The future of data storage: ”Digital Polymers”

Lucas Trognon, Mahammad Ismayilzada, Harold Benoit
EPFL, Switzerland

***Abstract*—This paper reports our work on using machine learning to quantify and identify different polymer sequences. It concludes on insights on how to encode information with polymers. Our repository can be found here: https://github.com/HaroldBenoit/ml4science-polymers**

I. Introduction

Clive Humby said: ”Data is the new oil”. Indeed, in 2020, each person in the world is producing about 1.7 megabytes of data every second. In just a single year, that amounts to 418 zettabytes.[1]

Our current data storage is based on magnetic and optical systems, which are energy-consuming to use, not extremely durable, and therefore, not future-proof.

But there is hope: storing data in biological molecules such as DNA or so-called ”digital polymers”. In nature, DNA encodes and stores massive amounts of readable genetic information in tiny volumes (cells, bacteria, viruses) – and does so with a high degree of safety and reproducibility. Compared to conventional data-storage devices, such molecules are very stable, have a million-fold higher storage density, and consume 100 million times less energy to store the same amount of data as a hard drive. A year's worth of a person's data can be stored in just four grams of molecules.

To read this data, one way is to use nano-sized holes called nanopores, which latch onto a cell's membrane and form a tube-like channel through it. The polymer passes through the nanopore like a tunnel, steered by voltage, and its different components produce distinct electrical signals based on the bulkiness of the component, which can then be used to identify them and read the information stored in the sequence of molecules forming the polymer.

At this stage, the current goal is to build a library of molecules to encode information with polymers that can be recognized fast and accurately using nanopores.

To make things more tangible, an example of a polymer sequence describing the sequence of bits “010” is “AA00400AA”. “AA0” and ”0AA” mark the beginning and end of the sequence. “0” is the backbone of the polymer (non-bulky element) and encodes a “0” bit, and “4” is a bulky element and encodes a “1” bit.

To better understand which molecules are best to encode information, our research questions are:

1) Is it possible to reliably distinguish between the bulky elements ’2’, ’4’, and ’5’ for the same sequence?
2) Is it possible to reliably distinguish between the ’020’ and ’0220’ encoding?
3) Is it possible to reliably distinguish between using the ’0’ and ’6’ backbone?

Our task is to use machine learning to quantify and compare the difficulty of identifying different polymer sequences, as described above. Using this information, we will assess some pitfalls with different encodings (different backbone: ‘0’ $\sim$ ‘6’ or double-encoding: ‘2’ $\sim$ ‘22’) and ultimately find a basis to encode information in these polymers (bulky elements ‘1’ $\sim$ ‘2’ $\sim$ ‘5’). This report is a proof of concept but it's possible to extend the study to more molecules and tests.

II. Data Analysis and Processing

A. Data specificities

Our data is electrical relative current time series. Here are its specificities:

- The lengths of the relative current time-series span a wide range of values, ranging from 0.3 milliseconds to 2 seconds. One reason is that some polymers get stuck in the nanopore and do some sort of back and forth in it before finally leaving.
- Two different polymer sequence events are indistinguishable for the human eye. Both in time series, autocorrelation, and frequency domain representations.
- The relative current sensor has a slow response time in its measurements. Thus the relative current measured is highly dependent on the speed of the polymer passing through.

B. Data processing pipeline

Given the specificities of the data, an appropriate data processing pipeline was needed to make the data usable by ML algorithms, especially neural networks. Its details can be found in *’pipeline.py’*.

1) We observed that our datasets were imbalanced. Given that our classification models have the purpose of classifying different polymer sequences, it seems reasonable that every class is equally likely. Thus, we balance the datasets to have an equal size to remove any dataset size bias.
2) Concerning the length of events, there were two problems. First, some events were too short to contain any information about the polymer structure. Second, some

polymer designs are "biologically longer" which tends to result in longer events. To remove any bias and make our models more robust, we removed events that were outside of the 90%th percent quantiles of the smallest dataset (in terms of event lengths), to remove any dwell time bias.

3) We then process events in blocks or intervals, as will be described below.
4) Finally, we standardize all features.

