CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

This project presents a comprehensive comparative study of three state-of-the-art deep learning architectures—Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformers—for the task of cancer classification using high-dimensional genomic data. Cancer classification, based on gene expression and mutation profiles, plays a critical role in enabling early diagnosis, stratified treatment planning, and personalized medicine. With the increasing availability of large-scale genomic datasets, machine learning models, particularly deep learning, have become indispensable tools for mining patterns indicative of cancer subtypes.

The study utilizes publicly accessible genomic datasets that include gene expression levels and mutation statuses of cancer patients. Each model is rigorously trained and validated using these datasets, and their performance is evaluated across multiple critical dimensions: classification accuracy, precision, recall, F1-score, computational efficiency, and interpretability of predictions.

Beyond more performance comparison, the project also explores model architecture tuning, data preprocessing strategies, and visualization techniques for model explainability, such as attention maps in Transformers and saliency maps in CNNs. The overarching aim is to identify the most suitable deep learning framework that can be integrated into clinical decision support systems for precision oncology. Ultimately, the findings are intended to contribute to the broader field of biomedical data science by highlighting how different neural architectures respond to the challenges of complex, high-dimensional biological data.

1.2 PURPOSE OF PROJECT

The primary purpose of this project is to systematically investigate and compare the effectiveness of CNNs, RNNs, and Transformers for classifying cancer types based on genomic data, with a specific focus on understanding how these models handle the unique characteristics of biological datasets. Genomic data is inherently high-dimensional, noisy, and complex, often exhibiting non-linear relationships and hidden patterns that traditional machine learning methods struggle to capture.

By analyzing key genetic markers, gene expression profiles, and mutational signatures, the project aims to uncover how each model architecture leverages these features to make informed predictions. The research places strong emphasis on identifying which model provides the best trade-off between predictive performance and real-world usability, considering metrics like:

- Accuracy: to measure overall correctness.
- Precision and Recall: to evaluate the model's robustness in identifying cancer subtypes.
- F1-score: to provide a balanced view of precision and recall.
- Computational efficiency: including training time, scalability, and hardware requirements.
- Interpretability: which is crucial in medical applications for gaining trust from healthcare professionals.

In the long term, the project seeks to build a generalizable and interpretable deep learning pipeline that can assist oncologists and researchers in making early, accurate, and personalized cancer diagnoses, ultimately accelerating the shift toward precision medicine. The goal is not only academic but also translational—transforming computational insights into actionable clinical tools.

1.3 SCOPE OF PROJECT

The scope of this project is confined to the application, evaluation, and comparison of deep learning methodologies for genomic-based cancer classification. It concentrates on leveraging three powerful neural network architectures—CNNs, RNNs, and Transformers—to address the classification task using input features derived exclusively from genomic datasets. These include gene expression matrices and mutation profiles that provide a molecular snapshot of patient samples.

The study is focused on the following core areas:

- Data Preprocessing: Normalization, encoding, and dimensionality reduction techniques suitable for genomic data.
- Model Implementation: Building and training CNN, RNN, and Transformer models using Python and deep learning frameworks such as TensorFlow or PyTorch.
- Performance Evaluation: Comparative assessment using standard classification metrics as well as hardware/resource consumption.
- Model Explainability: Integrating tools like SHAP, LIME, and attention heatmaps to assess how models arrive at their predictions.
- Dataset Utilization: Restricted to publicly available genomic datasets with potential to expand to other datasets in future work.

Importantly, the scope excludes traditional machine learning algorithms (e.g., SVM, Random Forest) and non-genomic clinical data. The rationale is to maintain a focused investigation on how modern deep neural networks alone handle raw and processed genomic inputs. This project is designed to act as a foundation for future multidisciplinary research involving bioinformatics, computational biology, and clinical genomics, aiming to bridge the gap between AI advancements and practical healthcare solutions.

CHAPTER 2

LITERATURE SURVEY

Meenu Vijarania, Neha Goel, Akshat Agarwal, and Celestine Iwendi (2024) 'Autonomous Breast Cancer Detection Using a One-Dimensional Convolution Neural Network with Long Short-Term Memory'

In recent years, the combination of one-dimensional convolutional neural networks (Conv1D) and Long Short-Term Memory (LSTM) networks has gained popularity for biomedical applications, particularly in genomic-based cancer classification. Conv1D layers are effective at extracting local patterns from sequential data such as gene expression profiles, where they act as feature extractors by capturing motifs or short subsequences. However, they are limited in modeling long-term dependencies across the sequence. To address this, LSTM networks are incorporated due to their ability to retain information over longer sequences and capture temporal or sequential relationships. By integrating Conv1D with LSTM, the hybrid model benefits from both local pattern recognition and long-range sequence dependency modeling, making it well-suited for complex biological datasets. Several studies have demonstrated the effectiveness of Conv1D-LSTM architectures in cancer prediction tasks. For instance, models using this approach have shown improved accuracy and robustness in classifying different cancer types based on gene expression, DNA methylation, or miRNA data. The synergy between convolutional and recurrent layers enables these models to better understand the underlying biological processes, ultimately supporting more accurate and interpretable cancer diagnostics.

Mirco Gallazi, Sara Bivaschi, Alexandro Bulgheroni, Silvia Corchs, and Ignazio Gallo (2024) 'A Large Dataset to Enhance Skin Cancer Classification With Transformer Based Deep Neural Network'

Skin cancer classification has been a growing focus in the field of medical image analysis due to the increasing incidence and the need for accurate and early diagnosis. Traditional convolutional neural networks (CNNs) have been widely used for this task, offering strong performance in feature extraction from dermoscopic images. However, CNNs are often limited in capturing long-range dependencies and global context, which are crucial for differentiating between visually similar skin lesion types. To address these limitations, transformer-based models, particularly Vision Transformers (ViTs), have been introduced for skin cancer image classification. These models use self-attention mechanisms to learn relationships across the entire image, providing a more comprehensive understanding of skin lesion structures. Despite their advantages, transformer models generally require large-scale annotated datasets to perform effectively. In response to this, researchers have developed a large, diverse, and high-quality skin cancer image dataset to support the training of transformer-based deep learning models. This dataset enhances the capability of ViTs and hybrid models to generalize across various skin lesion types. Several studies have reported that transformer models, especially when trained on large datasets, outperform traditional CNNs in terms of classification accuracy, precision, and robustness. The integration of transformer architectures into medical imaging workflows is proving to be a promising direction, offering improved performance and potentially aiding dermatologists in real-time, accurate skin cancer diagnosis.

Jabed Omar Bappi, Mohammed Abu Tareq Rony, Samah Alshanthri (2024) 'A Novel Learning Approach for Accurate Cancer Type and Subtype Identification'

Recent advances in cancer genomics and machine learning have catalyzed the development of sophisticated tools for cancer classification. Traditional approaches, such as gene expression profiling and histopathological analysis, though valuable, often lack the precision and scalability necessary for distinguishing between cancer subtypes. Several studies have employed machine learning algorithms like support vector machines, random forests, and k-nearest neighbors, yet their performance remains limited when faced with complex, high-dimensional data. More recently, deep learning models, especially convolutional neural networks and recurrent neural networks, have demonstrated superior accuracy in handling such complexity by automatically extracting hierarchical features. These models still face challenges in generalization and interpretability. Addressing these limitations, the current study introduces a novel deep learning framework that integrates feature selection and dimensionality reduction, resulting in enhanced classification performance and robustness across multiple cancer types and subtypes. This approach builds upon and surpasses prior methodologies by leveraging the power of deep neural networks combined with biologically relevant input features. To address these, recent studies have focused on hybrid models and feature optimization strategies. The current work builds upon these advancements by introducing a deep learning model that combines a data-driven feature extraction pipeline with an optimized classification framework, achieving improved accuracy and robustness across diverse cancer types and subtypes. This approach not only enhances the predictive capability but also contributes to more interpretable and scalable cancer classification models in the biomedical domain.

Halawani, Buchert, and Chen (2024) 'Utilizing Neurons to Interrogate Cancer: Integrative Analysis of Cancer Omics Data with Deep Learning Models'

The integration of deep learning models in cancer research has revolutionized the analysis of complex omics data, enabling more accurate predictions and insights into tumor biology. Traditional methods often struggled with the high dimensionality and heterogeneity of genomic datasets, limiting their effectiveness. Recent advancements have demonstrated that deep learning architectures, such as artificial neural networks, can effectively capture intricate patterns within multi-omics data, including genomics, transcriptomics, and proteomics. This integrative approach facilitates a comprehensive understanding of cancer mechanisms, leading to improved diagnostic and prognostic models. Moreover, the application of these models has shown promise in identifying potential therapeutic targets by uncovering previously unrecognized associations within the data. However, challenges remain in ensuring the interpretability of deep learning models and in managing the computational demands associated with processing large-scale omics datasets. Ongoing research is focused on developing more efficient algorithms and enhancing model transparency to translate these computational advancements into clinical practice effectively.For instance, the development of models like DeepMoIC, which utilizes deep graph convolutional networks, has demonstrated superior performance in integrating multi-omics data for cancer subtype classification. Additionally, frameworks like DeepKEGG incorporate biological pathway information into the integration process, enhancing the interpretability and accuracy of predictions. Despite these advancements, challenges remain in terms of data heterogeneity, model interpretability, and computational demands. Addressing these issues is crucial for the clinical translation of deep learning-based multi-omics integration methods, ultimately aiming to facilitate personalized cancer treatment strategies

Malliga Subramaniyan, Jaehyuk Cho and Veerappampalayam Sathishkumar Eswaramoorthi ,(2023) 'Multiple Types of Cancer Classification Using CT/MRI Images Based on Learning Without Forgetting Powered Deep Learning Models'

Cancer classification using medical imaging has become a vital area of study due to the growing need for accurate and automated diagnostic tools. Numerous works have investigated the application of deep learning models, particularly convolutional neural networks, to classify tumors from CT and MRI images. These models are capable of learning discriminative features directly from raw image data, eliminating the need for manual feature extraction. A notable challenge in such systems is the issue of catastrophic forgetting, where models lose previously acquired knowledge when trained incrementally on new tasks. To mitigate this, the Learning Without Forgetting framework has been introduced, allowing models to adapt to new data while preserving prior learning. This approach has shown effectiveness in handling multiple cancer types without retraining from scratch. Studies incorporating LwF alongside transfer learning and pre-trained networks have reported improvements in classification accuracy, particularly in complex datasets involving brain, lung, and breast cancer scans. The combination of deep learning techniques with strategies for continual learning presents a promising direction for building robust, scalable, and memory-efficient diagnostic models, although challenges such as cross-domain variability and model interpretability still persist.

Xinfeng Zhang, Dianning He, Yue Zheng, Huaibi Huo (2023) 'Deep Learning-Based Analysis of Breast Cancer Using Advanced Ensemble Classifier and Linear Discriminant Analysis'.

Numerous studies have explored the application of machine learning and deep learning techniques in breast cancer detection and diagnosis. Traditional classifiers such as Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM) have been widely used for their interpretability and relatively good accuracy. However, they often face limitations in handling complex, high-dimensional data. To overcome these challenges, recent research has focused on ensemble methods and deep learning frameworks that can learn intricate patterns from large datasets. Convolutional Neural Networks (CNNs) have demonstrated remarkable performance in image-based diagnostics by effectively extracting spatial hierarchies of features. Moreover, the integration of feature reduction techniques like Linear Discriminant Analysis (LDA) has proven to enhance model performance by minimizing dimensionality while preserving class-discriminatory information. Several hybrid and ensemble models have been proposed, combining the strengths of different algorithms to improve classification accuracy and robustness. These advancements underscore a growing trend towards leveraging hybrid deep learning architectures for more reliable breast cancer prediction.These combined approaches illustrate a shift toward more intelligent and robust diagnostic systems that can support clinicians in making data-driven decisions for early breast cancer diagnosis.The integration of LDA with ensemble deep learning models has been shown to improve classification performance in breast cancer detection, offering better generalization and interpretability.

Chengxuan Lin, Jing Quin, and Rongshan (2023) 'Hybrid Graph Convolutional Network With Online Masked Autoencoder for Robust Multimodal Cancer Survival Prediction'

The paper "Hybrid Graph Convolutional Network with Online Masked Autoencoder for Robust Multimodal Cancer Survival Prediction" presents a novel approach to enhancing cancer survival prediction by integrating multimodal data through advanced machine learning techniques. The authors propose a hybrid model that combines Graph Convolutional Networks (GCNs) with an online masked autoencoder, aiming to effectively capture complex relationships within multimodal datasets. This integration seeks to improve the robustness and accuracy of survival predictions, addressing challenges associated with heterogeneous data sources in oncology research.In their study, the authors address the challenges of cancer survival prediction by proposing a Hybrid Graph Convolutional Network (HGCN) integrated with an online masked autoencoder. This approach effectively models multimodal patient data—including pathological, clinical, and genomic features—into interpretable graphs. HGCN combines the strengths of Graph Convolutional Networks (GCNs) and Hypergraph Convolutional Networks (HCNs) to enhance intra- and inter-modal interactions through node message passing and hyperedge mixing mechanisms. To handle missing modalities in clinical scenarios, the model incorporates an online masked autoencoder paradigm that captures intrinsic dependencies between modalities and generates missing hyperedges for inference. Extensive experiments on six cancer cohorts from The Cancer Genome Atlas (TCGA) demonstrate that HGCN significantly outperforms state-of-the-art methods in survival prediction.

Zhilong Lv, Ying Wang, and Fa Zhang (2022) 'Transformer-Based Survival Analysis Model Integrating Histopathological Images and Genomic Data for Colorectal'

Integrating histopathological images with genomic data has become a pivotal approach in colorectal cancer prognosis. Traditional survival analysis models often analyze these data types separately, which may overlook the intricate relationships between genetic alterations and tissue morphology. Recent advancements have focused on developing models that can simultaneously process both modalities to enhance predictive accuracy. For instance, transformer-based architectures have been employed to capture complex dependencies within and between histopathological and genomic features, leading to more robust survival predictions. These integrative models leverage the strengths of deep learning to automatically extract meaningful patterns from high-dimensional data, facilitating a more comprehensive understanding of tumor behavior. Despite these advancements, challenges remain in effectively harmonizing heterogeneous data sources and ensuring the interpretability of the resulting models. Ongoing research aims to address these issues by developing more sophisticated integration techniques and validation strategies to translate these models into clinical practice.

