Nucl2Vec : Local alignment of DNA sequences using Distributed Vector Representation

Prakhar Ganesh, *Senior, IIT Delhi,* Gaurav Gupta, *MTech, IIT Delhi,* Shubhi Saini, *MTech, IIT Delhi,* and Kolin Paul, *Professor, IIT Delhi*

Abstract—The Next Generation Sequencing Technique (NGS) has provided an affordable and fast method for sequencing genetic data. However, generation of whole genome sequences and extraction of relevant information from this data is still a time consuming and computationally expensive process. With the increasing size of database in genomics everyday, we need to create tools that can provide faster processing. In this paper, we demonstrate a novel approach to encode DNA sequences using skipgram model. Our method is capable of successfully compressing the input feature size, while retaining comparative information required for further processing. We test our encoding method by performing pseudo local alignment of DNA reads with respect to a reference genome. We are able to significantly improve the speed of alignment, while achieving competitive accuracy. We provide extensive experimentation as well theoretical insights into the performance of our method, and compare it with industry standards like BWA-MEM and BLAST.

Index Terms—Genome, Alignment, Local Alignment, Distributed vector representation, Skipgram, Nearest Neighbor

1 Introduction

Genome sequencing is figuring out the order of DNA nucleotides (bases) Adenine, Guanine, Cytosine and Thymine (A, G, C and T respectively) that make up an organism's DNA. Available methods of genome sequencing can only handle short stretches of DNA at a time [1] and therefore the genome is sequenced in pieces [2]. In this method, the genome is sequenced by breaking it into small segments, known as reads. The reads generated using these high throughput sequencers are fragments of length 100-900 bps with error rate of 1%-2% per read [3]. To facilitate variant discovery with high confidence, these reads are generated with average 30x coverage of the DNA. Due to large number of short length reads produced with high sampling rate, their alignment, along with error isolation, is a computationally intensive task.

Alignment is the process by which we align the reads to their corresponding (most likely) locations in the reference genome [4]. Computational approaches to sequence alignment generally fall into two categories: global alignments and local alignments. Calculating global alignment is a form of global optimization that forces the alignment to span the entire length of all query sequences. By contrast, local alignments identify regions of similarity within long sequences that are often widely divergent overall (Fig. 1). Because of the computational burden, it is impossible for non-heuristic local alignment methods to scale up with the increasing size of DNA datasets and thus many different heuristics have been proposed over time [5].

BWA-MEM and BLAST are two of the most popular sequence alignment tools for raw NGS reads. BWA-MEM [6] is based on Burrows-Wheeler Transform (BWT) algorithm and attempts to find at each query position the longest exact match covering the position. It uses seed-and-extend paradigm along with BWT-index for faster search and various provisions in the algorithm like re-seeding are focused on recovering missed hits, which makes it one of the most commonly used short reads sequence alignment method.

Fig. 1: Global vs Local Alignment

However, since it tries to match the complete read, BWA-MEM does not scale well with read length. Basic Local Alignment Search Tool (BLAST) [7] is a heuristic search which seeks partial reads of length W that score at least T when aligned at the query position and are then extended in both directions in an attempt to find a locally optimal ungapped alignment known as high scoring pair (HSP). It is the official database search tool for the NCBI website[1].

Similar to database search in genomics, another field of research which values faster alignment over accuracy is metagenomics [8]. The field of metagenomics is the study of genetic material from environment without the need to isolate individual species. It has recently emerged along with affordable and faster genome sequencing techniques. Metagenomics requires binning the reads to identify species diversity [9] and prefers methods like BLAST over BWA-MEM. With the increasing size of databases in genomics, it is getting more and more important to develop faster search tools to do rapid classification in lieu of accuracy loss.

The field of genomics enjoys a level of parallelism with common computer vision (CV) problems like image classification, object detection etc. and is also connected to natural language processing (NLP) if one views DNA as a string of alphabets. Due to the success of deep learning in CV and NLP and their abstract connections with this

1. https://blast.ncbi.nlm.nih.gov/Blast.cgi

field, a great deal of work has been done in recent years to incorporate these methods, more specifically convolutional neural networks (CNN), in the field of Bioinformatics [10]–[16]. However, most of these approaches use primitive one hot vector encoding to encode nucleotides into vector representations. These encodings have evident limitations, as they do not translate any sequence related information (bi-nucleotide, tri-nucleotide relationships etc.). Moreover the size of input is increased by at least four folds (since one hot encodings map to a vector of vocabulary size) which inadvertly increase the processing time.

In this paper, we propose a novel encoding method, namely Nucl2Vec, to create compact representations for both the sequenced reads and the reference DNA and further use them to do local alignment. Our encoding method has the following key features:

- Highly compressed mathematical representations which facilitates faster processing.
- Distributed nature of the encodings which provides them comparative value responsible for alignment accuracy.
- Encodings trained using deep learning which means they can learn latent features that cannot be manually assigned.
- Length of encoded read independent of the actual read length, making our method robust to varying read sizes.
- Encodings trained in an unsupervised manner which allows them to adapt easily to any alignment algorithm (we use 1NN in this paper).