The processing part goes as follows:

- We divide the event in equally sized intervals. Either the number of intervals can be defined by the parameter *num_blocks* and the interval length is adapted for each event, or the interval length can be defined by the parameter *block_size*. With *num_blocks* $= 3$, we obtain:

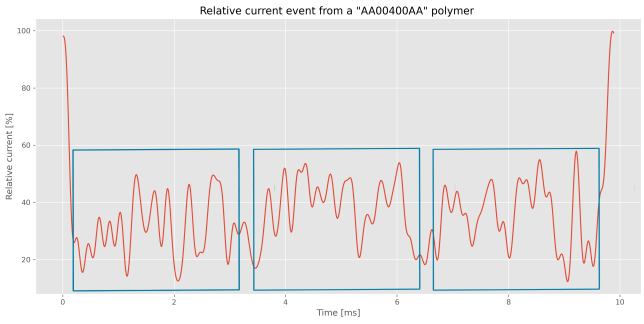

Figure 1. Division of the relative current event into equally sized intervals.

- For every interval, we extract a vector of real numbers by applying aggregating functions. This gives us:

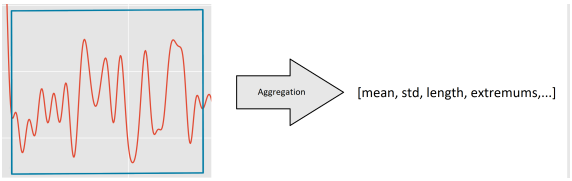

Figure 2. Performing aggregation on an interval.

This entire processing pipeline lets us have a fixed-length feature vector for every event, which makes the dataset readily usable for neural networks and other ML methods.

All parameters such as *num_blocks* or the aggregating functions applied to an interval are modular and can be defined for each use case. Furthermore, this pipeline can also be applied to transformations of the relative current such as its *Fourier Transform.* We will define below, for each subproblem, the parameters and transformations used.

III. Multi-class classification

A. Problem

Our first task is distinguishing between 3 polymers using different molecules as bulky elements ('2', '4', and '5'). For easier reference, we will call them *polymer-2*, *polymer-4* and *polymer-5* sequences based on their biological specifications. Additionally, the same non-bulky element ('6') is used as a backbone for all sequences. This is an interesting problem to explore because it can allow us to increase information density by using a combination of 2 or more molecules to achieve base-3 encoding and beyond.

B. Methods

1) Data Processing: Our problem is a multi-class classification task where input data is a batch of sequences of *(time, current)* pairs and output is one of the 3 labels corresponding to the different polymer types. Data is preprocessed as outlined above in the data processing pipeline section and each subsequence of the data is represented with the following features (or aggregating functions as we called them earlier):

- Basic features (*sequence length, dwell time, minimum, maximum, average and median relative current, standard deviation of relative current*). *Dwell time* is defined to be the difference between the start and end timestamps of the subsequence.
- Extrema features (*number of peak extrema, number of low extrema, average peak and low extrema, standard deviation of peak and low extrema*). The lower bound on the difference between 2 nearby extrema is controlled by the parameter *extrema_th*. This helps to prune insignificant local extremums.
- FFT (Fast Fourier Transform) features (*maximum, minimum, and average FFT amplitude, standard deviation of the FFT amplitude, FFT dwell time*). These features are calculated based on the absolute value of the FFT transformation applied to the subsequence.

2) Modelling: In the modeling stage, a *Random Forest Classifier* is used as a baseline and two variations of *LSTM* recurrent neural networks followed by multi-layer perceptron (MLP) are used as benchmark models. The first variation (*VanillaLSTM*) is the default LSTM implementation whereas the second variation (*MultiOutputLSTM*), feeds outputs of all the LSTM cells into the following dense network layer. Exact details of the layers of the networks can be found in *'models.py'*. Finally, besides training a multi-class classification model, we also experiment with the pairwise binary classification of given polymer sequences to confirm our results further. Models for multi-class classification are hyperparameter tuned using Grid Search with K-fold cross-validation (k=4). Parameters tuned are *num_blocks*, *extrema_th*, *by_quantile*, *learning_rate* (only for LSTM models) and *n_estimators, min_samples_leaf, min_samples_split* (only for Random Forest model). Once

	Accuracy	F1 Score
RF (multi-class)	70	71
VanillaLSTM (multi-class)	72	72
MultiOutputLSTM (multi-class)	70	70
VanillaLSTM (polymer-2,4)	89	89
VanillaLSTM (polymer-4,5)	86	87
VanillaLSTM (polymer-2,5)	74	76

Table I
COMPARISON OF CLASSIFICATION TASKS (%) USING DIFFERENT MODELS. ALL METRICS ARE REPORTED ON A TEST SET.

