**Computer architecture – classifying DNA sequences using CNN**

Avichay Hayne, Alon Reicher, Asaf Koren

**Introduction:**

In this project, our focus revolves around optimizing computational processes in neural networks for the purpose of expediting viral sequencing analysis in our contemporary era, wherein speed and efficiency are crucial.

Within the scope of our research, we are working with a diverse dataset of 100 classes comprising different viral sequences, with each class containing 50 distinct sequences. This comprehensive collection allows us to explore the genetic diversity of viruses extensively and enhances the robustness of our analysis.

Several challenges present themselves during our investigation. One of the key challenges involves efficiently managing large and complex datasets, which is vital for the success of our research. We are dedicated to devising a well-structured dataset management approach that enables effective navigation through the vast amount of genetic information.

Additionally, we are actively addressing the issue of sequence length variation, which is common in viral genetic data. Our aim is to develop an effective strategy to handle variable sequence lengths, allowing us to extract meaningful insights and patterns from these diverse sequences with precision.

In summary, this project delves into optimizing computational processes in neural networks for viral sequencing analysis. With a diverse dataset and thoughtful consideration of challenges, we are committed to advancing the field in a polite and level-headed manner.

**Problem formulation:**

1. Problem setting:

A genome data being sequenced at an edge node (e.g.: doctor’s office, etc.), is represented as a string , for alphabet  
 . The lineage of the sequence is unknown, the sequence is then being processed by a CNN which in turn outputs the most fitting lineage statistically to the sequence.

1. Deep neural network:

A Deep Neural Network (DNN) can be formally defined as a parametric function approximator that consists of multiple layers of interconnected nodes, where each node represents a neuron. Let X be the input data, Y be the output, and W be the set of earnable parameters, which include weights and biases of the neurons. The DNN can be represented as a composition of L layers, denoted as , where represents the transformation at layer *l* and is defined recursively as follows:

for and where represents represents the weights and biases at layer *l*, and is the activation function applied element-wise to the output of each neuron at layer l. The DNN learns the optimal values of W through the process of training, where it aims to minimize a predefined loss function that quantifies the discrepancy between the predicted output   
 and the true output Y. The optimization of is typically performed using techniques such as stochastic gradient descent and backpropagation, allowing the DNN to adapt and learn complex representations from the input data, making it a powerful tool for various machine learning tasks.

1. Data preprocessing:

A genome data is represented as a string , for alphabet . Sequenced genomes usually are very long and not consistent in length when comparing to different sequences. To compensate for this problem, we have used the K-mer method. K-mer is a fundamental concept in bioinformatics and genomics that refers to a short subsequence of length k extracted from a longer DNA or RNA sequence. Representing DNA sequences as sets of overlapping k-mers, where each k-mer is encoded into a binary vector based on its presence or absence in the sequence, allows for effective data representation and analysis. K-mers capture local patterns and dependencies in DNA sequences, making them valuable for various genomics-related tasks, such as DNA sequence classification, motif discovery, and gene expression prediction. While k-mers may lose the chronological order of the sequence, their simplicity and ability to handle variable-length data make them widely used in genomics research and an essential tool for efficiently processing large-scale genomic datasets.

**Implementation:**

1. Data preparation:

Initiating the data preparation phase, we curated a dataset comprising 5000 Fasta files, each representing a distinct viral sequence from 100 diverse classes. An initial preprocessing step involved the removal of an extraneous first line from all files to ensure uniformity.

We processed each file and organized the data into a structured format. By creating a data frame with two columns, we captured the relevant information: the sequence itself, and its corresponding class label.

To extract meaningful features from the DNA sequences, we used K-mers of size 6, partitioning each sequence into a series of overlapping substrings.

1. Training the Neural Network:

We developed a neural network architecture consisting of five convolutional layers. Each layer carries out a series of operations, including convolution, batch normalization, activation using Rectified Linear Units (ReLU), and max-pooling, which condenses the most important information.

The complexity of learned features is controlled by the number of output channels in each layer. This parameter influences the network's ability to grasp patterns in the data, crucial for accurate classification.

Following the feature extraction process, the architecture transitions to two fully connected layers. These layers, with a dropout layer in between, are focused on the classification task. The insights discovered from the previous layers are compacted into a flattened representation that navigates the neural connections.

