

A SYSTEM AND METHOD OF PREDICTING A VARIANT USING MUTATIONAL SIGNATURES AND DEEP LEARNING

Field of the Invention

5 This invention relates to a system and method in the bioinformatics field dedicated to Deoxyribonucleic acid (DNA) variant refinement. Bioinformatics is an interdisciplinary field that develops methods and software tools for understanding biological data. It is essential for the management of data in modern biology and medicine. The system and method according to this invention automates and standardizes inspections of samples to accurately identifies genetic variants. In particularly, the system and method according to this invention can efficiently predict a variant using mutational signatures and deep learning.

Prior Art

15 Next-generation sequencing technologies suffer from inaccuracies due to the existence of artifacts during the sample preparation and sequencing stage. These artifacts present a challenge for accurately calling genetic variants due to high false positives and false negatives. High error rates degrade the quality of downstream analyses' results and clinical interpretations, preventing the patient from receiving an accurate diagnosis and therapeutic opportunities.

20 As of now, the standard practice for somatic variant refinement is to manually inspect the variants using some type of genomics viewer program and this is usually performed by trained personnel such as bioinformaticians. Efforts have been made to develop automated methods for variant refinement but progress is slow due to computational limitations. Manual inspection is still the most commonly used method within the industry.

Therefore, those skilled in the art are striving to provide a system and/or a method that automates and standardizes inspections and reduces the variability of work performed by different individuals or labs, and thereby improving reproducibility.

Summary of the Invention

30 The above and other problems are solved and an advance in the state of the art is made by a system and method in accordance with this invention. A first advantage of the system and method in accordance with this invention is the system and method is light on computational requirements and is capable of running on a simple local machine. A second advantage of the system and method in accordance with this invention is that the system

and method do not use BAM files which are extremely bulky in size and require more labour and time to analyze. A third advantage of the system and method in accordance with this invention is that the system and method automates and standardizes the variant refinement process which increases the reproducibility by bioinformaticians and researchers.

5 A first aspect of the disclosure describes a method of predicting a mutation type of each of the plurality of variants in a single sample file. The method comprises: constructing a first data structure containing features of a plurality of mutation types based on a plurality of mutational signatures; reconstructing a second data structure containing site features from the single sample file; building a third data structure containing reference context
10 features for each of the plurality of variants in the single sample file; feeding the first, second and third data structures to a neural network architecture; receiving from the neural network architecture a probability value for each of the plurality of variants being real or artifactual mutation; and generating a report comprising a list of variants in the single sample file with corresponding one of predicted real and artifactual mutations.

15 According to an embodiment of the first aspect of the disclosure, the plurality of mutational signatures consists of 78 single-base substitutions (SBS) mutational signatures obtained from a Catalogue Of Somatic Mutations In Cancer (COSMIC) database.

According to an embodiment of the first aspect of the disclosure, the 78 SBS mutational signatures are segregated into 96 different contexts to form the first data
20 structure.

According to an embodiment of the first aspect of the disclosure, the step of reconstructing the second data structure containing site features from the single sample file comprises: extracting from an "INFO" column and a "SAMPLE" column from the single sample file to form the second data structure.

25 According to an embodiment of the first aspect of the disclosure, the step of reconstructing the second data structure containing site features from the single sample file further comprises: applying one-hot encoding for discrete feature extracted from the single sample file, and applying standard scaling for continuous feature extracted from the single sample file.

30 According to an embodiment of the first aspect of the disclosure, the second data structure comprises columns determined by a number of categories in the single sample file and rows determined by a number of rows in each category in the single sample file.

According to an embodiment of the first aspect of the disclosure, the step of building the third data structure containing reference context features for each of the plurality of
35 variants in the single sample file comprises: locating a plurality of mutation sites in a reference sequence file that correspond to each of the plurality of variants, the reference

sequence file containing a full human genome sequence; retrieving 10 bases at the left and right of each located mutation site in the reference sequence file to form a total of 21 bases for each of the plurality of variants; and translating the 21 bases to form a 21 dimensional vector for each of the plurality of variants to form the third data structure.

According to an embodiment of the first aspect of the disclosure, the step of feeding the first, second and third data structures to the neural network architecture comprises: passing the first data structure into a first embedding layer and converting into a point in a high-dimensional space, coordinates of which constitute an embedding vector for the type; parsing the second data structure into a linear layer; passing the third data structure into a second embedding layer and converting the 21-dimensional vector for each of the plurality of variants into a two-dimensional context matrix; concatenating the outputs from the first embedding layer, linear layer and second embedding layer and feeding into a five-layer fully connected network, each layer in the five-layer fully connected network consists of dropout linear transformation, ELU (Exponential Linear Unit) activation function, and highway module with the following expressions,

$$\begin{aligned} Linear_i(X) &= W_i X + b_i \\ ELU(x) &= \begin{cases} x, & \text{if } x > 0 \\ \alpha * (\exp(x) - 1), & \text{if } x \leq 0 \end{cases} \\ H_{i+1} &= H_i + ELU(Linear(H_i)) \end{aligned}$$

where W and b are learnable parameters, the ELU hyperparameter α controls a value at which the activation function saturates for negative inputs, the highway module connects the input of each layer to the end, solving the gradient vanishing problem caused by overly deep networks, wherein a last layer of the five-layer fully connected network contains a sigmoid function that maps the scalar output from the previous layer to a value between 0 and 1 for each of the plurality of variants.

According to an embodiment of the first aspect of the disclosure, the method further comprises: determining a real mutation in response to the value being more than a predefined threshold; and determine an artifactual mutation in response to the value being equal to or less than the predefined threshold.