We use different reference DNAs and corresponding sets of Illumina reads for evaluation. Our approach is able to provide compact yet accurate representation of the sequenced reads which are then matched with the reference for faster local alignment. We compare our performance, both in terms of alignment accuracy and speed, with two commonly used alignment tools BWA-MEM and BLAST. We also provide theoretical insights into the alignment speed of various methods.

The rest of the paper is organized as follows. Section II reviews the related work. Section III formally defines our framework to perform local alignment. Section IV presents our proposed encoding method along with training details. Section V presents an extensive analysis of our algorithm's performance and Section VI concludes the paper.

2 Related Work

Distributed vector representation for creating word embeddings (commonly known as Word2Vec) was first made popular by Mikolov et al. [17], [18]. Since then, the creation of pre-trained word embedding have been improved upon repeatedly in NLP [19]. The core idea behind this is two fold. One, these embeddings are believed to be a good semantic representation of the word and two, they are pre-trained and are thus easily adaptable across various models/tasks.

Deep learning methods in genomics have been gaining popularity in recent times with a great deal of work being done in a try to replace the classical benchmarks or provide a better analysis of the data at hand [10]–[16]. Although limited, some studies do exist on distributed vector representation in genomics [20]–[22]. BioVec and ProtVec [20] uses skipgram based model to represent protein sequences. Their work is focused towards classification and encodes long sequences of protein into large encoding vectors. Dna2Vec [21] uses encoding similar to Word2Vec. But instead of compressing information, they expanded k-mers ($3 \leq k \leq 8$) into continuous vectors of 100 dimensions, purely for the purpose of detailed comparative analysis.

Aoki and Sakakibara [22] have used skipgram based encoding to do alignment and generate motif of non-coding regions of RNA. They have encoded 4-mers into 12 length dimensional vector. They provide an encoding which is trained with the complete pipeline and thus is bound to that particular task and model used, and the length of their encoding depends on the read length which makes it more computationally expensive to handle longer reads. We on the other hand train encodings separately in an unsupervised manner, which allows them to be used freely for any ML model. We also provide fixed length encodings for reads of varying length. Aoki and Sakakibara [22] decreased the length of encoding dimension as compared to Dna2Vec [21] but we aim to further compress the encoding length to substantially decrease the computational expenses.

3 Framework

In order to evaluate the performance of our proposed encoding method as well as perform several ablation studies, we create a framework using 1-nearest neighbour (1NN) algorithm to do pseudo-alignment. Broadly, our framework can be divided into 2 separate parts, 1) Encoding the reference DNA and creating a 1NN tree. Needs to be done only once for every reference DNA. 2) Encoding the input read and finding the best match in the reference 1NN tree.

Nearest Neighbors algorithm is one of the most commonly used classification algorithms. 1NN creates an index tree from all available data and uses this tree to predict the clusters of a new sample point based on some similarity measure. It is a non-parametric technique, which means that it does not make any assumptions on the underlying data distribution [23]. The algorithm is known to work well with large datasets, however carries with it the curse of dimensionality, which means the data sparseness increases exponentially with the feature dimension. This further reinstates the requirement of a highly efficient feature encoding.

To create the 1NN tree, we first extract all kmers (a sequence of nucleotides of length 'k') present in the reference DNA, for a pre-defined value of k. We then encode these kmers using Nucl2Vec, which along with their alignment position, are used as "training data" for 1NN algorithm. The encoded kmers are used to prepare the 1NN tree and vector L2 distance is used as a measure of similarity, thus hypothesizing that similar kmers will have similar encoding. While the algorithm is usually considered computationally heavy, with appropriate optimization and pre-processing we can reduce the computation burden of the inference process. We simply use the K-Nearest Neighbor classifier (with K=1) provided by *sklearn* library in Python [24].

Next we encode the reads using Nucl2Vec and then use these encodings to search in the reference 1NN Tree for the best possible match. Due to the approximate nature of our algorithm, we actually create multiple encodings for a single read and try to match each of them individually in the 1NN tree. The match with the least L2 similarity

distance is considered the correct alignment position for the read. However, if this minimum similarity distance is greater than some pre-decided threshold, we consider that the read cannot be aligned anywhere on the reference DNA and thus is given the alignment value '-1'. Refer to Algo 1 for the pseudocode.

Algorithm 1 Framework for Local Alignment

```
procedure PROCESSREFERENCE(reference)                    ▷ To create 1NN Tree
    allEncodings ← []
    for i from 0 to len(reference)-k with jump of '1' do
        currKmer ← reference[i:i+k]
        enc ← ENCODEKMER(currKmer)
        allEncodings.append(enc)
    referenceTree ← Create 1NN Tree using allEncodings and their index
    return referenceTree

procedure FINALALIGNMENT(reference, readsArray, thresholdDis)
    reference1NNTree ← PROCESSREFERENCE(reference)
    alignment ← []
    for read in readsArray do
        Encodings ← ENCODEREAD(read)
        EncodingsPos ← Alignment for Encodings from reference1NNTree
        EncodingsDis ← Corresponding L2 Distance
        optDis ← Minimum(EncodingsDis)
        optPos ← Corresponding alignment position
        if optDis ≤ thresholdDis then
            alignment.append(optPos)
        else
            alignment.append(-1)
    return alignment
```

4 NUCL2VEC ENCODINGS

In this section, we define our skipgram-based Nucl2Vec encodings. We first focus on the details of the training method that we use to create the base embeddings, along with some qualitative analysis. We then use these embeddings to create encoding representations for both the input reads as well as the reference DNA. Finally, we combine our encoding method with the framework setup to define the complete pseudo-alignment method.

