

ANKARA YILDIRIM BEYAZIT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES



**A ROBUST DEEP-LEARNING-BASED DETECTOR
FOR PRE-MIRNA CLASSIFICATION**

Ph.D. Thesis by

Abdulkadir TAŞDELEN

Department of Computer Engineering

August, 2021

ANKARA

A ROBUST DEEP-LEARNING-BASED DETECTOR FOR PRE-MIRNA CLASSIFICATION

A Thesis Submitted to

The Graduate School of Natural and Applied Sciences of

Ankara Yıldırım Beyazıt University

**In Partial Fulfillment of the Requirements for the Degree of Doctor of
Philosophy in Computer Engineering, Department of Computer Engineering**

by

Abdulkadir TAŞDELEN

August, 2021

ANKARA

PH.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**A Robust Deep-Learning-Based Detector For Pre-miRNA Classification**” completed by **Abdulkadir TAŞDELEN** under the supervision of **Assoc. Prof. Dr. Baha ŞEN** and we certify that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Baha ŞEN

Supervisor

Prof. Dr. Fatih Vehbi ÇELEBİ

Jury Member

Prof. Dr. H. Haldun GÖKTAŞ

Jury Member

Prof. Dr. Şahin EMRAH

Jury Member

Assoc. Prof. Dr. Yakup KUTLU

Jury Member

Prof. Dr. Sadettin ORHAN

Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information, and documents are obtained in the framework of academic and ethical rules,
- All information, documents, and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of the above statements, I accept to renounce all my legal rights.

Date:

Signature:

Name & Surname: Abdulkadir TAŞDELEN

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest appreciation to my supervisor Assoc. Prof. Dr. Baha ŞEN for his unlimited support and helpful recommendations throughout my thesis. Without his guidance and persistent support, this dissertation would not have been possible.

I also thank Prof. Dr. Fatih V. ÇELEBİ and Prof. Dr. H. Haldun GÖKTAŞ for being my major committee members. Their suggestions and comments motivated me and guided me at all times.

Next, I would like to thank my jury members, Prof. Dr. Şahin EMRAH and Assoc. Prof. Dr. Yakup KUTLU. Their criticism and comments enabled my thesis to take its final form.

Finally, I would like to thank my wife, children, parents, all my family members, and my friends. They supported me unconditionally throughout my thesis studies. They were with me at every moment and encouraged me whenever I needed them. I am very grateful to all of them for their patience and support.

July 28, 2021

Abdulkadir TAŞDELEN

A ROBUST DEEP-LEARNING-BASED DETECTOR FOR PRE-MIRNA CLASSIFICATION

ABSTRACT

MiRNA (or MicroRNA) is a tiny, single-stranded, and non-coding RNA structure of roughly 20-22 nucleotides. Findings from biological research indicate that it plays a regulatory role in a wide range of endogenous processes.

In computational biology, classifying mature miRNA is not efficient since its short length and limited features. Thus, scientists are using precursor miRNAs with longer sequences and more structural features. Pre-miRNAs can be grouped as mirtrons and canonical miRNAs. The main differences come from their biogenesis process. In contrast to canonical miRNAs, mirtrons are less conserved. And also it is not easier to be identified. The conventional machine-learning-based pre-miRNA classification methods depend on manual feature extraction. Besides, they rely on either structure of sequence or structure of spatial of pre-miRNAs.

In this dissertation, we propose a hybrid deep learning method based on the convolutional neural networks and long-short term memory networks to overcome the limitations of previously developed machine learning methods and obtain robust results. According to the our proposed model's result, in 95 percent confidence interval, we got 0.943 (± 0.014) accuracy, 0.935 (± 0.016) sensitivity, 0.948 (± 0.029) specificity, 0.925 (± 0.016) F1 Score and 0.880 (± 0.028) Matthews Correlation Coefficient. Therefore, the prediction resulted in the best for accuracy (2.51 percent), F1 Score (1.00 percent), and Matthews Correlation Coefficient (2.43 percent) when compared to the closest results. In addition, the average of sensitivity has the highest value as Linear Discriminant Analysis. The results show that the hybrid CNN-LSTM networks can be employed to get higher prediction performance for pre-miRNA classification.

Keywords: Mirtron, canonical miRNA, pre-miRNA classification, convolutional neural networks, long short-term memory

PRE-MİRNA SINIFLANDIRMASI İÇİN DERİN ÖĞRENME TEMELLİ GÜÇLÜ BİR DEDEKTÖR

ÖZ

MiRNA (veya MicroRNA), yaklaşık 20-22 nükleotitten oluşan küçük, tek sarmallı ve kodlamayan bir RNA yapısıdır. Biyolojik araştırmalardan elde edilen bulgular, çok çeşitli endojen süreçlerde düzenleyici bir rol oynadığını göstermektedir.

Hesaplamalı biyolojide, olgun miRNA'nın sınıflandırılması, kısa uzunluğu ve sınırlı özellikleri nedeniyle verimli değildir. Bu nedenle, bilim adamları daha uzun dizilere ve daha yapısal özelliklere sahip öncü miRNA'ları kullanmaktadır. Pre-miRNA'lar, mirtronlar ve kanonik miRNA'lar olarak gruplandırılabilir. Bu yapıların ana farklılıklar biyogenez süreçlerinden kaynaklanmaktadır. Kanonik miRNA'ların aksine, mirtronlar daha az konservatiftir. Ve ayrıca tanımlanması da daha kolay değildir. Geleneksel makine öğrenimi tabanlı pre-miRNA sınıflandırma yöntemleri, manuel özellik çıkarımına bağlıdır. Ayrıca, pre-miRNA'ların sekans yapısına veya uzamsal yapısına dayanırlar.

Bu tezde, daha önce geliştirilmiş makine öğrenmesi yöntemlerinin sınırlamalarını aşmak ve daha iyi sonuçlar elde etmek için evrişimli sinir ağlarına ve uzun kısa süreli bellek ağlarına dayalı hibrit bir derin öğrenme yöntemi öneriyoruz. Önerilen modelimizin sonucuna göre, yüzde 95 güven aralığında, 0,943 ($\pm 0,014$) doğruluk, 0,935 ($\pm 0,016$) duyarlılık, 0,948 ($\pm 0,029$) özgüllük, 0,925 ($\pm 0,016$) F1 Skoru ve 0.880 ($\pm 0,028$) Matthews Korelasyon Katsayısı elde ettik. Modelimiz, en yakın sonuçlarla karşılaştırıldığında doğruluk (yüzde 2,51), F1 Skoru (yüzde 1,00) ve Matthews Korelasyon Katsayısı (yüzde 2,43) için en iyi sonucu verdi. Ayrıca duyarlılık ortalaması da Lineer Diskriminant Analizi gibi en yüksek değere sahiptir. Sonuçlar, hibrit CNN-LSTM ağlarının pre-miRNA sınıflandırma için daha yüksek tahmin performansı elde etmede kullanılabileceğini göstermektedir.

Anahtar Kelimeler: Mirtron, kanonik miRNA, pre-miRNA sınıflandırması, evrişimli sinir ağları, uzun-kısa süreli bellek

CONTENTS

PH.D. THESIS EXAMINATION RESULT FORM.....	ii
ETHICAL DECLARATION	iii
ACKNOWLEDGEMENTS.....	iv
ABSTRACT	v
ÖZ.....	vi
CONTENTS.....	vii
ABBREVIATIONS	ix
LIST OF TABLES	x
LIST OF FIGURES	xi
CHAPTER 1 - INTRODUCTION.....	1
1.1 A brief history of miRNAs.....	2
1.2 The biogenesis of miRNAs	4
1.3 Canonical miRNAs	6
1.4 Mirtrons.....	7
1.5 The miRBase Database	8
CHAPTER 2 - PRE-MIRNA CLASSIFICATION	11
2.1 Convolutional Neural Networks	13
2.2 Long Short-Term Memory Networks	14
2.3 The Hybrid CNN and LSTM Networks.....	16
CHAPTER 3 - MATERIALS AND METHODS	19
3.1 Problem Statement	19
3.2 Training and Test Datasets.....	20
3.3 Resampling.....	20
3.3.1 Hold-out random subsampling.....	21
3.3.2 k -folds random subsampling	21
3.3.3 k -folds cross-validation	21
3.3.4 Leave-one-out cross-validation (LOOCV)	22

3.3.5 Leave-p-out cross-validation (LpOCV)	22
3.3.6 Stratified k-folds cross-validation	22
3.3.7 Imbalanced datasets and the model cross-validation procedure	23
3.4 Preprocessing of Data	24
3.5 The Model Architecture	25
3.6 Activation Functions	29
3.6.1 Sigmoid function.....	29
3.6.2 Hyperbolic tangent function (Tanh).....	30
3.6.3 Rectified linear unit function (ReLU).....	31
3.6.4 Leaky rectified linear unit function (Leaky ReLU)	31
3.7 Optimization Algorithms	32
3.7.1 Adagrad	34
3.7.2 Adadelata	35
3.7.3 RMSprop	35
3.7.4 Adam	36
3.8 Method Evaluation Criteria.....	36
CHAPTER 4 - RESULTS AND DISCUSSIONS	40
4.1 The model summary.....	40
4.2 Performance of the proposed network	41
4.3 Performance comparison.....	43
4.4 Hyper-parameters	44
4.5 Limitations	44
4.6 Future Studies.....	45
CHAPTER 5 - CONCLUSION.....	47
REFERENCES.....	49
APPENDICES	60
Appendix A- Detailed architecture of the model	61
Appendix B- Results of stratified 5-folds Cross-Validation	62
CURRICULUM VITAE.....	99

ABBREVIATIONS

ACC	Accuracy
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DGCR	DiGeorge Syndrome Chromosomal Region
DL	Deep Learning
DNN	Deep Neural Network
DT	Decision Trees
FN	False Negative
FP	False Positives
LDA	Linear Discriminant Analysis
LR	Logistic Regression
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
miRNA	Micro Ribonucleic Acid
ML	Machine Learning
NB	Naive Bayes
NN	Neural Network
nt	Nucleotide
PRE	Precision
RF	Random Forest
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network
SEN	Sensitivity
SGD	Stochastic Gradient Descent
SPE	Specificity
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UTR	Untranslated Region

LIST OF TABLES

Table 1.1 miRNA sequence examples from the miRBase database.	4
Table 1.2 Mirtron sequence examples.....	8
Table 3.1 Distribution of the training and test samples for each fold.	24
Table 3.2 “One-hot” encoding for base sequence.	24
Table 3.3 “One-hot” encoding for outputs.	25
Table 3.4 Detailed parameters for each convolutional.....	27
Table 3.5 The designed model with parameters.....	29
Table 4.1 Performance of the proposed CNN-LSTM network for each fold.....	41
Table 4.2 Performance comparison of pre-miRNA classification.	43

LIST OF FIGURES

Figure 1.1 A brief depiction of the biogenesis of miRNA.....	5
Figure 1.2 Representation of a Canonical miRNA (A) and a Mirtron (B).....	6
Figure 1.3 A miRNA sequence example, <i>hsa-let-7a-1</i> , from miRBase.....	9
Figure 2.1 An example of a simple Neural Network.	11
Figure 2.2 An example of a Deep Neural Network.....	12
Figure 2.3 An example of a convolutional neural network.....	14
Figure 2.4 Brief architecture of an LSTM cell.....	16
Figure 2.5 A basic architecture of hybrid CNN and LSTM network.....	17
Figure 3.1 Visualization of stratified k -folds cross-validation with $k=5$	23
Figure 3.2 A brief architecture with visualization of the model	26
Figure 3.3 Illustration of the convolution with 6 x 4 filter size.	28
Figure 3.4 Illustration of Sigmoid function.....	30
Figure 3.5 Illustration of Hyperbolic Tangent function (Tanh)	30
Figure 3.6 Illustration of Rectified Linear Unit function (ReLU).....	31
Figure 3.7 Illustration of Leaky Rectified Linear Unit function (LReLU)	32
Figure 3.8 Visualization of the confusion matrix.....	38

CHAPTER 1

INTRODUCTION

MiRNA (or MicroRNA) is a tiny, single-stranded, and non-coding RNA structure of roughly 20-24 nucleotides[1]. Findings from biological research indicate that it plays a regulatory role in a wide range of endogenous processes. To explain briefly, they bind to the 5' untranslated region (UTR), coding sequence, or 3'UTR of target messenger RNA (mRNA) post-transcriptionally [2,3]. Therefore, they infer the translation process and alter or prevent the presence of the protein product that should normally be produced[4]. That regulatory role of the miRNA shows that the identification and classification of miRNAs are fundamental problems in computational biology.

Comprehensive databases are being built with newly discovered miRNA sequences. These databases contain both miRNA sequences and annotations. The most popular one, the miRBase (v22) database [5], includes 38,589 hairpin precursors and 48,860 mature microRNAs from 271 organisms such as humans, flies, mice, and worms. Besides, the database has 1,917 hairpin precursors and 2,654 mature miRNA sequences in the human category [5]. Although this number is small, it has an important place in scientific research. Because recent studies show that miRNAs control approximately 33% of all protein-coding genes' activities in mammals [6]. This ratio is a significant indicator of how critical these short sequences are in biological processes. Additionally, many studies reveal that miRNAs are associated with plenty of human diseases such as cancer, cardiovascular diseases, or autoimmune diseases [7,8,17–26,9,27,28,10–16].

It is predicted that the expression of miRNA profiling can be a novel diagnostic tool in the near future. In that context, Saini et al. [29] argue that a precise understanding of the elusive multilevel regulation of miRNA expression is a prerequisite for explaining the cause of a wide variety of diseases. Studies show that miRNA offers potential targets for both the diagnosis and treatment of many diseases. Moreover,

the inclusion of certain miRNA expression profiles as biomarkers could lead to significant advances in facilitating disease diagnosis and treatment. Classification and identification of diseases, and monitoring of their prognosis are also the other outputs for using those biomarkers [29].

When we consider both the roles of miRNAs in protein synthesis and their relationship with diseases, the importance of making a classification with high accuracy becomes clearer. Besides, due to the correlation between miRNAs and different diseases, scientific studies to understand the function and mechanism of miRNAs are increasing rapidly [30,31].

The miRNA biogenesis has different steps and cellular mechanisms. Some of the processes occur in the nucleus and others in the cytoplasm. According to the biogenesis differences, we can group miRNAs into two categories. The first is mirtrons and the second is canonical miRNA [32]. A clear understanding of the different biogenesis structures of the miRNA will contribute to more reliable results of the current studies and higher accuracy of classification. In addition, it is thought that it will have an important place in the development of new tools and databases with different features such as disease diagnosis, classification, monitoring, and treatment [29].

In computational biology, classifying mature miRNA is not efficient since its short length (about 22 nucleotides) and limited features. Thus, scientists are using precursor miRNAs with longer sequences and more structural features. In contrast to canonical miRNAs, mirtrons are less conserved. And also it is not easier to be identified [32]. The conventional machine-learning-based pre-miRNA classification methods depend on manual feature extraction. Besides, they rely on either structure of sequence or structure of spatial of pre-miRNAs.

1.1 A brief history of miRNAs

The first miRNA sequence is discovered in *Caenorhabditis elegans* by Ambros and Ruvkun's studies in 1993. They found that the *lin-4* was a small non-coding RNA.

Interestingly, they discovered that this small sequence was not a protein-coding RNA. Moreover, they revealed that the downregulation of *LIN-14* protein is crucial during the progression of these organisms' first larval stage *-L1-* to the second larval stage *-L2-*. In addition, a gene, *lin-4*, has a significant effect on the progress of the downregulation of the *LIN-14* protein even though it is not an active protein [33–35].

A couple of years later than the first discovery of *lin-4*, two separate groups reported *let-7*, the new miRNA (also the 2nd one), in 2000 [36,37]. Studies have shown that *let-7* includes 21 nucleotides. And it is active in the regulation of timing in transition from the 4th stage *-L4-* to adult *Caenorhabditis elegans*' larval development[36]. Moreover, in recent studies, the homologs of this gene also were discovered in humans and several organisms[38].

In the years that followed the discovery, studies of cloning large numbers of small, noncoding, and approximately 19 to 24 nucleotides in length RNAs from humans, flies, and worms were conducted in many laboratories[2]. The discovery of these large numbers of small RNAs led to the need for existence to be verified and to store them in the online corpus. In 2002, an online database, called miRBase, was launched to meet these requirements[39,40]. This database is flexible to use for searching any miRNA of selected organisms. For instance, according to the listing results of humans, we can select to retrieve detailed information of any sequences like *hsa-mir-100* has with 80 nucleotides-long-sequence “CCUGUUGCCACA AACCRCGUAGA UCCGAACUUGUGGUAAUAGUCCGCACAGCUUGUAUCU AUAGGUAUGGUCUGUUAGG” [5].

Protein synthesis is the basis of living things and has a critical place in vital activities. The direct or indirect effects that occur during protein synthesis may have consequences that negatively affect the vital activities of the cell. Recent studies reveal that miRNAs control about 33% of all protein-coding genes' activities in mammals [6].

In Table 1.1, arbitrarily selected ten miRNA sequence examples derived from the miRBase database are shown [5]. It is clearly understood from the table, the order

and number of the sequences of miRNA are the main features that distinguish them from each other.

Table 1.1 miRNA sequence examples from the miRBase database.

Name	Sequence
hsa-mir-99a	CCCAUUGGCAUAAACCRCGUAGA UCCGAUCUUGUGGUGAAGUG GACCRGCACAAGCUCGCUUCUAUGGGUCUGUGUCAGUGUG
hsa-mir-100	CCUGUUGCCACAAACCRCGUAGA UCCGAACUUGUGGUAUUAGU CCGCACAAGCUUGUAUCUAUAGGUAUGUGUCUGUUAGG
hsa-mir-101-1	UGCCCUGGCUCAGUUAUCACAGUGCUGAUGCUGUCUAUUCUAA AGGUACAGUACUGUGAUAACUGAAGGAUGGCA
hsa-mir-29b-1	CUUCAGGAAGCUGGUUUCAUAUGGUGGUUUAGA UUUAAAUAG UGAUUGUCUAGCACCRAUUUGAAAUCAGUGUUCUUGGGGG
hsa-mir-29b-2	CUUCUGGAAGCUGGUUUCACAUGGUGGCUUAGA UUUUUCCAU CUUUGUAUCUAGCACCRAUUUGAAAUCAGUGUUUUAGGAG
hsa-mir-103a-2	UUGUGCUUUCAGCUUCUUUACAGUGCUGCCUUGUAGCAUUCAG GUCAAGCAGCAUUGUACAGGGCUAUGAAAGAACCRA
hsa-mir-105-1	UGUGCAUCGUGGUCAAAUGCUCAGACUCCUGUGGUGGCUGCUC AUGCACCRACGGAUGUUUGAGCAUGUGCUACGGUGUCUA
hsa-mir-105-2	UGUGCAUCGUGGUCAAAUGCUCAGACUCCUGUGGUGGCUGCUU AUGCACCRACGGAUGUUUGAGCAUGUGCUAUGGUGUCUA
hsa-mir-106a	CCUUGGCCAUGUAAAAGUGCUUACAGUGCAGGUAGCUUUUUG AGAUCUACUGCAAUGUAAGCACUUCUACA UUAACCRAUGG
hsa-mir-16-2	GUUCCACUCUAGCAGCACGUAAAUAUUGGCGUAGUGAAAUAU AUAUAAAACACCRAAUUUACUGUGCUGCUUUAGUGUGAC

1.2 The biogenesis of miRNAs

The miRNA biogenesis has different steps and cellular mechanisms. Some of the processes occur in the nucleus and others in the cytoplasm. According to the biogenesis differences, we can group miRNAs into two categories. The first one is mirtrons and the second one is canonical miRNA [32].