According to an embodiment of the first aspect of the disclosure, the predefined threshold is 0.5.

Brief Description of the Drawings

The above and other features and advantages in accordance with this invention are described in the following detailed description and are shown in the following drawings:

Figure 1 illustrating an overview of the system in accordance with an embodiment of this disclosure;

Figure 2 illustrating a process flow executed by a pre-processing module in accordance with an embodiment of this disclosure;

5 Figure 3 illustrating a neural network architecture implemented by the deep learning module in accordance with an embodiment of this disclosure;

Figure 4 illustrating a performance comparison with and without the deep learning module based on 5 metrics;

Figure 5 illustrating a performance comparison without the deep learning module;

10 Figure 6 illustrating a performance comparison with the deep learning module;

Figure 7 illustrating an F1 performance comparison with the deep learning module for different trainsets;

Figure 8 illustrating an accuracy performance comparison with the deep learning module for different trainsets;

15 Figure 9 illustrating a precision performance comparison with the deep learning module for different trainsets;

Figure 10 illustrating a recall performance comparison with the deep learning module for different trainsets;

20 Figure 11 illustrating a specificity performance comparison with the deep learning module for different trainsets; and

Figure 12 illustrating an average performance comparison with the deep learning module for different trainsets.

Detailed Description

25 This invention relates to a system and method in the bioinformatics field dedicated to Deoxyribonucleic acid (DNA) variant refinement. Bioinformatics is an interdisciplinary field that develops methods and software tools for understanding biological data. It is essential for the management of data in modern biology and medicine. The system and method according to this invention automates and standardizes inspections of samples to accurately identifies genetic variants. In particularly, the system and method according to this invention can efficiently predict a mutation type of a variant using mutational signatures and deep learning. It is envisioned that a system and method in accordance with embodiments of this invention may be used for predicting a mutation type from human cancer samples using mutational signatures and deep learning. Specifically, the system and method in accordance with embodiments of this invention receives a file containing a

30

35

single sample and outputs a probability of a mutation type in relation to the single sample being real or artifactual.

This system and method can be written in Python programming language or any other programming languages without departing from the invention. Along with each Variant Call Format (VCF) file (each containing a single sample), the system and method according to this invention generates a report containing the real and artifactual mutations. Alternatively, the invention generates 2 separate VCF files containing real and artifactual mutations respectively. The system and method is light on computational requirements and is capable of running on a simple local machine such as a desktop or laptop. The performance and speed of system and method according to this invention have been validated using open-source datasets.

The system and method according to this invention uses mutational signatures for variant filtering and it avoids the use of Binary Alignment Mapping (BAM) files which are extremely bulky in size and require more labour and time to analyze. BAM file is a compressed binary representation of a Sequence Alignment Mapping (SAM) file that is used to represent aligned sequences up to 128Mb. These files contain alignment characteristics such as matches, mismatches, and gaps represented in the Concise Idiosyncratic Gapped Alignment Report (CIGAR) format. The BAM or equivalent files are consumed downstream through variant calling algorithms to identify genetic mutations such as single nucleotide polymorphisms (SNPs) consisting of transversions and transitions, insertions and deletions (InDels), and tumor mutation burden. The identified variants are stored in a tab-delimited text file format commonly known as the Variant Call Format (VCF).

Figure 1 illustrates the overview of the system 100 for predicting a mutation type in accordance with this disclosure. The system 100 includes a pre-processing module 110 and a deep learning module 120. The system 100 focuses on single-base substitutions (SBS) mutational signatures linked to sequencing artifacts. Upon receiving the input files 130, the pre-processing module 110 generates the required data structures and feed the data structures to the deep learning module 120 and generates a report 140 containing real and artifactual mutations respectively. Alternatively, the report 140 may be two separate VCF files containing real and artifactual mutations respectively. The input files includes a VCF file and a mutational signatures data file. The deep learning module 120 has a comprehensive embedding representation to enhance the algorithm performance. Upon training completion by the deep learning module 120, the system 100 can conduct inference efficiently without the need for a customized variant refinement optimization process for every new cancer sample. Further details on the processes of the pre-processing module 110 and the deep learning module 120 will be described below.

The system 100 is light on computational requirements and is capable of running on a simple local machine. Other than some initial parameters that the user can choose, all the other variant refinement optimization steps are automatically performed by the software from start to finish. The system 100 may be executed on typical processing systems such as a desktop computer, laptop computer, or other computer terminals. The local machine executes applications that perform the required processes required by the system 100 in accordance with this disclosure.

Processes are stored as instructions in a media that are executed by a processing system in a simple local machine or a virtual machine to provide the method and/or system in accordance with this disclosure. The instructions may be stored as firmware, hardware, or software. The processing system will now be briefly described. The processing system includes Central Processing Unit (CPU) and memory. CPU is a processor, microprocessor, or any combination of processors and microprocessors that execute instructions to perform the processes in accordance with the present invention. The CPU is connected to memories to transmit data and instructions between the memories and CPU. The memory includes non-volatile memory, such as a Read Only Memory (ROM) and/or volatile memory. Non-volatile memory stores instructions and data needed to operate various sub-systems of processing system and to boot the system at start-up. One skilled in the art will recognize that any number of types of memory may be used to perform this function. Volatile memory such as Random Access Memory (RAM) stores the instructions and data needed by CPU to perform software instructions for processes such as the processes required for providing a system in accordance with this invention. One skilled in the art will recognize that any number of types of memory may be used as volatile memory and the exact type used is left as a design choice to those skilled in the art. The processing system may also include I/O device, keyboard, display, network device and any number of other peripheral devices to exchange data with CPU for use in applications being executed by CPU.

