GENE EXPRESSION INFERENCE WITH DEEP LEARNING

A Project Report Submitted in Partial Fulfilment
of the Requirements for the Award of the Degree of

Bachelor of Technology
in
Computer Science and Engineering

By
Oishik Mondal (Roll No.: 11100119007)
Samrat Mondal (Roll No.: 11100119019)
Ambar Nil Midya (Roll No.: 11100119033)
Sounak Saha (Roll No.: 11100119023)
Souhardya Halder (Roll No.: 11100719020)

Under the Supervision of

Dr. Shemim Begum

Department of Computer Science and Engineering
Government College of Engineering and Textile Technology,
Berhampore, Murshidabad-742101, India.

MAY 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GOVT. COLLEGE of ENGG. & TEXTILE TECHNOLOGY
BERHAMPORE, MURSHIDABAD-742101

CERTIFICATE

This is to certify that the thesis entitled, **"Gene Expression Inference with Deep Learning"** submitted by roll no.: 11100119007, 11100119019, 11100119033, 11100119023, 11100719020 in partial fulfilment of the requirements for the award of **Bachelor of Technology**

Degree in **Computer Science and Engineering** during 2019-2023 session at the Government College of Engineering and Textile Technology, Berhampore is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date

Assistant Professor
Department of Computer Science and Engineering
Govt. College of Engineering & Textile Technology
Berhampore, Murshidabad-742101

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude and appreciation to all those who have contributed to the successful completion of our final year project. Without their support, guidance, and encouragement, this endeavour would not have been possible.

Foremost, we extend our deepest thanks to our project mentor, Dr. Shemim Begum, for her invaluable guidance, constant support, and mentorship throughout the project. Their expertise, insights, and constructive feedback played a crucial role in shaping our work and ensuring its quality. We are grateful for their patience, dedication, and commitment to our project. We would like to extend our gratitude to all the participants and individuals who generously shared their time, expertise, and resources for this project. Their willingness to collaborate and provide valuable insights contributed significantly to the success of our research. In conclusion, we sincerely thank everyone who has contributed directly or indirectly to the completion of our final year project. Their support, guidance, and encouragement have been invaluable, and we are truly grateful for their contribution to our academic and personal growth.

[Date] **Students Names**

Oishik Mondal

Samrat Mondal

Ambar Nil Midya

Sounak Saha

Souhardya Halder

Contents

Chap-1 Introduction Page No

Chap-2 Literature Review Page No

Chap-3 Proposed Methodology Page No

Chap-4 Implementation Details and Results Page No

Chap-5 Conclusion and future work Page No

References Page No

Introduction

A microarray, also known as a DNA microarray or gene chip, is a powerful tool used in molecular biology and genetics to measure the expression levels of thousands of genes simultaneously. It allows researchers to analyse the activity of genes by detecting and quantifying the presence or absence of specific nucleotide sequences.

The microarray technology consists of a solid surface, typically a glass slide or silicon chip, onto which DNA or RNA molecules are immobilized in an organized grid or array. These DNA or RNA molecules, known as probes, are designed to be complementary to specific genes or gene fragments of interest. When a sample containing fluorescently labelled DNA or RNA molecules, called the target, is applied to the microarray, it hybridizes or binds to the complementary probes on the array.

The hybridization process is usually followed by detection of the bound targets using fluorescence or other methods. The intensity of the fluorescence signal at each spot on the microarray corresponds to the abundance or expression level of the corresponding gene in the sample. By comparing the signal intensities between different samples or conditions, researchers can identify genes that are differentially expressed, providing valuable insights into gene function, disease mechanisms, and drug discovery.

Microarrays are widely used in genomics research, including gene expression analysis, DNA sequencing, SNP genotyping, comparative genomic hybridization, and epigenetics studies. They have significantly contributed to our understanding of gene regulation, disease diagnostics, and personalized medicine. However, with advancements in sequencing technologies, such as next-generation sequencing, microarrays are now being complemented or replaced by more high-throughput methods for certain applications.

Microarray data refers to the information obtained from microarray experiments, which are powerful tools used in genomics research to study gene expression and genetic variations. Microarrays allow scientists to simultaneously measure the expression levels of thousands of genes or genetic sequences in a sample.

The characteristics of microarray data can vary depending on the specific experimental design and technology used, but here are some general features:

1. High-dimensional data: Microarray experiments often involve thousands to tens of thousands of genes or genetic features being simultaneously measured. As a result, microarray data is considered high-dimensional since it involves a large number of variables.

2. Gene expression levels: Microarray data primarily captures gene expression levels, which represent the activity of genes in a given sample. Gene expression levels are typically measured by quantifying the intensity of fluorescent signals emitted from labelled RNA or DNA molecules bound to the microarray chip.

3. Relative quantification: Microarray data provides relative quantification of gene expression rather than absolute measurements. Gene expression levels are usually compared across different samples or experimental conditions. This allows researchers to identify genes that are upregulated or downregulated in specific biological contexts.

4. Normalization: Microarray data often requires normalization procedures to correct for technical biases and variations in the experimental process. Normalization methods aim to ensure that the gene expression values are comparable across different samples and conditions, making it easier to identify true biological differences.

5. Noise and variability: Microarray data can be subject to various sources of noise and variability, such as experimental artifacts, batch effects, and biological variation. Careful statistical analysis and quality control measures are necessary to distinguish true signals from noise and account for sources of variability.

6. Data preprocessing: Microarray data typically undergoes preprocessing steps, including background correction, normalization, and data transformation, to enhance data quality and facilitate downstream analysis. These preprocessing steps can help remove systematic biases and improve the reliability of the results.

7. Dimensionality reduction: Due to the high-dimensional nature of microarray data, dimensionality reduction techniques are often employed to simplify the data and extract meaningful patterns. Techniques like principal component analysis (PCA) or clustering methods can help identify groups of genes or samples with similar expression profiles.

