Classification of Exon and Intron Boundaries Final Paper


## Abstract

The goal was to accurately classify exon and intron boundaries based on DNA sequences. Scientists can learn more about proteins if divisions between exons and introns are clear. We used multiple different machine learning approaches including a logistic regression model, a multilayer perceptron, a LSTM, and a model that included a LSTM, autoencoder and a multilayer perceptron. LSTMs performed well, pointing to the idea that order of nucleotides is important when classifying DNA sequences.

## Introduction

Identifying boundaries between exons and introns is crucial for understanding how proteins are made. Exons are segments of DNA that code for proteins, while introns are segments of DNA that do not code for proteins. Proteins have a wide array of functions like improving immune health, and helping the body to recover from injury. For this reason being able to identify parts of DNA that code for proteins is important because it tells scientists the exact sequence of DNA that is used to create a specific protein. This information can later be used to better understand genetic disorders that result from mutation. The research task is supervised and aimed at classifying DNA sequences. The data is numerical and the outputs are labels.

## Background

There are many different approaches to classifying DNA sequences. A newer approach is using a convolutional multilayer perceptron to accurately classify sequences. A Kanazawa University research group saw great success in using convolutional multilayer perceptrons to classify a variety of DNA sequence datasets. They were able to achieve improved accuracy compared to previous models across all 12 datasets they used. Amongst machine learning approaches one hot encoding has become a popular way of numerically representing DNA sequences. In one hot encoding each letter or grouping of letters is represented by a vector of zeros and a singular one. The position of the one in the vector denotes the type letter or grouping of letters that the vector represents. While this is a good way of numerically representing DNA sequences it can be computational expensive. Other non-machine learning approaches have been used to classify DNA sequences, such as characterizing based on regularity of nucleotide strings.

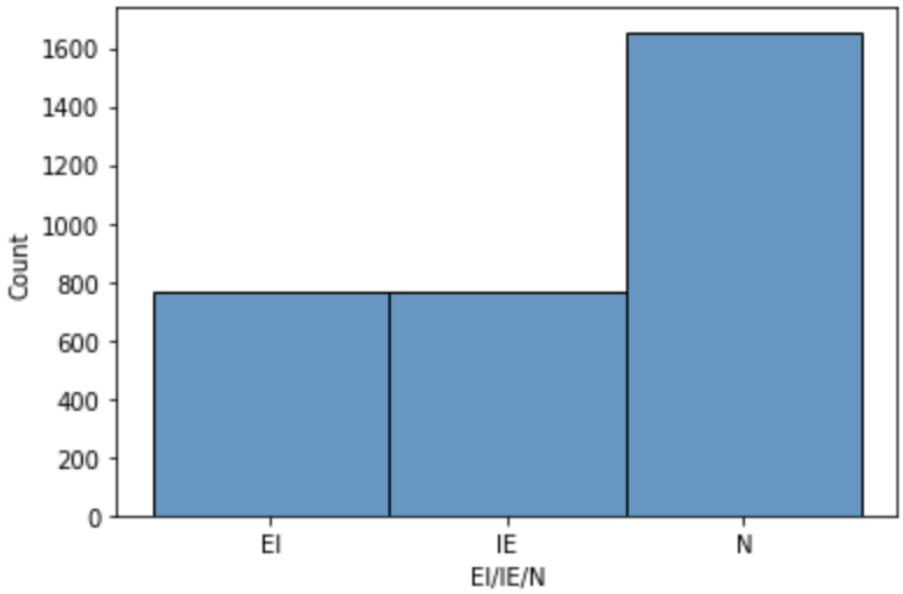
This approach represents DNA sequences as matrices which can then be analyzed to determine "structural characteristics" of the sequences.

**Dataset**

We used the splice junction gene sequences data set from the UCI machine learning database for this research task. It includes 3,190 DNA sequences, 639 of them were not used. Each DNA sequence is 60 nucleotides (units that make up DNA) long and is labeled an intron to exon boundary (IE), an exon to intron boundary (EI), or neither (N). The dataset contains 1350 N, 614 IE and 587 EI.