A brief overview of the biogenesis of miRNA including steps in the nucleus and cytoplasm is illustrated is depicted in Figure 1.1.

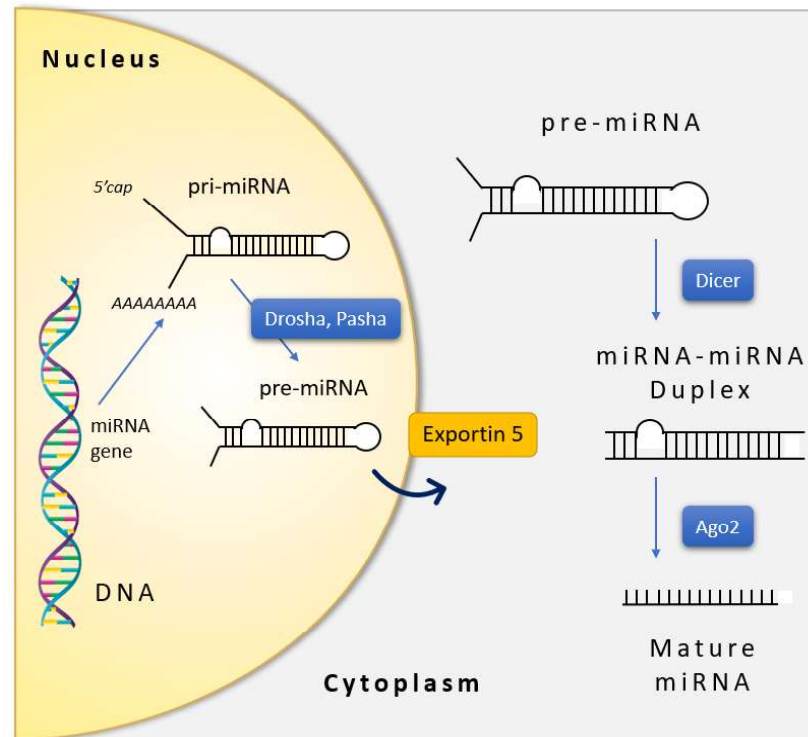


Figure 1.1 A brief depiction of the biogenesis of miRNA.

The initial phase in the biogenesis of a miRNA starts with the transcription of miRNA genes. In this phase, it forms a primary miRNA (pri-miRNA) hairpin. Following the existence of the pri-miRNA, the formation process of the pre-miRNA begins [41–43].

In Figure 1.2, a brief representation of a Canonical miRNA (A) and a Mirtron (B). is illustrated with the structure consisting of stem-loop, terminal-loop, and basal segments[44].

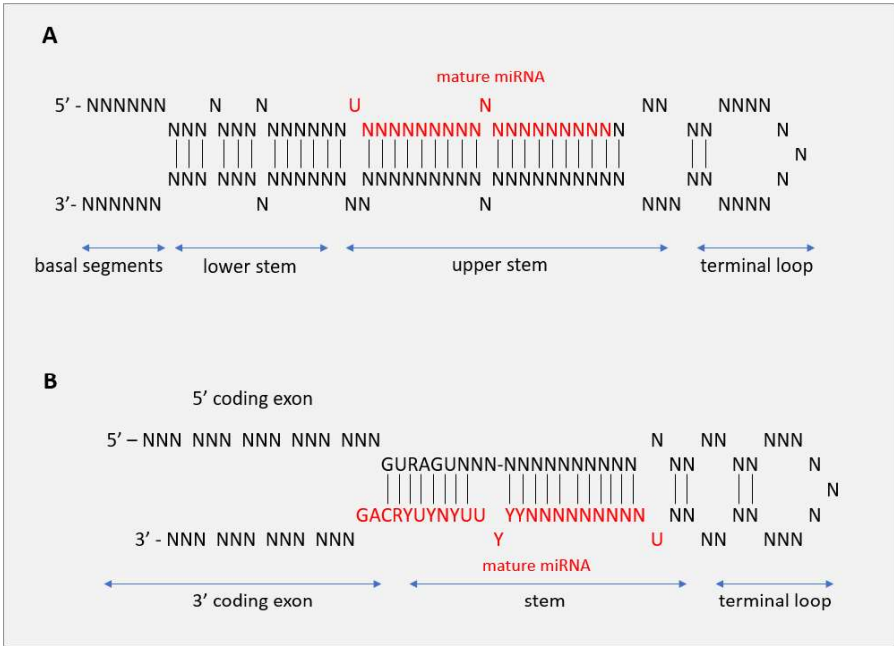


Figure 1.2 Representation of a Canonical miRNA (A) and a Mirtron (B).

1.3 Canonical miRNAs

In the canonical pathway, pre-miRNAs with the hairpin structure are formed by the microprocessor complex consisting of DGCR and Drosha by dividing pri-miRNAs in the nucleus[45,46]. Then, the pre-miRNAs are produced in the nucleus and then transported into the cytoplasm by exportin-5. Following this, pre-miRNAs are cleaved in the cytoplasm into small RNA duplexes by different RNase III enzyme Dicer and ultimately mature miRNA are produced[47,48]. As a result, The biogenesis of the canonical miRNA pathway needs Dicer and Drosha which are two different RNaseIII enzymes [49].

Mark et. al. [49] conducted a study on these two enzymes in mice. Their study reveals that both Dicer and Drosha enzymes are particularly required in canonical miRNA biogenesis.

1.4 Mirtrons

The early discovery of Mirtrons was in *Drosophila melanogaster* [44] and *Caenorhabditis elegans* [50]. When their short RNAs were examined, it was revealed that they have short introns which have the same size as pre-miRNAs.

In the mirtron pathway, for bypassing the nuclear enzyme Drosha, it uses splicing to produce short pre-miRNA hairpin introns[50,51]. The next steps of those pre-miRNAs are in the same pathway as canonical miRNAs[52]. Mirtrons can also be classified into three categories: canonical, 3' tailed, and 5' tailed due to their sequence and structure[51]. Compared to canonical miRNAs, mirtron hairpins and small RNAs have numerous distinguishing features[53–55].

In a study [44], it is investigated that, in *Drosophila*, mirtrons generate approximately 22 nt regulatory RNAs. And also they proved that the mirtrons' biogenesis is different from canonical miRNAs. In addition, their findings reveal that mirtrons are an alternate source of miRNA-type regulatory RNAs and they show particularly that mirtron-like short RNAs may be related to Ago1 and need Ago1 to regulate targets that are matched according to their seeds [44].

Similar to the example above, there are hundreds of research on the discovery of new mirtrons. When we consider all of their results, they reveal that these single-stranded structures have an important role in gene silencing. Moreover, clarifying their pathways has paved the way for revealing more non-canonical pathways. [56].

In Table 1.2, arbitrarily selected ten mirtron sequence examples derived from the dataset (Dataset B) are shown.

Table 1.2 Mirtron sequence examples.

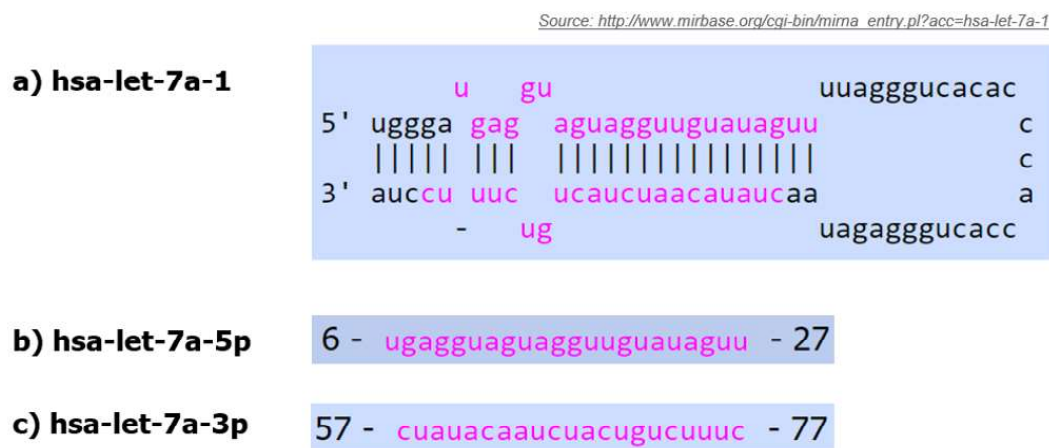
Name	Sequence
uc001api.1.2	TGGGGGTTATGGAACCRAGAAAATGGGATCATGCCATCTGGGT TGTTCCCTGGCTTCCAG
uc001api.1.6	CAGCAGGAGAAAAGCTGGTGAATCAGAAATGACTTCATTCCA CTCGGTTCTCTTCTGTTCTAG
uc001bfj.61	TTGGGGGGTGGTAACTGTAGGAGCAGCCCACAAAGCCCACTG CCCCCGCCCACGTTGCCACCRCCCAACCRCAAG
uc001bny.0	AGGGGGCACTGGGCAGACAGTGTGTGCAAGATAGAGTCCAGG GCCACCRTTCTGGCTGAGCCACACTGAGTCCATTACCRCCCAG
uc001bqa.8	CGGGGCAGAGAGAAGCCATGTGTGTCTGTCTGTCACTGACCRG TGGCTTTCCCTTTGCCTGCAG
uc001dxa.16	ATGGTGGGGATGTCAGGTGTGGGTTGTGAGCACACCRCAACCRG TGACCRTTGCCACCRCCAG
uc001eto.1	TCAGAAATAGGAAGGTGGAAGGGCTTCATGGATAATCTCCTTT TCCTCCACTATCTTGTCTTCTACAG
uc001fpr.2	TGGGGAGGGTGTGAGAGAGGTGCCCAAAGGCCCTAAAAGGT GAGCCTCTCCTCTCTCCCCACAG
uc001fxk.4	GTAAGTCTGGGGAGATGGGGGGAGCTCTGCTGAGGGTGCACA AGGCCCTGGCTCTACACACATCCCTGTCTTACAG
uc001gx0.1	AGGTAGAAGGAGAGAAAAGGAGGCTAGGAGTCAGCTGCTCAC CRCATTTCTCCCCCTGCCAG

1.5 The miRBase Database

The miRBase (Release 22.1 / October 2018) database is a comprehensive database of published miRNA sequences and annotations. The database has 38,589 hairpin precursors and 48,860 mature microRNAs from 271 organisms such as humans, flies, mice, worms, rats, and Arabidopsis. Moreover, the database has 1,917 hairpin precursors and 2,654 mature miRNA sequences in the human category [5].

It is nearly estimated that in mammals, almost always one-third of all protein-coding genes' activities are controlled by miRNAs[6]. Each record in the database belonging to a predicted hairpin portion (as a stem-loop sequence) of a miRNA transcript, with location information and validated sequence with a high probability of the mature miRNA[6].

It is shown in Figure 1.3 that a stem-loop sequence of a human miRNA (*has-let-7a-1*) and its two different subsequences (mature sequences) *has-let-7a-5p* and *has-let-7a-3p* [57].



- a)** Shows stem-loop sequence of *hsa-let-7a-1*, **b)** Shows mature sequence of *hsa-let-7a-5p*
c) Shows mature sequence of *hsa-let-7a-3p*

Figure 1.3 A miRNA sequence example, *hsa-let-7a-1*, from miRBase.

In the miRBase database, each data entry or in other words small RNA sequences has a score or confidence level for annotation. In that context, the sequences have to be satisfied some criteria to have a high confidence level score. To be in the annotation list with a high confidence level [58]:

- Minimum 10 reads must match without mismatch with each of the two possible mature miRNAs which are based on the hairpin precursor.

- The most frequent reads from each sequence of the precursor should match the mature miRNA duplex with 0-4 nt overhangs at the 3' ends.
- Minimum 50% of the readings mapped to each arm of the precursor of hairpin must have the same 5' end.
- The folding free energy of the proposed hairpin structure should be <-0.2 kcal/mol/nt.
- Minimum 60% of bases in mature sequences must match the predicted hairpin characteristics.

Those criteria show that the annotation of the sequences has a variety of restricted procedures even though they have a probability in the scoring of the sequence readings with a confidence level. We should take into account that probabilities when developing or constructing a deep-learning-based method for predicting the biological sequences especially miRNAs sequences derived from the miRBase database. Since having very few possibilities can lead to bias during the training or mistraining of the model.

CHAPTER 2

PRE-MIRNA CLASSIFICATION

Pre-miRNA identification and classification are one of the most important problems in computational biology in terms of small RNA structures. In previous studies, state-of-the-art machine-learning-based methods are applied to reveal superior results for this problem. Those earlier computational methods are widely based on manual feature extraction [59–66]. Therefore, the construction of a new model requires detailed and comprehensive knowledge about the research domain. Random forest (RF), decision trees (DT), and support vector machines (SVM) are the first examples of these classical machine-learning methods [59]–[66].

Neural networks (NNs) are preferred for figure out hidden patterns especially non-linear patterns. This is the strongest feature of NNs with respect to logistic or linear regression models. In Figure 2.1, an example of a simple NN has been placed with an input layer, a hidden layer, and an output layer. Three inputs (x_1 , x_2 , and x_3) feeds the network with five parameters in the hidden layer. And, the model gives a prediction or classification result in the output layer [67–69].

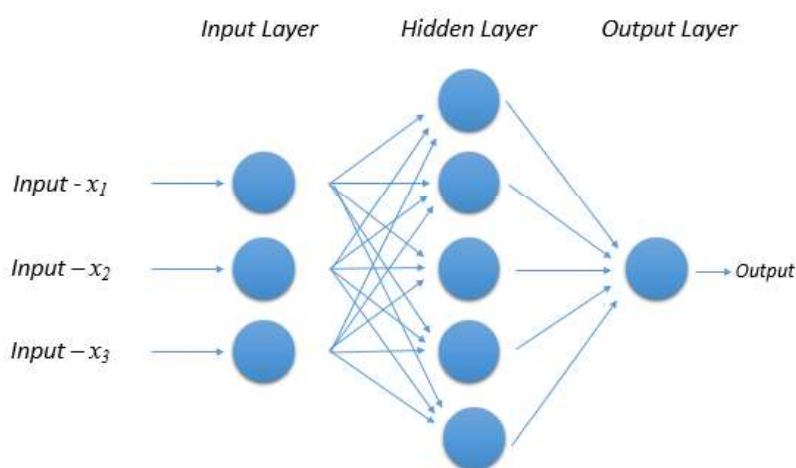


Figure 2.1 An example of a simple Neural Network.

Nowadays, Deep Learning (DL) methods are also has become to emerging and prominent study field in computational biology to achieve better performance in accordance with other machine learning methods[60,61,74,75,63–66,70–73].

DL provides multi-layered models for learning from training datasets by processing in their original form, which is not possible in conventional ML methods. These DL methods have significantly improved the performance of many data processing domains without any requirements of detailed domain knowledge for feature extraction. Since they extract the features automatically. Recognition of speech, detection of objects, and identification of face are common examples of DL-based models [67–69].

In Figure 2.2, an example of a simple Deep Neural Network (DNN) has been placed with an input layer, more than one hidden layer, and an output layer. Three inputs (x_1 , x_2 , and x_3) feeds the network with five parameters in each hidden layer. And, the model gives a prediction or classification result in the output layer. According to the type of data, the deeper networks could result in better performance [67–69].

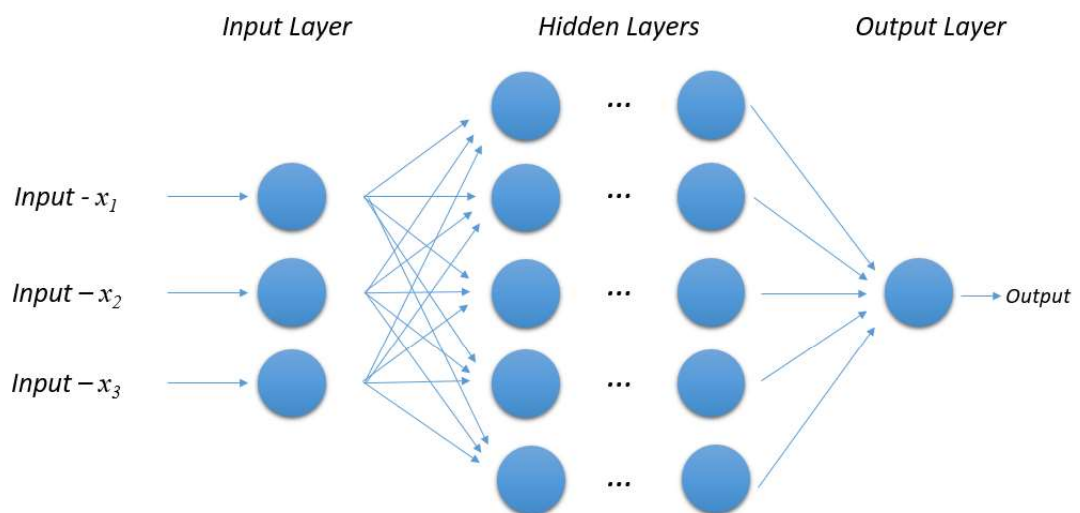


Figure 2.2 An example of a Deep Neural Network.

To give examples of application in the field of computational biology, the convolutional neural networks (CNN), a type of DL, have successfully applied for the pre-miRNA classification [32,63].

Zheng et. al. [31] applied the CNN method to extract features from the biological sequences of miRNAs automatically. Compared to other existing methods, CNN-based methods outperform to identify the miRNAs and extract features automatically from the raw input data without comprehensive domain knowledge[76–79].

In an article [80], it is determined that the spatial characteristics of these small structures have the same importance as others. Thus, they draw attention in the paper to the long-term dependencies. In addition, they investigated that the LSTM network is also feasible for using to the identification of pre-miRNAs.

When we consider the literature, we understand that there are various implementations of using cascaded or combined CNN and LSTM networks. Those networks are utilized from both spatial and structural characteristics of the input data.

2.1 Convolutional Neural Networks

The CNNs are a subset of deep learning methods. They perform higher performance in terms of comparison criteria according to the manual feature extraction methods. Thus they are used in a wide range of field i.e. classifying images[81,82], detecting objects[83,84], recognizing speech [85], computer vision[86], and analyzing videos [87]. In addition, it is also a commonly preferred method for bioinformatics and computational biology [88,89].

Unlike classical neural networks, CNNs contain multiple layers. So, the networks become deeper and deeper by adding layers. Moreover, a CNN includes weight, pooling, bias, and outputs through a non-linear activation function. A typical CNN architecture fundamentally consists of the input layer, convolution layers, pooling layers, fully connected layers, and output layer [79].

Figure 2.3 shows that the basic architecture of a typical convolutional neural network with the input layer, convolution layer, pooling layer, and fully connected layer.