Understanding the characteristics of microarray data is crucial for appropriate data analysis and interpretation. Researchers utilize various statistical and computational methods to explore these data features and gain insights into biological processes and disease mechanisms.

- *Machine Learning Feature Selection Drawbacks:*

Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn and make predictions or decisions without being explicitly programmed. It involves designing and implementing algorithms that can automatically learn from and improve with experience or data.

Machine learning algorithms can be broadly categorized into three types: supervised learning, unsupervised learning, and reinforcement learning:

1. Supervised Learning: In supervised learning, the algorithm is trained on labeled data, where each data point is associated with a known output or target value. The algorithm learns to make predictions by finding patterns and relationships between the input variables and the corresponding output labels. Common algorithms used in supervised learning include decision trees, support vector machines (SVMs), and neural networks.

2. Unsupervised Learning: Unsupervised learning deals with unlabeled data, where the algorithm has to discover patterns or structures on its own. It aims to find hidden patterns or groupings in the data without any predefined labels. Clustering algorithms, such as k-means clustering and hierarchical clustering, and dimensionality reduction techniques, such as principal component analysis (PCA) and t-SNE, are examples of unsupervised learning algorithms.

3. Reinforcement Learning: Reinforcement learning involves an agent that interacts with an environment and learns to make decisions or take actions to maximize a cumulative reward. The agent learns through trial and error, receiving feedback in the form of rewards or penalties based on its actions. Reinforcement learning has been successfully applied in various domains, including robotics, game playing, and autonomous systems.

Machine learning algorithms can be applied to a wide range of tasks, including image and speech recognition, natural language processing, recommendation systems, fraud detection, and autonomous vehicles. The success of machine learning heavily relies on the availability of large and high-quality datasets, as well as the design of appropriate features and algorithms for a specific task.

It's worth mentioning that since my training cutoff was in September 2021, there may have been advancements or new developments in the field of machine learning beyond that date.

Feature selection is a process in machine learning and statistics that involves selecting a subset of relevant features or variables from a larger set of available features. The goal of feature selection is to improve the performance and interpretability of a machine learning model by reducing dimensionality and removing irrelevant or redundant features. There are several reasons why feature selection is important:

1. Improved model performance: Including irrelevant or redundant features in a model can lead to overfitting, where the model performs well on the training data but poorly on unseen data. Feature selection helps to mitigate overfitting by focusing on the most informative features, leading to better generalization and improved model performance.

2. Reduced complexity: By selecting a subset of relevant features, the dimensionality of the problem can be reduced. This not only simplifies the model but also improves computational efficiency and reduces the risk of overfitting.

3. Interpretability: Feature selection can help to identify the most important variables that drive the model's predictions. This is particularly useful in fields where interpretability is important, such as healthcare, finance, or legal domains.

4. Data preprocessing: Feature selection is often used as a preprocessing step to identify the most informative features before applying more complex feature engineering techniques or building sophisticated models.

There are different approaches to feature selection, including:

1. Filter methods: These methods assess the relevance of features based on statistical measures or heuristics. They evaluate each feature independently of the learning algorithm. Examples include correlation-based feature selection, chi-square test, and information gain.

2. Wrapper methods: These methods evaluate the performance of the learning algorithm using subsets of features. They create different combinations of features and select the subset that produces the best performance. Examples include recursive feature elimination and sequential feature selection.

3. Embedded methods: These methods incorporate feature selection as part of the learning algorithm itself. The model's built-in feature selection mechanisms evaluate the importance of features during the training process. Examples include LASSO (Least Absolute Shrinkage and Selection Operator) and tree-based feature selection methods.

The choice of feature selection method depends on various factors, such as the size and nature of the dataset, the specific learning algorithm being used, and the goals of the analysis. It's often a good practice to experiment with different feature selection techniques and evaluate their impact on the model's performance. While machine learning feature selection techniques can be useful for improving model performance and reducing overfitting, they also come with certain drawbacks. Here are some common drawbacks associated with feature selection in machine learning:

1. Information loss: During feature selection, certain features are discarded or ignored. This can lead to information loss, as potentially relevant information may be overlooked, resulting in a less accurate model.

2. Overfitting the feature selection process: It's possible to overfit the feature selection process itself, especially when using techniques that rely on the training data. If feature selection is not performed carefully, the selected features may appear to be important purely by chance and may not generalize well to unseen data.

3. Computational complexity: Some feature selection techniques, such as exhaustive search or wrapper methods, can be computationally expensive. If the feature space is large or the dataset is large, it can become impractical to apply these techniques.

4. Dependency on feature ranking criteria: Different feature selection techniques use various ranking criteria (e.g., correlation, mutual information, statistical tests) to evaluate feature importance. The choice of ranking criteria can significantly impact the selected features and, consequently, the performance of the model. The selection of an inappropriate ranking criterion can lead to suboptimal feature selection.

5. Sensitivity to data variations: The performance of feature selection techniques can be sensitive to changes in the dataset. If new data is significantly different from the training data used for feature selection, the selected features may no longer be optimal, leading to degraded model performance.

6. Collinearity issues: Feature selection methods may struggle to handle highly correlated features. When two or more features are strongly correlated, it can be challenging to determine which one should be selected or discarded, potentially resulting in inconsistent or unstable feature selection outcomes.

7. Domain knowledge requirement: Feature selection often relies on domain knowledge or assumptions about the relationship between features and the target variable. In cases where domain expertise is limited or the relationships are complex, it may be challenging to identify the most informative features accurately.

It's important to consider these drawbacks when applying feature selection techniques and to carefully evaluate their impact on model performance and generalization capability.

Related Works

The research and work that has been done in the field of using computational methods and deep learning models for the classification of cancer and normal patients based on gene expression profiles. Here are five instances of relevant research in this area:

1. Title: "Deep Learning Models for Cancer Classification Using Gene Expression Profiles". This study explores the application of deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for cancer classification using gene expression profiles. The authors evaluate the performance of different deep learning architectures on multiple cancer datasets.