Tehnan I. A. Mohammad, and Absalom El-Shamir Ezugwu (2022) 'Enhancing Lung Cancer Classification and Prediction with Deep Learning and Multi-Omics Data'

Lung cancer classification and prediction have long posed significant challenges due to the complex and heterogeneous nature of the disease. Conventional diagnostic methods, while widely adopted, often fail to capture subtle molecular differences essential for early detection and subtype identification. To overcome these limitations, researchers have increasingly turned to the use of high-dimensional biological data derived from genomics, transcriptomics, proteomics, and epigenomics—collectively referred to as multi-omics data. Deep learning models, including convolutional neural networks (CNNs), deep neural networks (DNNs), and autoencoders, have proven to be highly effective in learning intricate patterns from such datasets without the need for manual feature engineering. Studies have also introduced hybrid and ensemble frameworks that integrate multiple learning architectures, further improving prediction accuracy and model robustness. The integration of multi-omics data, especially through late-stage fusion techniques, has been particularly effective in capturing the complementary nature of various biological layers. In addition, strategies such as dimensionality reduction and feature selection have enhanced model interpretability and reduced computational complexity. These developments have laid the foundation for more accurate, efficient, and biologically informed approaches to lung cancer classification, ultimately contributing to precision oncology effort

Mansoor Hayat, Nouman Ahamad, and Anam Nasir (2020) 'Hybrid Deep Learning EfficiencyNetV2 and Vision Transformer Model for Breast Cancer Histopathological Image Classification'

Deep learning has become an essential tool in medical imaging, particularly in the detection and classification of cancer. Traditional convolutional neural networks such as VGGNet, ResNet, and DenseNet have been widely used for analyzing histopathological images due to their strong ability to extract local features. While these models have achieved high accuracy in various classification tasks, they often fall short in capturing long-range dependencies and global contextual information. To address this, Vision Transformers have been introduced, which utilize self-attention mechanisms to model global relationships within images more effectively. Despite their promising results, ViTs are computationally intensive and typically require large amounts of data to perform optimally. To overcome the limitations of individual models, recent research has focused on hybrid deep learning approaches that combine CNNs and transformer architectures. EfficientNetV2, known for its high efficiency and performance, has been successfully integrated with ViTs in several studies, resulting in improved classification outcomes. These hybrid models harness CNNs' ability to extract detailed local features and ViTs' strength in capturing broader image context. Studies have shown that such combinations yield better performance in cancer classification tasks, including higher accuracy, precision, and F1-scores.

Salwani Daud, Hafiza Abas, Noor Azurati Ahmad,(2020) 'Breast Cancer Classification Using Deep Learning Approaches and Histopathology Image: Comparison Study'

Recent advancements in deep learning have significantly impacted the field of medical image analysis, particularly in the classification of breast cancer using histopathological images. Various studies have demonstrated the effectiveness of convolutional neural networks (CNNs) and transfer learning models in identifying malignant and benign tissues. Traditional machine learning techniques required manual feature extraction, but deep learning approaches have automated this process, resulting in higher accuracy and efficiency. Researchers have explored numerous architectures such as VGG16, ResNet, and Inception, often leveraging pre-trained models to compensate for limited medical datasets. Studies also highlight the importance of data preprocessing, augmentation, and patch-wise analysis for improving classification performance. The integration of these methods has shown promising results in early diagnosis and decision support systems for pathologists, positioning deep learning as a critical tool in the fight against breast cancer.Some researchers have also investigated hybrid frameworks that combine CNNs with attention mechanisms or recurrent layers to better capture spatial and contextual features. Overall, the literature suggests that deep learning, particularly CNN-based and hybrid models, holds great promise in the automated classification of breast cancer, offering valuable support tools for pathologists in clinical setting.

Wang Shao, ZhiHan and JunCheng (2020) 'Integrative Analysis of Pathological Images and Multi-Dimensional Genomic Data for Cancer Prognosis'

The integration of pathological imaging and genomic data has become a pivotal approach in early-stage cancer prognosis. Traditional prognostic methods often rely solely on either histopathological examination or genomic analysis, which may not capture the comprehensive landscape of tumor biology. Recent advancements emphasize the synergistic potential of combining these modalities to enhance predictive accuracy. Studies have demonstrated that integrating histopathological images with multi-dimensional genomic data can uncover intricate patterns and biomarkers that are otherwise undetectable when each data type is analyzed in isolation. This integrative strategy facilitates a more nuanced understanding of tumor heterogeneity, leading to improved prognostic models and personalized treatment plans. Moreover, the advent of machine learning algorithms has further propelled this field, enabling the efficient processing and analysis of complex datasets to identify novel prognostic indicators. Such interdisciplinary approaches underscore a paradigm shift towards more precise and individualized cancer care.

Bo Yang, Shanmin Pang, Xqequn Shang, and Minghun Han, (2019) 'Integrating Multi-Omic Data With Deep Subspace Fusion Clustering for Cancer Subtype Prediction'

The paper "Prediction and Interpretation of Cancer Survival Using Graph Convolution Neural Networks" introduces Surv_GCNN, a novel approach employing Graph Convolutional Neural Networks (GCNNs) for cancer survival prediction across 13 cancer types using The Cancer Genome Atlas (TCGA) dataset. Surv_GCNN constructs graphs based on gene expression data, utilizing correlation analysis and the GeneMania database to model relationships between genes. The model integrates clinical data to enhance prediction accuracy and interprets the learned features to identify potential gene markers associated with cancer survival. Comparative analyses demonstrate that Surv_GCNN outperforms traditional methods like Cox-PH and Cox-nnet in several cancer types.Several recent studies have explored deep learning approaches for survival analysis in oncology, particularly focusing on integrating high-dimensional omics data. However, traditional models often struggle to capture the complex interdependencies among genes and between clinical features. The Surv_GCNN framework addresses this limitation by representing gene expression profiles as structured graphs, allowing the model to learn both local and global interactions. This graph-based representation not only improves predictive performance but also enhances biological interpretability, as it enables the identification of key gene interactions that contribute to patient survival outcomes.

Bo Yang, Yupei Zhang, Shanmin Pang, Xuequn Shang, Xueqing Zhao, Minghui Han (2019) 'Integrating Multi-Omic Data With Deep Subspace Fusion Clustering for Cancer Subtype Prediction'

Integrating multimodal genomic data has become a pivotal approach in breast cancer research, aiming to enhance the accuracy of survival predictions and uncover prognostically relevant subtypes. Building upon this concept, Subramanian et al. explored the use of canonical correlation analysis (CCA) methods for fusing histology and genomic data to predict breast cancer survival outcomes. Their study demonstrated that such multimodal fusion could effectively capture the complex interplay between different data types, leading to improved prognostic assessments. Similarly, Mondol et al. introduced MM-SurvNet, a deep learning-based framework that integrates histopathological imaging, genetic, and clinical data for survival risk stratification in breast cancer. By employing vision transformers and cross-attention mechanisms, their model achieved superior performance, underscoring the potential of advanced computational techniques in multimodal data fusion. These studies collectively highlight the growing emphasis on leveraging diverse data sources and sophisticated analytical methods to refine breast cancer prognosis and inform personalized treatment strategies.

Tanima Takur , Isha Batra , Arun Malik, and Deepak Ghimmire (2019) 'RNN-CNN Based Cancer Prediction Model for Gene Expression'

The classification of cancer using gene expression data has been a key focus in biomedical research due to the complexity and variability of genomic patterns across different cancer types. Traditional statistical methods and early machine learning approaches, such as support vector machines and decision trees, often required extensive manual feature engineering and struggled to generalize across large datasets. To overcome these limitations, deep learning models have gained popularity for their ability to automatically learn intricate patterns in high-dimensional biological data. Among these, Convolutional Neural Networks (CNNs) have shown strong performance in extracting spatial features, while Recurrent Neural Networks (RNNs) are effective in modeling sequential dependencies inherent in gene expression profiles. Studies combining CNN and RNN architectures have demonstrated improved accuracy and robustness in various bioinformatics applications. The hybrid approach leverages the spatial encoding of CNNs alongside the temporal modeling of RNNs, enabling better representation of genomic structures. The RNN-CNN-based model presented in this work extends upon these findings, offering a more efficient and accurate framework for predicting cancer types by capturing both local and sequential gene expression patterns.CNNs are effective at extracting spatial features, while RNNs capture temporal and sequential relationships, making their combination particularly powerful. Prior studies demonstrated the potential of hybrid models for improving classification accuracy in bioinformatics tasks.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Cancer classification using genomic data has traditionally relied on statistical methods and classical machine learning algorithms such as decision trees, support vector machines, and logistic regression. These models often require extensive feature engineering and are limited in their ability to capture the complex, high-dimensional patterns present in genomic datasets. In recent years, deep learning approaches have started to outperform traditional methods due to their capacity to automatically extract meaningful features and detect non-linear relationships in data. However, many existing systems are still constrained by limited datasets, lack of model interpretability, and insufficient computational scalability.

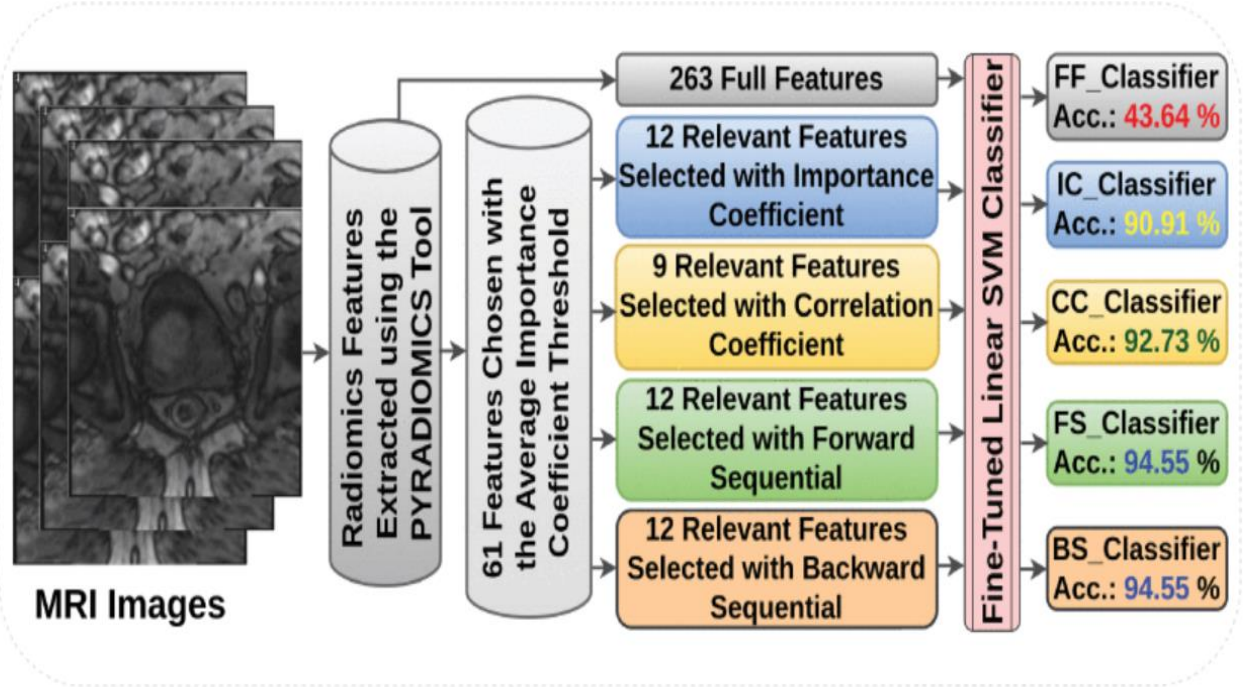


Fig. 3.1 Block Diagram of Existing Model

3.2 PROPOSED SYSTEM

The proposed system utilizes advanced deep learning models to enhance the accuracy and reliability of cancer classification based on genomic data. Specifically, the study focuses on a comparative analysis of three key architectures:

- Convolutional Neural Networks for spatial feature extraction,
- Recurrent Neural Networks for sequential gene expression modeling, and
- Transformer models for attention-based learning and long-range dependency capture.

Publicly available genomic datasets containing gene expression levels and genetic markers are used for model training and evaluation. The system includes modules for data preprocessing, model training, hyperparameter tuning, and performance assessment. The trained models are evaluated based on metrics such as accuracy, precision, recall, and F1-score. The final system aims to support medical professionals with a data-driven tool for early diagnosis and personalized treatment planning.

3.3 ADVANTAGES

- Improved Accuracy: Deep learning models offer better feature representation and classification accuracy compared to traditional methods.
- Automated Feature Extraction: Reduces the need for manual feature engineering through hierarchical learning.
- Comparative Evaluation: Enables performance benchmarking between CNNs, RNNs, and Transformers, helping select the most effective model.
- Scalability: The system is designed to handle large genomic datasets and can be expanded to include more cancer types or genomic features.
- Interpretability and Clinical Relevance: Attention mechanisms and visualization tools help interpret model predictions, making the system more acceptable in a clinical setting.

3.4 REQUIREMENT SPECIFICATION

These requirements ensure smooth development and deployment:

- Platform: Cross-platform (Windows, Linux)
- Programming Language: Python
- IDE: Visual Studio Code
- Functionality: Load genomic datasets, preprocess data, train and evaluate deep learning models, visualize results
- Performance Metrics: Accuracy, Precision, Recall, F1-Score
- Extensibility: Can incorporate additional models, datasets, or interpretability modules
- Security: Ensure privacy and integrity of genomic data
- User Interaction: CLI or optional web UI for research demonstration purposes

3.5 HARDWARE REQUIREMENTS

The hardware requirements for the cancer classification system are as follows:

- Processor: Intel Core i5 or above
- RAM: 8 GB minimum (16 GB recommended)
- GPU: NVIDIA CUDA-enabled GPU
- Storage: Minimum 100 GB free space for datasets and models
- Monitor: Standard 15.6" or larger
- Input Devices: Standard keyboard and optical mouse

3.6 SOFTWARE REQUIREMENTS

The software requirements for the cancer classification system are as follows:

- **Programming Language:** Python 3.8 or higher
- **Development Tools:** VS Code
- **Libraries:** TensorFlow/Keras or, NumPy, Pandas, Scikit-learn, Matplotlib,
- **Version Control:** Git

3.7 PYTHON

Python is a versatile, high-level programming language that supports multiple paradigms including object-oriented and functional programming. Its simplicity and extensive ecosystem make it ideal for scientific computing and deep learning research. Libraries such as TensorFlow, PyTorch, and Scikit-learn provide powerful tools for model development, training, and evaluation. Python's readability and active open-source community enhance rapid prototyping and reproducibility in research projects like cancer classification.

