures

Abbreviations

CTD^2	Cancer Target Discovery and Development
IC_{50}	Half Maximal Inhibitory Concentration
R^2	Coefficient of Determination
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASCII	American Standard Code for Information Interchange
BRCA	Breast Cancer Gene
CCLE	Cancer Cell Line Encyclopedia
CNN	Convolution Neural Network
CNV	Copy Number Variation
DL	Deep Learning
DNA	Deoxyribonucleic Acid
DNN	Deep Neural Network
DREAM	Dialogue on Reverse Engineering Assessment and Method
FCNN	Fully Connected Neural Network
GDSC	Genomics of Drug Sensitivity in Cancer

GNN	Graph Neural Network
GRU	Gated Recurrent Units
LSTM	Long Short-Term Memory
MAF	Mutation Annotation Format
ML	Machine Learning
MSE	Mean Squared Error
NCI	National Cancer Institute
NLP	Natural Language Process
NN	Neural Network
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network
RSEM	RNA-Seq by Expectation-Maximization
SMILES	Simplified Molecular Input Line Entry System
SVM	Support Vector Machine
TCGA	The Cancer Genome Atlas
TPM	Transcripts Per Million
UCSC	University of California, Santa Cruz

1

Introduction

1.1 Context and Motivation

Diverse commonly prescribed drugs have unexpected side effects based on individual inherited genetic variants. Despite the advances in high-throughput sequencing technology, due to the low frequency of some genetic variants and the inherent complexity of drug development, association studies are mainly conducted after the drug is approved and adverse reactions are reported. To tackle this problem, pharmacogenomics focuses on the association between genetic variants and drug responses to enhance the development of effective, safe medications and doses, aiming to build a scenario where drug prescription is based on patient profile [1].

Since cancer is a leading cause of death worldwide, it increased the demand to predict drug response and identify novel anticancer drugs, resulting in an exhaustive process due to intra- and inter-tumor heterogeneity. Therefore, several projects, like The Cancer Genome Atlas (TCGA) Program, are making efforts to catalog genetic mutations for cancer and study the entire spectrum of genomic changes involved in human cancer through the application of genome analysis technologies [2]. These researches provide a massive amount of publicly available data, enabling the development of bioinformatics tools to highlight candidate cancer biomarkers and drug targets. Therefore, the main goal is to construct an efficient model able to correlate genomic information with anticancer compounds. Figure 1.1 illustrates the field of cancer pharmacogenomics and how the different parts that compose it are interconnected. As may be noted, identifying the best drug(s) requires an understanding of tumor genomics. However, it is also necessary to incorporate chemical descriptors of drugs to understand drugs' mechanisms [3].

Considering the sizeable combinatorial scale of the problem, this can benefit from

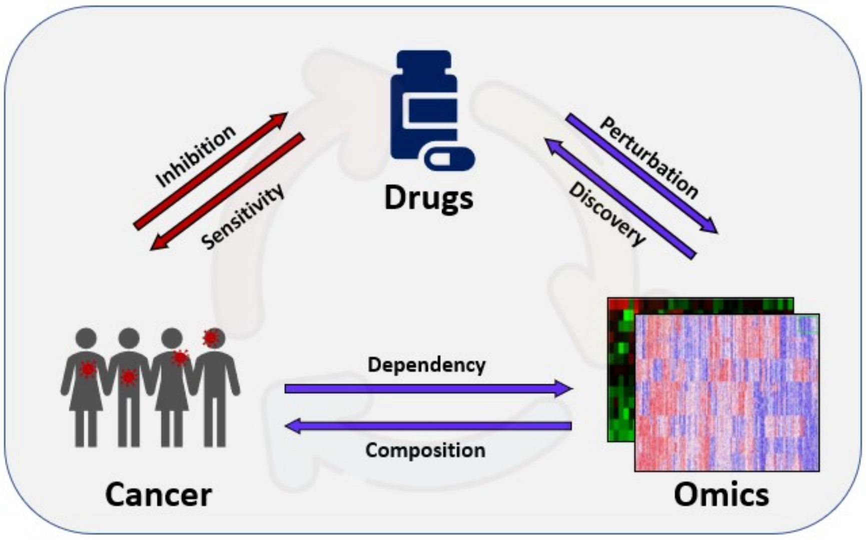


Figure 1.1: Overview of pharmacogenomics in cancer where three components are linked: drugs, genomics, and patient responses.

recent advances in Deep Learning (DL) methods. DL technologies can be applied directly to raw data, automatically learn useful features and make predictions without prior knowledge [3]. However, the limited drug response data compared to cell line screening data resulted in a barrier to using these algorithms. In recent years, the Genomics of Drug Sensitivity in Cancer (GDSC) project was able to relate more than 1000 human cancer cell lines with a wide range of anticancer compounds, identifying alterations in cancer genomes that strongly influence patient response [4].

Although computational models have been proposed for this issue, there is still room for improving the prediction performance by combining multiple types of drug and genomic data. This work presents two deep neural network models to predict the effect of anticancer drugs in tumors through the half-maximal inhibitory concentration (IC_{50}). Both models are trained using public mutation and expression profiles of many cell lines to capture the genomic heterogeneity that underlies human cancer and also the drug's candidate in the format of Simplified Molecular Input Line Entry Strings (SMILES) notation to extract relevant features from the chemical compounds. Furthermore, the models were trained with drug sensitivity data correlated to the genomic data retrieved from the GDSC project to identify genetic features that are predictive of sensitivity. Therefore, this study aims to address the challenge of using computational methods to simply and accurately predict the effect of genetic variants on a given drug.

1.2 Background

Gene Expression

Genes encode proteins, on the other hand, proteins dictate cell function, i.e., the thousands of genes expressed in a particular cell determine what that cell can do. Therefore, gene expression describes the process where the information encoded in a gene is used to make RNA molecules or non-coding RNA molecules. The RNA molecules are responsible for encoding the proteins, whereas non-coding RNA molecules play important roles in diverse biological processes but do not have apparent protein-coding roles. Thereby, gene expression acts as an "on/off switch" to control when and where RNA molecules and proteins are made [5]. From one side, gene overexpression is the process that causes abundant target protein expression. On the other side, gene underexpression can lead to a lack of protein expression. Thus, this process is essential to maintain cellular structure and function, changing considerably under different conditions such as environmental factors like diet, drugs, or exposure to toxins [6, 7].

The gene expression analysis provides quantitative information about the RNA molecules in cells and tissues through microarrays, transcriptome profiling by RNA sequencing (RNA-Seq), and other methods that measure the levels of RNA molecules in biological systems. The classical analysis of RNA-Seq data focuses on finding genes that present differential expression between groups. The main advantages of this method, compared with microarrays-based approaches, are the increase in sensitivity, the capacity to detect unannotated transcripts, and the digital quantification of RNA molecules [8]. The increased availability of genomic technologies and the development of processing techniques have enabled the analysis of extensive amounts of data. As a result, gene expression profiling has become a tool for detailing tumor molecular profiles [9].

Naturally, genetic variants (or mutations) affect gene expression levels and consequently impact protein levels, cell morphology, or disease phenotypes [10]. In this matter, gene expression has proven to be a powerful tool from which conclusions can emerge regarding the genes which are more impactful in certain diseases.

Gene Mutations and Cancer

One of the keys to understanding cancer is knowing which genes and mutations promote tumor development, allowing the growth of cells and tumor progression.

It has been proven that tumor cells are the result of a course of genetic and epigenetic events caused by DNA or environmental changes [11]. These alterations can origin different genetic alterations like chromosomal changes, gene mutations, and epigenetics events. Among the gene mutations are included base substitutions, rearrangements, and small insertions and deletions. In this way, these changes in specific genes, and different patterns of gene mutations can lead to a disruption of genetic pathways, such as cell cycle control and cell signaling, contributing to tumor development [12].

Drug Response

Drugs play a significant role in modern medicine as they treat, control, cure, or prevent a disease or disorder. However, drug discovery and development is a complex, long and expensive process. Although several strategies have been suggested in the artificial intelligence field, there remain new techniques and pathways to be pursued [13]. Pharmacogenomics by combining pharmacology (the science of drugs) and genomics (the study of the genome and gene functions) may enhance drug discovery and development through the identification of genetic variants that predispose individuals to adverse drug reactions [14]. Consequently, can be examined the possibility of subpopulation-specific drug development.

Changes in DNA sequences can lead to different expression or function of proteins that are targeted by drugs making individuals respond differently to drugs, sometimes with unpredictable side effects. Despite the idea proposed in the 1950s that genes regulate some drug responses, the description of the first human gene containing a DNA sequence with alterations that impact drug metabolism just took place in the late 1980s. Nonetheless, there are still many challenges to overcome in order to understand the total contribution of genetic variants to drug effects and transfer this knowledge into clinical practice [1].

In precision medicine, the goal is to design treatments according to the characteristics of a particular cohort or even individual patients. Therefore, it depends on the effective translation of high-throughput profiling data into clinically meaningful results. In this matter, the amount of drug response profiles spanning a wide range of drugs and genetic analysis of different cohorts is essential to identify drug response biomarkers and develop predictive models of drug sensitivity [15].

One of the most used measures of effectiveness in inhibiting biological functions is the IC_{50}. The IC_{50} value indicates the concentration of an inhibitory substance that is required to suppress a given biological function by half. Therefore, large IC_{50} values correspond to compounds that interact less efficiently with a cell than drugs with small IC_{50} values [16].

1.3 Objectives

The main goal of this master thesis is to address the challenge of using deep learning methods to accurately predict the effect of genetic variants on a given drug. Thus, the goal is to develop two half-maximal inhibitory concentration predictors of anticancer drug responses. In this respect, gene expression and gene mutation, and SMILES notation are used as descriptors of cell lines and drugs, respectively. Given a pair of drug-cell line, the models predict the IC_{50}, which portrays the impact of the genetic variant on the drug. Therefore, the following points summarize the objectives of this project:

1. Overview of DL techniques and their current applications to anticancer drug prediction.
2. Explore different deep architectures and the best hyperparameters to train the prediction models.
3. Implement two autoencoders in order to extract relevant information regarding tumors.
4. Implement two prediction models based on RNNs and CNNs that can predict the half-maximal inhibitory concentration of anticancer drug responses.
5. Evaluate the proposed models and compare them with different state-of-the-art approaches.

1.4 Workflow

The significant rise in the prevalence of cancer worldwide in the last decade [17] led to the pursuit of anticancer drug response prediction approaches. This work focuses on aligning computational methods with the science of pharmacogenomics to increase the potential to yield a new set of diagnostic tools that can be used to individualize and improve drug therapy. In this regard, different deep learning architectures will be explored to construct models that predict the drug response, i.e., the impact of genetic variation on treatment response. These models allow to automatically identify hidden and complex patterns and relationships between genetic features (gene expression and gene mutation) and molecules' descriptors to predict drug response. Accordingly, they learn this relation using intrinsic information of the cell lines and drugs, without needing to verify each interaction experimentally and surpassing the traditional approaches on its capacity to predict the half-maximal inhibitory concentration. The main goal is to translate knowledge of human genome variability into better therapeutics, decreasing the frequency of adverse drug effects.

Figure 1.2 shows the contrast between traditional medicine and drug response prediction. Drug selection is subjective, and typically it is required weeks of symptom evaluation to determine treatment efficacy, emerging the need to develop pharmacogenomic/computational approaches that can support decisions regarding optimal drug choice, and identify poor drug responses.

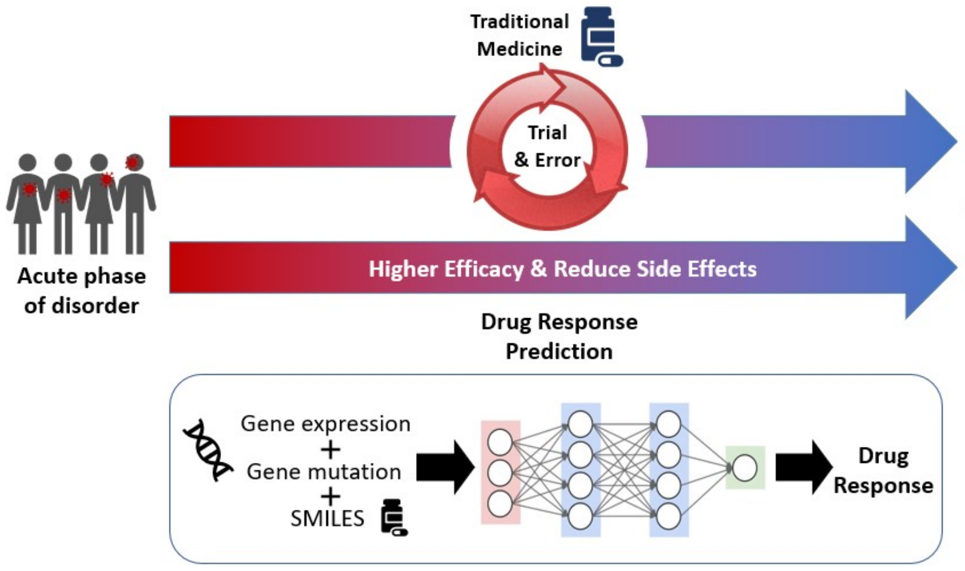


Figure 1.2: Traditional medicine versus drug response prediction aligned with computational methods: the use of genetic and compound features.

1.5 Scientific Outcomes

During this thesis, the work developed resulted in contributions submitted to international journals and presented at national/international conferences.

Papers

- Filipa G. Carvalho, Maryam Abbasi, Bernardete Ribeiro, Joel P. Arrais. "Predicting Drug Activity against Cancer through Genomic Profiles and SMILES". "Mining healthcare: AI and deep learning for medicine" in the journal Artificial Intelligence in Medicine (Elsevier), 2022. (In preparation)

- Filipa G. Carvalho, Maryam Abbasi, Bernardete Ribeiro, Joel P. Arrais. "Deep Model for Anticancer Drug Response through Genomic Profiles and Compound Structures". IEEE CBMS, IEEE 35th International Symposium on Computer Based Medical Systems, 2022. (Presented on July 2022) - **Best Paper Award.**

Posters

- Filipa G. Carvalho, Maryam Abbasi, Bernardete Ribeiro, Joel P. Arrais. "Deep Modelling for Anti-Cancer Drug Response through Gene Expression and Mutation Data". ESCI, 56th Annual Scientific Meeting of the European Society for Clinical Investigation (Presented on June 2022) - **Best Poster Award.**
- Filipa G. Carvalho, Maryam Abbasi, Bernardete Ribeiro, Joel P. Arrais. "Predicting Anticancer Drug Response through Deep Learning". VIII EJIBCE, Structural Computational Biology Meeting of Junior Researchers (Presented on December 2021).

All the implemented models and datasets for this study are publicly available on:

- https://github.com/larngroup/DeepModel-Anticancer-Predictor.

1.6 Document Structure

This thesis is composed of six chapters. The current and first chapter aims to clarify the context of the problem and how this work intends to contribute to its solution. Chapter 2, State of the Art, presents a review of various research works developed using computational approaches to improve the prediction of the effect of a genetic variant on a given drug. Chapter 3, Deep Learning Models, provides a theoretical overview of the architectures used to build the proposed models. Chapter 4, Methods, describes the contribution of this master thesis and englobes the different models that were implemented. Chapter 5, Results and Discussion, covers the performed experiments and the obtained results. Finally, chapter 6 concludes this study and summarizes the possible future approaches to the proposed work.