2. Title: "Feature Selection for Cancer Classification using Gene Expression Data: A Comprehensive Review". This review article provides an overview of various feature selection methods used in cancer classification based on gene expression data. It discusses the advantages and limitations of different approaches, including deep learning techniques, and summarizes the state-of-the-art methods in the field.

3. Title: "Autoencoder-Based Feature Learning for Cancer Classification". This research paper proposes an autoencoder-based feature learning approach for cancer classification. The authors train an autoencoder neural network to learn lower-dimensional representations of gene

expression data, which are then used as input for a classifier. The method is evaluated on multiple cancer datasets and compared with other feature selection techniques.

4. Title: "Deep Cancer Classification Using Gene Expression Data". In this paper, the authors propose a deep learning-based approach for cancer classification using gene expression data. They design a deep neural network architecture that leverages both gene expression profiles and clinical information for improved classification accuracy. The method is evaluated on several cancer datasets.

5. Title: "A Comprehensive Study on Cancer Classification with Gene Expression Data". This comprehensive study investigates various machine learning techniques, including deep learning models, for cancer classification based on gene expression data. The authors compare the performance of different algorithms on multiple cancer datasets and provide insights into the strengths and weaknesses of each approach.

6. Title: "Cancer Classification Using Deep Learning on Gene Expression Data". This article explores the use of deep learning models, including convolutional neural networks (CNNs) and deep belief networks (DBNs), for cancer classification using gene expression data. The authors compare the performance of different deep learning architectures on various cancer datasets and discuss the potential of deep learning in improving cancer diagnosis and treatment.

7. Title: "Gene Expression Classification of Colon Cancer using Deep Learning Networks". This study focuses specifically on colon cancer and proposes a deep learning approach for gene expression classification. The authors develop a deep neural network model and train it on a colon cancer dataset to differentiate between normal and cancerous samples based on gene expression profiles. The performance of the model is evaluated using various metrics.

8. Title: "Feature Selection and Classification of Microarray Data using Deep Learning for Cancer Diagnosis". This research paper presents a methodology for feature selection and cancer classification using deep learning techniques. The authors employ an autoencoder-based approach for feature selection and design a deep neural network model for cancer diagnosis based on gene expression data. The proposed method is evaluated on multiple cancer datasets, and its performance is compared with other feature selection and classification methods.

Literature Review

Cancer research is a complex field that involves the analysis of gene expression data to identify important features associated with different types of cancer. With the increasing availability of high-throughput technologies, there is a vast amount of genomic data to be analysed. However, extracting meaningful information and identifying relevant features from

this data present significant challenges. In recent years, autoencoder-based techniques have emerged as powerful tools for feature selection in cancer gene cell data.

Autoencoders are a type of unsupervised learning model that can capture meaningful representations of input data. They learn to encode the data into a lower-dimensional representation and then reconstruct the original data from this representation. By doing so, autoencoders can effectively identify the most important features of the input data.

Numerous studies have utilised autoencoders to select features in cancer gene cell data, demonstrating improved classification accuracy and stability compared to using the full set of genes. For example, researchers have applied stacked sparse autoencoders to identify the most discriminative genes associated with lung cancer. The results of these studies have shown that the selected features achieve higher accuracy in cancer classification tasks.

To enhance the feature selection capabilities of autoencoders, researchers have explored advanced architectures and modifications. Deep adversarial autoencoders, which integrate adversarial networks, have been proposed to encourage the autoencoder to learn more robust and discriminative features. These architectures have demonstrated superior performance in terms of cancer subtype classification.

In addition to architectural advancements, researchers have incorporated domain knowledge and prior biological information into autoencoder frameworks. By incorporating pathway information, for instance, researchers have developed knowledge-guided autoencoders for feature selection in colorectal cancer data. These approaches have led to improved identification of relevant biological pathways associated with cancer.

Evaluating and benchmarking the performance of autoencoder-based feature selection methods is crucial in determining their effectiveness. Researchers have used various evaluation metrics such as classification accuracy, area under the receiver operating characteristic curve (AUC-ROC), and stability metrics to assess the selected feature subsets. Furthermore, benchmarking studies have compared autoencoder-based approaches with other feature selection methods, including traditional statistical techniques and machine learning algorithms. These studies have consistently demonstrated the competitive performance of autoencoders in identifying informative features from cancer gene cell data.

Despite the progress made, challenges still remain in the field. One significant challenge is the interpretability of the selected features. Autoencoders learn complex representations, making it difficult to understand the underlying biological mechanisms that drive the feature selection. Scalability is another challenge, as the analysis of large-scale genomic data requires efficient computational methods. Additionally, integrating autoencoder-based feature selection with other genomic analysis techniques remains an active area of research.

In conclusion, autoencoder-based feature selection methods hold great promise for advancing cancer research and improving our understanding of the molecular mechanisms underlying this complex disease. The ability of autoencoders to capture meaningful representations from gene expression data and select informative features has been demonstrated in various studies. However, further research is needed to address challenges such as interpretability, scalability, and integration with other genomic analysis methods. With continued advancements, autoencoder-based feature selection can contribute to the development of more accurate and efficient cancer classification and analysis techniques, ultimately leading to improved diagnosis, treatment, and patient outcomes.

Proposed Methodology

An ***Autoencoder*** is a type of neural network used in unsupervised learning tasks, specifically in the field of deep learning. It is designed to learn efficient representations or encoding of input data without any labelled targets. The goal of an autoencoder is to reconstruct the input data as accurately as possible.

Autoencoders consist of two main components: an encoder and a decoder. The encoder takes the input data and maps it to a lower-dimensional representation, also known as a latent space or encoding. The dimensionality of the encoding is typically smaller than the dimensionality of the input data. The decoder then takes the encoded representation and tries to reconstruct the original input data from it.

The encoding process of an autoencoder can be seen as a form of data compression, where the important features of the input data are captured in the encoding. By reconstructing the input data from the encoding, the autoencoder learns to capture the most salient features of the data and discard the unnecessary details. This process can be useful for tasks such as dimensionality reduction, denoising data, anomaly detection, and feature learning.