3.8 VS CODE

Visual Studio Code is a powerful, lightweight, and extensible source code editor developed by Microsoft. It supports development in a wide range of programming languages including Python, which is extensively used in this project. VS Code is available for Windows, macOS, and Linux and has become a popular choice among developers due to its user-friendly interface and extensive extension marketplace. VS Code provides rich support for editing, debugging, version control, and task running. It includes features like IntelliSense (code completion and suggestion), built-in Git integration, syntax highlighting, and code navigation VS Code serves as the Integrated Development Environment (IDE) to write, test, and debug the Python scripts for the deep learning models. It allows seamless integration with virtual environments, version control systems like Git, and AI development tools like TensorFlow and PyTorch. With extensions like Python, Jupyter, and GitLens, VS Code enhances productivity and simplifies the development process, making it an ideal choice for implementing and managing the cancer classification models in this study.

Chapter 4

SYSTEM DESIGN

4.1 System Design

The system design for the Cancer Classification System utilizing genomic data integrates several components, each aimed at addressing the complexity of genomic datasets and ensuring effective cancer prediction. The core of the system involves deep learning models—CNNs, RNNs, and Transformers—trained to analyze large-scale genomic data. The system is designed with a focus on computational efficiency, interpretability, and predictive accuracy. The user interface of the system is designed for medical professionals, providing intuitive access to predictions, insights, and visualizations of the cancer classification process. At the backend, the system leverages machine learning algorithms capable of processing genomic data, such as gene expression data and key genetic markers. The models are trained using publicly available genomic datasets and assessed for their ability to classify different types of cancer based on various genomic features. The system is designed to recommend personalized treatment strategies based on the classification results, assisting doctors in making data-driven decisions for early diagnosis and treatment planning.

4.2 System Architecture

The architecture for the Cancer Classification System is built on a multi-layered design. At the core is the backend server, which houses the machine learning models and is responsible for processing genomic data, training deep learning models, and generating predictions.

1. Frontend Layer: The user interface (UI) is accessible via a web or mobile application. This interface allows doctors to input patient genomic data and receives real-time cancer classification predictions. It displays the results, visualizations, and model explanations in a user-friendly format.
2. Backend Layer: The backend is responsible for managing data, training deep learning models, and handling prediction requests. The backend integrates deep

learning architectures such as CNNs, RNNs, and Transformers, which are trained to detect and classify genetic patterns in the genomic data.

3. Database Layer: The database holds patient information, genomic data, and model results. It is designed for fast access and retrieval of large datasets, which are essential for training models and performing predictions.

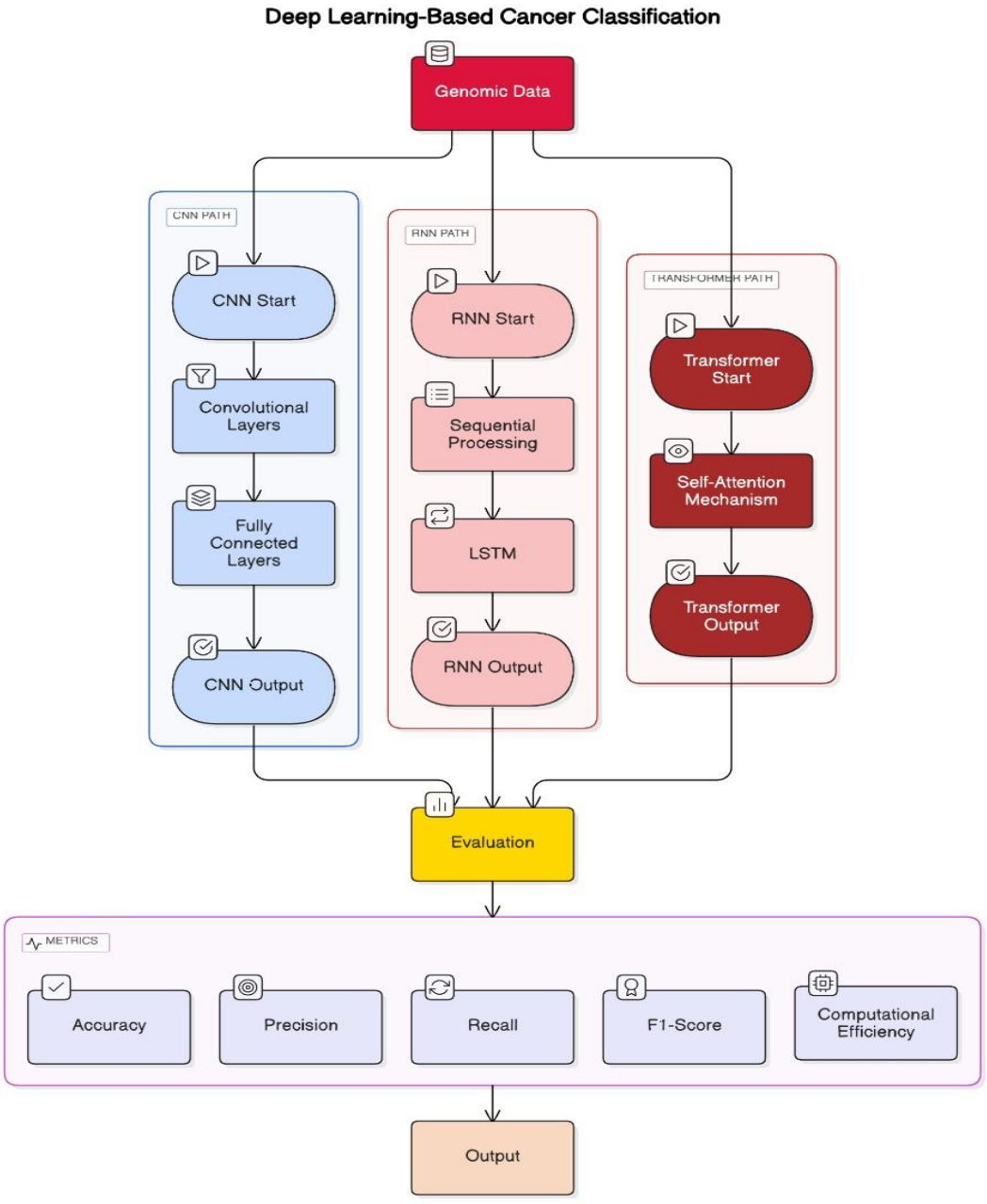


Figure 4.1 Architecture Diagram of the proposed System

4.3 Genomic Data Processing

Processing genomic data is a critical component of this system. The data is typically unstructured and noisy, which can hinder effective model training. The following steps are involved in data processing:

- Data Collection: Genomic datasets, such as gene expression data and DNA sequencing information, are collected. Public datasets like TCGA.
- Data Preprocessing: This step involves cleaning the data by handling missing values, normalizing gene expression values, and transforming data into formats suitable for training machine learning models.
- Feature Selection: Relevant genetic features and markers that contribute to cancer classification are selected. Feature selection helps reduce the dimensionality of the data, focusing on the most predictive genes or markers.

4.4 Deep Learning Models

Convolutional Neural Networks, Recurrent Neural Networks, and Transformer models are employed to address the problem of cancer classification:

1. **CNNs**: Convolutional Neural Networks are effective in genomic analysis due to their ability to automatically extract features from high-dimensional data. They treat gene expression profiles as structured input, identifying patterns linked to specific cancer types. CNNs use filters to detect local patterns such as co-expressed gene clusters or mutation hotspots. Through multiple layers, they learn complex interactions between genes. Their architecture preserves spatial relationships, which is useful for genomic data with positional dependencies.

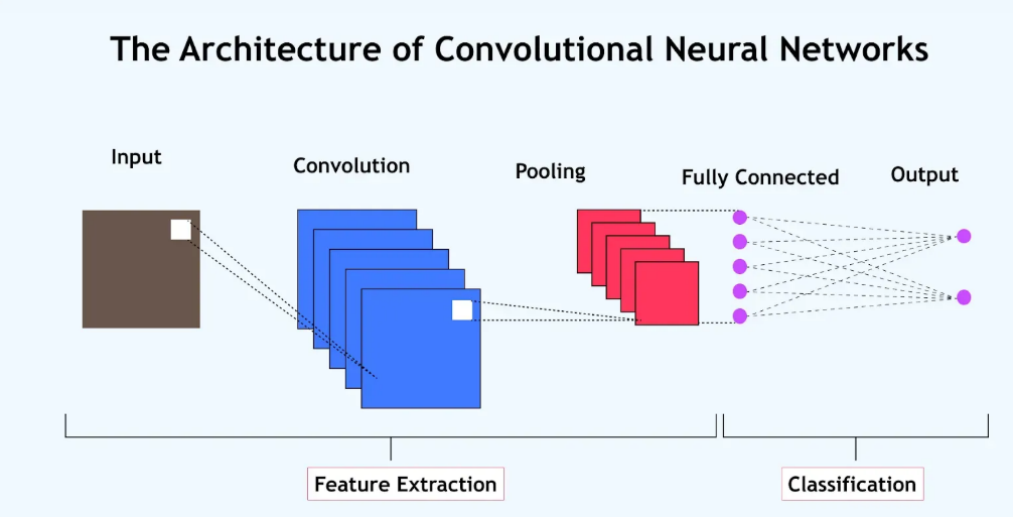


Figure 4.2: Architecture Diagram for CNN

2. **RNNs**: Recurrent Neural Networks are ideal for handling sequential genomic data, especially where gene expression changes over time or follows an ordered pattern. They maintain a memory of previous inputs, allowing them to capture dependencies and relationships across gene sequences. This makes RNNs effective for modeling temporal dynamics in cancer progression. Advanced variants like LSTM improve their ability to learn long-term patterns by addressing issues like vanishing gradients. Although RNNs can be computationally intensive, they provide a deep contextual understanding of how gene expression evolves, offering valuable insights for cancer classification based on time-series or sequential genomic data.

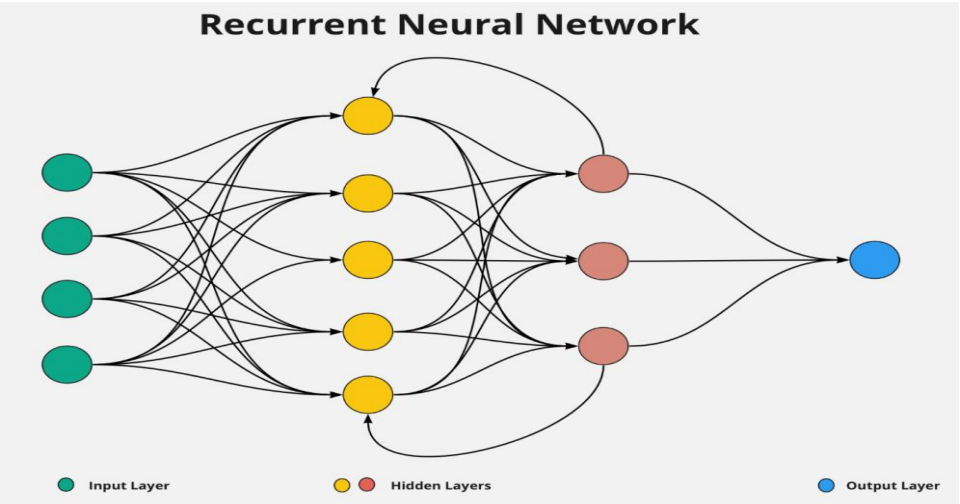


Figure 4.3: Architecture Diagram of RNN

3. **Transformer Models**: Transformers are advanced deep learning models known for their self-attention mechanisms, which allow them to capture long-range dependencies in data without relying on sequential processing. Unlike RNNs, Transformers process all input data simultaneously, making them highly efficient and scalable for large genomic datasets. In cancer classification, they excel at identifying complex interactions between genes, regardless of their positions in the sequence. This enables them to detect subtle patterns and correlations that traditional models might miss. Their ability to focus on the most relevant features in the data leads to high prediction accuracy and makes them particularly powerful for analyzing intricate genomic relationships in cancer diagnosis.

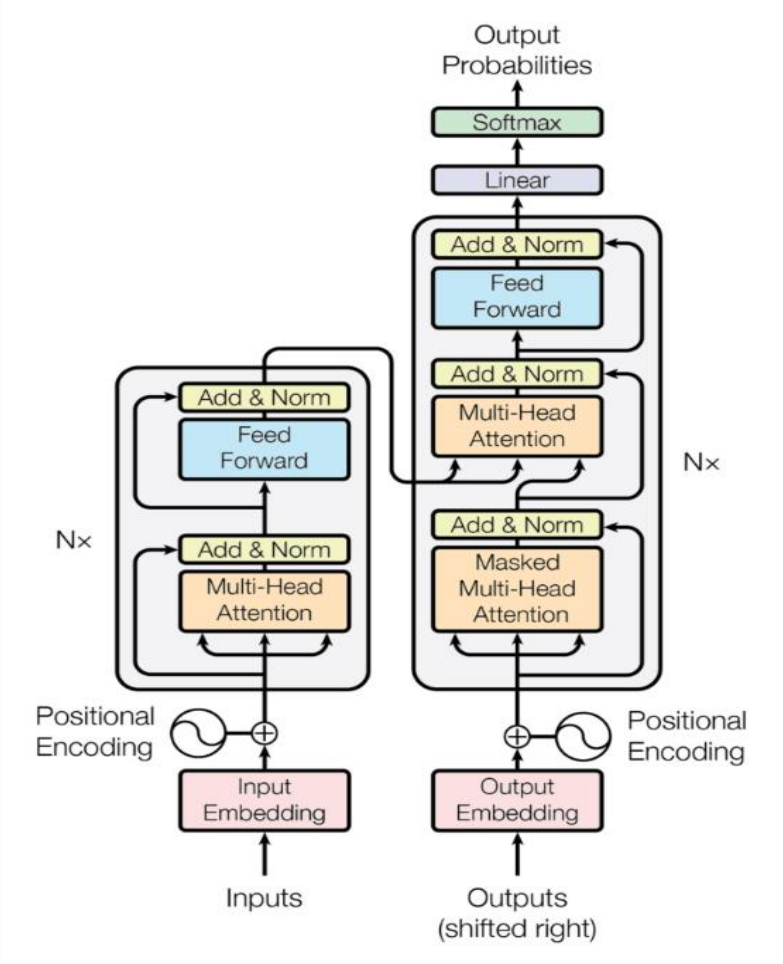


Figure. 4.4: Architecture Diagram of Transformer model

4.5 Algorithm Used

Deep learning models like CNNs, RNNs, and Transformers play a crucial role in cancer classification by analyzing genomic data. They help classify different cancer types, enabling early detection and personalized treatment. These models predict disease progression, helping doctors assess severity and survival chances. They also forecast treatment responses based on genomic profiles, enhancing precision medicine strategies. Additionally, AI-driven techniques uncover biomarkers linked to cancer behaviors, contributing to targeted therapies. By offering accurate predictions, they assist oncologists in diagnosis and treatment decision-making. Their integration in cancer genomics improves patient stratification and personalization of care, leading to better clinical outcomes.

