Splice sequence detection in DNA based on Chaos Game Representations and Convolutional Neural Networks

Clara Escorihuela and Joana Palés

December 2020

1 Introduction

The DNA is a specific sequence determined by the combination of four nucleotids (Adenine, Timine, Guanine and Citosine), that presents codificant (exon) and non-codificant (intron) regions. These regions are separated by the boundary exon-intron sequences, which are essential for the synthesis of proteins and the signal propagation in metabolic routes.

Numerous papers have proven that severe diseases, such as cancer and muscular distrophy, are caused by the absence of these boundary sequences in the DNA. Moreover, they suggest that tracking their fault will facilitate the early disease's diagnosis.

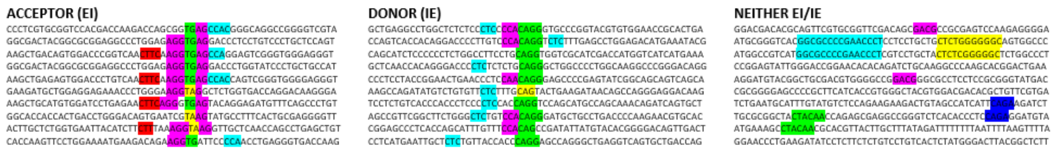
This project aims at detecting the absence of the exon-intron boundary regions in DNA by using the Chaos Game Theory (CGT) to transform a DNA sequence into a 3D image, and a Convolution Neural Network (CNN) to determine the presence or fault of the sequence [1]. The results show that the exposed CNN is able to detect small sequences not heavily modified with high efficiency.

2 Methodology

2.1 Dataset

Two datasets has been used to test the behavior of the algorithm.

1. **Molecular Biology (Splice-junction Gene Sequences) Data Set from UCI** [2]: It corresponds to a sequential domain-theory with 3190 DNA sequence instances of 60 nucleotide bases, where 768 are donor (intron-exon), 769 are acceptor (exon-intron) and 1655 are neither donor or acceptor sequences. Figure 1 presents the first 10 sequences of each group where the common boundary regions are highlighted in different colors. As it is shown all the boundary sequences are very similar but not identical, because some present additional nucleotides.



ACCEPTOR (EI)

```
CCCTCGTGCGGTCCACGACCAAGACCAGCGGTGAGCCACGGGCAGGCCGGGGTCGTA
GGCGACTACGGCGCGGAGGCCCTGGAGAGGTGAGGACCCTCCTGTCCCTGCTCCAGT
AAGCTGACAGTGGACCCGGTCAACTTCAAGGTGAGCCAGGAGTCGGGTGGGAGGGT
GGCGACTACGGCGCGGAGGCCCTGGAGAGGTGAGGACCCTGGTATCCCTGCTGCCAT
AAGCTGAGAGTGGACCCTGTCAACTTCAAGGTGAGCCACCAGTCGGGTGGGGAGGGT
GAAGATGCTGGAGGAGAAACCCTGGGAAGGTAGGCTCTGGTGACCAGGACAAGGGA
AAGCTGCATGTGGATCCTGAGAACTTCAGGGTGAGTACAGGAGATGTTTCAGCCCTGT
GGCACCACCACTGACCTGGGACAGTGAATCGTAAGTATGCCTTTCACTGCGAGGGGTT
ACTTGCTCTGGTGAATTACATCTTCTTTAAAGGTAAGGTTGCTCAACCAGCCTGAGCTGT
CACCAAGTTCCTGGAAAATGAAGACAGAAGGTGATTCCCCAACCTGAGGGTGACCAAG
```

DONOR (IE)

```
GCTGAGGCCTGGCTCTCTCCCTCCCCACAGGGTGCCCGGTACGTGTGGAACCGCACTGA
CCAGTCACCACAGGACCCCTTGTCCCACAGGTCTCTTTGAGCCTGGAGACATGAAATACG
CAGCATCTCCCCCCTCTGGCCTTCCTGCAGGTGGTCGCATCGACCATGGTCATCATGAAA
GCTCAACCACAGGGACCCCTCTCTCTGCAGGGCTGGCCCCTGGCAAGGCCCGGGACAGG
CCCTCCTACCGGAACTGAACCCTCCAACAGGGAGCCCCGAGTATCGGCAGCAGTCAGCA
AAGCCAGATATGTCTGTGTTCTCTTTGCAGTACTGAAGATAACAGCCAGGGAGGACAAG
TCCTCTGTCACCCACCCTCCCCTCCACCAGGTCCAGCATGCCAGCAAACAGATCAGTGCT
AGCCGTTCGGCTTCTGGGCTCTGTCCACAGGGATGCTGCCTGACCCCAAGAACGTGCAC
CGGAGCCCTCACCAGCATTTGTTTCCACAGCCGATATTATGTACACGGGGACAGTTGACT
CCTCATGAATTGCTCTCTGTTACCACCCAGGAGCCAGGGCTGAGGTCAGTGCTGACCAG
```

