# Binary Classification of Genomic Signals and Regions using CNNs

Ahmed El-farra
*Department of Applied Mathematics*
*Western University*
aelfarr2@uwo.ca

Ahmed Faid
*Department of Computer Science*
*Western University*
afaid@uwo.ca

***Abstract*—Historically geneticists have focused their attention on only 2% percent of the human genome, known as coding sequences. The reason why, is that this part codes for amino acids and proteins which are the building blocks of our cells. However, with new evidence suggesting possible undiscovered functionality, many scientists are beginning to research this remaining 98% known as the non-coding sequences. Through the use of convoluted neural networks (CNNs) and by converting these genomic sequences to spectrogram images, we created 3 models to solve the binary classification problem: 'Does this spectrogram correspond to a protein coding region?'. The evaluation of our best performing CNN model achieves a 98% accuracy and F1-score of 0.98.**

## I. Background

In molecular biology, a genome is defined as the complete set of genetic information of an organism. A genome contains all the instructions necessary for an organism to maintain and propagate life [1]. A copy of this information is stored in every cell of the organism. The genome is composed primarily of DNA and is a relatively large molecule strung by a series of nucleotides arranged in a particular manner. When studying the genome, geneticists divide the genome into much smaller subsections known as genes. The main function of these genes is the production of RNA and proteins necessary for the cell's functions and for the organism's survival [2]. Genes which produce these proteins are known as "coding sequences" or "coding genes".

As mentioned previously, a coding sequence is a section of DNA which contains the instructions for making an RNA molecule or a protein [3]. Proteins are made of chains of amino acids. The order in which the amino acids are pieced together is encoded within the gene. To construct a protein, the gene creates a messenger RNA which contains information to help make the protein in a process called transcription. The messenger RNA is then read by a ribosome which allows it to assemble the protein from amino acids in a process called Transcription [4]. Despite the construction of proteins being the main function of the genome, most genes in the genome cannot produce proteins. These genes are known as non-coding genes [5]. For a long period of time these non-coding genes were considered to be nothing more than "noise" earning them the nickname Junk DNA. However, a recent research study by the name of the ENCODE project showed that many of these non-coding sequences (around 80%) are in fact biochemically functional [6].

Since then technological advances over the decade have begun to unravel the remarkable complexity of these genes. Several methods and tools have been used to study both coding and non coding genes, one of which is Convoluted Neural Networks. A Convolutional Neural Network (ConvNet/CNN) is an example of a Deep Learning algorithm which takes in an input image, assign learnable weights and biases to several aspects/parts of the image which allows it to differentiate one class of images from another [7]. CNN aims to automatically learn the spatial hierarchy of elements by backpropagating. The funadmental units of convolutional neural networks are the layers, each CNN consist of either a convolutional, pooling, fully connected layers and many more layers [8]. CNNs have been used extensively in medical applications such as the diagnosis of pneumonia through X-Ray images and the classification of possible stroke victims (see related works section for more detail). For our application we will be using CNNs to classify whether a gene is a coding or a non coding sequence. This will be achieved by representing each gene as a spectrogram and feeding them to the CNN for classification.

## II. Related Work

CNNs have been widely used in medical applications as a means of analyzing/predicting various illnesses and diseases. Some of these applications include the detection and subtype classification of a stroke based on the CT scan of stroke victim [9]. Another application of CNNs in the medical field includes the use of X-Ray images to diagnose and identify patients suffering with pneumonia [10].

CNNs are also popular in the field of genetics as there have been several research studies done to create CNN architectures that can identify and analyze different aspects of genetic sequences. Our project aims to find a accurate and relatively efficient CNN architecture to classify nucleotide sequences as coding/noncoding. Our aim is to build on existing literature/research on the problem such as Gonzalo Pajares paper on genomic sequence classification [11].

## III. Introduction

A genome holds a vast amount information relating to development, physiology, and evolution of a species. Jump ahead to today, and the nucleotide sequences of over two hundred and six thousand different species are publicly available in the GenBank database. It is a good time to be in Genomics. Historically geneticists have focused their attention to only two percent of the human genome, known as coding sequences. The reason why, is that this part codes for amino acids and proteins – the building blocks of our cells. However, as of recent their has been a growing body of research around the other 98%, known as non-coding sequence [5].