The inclusion of the dropout after the initial fully connected layer plays a vital role in preventing overfitting. By deactivating select neuron activations during training, dropout enhances the network's robustness and generalization capabilities.

We initialize a CountVectorizer instance, which forms the basis of BoW (Bag of Words) creation. This technique treats each DNA sequence as a distinct "document" and divides it into units known as "tokens." These tokens, in our case, correspond to k-mers (subsequences of length k).

Applying the fit\_transform method, the BoW model extracts and quantifies the occurrence of k-mers across the dataset, thereby constructing a numerical matrix. This matrix, denoted as X, captures the essence of each DNA sequence in terms of k-mer frequencies.

We split the data into training and testing with a test size of 0.2. Employing the cross-entropy loss function, we define the measure by which the model's predictions are compared to actual class labels. The optimizer, specifically the Adam optimizer, is tasked with iteratively adjusting the model's internal parameters to minimize the defined loss.

תמונה שמכילה טקסט, צילום מסך, צבעוני, קו

התיאור נוצר באופן אוטומטיThen, we run the model over 100 episodes and evaluate the accuracy and the loss of the model over the episode.

**Results:**

After 100 episodes we obtained the following results:

Accuracy: 0.8990

Precision: 0.9325

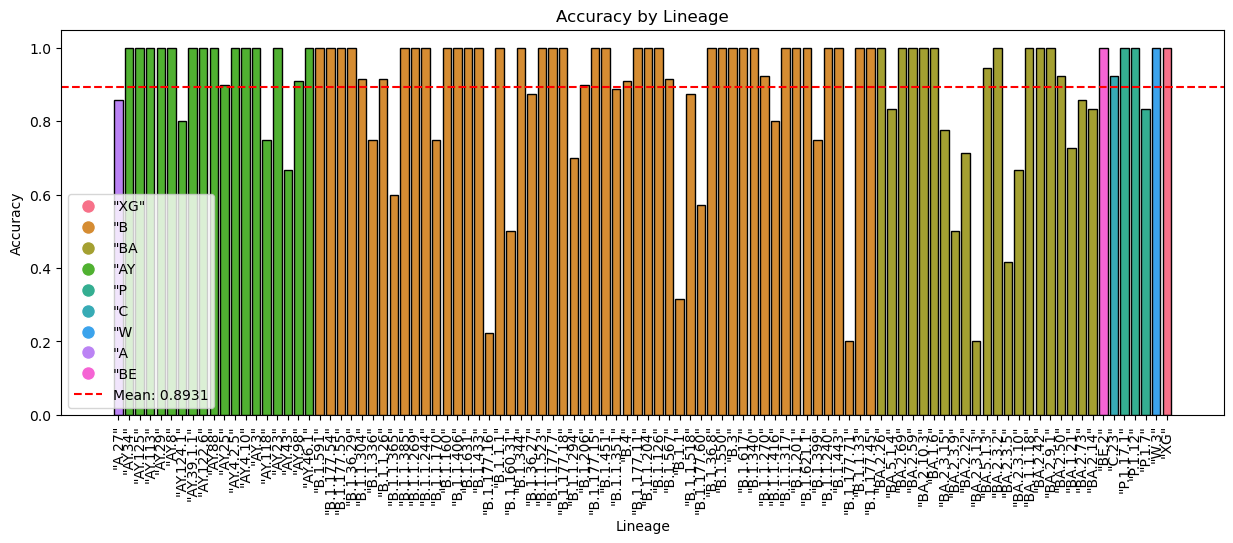
Recall: 0.8990

F1 Score: 0.8990

We can see the improvement over episodes number in the following graph:

Furthermore, we can analyze the individual accuracy of every class:

**TOP 3 Accuracy by lineage**

****

**Conclusion:**

In summary, the implementation of the ComplexCNN analysis model has yielded impressive results for accurate classification in text data. The model demonstrated a good and accurate performance, showcasing its ability to capture intricate patterns and dependencies within textual information. We also have confidence in the k-mer approach and the values it brought to our work.

For future work, we can say that the model lacks variety with the data that we introduced to him during evaluation and training. So for the future we can suggest training the model on larger scales of data.