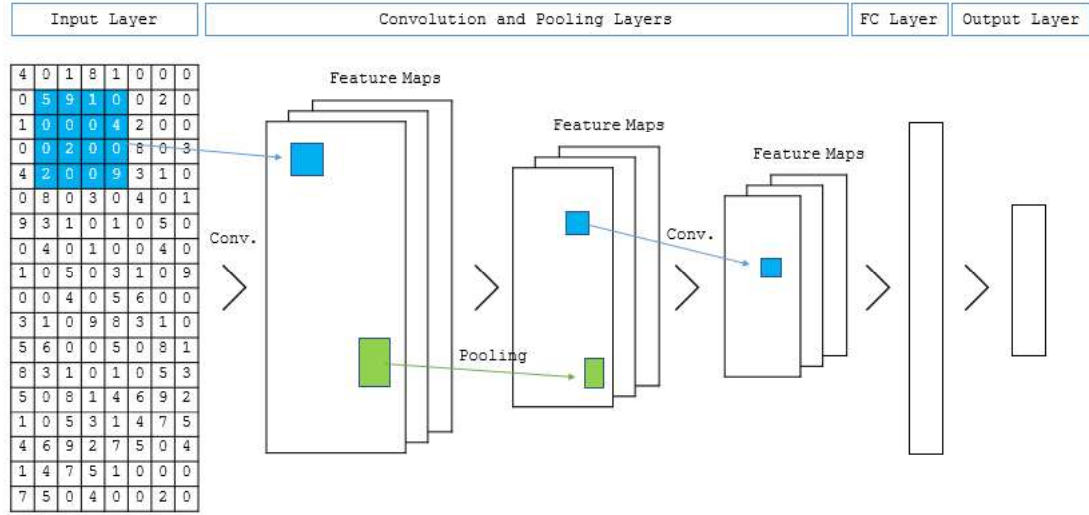


Figure 2.3 An example of a convolutional neural network.

The formulation of the convolutional layer is as follows:

$$F(i, j) = (I * K)(i, j) = \sum \sum I(i + m, j + n) K(m, n) \quad (2.1)$$

where I for the input matrix, K for a two-dimensional filter of size $m \times n$, and F for the output of a two-dimensional feature map. In addition, the convolutional layer representation is with $I * K$.

2.2 Long Short-Term Memory Networks

The classic RNNs can control random long-term dependencies in the input sequences. However, using backpropagation when training an RNN, lead to the computational problem since the gradients, allows us to adjust all weights, can tend

to zero or infinity. This problem, called the vanishing gradient problem, is solved partially by using LSTM [90].

An LSTM network, proposed by Sepp Hochreiter and Jürgen Schmidhuber, is a special class of RNN that uses a memory mechanism that assists to run successfully and learn faster than conventional RNNs [91,92]. Thus, long and short-term dependencies can be captured by LSTM in sequence [90].

The first architecture of the LSTM has input and output gates, and cells[90]. In 1999, forget gate which provides the LSTM to reset its state included in LSTM architecture[93]. One year later, the output activation function was omitted [90] and peephole connections were added into the architecture [94].

LSTM networks find practical solutions for the vanishing and exploding gradient problem of RNNs[95]. Apart from the RNNs, a cell state is used in the LSTM network to save long-term state information including input, forget, and output gates. Thus, the network can remember previous data and connect it with the present ones. And it gives solutions to complex problems that are hard to find a solution by previous RNNs [90,92].

The formulation of the LSTM memory cell is as follows:

$$f_t = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + b_f) \quad (2.2)$$

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + b_i) \quad (2.3)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c) \quad (2.4)$$

$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + b_o) \quad (2.5)$$

$$h_t = o_t \otimes \tanh(c_t) \quad (2.6)$$

where σ refers to the Sigmoid function, \tanh is a function that set values between $-1/1$, f , i , h , c , o are the forget gate, input gate, hidden vector, cell vectors, and output gate, respectively, W_{xf} refers the input/forget gate matrix, and W_{hf} refers the

hidden-forget gate matrix. The index of t is the time step. \otimes indicates to the vector product. Additionally, initial values of $c_0 = 0$ and $h_0 = 0$.

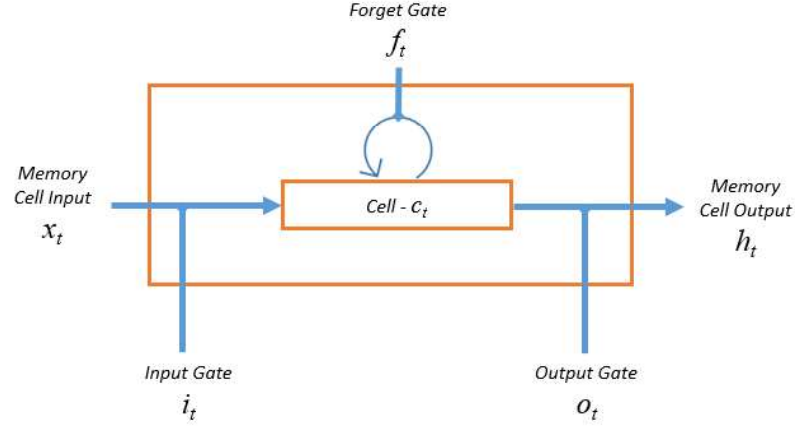


Figure 2.4 Brief architecture of an LSTM cell.

Figure 2.4 shows a brief architecture of an LSTM cell with input, forget, and output gates.

In computational biology, LSTM networks are also frequently used. For instance, Park et al. [80] illustrate that the spatial information of small RNA structures is as critical as the structures that construct miRNAs. Thus, they built their model with solely long-term dependencies and trained an LSTM-based framework for the identification of pre-miRNAs.

2.3 The Hybrid CNN and LSTM Networks

A cascaded or hybrid CNN and LSTM network is the combination of CNN layers that extract the feature from input data and LSTMs layers to provide sequence prediction [96]. CNN and LSTM are generally used for activity recognition, image labeling, and video labeling. Their common features are that they are developed for the application of visual time series prediction problems and generating textual annotations from image sequences [96,97].

Figure 2.5 shows that the basic architecture of the hybrid CNN and LSTM network with the input layer, visual feature extraction, sequence learning, and output layer, respectively [96].

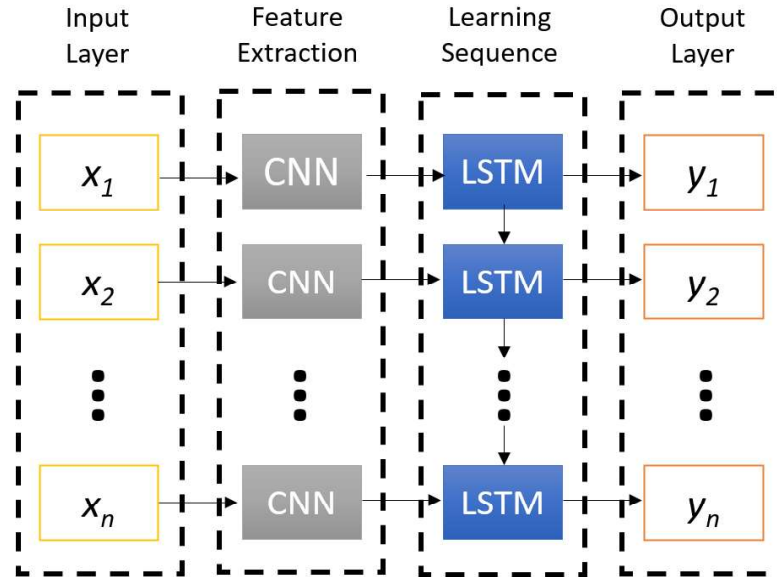


Figure 2.5 A basic architecture of hybrid CNN and LSTM network.

Many studies reveal that CNN-LSTM networks can provide better performance or solutions other than using the structural or spatial information of each data input separately. To explain with examples, Quang et al. [98] introduced DanQ which is a combination of CNN-LSTM frameworks for predicting the function of DNA sequences. In this model, the convolution layers capture the patterns of the sequences, and the recurrent layers capture long-term dependencies. Similarly, Pan et al. developed iDeepS [99], to predict the binding sequence and structure patterns from RNA-protein sequences. Their model extract features by using CNN layers and shows probably long-term dependencies by using bidirectional LSTM layers. Another example of using a hybrid CNN-LSTM network is Yan et al. research [100]. They applied this network for the four-class pathological image classification of

breast cancer disease. Their proposed model outperforms the state-of-the-art method with an accuracy of 90.5% which is higher than the previous models.

These examples of research reveal that using both spatial and sequential information provides higher performance compared to using only one of them, especially in computational biology.

CHAPTER 3

MATERIALS AND METHODS

In this study, we proposed a hybrid deep-learning-based method for human pre-miRNAs classification to predict pre-miRNAs by integrating two separate methods: CNNs and LSTM, respectively. First of all, we described the problem of pre-miRNAs classification. Later, we introduced the datasets, which are used to train and test our proposed method.

The datasets include both human mirtrons' and canonical miRNAs' sequences. For consistency, the same sequence data are used as the previous methods[32,101].

In our model, the initial step was started with the preprocessing of data for proper representation. Then the CNNs were deployed to the processed data for extracting features automatically. Thus, the model was converted to be ready for feeding the LSTM network. Next, the LSTM layer was used to perform temporal modeling following the CNN layer that convolves the input data. Then, we gave more details about CNN, LSTM, and CNN+LSTM networks. Finally, we described our proposed model and how to develop it in detail.

The model was implemented in the python programming language using Keras library (2.4.3) <https://github.com/keras-team/keras>, with the backend of TensorFlow (2.4.0).

3.1 Problem Statement

Many existing pre-miRNA classification methods rely on manual feature extraction. These methods focus on either spatial structure or sequential structure of pre-miRNAs. To overcome the limitations of previous models, we propose a nucleotide-level deep learning method based on a hybrid CNN and LSTM network together for pre-miRNAs classification. When we consider the structure and sequence of pre-

miRNAs, the problem is a binary sequence classification problem consisting of mirtrons and canonical miRNAs.

In literature, several models including machine-learning methods have been developed to find a solution for the problematic classification. On the other hand, they have approximately 90% of accuracy. In this study, in the pre-miRNA classification, our goal was to show how to accurately predict classes with a hybrid CNN-LSTM network.

3.2 Training and Test Datasets

The dataset consists of mirtrons and canonical miRNAs' data. We combine two different datasets in our preprocessing data phase with 707 canonical miRNAs and 417 mirtrons. The first dataset (Dataset A) consisted of mirtrons and canonical miRNAs derived from miRBase (Release 21, 06/14)[53]. And the second dataset (Dataset B) is derived from the study of Wen et al. which includes 201 mirtrons putative mirtrons data[53]. In total, we used 1,124 data in our model. This dataset was also the same for consistency used by Zheng et al. and Rorbach et al. in their research[32,101].

3.3 Resampling

The resampling of the data is an important and initial step in developing a machine learning model. In the literature, it is stated that it would be appropriate to divide the training set and test set in different ratios such as 80%-20% or 70%-30% depending on the size of the data. In addition to the size of these subsets, split the dataset into training and test set properly affects the performance of a model directly. Therefore, several approaches are developed to train and validate a model.

The most common types of resampling approaches[102] are Hold-out Random Subsampling, k -folds Random Subsampling, k -folds Cross-Validation, Leave-one-

out Cross-Validation (LOOCV), Leave-p-out Cross-Validation (LpOCV), and Stratified k-folds Cross-Validation.

3.3.1 Hold-out random subsampling

One of the simplest data resampling approaches among the others is the hold-out random subsampling method. It randomly samples some entries from the data set for the test set and the remaining entries are used for the training set.

In hold-out random subsampling, the training set generally contains approximately 70%, 75%, 80%, or 90% of entries, and the test set approximately 30%, 25%, 20%, or 10% of existing entries. If the data set is large enough then the observed test error can be a reliable estimate of the true error of the model for new, unseen entries[102].

3.3.2 k -folds random subsampling

The k -folds random subsampling uses the same algorithm to split the dataset to TRAIN and test set as hold-out random subsampling method. It repeats the splitting iteration k times. In an iteration, the training set is used for learning and the corresponding test set is used for evaluation. In total, k models are going to be trained and tested. The average performance of all k test sets is determined the results. While there are no common entries for any training and test set pairs, any two training sets or two test sets can overlap [102].

3.3.3 k -folds cross-validation

The repeated random resampling methods have an overlapping problem in different training and test tests. In contrast to these methods, Cross-validation provides a solution to this problem and prevents overlapping different subsets for all iterations [102].

The k -folds cross-validation splits dataset to train and test set about the same size in each iteration. It repeats the splitting iteration k times and constructs disjoint subsets. In an iteration, the training set is used for learning and the corresponding test set is used for evaluation. In total, k models are going to be trained and tested. The average performance of all k test sets is determined by the results of the cross-validation [102].

3.3.4 Leave-one-out cross-validation (LOOCV)

Leave-one-out cross-validation (LOOCV) is a special form of the k -folds CV. In this method, k is equal to the number of entries. For instance, if the dataset has 50 entries ($n=50$), the k value is equal to 50 then the method starts from the first entry and leave out it for testing and the rest of the dataset is used for training. In total, n models are going to be trained and tested. Compared to the k -folds CV method, LOOCV has a high cost of computation especially in large datasets [102].

3.3.5 Leave-p-out cross-validation (LpOCV)

Leave-p-out cross-validation (LpOCV) is a special form of the LOOCV. In this method, p is equal to the number of the selected entries in each iteration. For instance, if the dataset has 50 entries, the p value is equal to 3 then the method leaves out 3 entries for testing and the rest of the dataset used for training. In total, n models are going to be trained and tested with $p=3$. Similar to the LOOCV method, LpOCV has a high cost of computation especially in large datasets [102].

3.3.6 Stratified k-folds cross-validation

Stratified k -folds cross-validation (CV) is a resampling procedure that splits the dataset into folds according to the output categories and ensures that each fold has the same proportion.

Figure 3.1 shows the illustration of stratified k -folds cross-validation with $k=5$. In this example, Class A has 30%, and Class B has 70% of all entries.

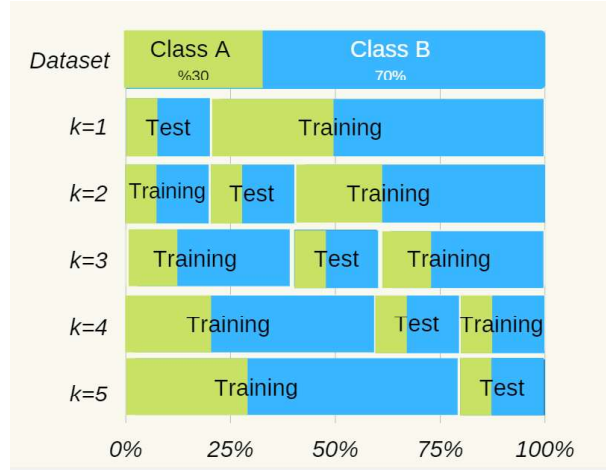


Figure 3.1 Visualization of stratified k -folds cross-validation with $k=5$

3.3.7 Imbalanced datasets and the model cross-validation procedure

In real-life applications, datasets are generally not equal-size sub-classes. Therefore, the selection of the right resampling method has a great deal amount of importance. For the construction of a reliable deep learning model, the first and foremost step is the analysis of the dataset whether it is balanced or not. The stratified 5-folds CV is useful for imbalanced datasets[103]. Hence, in this study, we used a stratified 5-folds CV for training and evaluating our model[103]. At each iteration, it divided the data into training and test sets with an 80-20% split. In the next iteration, it used the other percentile as the training and test set.

Table 3.1 shows the distribution of the training and test datasets at each iteration in stratified 5-folds CV.

Table 3.1 Distribution of the training and test samples for each fold.

	Training Dataset		Test Dataset		Total	
	<i>Sequence #</i>	<i>%</i>	<i>Sequence #</i>	<i>%</i>	<i>Sequence #</i>	<i>%</i>
Mirtron	334	80%	83	20%	417	37%
Canonical miRNA	566	80%	141	20%	707	63%
Total	900	80%	224	20%	1,124	100%

3.4 Preprocessing of Data

We prepared each sequence of pre-miRNAs -according to maximum length- with the same length (164) by padding process. It is represented in the database by adenine A, thymine T, guanine G, cytosine C, and uracil U. The word “N” is used for keeping the sequences in the same length. Like Zheng et al. [32]., “one-hot” encoding is used to encode each base of the pre-miRNAs ’ sequences (Table 3.2). Then, we converted each sequence into a vector with a dimension of (164, 4) by the vectorization process.

Table 3.2 “One-hot” encoding for base sequence.

Base Name	Encoded Base			
A	1	0	0	0
U/T	0	1	0	0
G	0	0	1	0
C	0	0	0	1
N	0	0	0	0

Similarly, we encoded each output [0,1] for false and [1,0] for true into a vector with a dimension of (164, 4) by the vectorization process. The following table shows the encoded output categories.

Table 3.3 “One-hot” encoding for outputs.

Output	Encoded	
True (1)	1	0
False (0)	0	1

3.5 The Model Architecture

The architecture of our model includes an input layer, three CNN layers wrapped by a time-distributed layer, a flatten layer, an LSTM layer, a dense layer, a dropout layer, and an output layer, respectively.

Figure 3.2 shows the illustration of the detailed architecture with visualization of our model. Before constructing the model, we ensured that each data has been transformed into an appropriate form to use. In this case, we used the padding process to guarantee the length (which is 164) of each miRNA sequence similar by adding “N” for each blank. The next step, named the vectorization step, is transforming the padded sequences to like a $m \times n$ matrix by using one-hot encoding.

When all data are padded and vectorized, the network became ready for the feature extraction process. In this stage, three convolution layers were used to extract features automatically from input sequences.

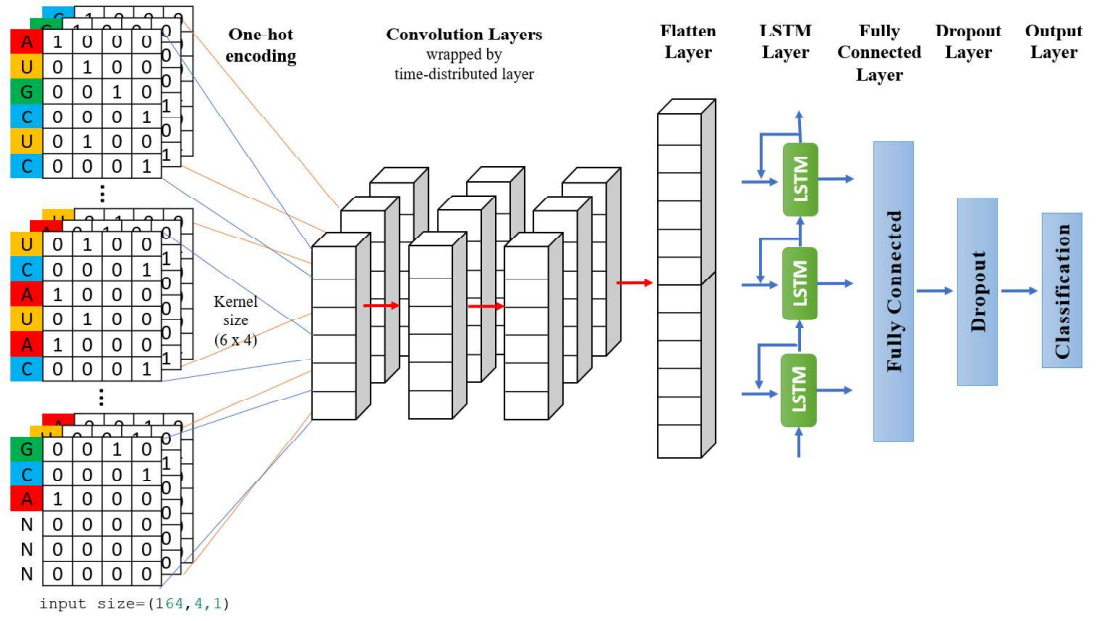


Figure 3.2 A brief architecture with visualization of the model

In the convolutional layers, the hyper-parameter of filter size was selected 128 since we received better results as it is in the literature. Furthermore, we employed (6x4) kernel size which means that kernel's height is 6 and the kernel's width is 4 for convolution operation because these dimensions are proved[32] with higher performance.