The system 100 is a software application stored in memory or virtual memory of a local machine for performing the processes in accordance with the disclosure. The software application includes a pre-processing module 110 and a deep learning module 120. Briefly, the processes executed by the 2 modules are as follows:

- 1) Pre-processing module 110 receives files 130 comprising a VCF file and a mutational signatures data file and generates the required data structures to be fed to the deep learning module. It then receives the probability of the real and artifactual for each position mutation type and generates a report 140 or 2 separate VCF files containing real and artifactual mutations respectively.

2) Deep learning module 120 receives the data structures from the pre-processing module 110 and identifies the relevant mutation type of each variant. The main purpose of the deep learning module 120 is to predict 2 outputs, Real or Artifactual mutation type. This can be achieved as accurately as possible by feeding as much details as possible into the model in order to identify the relevant mutation type of each variant.

The system and method in accordance with this invention is the ability to predict real and artifactual mutation of each variant from the raw VCF file.

A known method, as published by the Genome Medicine by Kim et al in 2019 with the title "FIREVAT: finding reliable variants without artifacts in human cancer samples using etiologically relevant mutational signatures" uses 4 formulations to split the unrefined VCF file into refined VCF file and artifactual VCF file. However, it is observed that this method tends to prematurely remove some variants of interest. Specifically, the 4 formulations having a higher false positive rate (i.e. treating a variant as artifactual when in reality the variant is real).

More details of the 2 modules will now be described as follows.

Pre-processing Module 110

The system 100 focuses on artifacts involving single-base substitutions (SBSs) or also known as single-nucleotide polymorphisms (SNPs). SNPs form the majority of the genetic variations found in the human genome. In a typical human genome, more than 99.9% of variants are characterized as SNPs and short indels with SNPs being 6-7x higher in frequency than short indels. SNPs are also responsible for the onset of multiple cancer types including breast cancer, chronic lymphocytic leukemia, neuroblastoma, gastric carcinogenesis, prostate cancer, and others.

The system 100 minimizes variability in the bioinformatics pipeline within the variant refinement phase by removing the need for manual inspection. The system 100 queries a VCF file instead of a BAM file which is much lighter in size and processing requirements. Hence, it can be run on a local machine easily and the user only needs to input the initial parameters.

Figure 2 illustrates the process 200 for preparing the data structures to be fed to the neural network architecture 300. Process 200 further receives the predicted data from the neural network architecture 300 and generates a report 140.

Before feeding data to the neural network architecture 300, we need to first construct a first data structure containing the features of mutation types 310 in step 205. Mutation types are also referred to as nucleotide contexts in this disclosure. All point mutations are classified into one of the 96 SBS mutation types, represented by an index

ranging from 0 to 95. The 96 SBS mutation types can be obtained from the Catalogue Of Somatic Mutations In Cancer (COSMIC) database. These 96 SBS mutation types are derived from 78 SBS mutational signatures inclusive of both valid and artifactual signatures. A total of 18 SBS mutational signatures have been classified as possible sequencing artifacts: SBS27, SBS43, and SBS45-SBS60. The remaining 60 SBS mutational signatures either have valid proposed aetiologies or unknown causes. As of now, the 78 SBS mutational signatures can be segregated into 96 different nucleotide contexts to form the first data structure. Specifically, the first 18 are obtained from SBS27, SBS43, and SBS45-SBS60 while the remaining 60 are valid mutational signatures that are found to be linked to mutational processes causing cancer. The SBS signatures are available from the COSMIC website: <https://cancer.sanger.ac.uk/signatures/sbs/>. The mathematical method to generate these signatures is described in this paper: <https://www.nature.com/articles/s41586-020-1943-3> and we have basically used the end result (artifactual and valid mutational signatures) posted by COSMIC in the website.

We will now briefly describe how the 78 SBS mutation signatures are segregated into 96 different nucleotide contexts to form the first data structure. There are 4 nucleotides A, T, C, G. As we are dealing with single base substitutions, there are 6 possible base substitutions: C>A, C>G, C>T, T>A, T>C, and T>G. The mutated base is represented by the pyrimidines of the base pair T and C instead of the purines A and G. C>A means that the base C mutated to base A. Each base substitution can be contextualized using the nucleotide bases to its left and right. For example, if the mutated base is C, it can be neighbored by A to its left and G to its right. This forms 1 possible context of ACG. Each base substitution can have 16 different contexts. For C>A, they are ACA, ACC, ACG, ACT, CCA, CCC, CCG, CCT, GCA, GCC, GCG, GCT, TCA, TCC, TCG, TGT. The 96 different nucleotide context can be found here: https://cancer.sanger.ac.uk/signatures/media/images/v3.2_SBS1_PROFILE_GA_GRCh37.original.jpg.

Step 205 may be a one-off process or an initialisation process for process 200. Specifically, step 205 need not be executed again after the initialialization. If the mutational signatures from the COSMIC database are updated with the latest ones, a user may simply download the latest mutational signature data files which are reflective of the updated version. In brief, step 205 may simply be downloading of mutational signatures from the COSMIC database and extracting 78 mutational signatures inclusive of both valid and artifactual signature are segregated into 96 different nucleotide contexts to form the first data structure. Among the 78 mutational signatures, the following 18 SBS artifactual mutational signatures are extracted: SBS27, SBS43, and SBS45-SBS60. One skilled in the

art will recognise that the COSMIC database may be updated to include more mutational signatures and hence, this disclosure is not limited to the above mentioned 78 SBS mutational signatures.

