Deep learning model to identify target mRNA of microRNA sequences

Domain background

From a review of currently developed Deep learning models in genomics[1], I realized there is still a long way to go before the full stock of machine learning techniques will be applied to genomics at its maximum potential. In particular, I got fascinated by the deepTarget model that has recently been proposed as a way to identify microRNA targets with 96% accuracy - therefore I decided to focus my project in this field. The repository of the mentioned work is available on GitHib at the following link: https://github.com/ailab-seoultech/deepTarget.

MicroRNAs (miRNAs), which are small non-coding RNA molecules that consist of about 22 nucleotides, are known to regulate more than 60% of protein coding genes of humans and other mammals at the RNA level. As miRNAs control the function of their target messenger RNAs (mRNAs) by regulating the expression of the targets, investigating miRNAs is important to understand various biological processes, including diseases. To predict targets of given miRNAs, numerous computational tools have been proposed[2].

Problem Statement

Two types of computational problems about miRNAs thus naturally arise in bioinformatics: miRNA host identification (i.e., the problem of locating the genes that encode pre-miRNAs) and miRNA target prediction (i.e., the task of finding the mRNA targets of given miRNAs). This project focuses on the target prediction problem, aiming at deploying a model that is able to recognize whether or not a sequence of microRNA belongs to a given target mRNA. The output will be a binary classification of miRNA-mRNA pairs that match/ don't match (i.e. mRNA is a target of the given miRNA).

Analysis

I will utilize experimental negative data collected from a 2020 paper 'Deep Learning-Based microRNA Target Prediction Using Experimental Negative Data'[3] that used the same dataset to train a deepTarget model.

---

[1] Lefteris Koumakis, 2020. Deep learning models in genomics; are we there yet?
[2] Lee Byunghan et al, 2016. deepTarget: End-to-end Learning Framework for microRNA Target Prediction using Deep Recurrent Neural Networks
[3] BYUNGHAN LEE, 2020. Deep Learning-Based microRNA Target Prediction Using Experimental Negative Data

The dataset is available here: https://github.com/ailab-seoultech/deepTarget

The dataset is already split into ‘train’ and ‘test’ with 65,426 and 10,930 data points respectively. The train set has a list of labelled pairs of miRNA and mRNA sequences.

The dataset has been generated using public data obtained from several datasets. It has been balanced by the creators by randomly sampling positive and negative pairs[3].

Data exploration

The dataset provided is pretty simple, including 'fasta' files, a training set and 10 test sets.

Fasta files

.fasta files include IDs and Sequences of all MRNAs and MIRNAs. Below the first of the first 5 mirna IDs and sequence.

```
hsa-miR-4777-5p
UUCUAGAUGAGAGAUAUAUAUA
hsa-miR-3908
GAGCAAUGUAGGUAGACUGUUU
hsa-miR-96-3p
AAUCAUGUGCAGUGCCAAUAUG
hsa-miR-3144-5p
AGGGGACCAAAGAGAUAUAUAG
hsa-miR-6509-3p
UUCCACUGCCACUACCUAAUUU
```

Training Set

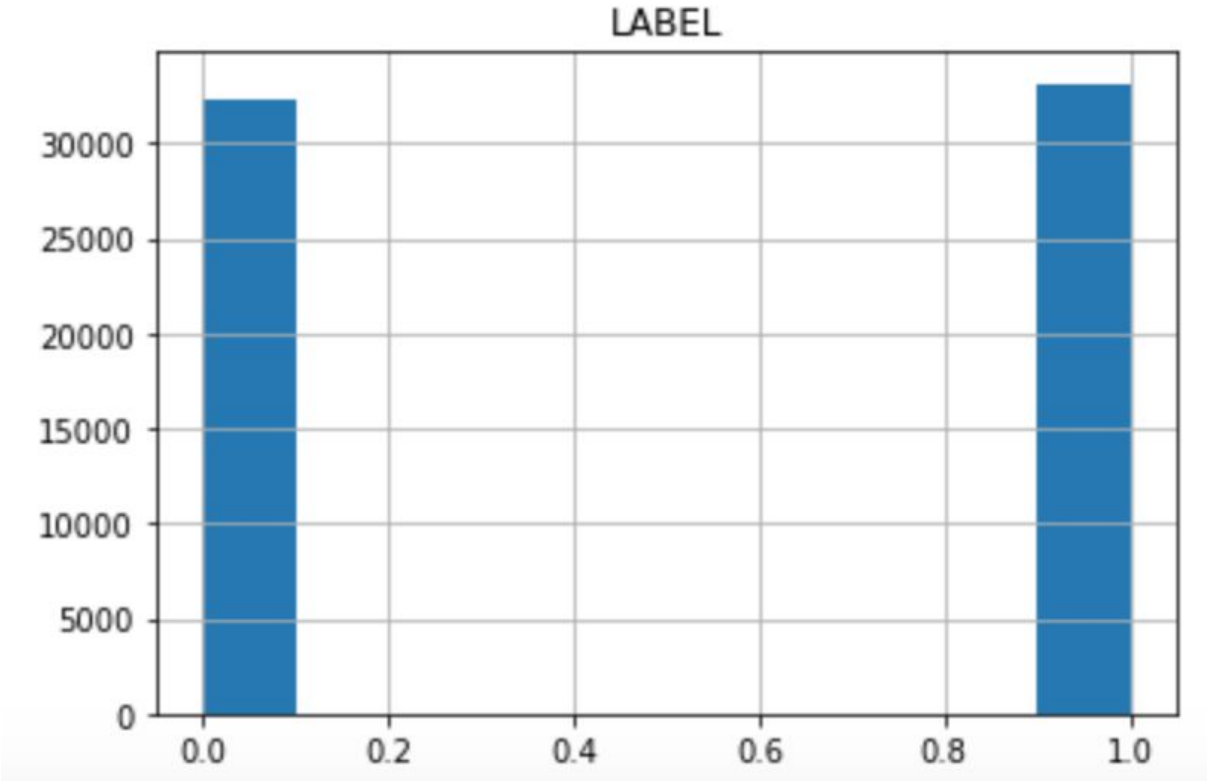
The training set includes:

- MIRNA_ID: identifier of the miRNA sequence (note this is not a unique identifier of the miRNA-mRNA pair because a miRNA can be listed multiple times, paired with multiple mRNA targets)
- ENSEMLBE_GENE and GENE_SYMBOL: identifiers of the gene encoding such miRNA and mRNA sequences. This enables for gene-level predictions of mRNA-miRNA pairs
- MIRNA_SEQ: list of strings with the sequence of basis of the miRNA (e.g. ‘UGAGGU…’)
- MRNA_SEQ: list of strings with the sequence of basis of the mRNA (e.g. ‘GCCTGC…’)
- LABEL: 0 or 1 if the mRNA is a target of the miRNA (1) or it is not (0)

Below a visualization of the training set.

	MIRNA_ID	ENSEMBL_GENE	GENE_SYMBOL	MIRNA_SEQ	MRNA_SEQ	LABEL
0	hsa-let-7a-5p	ENSG00000114573	ATP6V1A	UGAGGUAGUAGGUUGUAUAGUU	GCCTGCTATTGAGGAAAGGTATTCTTCTATACAACTTGTT	1
1	hsa-let-7a-5p	ENSG00000104497	SNX16	UGAGGUAGUAGGUUGUAUAGUU	ATTTGTTTAGTTTCCACGTAATCTTTATCTCTACCTAGAT	1
2	hsa-let-7a-5p	ENSG00000141682	PMAIP1	UGAGGUAGUAGGUUGUAUAGUU	GCACATTGTATATGATTCGGTTTATACATATTACCTTGTT	1
3	hsa-let-7a-5p	ENSG00000174010	KLHL15	UGAGGUAGUAGGUUGUAUAGUU	GAAGTTAGACACCTTTCTGCTAGACAACTTTGTGCCACTC	1
4	hsa-let-7b-5p	ENSG00000130402	ACTN4	UGAGGUAGUAGGUUGUGUGGUU	GGCCCTCATCTTCGACAACAAGCACACCAACTATACCATG	1

The only meaningful other visualization of the dataset is the distribution among ‘0’ and ‘1’, which is balanced as you can see in the picture below.



Test Set

The Test set includes 10 different sets to evaluate model performance. Each include:

- MIRNA_ID: identifier of the miRNA sequence
- MRNA_ID: identifier of the mRNA sequence
- LABEL: 0 or 1 if the mRNA is a target of the miRNA (1) or it is not (0)

Techniques

I have gone through an extensive elaboration of the dataset to preprocess different RNA basis and encode them into binary arrays (find step-by-step details in the Jupyter notebook).

Following the approach proposed by Byunghan (2020), I then exploit one-dimensional CNNs to model miRNAs (x^{mi}) and mRNAs (x^{m}) . Given (x^{mi}, x^{m}) pair where each x is a tuple of 4-dimensional binary vector of length k, the dimension of a binary vector is the number of channels (i.e., depth) of the given pair. After encoding, x is feedforwarded to be N_f feature maps of size $1 \times (k - L_f + 1)$ where the number of filters is N_f , and the size of a filter is $1 \times L_f$.