Bioinformaticians and Geneticist have been using several tools and algorithms to analyze/study different coding sequences. One example of these tools is Deep learning. Deep learning algorithms use a neural network to find associations between a set of inputs and outputs. These algorithms recognize relationships between vast amounts of data. For our project, our goal was to build a classification model that allowed us to differentiate with a high degree of accuracy between a coding sequence and a noncoding sequence. We chose to use a convoluted neural network (CNN) to build our model since the data on each gene is represented as a spectrogram. CNNs are very efficient at processing image data due to the fact that CNNs make the explicit assumption that the inputs it is receiving are images. This assumption makes the forward function more efficient to implement and allows them to vastly reduce the number of parameters in the network.

By testing and comparing multiple CNN architectures, we were able to achieve an accuracy rate as high as 98% with as few as 550,000 parameters which is a better performance than similar work done around the same problem.

## IV. Methods

### A. Research Objective

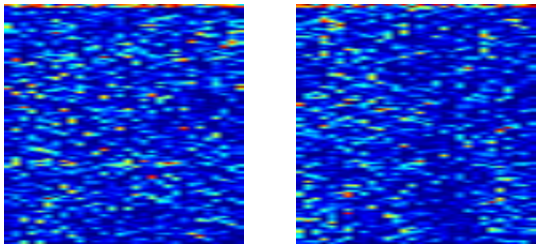
In this paper, we propose a novel method to build a binary classification model using genomics signals as input. The model will distinguish between two different classes coding (CDN) and non-coding (LNSD). This paper also experiments with shallow and deep models using three different CNN architectures to evaluate if a complex or simple model can perform as well as each other.

### B. Hypothesis

Our hypothesis is that the more complex the model is and the more trainable parameter the model has the higher the accuracy of classifying each region of the nucleotide sequence and the faster the model stabilizes and reaches equilibrium.

### C. Data

This section describes the spectrogram images being used to classify between coding and non-coding regions. The dimensions of the images are 224 x 224 with three channels (RGB) with resolution of $0.489\mu$m/pixel. These images were provided by Gerardo Mendizabal-Ruiz a researcher/bioinformatician at the Universidad de Guadalajara [11]. The sample size of our dataset is 2000 images, with a balanced dataset of 1000 images of each class. We use 800 images of each class for our training set and 200 images of each class for our testing set.



(a) Coding Region (CDN) (b) Non-Coding Region (LNSD)

Fig. 1: Spectrogram of both classes CDN and LNSD using genomic signals

### D. Data Preprocessing

In order to analyze performance implications we pre-process the spectrograms. The data is normalized to have values between 0 and 1 and then standardized to be uniformly distributed around the mean 0.485 $\mu$. The images are scaled to have identical widths and heights before feeding it into a CNN architecture. The data is noisy, and in order to denoise the spectrogram image we apply a Gaussian blur to reduce the image noise and also to enhance the image structures at different scales.

The data is categorized using one hot encoding. The dataset is made into a binary vector that has one element for each unique label and marking the class label with a 1 and all other elements 0. Each label in the dataset would be replaced with a vector (one column becomes two) [12].

### E. Performance Metrics

To evaluate the results of the experiment, many performance metrics will be used. The accuracy metric will be used to determine how well the model is performing in every epoch on both the training and testing set. Precision, recall and F1-score will be use to measure each CNN architecture accuracy of the dataset. Also to further evaluate the models accuracy a auc-roc curve will be plotted and the results will be tabulated using a confusion matrix [13]:

| | | Predicted | | |
|---|---|---|---|---|
| | | CDN | LNSD | Total |
| Predicted | CDN | $TP$ | $FN$ | $TP+FN$ |
| | LNSD | $FP$ | $TN$ | $FP+TN$ |
| | Total | $TP+FP$ | $FN+TN$ | $N$ |