In step 210, process 300 reconstructs a second data structure containing site features from a VCF file (a single sample). Specifically, the "INFO" and "SAMPLE" columns from the provided VCF file are extracted to form the second data structure. One-hot encoding and standard scaling are applied to the data in the "INFO" and "SAMPLE" columns to process the discrete and continuous features respectively to form the second data structure. Specifically, one-hot encoding is used for discrete feature while standard scaling is used for continuous feature.

An example of a discrete feature is alternate allele type which can be classified as "S", "M", or "I". In this case, one-hot encoding would be used to encode all the values for this feature. Encoding, in machine learning/deep learning context, is simply providing a numerical representation for categorical variables such as the alternate allele type.

One-hot encoding is a specific way of encoding using binary values 1 and 0. Using the alternate allele type feature as an example, there are 3 categories "S", "M" and "I". Initially, there is a single column representing this feature. With one-hot encoding, 3 new columns are created. For every category, 1 new column will be created. This means that 3 columns will be created and each column represents one of the category. So, the 3 new columns are "S", "M", "I".

For every row, if the initial value of the feature is "S", it would get the values [1, 0, 0] for columns ["S", "M", "I"] respectively. The value 1 for "S" column encodes the initial value "S" under the alternate type allele feature. The same applies to "M" and "I".

Raw data (1 row, 1 column)

Alternate allele type: S

Data after one-hot encoding (1 row, 3 columns)

Column S: 1

Column M: 0

Column I: 0

For standard scaling, it is used for continuous features. An example of such feature is allele frequency which takes on value from 0-1. e.g 0.1, 0.35, 0.88, 1.0.

Standard scaling is a scaling technique which makes the entire column centered around the mean with a unit standard deviation. Essentially, the mean of the column (allele frequency) becomes 0. Unit variance means every single value is divided by the standard

deviation aggregated over the entire column. Deep learning algorithms often require zero mean and unit variance for convergence and faster training time. The standard scaling technique can be expressed in the following expression

$$z=(x-\mu)/\sigma$$

where

z is the z-score

x is the original data point

μ is the column mean

σ is the column standard deviation

For example, raw data (3 rows, 1 column)

Allele frequency column:

- Row 1: 0.5

- Row 2: 0.8

- Row 3: 0.55

Data after standard scaling (3 rows, 1 column)

Standardized allele frequency:

- Row 1: -0.92

- Row 2: 1.38

- Row 3: -0.54

Note: $\mu = 0.62$ and $\sigma = 0.13$

Example calculation for row 1: standardized value = $(0.5-0.62)/0.13$

= -0.92

Essentially, the second data structure comprises columns determined by the number of categories in the VCF file and rows determined by the number of rows in each category in the VCF file. Usually, a subset of the total number of categories in the VCF file is chosen. This is part of the model training process where we experiment which combination of categories/features result in the best performing model. After being parsed by the Hap.py tool, data from a VCF file can be processed similarly to data from an Excel file with rows and columns.

For the development of deep learning models, a single VCF file is split into a training set (60%), validation set (20%), and testing set (20%).

The training set contains 60% of the rows and all the columns chosen after the experimentation process. This is used for the training process.

The validation set contains 20% of the rows and all the columns chosen after the experimentation process. It is used for hyperparameter tuning to achieve the best

performing deep learning model. After training 1 model on the training set with a set of hyperparameter values, we validate it using the validation set. The process is repeated for other combinations of hyperparameter values to come up with multiple models. Usually, the best performing model is tested a final time using the testing set.

5 The testing set contains 20% of the rows and all the columns chosen after the experimentation process. It is used for the final validation to test the performance of the best performing model from the previous validation. The result from the testing set simulates how the deep learning model will perform in real-life and provides a non-biased assessment.

10 Essentially, the second data structure always has the same/fixed number of columns as the categories in the VCF file are selected after the experimental process. The number of rows depend on which stage of the development process we are in: training, validation, or test stage. For the production process (i.e. after a model has been trained), the rows in second data structure would be determined by the number of rows in selected
15 categories in the VCF file. This means that during production process, all the rows in the selected categories in the VCF file will be included in the second data structure.

 In step 215, process 200 builds a third data structure containing reference context features by selecting 10 bases at the left and right of each mutation site (a total of 21 bases). The 10 bases are obtained from the VCF file and reference sequence file (in FASTA
20 format). Specifically, the VCF file contains the variant [A, T, C, G] with its chromosome and location. Hence, we locate the mutation site in the reference sequence file that corresponds to the variant [A, T, C, G] specified in the VCF file and retrieve the 10 bases at the left and right of the located mutation site in the reference sequence file to form a total of 21 bases, i.e. 10 (left of site) + 1 (mutation site) + 10 (right of site). The reference sequence file
25 contains the full human genome sequence in order. An example of the full genome sequence is the b37 human genome reference file created by the Broad Institute. Each base would be translated to a number (T:0, C:1, A:2, G:3, others:4), and the third data structure containing reference context feature 350 will be represented as a 21-dimensional vector. As described above, there will be a mutation site and we take the 10 bases to the
30 left and another bases to the right of this mutation site from the reference sequence file. Then, this will form the initial 21 bases. We will encode these bases into numbers using the representation (T:0, C:1, A:2, G:3, others:4) which form the following 21-dimensional vector. For example, [AGCTCGCTCGTCGCTGAAACC] will be encoded to [2 3 1 0 1 3 1 0
1 3 0 1 3 1 0 3 2 2 2 1 1] to form the 21-dimensional vector. Each VCF file contains a
35 number of variants and hence n number of 21-dimensional vectors would be generated to form the third data structure, where n is the number of variants in the VCF file.