2

Artificial Intelligence in Anticancer Drug Response Prediction

Resistance to anticancer compounds arises from a wide range of factors, including genetic mutations or epigenetic changes and other factors such as microbiome and different molecular mechanisms. So several approaches, based on different perspectives, have been reported and explored to solve the problem of predicting drug response. Among diverse computational methods presented in the literature stands out the use of machine learning algorithms like support vector machines and deep learning models applied to gene expression and mutation profiles and drugs' fingerprints. However, using the deep architecture to predict the drug response is still a great challenge, therefore remains the possibility for an improvement in the predictive performance and generalizability of the models [15].

2.1 Notations and Definitions

Artificial intelligence (AI) is the science of creating an intelligent machine capable of simulating human intelligence. It plays a relevant role in our daily routines by performing diverse tasks, such as natural language processing (NLP) [18], facial recognition [19], and, particularly, drug response prediction [15].

Figure 2.1 shows the relationship between artificial intelligence, machine learning, and deep learning. Machine learning is a branch of AI, which refers to systems that can learn without being explicitly programmed, i.e., learn by themselves from experience on a given sample of information [20]. These methods can be divided into supervised, unsupervised, or reinforcement learning (RL). Supervised learning algorithms are designed to make associations between inputs and outputs based on example input-output pairs. Once associations have been learned, the models can be used to predict unseen examples. Unsupervised methods are characterized by discovering hidden patterns in data without prior training examples. These algorithms are commonly associated with clustering problems, which goal is to identify and group similar data points without concern for the specific outcome. Lastly, RL is the task of learning actions in an environment in order to maximize cumulative rewards [21].

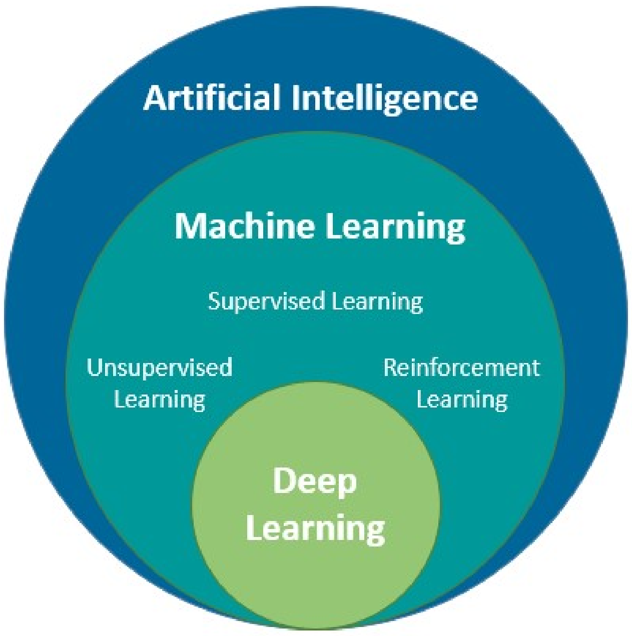


Figure 2.1: Relationship between Artificial Intelligence, Machine Learning, and Deep Learning: same context, different concepts.

Deep learning is a subfield of machine learning based on algorithms that replicate the structure and function of the brain, named Artificial Neural Networks (ANNs). An ANN is a computational model inspired by biological neural networks, capable of learning from experience on large datasets. It is established that biological learning is characterized by complex networks of neurons that can transmit information through synapses from several locations to the place of action, assimilating knowledge and performing actions. If the received stimulus is enough to trigger an action potential, i.e., reaches the threshold potential, the neuron transmits the signal along the axon, producing a response in the organism. In this matter, a single neuron cannot act alone, and nervous system function depends on groups of neurons that work together. However, it is relevant to note that, many times, the behavior of biological processes is difficult to interpret due to their inherent intra- and inter-variability. Therefore, an ANN is a simplification of biological neural networks composed of artificial neurons that communicate by transferring information from one neuron to another and are interconnected through multiple layers, defined as hidden layers [22, 23].

The ANN architecture is typically composed of an input layer, multiple hidden layers, and an output layer. The input layer brings the independent values, which constitute the initial data, into the network for posterior processing by the neurons that form the hidden layers. Then, the hidden layers are responsible to transform the weighted inputs according to the expected outcome. Finally, the output layer applies an activation function to the weighted sum of all the outputs provided by the previous layers. Thus, an artificial neuron, represented in Figure 2.2, is the basic building block of an ANN and it is organized into 5 elements [24]:

- **Input**: Independent values that are fed to the neuron.
- **Weights**: Vector of weights which determine the importance of each input.
- **Bias**: "extra neuron" that shifts the activation function by adding a constant to the input.
- **Activation function**: Responsible for the activation of each neuron by applying a transformation to the resulting value of the weighted sum. Usually, it is a non-linear function.
- **Output**: Result of applying the activation function to the sum of the previously weighted inputs and bias. The output of the i^{th} neuron can be mathe-

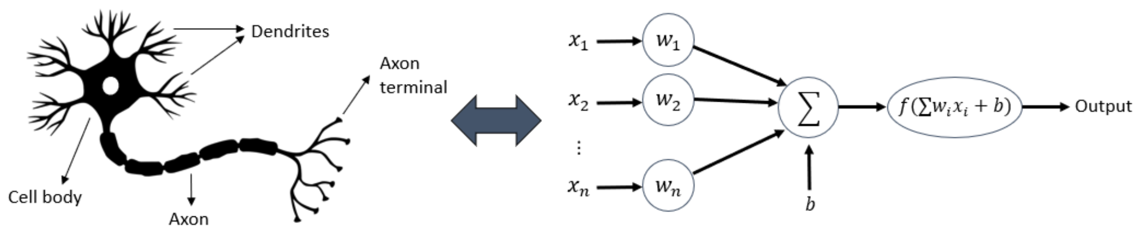


Figure 2.2: Comparison between biological and artificial neuron.

matically defined as:

$$y = f(\sum_j w_{ij} x_j + b_i) \tag{2.1}$$

, where w is the weight, x the input value and b the bias.

The most common form of learning, deep or not, is supervised learning, whose objective is to map an input to an output based on example input-output pairs. During the training stage, the neural network updates its adjustable weights by calculating, at each iteration, a loss function, which is the difference between the obtained output and the expected value. This process to find the best set of weights is performed by backpropagation, and how the weights are updated depends on the chosen optimizer [25]. The backpropagation algorithm computes the gradient of the loss function to fine-tune the neural network weights based on the error obtained in the previous iteration. Theoretically, the model becomes better as the weights are updated, and the error tends to decrease. The problem's type to solve determines the choice of the optimizer and the loss function [26].

2.2 Computational Methods in Anticancer Drug Response Prediction

The increase in the amount of digitalized data and available computational power has prompted research on the integration of AI in anticancer drug response prediction intending to increase efficacy, reduce side effects, and increase the rates of success in cancer recovery.

Although conventional machine learning techniques usually result in good performance, they are limited in their inherent capability to process raw data. Deep learning has been demonstrated to be very efficient at extracting hidden and complex patterns in high-dimensional space, resulting in better performance in most cases [15]. Therefore this approach has been applied to many domains of science, including image recognition and language translation [27].

The current work in the drug response field is focused on the design of frameworks that can predict drug sensitivity, i.e., predict how cancer cell lines would respond to both experimental compounds and drugs that have already been approved. In this context, several prediction models (deep or not) have been proposed through this issue [15].

2.3 Data Representation

2.3.1 Tumor and Cell Line Features

In living beings, genetic information in the cells flows from DNA to the mRNA to protein. The collection of genes that are transcribed from DNA, also referred to as the expression profile, is a determinant factor in cellular phenotype and function [28]. The gene expression profile is commonly defined in a matrix, where each row represents a gene, and each column represents a sample. In this way, each entry in the matrix displays the expression level of a particular gene in a sample/cell. Figure 2.3 presents a diagram of the gene expression matrix.

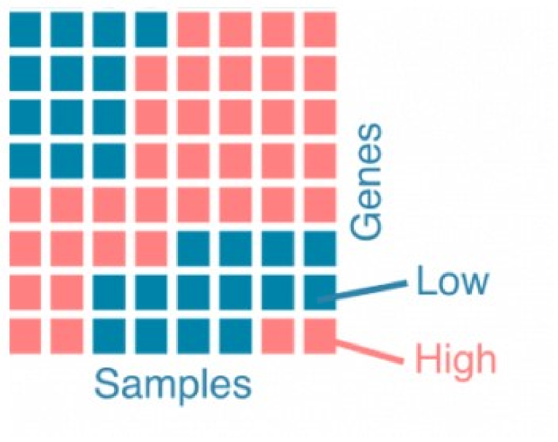


Figure 2.3: Schematic diagram of a gene expression matrix [1].

Similar to the gene expression profile, there are different features that can be used to characterize the cell lines, usually, somatic mutations, copy number variations, and other omics (epigenomics, proteomics) [29]. Somatic mutations are generally represented in a matrix alike gene expression, where a row corresponds to a gene and a column to a sample. However, in this case, each entry indicates the absence of mutation or the type of mutation present in the gene. Normally, these somatic mutation matrixes are binarized (presence/absence of an alteration).

2.3.2 Simplified Molecular-Input Line Entry System

Chemists have different strategies to represent molecules, being the International Union of Pure and Applied Chemistry (IUPAC) nomenclature the most used. How-

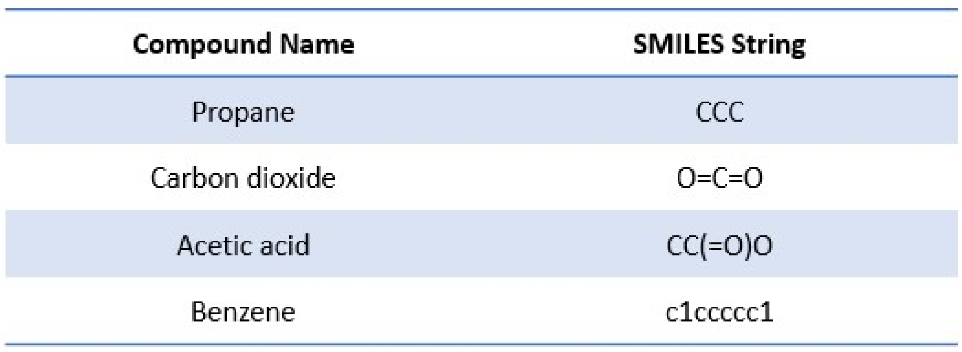
[1]Figure adapted from https://geneviatechnologies.com/bioinformatics-analyses/rna-seq-data-analysis/.

ever, this notation cannot be interpreted by computational methods. To overcome this problem, notation systems, particularly SMILES, have been developed to able the representation of molecules independent of the computer system in use [30].

Simplified Molecular-Input Line Entry System (SMILES) is a line notation for describing the structure of chemical compounds through short ASCII strings. This is a non-unique representation, i.e., for each molecule can exist several SMILES strings depending on the atom that started the encoding process. Nonetheless, it has the advantage of being an extremely compact notation once it does not contain 2D or 3D atom coordinates [31]. The encoding process follows a set of basic rules and conventions [32]:

- Atoms are represented by their atomic symbols;
- Single, double, triple, and aromatic bonds are represented by '-', '=', '#', and ':', respectively;
- Branches are specified in parentheses;
- Cyclic structures are represented by breaking a bond in any order and numbering it, followed by the remaining atoms and the repetition of the same number to designate ring closure. Atoms from aromatic rings are written in lower case;
- Implicit hydrogen atoms can be omitted;
- Explicit hydrogen atoms and charges are specified inside square brackets.

Figure 2.4 presents some examples of chemical compounds and their corresponding SMILES string.



Compound Name	SMILES String
Propane	CCC
Carbon dioxide	O=C=O
Acetic acid	CC(=O)O
Benzene	c1ccccc1

Figure 2.4: Example of chemical compounds and respective SMILES strings.

2.4 Review of Machine Learning Approaches

Regarding the context problem, most of the machine learning approaches are supervised due to the large drug screening datasets and their capacity to learn patterns among the data and make new predictions based on the extracted knowledge [33]. Ideally, the data used to train anticancer drug response prediction should come from patient cohorts however, the majority of the available data is from cell-line screening experiments. On this account, the prediction of drug response is still a challenge, as cell lines may not be representative of original tumors. Nevertheless, clinical data can be used to evaluate or refine the prediction models, improving their applicability to tumors [15].

In 2012, the National Cancer Institute (NCI) and the Dialogue on Reverse Engineering Assessment and Methods (DREAM) project presented a community challenge to predict drug response based on a cohort of genomic, epigenomic, and proteomic profiles measured in human breast cancer cell lines. The group concluded that gene expression was the most informative dataset in different approaches, but the combination of expression data with other omics data can enhance prediction performance. Similarly, further studies incorporate different types of genomics data applying ML methods, like SVM, to improve performance [34].

Iorio et al. (2016) integrated different combinations of molecular data, namely gene expression, somatic mutations, copy number alterations, and DNA methylation, to predict drug response by elastic nets and random forests models. They concluded that cell lines can portray the oncogenic alterations presented in tumors, and many of these alterations are associated with drug response [35].

Huang et al. (2017) used SVM combined with a recursive feature elimination approach to predict drug responses based on gene expression profiles. The model was trained on gene expression and drug response data from the National Cancer Institute 60 Human Cancer Cell Line Screen (NCI-60). This study revealed that the accuracy of the drug response related to a particular cancer type increased when the model was built on data from a variety of cancer types instead of using just the information about the same cancer type. This assessment reinforces the growing evidence that the information for a specific cancer drug response may not be in the tissue of origin. However, the mechanisms by which genomic signatures are shared across tissues remain unknown [36].

2.5 Review of Deep Learning Approaches

Most drug response prediction workflows have been developed based on deep learning, investigating different types of architectures, in the past few years. Unlike many traditional ML algorithms, DL can learn lower-dimensional representations of the input data without prior feature selection and has been shown to achieve better performance.

The DeepDSC model uses stacked deep autoencoders for dimensionality reduction and to predict drug sensitivity. First, an AE encodes the gene expression data which characterize the cancer cell lines. These encoded features are then merged with pre-computed molecular fingerprints and fed into a fully connected predictive network. DeepDSC was trained and evaluated on the GDSC dataset, and the authors found that the model's performance decreased drastically when testing with unseen drugs, showing that the model does not generalize well [37].

DeepDR is a deep learning model to predict drug response based on expression and mutation profiles of cancer cell lines and tumors. The model is composed of three DNNs: first, a pre-trained expression encoder and a mutation encoder, both using data retrieved from The Cancer Genome Atlas (TCGA); then, a drug response predictor that integrates the previous two networks. The model was trained on cell line data from CCLE and drug response data, measured by log scale of IC_{50} values, from GDSC. The DeepDR model achieved low mean square errors, close to 1.96, outperforming classical methods, like linear regression models and SVMs. Moreover, the model was applied to TCGA data to prove the applicability of DeepDR to tumors, and the authors demonstrated that DeepDR was able to identify novel resistance mechanisms underlying the drug response and new possible drug targets [38].

Another heterogeneous ensemble called MOLI is composed of three separated encoding subnetworks to learn representations from somatic mutations, CNVs, and gene expression profiles and a final subnetwork that concatenates all the features to classify the drug response of cell lines. MOLI was trained using GDSC data, and the authors observed that using multiple omics data was preferable to a single omics approach [39].