### F. CNN Architecture Design - Base Model

This is the base model we propose to deconstruct to see if we can construct a more simple CNN architecture. The base model consists of 7 convolutional layers, 2 max pooling layers,1 average pooling layer, 4 fully connected layers and softmax as its activation function. The model consists of $744,620$ parameters as shown in figure 2.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 222, 222]             896
            Conv2d-2          [-1, 64, 220, 220]          18,496
         MaxPool2d-3            [-1, 64, 73, 73]               0
            Conv2d-4            [-1, 80, 73, 73]           5,200
            Conv2d-5          [-1, 192, 71, 71]         138,432
         MaxPool2d-6          [-1, 192, 23, 23]               0
            Conv2d-7            [-1, 64, 21, 21]         110,656
            Conv2d-8            [-1, 90, 19, 19]          51,930
            Conv2d-9          [-1, 192, 17, 17]         155,712
        MaxPool2d-10            [-1, 192, 5, 5]               0
        AvgPool2d-11            [-1, 192, 1, 1]               0
           Linear-12                  [-1, 512]          98,816
           Linear-13                  [-1, 256]         131,328
           Linear-14                  [-1, 128]          32,896
           Linear-15                    [-1, 2]             258
          Softmax-16                    [-1, 2]               0
================================================================
Total params: 744,620
Trainable params: 744,620
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 50.61
Params size (MB): 2.84
Estimated Total Size (MB): 54.03
----------------------------------------------------------------
```

Fig. 2: CNN Architecture with 7 Convolutional Layers

## V. Results

We used a set of 2000 spectrogram images with a $80\%$ training data and $20\%$ testing data provided by Gerardo Mendizabal-Ruiz a research at the Universidad de Guadalajara [11]. The network was trained using a stochastic gradient descent optimizer with a learning rate $3x10^{-3}$ and a binary cross entropy loss function. We used $125$ epochs to evaluate each with model to check where the models are most stable and reach equilibrium and a batch size of $50$ images. To evaluate the trained models, we used a test set with $400$ spectrogram images that are used for prediction, and split in two classes (coding and non-coding) with both classes being balanced. We use several different performance metrics such as accuracy, precision, recall, and F1-score to evaluate our models performance. The code was implemented using PyTorch to implement the architectures and sklearn to evaluate key performance metrics. The training and testing set are both preprocessed in the same manner to avoid any inaccuracies. We used the same hyperparamters throughout each model to solely focus on the architectures performance and test how each model would behave when being deconstructed into a more simple model.