4.1 Skipgram training

An essential feature for any good embedding would be to contain not only the information regarding the composition of the pmer but also its interactions with other pmers. We use skipgram to get a distributed vector representation for every pmer, encoding similarity and composition. Skipgram has been typically used as a feature engineering technique in NLP. In our case, we will be training the skipgram model on pmers. Every permutation of nucleotides forming a pmer will be considered a word in our vocabulary. This will make the vocabulary size of our corpus = 4^p.

It should be noted that skipgram in NLP is sometimes used as the initial part of a bigger pipeline (for example, sentiment analysis, machine translation etc.), and the complete pipeline is trained end-to-end. Similar method is used by Aoki and Sakakibara [22] to train their encodings. However, in our case, we use skipgram in an unsupervised manner to first generate pmer embeddings (based of their similarity as described below) and then use these to create encodings for alignment using 1-Nearest Neighbors. Thus, the training of these embeddings is not directly connected to the alignment process and vice-versa. This gives us the freedom to use our embeddings with any ML algorithm for alignment.

The input to the skipgram model is a corpus of sentences and the underlying assumption is that words used together in the same sentence tend to have more similarity. So, to define similarity between pmers, we use the number of edits, or edit distance. Two pmers P and P' are at an edit distance of '1' if, *(i)* On substituting a nucleotide in P, we obtain P', *(ii)* On inserting a nucleotide in P followed by deletion from the right or left extreme (to maintain pmer length), we obtain P', or *(iii)* On deleting a nucleotide in P followed by insertion at the right or left extreme (to maintain pmer length), we obtain P'. For more details refer to Fig 2.

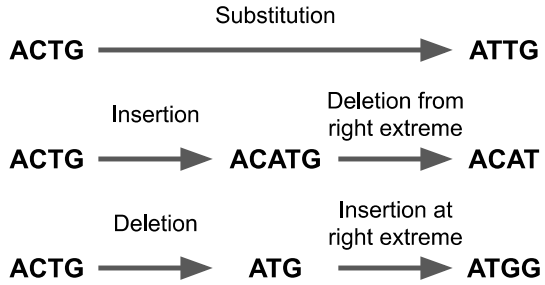

Fig. 2: Types of Edits

Now that we have defined a metric of similarity between pmers, we create 'sentences' for training of the skipgram model. For every pmer in our vocabulary, we collect all pmers at edit distance '1' from this main pmer and place them around it. The pmers are treated as words, and thus the sentences formed are datapoints for training Nucl2Vec base embeddings. Refer to Algo 2 for the pseudocode.

Algorithm 2 Nucl2Vec Training

```
function NEARBYSET(inputPmer)    ▷ Function to generate all pmers at an edit
distance '1' from the inputPmer
    for every location lo in the inputPmer do
        bp ← ['A', 'C', 'T', 'G']
        temP1, temP2, temP3, temP4, temP5 ← Copies of inputPmer
        for nucleotide b in bp do
            substitutePmer ← temP1.replaceWith(ele = b, loc = lo)
            insertionPmerRight ← temP2.insert(ele = b, loc = lo)[:-1]
            insertionPmerLeft ← temP3.insert(ele = b, loc = lo)[1:]
            deletionPmerRight ← temP4.delete(loc = lo) + 'b'
            deletionPmerLeft ← 'b' + temP5.delete(loc = lo)
            Insert pmers formed above into an array
    return array of all the Pmers formed

function CREATESENTENCE(inputPmer)    ▷ Function to create training corpus
sentence for the given inputPmer
    pmerArray ← NEARBYSET(inputPmer)
    pmerArray.insert(ele = inputPmer, loc = len(pmerArray)/2 )
    finalSentence ← Join all the words in pmerArray to create a sentence
    return finalSentence

procedure ENCODINGTRAINING(p)                    ▷ Create the training corpus
    allPmers ← All possible permutations of nucleotides of length 'p'
    trainingData ← []
    for pmer in allPmers do
        pmerSentence ← CREATESENTENCE(pmer)
        trainingData.append(pmerSentence)
    nucl2vecEmbedding ← Skipgram(trainingData)
    return nucl2vecEmbedding
```

Once the sentence corpus is created in the aforementioned manner, we train the skipgram model using heirarchial softmax, with window size equal to the number of words in the sentence. Due to the smaller corpus size (4^p), we are able to train our skipgram model efficiently without

any additional modifications like negative sampling, subsampling or dynamic context windows.

4.2 Qualitative Analysis of Base Embeddings

Skipgram training is probabilistic and thus every time we train on the complete vocabulary, we will get different embeddings for the pmers. However, the property that pmers which have similar embeddings will have smaller edit distance is maintained. We consider one such training run (with p=4 and embedding dimension=1) to do a qualitative comparison of few 4-mer embeddings as shown in Fig 3. Note that this is not a quantitative representation of the actual behavior of these embeddings and the examples shown here are handpicked to emphasize certain observations.