4.6 Evaluation Metrics

The effectiveness of each model is evaluated using the following metrics:

Accuracy: Measures the percentage of correctly classified instances.

- Helps assess overall model reliability in distinguishing between cancerous and non-cancerous cases.
- Can be misleading in imbalanced datasets, requiring complementary metrics for better evaluation.

Precision: Indicates the percentage of correct positive predictions.

- Reduces false positives, ensuring fewer healthy patients are incorrectly diagnosed with cancer.
- Important for clinical applications where incorrect diagnoses can lead to unnecessary treatments.

Recall: Measures the percentage of actual positives correctly predicted by the model.

- Crucial for identifying cancer cases early and minimizing false negatives.
- High recall ensures potential cancer cases are detected, reducing the chances of missed diagnoses.

F1-Score: The harmonic mean of precision and recall, offering a balanced metric when dealing with class imbalances.

- Helps provide a single performance measure when precision and recall show trade-offs.
- Useful in datasets with uneven class distribution to prevent skewed evaluation results.

.

CHAPTER 5

IMPLEMENTATION

5.1 Tools and Technologies Used

To implement the deep learning-based cancer classification system, the following tools and technologies were employed.

Table 5.1: Components and technologies used for implementation

Component	**Technology/Tool**
Programming Language	Python 3.8+
IDE	Visual Studio Code (VS Code)
Libraries/Frameworks	TensorFlow, Keras, NumPy, Pandas, Scikit-learn, Matplotlib, Seaborn
Data Preprocessing	Pandas, NumPy, Scikit-learn
Visualization	Matplotlib, Seaborn
Hardware	System with 8GB+ RAM and NVIDIA GPU (CUDA-enabled)
OS	Windows 10

Python was chosen due to its extensive ecosystem for data science and machine learning. TensorFlow and Keras provided the necessary deep learning capabilities for implementing CNN, RNN, and Transformer models. Visual Studio Code served as the primary development environment.

5.2 Dataset Overview and Preprocessing

The project utilized a publicly available genomic dataset, which includes gene expression data and cancer type labels. Since raw genomic data can be high-dimensional and noisy, proper preprocessing was crucial. The dataset link is provided below here https://www.kaggle.com/datasets/brsahan/genomic-data-for-cancer.The input field and output field of the dataset are Gene One and Gene Two whereas Cancer present is the Output field.

1. Loading the dataset using Pandas.
2. Handling missing values by dropping rows with NA values or using forward fill.
3. Label encoding was applied to convert categorical labels (cancer types) to numeric form.
4. Feature scaling using StandardScaler to normalize gene expression levels.
5. Train-test split using train_test_split from Scikit-learn with a stratified split to maintain label balance.

This preprocessing pipeline ensured the dataset was clean, and in a form suitable for deep learning model input.

5.3 Model Architectures

The core of the project involves implementing and comparing three deep learning models: CNN, RNN, and Transformer. Each model processes genomic data differently, offering diverse perspectives on classification.

5.3.1 Convolutional Neural Network

CNNs are effective at identifying spatial relationships in data. In this case, the input gene expression vectors were reshaped and passed through 1D convolutional layers to capture localized patterns.

Architecture Summary:

- Input Layer (reshaped as [samples, features, 1])
- Conv1D, MaxPooling1D
- Dropout layers for regularization

- Flatten and Dense layers
- Softmax Output layer for multi-class classification

Model Evaluation Metrics**:** Accuracy, Precision, Recall, F1-score, Training Time.

5.3.2 Recurrent Neural Network

RNNs are designed to handle sequential data, making them suitable for modeling gene expression profiles over sequences.

Architecture Summary:

- Input reshaped as sequences
- Simple RNN layer
- Dense layers with ReLU activation
- Sigmoid Output or Softmax

Advantage: Captures long-term dependencies in gene expression data

5.3.3 Transformer Model

Transformers utilize self-attention mechanisms to model long-range dependencies and relationships between all genes simultaneously. They are powerful for extracting contextual information across the entire gene expression profile.

Architecture Summary:

- Input layer reshaped to [samples, 1, features]
- Layer Normalization
- Multi-Head Self-Attention
- Global Average Pooling
- Fully Connected Dense layers with Tanh activation
- Sigmoid or Softmax Output Layer

Advantage**:** Superior handling of complex inter-gene relationships and interpretability through attention scores

5.4 Model Training and Evaluation

Each model was trained separately using:

- 80 Percent of data for training

- 20 Percent for testing
- Validation split from training data for early stopping

Common Training Parameters:

- Optimizer: Adam
- Loss Function: Cross-Entropy (Binary or Categorical)
- Batch Size: 32 or 50
- Epochs: 20–50 with EarlyStopping for overfitting prevention

Performance was evaluated using:

- Accuracy (overall correctness)
- Precision (correctness of positive predictions)
- Recall (coverage of actual positives)
- F1-score (harmonic mean of precision and recall)
- Training time (computational efficiency)

Table 5.2: Model Comparison and Insights

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
CNN	0.90	0.90	0.89	0.89
RNN	0.90	0.90	0.90	0.90
Transformer Model	0.96	0.96	0.94	0.95

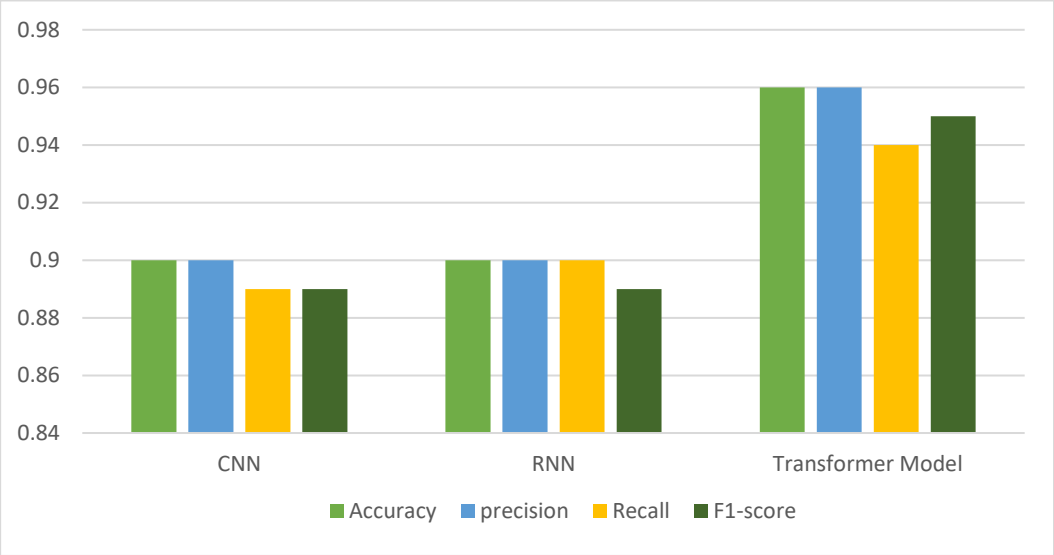


Figure 5.1: Graph for the model comparison

.

5.5.1 METRICS COMPARISON

The Transformer model achieves the highest accuracy (ROC: 0.99) with fewer misclassifications but takes the longest processing time (30 sec). CNN and RNN have similar accuracy (ROC: 0.96) and misclassification rates but are faster, with CNN being the quickest at 4.59 sec.

Table 5.2: Comparison of metrics

Model	ROC curve	Confusion Matrix	Time Taken
CNN	0.96	TP:270, TN:268, FP:35, FN: 30	4.59
RNN	0.96	TP:270, TN:268, FP:35, FN: 30	9.72
Transformer Model	0.99	TP:287, TN:275, FP:25, FN: 13	30.00

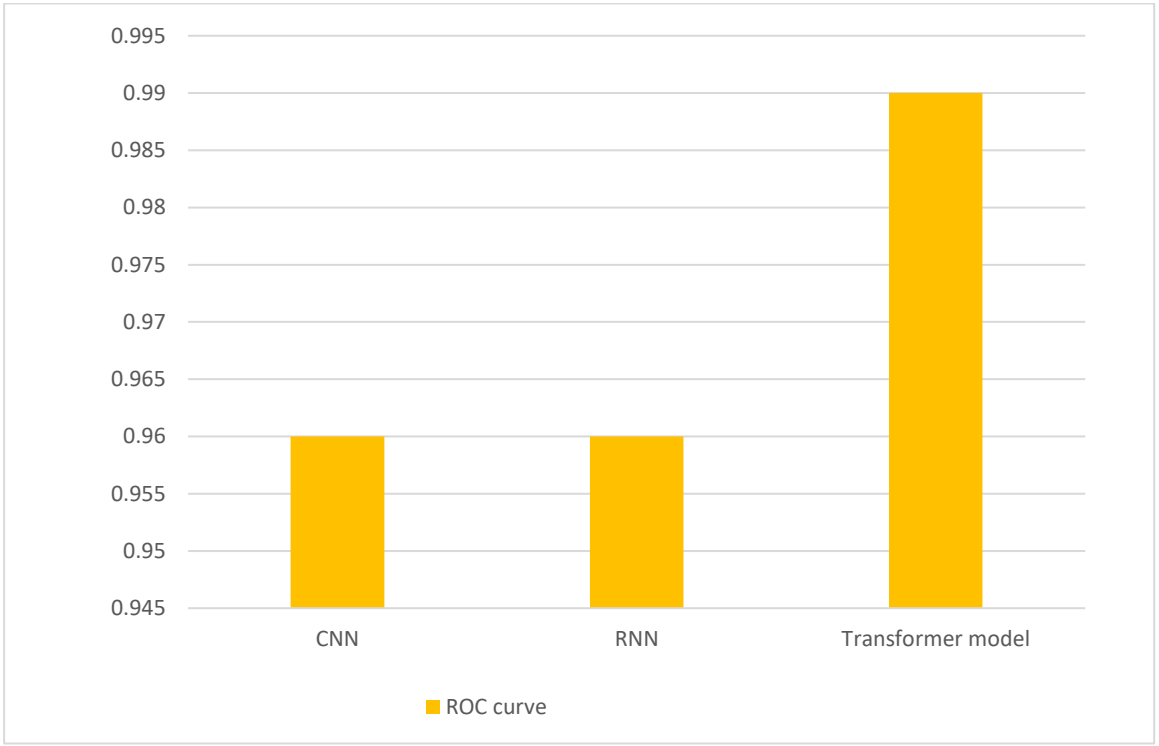


Figure 5.3: ROC curve Comparison

Figure 5.4: Confusion Matric Comparison

5.6 Code Snippets and Execution Logs

The complete source code for preprocessing, training CNN, RNN, and Transformer models, along with performance outputs, is included in **Appendix-2**. Each section is documented with comments for clarity and reproducibility.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

In conclusion, the integration of deep learning models for cancer classification using genomic data represents a significant breakthrough in precision medicine. By employing advanced architectures such as Convolutional Neural Networks, Recurrent Neural Networks, and Transformer models, this study has demonstrated the ability of deep learning techniques to excel in handling complex genomic datasets. These models have shown superior performance in feature extraction, pattern recognition, and classification, enabling more accurate cancer predictions and facilitating early diagnosis.

The comparative analysis of CNNs, RNNs, and Transformer models has provided valuable insights into their strengths, allowing for an informed choice in selecting the most suitable model for specific cancer classification tasks. With an emphasis on performance metrics such as accuracy, precision, recall, and F1-score, the proposed system aims to assist medical professionals in making data-driven decisions, ultimately contributing to improved diagnostic outcomes and personalized treatment planning.

This research highlights the transformative potential of deep learning in the healthcare domain, especially in cancer detection and diagnosis. As deep learning techniques continue to evolve, they can bring about significant improvements in the early detection of cancers and the development of tailored therapeutic strategies, ultimately improving patient outcomes. The growing availability of large-scale genomic datasets and advancements in computational power will continue to refine deep learning models for medical applications. By harnessing the power of deep learning, the future of cancer diagnosis and treatment holds immense promise for improving survival rates and quality of life for patients worldwide.

6.2 FUTURE ENHANCEMENTS

Looking ahead, several future enhancements can be made to further improve the cancer classification system and extend its capabilities:

1. **Incorporating Multi-Omics Data**

   While this study focused on genomic data, incorporating multi-omics data, such as transcriptomics, proteomics, and metabolomics, could provide a more comprehensive understanding of cancer biology. This would enhance the ability of the system to predict cancer subtypes and individual responses to treatments.

2. **Transfer Learning**

   Using transfer learning to leverage pre-trained models on large datasets, such as those from the ImageNet or other public repositories, could enhance the performance of the models, particularly when the available genomic data is limited. This would allow for better generalization and reduce the risk of overfitting.

3. **Real-Time Prediction**

   Integrating real-time genomic data analysis could provide medical professionals with on-the-fly cancer predictions, facilitating rapid diagnosis and treatment planning. Real-time processing would be particularly valuable in clinical settings where quick decision-making is crucial.

4. **Explainable AI**

   One of the key challenges in using deep learning models for healthcare is the interpretability of the results. Future enhancements could focus on developing explainable AI methods to provide insights into why a particular prediction was made, helping doctors trust and understand the model’s reasoning behind its decision.

5. **Improved Performance Metrics**

   Further research could explore additional performance metrics specific to healthcare applications, such as survival rate prediction, progression-free survival, or response to specific therapies, to offer more comprehensive insights beyond traditional classification metrics like accuracy and F1-score.

6. **Integration with Clinical Decision Support Systems**

   To facilitate real-world application, integrating the cancer classification system with existing Clinical Decision Support Systems would streamline workflows and ensure that predictions are directly actionable in clinical environments. This would allow for seamless data flow and decision-making.