For concatenation of all extracted features, a fully connected layer is used. In this convolutional stage, we wrapped the convolution layer in a time-distributed layer. We employed dropout (0.5) on the fully connected layer for regularization following a flatten layer. Then, one LSTM layer is designed with 100 units. Finally, due to the size of the number of output categories, the *softmax* layer is used for the output.

Detailed convolutional layer parameters such as strides, padding type, activation function deployed, and their values are demonstrated in Table 3.4.

Table 3.4 Detailed parameters for each convolutional.

Parameter Name	Value
filters	128
kernel_size	6 x 4
strides	(1, 1)
padding	valid
data_format	None
dilation_rate	(1, 1)
groups	1
activation	relu
use_bias	True
kernel_initializer	glorot_uniform
bias_initializer	zeros
kernel_regularizer	None
bias_regularizer	None
activity_regularizer	None
kernel_constraint	None
bias_constraint	None

Model is optimized for 30 epochs, 6 for batch size, and 0.1 for validation split by training. Keras' default values are used for the weights, learning rate, and bias. We used cross-entropy for optimization due to the nature of the one-hot encoding classification.

Figure 3.3 shows the convolution process of the one-hot encoded and vectorized input data with a 6 x 4 filter size.

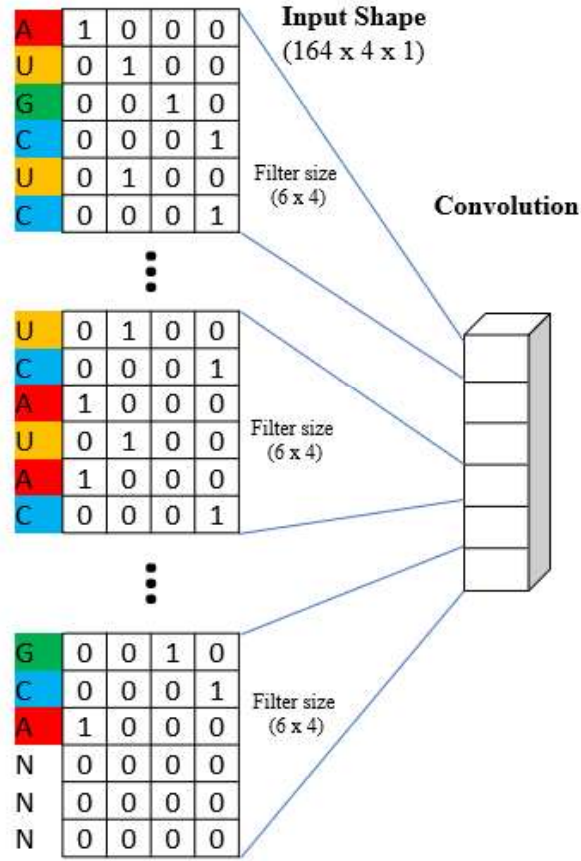


Figure 3.3 Illustration of the convolution with 6 x 4 filter size.

Table 3.5 gives the designed model details including an input layer, four time-distributed layers, an LSTM layer, a dense layer, a dropout layer, and an output layer with the number of the parameters and output shape of each layer. In that context, it is clear that the output shape is satisfying the binary classification proposal of the model with two different classes.

More details about the architecture of our proposed model with including all parameters are given in Appendix A.

Table 3.5 The designed model with parameters

Layer name	Output Shape	Param #
Input Layer	(None, 164, 4, 1)	
Time Distributed Layer 1	(None, None, 82, 2, 128)	3200
Time Distributed Layer 2	(None, None, 42, 1, 128)	393344
Time Distributed Layer 3	(None, None, 21, 1, 128)	393344
Time Distributed Layer 4	(None, None, 2688)	0
LSTM Layer	(None, 100)	1115600
Dense Layer	(None, 256)	25856
Dropout	(None, 256)	0
Output Layer	(None, 2)	514

3.6 Activation Functions

The purpose of the activation function is to introduce nonlinearity which enables neural networks to tackle very complex nonlinear problems. Commonly used activation functions are sigmoid function, hyperbolic tangent function (Tanh), rectified linear unit (ReLU), and leaky ReLU (LReLU).

3.6.1 Sigmoid function

The following equation shows the sigmoid function.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (3.1)$$

The following figure (Fig. 3.4) shows the visualization of the sigmoid function.

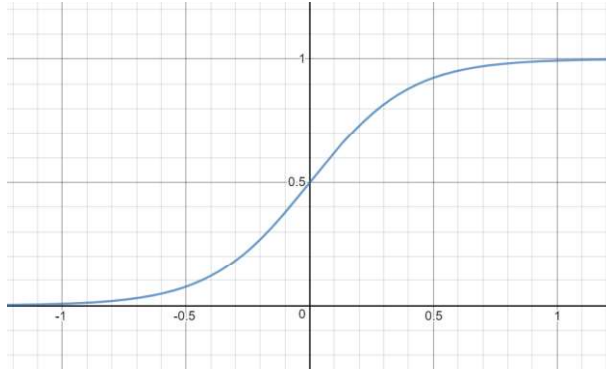


Figure 3.4 Illustration of Sigmoid function.

3.6.2 Hyperbolic tangent function (Tanh)

The following equation shows the hyperbolic tangent function (Tanh).

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (3.2)$$

The following figure (Fig. 3.5) shows the visualization of the hyperbolic tangent function.

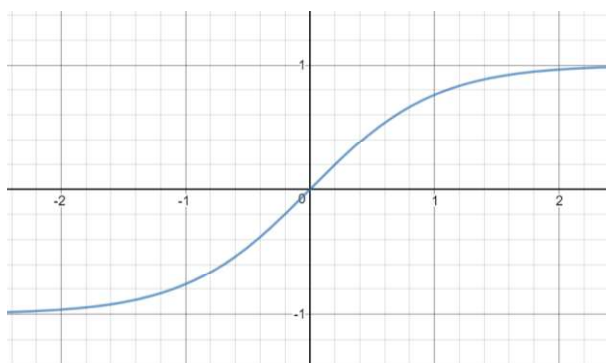


Figure 3.5 Illustration of Hyperbolic Tangent function (Tanh)

3.6.3 Rectified linear unit function (ReLU)

The following equation shows the rectified linear unit function (ReLU).

$$ReLU(z) = \begin{cases} z, & z > 0 \\ 0, & otherwise \end{cases} \quad (3.3)$$

The following figure (Fig. 3.6) shows the visualization of rectified linear unit function.

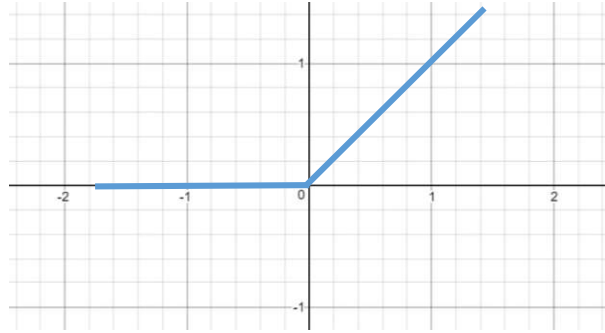


Figure 3.6 Illustration of Rectified Linear Unit function (ReLU)

3.6.4 Leaky rectified linear unit function (Leaky ReLU)

The following equation shows the leaky rectified linear unit function (Leaky ReLU).

$$LReLU(z) = \begin{cases} z, & z > 0 \\ \alpha z, & otherwise \end{cases} \quad (3.4)$$

The following figure (Fig. 3.7) shows the visualization of leaky rectified linear unit function where $\alpha=0.2$.

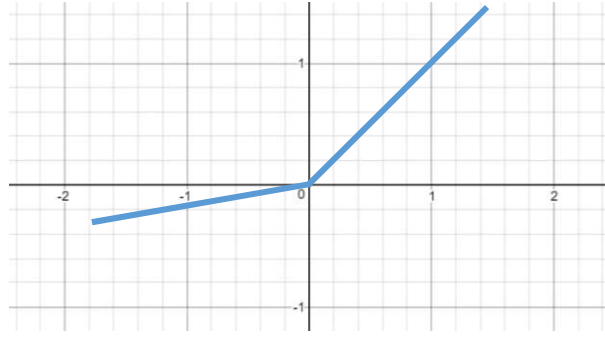


Figure 3.7 Illustration of Leaky Rectified Linear Unit function (LReLU)

3.7 Optimization Algorithms

Our model's primary goal is to decrease the mean loss function by determining a loss function. Thus, the right weights and biases for the network would be found during the training process by using backpropagation to update gradients on the parameters.

In this study, cross-entropy is defined as the loss function to minimize the mean loss function between the predicted distribution over actual labels[104].

The formulation of the cross-entropy is as following:

$$\text{Cross Entropy} = - \sum_{i=1}^n y_i \log s_i \quad (3.5)$$

where n for the number of labels, s_i for the label i 's predicted probability, y_i for the label i 's actual probability.

The gradient descent algorithm is one of the most common algorithms that are used to optimize the NNs loss function for an ML problem. It minimizes the objective function parameterized by the model. It updates parameters in a contrary way to the gradient of the objective function concerning the parameters. In another word, the gradient descent algorithm calculates the reverse direction of the gradient to figure out the local minimum area. Therefore, the algorithm moves in opposite directions of the gradients in each iteration. Finally, by the learning rate, the gradient descent targets to reach a local minimum step by step [105,106].

There are three types of gradient descent: Batch (Vanilla) Gradient Descent, Stochastic Gradient Descent (SGD), and Mini-batch Gradient Descent. The main differences between those algorithms are the amount of data that are used for calculating the gradient [93].

The Batch (Vanilla) Gradient Descent calculates the cost function's gradient. As indicated in the following formula, it computes the gradient for the whole dataset where η refers to the learning rate and θ refers to the parameter that we want to optimize [93].

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta) \quad (3.6)$$

In the last decade, the rapid growth of data size has become problematic for computational sciences in terms of time complexity and cost. In these circumstances, the performance of the statistical ML algorithms directly deals with time complexity rather than the size of the sample. Therefore, we use Stochastic Gradient Descent (SGD) when the training time is important for the developed model [105,106].

Stochastic gradient descent (SGD) estimates the gradient on a single arbitrarily selected example instead of calculating the gradient for each iteration. The formula is as follows:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i)}; y^{(i)}) \quad (3.7)$$

where it relies on the arbitrarily selected examples at each iteration ($t= 1, 2, 3, \dots$).

Although updating the model frequently learns faster and gives better performance, it takes a longer time for training large datasets. Thus, the time complexity gets higher than other variants of the gradient descent in SGD.

Mini-batch gradient descent is another type of gradient descent algorithm. In that algorithm, the training dataset splits into tiny batches. Those batches are performed to calculate the error of the model. In addition, it updates the coefficients of the

model for every mini batch of n training samples. The following formula shows the *calculation of θ* in the mini-batch gradient descent algorithm [93]:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \quad (3.8)$$

The well-known DL libraries include implementations of different algorithms to optimize gradient descent. The Keras library contains commonly used gradient descent optimizers such as Adagrad, Adadelata, RMSprop, and Adam algorithms, etc. [105,106].

3.7.1 Adagrad

Adagrad is a kind of gradient descent algorithm that optimizes the model by adapting the learning rate according to the frequency of the parameters. It adapts the learning rate update larger for rare parameters. On the contrary, it adapts the learning rate update smaller for rare parameters. Thus, if the dataset contains sparse data, it could be better to use this algorithm.

Even though most algorithms prefer a fixed value for the learning rate, *Adagrad* is tuning it automatically. This is the most powerful feature of this algorithm. The main point that *Adagrad* is criticized for is that the square gradients in the denominator grow over time, and therefore the learning rate decreases until it approaches infinity. Thus, after a while, the algorithm stops learning, and it becomes unable to learn new knowledge [105–107].

The parameter update function is as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\varepsilon I + \text{diag}(G_t)}} \cdot g_t \quad (3.9)$$

where θ is for the to be updated parameter, η is for the primary learning rate, ε is a small number that guarantees not to be the division of zero, I is for the identity matrix, g_t is the gradient estimate function in time step t .

$$g_t = \frac{1}{n} \sum_{i=1}^n \nabla_{\theta} L(x^{(i)}, y^{(i)}, \theta_t) \quad (3.10)$$

Where g_t refers to the sum of the outer product of the gradients until time step t .

$$G_t = \sum_{T=1}^t g_T, g_T^T \quad (3.11)$$

3.7.2 Adadelata

Adadelata is an extension of *Adagrad* that gives a solution to the growing square gradients problem in the denominator over time. Therefore the learning rate does not decrease dramatically and the algorithm does not stop learning, and it continues to learn new knowledge [105,106,108].

The method uses just first-order information. It adapts dynamically to the learning rate. Thus it gives promising results to noisy gradient information and other hyper-parameters concerning the stochastic gradient descent algorithm[108].

3.7.3 RMSprop

Geoff Hinton proposed the *RMSprop* optimizer. It is an unpublished adaptive learning rate method [105,106,109]. They introduced the method in the 6th lecture of an online course on Coursera named “Neural Networks for Machine Learning”[109].

RMSprop is an adaptive learning rate algorithm like *Adagrad*. As it is determined in Section 3.7.1, *Adagrad* adds element-wise scaling of the gradient which means that it keeps a running summary of squared gradients. Next, it adapts the learning rate by dividing it by this sum[109].

3.7.4 Adam

Adaptive Moment Estimation (Adam) calculates the adaptive learning rate for each parameter. *Adam* keeps an exponentially decaying mean of past gradients same to momentum in addition to RMSprop and Adadelata [105,106,110].

The method was announced in 2015 by Kingma et al. They evaluated the method by conducting empirical research on a variety of popular ML models, including logistic regression, NNs, and CNNs. They proved that the Adam method is effective for solving DL problems regardless of whether the models are large or the datasets are large. It is a combination of *AdaGrad* and *RMSProp*. Thus, the method is able to use sparse gradients like *AdaGrad* and non-stationary objectives like *RMSProp*[110].

Adam optimizer has some advantages over other methods. For instance, the parameter updates of *Adam* are estimated by a running mean of the gradient's first and second moment while *RMSProp* uses momentum[110].

Alpha, *beta1*, *beta2*, and *epsilon* are the configuration parameters of the *Adam* optimizer. Alpha refers to the learning rate, beta1 and beta2 refer to the first and the second momentum respectively, and the final parameter, epsilon, refers to the very small number that prevents zero division problems [110].

3.8 Method Evaluation Criteria

In this study, the evaluation of our model was measured on the test dataset at each fold. We calculated five different measure criteria for performance comparison in the analysis: accuracy (Acc.), sensitivity (Sen.), specificity (Spe.), F1 score, and Matthews Correlation Coefficient (MCC).

The evaluation criteria are calculated for performance evaluation with the number of true-positive (TP), false positive (FP), true-negative (TN), and false-negative (FN) by the following equations:

Accuracy indicates the overall correctness of prediction:

$$Acc. = \frac{TP + TN}{TP + FN + TN + FP} \quad (3.12)$$

Sensitivity (also known as Recall) is a true positive rate. It indicates the ratio of correctly classified actual positives:

$$Spe. = \frac{TP}{TP + FN} \quad (3.13)$$

Specificity, true negative rate, indicates the ratio of correctly classified actual negatives:

$$Sen. = \frac{TN}{TN + FP} \quad (3.14)$$

Precision indicates the positive predictive value:

$$Pre. = \frac{TP}{TP + FP} \quad (3.15)$$

F1-Score is a combination of the precision and recall of the model by harmonic mean:

$$F1\ Score = \frac{2TP}{2TP + FP + FN} \quad (3.16)$$

Matthews Correlation Coefficient (MCC) is a binary classifier. It measures the quality of a prediction:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (3.17)$$

where P shows the number of positives, and N shows the number of negatives.

		Predicted Label	
		P	N
Actual Label	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Figure 3.8 Visualization of the confusion matrix.

The basic structure of the confusion matrix is visualized in Figure 3.8.

In addition to the performance comparison criteria, we calculated also the mean, median, and standard deviation of the entire results of each iteration. In that context, a 95% Confidence Interval (CI) is selected.

The *mean* is the average number of the data:

$$\bar{x} = \frac{\sum_{i=1}^n x}{N} \quad (3.18)$$

where \sum represents the summation, x represents observations, k represents iteration number, and N represents the total observations' number.

The *median* is the number that is in the center/middle of a data set:

$$median = \left(\frac{N+1}{2}\right)^{th} \text{ observation} \quad (3.19)$$

where N represents the total observations' number and an odd number.

$$median = \frac{\left(\frac{N}{2}\right)^{th} \text{ observation} + \left(\frac{N+1}{2}\right)^{th} \text{ observation}}{2} \quad (3.20)$$

where N represents the total observations' number and an even number.

Standard Deviation measures the dispersion of data and indicates the variation in the values of the data set. It becomes smaller when the values of the data set are closer. And it becomes larger when the values of the data set are spread out. Additionally, it is generally represented by the Greek small letter sigma.

The following equation shows the calculation of Standard Deviation:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (3.21)$$

where n represents the total observations' number.

Confidence Interval (CI) shows the probability that a parameter is going to fall between a couple of values near the mean.

$$CI = \bar{x} \pm z_c * \frac{\sigma}{\sqrt{n}} \quad (3.22)$$

where n represents the total observations' number, \bar{x} mean of the sample, z_c value for confidence level, and σ is for standard deviation.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 The model summary

In our proposed model, we performed a hybrid CNN and LSTM network for human pre-miRNA classification since the automatic feature extraction without detailed domain experts from pre-miRNAs sequences. We started with preprocessing the datasets by converting the raw entries to vectors by using “one-hot” encoding. Next, we padded and vectorized the whole data. Then, we deploy the CNN layers that were wrapped by the time-distributed layer. And *relu* activation function is used in this convolution process. For concatenation of all extracted features, we employed a flatten layer for passing to the LSTM layer. Then, a 100-unit LSTM layer is designed. A dropout layer (0.5) followed that layer. Finally, following the fully connected layer, used one output layer with *softmax* activation function for binary classification of pre-miRNAs.

Figure 4.1 determines the model summary including layers, input, and output shape with the trainable and non-trainable parameters. It indicates that all parameters are trained by the network.

Layer (type)	Output Shape	Param #
=====		
time_distributed_216 (TimeDi	(None, None, 82, 2, 128)	3200
time_distributed_217 (TimeDi	(None, None, 41, 1, 128)	393344
time_distributed_218 (TimeDi	(None, None, 21, 1, 128)	393344
time_distributed_219 (TimeDi	(None, None, 2688)	0
lstm_54 (LSTM)	(None, 100)	1115600
dense_108 (Dense)	(None, 256)	25856
dropout_54 (Dropout)	(None, 256)	0
dense_109 (Dense)	(None, 2)	514
=====		
Total params: 1,931,858		
Trainable params: 1,931,858		
Non-trainable params: 0		

Figure 4.1 The model including layers, input, and output shape.

4.2 Performance of the proposed network

We applied a stratified 5-folds CV for training and testing our model. At each fold, we divided the data into training and test sets with an 80-20% split. In the next fold, we used the other percentile as the training and test set.