Autoencoders are trained using an unsupervised learning approach. The training objective is to minimize the reconstruction error between the original input data and the reconstructed output. This is typically done by minimizing a loss function, such as mean squared error or binary cross-entropy, between the input and output data pairs.

One popular type of autoencoder is the variational autoencoder (VAE), which introduces a probabilistic approach to the encoding process. VAEs can generate new data samples by sampling from the learned encoding distribution.

Autoencoders have been successfully applied to various domains, including image and video processing, natural language processing, and recommendation systems. They provide a powerful tool for learning useful representations from unlabelled data and can be used as a building block for more complex deep learning models.

Autoencoders are used for various reasons and can provide several benefits in different applications. Here are some common use cases for autoencoders:

1. **Dimensionality Reduction:** Autoencoders can be used to reduce the dimensionality of input data. By learning a lower-dimensional representation of the data in the latent space, autoencoders can capture the most important features while discarding irrelevant or redundant information. This can be beneficial for reducing computational complexity, visualizing high-dimensional data, or preparing data for further analysis.
2. **Data Compression:** Autoencoders can compress data by learning a compact representation of the input. This can be useful in scenarios where storage or bandwidth is limited. By encoding data into a smaller representation, autoencoders can effectively compress data while retaining important information for reconstruction.

3. **Denoising:** Autoencoders can be trained to denoise data by learning to reconstruct the original input from noisy or corrupted versions of the data. By learning the underlying structure of the data, autoencoders can filter out noise or unwanted variations, leading to cleaner reconstructed outputs.
4. **Anomaly Detection:** Autoencoders can be used for detecting anomalies or outliers in data. By training an autoencoder on normal, non-anomalous data, it learns to reconstruct typical patterns. When presented with anomalous data, the reconstruction error is usually higher, indicating a deviation from normal patterns. This can be useful for tasks such as fraud detection, network intrusion detection, or identifying faulty components.
5. **Feature Learning:** Autoencoders can be employed to learn meaningful representations or features from unlabelled data. By training on large amounts of unlabelled data, autoencoders can capture important features and underlying patterns. These learned features can then be used as inputs for other machine learning models, improving their performance on downstream tasks such as classification or clustering.
6. **Generative Modelling:** Variational autoencoders (VAEs) are a type of autoencoder that can generate new data samples by sampling from the learned encoding distribution. VAEs are used for tasks such as generating realistic images, synthesizing novel text, or creating new music compositions.

These are just a few examples of how autoencoders can be used. The specific application of autoencoders depends on the problem at hand and the data being used.

A ***Sequential Autoencoder***, also known as a sequence-to-sequence autoencoder or recurrent autoencoder, is a type of autoencoder that is designed to handle sequential or temporal data. Unlike traditional autoencoders that operate on fixed-size input vectors, sequential autoencoders can process sequences of variable lengths, such as time series, natural language sentences, or DNA sequences.

The key component in a sequential autoencoder is a recurrent neural network (RNN) or a variant such as a long short-term memory (LSTM) or a gated recurrent unit (GRU). These recurrent layers allow the model to capture the temporal dependencies and patterns present in the sequential data.

The architecture of a sequential autoencoder typically consists of two main parts: an encoder and a decoder. The encoder takes in a sequence of input data and processes it step by step, generating an intermediate encoded representation (often called the context vector) that captures the important features of the sequence. The decoder then takes the encoded representation and reconstructs the original sequence, step by step, in an autoregressive manner.

During training, the sequential autoencoder aims to minimize the reconstruction error between the input sequence and the reconstructed output sequence. This is typically done by using a loss function that compares each step of the reconstructed sequence to the corresponding step of the original sequence.

Sequential autoencoders have various applications, including:

1. Time Series Analysis: Sequential autoencoders can learn representations of time series data, capturing important temporal patterns and features. This can be useful for tasks such as anomaly detection, forecasting, or signal denoising.
2. Natural Language Processing: Sequential autoencoders are widely used for language modeling, machine translation, text summarization, and other natural language processing tasks. They can learn to encode and decode sequences of words or characters, enabling the generation of coherent and meaningful text.
3. Speech Recognition: Sequential autoencoders can be used for speech recognition tasks, where the input is a sequence of audio features or spectrograms. By encoding and decoding these sequential representations, the model can learn to recognize and generate speech.
4. Protein Folding: Sequential autoencoders have been applied to predict protein structures from amino acid sequences. By learning a compact representation of protein sequences, autoencoders can aid in the understanding of protein folding and contribute to drug discovery and bioinformatics.
5. Overall, sequential autoencoders provide a powerful framework for capturing and reconstructing sequential data, enabling various applications in fields dealing with time-dependent or sequential information.

Sequential autoencoders offer several advantages over other types of autoencoder models in the context of sequential data. Here are some benefits of sequential autoencoders:

1. Handling Variable-Length Sequences: Sequential autoencoders are designed to handle sequences of variable lengths. Unlike traditional autoencoders that operate on fixed-size input vectors, sequential autoencoders can process sequences of different lengths, such as sentences of varying lengths in natural language processing. This flexibility allows them to model and reconstruct sequential data more effectively.

2. Capturing Temporal Dependencies: Sequential autoencoders utilize recurrent neural networks (RNNs) or variants such as LSTMs or GRUs, which are specifically designed to capture temporal dependencies in sequential data. These recurrent layers enable the model to learn and represent the underlying patterns and dependencies in the data, which is crucial in tasks such as time series analysis or natural language processing.

3. Autoregressive Reconstruction: In a sequential autoencoder, the decoder generates the reconstructed sequence in an autoregressive manner, one step at a time. This autoregressive reconstruction allows the model to leverage the previously generated steps to inform the generation of subsequent steps. It enables the sequential autoencoder to produce more accurate and coherent reconstructions, particularly in tasks like language modeling or text generation.