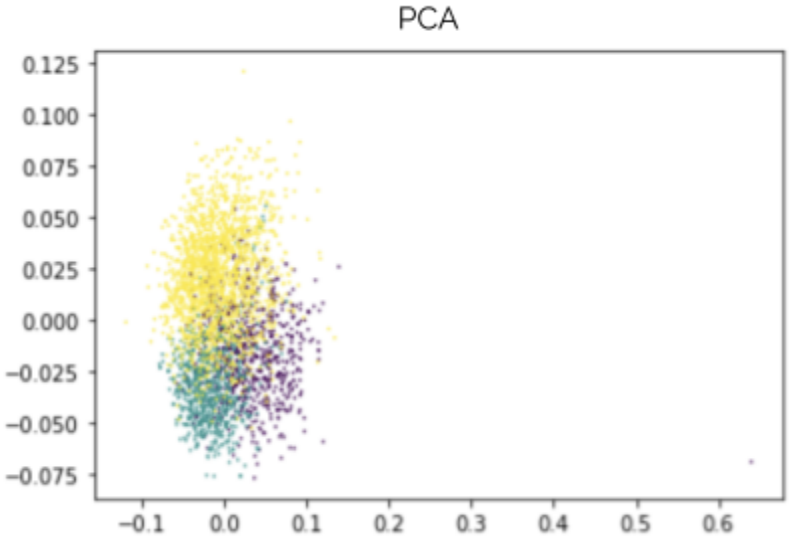
| | CLASS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | EI | 0.1260 | 0.0806 | 0.1260 | 0.1340 | 0.1340 | 0.1340 | 0.0806 | 0.1340 | 0.1340 | 0.0806 | ... | 0.0806 | 0.1335 | 0.0806 | 0.1340 | 0.1340 | 0.1340 | 0.1340 | 0.1340 | 0.0806 | 0.1340 |
| 1 | EI | 0.0806 | 0.1260 | 0.0806 | 0.0806 | 0.1335 | 0.0806 | 0.1260 | 0.1260 | 0.0806 | 0.0806 | ... | 0.1340 | 0.1260 | 0.1340 | 0.0806 | 0.0806 | 0.0806 | 0.0806 | 0.1260 | 0.1335 | 0.0806 |
| 2 | EI | 0.0806 | 0.0806 | 0.0806 | 0.1340 | 0.1335 | 0.0806 | 0.1340 | 0.0806 | 0.1335 | 0.1335 | ... | 0.0806 | 0.0806 | 0.1335 | 0.1335 | 0.1335 | 0.1335 | 0.1340 | 0.1340 | 0.1340 | 0.1340 |
| 3 | EI | 0.0806 | 0.1340 | 0.1335 | 0.1340 | 0.1260 | 0.0806 | 0.1340 | 0.1340 | 0.1340 | 0.1340 | ... | 0.1340 | 0.1340 | 0.1335 | 0.1335 | 0.0806 | 0.1260 | 0.1340 | 0.1340 | 0.1340 | 0.1335 |
| 4 | EI | 0.1340 | 0.1260 | 0.0806 | 0.1260 | 0.1340 | 0.1335 | 0.0806 | 0.0806 | 0.0806 | 0.1335 | ... | 0.0806 | 0.1260 | 0.0806 | 0.1260 | 0.1340 | 0.1340 | 0.1260 | 0.1340 | 0.1260 | 0.0806 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 3184 | N | 0.1335 | 0.1340 | 0.1335 | 0.1340 | 0.1335 | 0.1335 | 0.1340 | 0.1340 | 0.1340 | 0.1335 | ... | 0.1335 | 0.1335 | 0.1340 | 0.1340 | 0.1335 | 0.1340 | 0.1335 | 0.1340 | 0.1335 | 0.1335 |
| 3185 | N | 0.0806 | 0.1260 | 0.0806 | 0.1340 | 0.1335 | 0.1340 | 0.1340 | 0.1340 | 0.1260 | 0.0806 | ... | 0.0806 | 0.0806 | 0.1340 | 0.1260 | 0.1340 | 0.1260 | 0.0806 | 0.1340 | 0.1335 | 0.0806 |
| 3186 | N | 0.1335 | 0.1340 | 0.1335 | 0.1340 | 0.0806 | 0.0806 | 0.0806 | 0.0806 | 0.0806 | 0.1340 | ... | 0.1340 | 0.0806 | 0.1335 | 0.0806 | 0.1335 | 0.0806 | 0.1335 | 0.0806 | 0.1335 | 0.1340 |
| 3187 | N | 0.1260 | 0.1335 | 0.1335 | 0.1340 | 0.1335 | 0.1260 | 0.1340 | 0.1335 | 0.1335 | 0.1260 | ... | 0.1260 | 0.1340 | 0.1340 | 0.1260 | 0.1260 | 0.1260 | 0.1260 | 0.1340 | 0.1260 | 0.1260 |
| 3188 | N | 0.1260 | 0.0806 | 0.0806 | 0.1340 | 0.1335 | 0.0806 | 0.1340 | 0.1340 | 0.1335 | 0.1260 | ... | 0.1260 | 0.1260 | 0.0806 | 0.1335 | 0.1260 | 0.1340 | 0.1340 | 0.1260 | 0.1335 | 0.1335 |