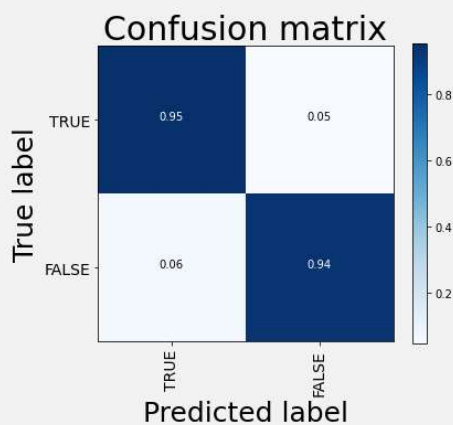
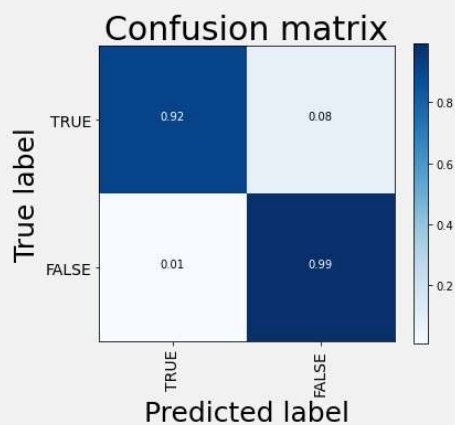
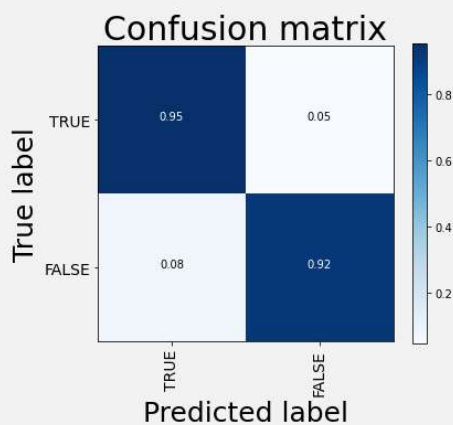
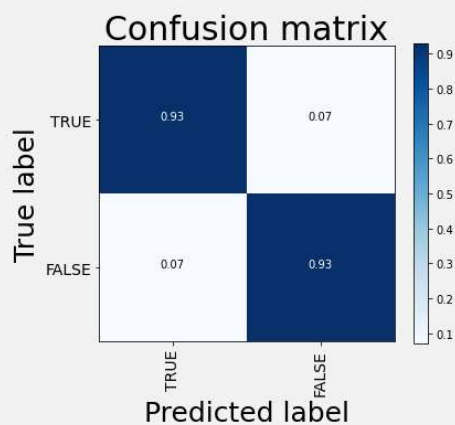
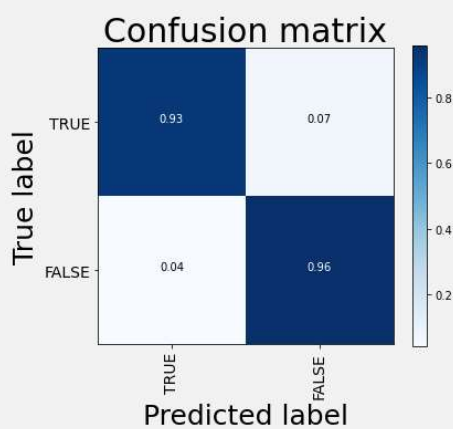
Table 4.1 shows the performance results of our proposed network at each iteration. Additionally, we calculated the mean, median, standard deviation, and confidence interval (CI) for each metric.

Table 4.1 Performance of the proposed CNN-LSTM network for each fold.

Fold #	Accuracy	Sensitivity	Specificity	F1 Score	MCC
1	0.942	0.952	0.937	0.924	0.878
2	0.964	0.916	0.993	0.950	0.924
3	0.933	0.952	0.922	0.914	0.862
4	0.929	0.929	0.929	0.907	0.850
5	0.946	0.928	0.957	0.928	0.885
Mean	0.943	0.935	0.948	0.925	0.880
Median	0.942	0.929	0.937	0.924	0.878
Std. Dev.	0.014	0.016	0.029	0.016	0.028
95% CI	0.931-0.955	0.921-0.949	0.923-0.973	0.910-0.939	0.855-0.905

Figure 4.2 illustrates the performance results of the confusion matrix of our proposed network at each iteration. In this context, true labels are used for the mirtron and false labels are used for the canonical miRNA.

Moreover, detailed architecture is depicted in Appendix A, and comprehensive results, parameters, and other outputs are also shared in Appendix B.

Fold Number: 1Fold Number: 2Fold Number: 3Fold Number: 4Fold Number: 5**Figure 4.2** The confusion matrix of each fold.

4.3 Performance comparison

Table 4.2 shows the performance comparison of the average values of the proposed method with the previous methods. The prediction resulted in 0.943 (%95 CI $\pm 0,014$) accuracy, 0.935 (%95 CI $\pm 0,016$) sensitivity, 0.948 (%95 CI $\pm 0,029$) specificity, 0.925 (%95 CI $\pm 0,016$) F1 Score and 0.880 (%95 CI $\pm 0,028$) MCC.

When compared to the closest results, our network revealed the best results for Acc., F1 Score, and MCC. These were 2.51%, 1.00%, and 2.43% higher than the closest result, respectively.

Table 4.2 Performance comparison of pre-miRNA classification.

Method Name	Acc.	Sen.	Spe.	F1 Score	MCC
Proposed Method*	0.943	0.935	0.948	0.925	0.880
CNN filter6 128 [32]	0.920	0.871	0.970	0.916	0.845
CNN concat filters [32]	0.910	0.846	0.975	0.904	0.827
Support Vector Machines[101]	**	0.926	0.945	0.901	0.859
Random Forest [101]	**	0.870	0.957	0.883	0.836
Linear Discriminant Analysis [101]	**	0.935	0.919	0.881	0.830
Logistic Regression [101]	**	0.875	0.941	0.867	0.816
Decision Tree [101]	**	0.861	0.943	0.863	0.808
Naive Bayes [101]	**	0.875	0.894	0.824	0.746

* Mean of the stratified 5-folds CV results.

** Data not available.

The mean of sensitivity had the highest value like Linear Discriminant Analysis and ranked first. These ratios indicate that the hybrid CNN and LSTM networks can be employed to achieve better performance for pre-miRNA classification compared with previous methods. Even though the results show that our model has a higher ratio according to accuracy, sensitivity, F1 score, and MCC; we have a lower ratio (94.8%) of correctly classified true negatives.

In unbalanced or skewed datasets, the number of examples of the minority class might not be sufficient for learning. That's way, the minor class is more often misclassified than the major class [111,112].

In this study, the number of positive and negative data in our training and test dataset is equally representative of the whole dataset. Therefore, we solve the misclassification problem at the data preparation level.

4.4 Hyper-parameters

Hyper-parameters are important parts of designing the model architecture in deep neural networks. For instance; the number of the hidden units, order of the layers, batch size, optimizer selection, and learning rate, etc. are commonly determined by the researchers. Those parameters directly affect the performance of the models.

In this dissertation, we also benefit from the other researcher's experiences in addition to ours. For example, Zheng et al. [32] investigated that kernel size (6x4) and unit number (128) of the CNNs produced better results according to other sizes and numbers in the pre-miRNA classification.

When we tested hyper-parameters like Zheng et al [32] we get the same performance results as they did. In our future work, we will take into account the experiences we have gained in these studies and we will do more extensive hyper-parameter optimization to ensure performance increase.

4.5 Limitations

Although we achieved robustness performance in our hybrid model, some issues are still limiting the construction of a superior model with the best performance in terms of all comparison criteria. In this context, the first and foremost issue is the lack of a sufficient amount of data in the datasets.

In this study, the total number of data is 1,124 in the whole dataset. Although the datasets include well-defined entries, it is critical to feed the model with more training and testing data to get more reliable results. The second issue occurs from the unbalanced ratio of classes.

In this dissertation, we had to use the imbalanced datasets that consist of 417 positive entries and 707 negative entries. The ratio of positive and negative entries was about 1/1.7. This imbalanced ratio may lead to limiting to get better performance results. Thus, we will focus on more comprehensive datasets constructed by either humans or other organisms in future research.

The other limitation stems from the annotation process of each small RNA sequence even though it has a variety of restricted procedures. In the miRBase database, each entry has a probabilistic score with a high confidence level. To ensure this confidence level, some criteria are applied to those sequences before adding them to the database. Thus, this probability must be considered when developing or constructing a deep-learning-based method for predicting miRNAs sequences derived from the miRBase database. Since having very few possibilities can lead to bias during the training or mistraining of the model. Therefore, we consider that this bias may affect our training and also testing process. By developing more precise annotation methods, we will be able to construct more robust models with better performance in terms of all comparison criteria.

4.6 Future Studies

We consider that the quality of data, the size of the dataset, the number of samples of each class, and the variety of the organisms are important issues that affect the results of the implementations. If all those parameters reach sufficient and required size and quality then we will ensure that the trained model will achieve robust classification prediction.

In future studies, we will benefit from new enhanced datasets for the construction of more successful models in terms of similar evaluation criteria. Thus, we would like

to receive better performance. Additionally, we will consider setting up new models with the newly developed deep learning-based methods. Those are the two main goals of our future studies. In that context, for instance, we will compare the performance of the models that consisting of different organism's pre-miRNAs. Similarly, we will test the same models with the same organism when the newest datasets existed.

During the COVID-19 pandemic, it is revealed that how computational biology studies are important in particular identification and/or classification of the sequences of viruses. Furthermore, those are also playing an important role in diagnosis of the coronavirus-related diseases. So, developing a deep-learning-based model may accelerate the process of the new treatments. In future studies, we will also take into account the virus-based sequence analysis issues and share our new experiences that are gained in robust artificial intelligence methods.

Finally, we would like to apply our proposed model to similar but not the same biological datasets to show if the model is proper or not for other research areas. Thus, we will continue to contribute to the literature of computational biology.

CHAPTER 5

CONCLUSION

This dissertation is an investigation of the human pre-miRNA classification problem through a convolutional neural network and long short-term memory network. In contrast to other methods, we considered both the sequence structure and the spatial information of each entry.

The preprocessing of data is the first but the most important stage of our study. Indeed, inappropriate preparation of the data will cause the network to be trained incorrectly and will make it difficult to obtain reliable results. Thus, we checked all outputs after the encoding, padding, and vectorization process. In addition, cascading the different neural networks is another issue in the model construction. Inappropriate network design may increase the bias and cause unexpected results. Therefore, we ensured that all the layers cascaded correctly.

Hyper-parameters determine the general characteristics of deep neural networks. The number of the hidden units, order of the layers, batch size, optimizer selection, and learning rate, etc. directly affect the performance of the methods. In this study, we utilized the previous researcher's experiments in addition to our experiments. For instance, Zheng et al. [32] discovered that kernel size (6x4) and unit number (128) of the CNN network produced the best results according to other sizes and numbers in the pre-miRNA classification. When we tested hyperparameters like Zheng et al. [32], we obtained similar performance results as they did. In our future work, we will take into account the experiences we have gained in these studies and we will do more extensive hyper-parameter optimization to ensure performance increase.

Despite the promising performance of our model, there are still some limitations. The first limitation comes from the total number of entries (1,124) in the datasets. Even though the datasets have well-defined data, it is important to feed the method with more training and testing data to obtain more reliable results. The second limitation is the unbalanced ratio of classes. In this study, the number of positive samples (417) was less than the number of negative samples (707). The ratio of positive and

negative samples was approximately 1/1.7. This imbalanced ratio may lead to limit accuracy and other matrices. Thus, we will focus on more comprehensive datasets in future research.

We consider that the quality and size of the related dataset are important for training a model and achieving robust classification prediction. In future studies, enhanced datasets may lead to the construction of more successful models in terms of similar evaluation parameters.

In this dissertation, we proposed a nucleotide-level hybrid deep learning method based on convolutional neural networks and long-short term memory networks together. In the data preprocessing phase, we used one-hot encoding to convert each base to a matrix of the same size by padding. Then, we employed three convolution layers wrapped by a time-distribution layer. For concatenation of all extracted features, we employed a flatten layer for passing to the LSTM layer. Next, we designed one LSTM layer following a dropout layer on the fully connected layer. Finally, for binary classification, the softmax activation function is used for specifying the outputs. Our results showed that the proposed method was successfully trained on the training dataset and had a better performance on the test dataset than the previous models[113].

The results indicated that the hybrid CNN and LSTM networks can be employed to achieve better performance for human pre-miRNA classification. In future work, we will study the investigation of new classification models that deliver better performance in terms of all the evaluation criteria.

REFERENCES

- [1] S.M. Hammond, An overview of microRNAs, *Adv. Drug Deliv. Rev.* 87 (n.d.).
- [2] D.P. Bartel, MicroRNAs: genomics, biogenesis, mechanism, and function, *Cell*. 116 (n.d.) 281–297.
- [3] V. Ambros, The functions of animal microRNAs, *Nature*. 431 (n.d.).
- [4] M. Bhaskaran, M. Mohan, MicroRNAs: History, Biogenesis, and Their Evolving Role in Animal Development and Disease, *Vet. Pathol.* (2014). <https://doi.org/10.1177/0300985813502820>.
- [5] A. Kozomara, M. Birgaoanu, S. Griffiths-Jones, MiRBase: From microRNA sequences to function, *Nucleic Acids Res.* 47 (2019) D155–D162. <https://doi.org/10.1093/nar/gky1141>.
- [6] W. Filipowicz, S.N. Bhattacharyya, N. Sonenberg, Mechanisms of post-transcriptional regulation by microRNAs: Are the answers in sight?, *Nat. Rev. Genet.* (2008). <https://doi.org/10.1038/nrg2290>.
- [7] G. Meister, T. Tuschl, Mechanisms of gene silencing by double-stranded RNA, *Nature*. 431 (n.d.).
- [8] X. Karp, V. Ambros, Encountering MicroRNAs in Cell Fate Signaling, *Science* (80-.). 310 (n.d.) 1288–1289.
- [9] A.M. Cheng, M.W. Byrom, J. Shelton, L.P. Ford, Antisense inhibition of human miRNAs and indications for an involvement of miRNA in cell growth and apoptosis, *Nucleic Acids Res.* 33 (n.d.) 1290–1297.
- [10] D. Kir, E. Schnettler, S. Modi, S. Ramakrishnan, Regulation of angiogenesis by microRNAs in cardiovascular diseases. *Angiogenesis*, (n.d.). <https://doi.org/10.1007/s10456-018-9632-7>.
- [11] J. Lu, G. Getz, E.A. Miska, E. Alvarez-Saavedra, J. Lamb, D. Peck, A. Sweet-Cordero, B.L. Ebert, R.H. Mak, A.A. Ferrando, J.R. Downing, T. Jacks, H.R. Horvitz, T.R. Golub, MicroRNA expression profiles classify human cancers, *Nature*. (2005). <https://doi.org/10.1038/nature03702>.
- [12] E.A. Miska, How microRNAs control cell division, differentiation and death, *Curr. Opin. Genet. Dev.* 15 (n.d.).

- [13] E.A. Mandujano-Tinoco, A. Garcia-Venzor, J. Melendez-Zajgla, V. Maldonado, New emerging roles of microRNAs in breast cancer, *Breast Cancer Res. treatment*, (n.d.) 10–1007.
- [14] R.P. Singh, The role of miRNA in inflammation and autoimmunity, *Autoimmun. Rev.* 12 (n.d.) 10–1016.
- [15] G.A. Calin, C. Sevignani, C.D. Dumitru, T. Hyslop, E. Noch, S. Yendamuri, M. Shimizu, S. Rattan, F. Bullrich, M. Negrini, C.M. Croce, Human microRNA genes are frequently located at fragile sites and genomic regions involved in cancers, *Proc. Natl. Acad. Sci. U. S. A.* (2004). <https://doi.org/10.1073/pnas.0307323101>.
- [16] C. Li, Y. Feng, G. Coukos, L. Zhang, Therapeutic microRNA strategies in human cancer, *AAPS J.* (2009). <https://doi.org/10.1208/s12248-009-9145-9>.
- [17] R. Garzon, G.A. Calin, C.M. Croce, MicroRNAs in cancer, *Annu. Rev. Med.* (2009). <https://doi.org/10.1146/annurev.med.59.053006.104707>.
- [18] Y. Zhao, E. Samal, D. Srivastava, Serum response factor regulates a muscle-specific microRNA that targets Hand2 during cardiogenesis, *Nature*. (2005). <https://doi.org/10.1038/nature03817>.
- [19] Y. Cheng, C. Zhang, MicroRNA-21 in cardiovascular disease, *J. Cardiovasc. Transl. Res.* (2010). <https://doi.org/10.1007/s12265-010-9169-7>.
- [20] Y. Cheng, R. Ji, J. Yue, J. Yang, X. Liu, H. Chen, D.B. Dean, C. Zhang, MicroRNAs are aberrantly expressed in hypertrophic heart: Do they play a role in cardiac hypertrophy?, *Am. J. Pathol.* (2007). <https://doi.org/10.2353/ajpath.2007.061170>.
- [21] E. Sonkoly, T. Wei, P.C.J. Janson, A. Sääf, L. Lundeberg, M. Tengvall-Linder, G. Norstedt, H. Alenius, B. Homey, A. Scheynius, M. Stähle, A. Pivarcsi, MicroRNAs: Novel Regulators Involved in the Pathogenesis of Psoriasis?, *PLoS One*. (2007). <https://doi.org/10.1371/journal.pone.0000610>.
- [22] Y.S. Lee, A. Dutta, MicroRNAs in Cancer, *Annu. Rev. Pathol. Mech. Dis.* (2009). <https://doi.org/10.1146/annurev.pathol.4.110807.092222>.
- [23] Y. Peng, C.M. Croce, The role of micrornas in human cancer. *Signal transduction targeted therapy*, 15004 (n.d.).
- [24] S. Qin, C. Zhang, Micrornas in vascular disease, *J. Cardiovasc. Pharmacol.* 57

(n.d.).