The tCNNS model is composed of two convolutional neural networks branches followed by a fully connected network. These separate CNNs learn representations from genomic features and compound data, represented by the one-hot encoding of SMILES strings. Distinct from the previously described methods, the two encoders are 1D convolutional networks that extract the features of the different types of data. The model obtained a value of R^2 around 0.826 when predicting drug response for unknown cell line-drug pairs. However, the model performed much worse when making predictions for unknown drugs [40].

Another model based on CNN is CDRscan which consists of an ensemble of five convolutional networks with different architectures. Four of them are a dual convergence model that performs a sequence of convolutions to each input data type (mutations on cancer cell lines and molecular fingerprints). Subsequently, the convoluted features are merged and, then another set of convolutions are applied to the data before predicting the IC_{50} values. The remaining model receives all the descriptors as input and they are convoluted together. CDRscan achieved higher performance compared to random forest and CNN, reflected in an MSE value of 1.14 and an R^2 value of 0.84. The authors demonstrated that the dual convergence models performed better and argued that the ensemble approach improves the generalizability of the model [41].

Nevertheless, in the last decade, emerged a deep learning technique, named attention mechanism, addressed to improve the performance of the encoder-decoder model for NLP and machine translation. Attention in deep learning can be interpreted as a vector of importance weights, i.e., the attention vector estimates how a pixel in an image or a word in a sentence is strongly correlated with other elements. The mechanism allows for the utilization of the most relevant parts of the input sequence by a weighted combination of all of the encoded input vectors, with the most relevant values being attributed to the highest weights [42].

This mechanism is now used in diverse problems like image captioning, and it has already been implemented to predict drug response. PaccMann is a DL model for drug response prediction that uses a multimodal attention-based convolutional encoder. This model receives gene expression data and SMILES strings of compounds to predict the IC_{50}. PaccMann is constituted by a gene expression encoder, a compound encoder, and a final prediction network. The gene expression encoder incorporates an attention mechanism that generates weights according to the relevance of the input features. Concerning the compound encoder, the authors studied different architectures and concluded that the attention encoders achieved the best performance. Also, they show that attention-based SMILES encoders outperformed a feedfoward model using Morgan fingerprints [43].

Manica et al. used attention-based encoders that act on gene expression profiles to focus on genes that are more relevant for IC_{50} prediction. In addition, they demonstrated that SMILES attention-encoders performed better than other architectures like DNNs. So the authors explored different models using attention mechanisms on GDSC data, and the multiscale convolutional attention (MCA) model surpass previously reported results. This model is composed of three parallel channels of convolutions over SMILES strings and another channel to operate on the token level. To each channel is added a gene attention layer and, then, the processed SMILES and filtered genes are fed into a multihead of contextual-attention layers. Finally, the outputs are concatenated and pass through a set of dense layers to predict the IC_{50}. Although achieving good performance, the range of the predicted values was narrowed to a small set, and the majority of the tests were done with normalized IC_{50} values, which means that the lower error is due to a smaller interval scale [44].

In a similar way, Zuo et al. implemented a deep learning model based on multitasking and self-attention to predict drug response from cancer genomic signatures and compound data. The model includes the application of Graph Neural Network (GNN) to extract drugs' features into continuous vectors and a convolutional neural network to encode gene expression and mutations profiles simultaneously. Then, is applied the self-attention mechanism to integrate the structural similarity into the vector of gene signature and, lastly, the drug vector and the genetic vector are concatenated to predict the IC_{50}. The model was trained with genomic data from CCLE and drug response data from GDSC and outperformed other models by obtaining a RMSE of 1.16 and a R^2 of 0.73 [45].

3

Deep Learning Models

This work comprises different deep learning models; accordingly, it is fundamental to describe them in detail. Thus, this chapter starts by explaining the mechanism of autoencoders and their application. Afterward, the core principles of RNNs are presented adjoining specific architectures like Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU). Additionally, the fundamentals of Convolutional Neural Networks (CNN) are explained. Ultimately, the Fully Connected Neural Network (FCNN) is introduced as the model applied to predict the IC_{50}.

3.1 Autoencoders

An autoencoder is an unsupervised neural network that consists of an encoder and a decoder. The encoder is a network that maps high-dimensional input data into a lower-dimensional vector (denoted by latent or context vector). On the other hand, the decoder is responsible for reconstructing the original data from this lower dimension representation. The goal of the training is to minimize the loss between the original input x and the reconstructed output $\hat{x}$ (Figure 3.1) through backpropagation. Therefore, the autoencoder finds the weights for both networks by comparing the input to its reconstruction without needing target labels [46].

The context vector represents a compression of the original data, and it can be seen as an encoding of the input that extracts the most informative features from the high dimensional space [3]. Thereby, the lower the dimensionality of the latent space, the lower the quality of the reconstruction. It is still important to note that the available data should match the research goal once the model does not work for any possible input, i.e., it works with data that belong to the same distribution as the training data. So, it learns how to efficiently compress data and reconstruct this reduced encoded representation to a representation as close to the original input as

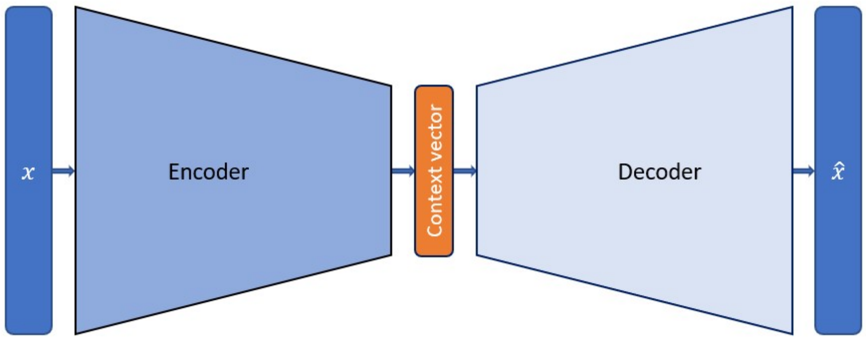


Figure 3.1: Schematic diagram of an autoencoder.

possible [47].

A traditional approach for dimensionality reduction is Principal Component Analysis (PCA). This technique is an orthogonal linear transformation of data into a lower-dimensional space such that the squared reconstruction error is minimized. Although having a similar objective to PCA, autoencoders often obtain a superior reconstruction due to the non-linearity activation functions [46].

3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks designed to process sequential data, such as speech, language, and molecules. These networks examine an input sequence one element at a time, taking into account the history of all the past elements to predict future outputs [26].

RNNs have loops that can keep a sort of memory to allow the information to persist throughout time. This memory is kept in a vector designated as the hidden state h_t, which stores the states of previous inputs to generate the next output of the sequence. Therefore, in each time step t, the output $\hat{y}_t$ depends not only on the current input x_t but also on the previous hidden state h_{t-1}. Additionally, the RNN cell produces an updated hidden state, h_t, that will be included in the following time step. The mathematical formalism that describes the RNN mechanism is represented in Equations 3.1 and 3.2 [48]:

$$h_t = \sigma_1(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{3.1}$$

$$\hat{y}_t = \sigma_2(W_{yh}h_t + b_y) \tag{3.2}$$

where σ_1 and σ_2 represent activation functions, generally, tangent and softmax, respectively; W_{hx}, W_{hh}, and W_{yh} are weight matrices (input-to-hidden, hidden-to-hidden, and hidden-to-output, respectively), and b_h and b_y are bias vectors.

The left side of Figure 3.2 represents an RNN with a single recurrent layer, whereas the right side shows the unfolding of an RNN. The unfolded representation enables the visualization of how the network processes a complete sequence and how the hidden states are updated at each time step [26]. Compared to other ANNs, the RNNs reduce their complexity since the layers are independent, i.e., they do not memorize the previous outputs. Besides, weights and biases are the same for all layers, and therefore these layers can be joined into a single recurrent layer, decreasing the number of training parameters. It is also worth noting that this architecture is able to process inputs of different sizes, and the size of the input does not increase the size of the model [49].

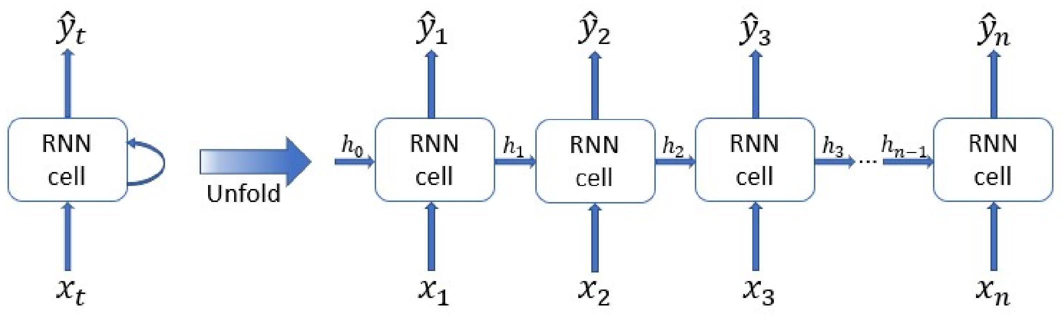


Figure 3.2: Schematic diagram of a recurrent layer and the respective unfolding representation.

In order to train a network, the loss should be computed and then backpropagated. In this specific case, the loss function $\mathcal{L}$ corresponds to the sum of the losses at each time step, and the backpropagation is done at each point in time (backpropagation through time) by differentiating the loss $\mathcal{L}$ concerning weight matrix W as expressed in Equation 3.3 [50].

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial W_t} \tag{3.3}$$

RNN is distinguishable from other types of ANN by being a simple and powerful model capable of processing sequential data, however, in practice, it has difficulties handling long-term dependencies. In fact, there is empirical evidence showing that it is hard to store information for very long [26], which can lead to exploding and vanishing gradients. The vanishing gradients arise when the gradients become exponentially smaller near zero, making it impossible to train the network. On the other side, exploding gradients occur when there is a large increase in the gradients during training. To tackle this problem, were proposed other networks, such as LSTMs and GRUs, specifically to avoid these problems [48].

3.2.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are a form of RNN designed to avoid problems that involve long-term dependencies. These architectures comprise a blockchain with gates that help control information flow. The key to LSTMs is the cell state that acts as "memory" by allowing information from previous intervals to be stored within the LSTM cell. The gate units decide when to keep or override information in the memory cell through a sigmoid neural net layer and a pointwise operation, which can be a sum or a multiplication [51]. Each sigmoid layer outputs a vector of values in the range [0,1], where a value of zero means that no information is retained, while a value of one means all the information is kept. Figure 3.3 demonstrates the structure of the "memory" blocks that compose the LSTM and how they interact [52].

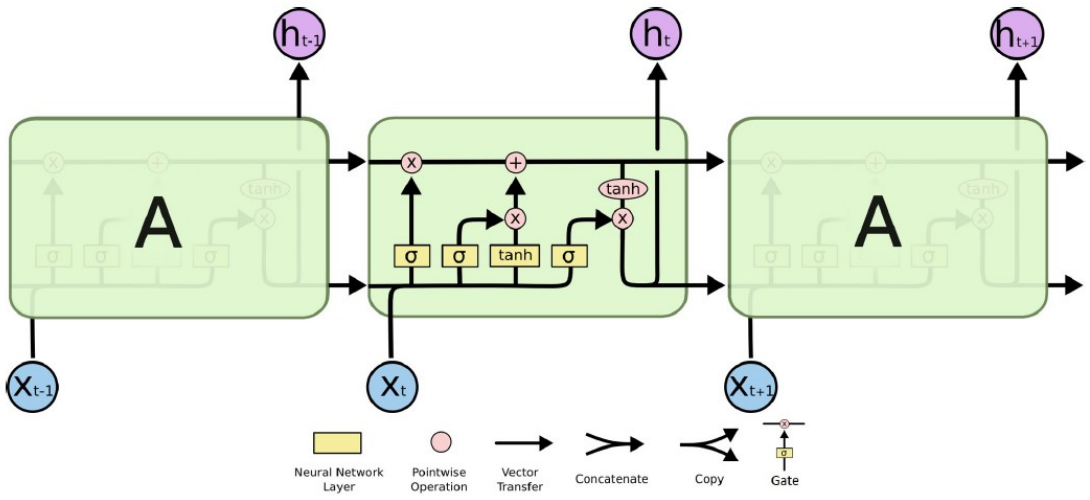


Figure 3.3: LSTM architecture [1].

[1]Figure from [52].

The cell state C_t can be described as the global memory of the LSTM network over all time steps. Therefore, it is dependent on three things: the long-term memory of the network, i.e., the previous cell state C_{t-1}, the output at the previous point in time, called the hidden state h_{t-1}, and the input data at the current time step x_t [53]. Figure 3.4 represents the cell state that enables a continuous flow of information without vanishing gradients problems.

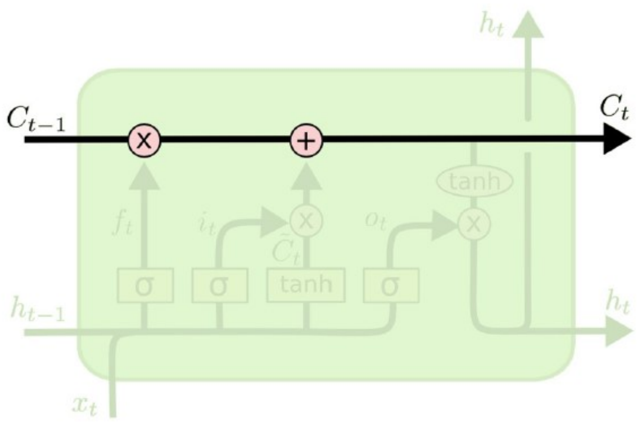


Figure 3.4: LSTM cell state [2].

An LSTM cell comprises four steps that are supported by Equations 3.4 to 3.9 [52].