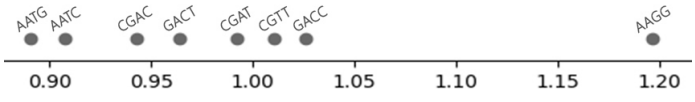

Fig. 3: Comparing a few random embeddings

We can see the close proximity of 4-mers with small edit distance, like AATG and AATC (edit distance = 1), CGAC and GACT (edit distance = 1). Similarly, 4-mers like AAGG and GACC (edit distance = 3), AAGG and CGAT (edit distance = 3) are sufficiently far from each other. However, there are also a few anomalies present : AATG and AAGG (edit distance = 1) are not nearby while CGTT and GACC (edit distance = 3) are in close proximity. This behavior is expected as we are trying to reduce the feature space, and is exacerbated when we force extreme compression.

Compressing pmers into smaller dimensions provides a great boost in the computation speed. Yet there is also a clear disadvantage present in extreme compression to create these embeddings, as can be noticed from the anomalies observed above. However, by stacking multiple pmers together, we can reduce the probability of a mismatch. For example, let us say that the probability of an incorrect match of pmers under such an embedding is *pr*. Then the probability of an incorrect match of pmers stacked together decreases to *pr*pr* for 2 pmers, *pr*pr*pr* for 3 pmers and so on. In conclusion, the more number of pmers are stacked together for matching, the lesser chance there is of an incorrect match. An experimental analysis of the effect of the number of pmers on accuracy is done later in this paper.

For the rest of the paper, pmers will denote nucleotide sequences used in Nucl2Vec training and 'c' will represent the number of such pmers stacked together. Thus, cpmers (length = c*p) will represent the complete nucleotide sequences involved during the alignment. Note that the term kmers used in the previous section to define all the nucleotide sequences extracted from the reference DNA is the same as the term cpmers defined here (i.e. k=c*p).

4.3 Nucl2Vec Encodings

To create encodings for the reference DNA, we first extract all cpmers present in the reference DNA. We then encode them using the Nucl2Vec embeddings that we have learned. A cpmer can be encoded by encoding every stacked pmer present in it separately and then stacking the results back together. For example, if the Nucl2Vec learned embeddings for 4-mers (p=4), then to encode a 16-mer (c=4 and p=4), we first divide it into 4-mers and then encode them separately.

For encoding a read, we take 4 cpmers : one each from the front and end, and two from the center of the read. This way, we only consider a segment of the read instead of considering the whole read sequence to create encodings. By using this heuristic, we are improving the performance of 1NN by reducing the size of search space. The idea is based on the intuition of how we solve jigsaw puzzles. While joining two pieces of a jigsaw puzzle, we emphasize more on how the corners fit rather than the content. Similarly, for finding correct alignments for the reads, we only consider small segments of the read (see Fig 4) and hypothesize that the rest of the read will match at the same location.

We take two contiguous regions from the center instead of evenly spaced regions because of the behavior of widely used high throughput genome sequencing methods. These methods are based on a common principle of extending a smaller piece of DNA by attaching nucleotides one at a time and tagging it with fluorescent dyes [25]. These methods are known to have more error prone corners compared to the center, as the DNA strands floating in the solution might get broken from the corner and reattached to other floating strands. Thus when aligning reads generated using these sequencing methods, we choose more regions from the center for better accuracy. Refer to Algo 3 for details.

Algorithm 3 Creating Final Encodings

```
function EncodeKmer(inputKmer)        ▷ Function to encode the inputKmer
    finalEncoding ← []
    for i from 0 to len(inputKmer)-p with jump of 'p' do
        currPmer ← inputKmer[i:i+p]
        enc ← nucl2vecEmbedding(currPmer)
        finalEncoding.extend(enc)
    return finalEncoding

function EncodeRead(read)
    front ← EncodeKmer(read[0:c*p])
    end ← EncodeKmer(read[len(read)-c*p:len(read)])
    centerleft ← EncodeKmer(read[len(read)/2-c*p:len(read)/2])
    centerright ← EncodeKmer(read[len(read)/2:len(read)/2+c*p])
    return [front, end, centerleft, centerright]
```

4.4 Complete Model

Our final proposed model takes an existing reference sequence and a set of NGS reads (to be aligned) as input and outputs an integer value for each read representing the index at which the read aligns the best or -1 if the read does not align properly anywhere. Algorithms 1, 2, 3 and Fig 5 describe the steps executed to obtain the final output.

Step 1 : We use edit distance as previously described to represent proximity between two pmers and learn Nucl2Vec base embeddings. Generation of these embeddings is a one time process.

Step 2 : The pre-processing involves creating encodings of every cpmer present in the reference DNA and building a 1-Nearest Neighbor tree.

Step 3 : Once the pre-processing of reference DNA is done, we proceed to align the reads. For each read, cpmers from the front, end and two cpmers from the center

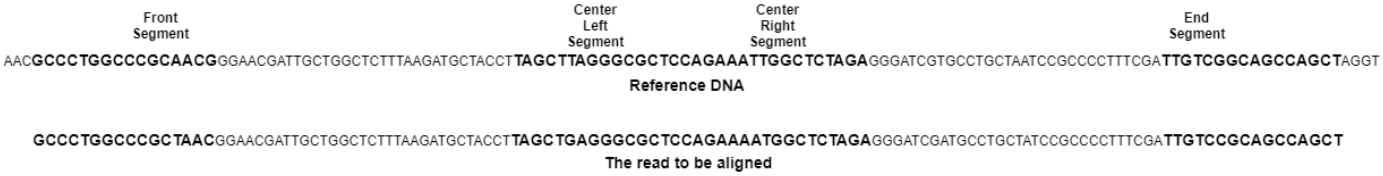

Fig. 4: Matching only a segment of the read (represented by bold nucleotides). The rest of the read seems to match at the same location too.