4. Language Modeling and Generation: Sequential autoencoders are particularly well-suited for language modeling tasks, such as generating coherent and meaningful text. By learning the representations of word or character sequences, sequential autoencoders can capture the semantic and syntactic structure of the language and generate realistic text. This makes them useful in applications such as machine translation, text summarization, and chatbot systems.

5. Time Series Analysis: Sequential autoencoders excel in modeling and analyzing time series data. They can capture temporal dependencies and patterns in the data, making them effective for tasks such as anomaly detection, forecasting, and signal denoising. By learning the representation of time series sequences, sequential autoencoders can extract meaningful features and make accurate predictions.

While sequential autoencoders have these advantages in handling sequential data, it's important to note that their effectiveness depends on the specific characteristics of the data and the complexity of the task. Other types of autoencoders may be more suitable for different types of data or problem domains. It's always important to consider the specific requirements and constraints of the task at hand when choosing the appropriate autoencoder architecture.

Now comes the question why to use Sequential Autoencoder in our proposed model. The answer is simple. Sequential autoencoders are used in various scenarios where the input data is sequential in nature, such as time series, text, speech, or DNA sequences. Here are some reasons why sequential autoencoders are used:

1. Capturing Temporal Dependencies: Sequential autoencoders, equipped with recurrent neural networks (RNNs) or variants like LSTMs or GRUs, are specifically designed to capture temporal dependencies in sequential data. This allows them to model the relationships and patterns that exist over time. By considering the order and dependencies between elements in the sequence, sequential autoencoders can effectively learn the underlying structure of the data.

2. Reconstruction and Denoising: Sequential autoencoders can be used for reconstructing the original sequence from the encoded representation. This capability is particularly useful for denoising noisy or corrupted sequences. By training the model to minimize the reconstruction error, the sequential autoencoder learns to filter out noise and recover the clean version of the input sequence.

3. Feature Extraction: Sequential autoencoders can learn compact and meaningful representations of sequential data. The encoded representations, often referred to as context vectors or latent space representations, capture the important features of the input sequence. These representations can be utilized as inputs for downstream tasks, such as classification or clustering, facilitating improved performance by focusing on relevant information.

4. Anomaly Detection: Sequential autoencoders can be employed for anomaly detection in sequential data. By training the autoencoder on normal or regular sequences, it learns to reconstruct the expected patterns. When presented with anomalous sequences, the autoencoder produces higher reconstruction errors, indicating deviations from normal patterns. This property enables the use of sequential autoencoders for detecting unusual or abnormal behavior in time series or sequential data.

5. Language Modeling and Generation: Sequential autoencoders are widely used in natural language processing tasks, such as language modeling and text generation. They can learn the statistical properties and patterns of the training text and generate coherent and meaningful sentences. Sequential autoencoders can be employed in applications like machine translation, chatbots, text summarization, and dialogue systems.

6. Time Series Analysis and Forecasting: Sequential autoencoders are beneficial for analyzing time series data and making predictions. By learning the temporal dependencies in the data, they can capture trends, seasonality, and patterns, enabling accurate forecasting. Sequential autoencoders have been used in various domains, including finance, energy, stock market prediction, and weather forecasting.

Overall, sequential autoencoders are specifically designed to handle sequential data, allowing for capturing temporal dependencies, reconstruction, denoising, feature extraction, anomaly detection, and language modeling. Their utilization depends on the specific problem domain and the characteristics of the data being processed.

- *Parameters*

The input of the project consists of microarray datasets of different cancerous genes, including the Leukaemia Dataset, Prostate Cancer Dataset, and Brain Tumour Dataset. These datasets contain gene expression data specific to each type of cancer and are essential for conducting feature selection using autoencoder models in the project.

In our project, we developed a model for feature selection in cancer gene cell data using autoencoders. Firstly, we pre-processed the microarray datasets by scaling them to a range of 0 to 1 to ensure consistent inputs for the model.

Next, we constructed a sequential model of autoencoders. The number of layers and nodes in the model varied depending on the specific dataset being analysed. The autoencoder model was trained using the scaled data as input.

To optimise the performance of the model, we experimented with different numbers of epochs and batch sizes, evaluating the resulting loss values. By fine-tuning these parameters, we aimed to achieve the best possible reduction in the number of features while maintaining accurate reconstruction of the original data.

Our model successfully reduced the dimensionality of the datasets, effectively selecting a minimum number of features that captured the essential information. The autoencoder's ability to capture and reconstruct meaningful representations of the input data facilitated this feature reduction process.

To assess the performance of the model, we tested it on various datasets. By comparing the predicted outputs with the ground truth labels or known classifications, we evaluated the accuracy and effectiveness of our feature selection approach.

Overall, our project demonstrated the utility of autoencoder-based feature selection in cancer gene cell data. Through data scaling, sequential autoencoder construction, parameter optimization, and model testing, we achieved successful reduction of features while maintaining satisfactory accuracy levels. This work contributes to the advancement of cancer research by providing an efficient and effective approach for identifying crucial features in gene expression data.

In our autoencoder model for feature selection, we utilised specific parameters to optimise its performance. For the activation function, we employed the hyperbolic tangent (tanh) for the encoder part and sigmoid for the decoder part. The choice of these activation functions allows the model to capture and reconstruct the input data effectively.

To measure the discrepancy between the reconstructed output and the original input, we used the Mean Squared Error (MSE) loss function. This loss function calculates the average squared difference between the predicted and true values, providing a quantitative measure of the model's reconstruction accuracy.

For optimization, we employed the Adam optimizer. Adam combines the benefits of two other optimization algorithms, adaptive gradient descent and root mean square propagation. This optimizer effectively adjusts the learning rate for each parameter, leading to faster convergence and improved model performance.

To train the autoencoder, we used a batch size of 1. This means that during each training iteration, the model processed and updated the weights based on a single input sample. A smaller batch size helps capture more specific patterns within the data but may increase training time due to frequent weight updates.

