This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

Machine Learned Replacement of N-Labels for Basecalled Sequences in DNA Barcoding

Eddie Y. T. Ma, Sujeevan Ratnasingham, and Stefan C. Kremer

Abstract—This study presents a machine learning method that increases the number of identified bases in Sanger Sequencing. The system post-processes a KB basecalled chromatogram. It selects a recoverable subset of N-labels in the KB-called chromatogram to replace with basecalls (A,C,G,T). An N-label correction is defined given an additional read of the same sequence, and a human finished sequence. Corrections are added to the dataset when an alignment determines the additional read and human agree on the identity of the N-label. KB must also rate the replacement with quality value of > 60 in the additional read. Corrections are only available during system training. Developing the system, nearly 850 000 N-labels are obtained from Barcode of Life Datasystems, the premier database of genetic markers called DNA Barcodes. Increasing the number of correct bases improves reference sequence reliability, increases sequence identification accuracy, and assures analysis correctness. Keeping with barcoding standards, our system maintains an error rate of $< 1\%$. Our system only applies corrections when it estimates low rate of error. Tested on this data, our automation selects and recovers: 79% of N-labels from COI (animal barcode); 80% from matK and rbcL (plant barcodes); and 58% from non-protein-coding sequences (across eukaryotes).

Index Terms—Bioinformatics (genome or protein) databases, Decision support, Machine learning, Biology and genetics, Neural nets

1 Introduction

This study develops and tests a system that automatically replaces ambiguous characters in a DNA sequence resulting from a DNA sequencing technology called Sanger Sequencing [1]. Finding the DNA sequence from a biological sample is called DNA sequencing. In this study, the ambiguous N-label from Sanger Sequencing is replaced with a discrete DNA base character (A, C, G, T). Data recorded from a Sanger Sequencing run is a timeseries plot composed of a four-channels; one channel for each base type (A, C, G, T), a chromatogram. A sequence of peaks in the chromatogram indicates the sequence of nucleotides present in the sampled DNA sequence. The problem of converting the chromatogram into a DNA sequence composed of A, C, G, T is termed *basecalling*, and is performed by basecalling algorithms. A particularly well-cited basecaller is PHRED [2], [3]. In addition to providing base labels for peaks in a chromatogram, PHRED also emits a quality value for each basecall representing an estimation of error. This PHRED score (Quality Value) indicates the probability of error stated as $QV = -10\log_{10}(p(\text{error}))$; i.e. every increase in ten is a fall of an order of magnitude in error. Besides the discrete labels A, C, G, T; sequences basecalled can also contain degenerate characters that represent more than one base at a single location. The degenerate label N represents any of the four bases. When the estimation of error is too high for a basecall in a given peak location, PHRED can label that base location with an N representing an ambiguity, rather than risk incorrectly assigning a discrete base label to that peak. The next step to obtain the DNA sequence from a sample is called *sequence finishing* and requires human intervention. This step is needed when there is ambiguity in a sequence, or if the sequence is not long enough for a particular application. Sequence finishing may include the use of additional Sanger Sequencing runs of the same sample of DNA resulting in additional chromatograms, such that ambiguities in parts of one chromatogram are clarified in the other. After these chromatograms are basecalled, the resulting sequences are aligned together. Using these additional chromatograms and basecalled sequences, a human editor may examine and manually edit the assembled sequence for any errors. A human may also apply contextualizing information to assist in error correction related to the sample, such as knowledge of biologically related specimens, knowledge of the reading frame, or subsequent amino acid composition of a translated DNA sequence. N-labels may now be replaced by the human editor with unambiguous base labels, and a final finished sequence can be produced. Additional human editing actions can also be performed during finishing, such as *deletion* of extra labels where no base exists, and *insertion* of new labels at peaks undetected by a basecaller.

This study uses human curated sequences for genetic markers and their corresponding chromatogram tracefiles from Barcode of Life Datasystems (BOLD) [4]. These sequences are genetic markers, the vast majority of which are recognized as DNA Barcodes. DNA Barcodes are used for rapidly identifying or verifying the species of an unknown specimen using only a short segment of DNA [5]. There are over 4.8 million barcode sequences in BOLD (as of May 2016; http://boldsystems.org). DNA Barcode data is chosen for its abundance and public accessibility. It has a combination of both curated and raw data needed for the development of an automation. Due to its volume use of