(center left and center right) are taken and encoded. These are matched in 1-nearest neighbors tree which outputs the match distance and predicted alignment location. Matches which have the match distance less than some threshold are considered, for all others we output '-1' (unaligned).

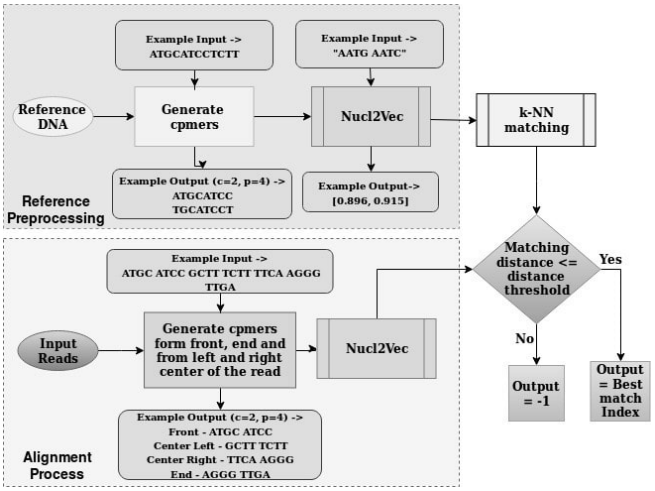

Fig. 5: Complete Pipeline Flowchart

5 Evaluation

We now evaluate the performance of our proposed alignment method. We first describe the experiment settings, evaluation metrics and the details of the dataset used. For hyperparameter search, a small subset of the dataset is used. We discuss the theoretical bounds for alignment time of our algorithm as well as of BWA-MEM and BLAST. Finally, we present experimental results to illustrate the performance of our approach under diverse settings and compare it with commonly used baselines BWA-MEM and BLAST.

5.1 Experiment setting and evaluation metrics

Pseudo local alignment of a query read is defined as the location in reference genome where query read aligns the best. A single read however can align at multiple locations in the reference. To calculate the quality of alignment, we use Needlemen Wunsch (NW) algorithm [26] to generate alignment scores. These scores are normalised by the read length to keep the final score between 0 and 1.

We define the problem statement as follows. The input to the model is a reference DNA and a set of reads. The output of the model is the alignment position l_i for each read. To score the alignment quality, we calculate the normalised NW score nw_i for every read which is aligned. The final accuracy of the model can be judged using an indicator function as,

$$Accuracy = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{(nw_i \geq nw_{thr})}, \quad (1)$$

where m is the total number of reads aligned and nw_{thr} is the alignment score threshold. While the evaluation seems like a binary score task, shifting the alignment a few position does not substantially hurt the NW score.

The above metric however does not consider reads that are left unaligned by the proposed solution. To better understand our alignment quality and analyse the unaligned reads, we also do a detailed comparison of our alignment predictions with BWA-MEM. We further analyse the mismatch instances using NW alignment scores at locations predicted by our model and BWA-MEM.

To test our proposed model, we have obtained data from the NCBI website for 3 different bacteria, namely Listeria Monocytogenes, Salmonella Enterica and Escherichia Coli. For each reference genome of bacteria we have experimented with 3 randomly chosen Illumina NGS reads sets [27]. Details regarding the reference genome and the reads are mentioned in Table 5 and 6.

5.2 Model Parameter Setting

There are a number of hyperparameters on which our model can be optimized. This includes the length of pmers and encoding dimension for the Nucl2Vec training, as well as the value of 'c' and number of cpmers used for creating final encodings. Various experiments have been performed with Escherichia Coli DNA and a subset of the corresponding Illumina NGS reads set to explore these hyperparameters.

TABLE 1: Performance comparison across varying hyperparameters for base embeddings.

Parameter Setting		Performance	
p	encoding-dimension	Accuracy (%)	Time per read (ms)
3	1	60.12	0.154
3	2	61.52	0.393
4	1	78.23	0.127
4	2	78.14	0.465
5	1	75.80	0.129
5	2	75.48	0.353
6	1	73.16	0.132
6	2	74.89	0.427

For the same value of encoding dimension, a higher value of 'p' means more number of nucleotides are included in the cpmer segment that we use for alignment. However, this also means that the anomalies between the encodings become more and more evident. Thus a balance between the two needs to be maintained to achieve the best accuracy.

From Table 1, we can see that encoding 4-mers seems to be the correct choice for the best accuracy. Also, we see that the variation in computation time dependents significantly on the encoding dimension. Since the accuracy is almost similar for both, we should prefer encoding in 1-dimension for faster prediction speed. We will use 4-mers with encoding dimension 1 for further analysis.

TABLE 2: Performance comparison across varying values of c.