- [25] M.S. Jamaluddin, Mirnas: roles and clinical applications in vascular disease, *Expert. Rev. Mol. Diagnostics*. 11 (n.d.) 79–89.
- [26] S.R. Dalal, J.H. Kwon, The role of microrna in inflammatory bowel disease, *Gastroenterol. Hepatol*. 6 (n.d.).
- [27] C.G. Chapman, J. Pekow, The emerging role of mirnas in inflammatory bowel disease: a review, *Ther. Adv. Gastroenterol*. 8 (n.d.) 4–22.
- [28] J. Hayes, P.P. Peruzzi, S. Lawler, Micrnas in cancer: biomarkers, functions and therapy, *Trends Mol. Med*. 20 (n.d.) 460–469.
- [29] V. Saini, R. Dawar, S. Suneja, S. Gangopadhyay, C. Kaur, Can microRNA become next-generation tools in molecular diagnostics and therapeutics? A systematic review, *Egypt. J. Med. Hum. Genet*. (2021). <https://doi.org/10.1186/s43042-020-00125-w>.
- [30] N.C. for B. Information, miRNA or MicroRNA, (n.d.). [https://pubmed.ncbi.nlm.nih.gov/?term=microrna or mirna](https://pubmed.ncbi.nlm.nih.gov/?term=microrna+or+mirna) (accessed November 15, 2020).
- [31] J. Alles, T. Fehlmann, U. Fischer, C. Backes, V. Galata, M. Minet, M. Hart, M. Abu-Halima, F.A. Grässer, H.P. Lenhof, A. Keller, E. Meese, An estimate of the total number of true human miRNAs, *Nucleic Acids Res*. (2019). <https://doi.org/10.1093/nar/gkz097>.
- [32] X. Zheng, S. Xu, Y. Zhang, X. Huang, Nucleotide-level Convolutional Neural Networks for Pre-miRNA Classification, *Sci. Rep*. 9 (2019). <https://doi.org/10.1038/s41598-018-36946-4>.
- [33] R.C. Lee, R.L. Feinbaum, V. Ambros, The *C. elegans* heterochronic gene *lin-4* encodes small RNAs with antisense complementarity to *lin-14*, *Cell*. (1993). [https://doi.org/10.1016/0092-8674\(93\)90529-Y](https://doi.org/10.1016/0092-8674(93)90529-Y).
- [34] B. Wightman, I. Ha, G. Ruvkun, Posttranscriptional regulation of the heterochronic gene *lin-14* by *lin-4* mediates temporal pattern formation in *C. elegans*, *Cell*. (1993). [https://doi.org/10.1016/0092-8674\(93\)90530-4](https://doi.org/10.1016/0092-8674(93)90530-4).
- [35] R. Lee, R. Feinbaum, V. Ambros, A short history of a short RNA., *Cell*. (2004). [https://doi.org/10.1016/s0092-8674\(04\)00035-2](https://doi.org/10.1016/s0092-8674(04)00035-2).
- [36] B.J. Reinhart, F.J. Slack, M. Basson, A.E. Pasquienelll, J.C. Bettlnger, A.E.

- Rougvie, H.R. Horvitz, G. Ruvkun, The 21-nucleotide let-7 RNA regulates developmental timing in *Caenorhabditis elegans*, *Nature*. (2000). <https://doi.org/10.1038/35002607>.
- [37] F.J. Slack, M. Basson, Z. Liu, V. Ambros, H.R. Horvitz, G. Ruvkun, The lin-41 RBCC gene acts in the *C. elegans* heterochronic pathway between the let-7 regulatory RNA and the LIN-29 transcription factor, *Mol. Cell*. (2000). [https://doi.org/10.1016/S1097-2765\(00\)80245-2](https://doi.org/10.1016/S1097-2765(00)80245-2).
- [38] A.E. Pasquinelli, B.J. Reinhart, F. Slack, M.Q. Martindale, M.I. Kuroda, B. Maller, D.C. Hayward, E.E. Ball, B. Degnan, P. Müller, J. Spring, A. Srinivasan, M. Fishman, J. Finnerty, J. Corbo, M. Levine, P. Leahy, E. Davidson, G. Ruvkun, Conservation of the sequence and temporal expression of let-7 heterochronic regulatory RNA, *Nature*. (2000). <https://doi.org/10.1038/35040556>.
- [39] S. Griffiths-Jones, The microRNA registry, *Nucleic Acids Res*. (2004). <https://doi.org/10.1093/nar/gkh023>.
- [40] S. Griffiths-Jones, H.K. Saini, S. Van Dongen, A.J. Enright, miRBase: Tools for microRNA genomics, *Nucleic Acids Res*. (2008). <https://doi.org/10.1093/nar/gkm952>.
- [41] H. Siomi, M.C. Siomi, Posttranscriptional Regulation of MicroRNA Biogenesis in Animals, *Mol. Cell*. (2010). <https://doi.org/10.1016/j.molcel.2010.03.013>.
- [42] J. Han, Molecular basis for the recognition of primary microRNAs by the Drosha-DGCR8 complex, *Cell*. 125 (n.d.) 10–1016.
- [43] J. O'Brien, H. Hayder, Y. Zayed, C. Peng, Overview of microRNA biogenesis, mechanisms of actions, and circulation, *Front. Endocrinol. (Lausanne)*. (2018). <https://doi.org/10.3389/fendo.2018.00402>.
- [44] K. Okamura, J.W. Hagen, H. Duan, D.M. Tyler, E.C. Lai, The Mirtron Pathway Generates microRNA-Class Regulatory RNAs in *Drosophila*, *Cell*. 130 (2007) 89–100. <https://doi.org/10.1016/j.cell.2007.06.028>.
- [45] A.M. Denli, B.B. Tops, R.H. Plasterk, R.F. Ketting, G.J. Hannon, Processing of primary microRNAs by the microprocessor complex, *Nat*. 432 (n.d.).
- [46] R.I. Gregory, K.P. Yan, G. Amuthan, T. Chendrimada, B. Doratotaj, N.

- Cooch, R. Shiekhattar, The Microprocessor complex mediates the genesis of microRNAs, *Nature*. (2004). <https://doi.org/10.1038/nature03120>.
- [47] Y. Lee, C. Ahn, J. Han, H. Choi, J. Kim, J. Yim, J. Lee, P. Provost, O. Rådmark, S. Kim, V.N. Kim, The nuclear RNase III Drosha initiates microRNA processing, *Nature*. (2003). <https://doi.org/10.1038/nature01957>.
- [48] E. Lund, S. Guttinger, A. Calado, J.E. Dahlberg, U. Kutay, Nuclear export of microRNA precursors, *Science* (80-.). 303 (n.d.) 10–1126.
- [49] M.M.W. Chong, G. Zhang, S. Cheloufi, T.A. Neubert, G.J. Hannon, D.R. Littman, Canonical and alternate functions of the microRNA biogenesis machinery, *Genes Dev.* (2010). <https://doi.org/10.1101/gad.1953310>.
- [50] J.G. Ruby, C.H. Jan, D.P. Bartel, Intronic microRNA precursors that bypass Drosha processing, *Nature*. (2007). <https://doi.org/10.1038/nature05983>.
- [51] J.O. Westholm, E.C. Lai, Mirtrons: MicroRNA biogenesis via splicing, *Biochimie*. (2011). <https://doi.org/10.1016/j.biochi.2011.06.017>.
- [52] E. Berezikov, W.J. Chung, J. Willis, E. Cuppen, E.C. Lai, Mammalian Mirtron Genes, *Mol. Cell*. (2007). <https://doi.org/10.1016/j.molcel.2007.09.028>.
- [53] J. Wen, E. Ladewig, S. Shenker, J. Mohammed, E.C. Lai, Analysis of Nearly One Thousand Mammalian Mirtrons Reveals Novel Features of Dicer Substrates, *PLoS Comput. Biol.* (2015). <https://doi.org/10.1371/journal.pcbi.1004441>.
- [54] B. Fromm, A Uniform System for the Annotation of Vertebrate microRNA Genes and the Evolution of the Human microRNAome, in: *Annu. Rev. Genet.* 49, n.d.: pp. 213–242.
- [55] Y.K. Kim, V.N. Kim, Processing of intronic microRNAs, *EMBO J.* (2007). <https://doi.org/10.1038/sj.emboj.7601512>.
- [56] A.M. Abdelfattah, C. Park, M.Y. Choi, Update on non-canonical microRNAs, *Biomol. Concepts*. (2014). <https://doi.org/10.1515/bmc-2014-0012>.
- [57] M. Database, miRNA entry, (n.d.). http://www.mirbase.org/cgi-bin/mirna_entry.pl?acc=hsa-let-7a-1 (accessed November 15, 2020).
- [58] Kozomara Ana, Griffiths-Jones Sam, miRBase: annotating high confidence microRNAs using deep sequencing data, *Nucleic Acids Res.* (2013).
- [59] I.L. Hofacker, Vienna RNA secondary structure server, *Nucleic Acids Res.* 31

(n.d.) 3429–3431.

- [60] K.L.S. Ng, S.K. Mishra, De novo SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures, *Bioinformatics*. (2007). <https://doi.org/10.1093/bioinformatics/btm026>.
- [61] P. Jiang, H. Wu, W. Wang, W. Ma, X. Sun, Z. Lu, MiPred: Classification of real and pseudo microRNA precursors using random forest prediction model with combined features, *Nucleic Acids Res.* (2007). <https://doi.org/10.1093/nar/gkm368>.
- [62] M.D.S. Demirci, J. Baumbach, J. Allmer, M.D. Sacar Demirci, J. Baumbach, J. Allmer, On the performance of pre-microrna detection algorithms, *Nat. Commun.* 8 (n.d.) 330. <https://doi.org/10.1038/s41467-017-00403-z>.
- [63] X. Zheng, X. Fu, K. Wang, M. Wang, Deep neural networks for human microRNA precursor detection, *BMC Bioinformatics*. (2020). <https://doi.org/10.1186/s12859-020-3339-7>.
- [64] B.T. Do, V. Golkov, G.E. Gürel, D. Cremers, Precursor microRNA identification using deep convolutional neural networks, *BioRxiv*. (2018). <https://doi.org/10.1101/414656>.
- [65] J. Cordero, V. Menkovski, J. Allmer, Detection of pre-microRNA with convolutional neural networks, *BioRxiv*. (2019). <https://doi.org/10.1101/840579>.
- [66] C. Xue, Classification of real and pseudo microrna precursors using local structure-sequence features and support vector machine, *BMC Bioinformatics*. 6 (n.d.).
- [67] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, *Nature*. (2015). <https://doi.org/10.1038/nature14539>.
- [68] W. Jones, K. Alasoo, D. Fishman, L. Parts, Computational biology: deep learning, *Emerg. Top. Life Sci.* (2017). <https://doi.org/10.1042/etls20160025>.
- [69] C. Angermueller, T. Pärnamaa, L. Parts, O. Stegle, Deep learning for computational biology, *Mol. Syst. Biol.* (2016). <https://doi.org/10.15252/msb.20156651>.
- [70] M.D. Sacar Demirci, J. Baumbach, J. Allmer, On the performance of pre-microRNA detection algorithms, *Nat. Commun.* 8 (n.d.) 330.

<https://doi.org/10.1038/s41467-017-00403-z>.

- [71] S. Gambhir, S.K. Malik, Y. Kumar, Role of Soft Computing Approaches in HealthCare Domain: A Mini Review, *J. Med. Syst.* (2016). <https://doi.org/10.1007/s10916-016-0651-x>.
- [72] M. Peker, B. Şen, D. Delen, Computer-aided diagnosis of Parkinson's disease using complex-valued neural networks and mRMR feature selection algorithm, *J. Healthc. Eng.* (2015). <https://doi.org/10.1260/2040-2295.6.3.281>.
- [73] B. Şen, M. Peker, Novel approaches for automated epileptic diagnosis using FCBF selection and classification algorithms, *Turkish J. Electr. Eng. Comput. Sci.* (2013). <https://doi.org/10.3906/elk-1203-9>.
- [74] M. Peker, B. Sen, D. Delen, A novel method for automated diagnosis of epilepsy using complex-valued classifiers, *IEEE J. Biomed. Heal. Informatics.* (2016). <https://doi.org/10.1109/JBHI.2014.2387795>.
- [75] F. Atasoy, B. Sen, F. Nar, I. Bozkurt, Improvement of Radial basis Function Interpolation Performance on Cranial Implant Design, *Int. J. Adv. Comput. Sci. Appl.* (2017). <https://doi.org/10.14569/ijacsa.2017.080811>.
- [76] D.S. Huang, A constructive approach for finding arbitrary roots of polynomials by neural networks, *IEEE Trans. Neural Networks.* (2004). <https://doi.org/10.1109/TNN.2004.824424>.
- [77] Y. Zhang, D. Zhang, G. Mi, D. Ma, G. Li, Y. Guo, M. Li, M. Zhu, Using ensemble methods to deal with imbalanced data in predicting protein-protein interactions, *Comput. Biol. Chem.* (2012). <https://doi.org/10.1016/j.compbiolchem.2011.12.003>.
- [78] J.P. Albuquerque Vieira, R.S. Moura, An Analysis of Convolutional Neural Networks for Sentence Classification, (n.d.).
- [79] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, *Commun. Acn.* 60 (n.d.) 10–1145.
- [80] S. Park, S. Min, H. Choi, S. Yoon, deepMiRGene: Deep Neural Network based Precursor microRNA Prediction, (2016). <http://arxiv.org/abs/1605.00017> (accessed June 6, 2021).
- [81] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature

- embedding, in: MM 2014 - Proc. 2014 ACM Conf. Multimed., 2014. <https://doi.org/10.1145/2647868.2654889>.
- [82] Z.Q. Zhao, B.J. Xie, Y.M. Cheung, X. Wu, Plant leaf identification via a growing convolution neural network with progressive sample learning, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2015. https://doi.org/10.1007/978-3-319-16808-1_24.
- [83] Y. Xiang, W. Choi, Y. Lin, S. Savarese, Subcategory-Aware convolutional neural networks for object proposals & detection, in: Proc. - 2017 IEEE Winter Conf. Appl. Comput. Vision, WACV 2017, 2017. <https://doi.org/10.1109/WACV.2017.108>.
- [84] R.L. Galvez, A.A. Bandala, E.P. Dadios, R.R.P. Vicerra, J.M.Z. Maningo, Object Detection Using Convolutional Neural Networks, in: IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON, 2019. <https://doi.org/10.1109/TENCON.2018.8650517>.
- [85] O. Abdel-Hamid, A.R. Mohamed, H. Jiang, G. Penn, Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition, in: ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc., 2012. <https://doi.org/10.1109/ICASSP.2012.6288864>.
- [86] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, F.F. Li, Large-scale video classification with convolutional neural networks, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2014. <https://doi.org/10.1109/CVPR.2014.223>.
- [87] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, A.Y. Ng, Multimodal deep learning, in: Proc. 28th Int. Conf. Mach. Learn. ICML 2011, 2011.
- [88] B. Alipanahi, A. Delong, M.T. Weirauch, B.J. Frey, Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning, Nat. Biotechnol. (2015). <https://doi.org/10.1038/nbt.3300>.
- [89] J. Zhou, O.G. Troyanskaya, Predicting effects of noncoding variants with deep learning-based sequence model, Nat. Methods. (2015). <https://doi.org/10.1038/nmeth.3547>.
- [90] K. Greff, R.K. Srivastava, J. Koutnik, B.R. Steunebrink, J. Schmidhuber, LSTM: A Search Space Odyssey, IEEE Trans. Neural Networks Learn. Syst.

- (2017). <https://doi.org/10.1109/TNNLS.2016.2582924>.
- [91] F. Gers, Long short-term memory in recurrent neural networks, *Neural Comput.* (2001).
 - [92] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, *Neural Comput.* (1997). <https://doi.org/10.1162/neco.1997.9.8.1735>.
 - [93] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with LSTM, in: *IEE Conf. Publ.*, 1999. <https://doi.org/10.1049/cp:19991218>.
 - [94] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with LSTM, *Neural Comput.* (2000). <https://doi.org/10.1162/089976600300015015>.
 - [95] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.* (1998). <https://doi.org/10.1142/S0218488598000094>.
 - [96] J. Donahue, L.A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, T. Darrell, Long-Term Recurrent Convolutional Networks for Visual Recognition and Description, *IEEE Trans. Pattern Anal. Mach. Intell.* (2017). <https://doi.org/10.1109/TPAMI.2016.2599174>.
 - [97] T.N. Sainath, O. Vinyals, A. Senior, H. Sak, Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks, in: *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, 2015. <https://doi.org/10.1109/ICASSP.2015.7178838>.
 - [98] D. Quang, X. Xie, DanQ: A hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences, *Nucleic Acids Res.* 44 (2016) e107–e107. <https://doi.org/10.1093/nar/gkw226>.
 - [99] X. Pan, P. Rijnbeek, J. Yan, H. Bin Shen, Prediction of RNA-protein sequence and structure binding preferences using deep convolutional and recurrent neural networks, *BMC Genomics.* 19 (2018) 511. <https://doi.org/10.1186/s12864-018-4889-1>.
 - [100] R. Yan, F. Ren, Z. Wang, L. Wang, Y. Ren, Y. Liu, X. Rao, C. Zheng, F. Zhang, A Hybrid Convolutional and Recurrent Deep Neural Network for Breast Cancer Pathological Image Classification, in: *Proc. - 2018 IEEE Int.*

- Conf. Bioinforma. Biomed. BIBM 2018, 2019.
<https://doi.org/10.1109/BIBM.2018.8621429>.
- [101] G. Rorbach, O. Unold, B.M. Konopka, Distinguishing mirtrons from canonical miRNAs with data exploration and machine learning methods, *Sci. Rep.* 8 (n.d.) 10–1038.
 - [102] D. Berrar, Cross-validation, in: *Encycl. Bioinforma. Comput. Biol. ABC Bioinforma.*, 2018. <https://doi.org/10.1016/B978-0-12-809633-8.20349-X>.
 - [103] H. He, Y. Ma, Imbalanced learning: foundations, algorithms, and applications, (2013).
 - [104] X.-H. Wu, J.-Q. Wang, Cross-Entropy Measures Of Multivalued Neutrosophic Sets And Its Application In Selecting Middle-Level Manager, *Int. J. Uncertain. Quantif.* 7 (n.d.) 10–1615.
 - [105] I. Tolstikhin, O. Bousquet, B. Schölkopf, K. Thierbach, P.L. Bazin, W. de Back, F. Gavriilidis, E. Kirilina, C. Jäger, M. Morawski, S. Geyer, N. Weiskopf, N. Scherf, I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, E. Musk, Neuralink, M.A. Hjortsø, P. Wolenski, S. Ruder, W. Grathwohl, R.T.Q. Chen, J. Bettencourt, I. Sutskever, D. Duvenaud, C. Doersch, An overview of gradient descent optimization algorithms, *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*. (2018).
 - [106] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *J. Mach. Learn. Res.* (2011).
 - [107] A.A. Lydia, F.S. Francis, Adagrad-An Optimizer for Stochastic Gradient Descent, 2019.
 - [108] M.D. Zeiler, Adadelata: An adaptive learning rate method, *ArXiv Prepr.* 1212.5701 (2012).
 - [109] T. Tieleman, G. Hinton, Lecture 6.5 - rmsprop, COURSERA Neural Networks Mach. Learn. (2012).
 - [110] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., International Conference on Learning Representations, ICLR, 2015. <https://arxiv.org/abs/1412.6980v9> (accessed June 7, 2021).

- [111] J.M. Johnson, T.M. Khoshgoftaar, Survey on deep learning with class imbalance, *J. Big Data.* 6 (2019) 1–54. <https://doi.org/10.1186/s40537-019-0192-5>.
- [112] S. Kotsiantis, D. Kanellopoulos, P.E. Pintelas, D. Kanellopoulos, P. Pintelas, Handling imbalanced datasets: A review *Data preprocessing View project Machine Learning and Data Mining View project Handling imbalanced datasets: A review*, n.d. <https://www.researchgate.net/publication/228084509> (accessed June 8, 2021).
- [113] A. Tasdelen, B. Sen, A hybrid CNN-LSTM model for pre-miRNA classification, *Sci. Rep.* 11 (2021) 14125. <https://doi.org/10.1038/s41598-021-93656-0>.