7. **Scalability and Efficiency**

   As genomic datasets continue to grow in size and complexity, optimizing the system for computational efficiency will be essential. Future work could focus on enhancing the system's scalability, enabling it to handle large datasets more efficiently, possibly through cloud computing or distributed processing frameworks.

8. **User-Centric Interface**

   A user-friendly interface for oncologists and healthcare professionals is crucial for widespread adoption. Future enhancements should focus on creating intuitive dashboards that display predictions, model confidence, and associated data visualizations in a clear and actionable manner.

9. **Global Collaboration and Dataset Expansion**

   Collaborating with global research institutions and hospitals to create a more diverse and extensive dataset would improve the generalization of the models. Incorporating genetic variations from different populations could enhance the model's ability to predict cancers across various demographics, improving its effectiveness in diverse settings.

10. **Personalized Medicine Integration**

Enhancing the system to integrate personalized medicine approaches could significantly improve treatment outcomes. By leveraging patient-specific genomic and clinical data, the model could recommend tailored therapies that align with an individual's genetic makeup, reducing adverse effects and improving efficacy.

11. **Federated Learning for Privacy-Preserving AI**

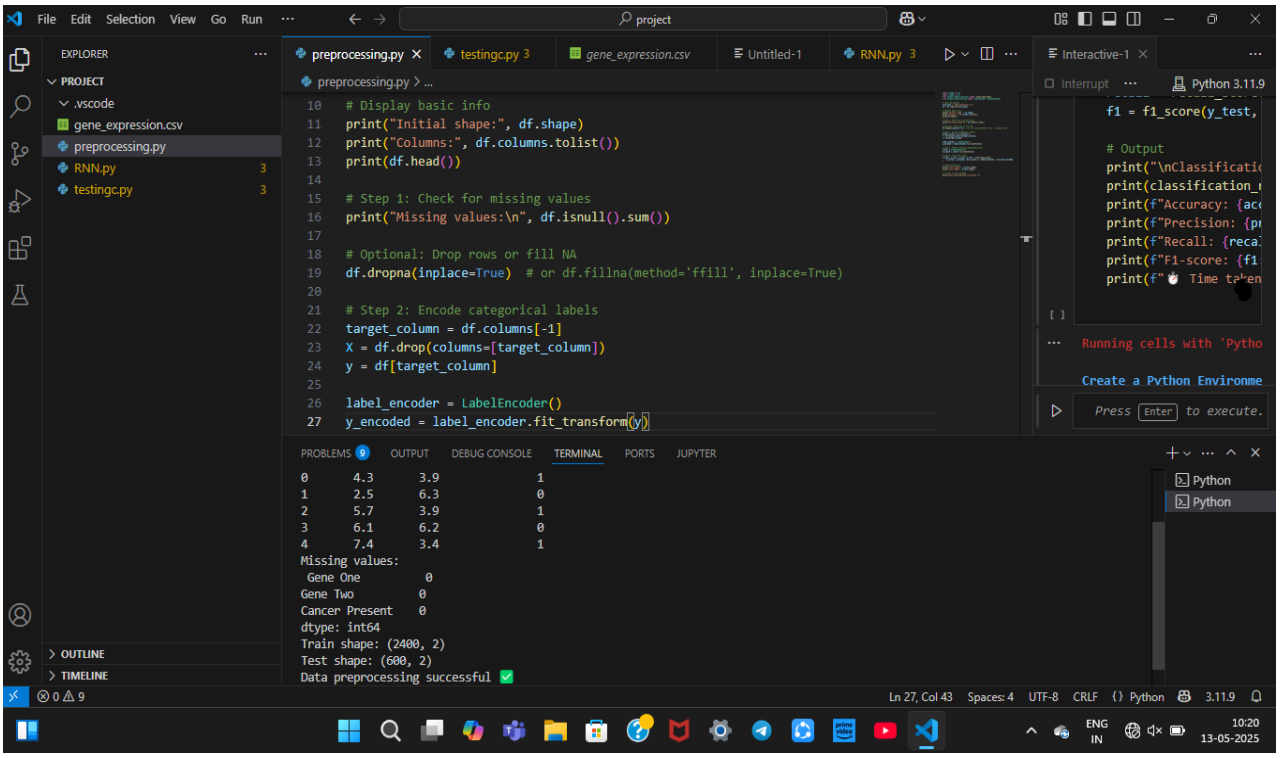
Since genomic data is highly sensitive, implementing federated learning would allow hospitals and research institutions to collaboratively train AI models without sharing raw patient data. This decentralized approach ensures data privacy and security while maintaining predictive accuracy across multiple healthcare centers.

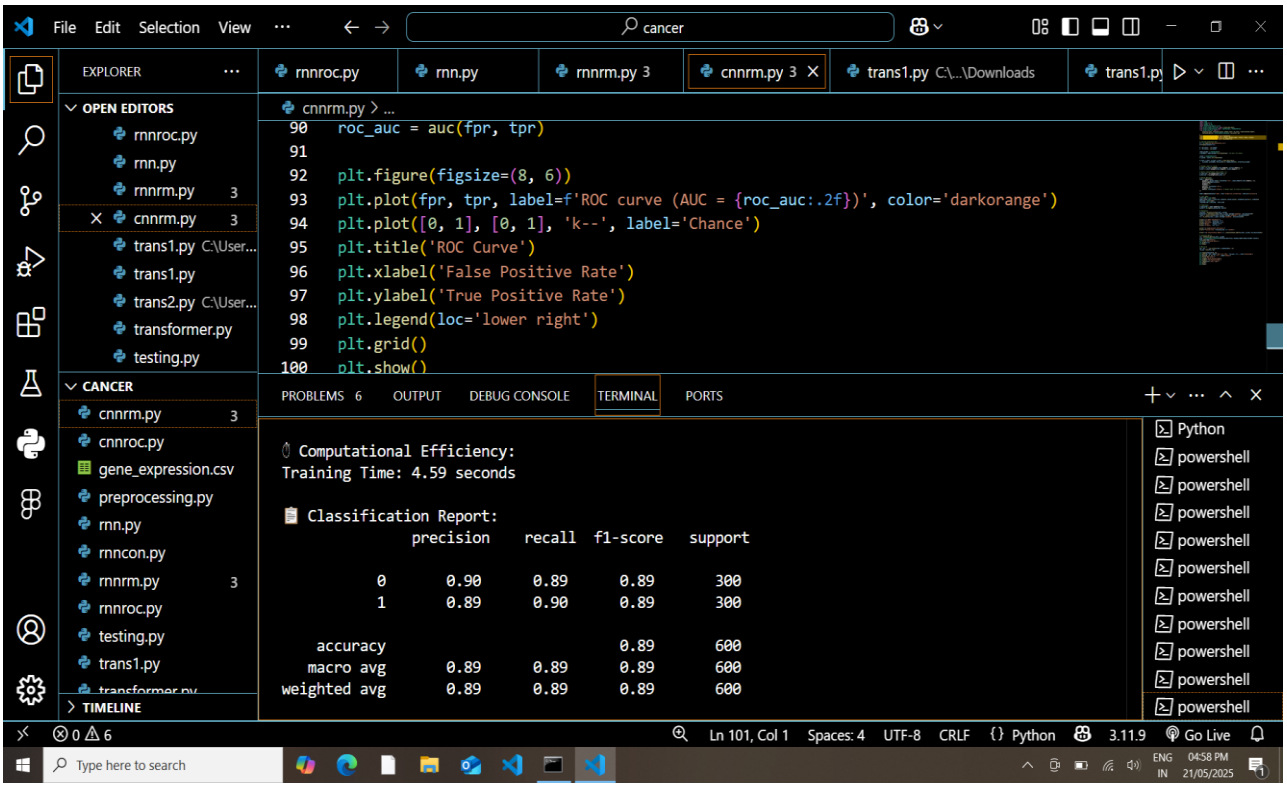
12. **Early Detection and Preventive Diagnostics**

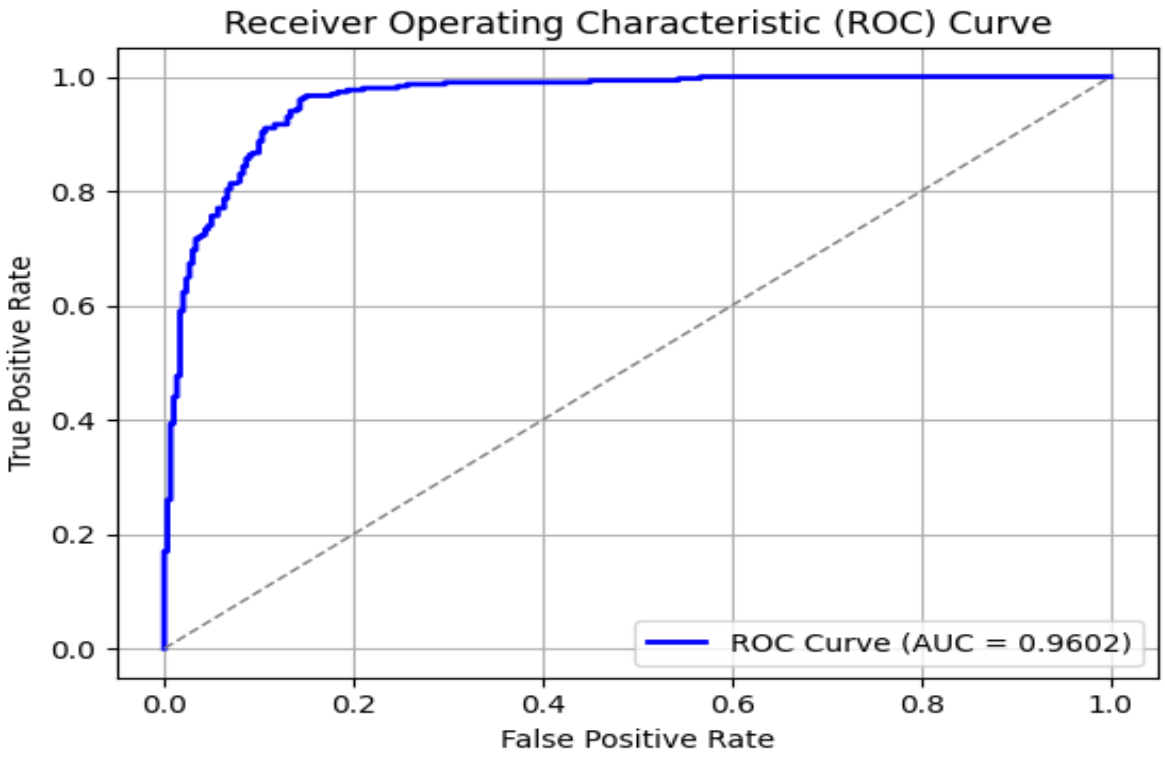
Expanding the system’s capabilities to focus on early cancer detection using longitudinal patient data and health records could help identify potential cancer risks before symptoms appear. AI-driven preventive diagnostics could assist in proactive healthcare interventions, increasing survival rates through earlier treatments.