NEITHER EI/IE

```
GGACGACACGCAGTTCGTGCGGTTCGACAGCGACGCCGCGAGTCCAAGAGGGGA
ATGCGGGTCACGGCGCCCCGAACCCTCCTCCTGCTGCTCTGGGGGGCAGTGGCCC
ATGGCCGTCATGGCGCCCCGAACCCTCGTCCTGCTACTCTCGGGGGCTCTGGCCCT
CCGGAGTATTGGGACCGGAACACACAGATCTGCAAGGCCCAAGCACGGACTGAA
AGGATGTACGGCTGCGACGTGGGGCCGGACGGGCGCCTCCTCCGCGGGTATGAC
CGCGGGGAGCCCCGCTTCATCACCGTGGGCTACGTGGACGACACGCTGTTCGTGA
TCTGAATGCATTTGTATGTCTCCAGAAGAAGACTGTAGCCATCATTCAGAAGATCT
TGCGCGGCTACTACAACCAGAGCGAGGCCGGGTCTCACACCCTCCAGAGGATGTA
ATGAAAGCCTACAACGCACGTTACTTGCTTTATAGATTTTTTTTAATTTTAAGTTTTA
GGAACCCTGAAGATATCCTCTTCTCTGTCCTGTCACTCTATGGGACTTACGGCTCTT
```

Figure 1: DNA samples with common boundary sequences highlighted for acceptor, donor and neither groups. Acceptors present AGGTGAG and donors have CAGG as common boundary sequences in almost all the DNA strings, while neither IE/EI present any common sequence.

2. **Molecular Biology (Simulated/fake dataset):** A simulation to the above dataset has been created to test the algorithm behaviour. The different parameters that can be adapted to modify the dataset are: the number of sequences, the sequence's length, the common sequence (boundary sequence), and the number and position of the added nucleotides in the boundary sequence.

2.2 Chaos Game

The Chaos Game allows to represent a large sequence (DNA in this case) into a graphical frame thanks to the fractal geometry theory [3]. The main idea is to represent the abundance of all k-mers (inner sequences of length k) in a given larger sequence. Firstly, we determine all the possible k-mers given a specific k integer, secondly we calculate their appearance probability and finally we locate them in a frequency matrix table (FMT) as figure 2 shows. Consequently, the value of each square (pixel in an image) will depend on the abundance of the corresponding k-oligo, and the number of pixels will be determined by the k-value (4^k).

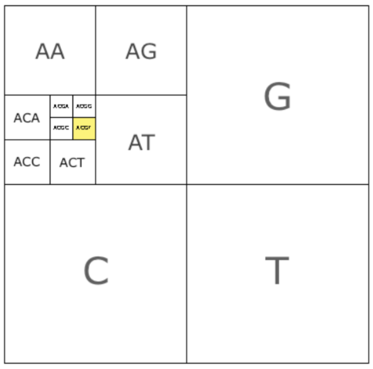


Figure 2: FMT for K-oligos

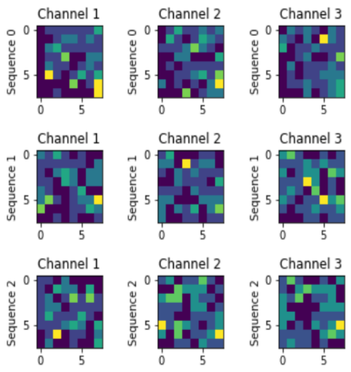


Figure 3: FMT for sequences 0, 1 and 2 and k = 3.

The first step has been to transform all the DNA sequences into FMTs. However, as mentioned before, the boundary regions are not identical in the acceptor and donor sequences because they include different nucleotide in diverse positions. Therefore, in order to find common sequences with little differences we propose to transform each ADN into three different FMT, each one considering different k-oligo elements: The first one including all the consecutive nucleotides (A***GGTG***AG) the second without considering the second nucleotide (A***G***G***TGA***G), and the third without considering the second and the third nucleotide (A***G***GT***GAG***). Finally, each image will present $(2^k, 2^k, 3)$ dimensions.

2.3 Convolutional Neural Networks

As stated before, the input images of the CNN will be of sizes $(2^k, 2^k, 3)$. Since the size of the image is small (k usually around 4) and the size of the dataset is also quite small, to avoid overfitting issues, the chosen CNN is simple (figure 4). It consists of two convolutional layers with number of filters 64 and 128 respectively, kernel size of 2x2, without stride, with padding to conserve the image sizes and activation function SeLU. Besides, a Dropout layer of 20% and Batch Normalization is placed after each convolutional layer. A maximum pooling layer of size 2x2 are placed in-between the two convolutional layers. Finally, a Dense layer with one neuron is placed on top with a Sigmoid activation function to classify the images. The weight initializer of all the layers is the default by Keras, the Glorot uniform. In order to select these hyperparameters, the technique of 5-fold-cross-validation has been used.