APPENDICES

Appendix A: Detailed architecture of the model

Appendix B: Results of stratified 5-folds Cross-Validation



Appendix B- Results of stratified 5-folds Cross-Validation

Results for fold number 1

TRAIN INPUT #: 899

TRAIN OUTPUT #: 899

Model: "sequential_54"

Layer (type)	Output Shape	Param #
=====		
time_distributed_216 (TimeDistributed)	(None, None, 82, 2, 128)	3200
=====		
time_distributed_217 (TimeDistributed)	(None, None, 41, 1, 128)	393344
=====		
time_distributed_218 (TimeDistributed)	(None, None, 21, 1, 128)	393344
=====		
time_distributed_219 (TimeDistributed)	(None, None, 2688)	0
=====		
lstm_54 (LSTM)	(None, 100)	1115600
=====		
dense_108 (Dense)	(None, 256)	25856
=====		
Drop out_54 (Drop out)	(None, 256)	0
=====		
dense_109 (Dense)	(None, 2)	514
=====		

Total parameters: 1,931,858

Trainable parameters: 1,931,858

Non-trainable parameters: 0

None

Epoch 1/30

135/135 [=====] - 49s 52ms/step - los:

0.4912 - accr: 0.7532 - val_loss: 0.2403 - val-accr.: 0.8778

Epoch 2/30

```

135/135 [=====] - 6s 44ms/step - los:
0.2609 - accr: 0.9018 - val_loss: 0.2402 - val-accr.: 0.9000
Epoch 3/30
135/135 [=====] - 6s 44ms/step - los:
0.1995 - accr: 0.9246 - val_loss: 0.1855 - val-accr.: 0.9111
Epoch 4/30
135/135 [=====] - 6s 44ms/step - los:
0.1659 - accr: 0.9536 - val_loss: 0.1958 - val-accr.: 0.9444
Epoch 5/30
135/135 [=====] - 6s 44ms/step - los:
0.0990 - accr: 0.9649 - val_loss: 0.2544 - val-accr.: 0.9222
Epoch 6/30
135/135 [=====] - 6s 44ms/step - los:
0.0534 - accr: 0.9818 - val_loss: 0.2155 - val-accr.: 0.9222
Epoch 7/30
135/135 [=====] - 6s 44ms/step - los:
0.0594 - accr: 0.9789 - val_loss: 0.2765 - val-accr.: 0.9333
Epoch 8/30
135/135 [=====] - 6s 45ms/step - los:
0.0201 - accr: 0.9925 - val_loss: 0.4356 - val-accr.: 0.9000
Epoch 9/30
135/135 [=====] - 6s 45ms/step - los:
0.0214 - accr: 0.9933 - val_loss: 0.3827 - val-accr.: 0.9111
Epoch 10/30
135/135 [=====] - 6s 44ms/step - los:
0.0401 - accr: 0.9891 - val_loss: 0.1834 - val-accr.: 0.9444
Epoch 11/30
135/135 [=====] - 6s 44ms/step - los:
0.0163 - accr: 0.9968 - val_loss: 0.3285 - val-accr.: 0.9333
Epoch 12/30
135/135 [=====] - 6s 44ms/step - los:
0.0028 - accr: 0.9995 - val_loss: 0.3720 - val-accr.: 0.9333
Epoch 13/30
135/135 [=====] - 6s 44ms/step - los:
0.0084 - accr: 0.9965 - val_loss: 0.3336 - val-accr.: 0.9222
Epoch 14/30

```

```

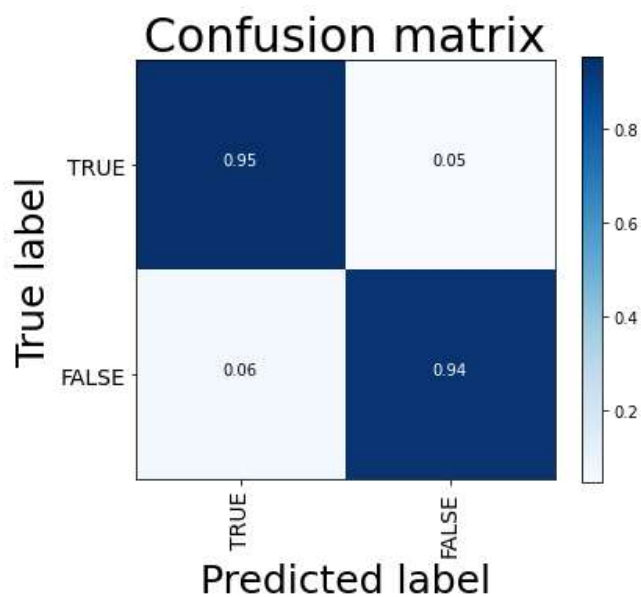
135/135 [=====] - 6s 44ms/step - los:
0.0384 - accr: 0.9920 - val_loss: 0.4559 - val-accr.: 0.9333
Epoch 15/30
135/135 [=====] - 6s 44ms/step - los:
0.0044 - accr: 0.9996 - val_loss: 0.3008 - val-accr.: 0.9444
Epoch 16/30
135/135 [=====] - 6s 44ms/step - los:
0.0016 - accr: 0.9998 - val_loss: 0.3533 - val-accr.: 0.9222
Epoch 17/30
135/135 [=====] - 6s 45ms/step - los:
0.0029 - accr: 0.9986 - val_loss: 0.4718 - val-accr.: 0.9333
Epoch 18/30
135/135 [=====] - 6s 44ms/step - los:
0.0053 - accr: 0.9959 - val_loss: 0.5526 - val-accr.: 0.9333
Epoch 19/30
135/135 [=====] - 6s 44ms/step - los:
7.5066e-04 - accr: 0.9998 - val_loss: 0.5956 - val-accr.:
0.9333
Epoch 20/30
135/135 [=====] - 6s 44ms/step - los:
8.2859e-04 - accr: 0.9994 - val_loss: 0.6136 - val-accr.:
0.9333
Epoch 21/30
135/135 [=====] - 6s 44ms/step - los:
4.0055e-04 - accr: 0.9998 - val_loss: 0.6338 - val-accr.:
0.9333
Epoch 22/30
135/135 [=====] - 6s 44ms/step - los:
0.0017 - accr: 0.9991 - val_loss: 0.6582 - val-accr.: 0.9333
Epoch 23/30
135/135 [=====] - 6s 44ms/step - los:
1.7725e-04 - accr: 1.0000 - val_loss: 0.6781 - val-accr.:
0.9333
Epoch 24/30
135/135 [=====] - 6s 45ms/step - los:
0.0011 - accr: 1.0000 - val_loss: 0.7194 - val-accr.: 0.9333

```

```

Epoch 25/30
135/135 [=====] - 6s 44ms/step - los:
0.0058 - accr: 0.9956 - val_loss: 0.7030 - val-accr.: 0.9333
Epoch 26/30
135/135 [=====] - 6s 44ms/step - los:
8.2705e-04 - accr: 0.9992 - val_loss: 0.7452 - val-accr.:
0.9333
Epoch 27/30
135/135 [=====] - 6s 44ms/step - los:
9.2296e-04 - accr: 0.9996 - val_loss: 0.7386 - val-accr.:
0.9222
Epoch 28/30
135/135 [=====] - 6s 44ms/step - los:
0.0013 - accr: 0.9996 - val_loss: 0.8092 - val-accr.: 0.9111
Epoch 29/30
135/135 [=====] - 6s 44ms/step - los:
6.8659e-04 - accr: 0.9997 - val_loss: 0.8282 - val-accr.:
0.9222
Epoch 30/30
135/135 [=====] - 6s 44ms/step - los:
0.0023 - accr: 1.0000 - val_loss: 0.8592 - val-accr.: 0.9222
Accr: 94.22%
overall score: [94.22222375869751]
-----

```



-----Confusion Matrix Stats-----

```
[[ 79   4]
```

```
 [  9 133]]
```

Accr Score : 0.9422222222222222

Report :

	pre	rec	F1_Score	supp.	
0		0.90	0.95	0.92	83
1		0.97	0.94	0.95	142
accr				0.94	225
macro avg		0.93	0.94	0.94	225
weighted avg		0.94	0.94	0.94	225


```

[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]]]]
vectorized_Ytrain: [0, 1]
vectorized_Ytest: [0, 1]
Model: "sequential_55"

```

Layer (type)	Output Shape	Param #
=====		
time_distributed_220 (TimeDi	(None, None, 82, 2, 128)	3200

time_distributed_221 (TimeDi	(None, None, 41, 1, 128)	393344

time_distributed_222 (TimeDi	(None, None, 21, 1, 128)	393344

time_distributed_223 (TimeDi	(None, None, 2688)	0

lstm_55 (LSTM)	(None, 100)	1115600

dense_110 (Dense)	(None, 256)	25856

drop out_55 (Drop out)	(None, 256)	0

dense_111 (Dense)	(None, 2)	514

```

=====
Total parameters: 1,931,858
Trainable parameters: 1,931,858
Non-trainable parameters: 0

-----
None
Epoch 1/30
135/135 [=====] - 10s 52ms/step - los:
0.5718 - accr: 0.6931 - val_loss: 0.2905 - val-accr.: 0.8556
Epoch 2/30
135/135 [=====] - 6s 45ms/step - los:
0.3264 - accr: 0.8651 - val_loss: 0.2033 - val-accr.: 0.9111
Epoch 3/30
135/135 [=====] - 6s 45ms/step - los:
0.2207 - accr: 0.9146 - val_loss: 0.1986 - val-accr.: 0.9000
Epoch 4/30
135/135 [=====] - 6s 45ms/step - los:
0.1864 - accr: 0.9231 - val_loss: 0.2101 - val-accr.: 0.8889
Epoch 5/30
135/135 [=====] - 6s 44ms/step - los:
0.1295 - accr: 0.9610 - val_loss: 0.2466 - val-accr.: 0.9111
Epoch 6/30
135/135 [=====] - 6s 45ms/step - los:
0.0766 - accr: 0.9759 - val_loss: 0.2225 - val-accr.: 0.9222
Epoch 7/30
135/135 [=====] - 6s 45ms/step - los:
0.0569 - accr: 0.9782 - val_loss: 0.2018 - val-accr.: 0.9222
Epoch 8/30
135/135 [=====] - 6s 45ms/step - los:
0.0339 - accr: 0.9948 - val_loss: 0.3281 - val-accr.: 0.9111
Epoch 9/30
135/135 [=====] - 6s 44ms/step - los:
0.0169 - accr: 0.9981 - val_loss: 0.2788 - val-accr.: 0.9222
Epoch 10/30
135/135 [=====] - 6s 44ms/step - los:
0.0148 - accr: 0.9957 - val_loss: 0.2994 - val-accr.: 0.9333

```

Epoch 11/30

135/135 [=====] - 6s 44ms/step - los:
0.0145 - accr: 0.9955 - val_loss: 0.3218 - val-accr.: 0.9333

Epoch 12/30

135/135 [=====] - 6s 45ms/step - los:
0.0019 - accr: 0.9993 - val_loss: 0.4732 - val-accr.: 0.9333

Epoch 13/30

135/135 [=====] - 6s 44ms/step - los:
0.0244 - accr: 0.9964 - val_loss: 0.3704 - val-accr.: 0.9333

Epoch 14/30

135/135 [=====] - 6s 44ms/step - los:
0.0618 - accr: 0.9762 - val_loss: 0.3223 - val-accr.: 0.9000

Epoch 15/30

135/135 [=====] - 6s 44ms/step - los:
0.0467 - accr: 0.9784 - val_loss: 0.4139 - val-accr.: 0.9111

Epoch 16/30

135/135 [=====] - 6s 44ms/step - los:
0.0243 - accr: 0.9921 - val_loss: 0.2949 - val-accr.: 0.9000

Epoch 17/30

135/135 [=====] - 6s 44ms/step - los:
0.0162 - accr: 0.9954 - val_loss: 0.3067 - val-accr.: 0.9444

Epoch 18/30

135/135 [=====] - 6s 45ms/step - los:
0.0095 - accr: 0.9937 - val_loss: 0.3863 - val-accr.: 0.9444

Epoch 19/30

135/135 [=====] - 6s 45ms/step - los:
0.0016 - accr: 0.9985 - val_loss: 0.3966 - val-accr.: 0.9444

Epoch 20/30

135/135 [=====] - 6s 45ms/step - los:
0.0056 - accr: 0.9966 - val_loss: 0.3798 - val-accr.: 0.9444

Epoch 21/30

135/135 [=====] - 6s 45ms/step - los:
0.0031 - accr: 0.9999 - val_loss: 0.4058 - val-accr.: 0.9444

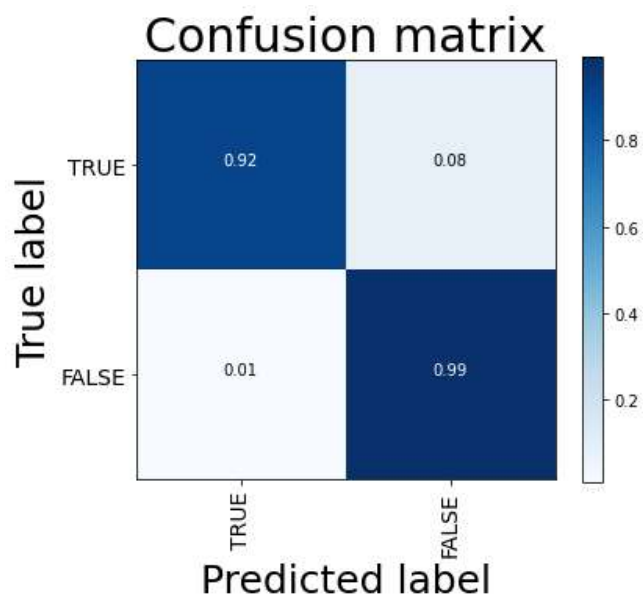
Epoch 22/30

135/135 [=====] - 6s 45ms/step - los:
0.0041 - accr: 0.9960 - val_loss: 0.4586 - val-accr.: 0.9444

```

Epoch 23/30
135/135 [=====] - 6s 44ms/step - los:
0.0042 - accr: 0.9970 - val_loss: 0.4335 - val-accr.: 0.9444
Epoch 24/30
135/135 [=====] - 6s 44ms/step - los:
0.0068 - accr: 0.9943 - val_loss: 0.4788 - val-accr.: 0.9444
Epoch 25/30
135/135 [=====] - 6s 45ms/step - los:
0.0036 - accr: 0.9974 - val_loss: 0.4501 - val-accr.: 0.9444
Epoch 26/30
135/135 [=====] - 6s 44ms/step - los:
0.0021 - accr: 0.9994 - val_loss: 0.5053 - val-accr.: 0.9444
Epoch 27/30
135/135 [=====] - 6s 44ms/step - los:
0.0052 - accr: 0.9969 - val_loss: 0.5322 - val-accr.: 0.9444
Epoch 28/30
135/135 [=====] - 6s 45ms/step - los:
0.0026 - accr: 0.9973 - val_loss: 0.5628 - val-accr.: 0.9444
Epoch 29/30
135/135 [=====] - 6s 44ms/step - los:
0.0044 - accr: 0.9958 - val_loss: 0.5955 - val-accr.: 0.9444
Epoch 30/30
135/135 [=====] - 6s 44ms/step - los:
0.0060 - accr: 0.9964 - val_loss: 0.6056 - val-accr.: 0.9444
Accr: 96.44%
overall score: [94.22222375869751, 96.44444584846497]
-----

```



-----Confusion Matrix Stats-----

```
[[ 76   7]
```

```
 [  1 141]]
```

Accr Score : 0.9644444444444444

Report :

	pre	rec	F1_Score	supp.	
0		0.99	0.92	0.95	83
1		0.95	0.99	0.97	142
accr				0.96	225
macro avg		0.97	0.95	0.96	225
weighted avg		0.97	0.96	0.96	225


```

[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]]]
vectorized_Ytrain: [0, 1]
vectorized_Ytest: [0, 1]
Model: "sequential_56"

```

Layer (type)	Output Shape	Param #
=====		
time_distributed_224 (TimeDi	(None, None, 82, 2, 128)	3200

time_distributed_225 (TimeDi	(None, None, 41, 1, 128)	393344

time_distributed_226 (TimeDi	(None, None, 21, 1, 128)	393344

time_distributed_227 (TimeDi	(None, None, 2688)	0

lstm_56 (LSTM)	(None, 100)	1115600

dense_112 (Dense)	(None, 256)	25856

drop out_56 (Drop out)	(None, 256)	0

dense_113 (Dense)	(None, 2)	514

=====

Total parameters: 1,931,858

Trainable parameters: 1,931,858

Non-trainable parameters: 0

None

Epoch 1/30

135/135 [=====] - 10s 53ms/step - los:
0.5409 - accr: 0.7184 - val_loss: 0.2327 - val-accr.: 0.9222

Epoch 2/30

135/135 [=====] - 6s 45ms/step - los:
0.2867 - accuracy: 0.8953 - val_loss: 0.1809 - val-accr.uracy:
0.9444

Epoch 3/30

135/135 [=====] - 6s 45ms/step - los:
0.2628 - accr: 0.9000 - val_loss: 0.1077 - val-accr.: 0.9444

Epoch 4/30

135/135 [=====] - 6s 45ms/step - los:
0.1320 - accr: 0.9529 - val_loss: 0.2479 - val-accr.: 0.9111

Epoch 5/30

135/135 [=====] - 6s 45ms/step - los:
0.0727 - accr: 0.9737 - val_loss: 0.1327 - val-accr.: 0.9556

Epoch 6/30

135/135 [=====] - 6s 45ms/step - los:
0.0738 - accr: 0.9808 - val_loss: 0.1062 - val-accr.: 0.9667

Epoch 7/30

135/135 [=====] - 6s 45ms/step - los:
0.0486 - accr: 0.9867 - val_loss: 0.2622 - val-accr.: 0.9444

Epoch 8/30

135/135 [=====] - 6s 45ms/step - los:
0.0447 - accr: 0.9815 - val_loss: 0.1064 - val-accr.: 0.9556

Epoch 9/30

135/135 [=====] - 6s 45ms/step - los:
0.0452 - accr: 0.9872 - val_loss: 0.2820 - val-accr.: 0.9333

Epoch 10/30

135/135 [=====] - 6s 45ms/step - los:
0.0378 - accr: 0.9858 - val_loss: 0.1128 - val-accr.: 0.9556
Epoch 11/30

135/135 [=====] - 6s 44ms/step - los:
0.0343 - accr: 0.9903 - val_loss: 0.1710 - val-accr.: 0.9444
Epoch 12/30

135/135 [=====] - 6s 45ms/step - los:
0.0055 - accr: 0.9991 - val_loss: 0.1435 - val-accr.: 0.9667
Epoch 13/30

135/135 [=====] - 6s 45ms/step - los:
0.0126 - accr: 0.9979 - val_loss: 0.1993 - val-accr.: 0.9222
Epoch 14/30

135/135 [=====] - 6s 46ms/step - los:
0.0076 - accr: 0.9976 - val_loss: 0.2803 - val-accr.: 0.9444
Epoch 15/30

135/135 [=====] - 6s 45ms/step - los:
0.0192 - accr: 0.9903 - val_loss: 0.2391 - val-accr.: 0.9333
Epoch 16/30

135/135 [=====] - 6s 45ms/step - los:
0.0215 - accr: 0.9905 - val_loss: 0.2673 - val-accr.: 0.9222
Epoch 17/30

135/135 [=====] - 6s 45ms/step - los:
0.0121 - accr: 0.9959 - val_loss: 0.2551 - val-accr.: 0.9333
Epoch 18/30

135/135 [=====] - 6s 45ms/step - los:
0.0412 - accr: 0.9852 - val_loss: 0.4672 - val-accr.: 0.9333
Epoch 19/30

135/135 [=====] - 6s 45ms/step - los:
0.0145 - accr: 0.9947 - val_loss: 0.4114 - val-accr.: 0.9222
Epoch 20/30