Thereafter, we feed the first, second and third data structures to the neural network architecture 300 to predict 1 of the 2 possible outputs (real or artefactual) in step 220. Finally, we adopted the Hap.py tool to process the original annotation file as the ground-truth labels in this study during the training phase. Specifically, we have our own refined VCF file produced by the deep learning module and a benchmark VCF file which contains the ground-truth labels (correct answer). If the variant is found in both the refined VCF file and the benchmark VCF file, it is regarded as a real variant (true positive). If the variant is found in the refined VCF file but not in the benchmark VCF file, it is regarded as an artifactual variant (false positive). Hap.py tool is simply a library in Python to perform this comparison for us and one skilled in the art will recognise that other comparison tool may be used without departing from the disclosure.

Eventually after the neural network model has been trained, the neural network architecture 300 would be able to output a probability value for each variant in the VCF file. The probability refers to the probability of the mutation being real or artifactual (2 class outputs). The last layer of the five-layer fully connected network 360 contains a sigmoid function that maps the scalar output from the previous layer to a value between 0 and 1. This value corresponds to the probability. Usually, a default threshold of 0.5 is used. If the probability value is more than 0.5, the mutation is treated as real. Otherwise, the mutation is treated as artifactual. One skilled in the art will recognise that different threshold can be chosen or experimented to balance the sensitivity of the deep learning module without departing from the disclosure.

After step 220, process 200 generates the report 140 or 2 separate VCF files containing the real and artifactual mutations respectively in step 235. The report 140 would contain a list of variants in the VCF file with corresponding predicted mutation which is either real or artifactual mutation. Alternatively, process 200 divides the report 140 to two separate VCF files where one of the VCF files contains a list of variants predicted as real mutation and the other VCF file contains a list of variants predicted as artifactual mutation. In another embodiment, process 200 generates both report 140 and 2 separate VCF files containing the real and artifactual mutations respectively in step 235.

Process 200 ends after step 235.

Deep Learning Module 120

A supervised deep neural network-based method is provided The deep learning module 120 has a comprehensive embedding representation to enhance the algorithm performance.

The deep learning module 120 employs a neural network architecture 300 as shown in figure 3. Figure 2 illustrates the process 200 for preparing the data structures to be fed

to the neural network architecture 300. The neural network architecture 300 receives the first, second and third data structures and predicts 1 of the 2 possible outputs (real or artefactual) for each variant in step 220.

In the training phase, we adopted the Hap.py tool to process the original annotation file as the ground-truth labels in this study. Specifically, we have our own refined VCF file produced by the deep learning module and a benchmark VCF file which contains the ground-truth labels (correct answer). If the variant is found in both the refined VCF file and the benchmark VCF file, it is regarded as a real variant (true positive). If the variant is found in the refined VCF file but not in the benchmark VCF file, it is regarded as an artifactual variant (false positive). Hap.py tool is simply a library in Python to perform this comparison for us and one skilled in the art will recognise that other comparison tool may be used without departing from the disclosure. The model may need to be retrained after a predetermined time period of if the COSMIC database has been updated. One skilled in the art will recognise that this is a design choice left to those skilled in the art.

Figure 3 illustrates the neural network architecture 300 of the deep learning module 120. A learnable embedding layer to map the 96-dimensional mutation type into a high-dimensional space is employed. Each mutation type 310 is passed into an embedding layer 311 and converted into a point in the high-dimensional space, the coordinates of which constitute an embedding vector for the type. The parsed site features including INFO 320 and SAMPLE 330 are combined and fed into a linear layer 340. As mentioned above, INFO 320 and SAMPLE 330 correspond to the "INFO" and "SAMPLE" columns from the provided VCF file. The reference context feature 350 is passed into another embedding layer 351, converting the 21-dimensional vector into a two-dimensional context matrix. In this regard, we used a two-layer 1D-convolution 352 to model the multiple sequences and extract the backward and forward relationships in the context matrix. The context matrix was transformed into a vector.

After extracting the mutation types, site features, and reference context features, the embedding vectors are concatenated in the hidden state 361 and fed into a five-layer fully connected network 360 to output a probability value for each variant. The probability refers to the probability of the mutation of each variant being real or artifactual (2 class outputs). The last layer of the five-layer fully connected network 360 contains a sigmoid function that maps the scalar output from the previous layer to a value between 0 and 1. This value corresponds to the probability. Usually, a default threshold of 0.5 is used. If the probability value is more than 0.5, the mutation is treated as real. Otherwise, the mutation is treated as artifactual. One skilled in the art will recognise that different threshold can be

chosen or experimented to balance the sensitivity of the deep learning module without departing from the disclosure.

Each layer in the five-layer fully connected network 360 consists of dropout linear transformation, ELU (Exponential Linear Unit) activation function, and highway module.

The dropout linear transformation was used at the beginning of each layer to randomly set the unit weights to 0, helping the model avoid overfitting specific features. Equations (1), (2), and (3) show the formulas for the dropout linear transformation, activation function, and highway module respectively:

$$Linear_i(X) = W_iX + b_i \quad (1)$$

$$ELU(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (\exp(x) - 1), & \text{if } x \leq 0 \end{cases} \quad (2)$$

$$H_{i+1} = H_i + ELU(Linear(H_i)) \quad (3)$$

where W and b are learnable parameters. The ELU hyperparameter α controls the value at which the activation function saturates for negative inputs. The highway module connects the input of each layer to the end, solving the gradient vanishing problem caused by overly deep networks.