- **Forget gate** (Figure 3.5a): the forget gate is the first step, and it determines which information is relevant from the previous hidden state and from the current input. This network generates a vector that is passed through a sigmoid function and then pointwise multiplied with the previous cell state.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.4}$$

- **Store** (Figure 3.5b): this step, which includes two phases, decides what new information should enter the cell state, given the previous hidden state and new input data. First, similarly to the forget gate, the input gate has a sigmoid activation function that acts as a filter and identifies which components of the "new memory vector" are worth retaining. Second, the input data is passed through a *tanh* layer that creates a new vector $\tilde{C}_t$ which is composed of new candidates that could be added to the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.5}$$

[2]Figure from [52].

$$\tilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.6}$$

- **Update** (Figure 3.5c): the old state C_{t-1} is multiplied by f_t (first term of Equation 3.7) to forget the information that was defined in step 1. Then, the output of step 2 is pointwise multiplied, and the combined vector is added to the cell state (second term of Equation 3.7), resulting in the long-term memory of the network C_t being updated.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t} \tag{3.7}$$

- **Output gate** (Figure 3.5d): the output will be based on the newly updated cell state, the previous hidden state, and the new input data. First, h_{t-1} and x_t are passed through the sigmoid-activated neural network to obtain a filtered vector. Second, a tanh layer is applied to the newly updated cell state. Finally, the vectors resulted from these two processes are pointwise multiplied to output the new hidden state h_t.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.8}$$

$$h_t = o_t * tanh(C_t) \tag{3.9}$$

3.2.2 Gated Recurrent Units

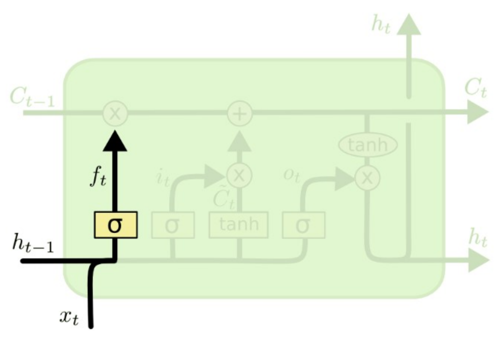
The above mechanism describes the most general version of LSTMs, however, over the years, other variants of this architecture have been proposed. One of the most successful approaches is the Gated Recurrent Units (GRU), which is simpler due to having fewer training parameters. Instead of three gates, GRU replaces the forget and input gates with reset and update gates and merges the cell state and the output gate. Therefore, the hidden state h_t is the only output and is determined by the following expressions (Equations 3.10 to 3.13):

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{3.10}$$

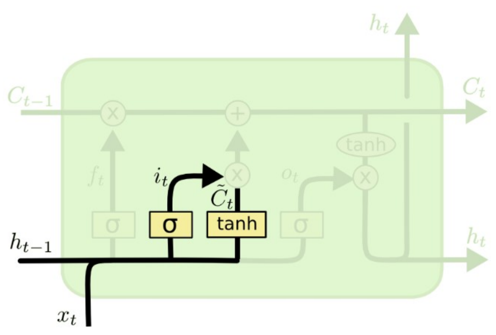
$$\tilde{h_t} = tanh(W \cdot [r_t * h_{t-1}, x_t] + b) \tag{3.11}$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{3.12}$$

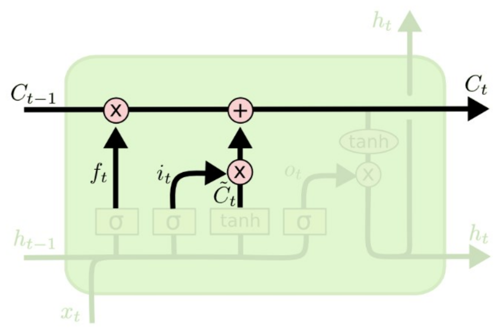
[3]Figure from [52].



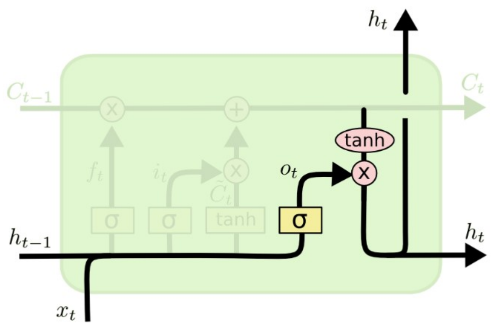
(a) Forget gate.



(b) Store.



(c) Update of the cell state.



(d) Output gate.

Figure 3.5: Steps to update an LSTM cell [3].

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{3.13}$$

The reset gate decides the amount of past information from the previous hidden state that should be forgotten. Similar to LSTMs, a sigmoid function filters this information generating a vector r_t, which values belong to the interval between 0 and 1. Then, the new cell state $\tilde{h}_t$ is obtained from vector r_t. The update gate determines the amount of previous information, i.e., from previous time steps, that needs to pass along to the next state. So, the gate outputs a vector z_t which is combined with the cell state $\tilde{h}_t$ to obtain the final hidden state h_t. Figure 3.6 represents the diagram of GRU architecture and the respective flow of information [54].

3.3 Convolutional Neural Network

Convolutional neural networks (CNNs) are a type of neural network that can detect patterns from data composed of multiple arrays. There are three main types of CNNs: 1D for signals and sequences; 2D for images [55] and voice recognition; and 3D for video or volumetric images.

[4]Figure from [52].

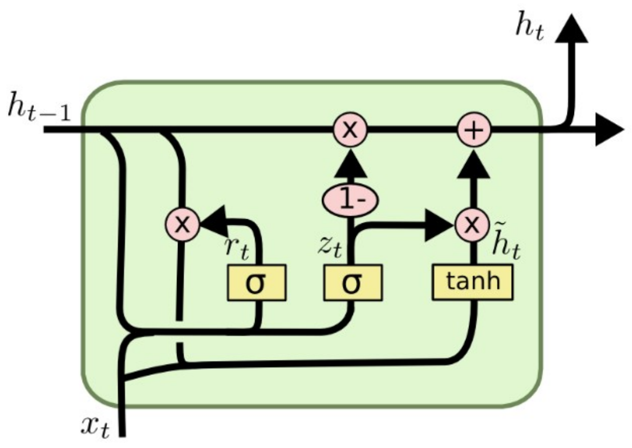


Figure 3.6: GRU architecture [4].

Figure 3.7 presents the basis of a CNN architecture. The convolutional layers are the main layers of this architecture, and their objective is to condense the input by extracting relevant features and producing feature maps used as input in the following layer. In the first convolutional layer, neurons filter simple features like edges. The neurons in the next layers learn to compile the information to obtain a bigger picture of the input, detecting high-order features [56].

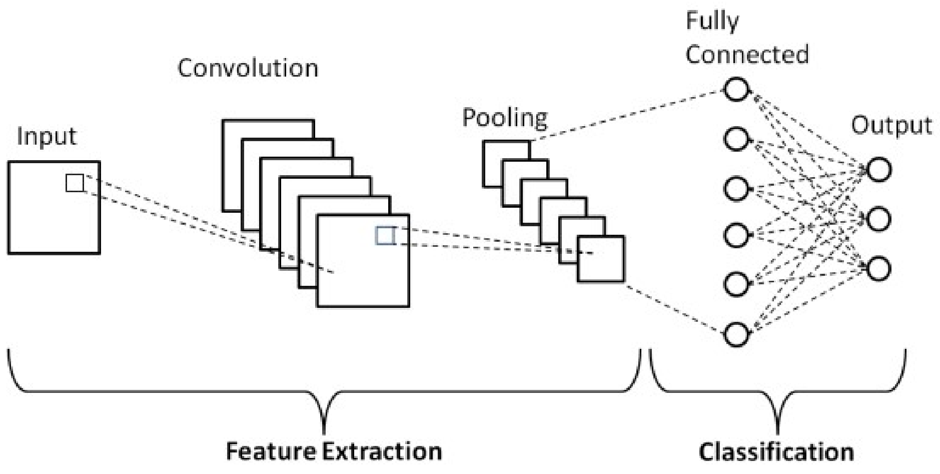


Figure 3.7: CNN architecture [5].

The basic units that constitute the convolutional layers are filters, which correspond to arrays of weights that spread along with the entire input. When the data

[5]Figure from https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697.

encounter a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a feature map. The final number of feature maps corresponds to the number of filters in the layer [57]. Initially, the weights of the filters are randomized, but during training are updated according to the objective.

Convolution is the key operation of this network, and its described as an element-wise product between each element of the kernel and each location of the input tensor, followed by the sum of all the results (Figure 3.8). Afterward, an activation function, e.g., ReLU, is applied on every element of the output of the convolution layer [56].

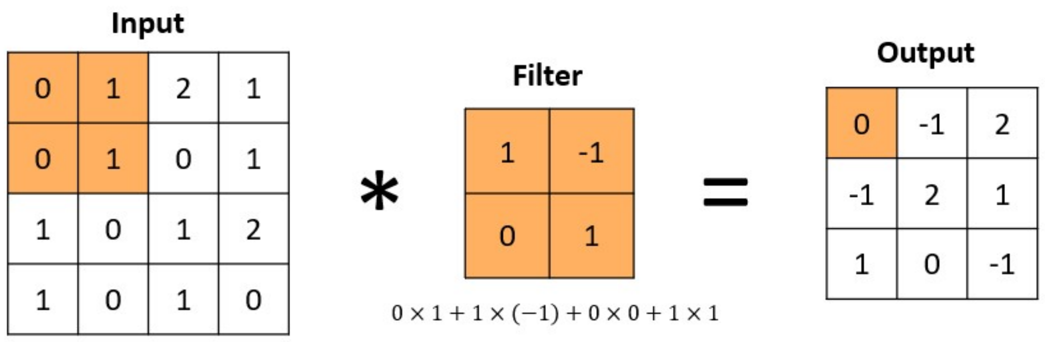


Figure 3.8: Convolution operation with zero-padding and stride equals to 1.

The output size of each feature map can be given by:

$$Output\ size = \frac{Input\ size - Filter\ size + 2 * Padding}{Stride} + 1 \tag{3.14}$$

The distance between two successive filter positions is designated as a stride, which also characterizes the convolution operation. Usually, this value is 1 in both directions, however, a stride superior to 1 can be used in order to reduce the samples of the feature maps.

Depending on the problem, padding and pooling are other hyperparameters that could be set before the training process starts to improve the model's performance. Padding, typically zero-padding, is a simple process of adding zeros on the borders of the input to give further control of the dimensionality of the output volumes [57]. On the other hand, pooling is an alternative technique to perform the downsampling of the feature maps. Pooling reduces width and height by substituting a specific location with the result from an operation performed on the nearby values. Max-pooling and average-pooling are the most used pooling approaches in CNNs, where is extracted the maximum and the average value, respectively, within a selected neighborhood area of the feature map. This process allows decreasing the computational power required to process the data and extracting dominant features with a translation invariance to small shifts and distortions. It is worth noting that, contrary to height and width, the depth dimension of feature maps remains unchanged [56]. The output of a pooling layer is given by the following expression.

$$Output\ pooling = \frac{Input\ size - Pool\ size}{Stride} + 1 \tag{3.15}$$

Figure 3.9 represents the pooling operation, which reduces the spatial size of each feature map by extracting a representative feature (maximum or mean). The resulting feature vector is then used as the input of an FCNN architecture.

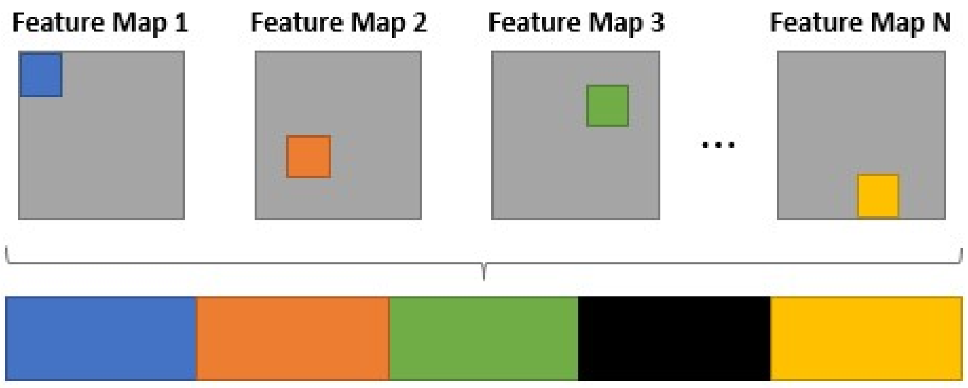


Figure 3.9: Pooling operation and the resulting feature vector.

3.4 Fully Connected Neural Network

Fully Connected Neural Networks (FCNNs) are analogous to traditional forms of ANNs, where all the neurons are interlinked, i.e., each node in a fully-connected layer is directly connected to every node in both the previous and the next layers (Figure 3.10).

[6]Figure from https://www.gabormelli.com/RKB/Fully-Connected_Neural_Network

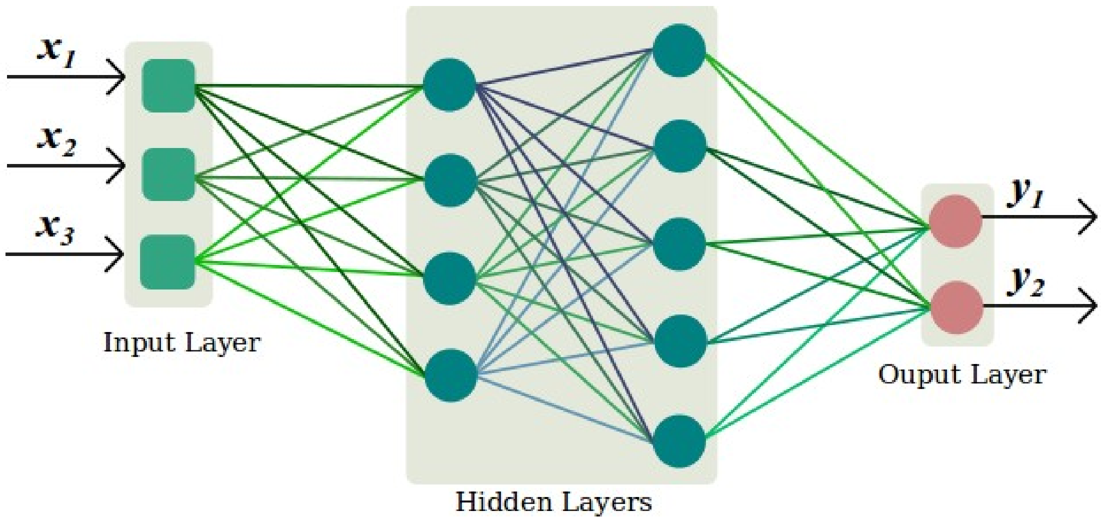


Figure 3.10: Fully Connected Neural Network Architecture with 2 hidden layers [6].

Due to the considering number of nodes and connections, the FCNN comprises many parameters that require complex computational. Therefore, the dropout technique removes nodes and connections to obtain a simpler network [58].

3.5 Regularization Techniques

Underfitting occurs when the model performs poorly on the training data and on the test set. On the other side, overfitting occurs when the model performs well on the training data but does not perform well on the test set [59]. So, regularization techniques are a set of methods used to prevent overfitting by improving the generalization of the model and, consequently, reducing the generalization error. On the other hand, a simpler model can lead to underfitting. Therefore, it is crucial to choose the appropriate complexity in the model. The following are the most commonly used regularization techniques:

- **Dropout**: Dropout is a form of regularization that randomly drops some nodes in a fully connected layer (Figure 3.11), i.e., the contribution of the node to the corresponding activation function is set to 0, and, consequently, the gradients for these nodes drop to zero as well [56]. This method ensures that the network learns a more robust set of features that perform equally well with random subsets of the node selected. So, the network can learn a better-generalized mapping from input to output, enabling the reduction of overfitting [60].

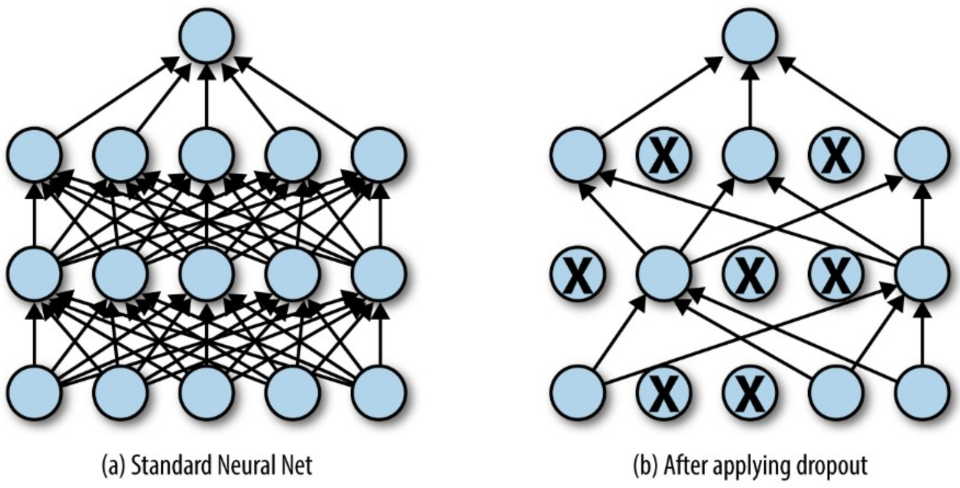


Figure 3.11: Dropout randomly removes neurons from a network while training [7].