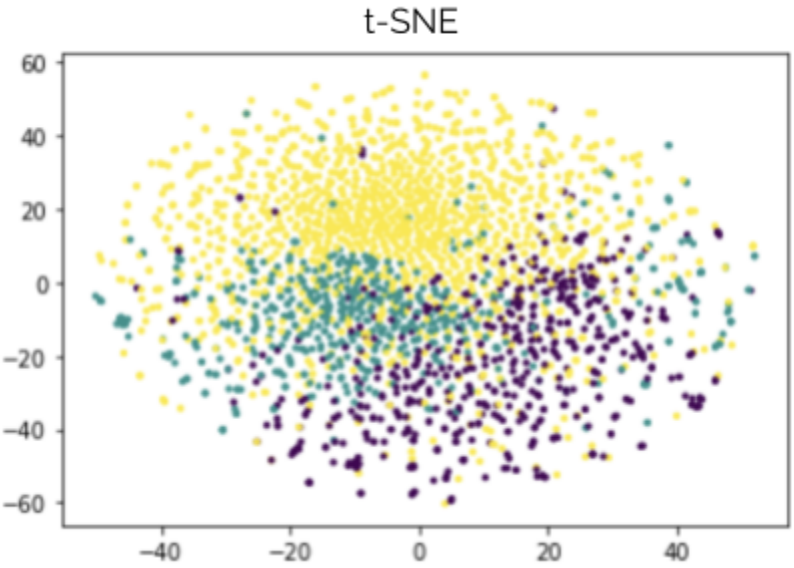
The first 30 nucleotides of an intron to exon boundary are an intron segment and the last 30 nucleotides are an exon segment. The opposite is true for exon to intron boundaries. DNA segments that are classified as neither may contain both exons and intron segments (or only one), but there is not a 30 nucleotide to 30 nucleotide split of an intron and exon segment.