	Polymer-2	**Polymer-4**	**Polymer-5**
Polymer-2	*59%*	*8.9%*	*32%*
Polymer-4	*4.8%*	*83%*	*12%*
Polymer-5	*15%*	*12%*	*74%*

Table II
CONFUSION MATRIX FOR MULTI-CLASS CLASSIFICATION BETWEEN THE THREE POLYMERS USING VANILLALSTM

tuned, final models are trained with tuned parameters for a high number of epochs where applicable. More details can be found in the corresponding notebook under *'multi-class'* folder.

C. Results

We report the final model accuracy and F1 Score results for all models in *Table I.* and the confusion matrix in *Table II.* They will be further discussed in the subsection *Polymer design* of the conclusion of the report. Final model weights for multi-class classification have been saved under the *'multi-class'* folder.

IV. DISTINGUISHING BETWEEN PAIRS AND SINGLETONS OF BULKY ELEMENTS

A. Describing the problem

In this task, we are trying to reliably distinguish between using one bulky element '020' and two bulky elements '0220'. We will reference this as single and double bulky encoding. Going back to the measuring process, our data is the relative current which represents how obstructed the nanopore is while the polymer passes through. 100% relative current means the pore is fully open. 0% means it's fully closed. Because we cannot control exactly the movement of polymers through the pore, the amount of time a polymer blocks the pore cannot be controlled. This can potentially be a problem as it would be thus hard to distinguish between single and double bulky elements if the captor has already reached full saturation at 0% at the first bulky element.

B. Methods

1) Data processing and features: We use the same features and processing as defined in the first task. *num_blocks* $= 6$ was determined to be the best parameter for this problem.

2) Model: Similarly, to the above, we use a LSTM model.

3) Structure and hyper-parameter tuning: Various parameters of the model such as the number and structure of the dense layers, their activation functions, the number of stacked LSTMs, hidden dimensions, and even the features that were being fed to (with or without FT) have been tuned with the help of grid search. We've found that increasing the model complexity could sometimes increase the test accuracy but was slowing down the training immensely. To facilitate our investigative work on the data, we've opted for a lighter model that still reached good accuracy. In the end, the structure of our model is described in the *PReLULSTM* class in *'models.py'* with parameters: $input\_dims = 20$, $hidden\_dim = 64$, $num\_layers = 1$ and $output\_dim = 2$.

4) Results: K-fold cross validation (k=5) yields an **accuracy of 95.62%.** in 150 $epochs$ with a 0.005 $learning\_rate$. SVMs and Random Forests were tried as well, but they achieved poorer accuracy.

V. DISTINGUISHING THE BACKBONES

A. The problem

In this task, we are trying to determine if it is possible to reliably distinguish between using the '0' backbone (e.g. "AA00400AA") and the '6' backbone (e.g. "AA66466AA"). This task is interesting as it lets us test the sensitivity of the nanopore method. Indeed, backbones are non-bulky elements and therefore, their "current footprint" is less dramatic than one of a bulky element. In terms of expected differences in the data, using the '6' backbone makes the polymer biologically "longer", which in terms, results in longer length relative-current events. Indeed, the length distributions between the two datasets were quite dramatically different. As said above, to mitigate this length bias, datasets were adjusted such that the events outside of the "0" backbone 90%th quantile length distribution were removed.

Given these adjustments, we hope that high classification accuracy can be taken as a sign that the nanopore method is sensitive enough to detect small differences. We will examine feature importance later to support our point.

B. The model

4 different models were tried: Gradient Boosting Classifier, LSTM, and Random Forest. All 3 models achieved a similar accuracy level. Given the diminishing returns of complex models such as LSTMs when taking into account training time and our computing power, we decided to use a Random Forest model for this task.

1) Hyperparameter tuning: We tried changing the *num_blocks* parameter for building the features over the range [1,10]. *num_blocks* $= 5$ was found to be the best parameter for our features.

We used the 'RandomForestClassifier' implementation from sklearn for our model. We did a grid search

over these parameters: *'boostrap'*, *'max_depth'*, *'criterion'*, , *'max_features'*, *'min_samples_leaf'*,, *'min_samples_split'*, and *'n_estimators'*. For each combination of the parameters, we used k-fold cross-validation with $k = 5$.