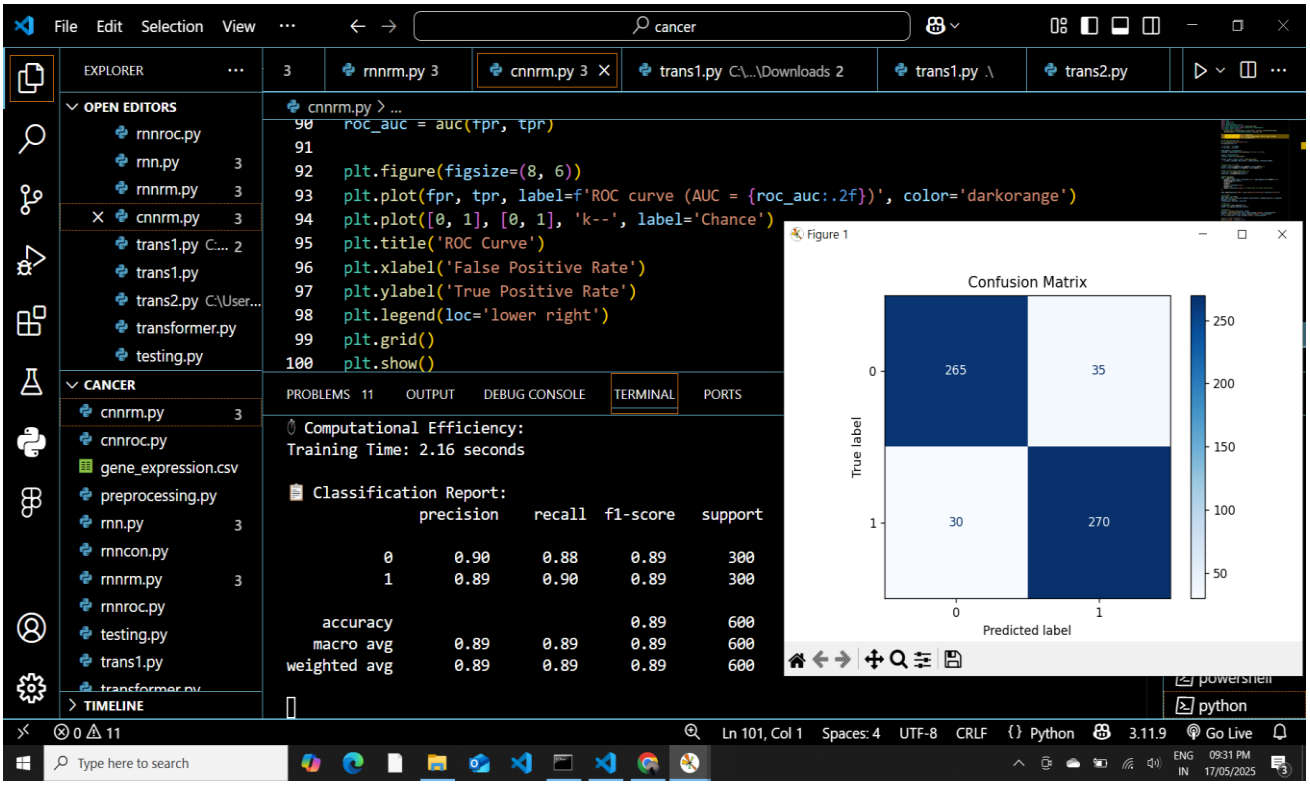
APPENDIX 1

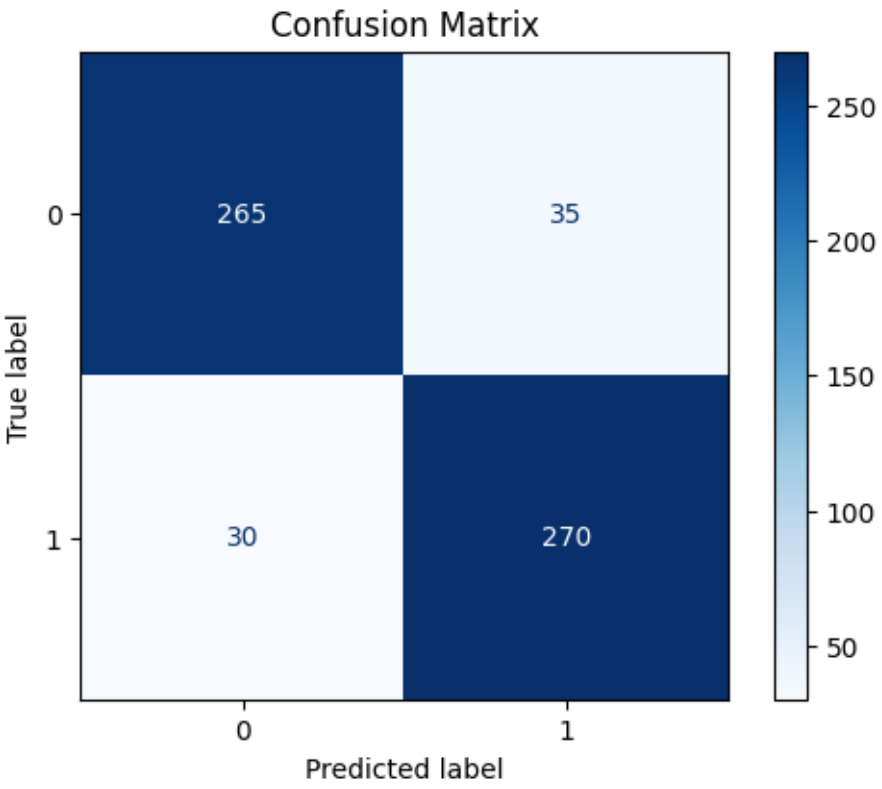
SCREENSHOTS

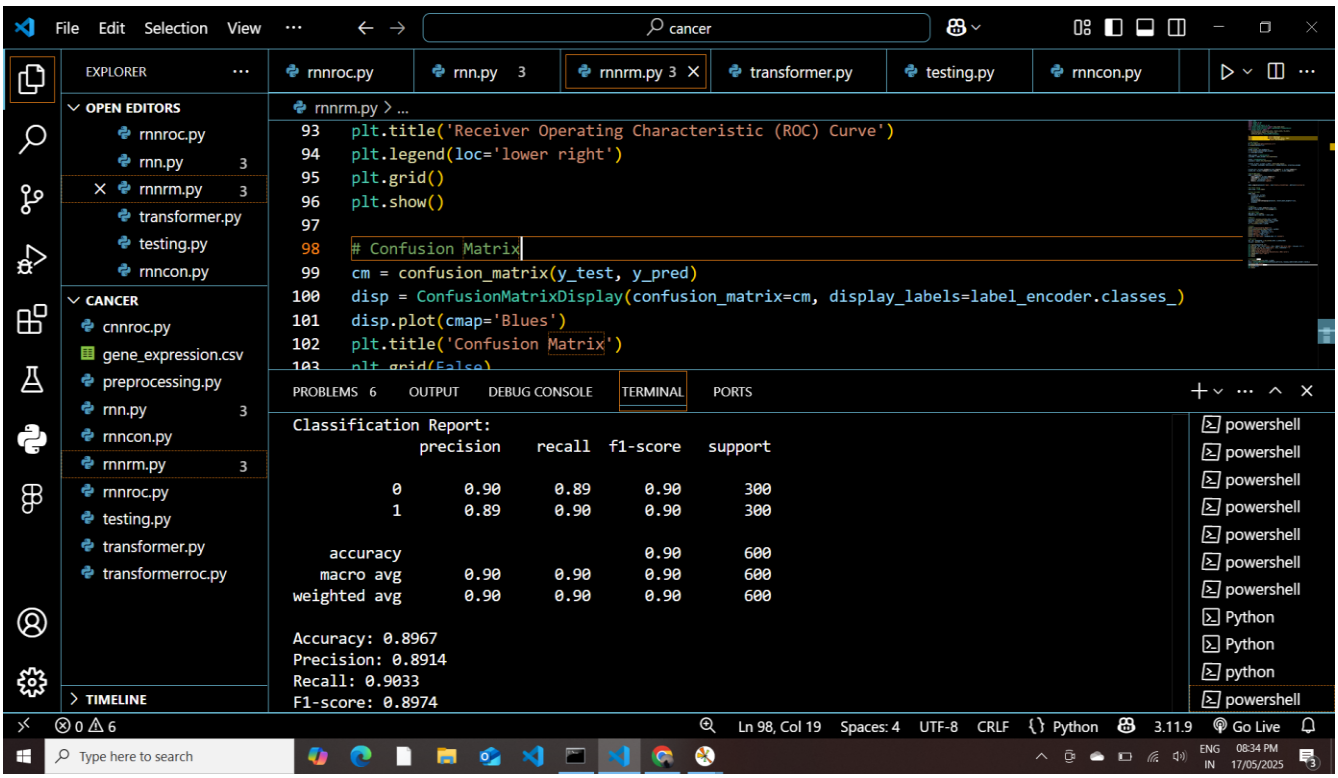


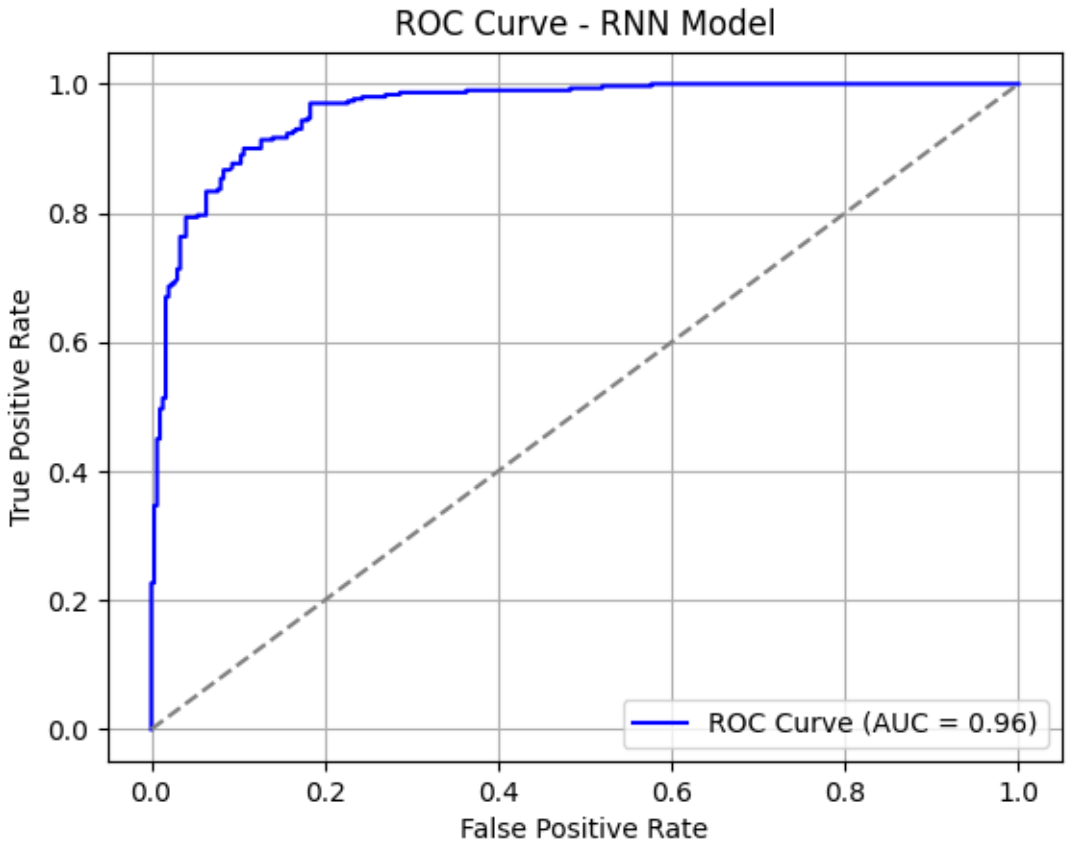


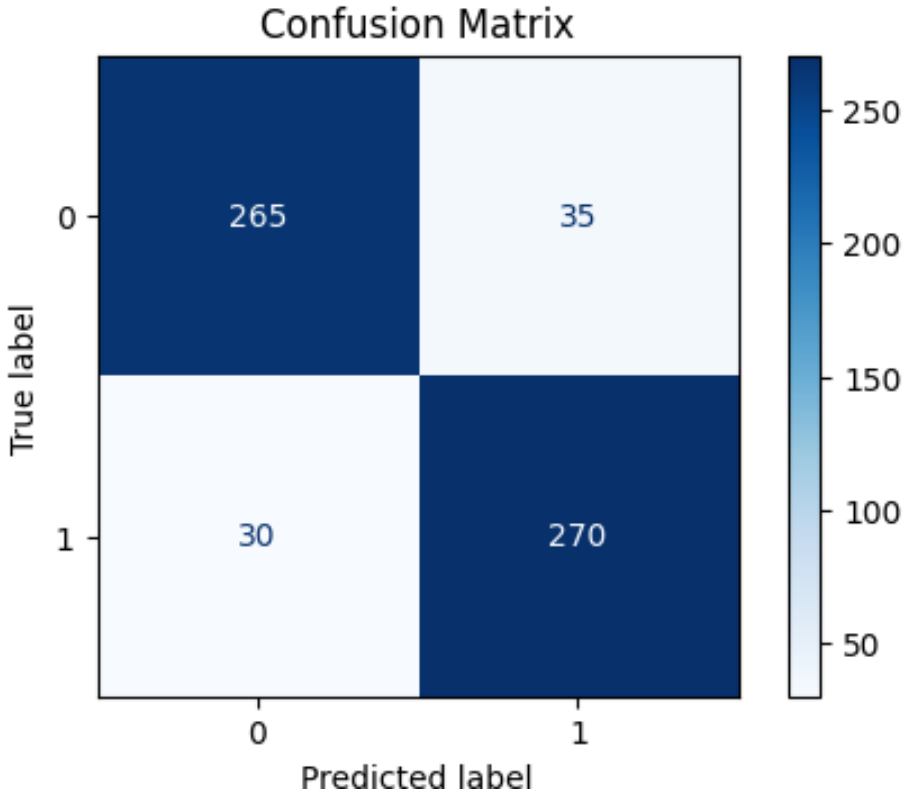

Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC Curve (AUC = 0.9602)
0.0
0.2
0.4
0.6
0.8
1.0

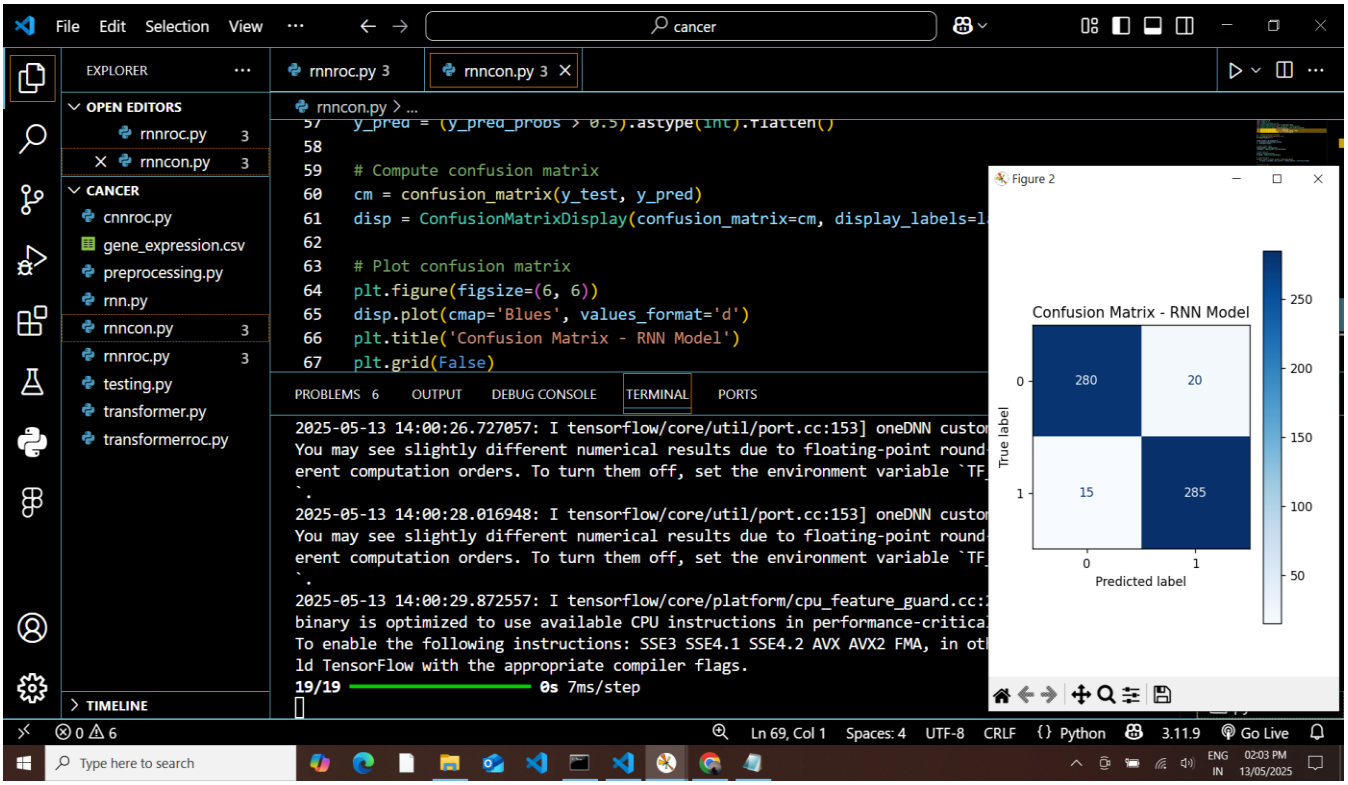

File Edit Selection View
cancer
EXPLORER
OPEN EDITORS
rnnroc.py
rnn.py 3
rnnrm.py 3
cnnrm.py 3
trans1.py C:... 2
trans1.py
trans2.py C:\User...
transformer.py
testing.py
CANCER
cnnrm.py 3
cnnroc.py
gene_expression.csv
preprocessing.py
rnn.py 3
rnncon.py
rnnrm.py 3
rnnroc.py
testing.py
trans1.py
TIMELINE
rnnrm.py 3
cnnrm.py 3
trans1.py C:\...\Downloads 2
trans1.py .\
trans2.py
cnnrm.py > ...
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc_auc:.2f})', color='darkorange')
plt.plot([0, 1], [0, 1], 'k--', label='Chance')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid()
plt.show()
PROBLEMS 11
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Computational Efficiency:
Training Time: 2.16 seconds
Classification Report:
precision recall f1-score support
0 0.90 0.88 0.89 300
1 0.89 0.90 0.89 300
accuracy 0.89 600
macro avg 0.89 0.89 0.89 600
weighted avg 0.89 0.89 0.89 600
Figure 1
Confusion Matrix
True label
Predicted label
265
35
30
270
0
1
250
200
150
100
50
python
Ln 101, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.9 Go Live
Type here to search