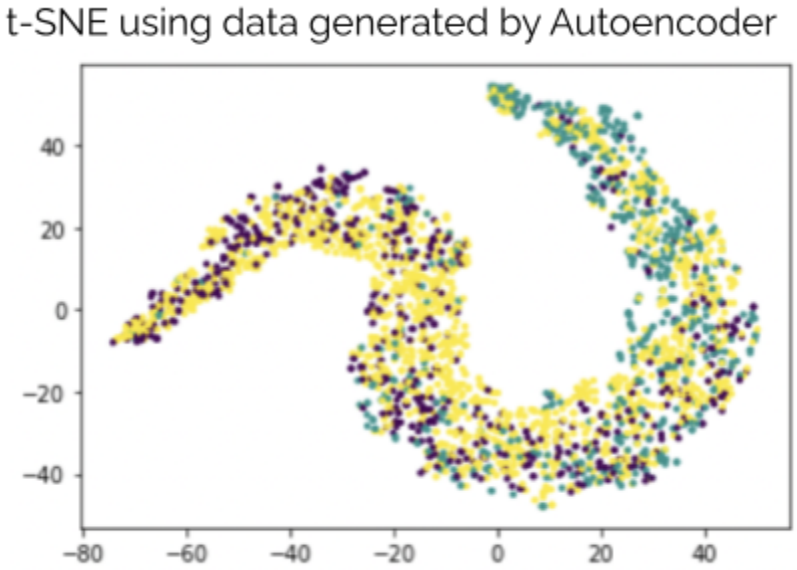



Because of this discrepancy, we upsampled the IE and EI to 1350 DNA sequences using the resample tool from sklearn.utils. Each nucleotide was originally represented by letters (A, T, G, or C), which we then replaced by a number (0.1260 for A, 0.0806 for G, 0.1340 for C, 0.1335 for T ) which corresponds to its electron-ion interaction pseudopotential (EIIP).

Prior to upsampling, we performed PCA and t-SNE to better visualize the dataset. Both are techniques that attempt to group data without the knowledge of the labels.





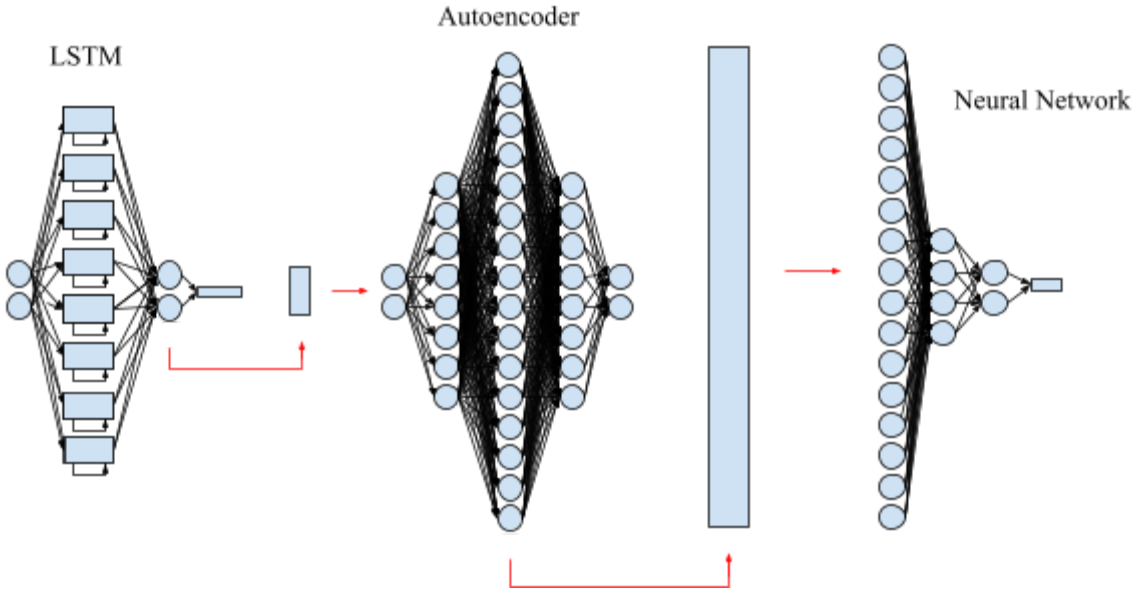




The yellow data points are N, the purple are EI, and blue are IE. While there is some overlap between the clusters, it can be seen that there is some natural grouping of each color. Specifically, the PCA approach seems to do the best job of organizing the points.