Parameter Setting	Performance	
cp (p=4)	Accuracy (%)	Time per read (ms)
8	0.1	0.031
12	58.47	0.051
16	78.23	0.127
24	77.91	0.511
32	78.02	2.171

Next, we experiment with different values of 'c*p' (for p=4) and reported the accuracy and prediction time against the length of segment used. From Table 2, it can be observed for the given configuration that the maximum accuracy is achieved at cp = 16 (c=4), after which it does not increase but instead decreases (this may happen due to the curse of dimensionality). Also we can see from Table 2 that the prediction time increases exponentially, as expected. Thus, cp=16 (c=4) seems to be the best choice for segment length.

TABLE 3: Performance comparison across different region choices.

Parameter Setting	Performance	
Matching cpmers	Accuracy (%)	Time per read (ms)
Front	78.23	0.127
+ End	89.44	0.252
+ Left Center	93.22	0.394
+ Right Center	93.49	0.513

Since we are using 16-mer segments for alignment, we have to determine how many and from where these segments should be taken. The possibilities include the front segment, the end segment, the left of the center and the right of the center. We experimented with the combinations of these and the accuracy and prediction time are reported in Table 3. As expected, the prediction time increases linearly with the number of segments, as increasing the number of segments means doing the 1NN tree search that many times. We can also note that the accuracy starts saturating as we increase the number of segments used. Since there is an increase in accuracy between 3 segments and 4 segments, we will use 4 segments for the future experiments.

The final experiment is done on complete dataset and resukts are compared with baseline methods like BWA-MEM and BLAST. The value of hyper-parameters used is shown in Table 4.

Hyper-Parameter	Value
p	4
cp	16
encoding-dimension	1
cpmers locations from each read	Front, End, Left Center and Right Center

TABLE 4: Model Hyper-Parameters

5.3 Alignment time : Theoretical bounds

In this section, we do a theoretical analysis of various alignment methods to better understand why our proposed solution is faster than the existing solutions. We first start with BWA-MEM, which is based on BWT-index creation. Since BWA-MEM tries to match the entire read, its theoretical time complexity for searching a read alignment is $O(|w|)$, where $|w|$ represents the read length. In tasks that require searching a single read through a large database, BWA-MEM performs poorly as it takes significantly large amount of time. However, in tasks which require reconstructing genome sequences from a large set of reads, BWA-MEM is a favorable choice as it provides good alignment accuracy.

BLAST on the other hand has a theoretical time complexity of $O(|w| * n)$, where $|w|$ is same as above and n represents the reference DNA length. However, due to it's heuristic nature the actual performance is way faster than its theoretical bound. This makes BLAST suitable for searching through large databases, but its heuristic nature makes it a poor choice for alignment of large sets of reads.

Our proposed solution uses KD-tree for 1-nearest neighbour and due to the very small dimension of search vector, we can achieve time complexity of $O(s * logn)$, where n denotes the length of the reference DNA and s denotes the number of different cpmers we use for alignment. This gives us two major advantages. Firstly, while the alignment speed for both BWA-MEM and BLAST depend on the read length, our alignment speed depends only on the length of the reference DNA. Secondly, since the time complexity of our algorithm depends on logarithmic progression and the length of DNA is usually of the order of millions, the variations in DNA length have no practical effect on the alignment time performance of our algorithm.

Thus, our proposed solution provides an almost constant alignment speed independent of the read length as well as the reference DNA length. This gives our algorithm a significant edge over other solutions in extreme cases, for example with unusually long read or reference. Since we try to encode reads of all length into same size vector, longer reads might get a slightly less accurate representation as compared to shorter reads, which can hurt our accuracy. However, the tradeoff between speed and accuracy is in our favor, with more detailed experiments provided in the next subsection.

5.4 Accuracy and Time Analysis

We do a accuracy as well alignment time comparison for BWA-MEM, BLAST and Nucl2Vec and provide the results in Table 5. The reference DNA as well as the set of reads used have a varying range of properties in order to fully explore the capabilities of our model. For example, for Listeria Monocytogenes, the read set SRR9063775 (row 2) contains extremely long reads with a average length of 241. On the other hand, the read set SRR9066644 (row 3) for the same reference DNA contains smaller reads with a average length of 138, but the total number of reads in the set are significantly more than any other sets in our experiments.

We do not go into a detailed discussion of the computational cost of index generation here as it is done just once for every reference DNA and is comparable for all

TABLE 5: Accuracy and Time comparison of BWA-MEM, BLAST and Nucl2Vec

				BWA-MEM		BLAST		Nucl2Vec	
Bacteria	**Illumina Read Sets**	**Av. read length** (bp)	**No. of Reads**	**Accuracy** (%)	**Time** (sec)	**Accuracy** (%)	**Time** (sec)	**Accuracy** (%)	**Time** (sec)
Listeria Monocytogenes (2.9M bp)	SRR9062615	234	425.0k	91.74	145	88.29	124	96.02	57
	SRR9063775	241	281.8k	86.64	172	80.18	78	90.8	39
	SRR9066644	138	1.8M	96.92	304	94.55	456	93.85	224
Salmonella Enterica (4.8M bp)	SRR9067199	200	422.5k	96.4	107	90.81	128	94.01	52
	SRR9067207	218	693.9k	95.91	273	85.13	256	91.56	87
	SRR9067279	217	573.9k	97.15	162	93.45	173	91.6	72
Escherichia Coli (4.6M bp)	SRR9067016	238	516.9k	93.34	226	88.85	161	90.15	76
	SRR9067573	199	534.9k	95.66	164	93.95	175	86.98	68
	SRR9067633	241	665.2k	93.57	289	90.41	248	89.43	88

three algorithms. Also, since the 1NN tree used here is a replaceable framework that we use to show the importance of our encodings, we believe that a discussion into the details of index generation is not in the scope of this paper.