- **Batch Normalization**: The goal of this technique is to overcome a phenomenon designated as internal covariate shift, which describes the change in the distribution of network activations due to the change in network parameters during training [61]. So, batch normalization normalizes the input values of the following layer, improving gradient flow through the network, avoiding exploding or vanishing gradients, and preventing overfitting.
- **Early Stopping**: Early stopping allows for interrupting the training process if there is no improvement of the evaluation metric in a pre-chosen number of epochs. Accordingly, early stopping requires a validation set, therefore, some samples are not fed into the model, decreasing the size of the training data. However, it has the advantage of requiring almost no changes in the training procedure. Whenever the error on the validation set improves, is stored a copy of the model parameters, so once the training algorithm finishes, it is possible to recover the best set of parameters [62].
- **Reduce Learning Rate on Plateau**: This technique reduces the learning rate once it hit a plateau, i.e., the point when the change in loss is less than a threshold θ. Learning rate is one of the most critical hyperparameters. While increasing the learning rate allows the model to learn faster, it may result in a non-optimal final set of weights. However, a smaller learning rate may allow the model to acquire an optimal set of weights, but it will take too long to

[7]Figure from https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html

train. So, this callback, as represented in Figure 3.12, is designed to reduce the learning rate when the model stops improving with the hope of fine-tuning model weights [63].

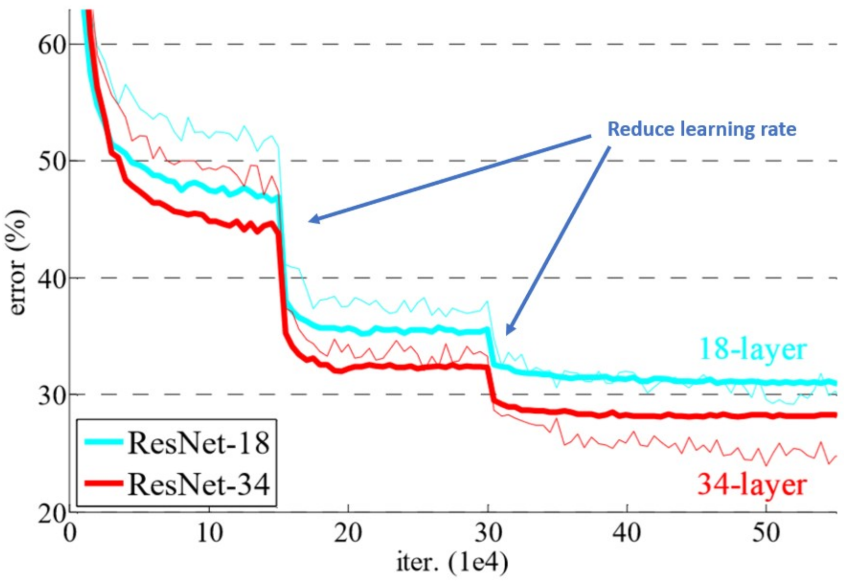


Figure 3.12: Reduce learning rate on plateau [8].

[8]Figure adapted from https://cntk.azurewebsites.net/pythondocs/Manual_How_to_use_learners.html

4

Methods

This chapter presents a general overview of the proposed drug response framework. This work can be divided into two main steps. First, two autoencoders were implemented to apprehend the genetic features of tumors. Both are previously trained so that the knowledge can be transferred to a prediction model whose goal is to accurately predict the half-maximal inhibitory concentration. Therefore, in the second step, two types of deep neural networks were explored: Recurrent Neural Networks and Convolutional Neural Networks. The recurrent model resorts to LSTM and GRU cells, while the convolution model comprises a series of convolution layers. Both models are trained using cancer cell line features and SMILES strings as input data. Finally, the effect of the different hyperparameters on the efficiency of the models is evaluated.

4.1 Data Preprocessing

Gene Expression and Gene Mutation

There are many procedures to decode phenotypic information from RNA-sequencing (RNA-seq) concerning the gene expression data. In this study, we analyze two types of gene expression quantification, namely TPM and RSEM. TPM stands for transcripts per million; in other words, if TPM equals 4 for gene BRCA, for every 1,000,000 RNA molecules in the RNA-seq sample, four molecules came from that gene [64]. On the other hand, RSEM represents RNA-Seq by Expectation-Maximization and is a software tool that has proven to achieve better performance than other quantification methods. This algorithm consists of two steps, first calculates expected read counts given current expression levels and then computes expression values maximizing likelihood given expected read counts [65]. Thus, the expression data was normalized according to the quantification method, $log_2(TPM + 1)$

for TPM data, and a z-score transformation was applied to RSEM data.

The mutation data is constituted by Mutation Annotation Format (MAF) files, which integrate the discovered or validated mutations and the respective category of these mutations. We only considered four types of non-synonymous mutations, particularly missense and nonsense mutations and frameshift insertions and deletions, which have proven to be the best candidates for pathogenic variants [66]. Missense mutations and nonsense mutations are characterized by coding a different amino acid and interrupting protein synthesis before it is completed. Frameshift mutations are caused by a nucleotide deletion or insertion in a DNA sequence that shifts the grouping for all downstream amino acids, usually resulting in a nonfunctional protein [67]. In this way, the mutation data was expressed in binary matrices, where 0 represents wild-type, and 1 represents a mutation.

Drugs

SMILES strings were used as the drugs' input data, each composed of a unique set of characters representing the chemical structure. Therefore, each character is considered a feature, and in order to have the same amount of features, is defined a threshold for the length of the SMILES strings. Accordingly, this threshold was set to 90.

There are different ways of encoding categorical data, but in the present work, embedding was used to convert SMILES strings. The preprocessing of this technique is represented in Figure 4.1 and comprehends three steps: tokenization, padding, and encoding. Tokenization involves converting data into unique characters (tokens) that characterize all the essential information based on a dictionary with all the existing tokens. Then, the padding of the SMILES strings assures that all sequences have the same size. So, SMILES that have a size smaller than the pre-defined threshold are padded with a character until they reach the same size, in this case, character 'A'. Finally, the SMILES strings are transformed into an encoded vector. In this type of encoding, each token is converted to the correspondent index in the previously constructed dictionary. It is important to note that atoms composed of two letter symbols (e.g. Br and Cl) and sections of SMILES enclosed in brackets (e.g. [N+] and [C@@H]) were considered as a single token.

The embedding layer works as a lookup table that transforms each encoded token into a fixed-length dense vector, resulting in a matrix of size |(dictionary length) x (embedding dimension)|. The embedding values are trainable parameters, i.e.,

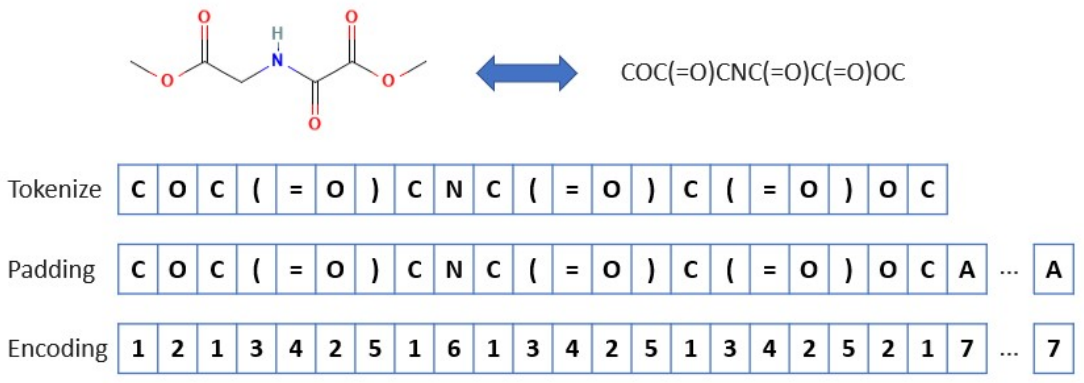


Figure 4.1: Encoding of the compound Dimethyloxalylglycine.

weights initially random that allow the model to obtain its own representation for each token. So, as represented in Figure 4.2, the output of the embedding layer is a matrix of size |(input data length) x (embedding dimension)| that is given as input to the following layers [68]. This approach is particularly interesting because it can reduce the dimensionality over one-hot encoding as it enables the control of the number of features, and it is capable of learning the context of a token so that similar tokens have similar embeddings [69].

$$\underbrace{[1 \quad 2 \quad 1 \quad 3 \quad 4]}_{\text{Encoded Vector}} \times \underbrace{\begin{bmatrix} 8 & 2 & 1 & 9 \\ 6 & 5 & 4 & 0 \\ 7 & 1 & 6 & 2 \\ 1 & 3 & 5 & 8 \\ 0 & 4 & 9 & 1 \end{bmatrix}}_{\text{Embedding Weight Matrix}} = \underbrace{[30 \quad 38 \quad 66 \quad 39]}_{\text{Embedding Output}}$$

Figure 4.2: An encoded vector is passed through an embedding layer, i.e., each encoded drug is converted into a fixed length vector of defined size.

4.2 Autoencoders Module

The autoencoders modules are adopted to map a high-dimensional space into a lower-dimensional representation. In the present study, each cell line was repre-

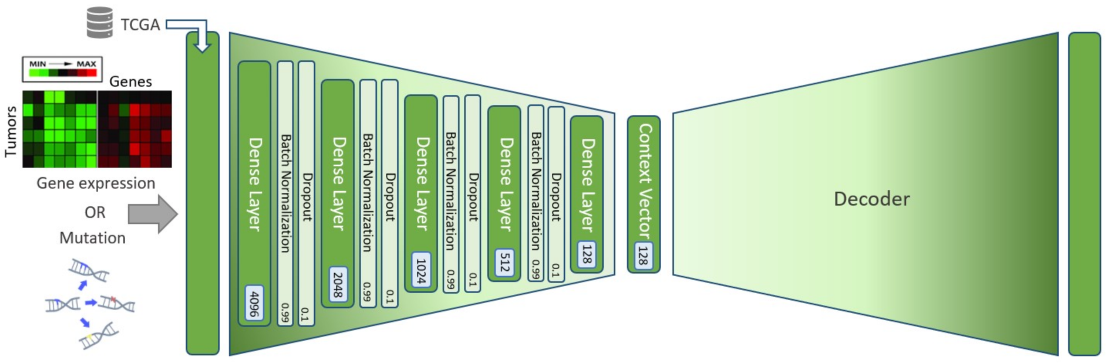


Figure 4.3: The general framework of the autoencoders with mutation and expression profiles of TCGA dataset to extract crucial features from high-dimensional data.

sented with more than a thousand genes, resulting in a considerable amount of data. Therefore, it can be obtained a reduced dimension of complex data with the crucial features captured from the original observations by using autoencoders. In this manner, two autoencoders were pre-trained with TCGA expression and mutation data to acquire high-order features. The encoder contains five dense layers, each followed by batch normalization and dropout, to make the model more flexible and the latent vector more robust. For the decoder, the context vector serves as input to a set of layers that are symmetric to the encoder's layers. The autoencoders are trained using the mean squared error (MSE) between the input and the predicted output as the loss function, and the NN weights are then updated using the Adam optimizer.

Figure 4.3 shows the architecture of the proposed autoencoder applied to train with gene expression and mutation data separately. This model is used to recover the weights that are expertise in the features of this domain. Later, this information is applied to novel data to improve the prediction task so that the predictor can have a better starting point than from scratch.

4.3 RNN Prediction Model

Figure 4.4 illustrates the structure of the proposed prediction model that consists of three modules: two encoders to apprehend the genetic features of the cell lines and an RNN architecture to capture the SMILES strings' characteristics of the drugs. First, we implemented expression (Figure 4.4a) and mutation encoders (Figure 4.4b)

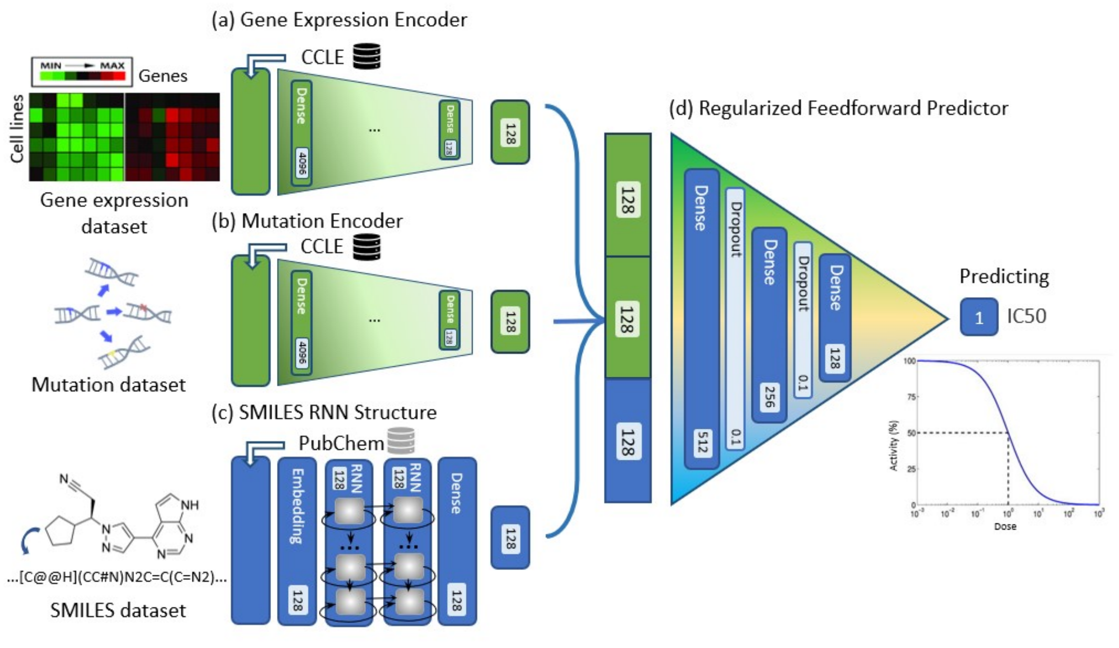


Figure 4.4: The general workflow of the proposed prediction model. (a) Gene expression encoder. (b) Mutation encoder. Both encoders were trained with CCLE data and initialized with the weights of the pre-trained autoencoders. (c) SMILES RNN structure: an RNN architecture is used to capture the relevant features of SMILES strings from the PubChem dataset. (d) Regularized Feedforward Predictor ingests the genetic's and drug's deep representations to predict the IC_{50} value that demonstrates the correlation between a cell line and a drug.

that were initialized with the weights of the pre-trained autoencoders. Thus, the two encoders have the same architecture as the encoders from the autoencoders with the best performance. In their training data, we included the CCLE dataset.