**Methodology/Models**

We used four different models to classify the DNA sequences. All four models utilized the sklearn library. Initially, there was an instance name given for each DNA sequence. This feature was removed. We split the data into 80% training and 20% test. The data was originally stored as a dataframe, but we later converted to tensor. The initial model we used was a logistic regression algorithm. This was the baseline model used to classify the sequences. We also used a multilayer perceptron. It included 6 layers (60 units, 256 units, 128 units, 64 units, 32 units, 3 units). We used SGD as the optimizer and categorical cross entropy as the loss function. We also utilized A LSTM recurrent neural network. Recurrent neural networks use inputs as well as prior outputs to improve the algorithm. Comparatively, standard multilayer perceptrons do not use prior outputs to improve performance. This attribute of recurrent networks makes them great for sequences where the order is important. A LSTM is a type of recurrent neural network that has a longer memory of previous outputs than a standard recurrent neural network. LSTMs can better make use of earlier parts of a sequence than a standard recurrent neural network. This suits this problem well due to the length of the DNA sequences. The LSTM model includes 3 LSTM layers (60 units, 32 units, 16 units) and 1 dense layer (3 units). Training and testing data was reshaped to better fit the LSTM model. We compiled it using adam as an optimizer and mean absolute error as a loss function. Finally, we used a LSTM, autoencoder and multilayer perceptron model to classify DNA sequences. We first passed the data through the LSTM. We made use of an autoencoder to expand the initially 60 dimensional data into new 256 dimensional data. This new data was fed into a multilayer perceptron.

LSTM, Autoencoder, and Neural Network Model



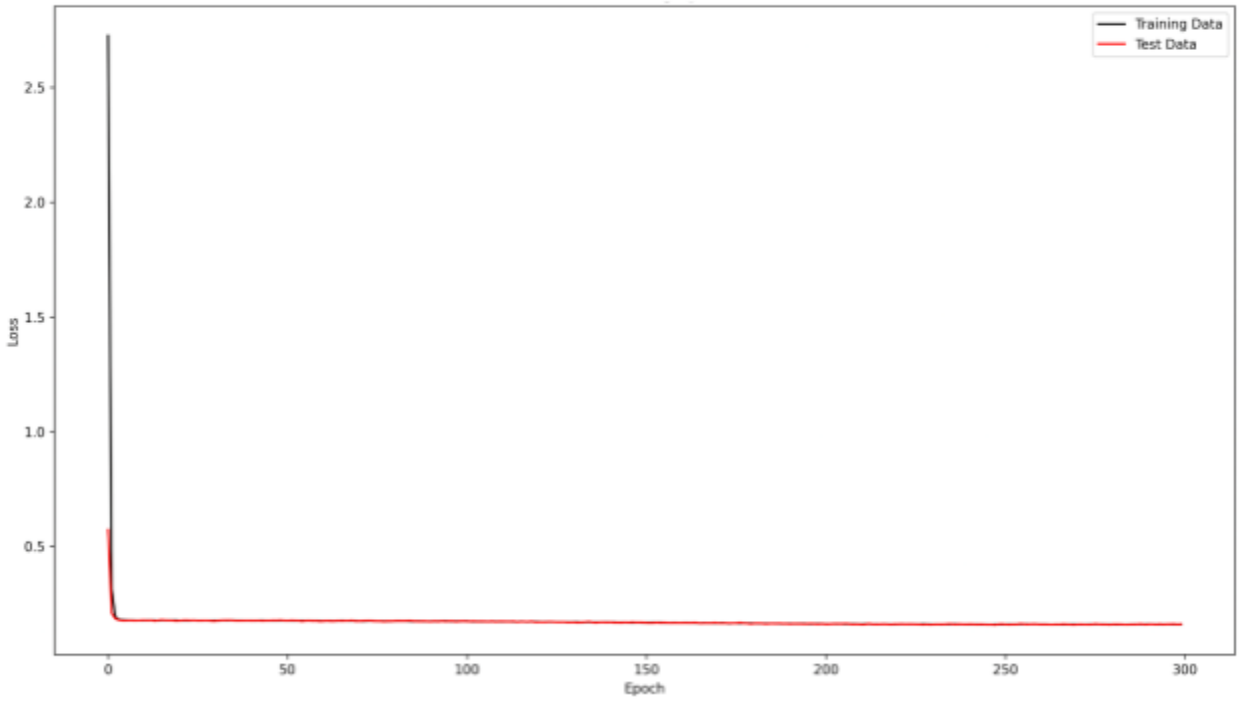

The LSTM consisted of 4 layers: input (60 units), LSTM layer (256 units), dense layer (64 units), output layer (3 units). We compiled it using adam as an optimizer and mean absolute error as a loss function. The autoencoder consisted of five layers: input (64 units), encoding (256 units), bottleneck (512 units), decoding (256 units) and output (64 units). We compiled the autoencoder using adam as an optimizer and mse as a loss function.