3) *Cost Function*: Equation (17) defines the cost function.

$$L = L_{BCE} + \gamma L_{L2} \quad (4)$$

L_{BCE} refers to the binary cross-entropy loss while γL_{L2} is the L2 regularization which adds a penalty term to reduce overfitting. This cost function is used for training purposes only. We adopted the mini-batch gradient descent algorithm for training and each batch consisted of 5000 variants. The Adam optimizer was used to update the network weights. All hyperparameters were tuned using grid search and evaluated on the validation set. The Genome in a Bottle (GIAB) consortium reference samples (HG001-HG007) were used for training and testing. Two different training methods were used. The first method built 1 model using 1 VCF file. Within each file, the SNPs were split such that 60% of them were part of the train set, 20% of them were part of the validation set, and 20% of them were part of the test set. The second method used all the SNPs from 1 VCF file and tested them on all the remaining files. For example, a model was trained on all the SNPs from the HG001 VCF file and evaluated on the SNPs from the HG002-HG007 files. Table 1 shows the final values chosen for the hyperparameters.

Hyperparameter	Values
Context width	21
Base embedding size	3
Mutation type embedding size	20
CNN layer	2

CNN kernel	3x3
MLP layer	5
ELU	1
Max epoch	200
Learningrate	1.00E-04
L2 penalty	1.00E-05
Dropout	0.1

Table 1

J. Evaluation Data Sources and Processing

The performance and speed of FINDEL have been evaluated using high-quality open-source datasets. For the training and validation, 4 different datasets containing a total of 33 VCF files have been used.

1) *HCC1954 Cell Line*: This breast cancer cell line whole genome sequencing (WGS) data from a 61 years old Asian female was initiated on October 30 1995 and took around 4 months to establish. We downloaded the VCF file and the benchmarking results from the International Cancer Genome Consortium Data Portal. The reads were aligned to the hs37d5 reference genome. The benchmarking results containing the true positive and false positive variants were stored in 2 mutation annotation format (MAF) files which are tab-delimited text files with aggregated mutation information on a project level. No processing was required since the VCF file was provided directly. A total of 1 VCF file was obtained.

2) *Formalin-Fixed Paraffin-Embedded (FFPE) and Fresh Frozen Whole Exome Sequencing (WES) Samples*: This study was part of an attempt by Samsung Medical Center to compare the WES data between FFPE and fresh frozen samples. The datasets were obtained from the Sequence Read Archive (SRA) with the accession number PRJNA301548. The samples were sequenced using the Illumina HiSeq 2000 sequencing system. We used the DNA sequencing data from the following 3 runs: SRR2911437, SRR2911438, and SRR2911453. The paired-end reads were aligned to the HG19 reference genome using the Burrows-Wheeler Alignment tool with the Maximal Exact Match option, more commonly known as the BWA-MEM algorithm. After the reads were aligned, Mutect2 was chosen for the variant calling process. The tumor-only mode with default settings was used. A total of 3 VCF files was obtained.

3) *Whole Exome Sequencing of Biological and Technical Replicates in Breast Cancer Samples*: This study was conducted by the Memorial Sloan Kettering Cancer Center on 3 Feb 2016. The datasets were obtained from the SRA with the accession

number SRP070662. The samples were sequenced using the Ion PGM sequencing system. We used the DNA sequencing data from the following 22 runs: SRR3182418, SRR3182419, SRR3182421, SRR3182423, SRR3182424, SRR3182427, SRR3182429, SRR3182430, SRR3182431, SRR3182432, SRR3182436, SRR3182437, SRR3182440, SRR3182442, SRR3182444, SRR3182446, SRR3182447, SRR3182474, SRR3182475, SRR3182476, SRR3182477, and SRR3182478. The processing steps are the same as the FFPE and fresh frozen WES datasets. A total of 22 VCF files was obtained.

4) *Genome in a Bottle (GIAB) Consortium Reference Samples*: The GIAB consortium is jointly hosted by the National Institute of Standards and Technology (NIST) and the Joint Initiative for Metrology in Biology (JIMB) to develop the technical infrastructure and improve clinical insights from whole human genome sequencing. GIAB has a total of 7 human genomes: 1 pilot genome of Utah/European ancestry (NA12878/HG001) from the HapMap project, and 2 son/father/mother trios of Ashkenazi Jewish (HG002/HG003/HG004) and Han Chinese (HG005/HG006/HG007) ancestry from the Personal Genome Project. Each human genome is also accompanied by benchmark variant calls and regions to validate variant calling pipelines. These are considered the gold standard benchmarks within the bioinformatics community. The processing steps are mostly the same as the FFPE and fresh frozen WES datasets with some differences. The b37 reference genome was used in place of the HG19 reference genome. Both originate from the GRCh37 reference genome with some minor differences. However, from a bioinformatics pipeline technical perspective, these variations of reference genomes are not directly interchangeable due to contig name changes. The change was due to the addition of the germline resource and the panel of normals (PoN) during the Mutect2 variant calling process. Both VCF files used the b37 reference genome. Therefore, the alignment phase required the use of the same reference genome as well. A total of 7 VCF files was obtained.

Test results

The FINDEL system 100 has been evaluated based on its performance and speed on the evaluation data. For performance evaluation, benchmark VCF files are required as they contain the ground truth variant calls. The datasets without benchmark VCF files will be used for speed evaluation. The following 5 metrics will be used for performance evaluation:

$$precision = \frac{TP}{TP+FP} \quad (18)$$

$$recall = \frac{TP}{TP+FN} \quad (19)$$

$$F1 = \frac{2*precision*recall}{precision+recall} \quad (20)$$

$$accuracy = \frac{TP+TN}{TP+FN+TN+FP} \quad (21)$$

$$specificity = \frac{TN}{TN+FP} \quad (22)$$

where TP refers to true positive, TN refers to true negative, FP refers to false positive, and FN refers to false negative. Elapsed time, in either minutes or seconds, will be used for comparing the speed of the models. From now on, “PY AR” will be used to refer to the Python model of the known method of using the 4 formulations while “DL” will be used to refer to the system 100. These terms are mainly used in the comparison of the model results pre and post-deep learning.