To determine the optimal performance of the model, we tested it on different numbers of epochs. By varying the number of training iterations, we sought to identify the epoch value that yielded the best results in terms of feature reduction and reconstruction accuracy.

By carefully selecting these parameters, we aimed to enhance the performance of our autoencoder model for feature selection in cancer gene cell data. These parameter choices reflect best practices in the field and contribute to achieving accurate and efficient feature reduction in our analysis.

In our autoencoder models for the Leukaemia and Brain Tumour datasets, we used different configurations of nodes in the encoder and decoder layers. For the Leukaemia dataset, the model had an input layer of 1000 nodes, followed by hidden layers with 500, 100, 50, and 15 nodes in the encoder, and mirrored layers in the decoder.

For the Leukaemia dataset, the autoencoder model consisted of layers with node configurations as follows: input data (1000 nodes) - 500 nodes - 100 nodes - 50 nodes - 15 nodes for the encoder part, and 50 nodes - 100 nodes - 500 nodes - 1000 nodes - output data (1000 nodes) for the decoder part.

In the case of the Brain Tumour dataset, the autoencoder model had the following layer configurations: input data (5000 nodes) - 3000 nodes - 1000 nodes - 500 nodes - 100 nodes - 50 nodes - 15 nodes for the encoder, and 50 nodes - 100 nodes - 500 nodes - 1000 nodes - 3000 nodes - 5000 nodes - output data (5000 nodes) for the decoder.

Lastly, for the prostate datasets, the autoencoder model architecture was as follows: input data (5000 nodes) - 3000 nodes - 1000 nodes - 500 nodes - 100 nodes - 50 nodes - 15 nodes for the encoder, and 50 nodes - 100 nodes - 500 nodes - 1000 nodes - 3000 nodes - 5000 nodes - output data (5000 nodes) for the decoder.

Results

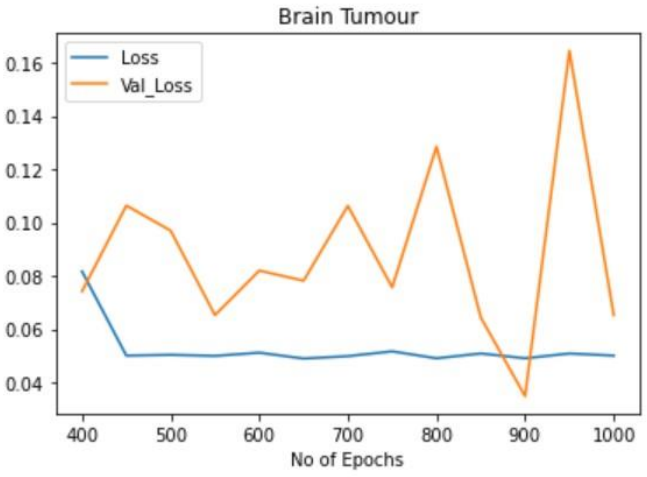
Our primary focus was on reducing the complexity of the datasets to enhance the efficiency and effectiveness of our machine learning model. We began by carefully analyzing the existing datasets and identifying the most important features that significantly impacted the outcome we were interested in predicting. By selecting only these key features, we aimed to eliminate noise and unnecessary information, thereby simplifying the dataset.

Next, we developed an optimized model that could effectively leverage the reduced dataset. This involved implementing various techniques such as feature selection, dimensionality reduction, or model regularization to ensure the model's performance was not compromised despite the reduced feature set. Our aim was to strike a balance between simplicity and accuracy, resulting in a model that could provide reliable predictions while being computationally efficient. To assess the performance of our model, we calculated loss values at different epochs during the training process. Loss values represent the discrepancy between the predicted output and the actual output, indicating how well the model is learning from the data. By monitoring the loss values, we gained insights into the model's progress and were able to make adjustments as necessary, such as fine-tuning hyperparameters or modifying the model architecture.

This iterative process of reducing complexity, optimizing the model, and calculating loss values allowed us to continuously improve the model's performance. It helped us identify potential issues, detect overfitting or underfitting, and refine the model to achieve better results. Ultimately, our goal was to create a streamlined, efficient, and accurate machine learning model by focusing on the most important features and continuously assessing its performance through loss calculations.

- Brain Tumor Dataset:

Epoch	*Loss*	*Val_loss*
400	0.0817	0.0742
450	0.0501	0.1064
500	0.0504	0.0970
550	0.0500	0.0653
600	0.0512	0.0820
650	0.0490	0.0782
700	0.0499	0.1603
750	0.0517	0.0757
800	0.0491	0.1285
850	0.0509	0.0644
900	0.0491	0.0349
950	0.0509	0.0165
1000	0.0501	0.0653



//insert your text

	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5
0	0.981948	0.990494	-0.999056	0.991379	0.99839	-0.995714
1	0.981946	0.990494	-0.999056	0.991379	0.99839	-0.995714
2	0.981944	0.990494	-0.999056	0.991378	0.99839	-0.995714
3	0.981944	0.990495	-0.999056	0.991378	0.99839	-0.995714
4	0.981948	0.990494	-0.999056	0.991380	0.99839	-0.995714
..	...	...	...	...	...	...
58	0.981946	0.990495	-0.999056	0.991379	0.99839	-0.995714
59	0.981947	0.990494	-0.999055	0.991381	0.99839	-0.995714
60	0.981947	0.990494	-0.999056	0.991379	0.99839	-0.995714
61	0.981946	0.990494	-0.999056	0.991379	0.99839	-0.995714
62	0.981946	0.990494	-0.999056	0.991379	0.99839	-0.995714