### A. CNN Architectures - Model 1

Our base model consists of 7 convolutional layers, 2 max pooling layers, 1 average pooling layer, 4 fully connected layers and softmax as its activation function. This model obtains a accuracy of $95\%$, F1-score of $0.96$ for the coding region(CDN) and $0.95$ for the non-coding region(LNSD), and an AUC that is $0.98$. The models performance stabilizes and reaches equilibrium around $60$ epochs which is shown in figure 2 and 3. The model has the most memory with $744,620$ total parameters. To avoid overfitting, L2 regularization determined on $0.0001$. Vanishing gradient while training the model is handled by ReLU for each convolutional layer. The momentum is given to be $0.9$ to avoid local minimal and training to converge faster.

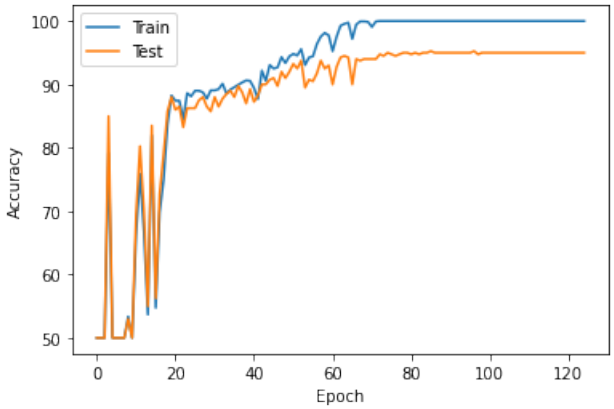



Fig. 3: Accuracy with Epochs on Training and Testing Set.

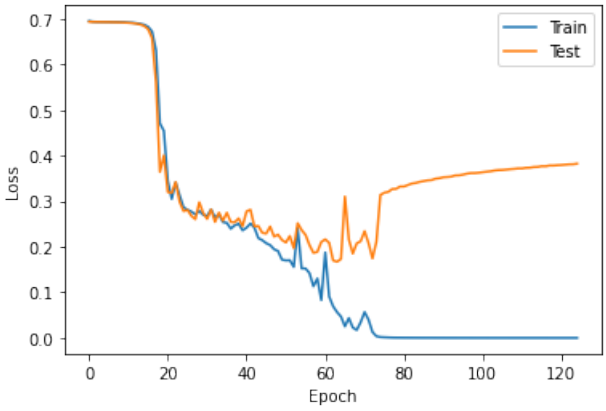



Fig. 4: Loss with Epochs on Training and Testing Set.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| CDN | 0.94 | 0.97 | 0.96 | 200 |
| LNSD | 0.97 | 0.94 | 0.95 | 200 |
| accuracy | | | 0.95 | 400 |
| macro avg | 0.96 | 0.96 | 0.95 | 400 |
| weighted avg | 0.96 | 0.95 | 0.95 | 400 |

Fig. 5: Precision, Recall, F1-Score of Model with 7 Convolutional Layers

### B. CNN Architectures - Model 2

The second CNN Architecture is the best performing model. It consists of $5$ convolutional layers, $2$ max pooling layers, $1$

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| CDN | 0.99 | 0.97 | 0.98 | 200 |
| LNSD | 0.98 | 0.99 | 0.98 | 200 |
| accuracy | | | 0.98 | 400 |
| macro avg | 0.98 | 0.98 | 0.98 | 400 |
| weighted avg | 0.98 | 0.98 | 0.98 | 400 |

Fig. 6: Precision, Recall, F1-Score of Model with 5 Convolutional Layers

average pooling layer, 3 fully connected layers and softmax as its activation function. This model obtains a accuracy of $98\%$ accuracy, F1-score of $0.97$ for the coding region(CDN) and $0.98$ for the non-coding region(LNSD), and an AUC that is $0.99$. The models performance stabilizes and reaches equilibrium around $60$ epochs which is shown in figure 7 and 8. The model has the most memory with $569,746$ total parameters.

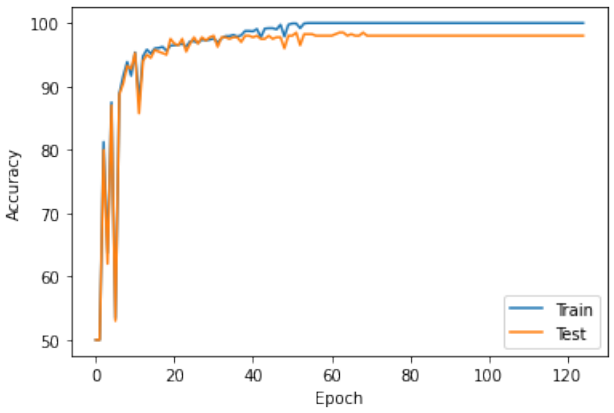



Fig. 7: Accuracy with Epochs on Training and Testing Set.

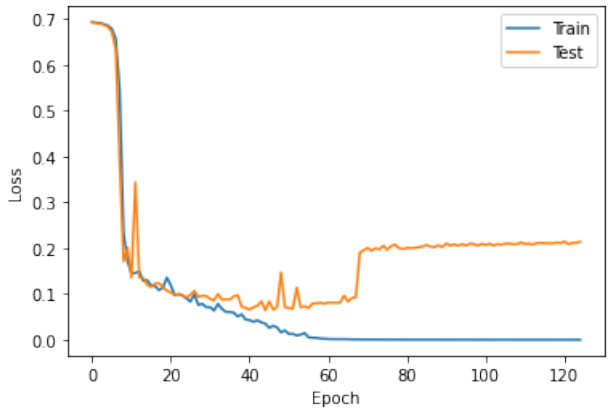



Fig. 8: Loss with Epochs on Training and Testing Set.

### C. CNN Architectures - Model 3

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| CDN | 0.50 | 1.00 | 0.67 | 200 |
| LNSD | 1.00 | 0.01 | 0.01 | 200 |
| accuracy | | | 0.50 | 400 |
| macro avg | 0.75 | 0.50 | 0.34 | 400 |
| weighted avg | 0.75 | 0.50 | 0.34 | 400 |

Fig. 9: Precision, Recall, F1-Score of Model with 3 Convolutional Layers

The third CNN Architecture consists of 3 convolutional layers, 1 max pooling layers, 1 average pooling layer, 2 fully connected layers and softmax as its activation function. This model obtains a accuracy of $95\%$ accuracy, F1-score of $0.67$ for the coding region(CDN) and $0.01$ for the non-coding region(LNSD), and an AUC that is $0.65$. The models performance stabilizes and reaches equilibrium around $60$ epochs which is shown in figure 10 and 11. The model has the most memory with $269,174$ total parameters. This is the worse performing model and has the simplest complexity of each models.