Autoencoder Model Summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Input-Layer (InputLayer) | [(None, 64)] | 0 |
| Encoder-Layer (Dense) | (None, 256) | 16640 |
| Encoder-Layer-Normalization (BatchNormalization) | (None, 256) | 1024 |
| Encoder-Layer-Activation (LeakyReLU) | (None, 256) | 0 |
| Bottleneck-Layer (Dense) | (None, 512) | 131584 |
| Decoder-Layer (Dense) | (None, 256) | 131328 |
| Decoder-Layer-Normalization (BatchNormalization) | (None, 256) | 1024 |
| Decoder-Layer-Activation (LeakyReLU) | (None, 256) | 0 |
| Output-Layer (Dense) | (None, 64) | 16448 |

Total params: 298,048
Trainable params: 297,024
Non-trainable params: 1,024

Autoencoder Loss by epoch




Once the training of the LSTM was complete the output layer of the LSTM was taken off and the 64 dimensional data was obtained. The 64 dimensional data was then fed through the autoencoder. Once the autoencoder was trained the last two layers of it were taken off and the

256 dimensional data was gathered. From here the 256 dimensional data was used as input to a new 5 layer multilayer perceptron (30 units, 32 units, 16 units, 8 units, 3 units). The multilayer perceptron, LSTM recurrent neural network, and LSTM, autoencoder, and multilayer perceptron model were all built using keras.

**Results and Discussion**

All four models performed relatively well (80-90% accuracy). Of the four the LSTM, autoencoder, and multilayer perceptron model performed the best (90% accuracy), the LSTM model performed the second best (88% accuracy), the multilayer perceptron performed the third best (85% accuracy), and the logistic regression model performed the worst (80% accuracy). The LSTM portion of the LSTM, autoencoder and multilayer perceptron model also performed at around 88% accuracy. This means that the autoencoder and multilayer perceptron accounted for a 2% increase in accuracy. The strong performance of the LSTM model and the LSTM portion of the novel model point to the importance of the ordering of nucleotides within DNA sequences in classifying the boundaries between exons and introns. It is worth highlighting that LSTM only slightly outperformed the multilayer perceptron, which does not take into account the order of a sequence. The amount of a specific nucleotide or patterns of nucleotides may have been identified by the multilayer perceptron to help it classify sequences.

## Conclusion

The goal was to correctly classify exon and intron boundaries and adequate results were gathered. We used multiple different machine learning approaches, LSTMs performed the best, proving the importance of the order of nucleotides when classifying DNA sequences. In the future, looking at LSTM and autoencoder models with significantly longer sequences may yield exciting results.

## Acknowledgements

I would like to thank Samuel Kwong for the guidance and advice on this project.

## References


Nair, Achuthsankar S, and Sivarama Pillai Sreenadhan. “A Coding Measure Scheme Employing Electron-Ion Interaction Pseudopotential (EIIP).” *Bioinformation*, Biomedical Informatics Publishing Group, 7 Oct. 2006, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1891688/.

*UCI Machine Learning Repository: Molecular Biology (Splice-Junction Gene Sequences) Data Set*, https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences).

Nguyen, Ngoc Giang, et al. “DNA Sequence Classification by Convolutional multilayer perceptron.” *Journal of Biomedical Science and Engineering*, Scientific Research Publishing, 19 Apr. 2016, https://www.scirp.org/journal/paperinformation.aspx?paperid=65923.

Sarkar, Manish, et al. “Splice Junction Classification Problems for DNA Sequences: Representation Issues.” *DTIC*, https://apps.dtic.mil/sti/citations/ADA409780.

Woods, Tonya, et al. “Characterizing Exons and Introns by Regularity of Nucleotide Strings - Biology Direct.” *BioMed Central*, BioMed Central, 8 Feb. 2016, https://biologydirect.biomedcentral.com/articles/10.1186/s13062-016-0108-7.