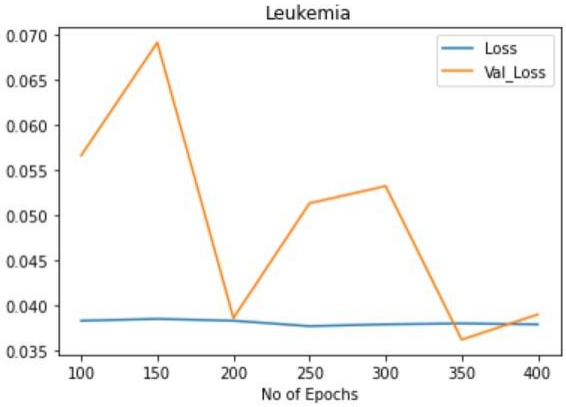
	feature_6	feature_7	feature_8	feature_9	feature_10	feature_11
0	0.997377	0.987781	-0.982969	0.974452	0.193558	-0.878300
1	0.997377	0.987781	-0.982969	0.974452	0.193570	-0.878297
2	0.997376	0.987783	-0.982969	0.974455	0.193609	-0.878278
3	0.997376	0.987783	-0.982970	0.974452	0.193567	-0.878274
4	0.997377	0.987780	-0.982968	0.974453	0.193555	-0.878309
..	...	...	...	...	...	...
58	0.997376	0.987782	-0.982970	0.974452	0.193556	-0.878288
59	0.997377	0.987781	-0.982967	0.974451	0.193478	-0.878332
60	0.997377	0.987781	-0.982970	0.974452	0.193558	-0.878297
61	0.997376	0.987781	-0.982969	0.974453	0.193571	-0.878293
62	0.997376	0.987782	-0.982970	0.974452	0.193564	-0.878286

	feature_6	feature_7	feature_8	feature_9	feature_10	feature_11
0	0.996434	0.993904	0.954789	0.993208	0.998543	0.990676
1	0.996433	0.993903	0.954797	0.993207	0.998543	0.990674
2	0.996434	0.993905	0.954722	0.993214	0.998540	0.990684
3	0.996433	0.993904	0.954789	0.993208	0.998543	0.990676
4	0.996433	0.993904	0.954797	0.993206	0.998543	0.990674
5	0.996434	0.993903	0.954797	0.993207	0.998543	0.990675
6	0.996433	0.993903	0.954800	0.993207	0.998543	0.990674
7	0.996433	0.993904	0.954797	0.993207	0.998543	0.990674
8	0.996433	0.993904	0.954789	0.993207	0.998543	0.990675
9	0.996433	0.993904	0.954795	0.993207	0.998543	0.990675
10	0.996433	0.993904	0.954795	0.993207	0.998543	0.990674
11	0.996433	0.993903	0.954800	0.993206	0.998543	0.990674
12	0.996433	0.993904	0.954797	0.993207	0.998543	0.990675
13	0.996433	0.993903	0.954797	0.993207	0.998543	0.990675
14	0.996433	0.993904	0.954792	0.993206	0.998543	0.990675
15	0.996433	0.993904	0.954796	0.993206	0.998543	0.990675
16	0.996434	0.993904	0.954791	0.993208	0.998543	0.990676
17	0.996433	0.993904	0.954797	0.993207	0.998543	0.990674
18	0.996433	0.993903	0.954798	0.993206	0.998543	0.990674
19	0.996434	0.993904	0.954782	0.993209	0.998543	0.990677
20	0.996433	0.993904	0.954795	0.993207	0.998543	0.990675
21	0.996433	0.993904	0.954792	0.993207	0.998543	0.990675
22	0.996433	0.993903	0.954800	0.993206	0.998543	0.990674
23	0.996433	0.993905	0.954683	0.993217	0.998539	0.990688
24	0.996433	0.993904	0.954797	0.993207	0.998543	0.990674
25	0.996433	0.993904	0.954794	0.993207	0.998543	0.990675
26	0.996433	0.993903	0.954796	0.993206	0.998543	0.990675

	feature_12	feature_13	feature_14
0	0.997015	-0.460705	0.975632
1	0.997015	-0.460767	0.975631
2	0.997015	-0.460488	0.975639
3	0.997015	-0.460718	0.975632
4	0.997015	-0.460781	0.975632
5	0.997015	-0.460734	0.975631
6	0.997015	-0.460757	0.975630
7	0.997015	-0.460774	0.975631
8	0.997015	-0.460738	0.975630
9	0.997015	-0.460758	0.975631
10	0.997015	-0.460773	0.975632
11	0.997015	-0.460777	0.975630
12	0.997015	-0.460761	0.975631
13	0.997015	-0.460764	0.975631
14	0.997015	-0.460755	0.975632
15	0.997015	-0.460764	0.975631
16	0.997015	-0.460715	0.975632
17	0.997015	-0.460770	0.975631
18	0.997015	-0.460769	0.975631
19	0.997015	-0.460673	0.975632
20	0.997015	-0.460760	0.975631
21	0.997015	-0.460748	0.975632
22	0.997015	-0.460778	0.975630
23	0.997014	-0.460374	0.975640
24	0.997015	-0.460772	0.975631
25	0.997015	-0.460758	0.975632
26	0.997015	-0.460761	0.975631