The initial version of the DL model was trained on the HCC1954 cell line dataset. Both performance and speed results of PY AR and DL are shown in Table 2. The elapsed time of PY AR is less than 3 minutes while the elapsed time of R AR is 305 minutes. The objective function value of PY AR is 80.34. The higher the value, the better the performance. The DL achieves a significantly higher objective function value of 89.21 with a training and inference time of 20 and less than 3 minutes respectively. However, the performance improvement based on this dataset alone is not robust enough. As seen from the table 2, there is very little labeled data available. Only 214 variants are labeled with 213 of them being classified as positive SNPs and only 1 negative SNP. From a supervised deep learning perspective, this is an imbalanced classification problem and the performance metrics might be biased. Even though DL correctly predicted 211 out of 213 positives SNPs, it failed to predict the only negative SNP. It can maximize the objective function value simply by predicting all SNPs to be positive as they are the majority of the labels. This poses a problem when we want to minimize false positives. On the other hand, PY AR can detect the only negative SNP but misses much more positive SNPs, achieving an overall lower objective function value. Therefore, the other labeled datasets are used to provide a more robust estimate of the model performances. The key takeaway here is the higher performance of the DL.

Model	Programming Language	Objective Function Value	Correctly Predicted / Total Positive SNPs	Correctly Predicted / Total Negative SNPs	Elapsed Time (Min)
PY AR (Before deep learning)	Python	80.34	201/213	1/1	<3

DL (After deep learning)	Python	89.21	211/213	0/1	Training 20
					Inference < 3

Table 2

The subsequent versions of the DL models were trained on the GIAB consortium reference samples (HG001-HG007). The following results were based on the first method of training using the same VCF file for both training and testing. The performance comparison between PR AR and DL based on the 5 metrics is given in figures 4.1 to 4.5. In figures 4.1 to 4.5, all values are in percentages unless otherwise indicated. All datasets used are from the GIAB consortium reference samples (HG001-HG007). Figure 4.1 shows a bar plot comparing the F1 score. As shown in figure 4.1, DL achieved higher score than PY AR on all datasets. Figure 4.2 shows a bar plot comparing the accuracy score. As shown in figure 4.2, DL achieved higher score than PY AR on all datasets. Figure 4.3 shows a bar plot comparing the precision score. As shown in figure 4.3, DL achieved higher score than PY AR on all datasets. Figure 4.4 shows a bar plot comparing the recall score. As shown in figure 4.4, PY AR achieved higher score than DL on all datasets. Figure 4.5 shows a bar plot comparing the specificity score. As shown in figure 4.5, DL achieved higher score than PY AR on all datasets. DL performed better on all the metrics (F1, accuracy, precision, specificity) except recall.

The performance comparison within PY AR is shown in figures 5.1 to 5.5. Specifically, figure 5.1 shows a bar plot comparing the F1 score; figure 5.2 shows a bar plot comparing the accuracy score; figure 5.3 shows a bar plot comparing the precision score; figure 5.4 shows a bar plot comparing the recall score; figure 5.5 shows a bar plot comparing the specificity score. Generally, higher scores were achieved on the human genomes from the Ashkenazi Jewish (HG002/HG003/HG004) ancestry than the Utah/European (HG001) and Han Chinese (HG005/HG006/HG007) ancestries for F1, accuracy, and precision. For recall, similar scores were achieved on the human genomes from all the ancestries. Results were mixed for specificity. The performance comparison within DL is given in figures 6.1 to 6.5. Specifically, figure 6.1 shows a bar plot comparing the F1 score; figure 6.2 shows a bar plot comparing the accuracy score; figure 6.3 shows a bar plot comparing the precision score; figure 6.4 shows a bar plot comparing the recall score; figure 6.5 shows a bar plot comparing the specificity score. Generally, higher scores were achieved on the human genomes from the Ashkenazi Jewish ancestry than the Utah/European and Han ancestries for F1, accuracy, and precision, and recall. Results were mixed for specificity.

The next set of results used the second training method where the train and test set consisted of different human genomes. The F1, accuracy, and precision performance comparisons within DL for different train sets (i.e. HG001, HG002, HG003, HG004, HG005, HG006 and HG007 samples) are given in figures 7.1 to 7.7, 8.1 to 8.7 and 9.1 to 9.7 respectively. Generally, higher scores were achieved on the human genomes from the Ashkenazi Jewish ancestry than the Utah/European and Han Chinese ancestries regardless of the train set used. The recall performance comparison is given in figures 10.1 to 10.7. Generally, higher scores were achieved on the human genomes from the Ashkenazi Jewish and Utah/European ancestries than the Han Chinese ancestry regardless of the train set used. The human genome from the Utah/European ancestry (HG001) had the best recall for all the train sets. The specificity performance comparison is given in figures 11.1 to 11.7. Generally, higher scores were achieved on the human genomes from the Ashkenazi Jewish ancestry than the Han Chinese ancestry regardless of the train set used. The human genome from the Utah/European ancestry (HG001) had the worst specificity for all the train sets. The average performance comparison using different train sets is given in figures 12.1 to 12.5. Generally, higher average scores on all 5 metrics were achieved when the human genomes from the Han Chinese ancestry were used as the train samples.