I then obtain h_{mi} and h_m features of miRNAs and mRNAs, that are then concatenated to generate an h feature - the output of a fully connected layer based on a weighted cross-entropy loss function to alleviate the imbalance problem between the number of target and non-target pairs:

$$Lw(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(w^{1}\, y_i \;\; log\,(p_i)\,) \;+\; (w^{0}(1-\,y_i)\;\; log\,(1-p_i)\,)$$

Then I trained a convolutional neural network to identify pairs of mRNA and miRNA that match and those who don't. I used a CNN-based architecture to learn sequence-based features that contribute to miRNA-mRNA interactions. The model will therefore learn how to classify pairs of miRNA and mRNA sequences that will match '1' versus those who will not match '0'.

Methodology

Data preprocessing

Much of the code of the Jupyter notebook involves the preprocessing of the data to be ready for a neural network. I inserted functions to read the data, encode each RNA basis into arrays and finally encode entire sequences of RNA. Then I developed functions to read the training set and the test set.

Then I defined how matching can happen: miRNAs exhibit their regulatory function by binding to the target sites present in the 3' untranslated region (UTR) of their cognate mRNAs. The seed sequence of a miRNA is defined as the first 2 to 8 nucleotides starting from 5'-end of miRNA. The degree of sequence complementarity formed by Watson-Crick (WC) pairings between the seed and 3' UTR of mRNA sequences determines site types.

Following the paper, I utilize relaxed site patterns, which covers most of the canonical site types (CSTs), non-canonical site types (NSTs), and context-dependent non-canonical site types (CDNSTs, to define the candidate target site (CTS). The details used are as follows:

- 10-mer-m6: six WC pairings from the miRNA nucleotides 1–10

- 10-mer-m7: seven WC pairings from the miRNA nucleotides 1–10
- Offset 9-mer-m7: seven WC pairings from the miRNA nucleotides 2–10

The picture below shows an example of nucleotides matching with an offset 9-mer-m7 pattern.

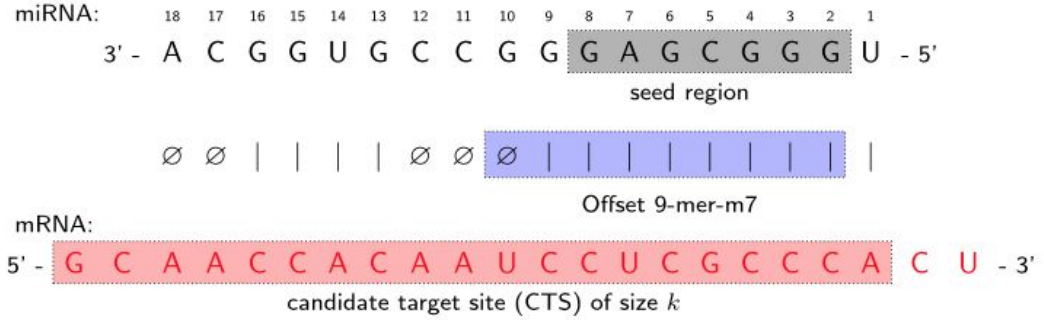


FIGURE 1. Definition of terminologies in interactions between miRNAs and mRNAs (best viewed in color). The gray, blue, and red box represent a seed region, a relaxed site pattern (e.g., Offset 9-mer-m7), and a CTS of size $k = 20$, respectively.

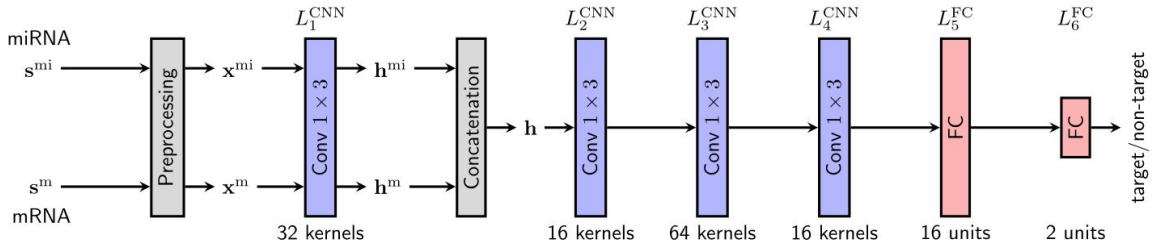
The ‘find_candidate’ function looks for potential matches according to the different seed match requested. Below an extract of the ‘find_candidate’ function. I have trained the model on all the seed_mathces and 10-mer-m6 resulted to be the one driving the best accuracy on the test set.

```
def find_candidate(mirna_sequence, mrna_sequence, seed_match):
   #find potential match with tolerance
[...]
   if seed_match == '10-mer-m6':
       SEED_START = 1
       SEED_END = 10
       SEED_OFFSET = SEED_START - 1
       MIN_MATCH = 6
       TOLERANCE = (SEED_END-SEED_START+1) - MIN_MATCH
   elif seed_match == '10-mer-m7':
       SEED_START = 1
       SEED_END = 10
       SEED_OFFSET = SEED_START - 1
       MIN_MATCH = 7
       TOLERANCE = (SEED_END-SEED_START+1) - MIN_MATCH
   elif seed_match == 'offset-9-mer-m7':
       SEED_START = 2
       SEED_END = 10
       SEED_OFFSET = SEED_START - 1
       MIN_MATCH = 7
       TOLERANCE = (SEED_END-SEED_START+1) - MIN_MATCH
[...]
```