Furthermore, in order to incorporate the relevant information that underlies the structure of each drug, was implemented an approach based on RNNs and SMILES notation (Figure 4.4c). The architecture of the RNN is suitable for processing sequential data once it possesses memory stems that are able to take into account the previous data to predict future outputs. Thereby, two types of RNN layers were tested, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). LSTM is used to solve problems that involve long-term dependencies and, typically, is more accurate on a larger dataset. On the other hand, GRU uses fewer training parameters and accordingly uses less memory and executes faster than LSTM.

Therefore, first, the drugs' SMILES strings shorter than the maximum length were padded. Then a dictionary-based approach was considered to encode the SMILES strings into integers based on the number of different tokens, resulting in a 34-token dictionary. After this processing, the dictionary-encoded SMILES string is passed through an embedding layer, followed by two RNN layers and one dense layer.

The resulting deep representations from expression and mutation encoders and RNN are then concatenated into a single feature vector, combining the most relevant sequential and structural features, and used as input for the regularized feed forward neural network (Figure 4.4d). Between dense layers is introduced a dropout layer to prevent overfitting. Finally, an output layer is added, which is composed of one neuron that returns the real-valued impact of genetic variants on a given drug measured in $log_{10}(IC_{50})$.

4.4 CNN Prediction Model

Furthermore, a model based on convolutional neural networks was built to explore the applicability of this type of architecture in the prediction of the half-maximal concentration. Figure 4.5 exhibits the framework of the proposed prediction model that, similar to the previous model, consists of three modules: two encoders to capture the genetics beyond the cell lines and a CNN architecture to extract the SMILES strings' characteristics of the drugs. First, the weights of the pre-trained autoencoders were fed into the new expression and mutation encoders, analogous to the RNN-based architecture, and included the CCLE dataset as their training data. After passing through the encoders, the resulting deep representations, once they are both related to the genes that are more relevant, were multiplied to compose a feature map describing the genetics of each cell line. At that point, the feature map is fed into two convolutional layers to uncover deep patterns present in the genetics. Finally, a pooling layer is applied to reduce the spatial size of each feature map to its maximum or mean representative feature, depending on the type of pooling used.

Moreover, to incorporate the relevant information of each drug, it was implemented an approach based on SMILES notation and, also, on CNNs. Once again, first, the drugs' SMILES strings shorter than the maximum length were padded, and a dictionary-based approach was considered to encode the SMILES strings. Afterward, the dictionary-encoded SMILES string is passed through an embedding layer. The deep representation obtained from this layer is seen as a feature map that is fed into two CNN layers, followed by a pooling layer.

The resulting deep representations from both pooling layers are then concatenated

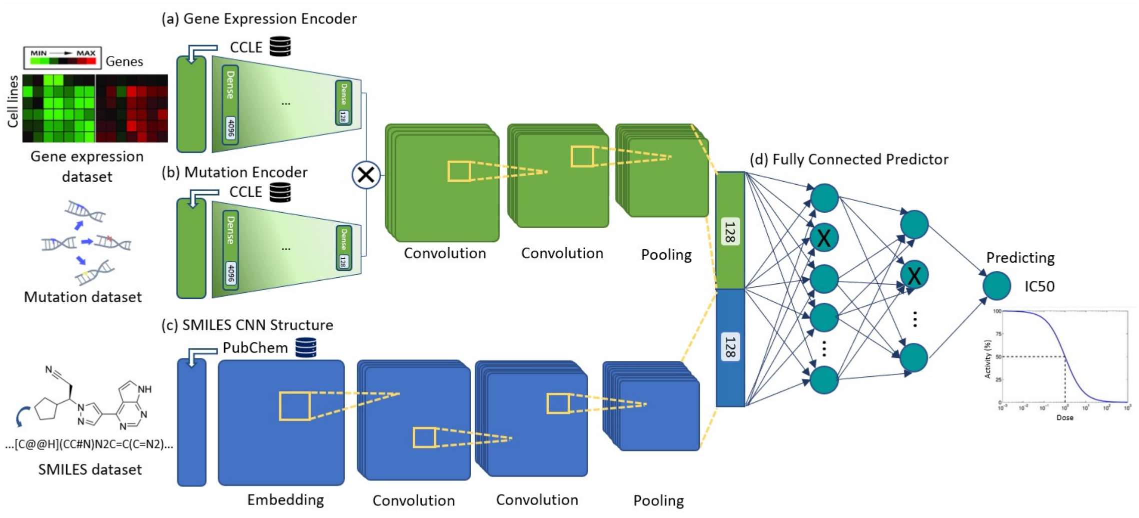


Figure 4.5: The general workflow of the proposed prediction model. (a) Gene expression encoder. (b) Mutation encoder. Both encoders were trained with CCLE data and initialized with the weights of the pre-trained autoencoders. (c) SMILES CNN structure: a CNN architecture is used to capture the relevant features of SMILES strings from the PubChem dataset. (d) Fully Connected Predictor ingests the genetic's and drug's deep representations to predict the IC_{50} value that demonstrates the correlation between a cell line and a drug.

into a single feature vector, combining the most relevant features of the input data, and used as input for the fully connected neural network. Between dense layers is introduced a dropout layer to prevent overfitting. At last, an output layer is added, which is composed of one neuron that returns the half-maximal concentration in $log_{10}(IC_{50})$.

4.5 Evaluation Metrics

Different metrics can be used to evaluate the model's performance. The choice of the metrics depends on the context of the problem and the target of the model. In this case, the model aims to predict a set of continuous values, and therefore it is necessary to use metrics designed to evaluate regressions. It is worth noting that, in some applications, looking at a single metric can be misleading, and finding a trade-off is essential to have a whole picture of the problem.

For performance comparison, the following metrics were used:

- **Mean Square Error** (MSE): average squared error between the predicted and actual values.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{4.1}$$

- **Root Mean Square Error** (RMSE): square root of mean squared error.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2} \tag{4.2}$$

- **Coefficient of determination** (R^2): measures how well observed outcomes are replicated by the model.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y_i})^2}{\sum(y_i - \overline{y})^2} \tag{4.3}$$

, where y_i represents the true values and $\hat{y_i}$ the predicted values.

5

Results and Discussion

This chapter starts with a description of all data used to train, validate and test both autoencoders and the prediction models. Then, the performed experiments to optimize the autoencoders are presented along with the obtained results. Furthermore, after finding the best parameters that guarantee the implementation of the autoencoders, some optimization experiments were carried out to explore the efficiency of the prediction models and the respective results are stated. Nonetheless, both models, the one based on RNN and the other on CNN, need to be able to accurately predict the half-maximal inhibitory concentration in order to validate its performance and therefore establish a relation between the drug and the genetic variant.

5.1 Datasets

The dataset used to pre-train the two autoencoders includes 36 pan-cancer tumors belonging to The Cancer Genome Atlas (TCGA) program. Thus, the TPM data along with the tumors' mutations were collected from the UCSC TumorMap [70], and the RSEM values and the respective mutations from the Firebrowse database [71]. Meanwhile, the omics data used to train the predictor belong to the Cancer Cell Line Encyclopedia (CCLE) database and was downloaded from CTD^2 Data Portal [72], and again both expression normalization were collected.

To establish the correlation between drugs and the cell lines, we also obtained drug response measured across 996 cell lines with 265 anticancer drugs by the half-maximal inhibitory concentration (IC_{50}) from the Genomics of Drug Sensitivity in Cancer (GDSC) Project. The IC_{50} was measured in µM and is represented in log scale. Finally, the SMILES strings were extracted from PubChem based on the compound nomenclature presented in the GDSC database [73].

Overall, in this study, we analyzed 604 cell lines with available expression, mutation, and IC_{50} data related to 224 drugs, resulting in 109,337 trios of mutation, expression, and SMILES vectors. The range of the $log_{10}(IC_{50})$ was from -9.00 to 12.8 with a standard deviation of 2.49.

Both datasets, i.e., TCGA and CCLE, utilize a large set of genes (over 10,000) to define each tumor and cell line, respectively. As a result, utilizing the complete set of genes required a huge computational effort and a more complex model. Hence, we decided to select the most informative genes among the 2128 genes explored in [44] to overcome this problem. After narrowing the data to these genes, we obtained 1954 genes for RSEM data and 1428 for TPM data.

5.2 Experimental Analysis and Results

5.2.1 Autoencoders

This section describes the experimental analysis and the grid search strategy that was employed to find the best architecture and set of parameters for both autoencoders (gene expression and mutation). This model aims to convert the gene expression and mutation profiles into latent space vectors and reconstruct them correctly. The smaller the mean squared error, the better the model's generalization.

The optimized architecture for this model was determined using five-fold cross-validation. The dataset is divided into 85% for training/validation and 15% for testing. Then, the training/validation data is divided into five folds to train the same amount of models, and these models are ultimately evaluated with an external test set. In order to evaluate the effect of each hyperparameter, several experiments were conducted with 5 epochs of 5-fold validation in each epoch. Rectified Linear Unit (ReLU) was chosen as the activation function for each dense layer, except the final output layer that uses a linear activation function. Regarding the optimizer function, Adam was used to adjust the network weights in each iteration of the training process, and the selected loss function was the mean squared error (MSE).

Table 5.1 shows the results obtained when keeping all the hyperparameters fixed except the number of layers and the respective hidden neurons of each layer. The batch normalization momentum was set to 0.99, the dropout to 0.2., and the batch size to 64. As can be noted, in each case, mutation autoencoder reflects the lowest error as they capture features from binary data, while gene expression autoencoders cover a range of values from -1.78 to 37.25. However, the model with five layers outperformed the model with six layers. For this reason, five encoder layers were used for the following experiments.

Table 5.1: Results for different number of layers and hidden neurons - Autoencoders (Batch size=64, Batch normalization momentum=0.99, Optimizer=Adam, Dropout=0.2).

Type of Data	**No. of Layers**	**Hidden neurons**	**MSE Exp**	**RMSE Exp**	**MSE Mut**	**RMSE Mut**
RSEM	6	[4096,2048,1024,512,256,128]	0.6287	0.7929	0.0086	0.0926
	5	**[4096,2048,1024,512,128]**	0.6065	0.7788	0.0072	0.0848
TPM	6	[4096,2048,1024,512,256,128]	0.4385	0.6622	0.0058	0.0764
	5	**[4096,2048,1024,512,128]**	0.3921	0.6262	0.0062	0.0788

Regarding the number of batch sizes, the results of the experiments are shown in Table 5.2 from which it can be concluded that a batch size of 64 produces a model with better generalization capabilities.

Table 5.2: Results for different number of batch size - Autoencoders (Dense layers=5, Batch normalization momentum=0.99, Optimizer=Adam, Dropout=0.2).

Type of Data	**Batch size**	**MSE Exp**	**RMSE Exp**	**MSE Mut**	**RMSE Mut**
RSEM	32	0.6266	0.7916	0.0116	0.1079
	64	0.6065	0.7788	0.0072	0.0848
	128	0.6371	0.7982	0.0097	0.0983
TPM	32	0.4141	0.6435	0.0087	0.0932
	64	0.3921	0.6262	0.0062	0.0788
	128	0.4571	0.6761	0.0085	0.0920

After setting the batch size to 64, the effect of using different dropout rates (0.1, 0.2, and 0.3) was evaluated. According to Table 5.3, a dropout value of 0.1 returned the best results in the test set.

After finding the best hyperparameters, both autoencoders were trained using 80% for training, 10% for validation, and 10% for testing, and the models were saved to apply later to the predictor. The number of epochs was set to 100, however reduce learning rate on plateau and early stopping were used to prevent overfitting.

Table 5.3: Results for different number of dropout - Autoencoders (Dense layers=5, Batch size=64, Batch normalization momentum=0.99, Optimizer=Adam).

Type of Data	Dropout	MSE Exp	RMSE Exp	MSE Mut	RMSE Mut
RSEM	**0.1**	0.5957	0.7718	0.0067	0.0820
	0.2	0.6065	0.7788	0.0072	0.0848
	0.3	0.6508	0.8067	0.0097	0.0983
TPM	**0.1**	0.3890	0.6237	0.0049	0.0699
	0.2	0.3921	0.6262	0.0062	0.0788
	0.3	0.4107	0.6409	0.0063	0.0791

5.2.2 Prediction Model - RNN

The primary purpose of the first set of tests realized regarding the autoencoders is to analyze and evaluate the set of parameters that obtains the lowest error in the reconstruction of genetic information. Therefore, transferring the weights of the encoders to the prediction model can be seen as the starting point for the training process followed by their update in response to this new problem.

Similarly to the previous section, it is presented a description of the experimental analysis and the grid search strategy that was employed to find the best architecture and set of parameters for the prediction model based on RNNs. This model aims to predict the half-maximal inhibitory concentration through the gene expression and mutation profiles of the cell lines and the SMILES strings that represent the drugs. The smaller the mean squared error and the higher the coefficient of determination, the better the model's generalization.

The optimized architecture for this model was determined using 80% for training, 10% for validation, and 10% for testing. In order to evaluate the effect of diverse hyperparameters, several experiments were conducted with 100 epochs. Once again, Rectified Linear Unit (ReLU) was chosen as the activation function for each layer, except for the final output layer that uses a linear activation function. The model was trained using the Adam Optimizer, and the loss function was the mean squared error (MSE). Furthermore, early stopping with the patience of 10 and reducing the learning rate on plateau were also applied to avoid too-tight adjustment for the training data. Too many epochs can result in overfitting, whereas too few may lead to an underfitted model. Thus, it is possible to adopt a large number of epochs because the training process finishes when the model's performance does not improve in the validation subset.

The first experiments performed comprise keeping all the hyperparameters fixed except for the type of recurrent neural network, i.e., Long Short Term Memory or Gated Recurrent Units. Table 5.4 shows the results obtained when training the model with a number of RNN Units of 128, a batch size of 128, and a dropout rate of 0.1. The decoder comprises 3 layers with the units 512, 256, and 128, in the order specified.

Table 5.4: Results for different types of RNN layers - RNN model (RNN Units=128, Decoder layers=[512, 256, 128], Batch Size=128, Dropout rate=0.1).

Type of Data	**Type of RNN**	**MSE**	**RMSE**	$\boldsymbol{R^2}$
RSEM	LSTM	1.2326	1.1102	0.7791
	GRU	1.0952	1.0465	0.8036
TPM	LSTM	1.2024	1.0965	0.7836
	GRU	1.1247	1.0605	0.7990

The results revealed that the architecture of GRU yielded better results than the LSTM and, therefore, the type of RNN was defined as GRU in the following experiments. Regarding the number of GRU units, the results are shown in Table 5.5 from which it can be concluded that 128 units produce the model with better efficiency.

Table 5.5: Results for different number of RNN units - RNN model (Type of RNN=GRU, Decoder layers=[512, 256, 128], Batch Size=128, Dropout rate=0.1).

Type of Data	**RNN Units**	**MSE**	**RMSE**	$\boldsymbol{R^2}$
RSEM	64	1.2360	1.1118	0.7793
	128	1.0952	1.0465	0.8036
	256	1.1040	1.0507	0.8014
TPM	64	1.1530	1.0739	0.7931
	128	1.1247	1.0605	0.7990
	256	1.2297	1.1089	0.7798

After setting the number of RNN units to 128, was evaluated the effect of using a different number of layers in the decoder. According to Table 5.6, hidden units of 512, 256, and 128, respectively, returned the best results.

Table 5.6: Results for different number of decoder layers and respective units - RNN model (Type of RNN=GRU, RNN Units=128, Batch Size=128, Dropout rate=0.1).