The overall software runtime is low as our approach queries VCF files instead of BAM files where the former has a much lower memory footprint. Moreover, the software can be easily run on a local machine. The system 100 eliminates the need for users to manually inspect BAM files using some form of genomics viewer program. This saves both labor and time, increasing the turnover of cancer sample interpretation. Doctors require less time for the clinical diagnosis of patients and can increase the range of therapeutic opportunities for them. More importantly, the automation and standardization of the variant refinement process increase reproducibility by different entities.

The usage of mutational signatures to partition the unrefined mutation set into real and artifactual mutation subsets was possible as various artifacts had characteristic mutational signatures that were different from the true variants. These mutational signatures are constantly updated and can be found in the COSMIC database. This essentially removes the need for bioinformaticians to manually inspect large BAM files and search for patterns that are representative of artifacts using domain knowledge. The manual inspection process is a major bottleneck impeding the progress of the downstream analyses after the variant calling process. Eliminating this laborious task greatly reduces the entire bioinformatics workflow time.

The deep learning module improves algorithmic performance and further reduces the number of false positives. It also utilizes the information from the VCF files beyond just the mutational signatures. Specifically, additional features are engineered from the "INFO" and "SAMPLE" columns of the VCF files. As seen from the results, the supervised deep learning greatly improves the score of most of the metrics. The disadvantage is that the initial training of the neural network might take quite some time. However, subsequent usage of the model, aka inference, takes little time. The deep learning model only needs to be retrained if there is evidence of model degradation. Also, the performance of the model is affected by the quality of the data. Generally, higher scores were achieved on the human genomes from the Ashkenazi Jewish ancestry than the Utah/European and Han Chinese ancestries due to the former having higher sequencing depth.

The above is a description of exemplary embodiments of a device in accordance with this disclosure. It is foreseeable that those skilled in the art can and will design alternative devices based on this disclosure.

Claims

1. A method of predicting a mutation type of each of the plurality of variants in a single sample file comprising:

5 constructing a first data structure containing features of a plurality of mutation types based on a plurality of mutational signatures;

reconstructing a second data structure containing site features from the single sample file;

10 building a third data structure containing reference context features for each of the plurality of variants in the single sample file;

feeding the first, second and third data structures to a neural network architecture;

receiving from the neural network architecture a probability value for each of the plurality of variants being real or artifactual mutation; and

15 generating a report comprising a list of variants in the single sample file with corresponding one of predicted real and artifactual mutations.

2. The method according to claim 1 wherein the plurality of mutational signatures consists of 78 single-base substitutions (SBS) mutational signatures obtained from a Catalogue Of Somatic Mutations In Cancer (COSMIC) database.

20 3 The method according to claim 1 wherein the 78 SBS mutational signatures are segregated into 96 different contexts to form the first data structure.

4. The method according to claim 3 wherein the step of reconstructing the second data structure containing site features from the single sample file comprises:

25 extracting from an "INFO" column and a "SAMPLE" column from the single sample file to form the second data structure.

5. The method according to claim 4 wherein the step of reconstructing the second data structure containing site features from the single sample file further comprises:

applying one-hot encoding for discrete feature extracted from the single sample file,

5 and

applying standard scaling for continuous feature extracted from the single sample file.

6. The method according to claim 5 wherein the second data structure comprises
10 columns determined by a number of categories in the single sample file and rows determined by a number of rows in each category in the single sample file.

7. The method according to claim 6 wherein the step of building the third data structure containing reference context features for each of the plurality of variants in the single
15 sample file comprises:

locating a plurality of mutation sites in a reference sequence file that correspond to each of the plurality of variants, the reference sequence file containing a full human genome sequence;

retrieving 10 bases at the left and right of each located mutation site in the reference
20 sequence file to form a total of 21 bases for each of the plurality of variants; and

translating the 21 bases to form a 21 dimensional vector for each of the plurality of variants to form the third data structure.

8. The method according to claim 7 wherein the step of feeding the first, second and
25 third data structures to the neural network architecture comprises:

passing the first data structure into a first embedding layer and converting into a point in a high-dimensional space, coordinates of which constitute an embedding vector for the type;

parsing the second data structure into a linear layer;

5 passing the third data structure into a second embedding layer and converting the 21-dimensional vector for each of the plurality of variants into a two-dimensional context matrix;

concatenating the outputs from the first embedding layer, linear layer and second embedding layer and feeding into a five-layer fully connected network, each layer in the
10 five-layer fully connected network consists of dropout linear transformation, ELU (Exponential Linear Unit) activation function, and highway module with the following expressions,

$$\begin{aligned} \text{Linear}_i(X) &= W_i X + b_i \\ \text{ELU}(x) &= \begin{cases} x, & \text{if } x > 0 \\ \alpha * (\exp(x) - 1), & \text{if } x \leq 0 \end{cases} \\ 15 \quad H_{i+1} &= H_i + \text{ELU}(\text{Linear}(H_i)) \end{aligned}$$

where W and b are learnable parameters, the ELU hyperparameter α controls a value at which the activation function saturates for negative inputs, the highway module connects the input of each layer to the end, solving the gradient vanishing problem caused by overly deep networks, wherein a last layer of the five-layer fully connected network
20 contains a sigmoid function that maps the scalar output from the previous layer to a value between 0 and 1 for each of the plurality of variants.

9. The method according to claim 8 further comprising:

determining a real mutation in response to the value being more than a predefined
25 threshold; and

determine an artifactual mutation in response to the value being equal to or less than the predefined threshold.

10. The method according to claim 9 wherein the predefined threshold is 0.5.