The training of the network has been done with batch size of 32 in an effort to obtain high accuracy, Adam as the optimizer and Binary Cross Entropy as the loss function. Two callbacks have been used to avoid overfitting and save training time: early stopping with patience of 4 and minimum delta of 0.01 and model checkpoint at the end of each epoch.

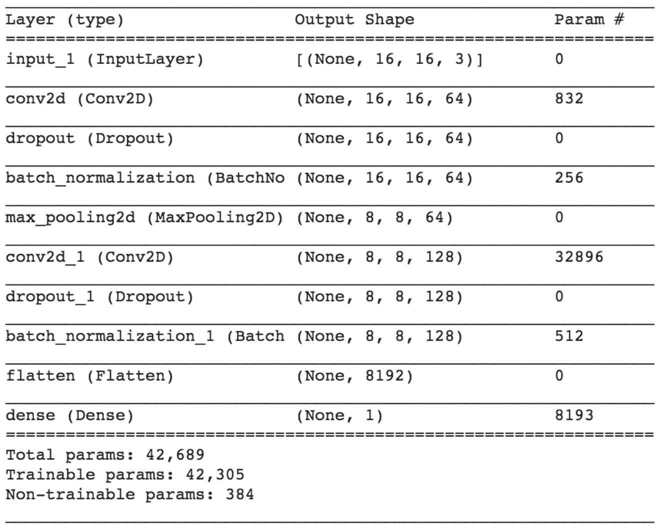

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 16, 16, 3)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 16, 16, 64)        832
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 64)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 8, 8, 128)         32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
batch_normalization_1 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 8193
=================================================================
Total params: 42,689
Trainable params: 42,305
Non-trainable params: 384
_________________________________________________________________
```

Figure 4: Structure of the CNN.

3 Results

Table 1 presents the performance metrics of the model trained and tested with the fake dataset with 3000 sequences, where 1500 include the boundary GGTGAG sequence of the donor EI group from the UCI dataset in three different ways: without any modificiation, and with 1 or x random additional nucleotids in aleatory positions. Table 1 and figure 5 show the performance using the UCI database, which contains approximately 1700 sequences per type (acceptor and donor).

	k = 3					k = 4				
	Acc	Spec	Prec	Rec	F1	Acc	Spec	Prec	Rec	F1
GGTGAG	87	89	89	85	87	95	98	98	92	95
GGTGAG + 1 random base	76	82	80	71	75	80	78	79	81	80
GGTGAG + x random bases	77	85	83	69	75	81	89	87	73	79

Table 1: Performance of the CNN on the simulated data, in percentage.

	k = 4				
	Acc	Spec	Prec	Rec	F1
Exon-Intron seq.	74	60	69	89	78
Intron-Exon seq.	76	80	78	72	75

Table 2: Performance of the CNN on the real data, in percentage.

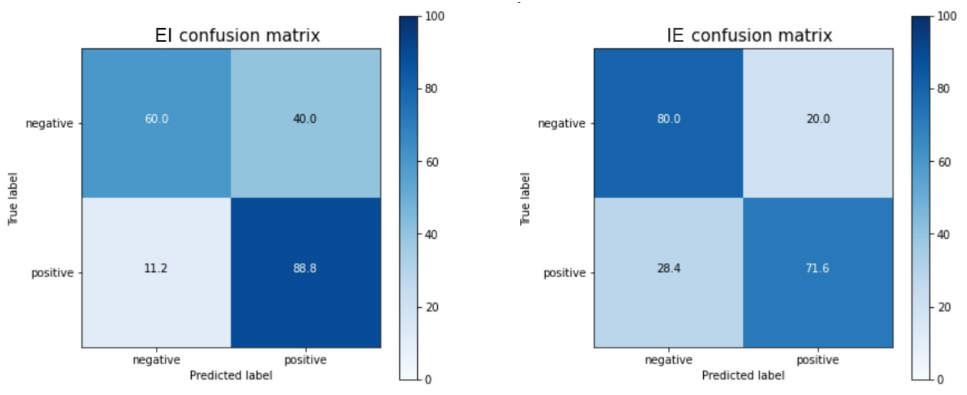


Figure 5: Confusion matrices for real data detecting acceptor sequences (left) and donor sequences (right).

4 Conclusion

We presented a CNN architecture able to detect common sequences in a DNA string by converting the DNA sequence into a 3D image thanks to the Chaos Game. This structure is extremely efficient to detect non-variant common sequences. Besides, thanks to the three channels image implementation, the detection of common sequences with variations is still accurate and creates a good research line to continue working on.

References

[1] R. Rizzo, A. Fiannaca, M. La Rosa, and A. Urso, "Classification experiments of dna sequences by using a deep neural network and chaos game representation," in *Proceedings of the 17th International Conference on Computer Systems and Technologies 2016*, pp. 222–228, 2016.

[2] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[3] H. J. Jeffrey, "Chaos game representation of gene structure," *Nucleic acids research*, vol. 18, no. 8, pp. 2163–2170, 1990.