Type of Data	**Decoder Units**	**MSE**	**RMSE**	$\boldsymbol{R^2}$
RSEM	**[512,256,128]**	1.0952	1.0465	0.8036
	[256,128]	1.1425	1.0886	0.7869
TPM	**[512,256,128]**	1.1247	1.0605	0.7990
	[256,128]	1.1874	1.0897	0.7870

Afterward, the impact of the number of batch sizes was also evaluated and the results are summarized in Table 5.7. The model with a batch size of 128 returned the best performance.

Table 5.7: Results for different number of batch size - RNN model (Type of RNN=GRU, RNN Units=128, Decoder layers=[512, 256, 128], Dropout rate=0.1).

Type of Data	**Batch Size**	**MSE**	**RMSE**	$\boldsymbol{R^2}$
RSEM	64	1.1121	1.0546	0.8013
	128	1.0952	1.0465	0.8036
	256	1.1459	1.0704	0.7954
TPM	64	1.1253	1.0608	0.7980
	128	1.1247	1.0605	0.7990
	256	1.1286	1.0623	0.7969

As a final experiment, the effect of using different dropout rates: 0.05, 0.1, and 0.2 was evaluated and presented in Table 5.8. As expected, the higher the dropout rate, the lower the performance of the model once a higher dropout rate introduces a significant amount of noise in the data.

In summary, despite having a higher error in autoencoders' performance, the model achieves better performance when using RSEM expression data instead of TPM data. These results demonstrate that the additional noise with dropout and batch normalization layers can produce a more robust model once a higher error in autoencoders does not correspond to the worst performance in the prediction model.

Figure 5.1 summarizes the obtained results for the RNN predictor. The x-axis shows the real IC_{50} values, and the y-axis shows the corresponding predictions. The goal is

Table 5.8: Results for different number of dropout rates - RNN model (Type of RNN=GRU, RNN Units=128, Decoder layers=[512, 256, 128], Batch Size=128).

Type of Data	Dropout	MSE	RMSE	R^2
RSEM	**0.05**	**1.0753**	**1.0370**	**0.8025**
	0.1	1.0952	1.0465	0.8036
	0.2	1.2636	1.1241	0.7773
TPM	**0.05**	1.1089	1.0530	0.8006
	0.1	1.1247	1.0605	0.7990
	0.2	1.3575	1.1651	0.7592

to have the points as close as possible to the diagonal line (perfect model). Thus, the dispersion of the points close to the diagonal line confirms the model's capacity to extract representative features of mutations, gene expression, and SMILES strings.

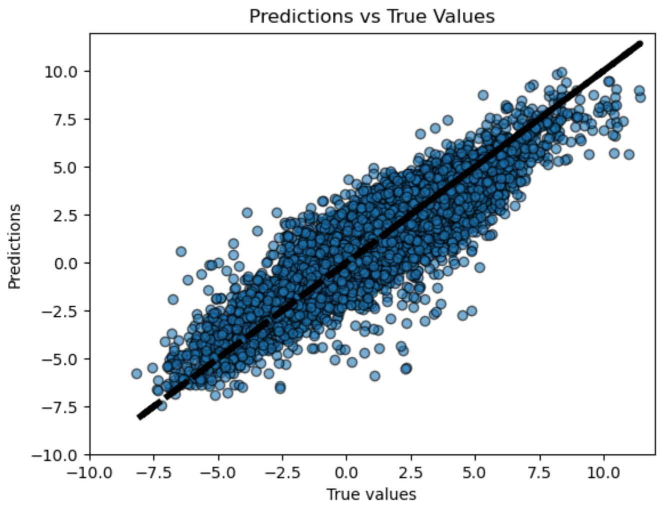


Figure 5.1: Predictions of $log_{10}(IC_{50})$ from the proposed model with RNNs against the true values for the testing set, where the diagonal line is the reference line (predicted = true value).

5.2.3 Prediction Model - CNN

After implementing the prediction model based on RNNs and finding the best hyperparameters, a new prediction model based on CNN was built to explore this type of architecture and try to surpass the previous results. The model aims to, once again, accurately predict the IC_{50} using gene expression, gene mutation, and

SMILES vectors. The main difference is that the CNN layers will see the data as feature maps instead of a sequence of values.

Below is described the experimental analysis and the grid search approach that was implemented to find the best architecture and set of parameters for the prediction model based on CNNs. The metrics used to evaluate the model were the mean squared error, the root mean squared error and the coefficient of determination.

The architecture for this model was determined using 80% for training, 10% for validation, and 10% for testing. Different tests were conducted with 100 epochs, however early stopping with the patience of 10 and reducing the learning rate on plateau were also applied. The model was trained using the Adam Optimizer, and the loss function was the mean squared error (MSE).

Initially, the experiments realized kept all the hyperparameters fixed except for the CNN filters in the CNN layers. Table 5.9 shows the results obtained when training the model with a filter length of 4 and 5, in this order, a batch size of 128, and a dropout rate of 0.05. The type of pooling chosen was the max-pooling.

Table 5.9: Results for different number of CNN filters - CNN model (Decoder dense layers=[256, 128], Filter length=[4,5], Type of pooling=max, Batch size=128, Dropout=0.05, Activation function=ReLU).

Type of Data	**CNN filters**	**MSE**	**RMSE**	$\boldsymbol{R^2}$
RSEM	[32,64]	1.3034	1.1417	0.7667
	[64,128]	1.2723	1.1279	0.7716

Afterward, the effect of the number of filter lengths was also evaluated and the results are summarized in Table 5.10. The model with a filter length of 4 in both CNN layers returned the best performance.

Table 5.10: Results for different number of filters' length - CNN model (Decoder dense layers=[256, 128], Filters=[64,128], Type of pooling=max, Batch size=128, Dropout=0.05, Activation function=ReLU).

Type of Data	**Filter length**	**MSE**	**RMSE**	$\boldsymbol{R^2}$
RSEM	**[4,4]**	1.1261	1.0612	0.7974
	[4,5]	1.2723	1.1279	0.7716
	[4,7]	1.2048	1.0977	0.7833

Moreover, was tested the impact of the type of pooling in the model's performance. Max-pooling and average-pooling were used, and the first one yielded better results. In this way, this parameter was set to 'max' in the following experiments.

Table 5.11: Results for different types of pooling - CNN model (Decoder dense layers=[256, 128], Filters=[64,128], Filters' length=[4,4], Batch size=128, Dropout=0.05, Activation function=ReLU).

Type of Data	Type of pooling	MSE	RMSE	R^2
RSEM	**Max**	1.1261	1.0612	0.7974
	Average	1.1423	1.0688	0.7938

Similar to the previous model, the effect of using different batch sizes: 64, 128, and 256, was evaluated and presented in Table 5.12. From this table, it can be concluded that a batch size of 128 outperforms the other values, and consequently, the parameter was fixed with the value of 128.

Table 5.12: Results for different number of batch size - CNN model (Decoder dense layers=[256, 128], Filters=[64,128], Filters' length=[4,4], Type of pooling=max, Dropout=0.05, Activation function=ReLU).

Type of Data	Batch Size	MSE	RMSE	R^2
	64	1.3117	1.1453	0.7656
RSEM	**128**	1.1261	1.0612	0.7974
	256	1.2212	1.1051	0.7808

Furthermore, the dropout rate was also evaluated, and the increase in this value led to a higher error in the model's performance, as presented in Table 5.13.

Table 5.13: Results for different number of dropout rates - CNN model (Decoder dense layers=[256, 128], Filters=[64,128], Filters' length=[4,4], Type of pooling=max, Batch size=128, Activation function=ReLU).

Type of Data	Dropout	MSE	RMSE	R^2
RSEM	**0.05**	1.1261	1.0612	0.7974
	0.1	1.1423	1.0688	0.7938

Finally, the activation function was varied to determine which one, ReLU or Leaky ReLU, returned the best performance. The results are shown in Table 5.14, and Leaky ReLU outperformed the ReLU function.

Table 5.14: Results for different types of activation functions (Decoder dense layers=[256, 128], Filters=[64,128], Filters' length=[4,4], Type of pooling=max, Batch size=128, Dropout=0.05).

Type of Data	Activation Function	MSE	RMSE	R^2
RSEM	ReLU	1.1261	1.0612	0.7974
	Leaky-ReLU	**1.0557**	**1.0275**	**0.8092**

Overall, the CNN approach was more successful as the mean squared error decreased and the coefficient of determination increased. Figure 5.2 resumes the obtained results for the CNN predictor. The x-axis represents the real IC_{50} values, and the y-axis represents the corresponding predictions. The dispersion of the points close to the diagonal line corroborates the model's performance.

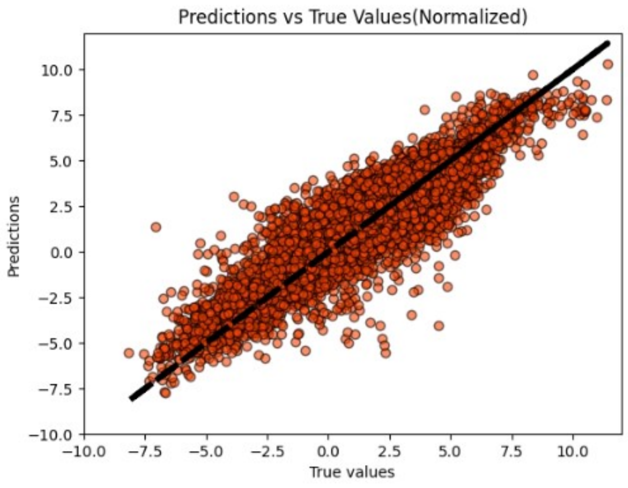


Figure 5.2: Predictions of $log_{10}(IC_{50})$ from the proposed model with CNNs against the true values for the testing set, where the diagonal line is the reference line (predicted = true value).

5.3 Comparison between RNN and CNN Prediction Models

In this work, it is presented two deep models with the same goal: to predict anticancer drug response. The objective was to see how these types of architectures could be used to tackle the problem. The overall frameworks demonstrated the efficiency of using RNNs and CNNs to extract deep representations over global descriptors. The results confirm that the introduction of information regarding chemical compounds reflects an increase in performance compared to DeepDR. Meanwhile, it is worth noting that although obtaining a good value of R^2, the MSE and RMSE are relatively high due to the wide range of IC_{50} values that the model covers. Nonetheless, the main goal was to validate the effectiveness of using structural and sequential data to extract meaningful deep representations of these types of data in the prediction of the impact of genetic variants on a given drug. On that account, considering the obtained results, we believe that the goal was fulfilled.

Table 5.15 presents a summarized comparison between the two proposed models regarding the implemented structure and the obtained results.

Table 5.15: Comparison between the two proposed models.

Parameter	Value	
	RNN model	**CNN model**
Hidden Neurons (Decoder)	[512,256,128]	[256,128]
Number of Layers	2 RNN layers	2 CNN layers
Optimizer	Adam	Adam
Activation Function	ReLU	Leaky ReLU
Activation Function (Output)	Linear	Linear
Batch size	128	128
Dropout	0.05	0.05
MSE	1.075	1.056
RMSE	1.037	1.028
R^2	0.803	0.809

There is one module that is common in both models concerning the encoding of gene expression and gene mutation. The main difference regarding this part is the series of convolutions applied to the encoded vectors in the CNN prediction model. In this way, once the gene mutation is correlated with the gene expression, it is possible to mix these two types of information which will result in one single vector with all the essential features.

With respect to the drugs' encoding, both models start with an embedding layer and the difference between them is the type of architecture applied after this layer which gives the name to the model. While RNNs have the ability to process temporal information, e.g., a sentence, CNNs are commonly used in solving problems related to spatial data. At first sight, it might seem that they are used to handle different problems, but it should be noted that some types of data can be processed by either architecture. An example of this is the field of drug discovery, where both systems have been effective [74, 75].

Finally, the decoder presents some different aspects. The major difference is the vector that is fed to the model. In the RNN prediction model, the vector is composed of three parts, one related to the gene expression, another to the gene mutation, and the last to the drugs. Whereas the CNN prediction model has a vector that can be divided into two parts, one related to the gene expression and gene mutation and the other with the drugs. In this way, the number of layers needed to decode the information differs in the two models, but the final goal is equivalent, which is to predict the IC_{50} value through a dense layer composed of one single neuron.

Overall, the two proposed prediction models surpassed previous state-of-the-art models, proving the applicability of both architectures in the pharmacogenomics field, specifically in anticancer drug response prediction.

Technical notes: We used Python 3.10.0 and Tensorflow [76] 2.8.0 to develop the proposed model. The GPU hardware used to train and sample the models was Nvidia RTX 3090 24GB of GDDR6 VRAM, using CUDA 11.2.

6 Conclusions

6.1 Summary

In this work, it was proposed and developed a deep learning approach for anticancer drug response prediction, capable of automatically extracting features (deep representations) from raw data: gene expression, gene mutation, and SMILES strings. In this way, we performed a comprehensive study on different architectures and their most common hyperparameters. First, two autoencoders were trained to capture the features of mutations and gene expression of the TCGA collection of tumors. Then, this knowledge is applied to the prediction models that use as input a combination of deep representations obtained from the RNNs/CNNs and deep descriptors representations from the genetic information. Our approach yielded better results, revealing the model's potential once it demonstrates a good correlation between predicted IC_{50} and the original data, surpassing other models' performances.

The novelty of our approach is to integrate the compound chemical structures and different types of omics data that typically are not considered together. By capturing the deep representations from a huge number of samples of TCGA data, we also increased the number of samples of the training data. Due to the advantage of the transferability of neural networks, the vast amount of TCGA data improves the model's performance by capturing representations of mutation and expression profiles of tumors that do not have an associated drug response.

Overall, the implemented models fulfilled the objective of predicting the drug response of cancer cell lines from integrated genomic profiles and compound structures achieving a mean squared error of 1.08 and 1.06, which surpasses previous state-of-the-art models.

6.2 Final Remarks

Deep learning has been demonstrating overall success in many prediction studies for its capacity to learn deep patterns from the data. Likewise, both models illustrate the remarkable ability to apply these approaches, specifically RNNs and CNNs, to automatically extract deep representations and use them to describe the impact of a genetic variant on a given drug. The obtained results showed that using these representations outperformed state-of-the-art models, demonstrating the importance of the extracted features and also the capacity to learn particular characteristics from cell lines and drugs meaningful to their interaction.

Regarding the RNN prediction model, it can be concluded that GRU cells clearly outperformed the competing types of RNN. Furthermore, the obtained results demonstrated that the use of RSEM data reflects a better model's efficiency, confirming that this information improves performance compared to other quantification methods. This model resulted in a mean squared error of 1.08 and a coefficient of determination of 0.80, proving that this is a viable way of predicting anticancer drug response.

Moreover, the construction of a model based on CNNs was developed to explore a different type of architecture in the context of this problem. The strategy applied by resorting to CNN layers surpassed the RNN approach by interpreting the data as a feature map instead of a sequence. After testing distinct hyperparameters the model achieved a mean squared error of 1.06 and a coefficient of determination of 0.81, demonstrating once again the model's capacity to predict drug response.

This work shows that deep learning models can be trained to extract relevant features from gene expression and mutation data, and SMILES strings, as the models obtained lower mean squared errors. Also was possible to conclude that the addition of SMILES strings benefits the prediction once the MSE decreased by 0.8 (compared to DeepDR). Besides, the model's potential is highlighted by a good correlation between the predicted IC_{50} and the original data.