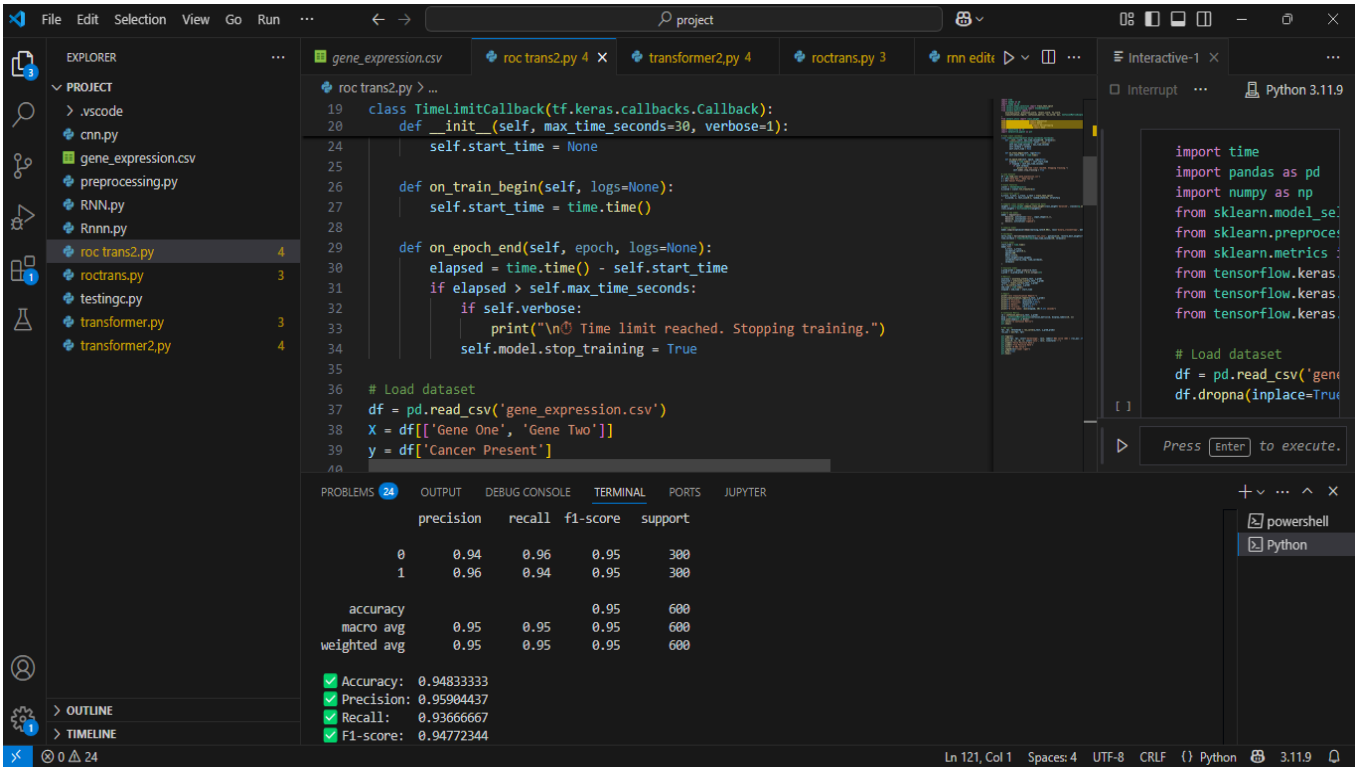

Confusion Matrix
265
35
30
270
0
1
True label
0
1
Predicted label
250
200
150
100
50


File Edit Selection View
cancer
EXPLORER
OPEN EDITORS
rnnroc.py
rnn.py 3
rnnrm.py 3
transformer.py
testing.py
rnncon.py
CANCER
cnnroc.py
gene_expression.csv
preprocessing.py
rnn.py 3
rnncon.py
rnnrm.py 3
rnnroc.py
testing.py
transformer.py
transformerroc.py
TIMELINE
rnnrm.py > ...
93 plt.title('Receiver Operating Characteristic (ROC) Curve')
94 plt.legend(loc='lower right')
95 plt.grid()
96 plt.show()
97
98 # Confusion Matrix
99 cm = confusion_matrix(y_test, y_pred)
100 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=label_encoder.classes_)
101 disp.plot(cmap='Blues')
102 plt.title('Confusion Matrix')
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Classification Report:
precision recall f1-score support
0 0.90 0.89 0.90 300
1 0.89 0.90 0.90 300
accuracy 0.90 600
macro avg 0.90 0.90 0.90 600
weighted avg 0.90 0.90 0.90 600
Accuracy: 0.8967
Precision: 0.8914
Recall: 0.9033
F1-score: 0.8974
powershell
Python
python
Ln 98, Col 19 Spaces: 4 UTF-8 CRLF Python 3.11.9 Go Live
Type here to search


ROC Curve - RNN Model
True Positive Rate
False Positive Rate
ROC Curve (AUC = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0


Confusion Matrix
True label
Predicted label
265
35
30
270
0
1
250
200
150
100
50


File Edit Selection View
cancer
EXPLORER
rnnroc.py 3
rnncon.py 3
OPEN EDITORS
CANCER
cnnroc.py
gene_expression.csv
preprocessing.py
rnn.py
rnncon.py
rnnroc.py
testing.py
transformer.py
transformerroc.py
TIMELINE
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=l
# Plot confusion matrix
plt.figure(figsize=(6, 6))
disp.plot(cmap='Blues', values_format='d')
plt.title('Confusion Matrix - RNN Model')
plt.grid(False)
PROBLEMS 6
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
2025-05-13 14:00:26.727057: I tensorflow/core/util/port.cc:153] oneDNN custo
You may see slightly different numerical results due to floating-point round
erent computation orders. To turn them off, set the environment variable `TF_
`.
2025-05-13 14:00:28.016948: I tensorflow/core/util/port.cc:153] oneDNN custo
You may see slightly different numerical results due to floating-point round
erent computation orders. To turn them off, set the environment variable `TF_
`.
2025-05-13 14:00:29.872557: I tensorflow/core/platform/cpu_feature_guard.cc:
binary is optimized to use available CPU instructions in performance-critica
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in ot
ld TensorFlow with the appropriate compiler flags.
19/19 0s 7ms/step
Figure 2
Confusion Matrix - RNN Model
True label
Predicted label
280
20
15
285
Ln 69, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.11.9
Go Live
Type here to search


File Edit Selection View Go Run
project
EXPLORER
PROJECT
.vscode
cnn.py
gene_expression.csv
preprocessing.py
RNN.py
Rnnn.py
roc trans2.py
roctrans.py
testingc.py
transformer.py
transformer2.py
gene_expression.csv
roc trans2.py 4
transformer2.py 4
roctrans.py 3
class TimeLimitCallback(tf.keras.callbacks.Callback):
def __init__(self, max_time_seconds=30, verbose=1):
self.start_time = None
def on_train_begin(self, logs=None):
self.start_time = time.time()
def on_epoch_end(self, epoch, logs=None):
elapsed = time.time() - self.start_time
if elapsed > self.max_time_seconds:
if self.verbose:
print("\n⏱ Time limit reached. Stopping training.")
self.model.stop_training = True
# Load dataset
df = pd.read_csv('gene_expression.csv')
X = df[['Gene One', 'Gene Two']]
y = df['Cancer Present']
Interactive-1
Python 3.11.9
import time
import pandas as pd
import numpy as np
from tensorflow.keras
from tensorflow.keras
from tensorflow.keras
# Load dataset
df = pd.read_csv('gene
df.dropna(inplace=True
Press Enter to execute.
PROBLEMS 24
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
JUPYTER
precision recall f1-score support
0 0.94 0.96 0.95 300
1 0.96 0.94 0.95 300
accuracy 0.95 600
macro avg 0.95 0.95 0.95 600
weighted avg 0.95 0.95 0.95 600
Accuracy: 0.94833333
Precision: 0.95904437
Recall: 0.93666667
F1-score: 0.94772344
powershell
Python
OUTLINE
TIMELINE
Ln 121, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.11.9

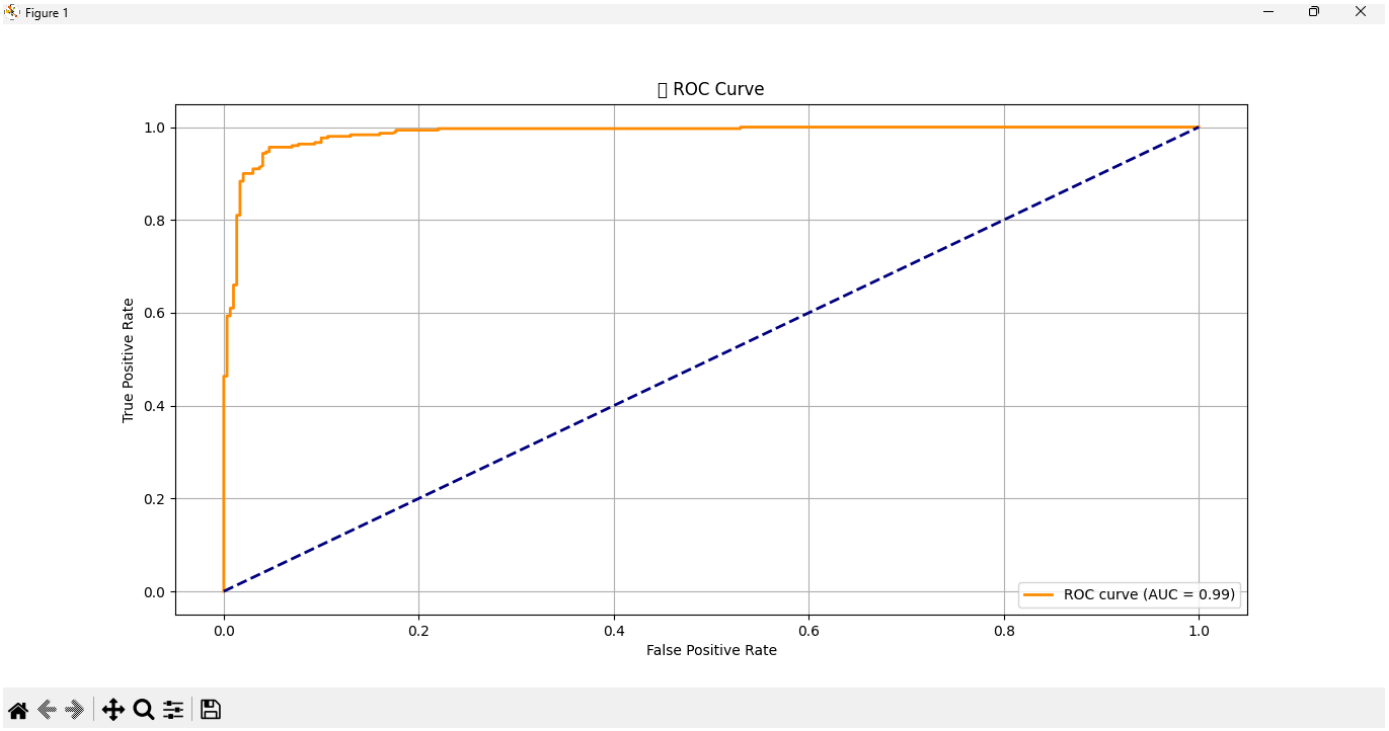
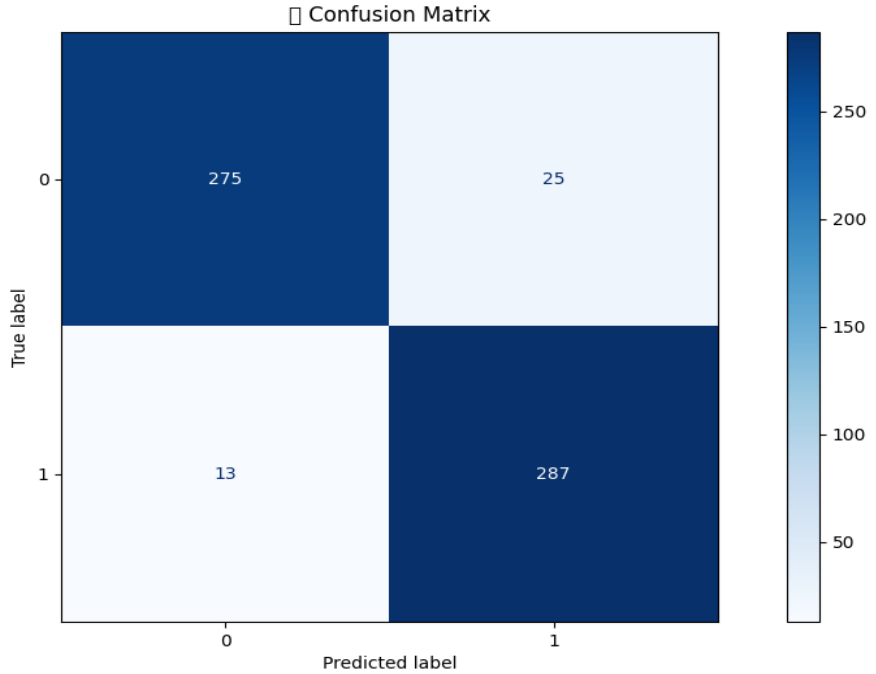
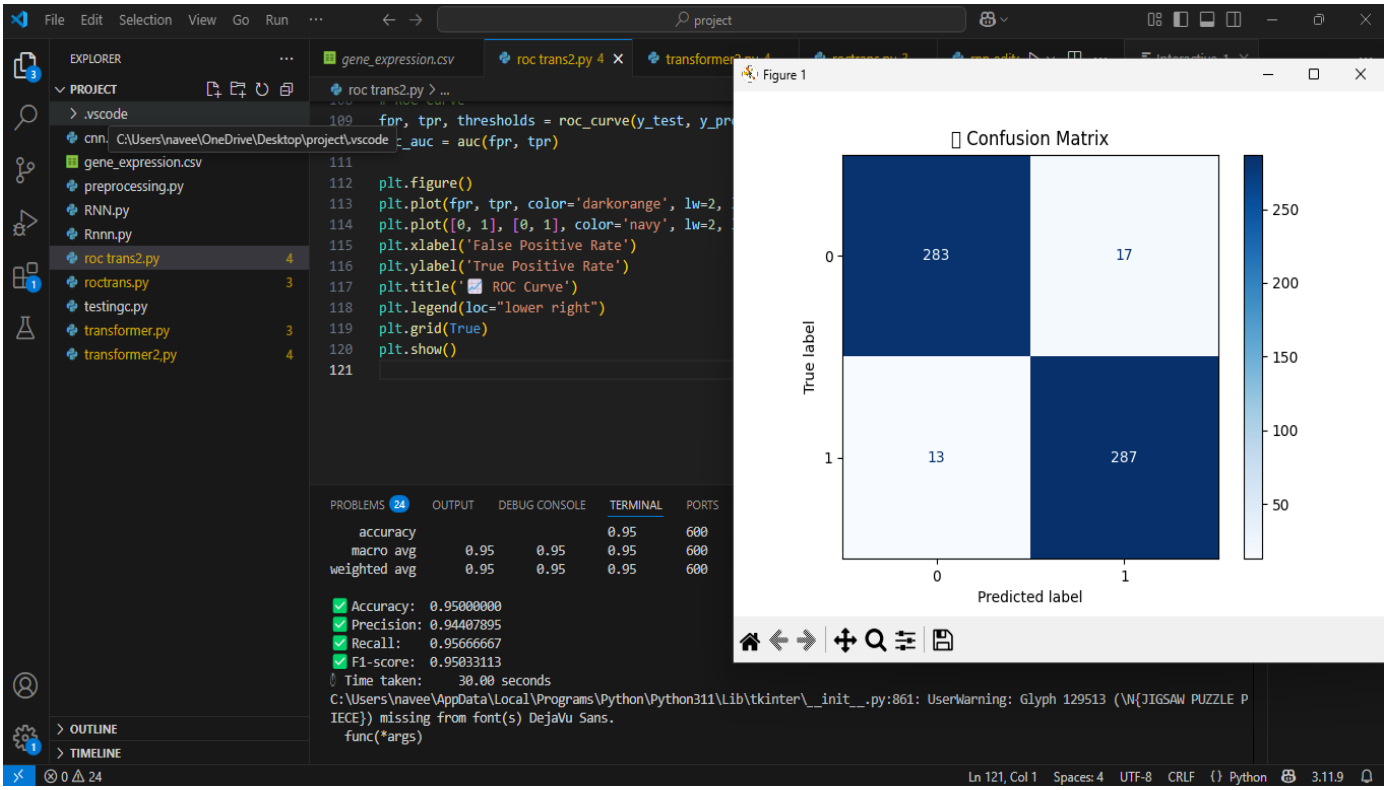