We first explore the accuracy of various methods. While there are certain exceptions present, in general it can be seen that BWA-MEM performs significantly better in terms of accuracy, while both BLAST and Nucl2Vec deliver similar performances with Nucl2Vec being slightly better. If we calculate a weighted average of all the performances provided in Table 5, with the number of reads being the weight, we get an average accurcay of 95.13% for BWA-MEM, 90.92% for BLAST and 91.98% for Nucl2Vec. Although the variation of accuracy with read length is sometimes haphazard, there is an internal trend present that can be seen from the table. Both BLAST and Nucl2Vec have more degradation in performance for longer reads as compared to shorter reads, which is expected since both of these methods use partial read matching.

Another measure of efficiency for these algorithms is the amount of time it takes to perform the alignment, which has been the focus of our work. Table 5 also provides time analysis for all the three algorithm. It can be noticed that on average our model (Nucl2Vec) is 2-3 times faster as compared to BLAST and BWA-MEM. We can notice that the time taken for BWA-MEM increase for sets with longer average read length, due to the reasons detailed in the previous subsection. Similarly, time taken for longer reference DNA is more for BLAST as compared to shorter reference DNA. However, as expected, Nucl2Vec provides an almost constant alignment time per read, independent of the average read length or reference DNA length.

5.5 Validation of results using BWA-MEM alignments

The accuracy comparison done in Table 5 is limited to reads which get aligned. However, it does not take into account the reads which are left unaligned by various algorithms. Also, while a read might get aligned by all algorithms, there can be discrepancy on the quality of alignment due to mismatch of alignment location provided by different algorithms. Thus, doing a comparative analysis of the alignments provided by our method with BWA-MEM is also an interesting point of study. Following cases were observed during analysis.

1) Case 1 : Our results do not match BWA-MEM results, but the normalised NW scores of both alignments are comparable (difference $\leq$ 0.1). This might be due to the possibility of a read correctly aligning at multiple locations in the reference.
2) Case 2 : BWA predicted the read to be unaligned, but our model found an alignment with acceptable NW score ($\geq$ 0.7).
3) Case 3 : Our predicted alignment is a small offset from the correct alignment position and our NW score is acceptable, but lower than the score at BWA predicted location. This may be due to different interpretation of best local alignment by CIGAR(BWA) and by L2 distance(1NN) (see Fig 6)

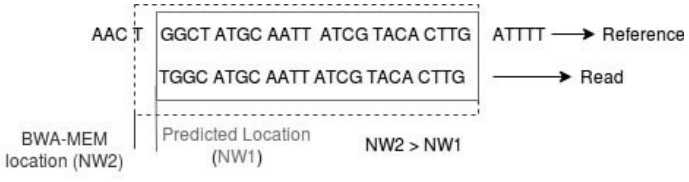

Fig. 6: Predicted Location mismatch with BWA-MEM

The errors from third case can be corrected by doing NW score check in the local neighborhood of our predicted location as shown in Fig 6. Since NW takes considerable amount of computation, this has not been included in our final model. The results on data set generated by our model were divided into following six categories

1) Category 1 : Predicted Location same as BWA location.
2) Category 2 : BWA shows no match but Nucl2Vec predicted location with acceptable NW score ($\geq$ 0.7).
3) Category 3 : BWA location and Nucl2Vec location differ, and the Nucl2Vec location NW score is low ($<$ 0.7).
4) Category 4 : BWA shows no match and Nucl2Vec predicted location with a low NW score ($<$ 0.7).
5) Category 5 : BWA location and Nucl2Vec location differ, however their NW scores are comparable (difference $\leq$ 0.1).
6) Category 6 : Nucl2Vec predicted location is -1, but BWA output location with alignment.

The accuracy comparison of our model against BWA-MEM is shown in Table 6. Category 1, 2 and 5 comprise the correct prediction percentage and the rest three categories amount to error percentage. It can be seen that, on an average, around 91-92% of our model alignment match with the alignments provided by BWA-MEM, and after category 2 and 5 results are included our model accuracy reaches around 95-96%. With this accuracy our model may be used for pseudo-alignment of NGS reads.