Implementation

I propose an end-to-end machine learning framework for functional miRNA target prediction following the deepTarget paper[2]. I will use the dataset from Byunghan (2016) to test the deepTarget following the paper from 2020[3]. This becomes the benchmark model for my project performance.

The proposed approach exploits one-dimensional convolutional neural networks (CNNs) based on sequence-to-sequence interaction learning framework.
Below an overview of the methodology proposed by the paper[3] that I will emulate and test with different Conv layers configurations.



The concatenated sequence representation h from the L_1 layer is fed to the consecutive L_2 –L_4 layers. The feature maps from the L_4 layer will be flattened and fed to L_5 layer, then the L_6 layer gets logits of the given miRNA-mRNA pair.

For training, I will optimize the weighted cross-entropy loss function using Adam optimizer.

Refinement

I have trained the model with multiple batch_sizes and number of epochs - turned out that 3 epochs are enough for this problem and avoid overfitting. However, increasing the batch size to 64 showed a better accuracy. I also tested different seed-matches and 10-mer-m6 resulted to be the best one to predict miRNA-mRNA targets.

Results

Since the dataset is balanced, the performance of the model against the benchmark will be evaluated using common binary classification performance metrics including accuracy [(TP + TN)/(TP + TN + FP + FN), where TP, FP, FN, and TN represent the numbers of true positives, false positives, false negatives, and true negatives, respectively], sensitivity [TP/(TP + FN)], specificity [TN/(TN + FP)], positive predictive value (PPV)[TP/(TP + FP)] and negative predictive value (NPV) [TN/(TN + FN)]. I will finally benchmark against the paper.
On all the 10 test sets, I achieved an average of 79% accuracy (see below).

set	accuracy	sensitivity	specificity	F-measure	PPV	NPV
set1	0.792714	0.796834	0.788969	0.785436	0.774359	0.810345
set2	0.80661	0.825	0.788969	0.806846	0.789474	0.824561
set3	0.79203	0.795337	0.788969	0.786172	0.777215	0.806373
set4	0.789668	0.790404	0.788969	0.785445	0.780549	0.798544
set5	0.792593	0.796438	0.788969	0.788413	0.780549	0.804401
set6	0.783042	0.776623	0.788969	0.774611	0.77261	0.792771
set7	0.79703	0.805627	0.788969	0.793451	0.781638	0.812346
set8	0.789474	0.790026	0.788969	0.781818	0.773779	0.804401
set9	0.79375	0.798956	0.788969	0.787645	0.77665	0.810345
set10	0.801471	0.814536	0.788969	0.800493	0.786925	0.816377
average	0.793838	0.798978	0.788969	0.789033	0.779375	0.808046

This is slightly higher with the results from the paper (see below).

TABLE 7. Gene-level prediction performance on each test dataset.

	accuracy	sensitivity	specificity	PPV	NPV	F-measure
set 1	0.7729	0.7388	0.8038	0.7735	0.7724	0.7557
set 2	0.7934	0.7825	0.8038	0.7924	0.7943	0.7874
set 3	0.7948	0.7850	0.8038	0.7870	0.8019	0.7860
set 4	0.7690	0.7323	0.8038	0.7796	0.7602	0.7552
set 5	0.7707	0.7354	0.8038	0.7790	0.7636	0.7565
set 6	0.7758	0.7455	0.8038	0.7778	0.7742	0.7613
set 7	0.7787	0.7519	0.8038	0.7819	0.7760	0.7666
set 8	0.7822	0.7585	0.8038	0.7790	0.7850	0.7686
set 9	0.7753	0.7441	0.8038	0.7766	0.7742	0.7600
set 10	0.7907	0.7769	0.8038	0.7908	0.7906	0.7838
average	0.7804	0.7551	0.8038	0.7817	0.7792	0.7681

Sensitivity, PPV and NPV are also slightly improved compared to the model performance. However, specificity is slightly lower (79% vs 80%).

To conclude, this model achieves a slightly higher accuracy compared to the deepTarget paper, while reducing the computational effort and time required for training: it gets to 79% accuracy in just 3 epochs, compared to the 10 epochs in the paper.