Figure 1
 ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.99)


 Confusion Matrix
True label
Predicted label
275
25
13
287
0
1
250
200
150
100
50


File Edit Selection View Go Run
project
EXPLORER
PROJECT
.vscode
gene_expression.csv
preprocessing.py
RNN.py
Rnnn.py
roc trans2.py 4
roctrans.py 3
testingc.py
transformer.py 3
transformer2.py 4
gene_expression.csv
roc trans2.py 4
roc trans2.py > ...
fpr, tpr, thresholds = roc_curve(y_test, y_pr
_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2,
plt.plot([0, 1], [0, 1], color='navy', lw=2,
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(' ROC Curve')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
PROBLEMS 24
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
accuracy 0.95 600
macro avg 0.95 0.95 0.95 600
weighted avg 0.95 0.95 0.95 600
Accuracy: 0.95000000
Precision: 0.94407895
Recall: 0.95666667
F1-score: 0.95033113
Time taken: 30.00 seconds
C:\Users\navee\AppData\Local\Programs\Python\Python311\Lib\tkinter__init__.py:861: UserWarning: Glyph 129513 (\N{JIGSAW PUZZLE P
IECE}) missing from font(s) DejaVu Sans.
func(*args)
OUTLINE
TIMELINE
Ln 121, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.9
Figure 1
Confusion Matrix
283
17
13
287
0
1
True label
Predicted label
250
200
150
100
50

APPENDIX 2

SOURCE CODE

Data Preprocessing :

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
file_path = 'gene_expression.csv'
df = pd.read_csv(file_path)
print("Initial shape:", df.shape)
print("Columns:", df.columns.tolist())
print(df.head())
print("Missing values:\n", df.isnull().sum())
df.dropna(inplace=True)  # or df.fillna(method='ffill', inplace=True)
target_column = df.columns[-1]
X = df.drop(columns=[target_column])
y = df[target_column]
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded
)
print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
#  Final success message
print("Data preprocessing successful ")
```

Convolutional Neural Network

```
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score, classification_report,
    confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc
)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Load and preprocess data
df = pd.read_csv("gene_expression.csv")
df.dropna(inplace=True)
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)  # 0 and 1 for binary
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded
)
```

```
# Reshape input for Conv1D
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Binary classification (num_classes = 2)
y_train_cat = to_categorical(y_train, 2)
y_test_cat = to_categorical(y_test, 2)

# Define CNN model
model = Sequential([
    Conv1D(32, kernel_size=2, activation='relu', input_shape=(X_train.shape[1], 1)),
    MaxPooling1D(pool_size=1),
    Dropout(0.3),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(2, activation='softmax')  # Output layer for binary classification
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Train model
start_time = time.time()
model.fit(X_train, y_train_cat, epochs=5, batch_size=32, validation_split=0.1,
verbose=0)
end_time = time.time()
training_time = end_time - start_time
```

```
# Predictions
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)
print("\n Model Evaluation:")
print(f"Accuracy:  {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall:    {recall:.4f}")
print(f"F1-Score:  {f1:.4f}")
print("\n Computational Efficiency:")
print(f"Training Time: {training_time:.2f} seconds")
print("\n Classification Report:\n", classification_report(y_test, y_pred,
zero_division=0))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=label_encoder.classes_)
disp.plot(cmap='Blues')
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```

```
# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_probs[:, 1])
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc_auc:.2f})', color='darkorange')
plt.plot([0, 1], [0, 1], 'k--', label='Chance')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid()
plt.show()
```

Recurrent Neural Network

```
import time
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    classification_report, roc_curve, auc,
    confusion_matrix, ConfusionMatrixDisplay
)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, Input
fr.om tensorflow.keras.callbacks import EarlyStopping
```

```
# Load dataset
df = pd.read_csv('gene_expression.csv')
df.dropna(inplace=True)

# Preprocessing
target_column = df.columns[-1]
X = df.drop(columns=[target_column])
y = df[target_column]
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded
)
X_train_rnn = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test_rnn = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))

# Build RNN model
model = Sequential([
    Input(shape=(1, X_train.shape[1])),
    SimpleRNN(32, activation='tanh'),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Start timing
start_time = time.time()
```

```
# Train the model
model.fit(
    X_train_rnn, y_train,
    validation_split=0.2,
    epochs=12,
    batch_size=32,
    callbacks=[EarlyStopping(patience=3, restore_best_weights=True)],
    verbose=1
)

# Predict
y_pred_probs = model.predict(X_test_rnn)
y_pred = (y_pred_probs > 0.5).astype(int)

# End timing
end_time = time.time()
elapsed_time = end_time - start_time

# Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Output
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
```

```
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")
print(f" Time taken: {elapsed_time:.2f} seconds")

# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc_auc:.4f})')
plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid()
plt.show()

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=label_encoder.classes_)
disp.plot(cmap='Blues')
plt.title('Confusion Matrix')
plt.grid(False)
plt.show()
```

Transformer Model :

```
import time
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    classification_report, confusion_matrix, roc_curve, auc, ConfusionMatrixDisplay
)
from sklearn.utils import class_weight
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
import matplotlib.pyplot as plt

# Time limit callback
class TimeLimitCallback(tf.keras.callbacks.Callback):
    def __init__(self, max_time_seconds=30, verbose=1):
        super(TimeLimitCallback, self).__init__()
        self.max_time_seconds = max_time_seconds
        self.verbose = verbose
        self.start_time = None

    def on_train_begin(self, logs=None):
        self.start_time = time.time()
```

```
    def on_epoch_end(self, epoch, logs=None):
        elapsed = time.time() - self.start_time
        if elapsed > self.max_time_seconds:
            if self.verbose:
                print("\n Time limit reached. Stopping training.")
            self.model.stop_training = True

# Load dataset
df = pd.read_csv('gene_expression.csv')
X = df[['Gene One', 'Gene Two']]
y = df['Cancer Present']

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)

# Compute class weights (for imbalanced data)
weights = class_weight.compute_class_weight(class_weight='balanced',
classes=np.unique(y_train), y=y_train)
class_weights = dict(enumerate(weights))

# Build the model
model = Sequential([
    Dense(32, activation='relu', input_shape=(2,)),
```

```
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy',
metrics=['accuracy'])

# Callbacks
early_stop = EarlyStopping(monitor='val_loss', patience=15,
restore_best_weights=True)
time_callback = TimeLimitCallback(max_time_seconds=30, verbose=1)

# Train model
start_time = time.time()
model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=100,
    batch_size=8,
    class_weight=class_weights,
    callbacks=[early_stop, time_callback],
    verbose=1
)
# Evaluate model
y_pred_probs = model.predict(X_test)
y_pred = (y_pred_probs > 0.5).astype(int)
```

```
# Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
end_time = time.time()
elapsed = end_time - start_time

# Report
print("\n Classification Report:")
print(classification_report(y_test, y_pred))
print(f" Accuracy:  {accuracy:9.8f}")
print(f" Precision: {precision:9.8f}")
print(f" Recall:    {recall:9.8f}")
print(f" F1-score:  {f1:9.8f}")
print(f" Time taken: {min(elapsed, 30):9.2f} seconds")

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0, 1])
disp.plot(cmap=plt.cm.Blues)
plt.title(" Confusion Matrix")
plt.show()

# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
```

```
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(' ROC Curve')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

References

1. Alanazi, S. A., Kamruzzaman, M. M., Islam Sarker, M. N., Alruwaili, M., Alhwaiti, Y., Alshammari, N., et al. (2021) ‘Boosting breast cancer detection using convolutional neural network’, Journal of Healthcare Engineering, Vol. 2021, pp. 1–11.
2. Barua, S. and Islam, M. S. (2024) ‘Breast cancer image classification using external attention multilayer perceptron-based transformer’, Proceedings of the 3rd International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE), pp. 1–6.
3. Chiao, J.-Y., Chen, K.-Y., Liao, K. Y.-K., Hsieh, P.-H., Zhang, G. and Huang, T.-C. (2019) ‘Detection and classification of breast tumors using mask R-CNN on sonograms’, Medicine, Vol. 98, No. 19.
4. Daly, M. B., Rosenthal, E., Cummings, S., Bernhisel, R., Kidd, J., Hughes, E., et al. (2023) ‘The association between age at breast cancer diagnosis and prevalence of pathogenic variants’, Breast Cancer Research and Treatment, Vol. 199, No. 3, pp. 617–626.
5. Fathy, W. E. and Ghoneim, A. S. (2019) ‘A deep learning approach for breast cancer mass detection’, International Journal of Advanced Computer Science and Applications, Vol. 10, No. 1, pp. 175–182.
6. George, Y. M., Zayed, H. H., Roushdy, M. I. and Elbagoury, B. M. (2014) ‘Remote computer-aided breast cancer detection and diagnosis system based on cytological images’, IEEE Systems Journal, Vol. 8, No. 3, pp. 949–964.
7. Ibraheem, A. M., Rahouma, K. H. and Hamed, H. F. A. (2021) ‘3PCNNB-Net: Three parallel CNN branches for breast cancer classification through histopathological images’, Journal of Medical and Biological Engineering, Vol. 41, No. 4, pp. 494–503.
8. Iqbal, S. and Qureshi, A. N. (2022) ‘Deep-hist: Breast cancer diagnosis through histopathological images using convolution neural network’, Journal of

Intelligent and Fuzzy Systems, Vol. 43, No. 1, pp. 1347–1364.

9. Islam, T., Hoque, M. E., Ullah, M., Islam, T., Nishu, N. A. and Islam, R. (2024) ‘CNN-based deep learning approach for classification of invasive ductal and metastasis types of breast carcinoma’, Cancer Medicine, Vol. 13, No. 16, pp. 70069.
10. LeCun, Y., Bengio, Y. and Hinton, G. (2015) ‘Deep learning’, Nature, Vol. 521, No. 7553, pp. 436–444.
11. Liu, S., Xu, C., Zhang, Y., Liu, J., Yu, B., Liu, X., et al. (2018) ‘Feature selection of gene expression data for cancer classification using double RBF-kernels’, BMC Bioinformatics, Vol. 19, No. 1, pp. 1–14.
12. Rathi, V. P. and Palani, S. (2012) ‘Brain tumor MRI image classification with feature selection and extraction using linear discriminant analysis’, arXiv preprint.
13. Sreelekshmi, V., Pavithran, K. and Nair, J. J. (2024) ‘SwinCNN: An Integrated Swin Transformer and CNN for Improved Breast Cancer Grade Classification’, Vol. 12.
14. Zhang, J., Saha, A., Zhu, Z. and Mazurowski, M. A. (2018) ‘Breast tumor segmentation in DCE-MRI using fully convolutional networks with an application in radiogenomics’, Medical Imaging: Computer-Aided Diagnosis, Vol. 10575.