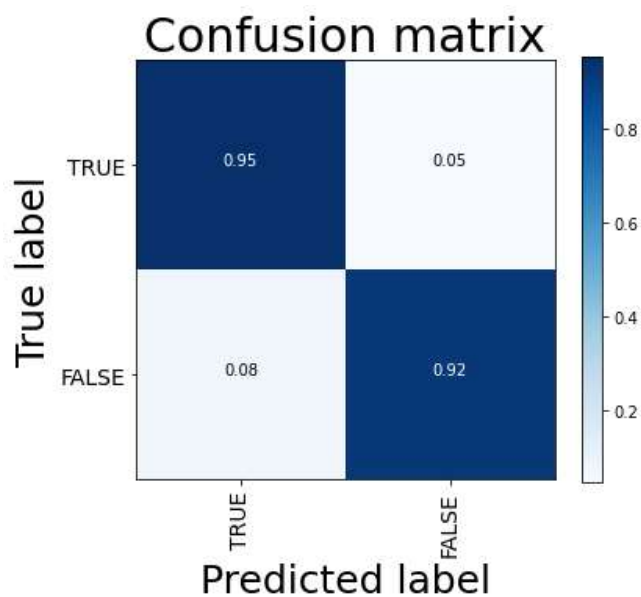
135/135 [=====] - 6s 45ms/step - los:
0.0104 - accr: 0.9978 - val_loss: 0.1066 - val-accr.: 0.9444
Epoch 21/30

135/135 [=====] - 6s 45ms/step - los:
0.0182 - accr: 0.9926 - val_loss: 0.3003 - val-accr.: 0.9444
Epoch 22/30

```

135/135 [=====] - 6s 45ms/step - los:
0.0085 - accr: 0.9967 - val_loss: 0.2853 - val-accr.: 0.9333
Epoch 23/30
135/135 [=====] - 6s 45ms/step - los:
0.0199 - accr: 0.9935 - val_loss: 0.1428 - val-accr.: 0.9333
Epoch 24/30
135/135 [=====] - 6s 45ms/step - los:
0.0302 - accr: 0.9891 - val_loss: 0.2026 - val-accr.: 0.9444
Epoch 25/30
135/135 [=====] - 6s 45ms/step - los:
0.0157 - accr: 0.9919 - val_loss: 0.2291 - val-accr.: 0.9444
Epoch 26/30
135/135 [=====] - 6s 45ms/step - los:
0.0049 - accr: 0.9980 - val_loss: 0.2300 - val-accr.: 0.9333
Epoch 27/30
135/135 [=====] - 6s 45ms/step - los:
0.0057 - accr: 0.9988 - val_loss: 0.3217 - val-accr.: 0.9333
Epoch 28/30
135/135 [=====] - 6s 46ms/step - los:
0.0051 - accr: 0.9985 - val_loss: 0.4105 - val-accr.: 0.9333
Epoch 29/30
135/135 [=====] - 6s 45ms/step - los:
0.0027 - accr: 0.9993 - val_loss: 0.3384 - val-accr.: 0.9333
Epoch 30/30
135/135 [=====] - 6s 45ms/step - los:
0.0070 - accr: 0.9966 - val_loss: 0.4477 - val-accr.: 0.9556
Accr: 93.33%
overall score: [94.22222375869751, 96.44444584846497,
93.33333373069763]
-----

```



-----Confusion Matrix Stats-----

```
[[ 80   4]
 [ 11 130]]
```

Accr Score : 0.9333333333333333

Report :

	pre	rec	F1_Score	supp.	
0		0.88	0.95	0.91	84
1		0.97	0.92	0.95	141
accr				0.93	225
macro avg		0.92	0.94	0.93	225
weighted avg		0.94	0.93	0.93	225


```

[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]]]]
vectorized_Ytrain: [0, 1]
vectorized_Ytest: [0, 1]
Model: "sequential_57"

```

Layer (type)	Output Shape	Param #
=====		
time_distributed_228 (TimeDi	(None, None, 82, 2, 128)	3200

time_distributed_229 (TimeDi	(None, None, 41, 1, 128)	393344

time_distributed_230 (TimeDi	(None, None, 21, 1, 128)	393344

time_distributed_231 (TimeDi	(None, None, 2688)	0

lstm_57 (LSTM)	(None, 100)	1115600

dense_114 (Dense)	(None, 256)	25856

drop out_57 (Drop out)	(None, 256)	0

dense_115 (Dense)	(None, 2)	514

```

=====
Total parameters: 1,931,858
Trainable parameters: 1,931,858
Non-trainable parameters: 0

-----
None
Epoch 1/30
135/135 [=====] - 11s 62ms/step - los:
0.5480 - accr: 0.6926 - val_loss: 0.2098 - val-accr.: 0.8778
Epoch 2/30
135/135 [=====] - 6s 46ms/step - los:
0.3032 - accr: 0.8814 - val_loss: 0.2484 - val-accr.: 0.8778
Epoch 3/30
135/135 [=====] - 6s 46ms/step - los:
0.2661 - accr: 0.8966 - val_loss: 0.1765 - val-accr.: 0.9333
Epoch 4/30
135/135 [=====] - 6s 48ms/step - los:
0.1683 - accr: 0.9480 - val_loss: 0.1580 - val-accr.: 0.9444
Epoch 5/30
135/135 [=====] - 6s 46ms/step - los:
0.1273 - accr: 0.9576 - val_loss: 0.1618 - val-accr.: 0.9444
Epoch 6/30
135/135 [=====] - 6s 46ms/step - los:
0.0644 - accr: 0.9745 - val_loss: 0.2076 - val-accr.: 0.9222
Epoch# 7/30
135/135 [=====] - 6s 46ms/step - los:
0.0440 - accr: 0.9812 - val_loss: 0.1727 - val-accr.: 0.9444
Epoch# 8/30
135/135 [=====] - 6s 46ms/step - los:
0.0386 - accr: 0.9864 - val_loss: 0.2101 - val-accr.: 0.9000
Epoch# 9/30
135/135 [=====] - 6s 46ms/step - los:
0.0112 - accr: 0.9990 - val_loss: 0.2058 - val-accr.: 0.9333
Epoch# 10/30
135/135 [=====] - 6s 46ms/step - los:
0.0904 - accr: 0.9671 - val_loss: 0.2374 - val-accr.: 0.9333

```

```

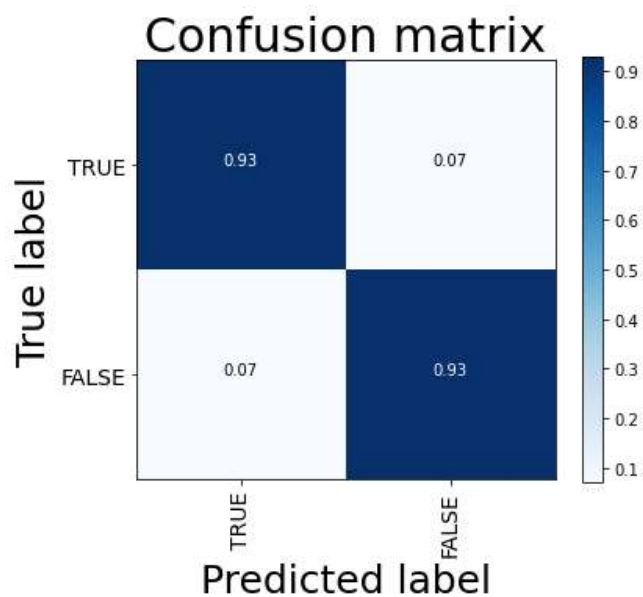
Epoch# 11/30
135/135 [=====] - 6s 45ms/step - los:
0.0281 - accr: 0.9917 - val_loss: 0.4328 - val-accr.: 0.9222
Epoch# 12/30
135/135 [=====] - 6s 46ms/step - los:
0.0184 - accr: 0.9970 - val_loss: 0.4064 - val-accr.: 0.9222
Epoch# 13/30
135/135 [=====] - 6s 47ms/step - los:
0.0214 - accr: 0.9945 - val_loss: 0.4001 - val-accr.: 0.9111
Epoch# 14/30
135/135 [=====] - 6s 45ms/step - los:
0.0246 - accr: 0.9935 - val_loss: 0.3270 - val-accr.: 0.9111
Epoch# 15/30
135/135 [=====] - 6s 46ms/step - los:
0.0075 - accr: 0.9953 - val_loss: 0.4708 - val-accr.: 0.9111
Epoch# 16/30
135/135 [=====] - 6s 45ms/step - los:
0.0018 - accr: 0.9988 - val_loss: 0.4852 - val-accr.: 0.9111
Epoch# 17/30
135/135 [=====] - 6s 46ms/step - los:
0.0017 - accr: 1.0000 - val_loss: 0.6630 - val-accr.: 0.9111
Epoch# 18/30
135/135 [=====] - 6s 46ms/step - los:
0.0024 - accr: 0.9978 - val_loss: 0.6146 - val-accr.: 0.9222
Epoch# 19/30
135/135 [=====] - 6s 46ms/step - los:
0.0032 - accr: 0.9969 - val_loss: 0.7220 - val-accr.: 0.9222
Epoch# 20/30
135/135 [=====] - 6s 45ms/step - los:
0.0024 - accr: 0.9993 - val_loss: 0.7783 - val-accr.: 0.9222
Epoch# 21/30
135/135 [=====] - 6s 46ms/step - los:
2.3000e-04 - accr: 1.0000 - val_loss: 0.8196 - val-accr.:
0.9222
Epoch# 22/30

```

```

135/135 [=====] - 6s 46ms/step - los:
6.8945e-04 - accr: 0.9998 - val_loss: 0.8519 - val-accr.:
0.9111
Epoch# 23/30
135/135 [=====] - 6s 46ms/step - los:
0.0036 - accr: 0.9972 - val_loss: 0.8702 - val-accr.: 0.9111
Epoch# 24/30
135/135 [=====] - 6s 45ms/step - los:
0.0023 - accr: 0.9982 - val_loss: 0.8929 - val-accr.: 0.9111
Epoch# 25/30
135/135 [=====] - 6s 46ms/step - los:
0.0065 - accr: 0.9938 - val_loss: 0.9087 - val-accr.: 0.9222
Epoch# 26/30
135/135 [=====] - 6s 46ms/step - los:
0.0047 - accr: 0.9958 - val_loss: 0.9352 - val-accr.: 0.9111
Epoch# 27/30
135/135 [=====] - 6s 46ms/step - los:
8.1164e-04 - accr: 0.9999 - val_loss: 0.9505 - val-accr.:
0.9111
Epoch# 28/30
135/135 [=====] - 6s 47ms/step - los:
0.0026 - accr: 0.9970 - val_loss: 0.9637 - val-accr.: 0.9222
Epoch# 29/30
135/135 [=====] - 6s 46ms/step - los:
0.0010 - accr: 0.9998 - val_loss: 0.9772 - val-accr.: 0.9222
Epoch# 30/30
135/135 [=====] - 6s 46ms/step - los:
6.8381e-04 - accr: 0.9995 - val_loss: 0.9911 - val-accr.:
0.9222
Accr: 92.89%
overall score: [94.22222375869751, 96.44444584846497,
93.33333373069763, 92.88889169692993]
-----

```



-----Confusion Matrix Stats-----

```
[[ 78   6]
 [ 10 131]]
```

Accr Score : 0.9288888888888889

Report :

	pre	rec	F1_Score	supp.	
0		0.89	0.93	0.91	84
1		0.96	0.93	0.94	141
accr				0.93	225
macro avg		0.92	0.93	0.92	225
weighted avg		0.93	0.93	0.93	225


```

[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]],
[[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0],
[0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0], [0],
[0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[0],
[0], [0], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [0]]]]
vectorized_Ytrain: [0, 1]
vectorized_Ytest: [0, 1]
Model: "sequential_58"

```

Layer (type)	Output Shape	Param #
=====		
time_distributed_232 (TimeDi	(None, None, 82, 2, 128)	3200

time_distributed_233 (TimeDi	(None, None, 41, 1, 128)	393344

time_distributed_234 (TimeDi	(None, None, 21, 1, 128)	393344

time_distributed_235 (TimeDi	(None, None, 2688)	0

lstm_58 (LSTM)	(None, 100)	1115600

dense_116 (Dense)	(None, 256)	25856

drop out_58 (Drop out)	(None, 256)	0

dense_117 (Dense)	(None, 2)	514

```
=====
Total parameters: 1,931,858
Trainable parameters: 1,931,858
Non-trainable parameters: 0
```

None

Epoch# 1/30

135/135 [=====] - 10s 54ms/step - los:
0.5680 - accr: 0.7127 - val_loss: 0.2561 - val-accr.: 0.8667

Epoch# 2/30

135/135 [=====] - 6s 47ms/step - los:
0.2800 - accr: 0.8831 - val_loss: 0.1980 - val-accr.: 0.9111

Epoch# 3/30

135/135 [=====] - 6s 47ms/step - los:
0.2028 - accr: 0.9208 - val_loss: 0.2080 - val-accr.: 0.9222

Epoch# 4/30

135/135 [=====] - 6s 46ms/step - los:
0.1375 - accr: 0.9506 - val_loss: 0.2268 - val-accr.: 0.9222

Epoch# 5/30

135/135 [=====] - 6s 48ms/step - los:
0.1055 - accr: 0.9668 - val_loss: 0.1737 - val-accr.: 0.9444

Epoch# 6/30

135/135 [=====] - 6s 47ms/step - los:
0.0621 - accr: 0.9800 - val_loss: 0.2303 - val-accr.: 0.9333

Epoch# 7/30

135/135 [=====] - 6s 46ms/step - los:
0.0463 - accr: 0.9845 - val_loss: 0.1688 - val-accr.: 0.9444

Epoch# 8/30

135/135 [=====] - 6s 46ms/step - los:
0.0401 - accr: 0.9881 - val_loss: 0.2598 - val-accr.: 0.9222

Epoch# 9/30

135/135 [=====] - 6s 48ms/step - los:
0.0122 - accr: 0.9962 - val_loss: 0.2288 - val-accr.: 0.9111

Epoch# 10/30

135/135 [=====] - 6s 46ms/step - los:
0.0358 - accr: 0.9899 - val_loss: 0.2872 - val-accr.: 0.9111

```

Epoch# 11/30
135/135 [=====] - 6s 46ms/step - los:
0.0229 - accr: 0.9928 - val_loss: 0.2806 - val-accr.: 0.9222
Epoch# 12/30
135/135 [=====] - 6s 47ms/step - los:
0.0034 - accr: 0.9999 - val_loss: 0.7442 - val-accr.: 0.9000
Epoch# 13/30
135/135 [=====] - 6s 46ms/step - los:
0.0402 - accr: 0.9836 - val_loss: 0.4403 - val-accr.: 0.9111
Epoch# 14/30
135/135 [=====] - 6s 46ms/step - los:
0.0106 - accr: 0.9965 - val_loss: 0.3031 - val-accr.: 0.9333
Epoch# 15/30
135/135 [=====] - 6s 47ms/step - los:
0.0323 - accr: 0.9856 - val_loss: 0.1701 - val-accr.: 0.9222
Epoch# 16/30
135/135 [=====] - 6s 46ms/step - los:
0.0337 - accr: 0.9924 - val_loss: 0.3364 - val-accr.: 0.9222
Epoch# 17/30
135/135 [=====] - 6s 46ms/step - los:
0.0052 - accr: 0.9987 - val_loss: 0.4150 - val-accr.: 0.9222
Epoch# 18/30
135/135 [=====] - 6s 47ms/step - los:
0.0011 - accr: 0.9998 - val_loss: 0.4636 - val-accr.: 0.9333
Epoch# 19/30
135/135 [=====] - 6s 46ms/step - los:
0.0024 - accr: 0.9992 - val_loss: 0.5114 - val-accr.: 0.9444
Epoch# 20/30
135/135 [=====] - 6s 47ms/step - los:
0.0053 - accr: 0.9967 - val_loss: 0.5941 - val-accr.: 0.9444
Epoch# 21/30
135/135 [=====] - 6s 46ms/step - los:
7.3541e-04 - accr: 0.9997 - val_loss: 0.5857 - val-accr.:
0.9333
Epoch# 22/30

```

135/135 [=====] - 6s 47ms/step - los:
 0.0026 - accr: 0.9983 - val_loss: 0.6108 - val-accr.: 0.9444
 Epoch# 23/30

135/135 [=====] - 6s 47ms/step - los:
 0.0030 - accr: 0.9994 - val_loss: 0.6302 - val-accr.: 0.9333
 Epoch# 24/30

135/135 [=====] - 6s 47ms/step - los:
 6.0186e-04 - accr: 0.9995 - val_loss: 0.6539 - val-accr.:
 0.9333

Epoch# 25/30

135/135 [=====] - 6s 47ms/step - los:
 0.0012 - accr: 0.9987 - val_loss: 0.6706 - val-accr.: 0.9333
 Epoch# 26/30

135/135 [=====] - 6s 46ms/step - los:
 0.0032 - accr: 0.9974 - val_loss: 0.7004 - val-accr.: 0.9333
 Epoch# 27/30

135/135 [=====] - 6s 46ms/step - los:
 0.0028 - accr: 0.9977 - val_loss: 0.7182 - val-accr.: 0.9333
 Epoch# 28/30

135/135 [=====] - 6s 47ms/step - los:
 0.0042 - accr: 0.9957 - val_loss: 0.7046 - val-accr.: 0.9333
 Epoch# 29/30

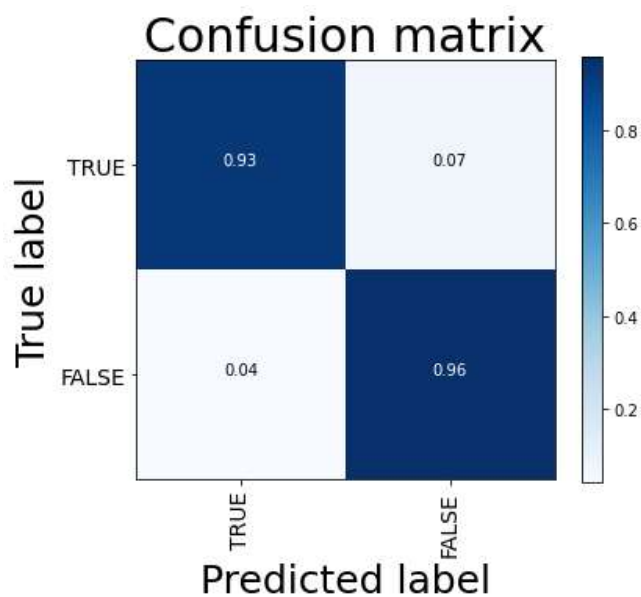
135/135 [=====] - 6s 46ms/step - los:
 2.8472e-04 - accr: 1.0000 - val_loss: 0.7255 - val-accr.:
 0.9333

Epoch# 30/30

135/135 [=====] - 6s 46ms/step - los:
 0.0021 - accr: 0.9981 - val_loss: 0.7223 - val-accr.: 0.9333

Accr: 94.64%

overall score: [94.22222375869751, 96.44444584846497,
 93.33333373069763, 92.88889169692993, 94.64285969734192]



-----Confusion Matrix Stats-----

```
[[ 77   6]
```

```
 [  6 135]]
```

Accr Score : 0.9464285714285714

Report :

	pre	rec	F1_Score	supp.
0	0.93	0.93	0.93	83
1	0.96	0.96	0.96	141
accr			0.95	224
macro avg	0.94	0.94	0.94	224
weighted avg	0.95	0.95	0.95	224