In summary, the main contribution of this master thesis is the proposal of two different deep learning methods for anticancer drug prediction, solely based on the use of gene expression and mutation profiles and SMILES strings to represent the cell lines and drugs, respectively, without depending on complex feature engineering and extraction.

6.3 Future Work

One of the main drawbacks of the proposed frameworks is the limited datasets available regarding the drug response. On that account, would be interesting to validate the model on more diverse and representative datasets. So, increasing the data would increase the capacity of the model to learn and identify more hidden patterns meaningful to the impact of a cell line on a given drug. With the increasing methods and studies to define the genetic basis of human cancers, the available drug screening data have been escalating, opening the path for making even more accurate predictions.

Moreover, this model can be adapted to incorporate other omics data or different drug representations. Distinct researches have demonstrated that integrating more information can contribute to the correct prediction of the IC_{50}, e.g., adding the drug's structure led to an increase in the model's performance. Hence, joining new meaningful information, which could be integrated into the proposed deep learning model, could provide better results. Regarding the drugs, there are different possibilities for their representation, for example, fingerprints, or different deep learning methods to process this type of data like graph neural networks that are able to learn features from molecular graphs. Thence, exploring the use of different techniques could further validate the practicality of the model as well as be integrated into it.

Overall, the proposed model paves the way for future applications such as drug repositioning, as our model predicts drug sensitivity for any given pair of cell lines and drug vectors, enabling approved or investigational drugs to be used outside the original medical indication. Ultimately, the ability to move backward, i.e., verify which genes and SMILES strings have more influence in the drug response would have a substantial impact on the whole process. This information could exploit modifications on the compounds to achieve certain properties and even contribute to the designing of novel patient-specific compounds. Along these lines, personalized treatments can become the future of patient care.

Bibliography

[1] M. V. Relling and W. E. Evans, “Moving towards individualized medicine with pharmacogenomics,” *Nature*, vol. 429, pp. 464–468, 2004.

[2] K. Tomczak, P. Czerwińska, and M. Wiznerowicz, “The cancer genome atlas (TCGA): an immeasurable source of knowledge,” *Contemporary oncology*, vol. 19, pp. A68–A77, 2015.

[3] Y.-C. Chiu, H.-I. H. Chen, A. Gorthi, M. Mostavi, S. Zheng, Y. Huang, and Y. Chen, “Deep learning of pharmacogenomics resources: moving towards precision oncology.” *Briefings in bioinformatics*, vol. 21, no. 6, pp. 2066–2083, 2020.

[4] F. Iorio, T. A. Knijnenburg, D. J. Vis, G. R. Bignell *et al.*, “A landscape of pharmacogenomic interactions in cancer,” *Cell*, vol. 166, pp. 740–754, 2016.

[5] N. H. G. R. Institute, “Gene expression,” https://www.genome.gov/genetics-glossary/Gene-Expression, 2022.

[6] G. Prelich, “Gene overexpression: uses, mechanisms, and interpretation,” *Genetics*, vol. 190, no. 3, pp. 841–854, 2012.

[7] G. Orphanides and D. Reinberg, “A unified theory of gene expression,” *Cell*, vol. 108, no. 4, pp. 439–451, 2002.

[8] T. V. de Jong, Y. M. Moshkin, and V. Guryev, “Gene expression variability: the other dimension in transcriptome analysis,” *Physiol Genomics*, vol. 51, no. 5, pp. 145–158, 2019.

[9] P. Savas, Z. L. Teo, and S. Loi, *Gene Expression Analysis: Applications.* Springer New York, 2016, pp. 137–149.

[10] S. Lutz, C. Brion, M. Kliebhan, and F. W. Albert, “Dna variants affecting the expression of numerous genes in trans have diverse mechanisms of action and evolutionary histories,” *PLoS Genetics*, vol. 15, no. 11, p. e1008375, 2019.

[11] B. V. Chakravarthi, S. Nepal, and S. Varambally, “Genomic and epigenomic alterations in cancer,” *The American journal of pathology*, vol. 186, no. 7, pp. 1724–1735, 2016.

[12] I. Lea, M. Jackson, X. Li, S. Bailey, S. Peddada, and J. Dunnick, “Genetic pathways and mutation profiles of human cancers: site- and exposure-specific patterns,” *Carcinogenesis*, vol. 28, no. 9, pp. 1851–1858, 2007.

[13] M. B. M. A. Rashid, “Artificial intelligence effecting a paradigm shift in drug development,” *SLAS Technology*, vol. 26, no. 1, pp. 3–15, 2021, special Collection: Artificial Intelligence in Process Automation.

[14] W. E. Evans and M. V. Relling, “Pharmacogenomics: Translating functional genomics into rational therapeutics,” *Science*, vol. 286, no. 5439, pp. 487–491, 1999.

[15] D. Baptista, P. G. Ferreira, and M. Rocha, “Deep learning for drug response prediction in cancer,” *Briefings in Bioinformatics*, vol. 22, no. 1, pp. 360–379, 2020.

[16] G. W Caldwell, Z. Yan, W. Lang, and J. A Masucci, “The ic50 concept revisited,” *Current topics in medicinal chemistry*, vol. 12, no. 11, pp. 1282–1290, 2012.

[17] G. B. of Disease 2019 Cancer Collaboration, “Cancer Incidence, Mortality, Years of Life Lost, Years Lived With Disability, and Disability-Adjusted Life Years for 29 Cancer Groups From 2010 to 2019: A Systematic Analysis for the Global Burden of Disease Study 2019,” *JAMA Oncology*, vol. 8, no. 3, pp. 420–444, 2022.

[18] Y. Juhn and H. Liu, “Artificial intelligence approaches using natural language processing to advance ehr-based clinical research,” *Journal of Allergy and Clinical Immunology*, vol. 145, no. 2, pp. 463–469, 2020.

[19] P. Kaur, K. Krishan, S. K. Sharma, and T. Kanchan, “Facial-recognition algorithms: A literature review,” *Medicine, Science and the Law*, vol. 60, no. 2, pp. 131–139, 2020.

[20] T. Panch, P. Szolovits, and R. Atun, "Artificial intelligence, machine learning and health systems," *J Glob Health*, vol. 8, no. 2, p. 020303, 2018.

[21] B. Mahesh, "Machine learning algorithms - a review," *International Journal of Science and Research (IJSR)*, vol. 9, no. 1, pp. 381–386, 2019.

[22] K. Suzuki, *Artificial neural networks: methodological advances and biomedical applications.* BoD–Books on Demand, 2011.

[23] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2016.

[24] P. P. Gallego, J. Gago, and M. Landín, "Artificial neural networks technology to model and predict plant biology process," in *Artificial Neural Networks.* IntechOpen, 2011, ch. 10.

[25] A. Krogh, "What are artificial neural networks?" *Nature biotechnology*, vol. 26, no. 2, pp. 195–197, 2008.

[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[27] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

[28] D. J. Lockhart and E. A. Winzeler, "Genomics, gene expression and dna arrays," *Nature*, vol. 405, no. 6788, pp. 827–836, 2000.

[29] P. S. Reel, S. Reel, E. Pearson, E. Trucco, and E. Jefferson, "Using machine learning approaches for multi-omics data analysis: A review," *Biotechnology Advances*, vol. 49, p. 107739, 2021.

[30] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[31] D. C. Elton, Z. Boukouvalas, M. D. Fuge, and P. W. Chung, "Deep learning for molecular design—a review of the state of the art," *Molecular Systems Design & Engineering*, vol. 4, no. 4, pp. 828–849, 2019.

[32] D. C. I. Systems, "SMILES - A Simplified Chemical Language," https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html.

[33] G. Adam, L. Rampášek, Z. Safikhani, P. Smirnov, B. Haibe-Kains, and A. Goldenberg, “Machine learning approaches to drug response prediction: challenges and recent progress,” *npj Precision Oncology*, vol. 4, no. 1, p. 19, 2020.

[34] J. C. Costello, L. M. Heiser, E. Georgii, M. Gönen, M. P. Menden, and N. J. Wang, “A community effort to assess and improve drug sensitivity prediction algorithms,” *Nature Biotechnology*, vol. 32, no. 12, pp. 1202–1212, 2014.

[35] F. Iorio, T. A. Knijnenburg, D. J. Vis, G. R. Bignell, M. P. Menden, M. Schubert *et al.*, “A landscape of pharmacogenomic interactions in cancer,” *Cell*, vol. 166, no. 3, pp. 740–754, 2016.

[36] C. Huang, R. Mezencev, J. F. McDonald, and F. Vannberg, “Open source machine-learning algorithms for the prediction of optimal cancer drug therapies,” *PLoS One*, vol. 12, no. 10, p. e0186906, 2017.

[37] M. Li, Y. Wang, R. Zheng, X. Shi, Y. Li, F.-X. Wu, and J. Wang, “Deepdsc: A deep learning method to predict drug sensitivity of cancer cell lines,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 2, pp. 575–582, 2021.

[38] Y.-C. Chiu, H.-I. H. Chen, T. Zhang, S. Zhang *et al.*, “Predicting drug response of tumors from integrated genomic profiles by deep neural networks,” *BMC Medical Genomics*, vol. 12, p. 18, 2019.

[39] H. Sharifi-Noghabi, O. Zolotareva, C. C. Collins, and M. Ester, “MOLI: multi-omics late integration with deep neural networks for drug response prediction,” *Bioinformatics*, vol. 35, no. 14, pp. i501–i509, 2019.

[40] P. Liu, H. Li, S. Li, and K.-S. Leung, “Improving prediction of phenotypic drug response on cancer cell lines using deep convolutional network,” *BMC Bioinformatics*, vol. 20, no. 1, p. 408, 2019.

[41] Y. Chang, H. Park, H.-J. Yang, S. Lee *et al.*, “Cancer drug response profile scan (cdrscan): A deep learning model that predicts drug effectiveness from cancer genomic signature,” *Scientific Reports*, vol. 8, p. 8857, 2018.

[42] Z. Niu, G. Zhong, and H. Yu, “A review on the attention mechanism of deep learning,” *Neurocomputing*, vol. 452, pp. 48–62, 2021.

[43] A. Oskooei, J. Born, M. Manica, V. Subramanian, J. Sáez-Rodríguez, and M. R. Martínez, "Paccmann: Prediction of anticancer compound sensitivity with multi-modal attention-based neural networks," *arXiv*, 2018.

[44] M. Manica, A. Oskooei, J. Born, V. Subramanian, J. Sáez-Rodríguez, and M. Rodríguez Martínez, "Toward explainable anticancer compound sensitivity prediction via multimodal attention-based convolutional encoders," *Mol. Pharmaceutics*, vol. 16, pp. 4797–4806, 2019.

[45] Z. Zuo, P. Wang, X. Chen, L. Tian, H. Ge, and D. Qian, "Swnet: a deep learning model for drug response prediction from cancer genomic signatures and compound chemical structures," *BMC Bioinformatics*, vol. 22, p. 434, 2021.

[46] T. Blaschke, M. Olivecrona, O. Engkvist, J. Bajorath, and H. Chen, "Application of generative autoencoder in de novo molecular design," *Molecular Informatics*, vol. 37, no. 1-2, p. 1700123, 2018.

[47] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[48] J. Nabi. (2019) "Recurrent Neural Networks (RNNs)". [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

[49] Aishwarya. (2022) "Introduction to Recurrent Neural Network". [Online]. Available: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[50] A. Amidi and S. Amidi. (2019) "Recurrent Neural Networks cheatsheet". [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[51] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[52] Colah. (2015) "Understanding lstm networks". [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[53] R. Dolphin. (2020) "LSTM Networks — A Detailed Explanation". [Online]. Available: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

[54] S. Kostadinov. (2017) ”Understanding GRU Networks”. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[55] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, “A guide to deep learning in healthcare,” *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[56] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[57] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” 2015.

[58] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” Aug. 2017.

[59] W. M. Van der Aalst, V. Rubin, H. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther, “Process mining: a two-step approach to balance between underfitting and overfitting,” *Software & Systems Modeling*, vol. 9, no. 1, pp. 87–111, 2010.

[60] A. Anwar. (2021) ”Types of Regularization in Machine Learning”. [Online]. Available: https://towardsdatascience.com/types-of-regularization-in-machine-learning-eb5ce5f9bf50

[61] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015.

[62] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[63] A. Mohanty. (2019) ”The Subtle Art of Fixing and Modifying Learning Rate”. [Online]. Available: https://towardsdatascience.com/the-subtle-art-of-fixing-and-modifying-learning-rate-f1e22b537303

[64] Y. Zhao, M. Li, M. Konaté, C. Li, D. Biswajit, K. Chris *et al.*, “TPM, FPKM, or normalized counts? a comparative study of quantification measures for the analysis of rna-seq data from the nci patient-derived models repository,” *Journal of Translational Medicine*, vol. 19, p. 269, 2021.

[65] B. Li and C. N. Dewey, “RSEM: accurate transcript quantification from rna-seq data with or without a reference genome,” *BMC Bioinformatics*, vol. 12, p. 323, 2011.

[66] J. McCarthy and B. Mendelsohn, “Interpreting the pathogenicity of genetic variants,” in *Precision Medicine: A Guide to Genomics in Clinical Practice*. McGraw-Hill Education, 2016.

[67] N. L. of Medicine, “What kinds of gene variants are possible?” https://medlineplus.gov/genetics/understanding/mutationsanddisorders/possiblemutations/, 2021.

[68] S. Jaeger, S. Fulle, and S. Turk, “Mol2vec: Unsupervised machine learning approach with chemical intuition,” *J. Chem. Inf. Model.*, vol. 58, no. 1, pp. 27–35, 2018.

[69] S. Zheng, X. Yan, Q. Gu, Y. Yang, Y. Du, Y. Lu, and J. Xu, “QBMG: quasi-biogenic molecule generator with deep recurrent neural network,” *Journal of Cheminformatics*, vol. 11, no. 1, p. 5, 2019.

[70] Y. Newton, A. Novak, T. Swatloski, D. McColl, S. Chopra *et al.*, “TumorMap: exploring the molecular similarities of cancer samples in an interactive portal.” *Cancer Research*, vol. 77, pp. e111–e114, 2017.

[71] B. I. from MIT & Harvard, “Firebrowse,” http://firebrowse.org/, 2019.

[72] N. C. Institute, “CTD^2 data portal,” https://ocg.cancer.gov/programs/ctd2/data-portal, 2022.

[73] W. Yang, J. Soares, E. J. E. Patricia Greninger *et al.*, “Genomics of drug sensitivity in cancer (GDSC): a resource for therapeutic biomarker discovery in cancer cells.” *Nucleic Acids Research*, vol. 41, pp. D955–D961, 2013.

[74] N. R. Monteiro, C. J. Simões, H. V. Ávila, M. Abbasi, J. L. Oliveira, and J. P. Arrais, “Explainable deep drug–target representations for binding affinity prediction,” *BMC bioinformatics*, vol. 23, no. 1, pp. 1–24, 2022.

[75] T. Pereira, M. Abbasi, J. L. Oliveira, B. Ribeiro, and J. Arrais, “Optimizing blood–brain barrier permeation through deep reinforcement learning for de novo drug design,” *Bioinformatics*, vol. 37, no. Supplement_1, pp. i84–i92, 2021.

[76] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for Large-Scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.