We achieve a **test accuracy of 96.03%** on the dataset with a '4' bulky element. We achieve a **test accuracy of 93.47%** on the dataset with a '5' bulky element. It, thus, seems that distinguishing backbones is easier when using the '4' bulky element. The final parameters can be found at the bottom of *'backbone/backbone_RF_model.ipynb'* file.

2) Feature importance: After achieving reasonably high accuracy, we decided to investigate further to understand what was driving the classification of our model and better understand our features. We, therefore, decided to use two methods to measure feature importance: "Gini impurity" and "Permutation importance".

3) Gini impurity: Gini impurity is the classic function, used in the CART algorithm, to measure the quality of a split. To make it simple, the higher the feature is in the tree, the higher its importance.

4) Permutation importance: The permutation feature importance is defined to be the decrease in a model score when a single feature value is randomly shuffled. This procedure breaks the relationship between the feature and the target, thus the drop in the model score is indicative of how much the model depends on the feature.

We only plot the Gini-based feature importances of our model because of the lack of space.

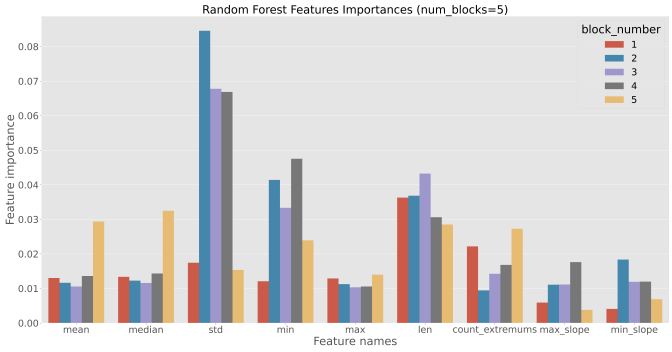

Figure 3. Gini impurity based feature importances of the random forest using the dataset with *num_blocks* $= 5$.

5) Interpretation: First, we can observe that the length is the 3rd most important feature according to Gini impurity. It is considered not significant according to the permutation importance criterion. This observation lets us be confident that our model isn't only using the length of the signal to differentiate the two backbones.

Second, the standard deviation and minimum are the two most important features for both measures. More specifically, the features coming from the 2nd, 3rd, and 4th block. We may interpret this as the 3 middle blocks capturing the "040" information from the "AA0040AA" polymer structure.

VI. Conclusion

A. Polymer design

1) Reading a sequence of bulky elements: We've managed so far to tell apart from having one or 2 consecutive bulky elements with a 95.6% accuracy, however similarly to the backbone case, we realized that the *"minimum current"* feature was one of the most important features for our models. Taking a closer look at the data, we observe that more than 75% of the *min* values of the double-bulky dataset ("AA662266AA") are below the first quartile of the *mins* of the single-bulky dataset("AA66266AA"). This means that a naive classifier using only this feature could reach 75% accuracy. Furthermore, in the double-bulky dataset, the relative current tends to get close to its minimum value of 0% (full current blockage), meaning the relative current could stay constant at 0% if we were to add more consecutive bulky elements. This would mean losing information from one of our most valuable features for our model and potentially damaging the accuracy.

2) Using different backbones: Encoding information using two different types of backbones seems possible since our ML methods can distinguish them. This would mean potentially doubling the number of possible bit sequences given a "fixed length" polymer.

3) Using different bulky elements: Based on our confusion matrix, distinguishing between polymer-2 and polymer-5 sequences proved to be the hardest one. This suggest that these 2 molecules may not be good candidate to represent encodings beyond base-2. However, using polymer-2 and polymer-4 could be used to switch to ternary encoding.

4) Going further: Although, there isn't a lot of litterature on using Transformers for Time Series Classification, to further increase the accuracy of our methods, the usage of Transformers might prove beneficial.

B. Experimental design

1) Pore design: As said earlier, one way to circumvent the saturation issue on the current sensor at 0% is potentially to have a wider nanopore. This way, we would have a finer range of values when a bulky element passes through. However, this could decrease the sensitivity of the sensor too much for a correct distinction between non-bulky elements.

2) Dataset balance: To have an unbiased accuracy measure , we've had to balance the datasets and throw out in some cases more than half of the captured data. It would be economically more efficient to measure the same amount of data for each polymer sequence.

References

[1] N. Papageorgiou, "Bacterial nanopores open the future of data storage," *EPFL*, 2020. [Online]. Available: https://actu.epfl.ch/news/bacterial-nanopores-open-the-future-of-data-stor-6/