TABLE 6: Accuracy Of Nucl2Vec model against BWA-MEM

Bacteria	Illumina Read Sets	Av. read length (bp)	Cat1 (%)	Cat2 (%)	Cat3 (%)	Cat4 (%)	Cat5 (%)	Cat6 (%)	Accuracy (1+2+5)
Listeria Monocytogenes (2.9M bp)	SRR9062615	234	92.69	0.76	1.57	0.49	2.09	2.4	95.54
	SRR9063775	241	88.2	1.27	3.27	0.87	3.64	6.02	93.11
	SRR9066644	138	92.49	1.06	1.92	0.57	1.78	2.18	95.33
Salmonella Enterica (4.8M bp)	SRR9067199	200	93.82	1.17	0.29	0.74	1.27	2.78	95.26
	SRR9067207	218	91.85	0.83	3.23	0.51	1.02	2.56	93.7
	SRR9067279	217	93.71	1.42	0.55	0.62	1.25	2.45	96.38
Escherichia Coli (4.6M bp)	SRR9067016	238	93.58	0.54	0.64	0.98	1.37	2.89	95.49
	SRR9067573	199	91.67	1.82	0.69	1.07	1.71	3.04	95.2
	SRR9067633	241	93.42	1.65	1.55	0.96	0.95	1.47	96.02

6 Challenges and Future Work

The aim of the paper is to provide a Machine Learning based method of generating features out of nucleotide sequences that represent more information than the traditional one-hot vector encodings while compressing the total size of the input data. We use our encodings to perform local alignment and compare its performance against existing industry standards. We found that our algorithm performs significantly faster while maintaining comparable accuracy. The Nucl2Vec embeddings have the capabilities of being used in other tasks as well as with other ML algorithms, as it is a method of feature representation.

The embeddings used in this paper were trained solely on mathematical intuitions. However, substitutions of certain nucleotide pairs might have more biological preference than others and this information might be used to influence pmer proximity while training Nucl2Vec embeddings. Adding such biological insights might increase the accuracy even further, and is a promising direction of future research. Also, we use a simple framework setup to do local alignment using these encodings, and it will be interesting to test the transfer learning capabilities of our encodings and see how well they work with other ML algorithms as well as in variety of tasks other than alignment.

References

[1] El Mustapha Bahassi and Peter J. Stambrook. Next-generation sequencing technologies: breaking the sound barrier of human genetics. *Mutagenesis*, 29(5):303–310, 2014.

[2] Pauline C Ng and Ewen F Kirkness. Whole genome sequencing. In *Genetic variation*, pages 215–226. Springer, 2010.

[3] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. Comparison of next-generation sequencing systems. *BioMed Research International*, 2012, 2012.

[4] Dr. Mel. Bwa, soap, maq, blat. . . omg alignment methods!, 2014.

[5] Gregory Kucherov. Evolution of biosequence search algorithms: a brief survey. *Bioinformatics*.

[6] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[7] Ph.D Ingrid Lobo. Basic local alignment search tool (blast).

[8] Philip Hugenholtz and Gene W Tyson. Metagenomics. *Nature*, 455(7212):481–483, 2008.

[9] Victor Kunin, Alex Copeland, Alla Lapidus, Konstantinos Mavromatis, and Philip Hugenholtz. A bioinformatician's guide to metagenomics. *Microbiol. Mol. Biol. Rev.*, 72(4):557–578, 2008.

[10] Jason R Miller, Sergey Koren, and Granger Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, 2010.

[11] Pooja Dixit and Ghanshyam I Prajapati. Machine learning in bioinformatics: A novel approach for dna sequencing. In *Advanced Computing & Communication Technologies (ACCT), 2015 Fifth International Conference on*, pages 41–47. IEEE, 2015.

[12] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning. *Nature Biotechnology*, 2015.

[13] David R. Kelley, Jasper Snoek, and John L. Rinn. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research*, 2016.

[14] Jack Lanchantin, Ritambhara Singh, Zeming Lin, and Yanjun Qi. Deep motif: Visualizing genomic sequence classifications. *CoRR*, abs/1605.01133, 2016.

[15] Haoyang Zeng, Matthew D. Edwards, Ge Liu, and David K. Gifford. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121–i127, 2016.

[16] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 2015.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[19] Yang Li and Tao Yang. Word embedding for understanding natural language: a survey. In *Guide to Big Data Applications*, pages 83–104. Springer, 2018.

[20] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287, 2015.

[21] Patrick Ng. dna2vec: Consistent vector representations of variable-length k-mers. *arXiv preprint arXiv:1701.06279*, 2017.

[22] Genta Aoki and Yasubumi Sakakibara. Convolutional neural networks for classification of alignments of non-coding rna sequences. *Bioinformatics*, 34(13):i237–i244, 2018.

[23] Adi Bronshtein. A quick introduction to k-nearest neighbors algorithm.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Jay Shendure and Hanlee Ji. Next-generation dna sequencing. *Nature biotechnology*, 26(10):1135–1145, 2008.

[26] Vladimir Likic. The needleman-wunsch algorithm for sequence alignment. *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne*, pages 1–46, 2008.

[27] National center for biotechnology information (ncbi)[internet]. bethesda (md): National library of medicine (us), national center for biotechnology information; [1988] – [cited 2017 apr 06]. available from: https://www.ncbi.nlm.nih.gov/.

Prakhar Ganesh Senior, Dept. of Computer Science and Engineering, Indian Institute of Technology Delhi

Gaurav Gupta M.Tech, Dept. of Computer Science and Engineering, Indian Institute of Technology Delhi

Shubhi Saini M.Tech, Dept. of Computer Science and Engineering, Indian Institute of Technology Delhi

Prof. Kolin Paul Professor, Dept. of Computer Science and Engineering, Indian Institute of Technology Delhi