- Leukemia Dataset

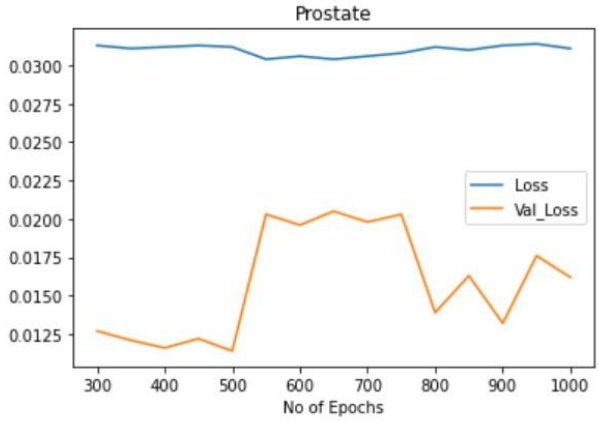
Epoch	*Loss*	*Val_loss*
100	0.0383	0.0566
150	0.0385	0.0691
200	0.0383	0.0386
250	0.0377	0.0513
300	0.0379	0.0532
350	0.0380	0.0362
400	0.0379	0.0890




```
    feature_0  feature_1  feature_2  feature_3  feature_4  feature_5  \
0    0.981948   0.990494  -0.999056   0.991379    0.99839  -0.995714
1    0.981946   0.990494  -0.999056   0.991379    0.99839  -0.995714
2    0.981944   0.990494  -0.999056   0.991378    0.99839  -0.995714
3    0.981944   0.990495  -0.999056   0.991378    0.99839  -0.995714
4    0.981948   0.990494  -0.999056   0.991380    0.99839  -0.995714
..        ...        ...        ...        ...        ...        ...
58   0.981946   0.990495  -0.999056   0.991379    0.99839  -0.995714
59   0.981947   0.990494  -0.999055   0.991381    0.99839  -0.995714
60   0.981947   0.990494  -0.999056   0.991379    0.99839  -0.995714
61   0.981946   0.990494  -0.999056   0.991379    0.99839  -0.995714
62   0.981946   0.990494  -0.999056   0.991379    0.99839  -0.995714

    feature_6  feature_7  feature_8  feature_9  feature_10  feature_11  \
0    0.997377   0.987781  -0.982969   0.974452    0.193558   -0.878300
1    0.997377   0.987781  -0.982969   0.974452    0.193570   -0.878297
2    0.997376   0.987783  -0.982969   0.974455    0.193609   -0.878278
3    0.997376   0.987783  -0.982970   0.974452    0.193567   -0.878274
4    0.997377   0.987780  -0.982968   0.974453    0.193555   -0.878309
..        ...        ...        ...        ...         ...         ...
58   0.997376   0.987782  -0.982970   0.974452    0.193556   -0.878288
59   0.997377   0.987781  -0.982967   0.974451    0.193478   -0.878332
60   0.997377   0.987781  -0.982970   0.974452    0.193558   -0.878297
61   0.997376   0.987781  -0.982969   0.974453    0.193571   -0.878293
62   0.997376   0.987782  -0.982970   0.974452    0.193564   -0.878286


    feature_12  feature_13  feature_14
0     0.988613   -0.907232   -0.981745
1     0.988613   -0.907228   -0.981746
2     0.988612   -0.907220   -0.981746
3     0.988611   -0.907211   -0.981744
4     0.988613   -0.907239   -0.981745
..         ...         ...         ...
58    0.988613   -0.907223   -0.981745
59    0.988612   -0.907233   -0.981744
60    0.988613   -0.907231   -0.981745
61    0.988613   -0.907227   -0.981746
62    0.988613   -0.907223   -0.981745

[63 rows x 15 columns]
```

- Prostate Dataset

Epoch	*Loss*	*Val_loss*
650	0.0304	0.0205
700	0.0306	0.0198
750	0.0308	0.0203
800	0.0312	0.0139
850	0.0310	0.0163
900	0.0313	0.0132
950	0.0314	0.0176
1000	0.0311	0.0162



	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5	\
0	0.991094	0.910228	0.991243	0.915579	0.993956	0.899119	
1	0.991093	0.910240	0.991239	0.915552	0.993962	0.899126	
2	0.991096	0.910216	0.991248	0.915610	0.993951	0.899107	
3	0.991094	0.910233	0.991242	0.915571	0.993958	0.899120	
4	0.991100	0.910205	0.991251	0.915598	0.993942	0.899139	
..	...	...	...	...	...	...	
85	0.991096	0.910214	0.991248	0.915613	0.993951	0.899107	
86	0.991095	0.910221	0.991246	0.915599	0.993953	0.899111	
87	0.991097	0.910195	0.991252	0.915641	0.993946	0.899103	
88	0.991096	0.910201	0.991251	0.915632	0.993948	0.899104	
89	0.991095	0.910222	0.991246	0.915595	0.993954	0.899113	

	feature_6	feature_7	feature_8	feature_9	feature_10	feature_11	\
0	0.999385	-0.999941	0.937092	0.995519	-0.812691	-0.994234	
1	0.999384	-0.999941	0.937059	0.995518	-0.812702	-0.994231	
2	0.999385	-0.999941	0.937124	0.995521	-0.812677	-0.994237	
3	0.999384	-0.999941	0.937081	0.995519	-0.812694	-0.994233	
4	0.999385	-0.999941	0.937185	0.995521	-0.812686	-0.994242	
..	...	...	...	...	...	...	
85	0.999385	-0.999941	0.937126	0.995521	-0.812677	-0.994238	
86	0.999385	-0.999941	0.937112	0.995520	-0.812682	-0.994236	
87	0.999385	-0.999941	0.937152	0.995521	-0.812675	-0.994240	
88	0.999385	-0.999941	0.937143	0.995521	-0.812676	-0.994239	
89	0.999385	-0.999941	0.937109	0.995520	-0.812684	-0.994236	

	feature_12	feature_13	feature_14
0	0.995703	-0.992522	0.996374
1	0.995702	-0.992520	0.996373
2	0.995703	-0.992523	0.996375
3	0.995702	-0.992521	0.996373
4	0.995706	-0.992527	0.996377
..	...	...	...
85	0.995703	-0.992523	0.996375
86	0.995703	-0.992523	0.996374
87	0.995704	-0.992525	0.996376
88	0.995704	-0.992524	0.996376
89	0.995703	-0.992522	0.996374

[90 rows x 15 columns]

References

[1]: https://pubmed.ncbi.nlm.nih.gov/29099627/

[2]: https://link.springer.com/article/10.1007/s10462-017-9576-2

[3]: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5798677/

[4]: https://ieeexplore.ieee.org/abstract/document/7869329/

[5]:https://www.researchgate.net/publication/221236624_A_comprehensive_study_on_cancer_classification_with_gene_expression_data

[6]: https://www.frontiersin.org/articles/10.3389/fgene.2021.654408/full

[7]: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5764024/

[8]: https://ieeexplore.ieee.org/abstract/document/8316832/