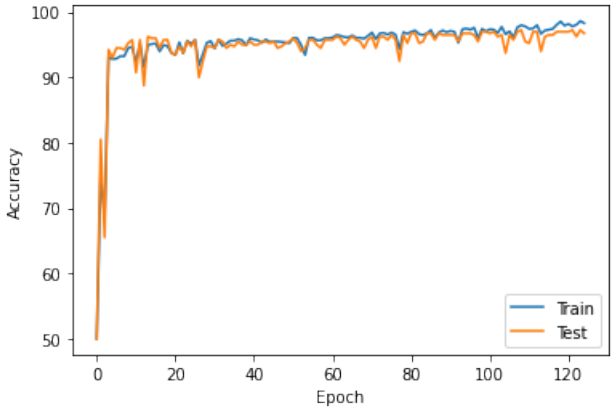



Fig. 10: Accuracy with Epochs on Training and Testing Set.

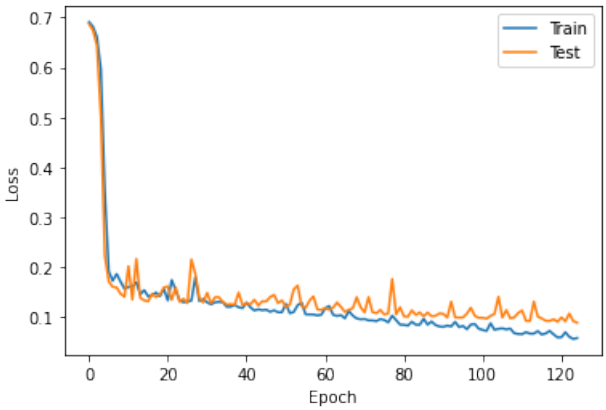
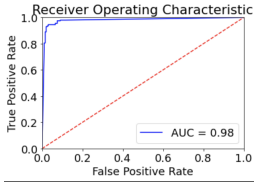


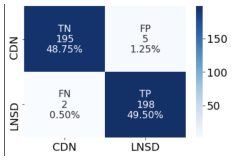

Fig. 11: Loss with Epochs on Training and Testing Set.

## VI. Conclusion

The spectrogram images have been successfully classified using all three CNN architectures with high accuracy. In this research, we concluded that pre-processing the data did not give us a better accuracy. After comparing all three CNN architectures our hypothesis was rejected. Our base model, the most complex model was out performed with a less complex model which only contains 5 convolutional layers. The model provided us with a $98\%$ accuracy for classifying coding regions and non-coding regions of nucleotide sequences with only $569,746$ parameters. The figures below show the AUC-ROC curve and the confusion matrix of this model.




(a) AUC - ROC curve




(b) Confusion Matrix of Model 2

Using the CNN models in this paper can provide researchers with a tool for genome annotation. Building upon the CNN architectures discussed, a better more accurate model can be built by adding more layers to the CNN or by using inception models. Using a bigger dataset can allow future researchers to improve the model accuracy and get a more representative measurement of its performance.

## References


[1] YourGenome ogranization. What is a genome? 2017.

[2] National Human Genome Research Institute. A brief guide to genomics. 2014.

[3] Coding sequence.

[4] Nature Publishing Group.

[5] What is noncoding dna?: Medlineplus genetics. U.S. National Library of Medicine, Jan 2021.

[6] Casey Luskin, September 5, and Casey LuskinAssociate Director. Junk no more: Encode project nature paper finds "biochemical functions for 80% of the genome". Apr 2017.

[7] Sumit Saha. A comprehensive guide to convolutional neural networks.

[8] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology, Jun 2018.

[9] Calvachi and P. Stroke detection and subtype classification using convolutional neural networks (cnns). iMedPub, Mar 2020.

[10] Lucky Agarwal Rahul Nijhawan Ankush Mittal Dimpy Varshni, Kartik Thakral. Pneumonia detection using cnn based feature extraction.

[11] J. Alejandro Morales, Román Saldaña, Manuel H. Santana-Castolo, Carlos E. Torres-Cerna, Ernesto Borrayo, Adriana P. Mendizabal-Ruiz, Hugo A. Vélez-Pérez, and Gerardo Mendizabal-Ruiz. Deep learning for the classification of genomic signals. Hindawi, May 2020.

[12] Jason Brownlee. Why one-hot encode data in machine learning? Jun 2020.

[13] Jason Brownlee. How to calculate precision, recall, and f-measure for imbalanced classification. Aug 2020.