- *E Ma and SC Kremer are with the School of Computer Science, University of Guelph, Guelph, Ontario, Canada.*
  E Ma's E-mail: ema@uoguelph.ca
- *S Ratnasingham is with the Biodiversity Institute of Ontario, University of Guelph, Guelph, Ontario, Canada*

Manuscript received —, 2015; revised —, 2016.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

Sanger Sequencing, it is the application that stands to benefit the most from this study. Most of the tracefiles indexed in BOLD have been basecalled with KB basecaller (v1.0 – v1.4.1; ABI, Foster City, California, USA), a basecaller that is based on PHRED. KB inherits the quality value established by PHRED. KB has been shown to modestly outperform PHRED in terms of the length of the recovered sequence, but the difference does not appear to affect the identification of species based on genetic markers (e.g. 16S ribosomal marker in microbes [6]). A selection from BOLD is obtained in order to build the dataset to perform this work. In total, nearly 850 000 N-labels are extracted from the data and included in this study.

To decide which N-labels can be used in training and testing our system in this study, it is necessary to evaluate what replacements should be considered correct. A multiple global sequence alignment [7] is done for: (1) the tracefile to edit, presented to the system during training and testing; (2) a sequence from a second tracefile for another Sanger Sequencing read of the same sample, which will supply the target correction for replacing N-labels; and (3) the corresponding human edited and finished sequence for this sample, which confirms the accuracy of the N-label corrections from the second tracefile. N-labels in the tracefile to edit are only considered suitable for editing if the second tracefile sequence has a basecall with QV > 60 (i.e. $p(\text{error}) < 10^{-6}$) aligned at that position, and if that basecall is also in agreement with the human sequence. These correct base labels (from items 2, and 3 above) are only available during training, and are unknown to the classifier during testing or in a production system (Figure 1). This study targets the automated replacement of N-labels emitted from the KB basecaller. The criteria described above for inclusion of N-labels in the dataset are motivated by the need to make the most accurate basecalls based on training our system on highly reliable ground-truth data. The best outcome is for a peak to be labelled correctly. As a fallback from emitting a base label with confidence, the N-label is used when there is uncertainty. While this is not the most desirable outcome, it is better than the third option – the emission of an incorrect basecall at a peak. The ordering of preference over these outcomes is shared by PHRED and KB, in that N-labels are preferred to base labels emitted with too low a confidence. This preference is inherited by our system. The direct assessment and replacement of N-labels as a post-processing of KB basecaller is a novelty of this work.

The process of alignment, and selecting N-labels for training and testing is repeated exhaustively for every pair of tracefiles corresponding to every sequence sampled in the dataset to ensure maximum opportunity to extract all N-labels that can be edited. The data is broken apart into three disjoint subsets for: training, validation, and testing. So far, we can keep our description of the system to develop generic, as we believe that nearly any system that fits under the broad umbrella of regression-based machine-learned classifiers can satisfy the needs of the problem. To go forward we must also consider the general needs of such a classifier.

Machine learned classifiers must be trained on a wide variety of data to be able to generalize onto unseen data. The total set of N-labels selected must contain sufficient variety in order to be considered a representative sample of what may be encountered in production. Additional details regarding the spread and characteristics of the data are explored in the section Application.

Beyond DNA Barcoding, applications that utilize Sanger Sequencing include forensics [8], [9], [10], medical diagnostics [11], [12], [13], [14], and personalized medicine [15], [16]. Like DNA Barcoding, these applications all have specific requirements and data quality standards meeting the needs of each their scientific question. In future, it is possible that the present system can be generalized onto these other applications, and validated using the quality control criteria that have been established by each community.

2 Data Representation

The machine learned classifier in this study must accommodate input data from windows of a tracefile, and must be capable of emitting output predictions for the central bases in that window. The classifier must be capable of coping with a real-valued input vector that represents both discrete values, as well as continuous values. Discrete values represent the basecalls from KB, and continuous values include KB quality values, and also scaled chromatogram values for a tracefile to edit. The basecalls (including N-labels to correct) are changed to a one-hot six-input encoding; i.e., each base occupies six input values, and is set to the value ON in vector positions corresponding to $/ACGTN-/$ (where '-' is the gap character), and the value OFF in all remaining input positions. Quality values are linearly scaled from the range $[0, 100]$ into a reasonable input range for the classifier, and occupy one input each. Finally, chromatogram values in the window are linearly scaled into the input range. The minimum and maximum value of this scaling is taken over all four chromatogram channels to preserve relative peak heights. The input window size is 9-bases, with 12-chromatogram samples per base.

The nine-input design was chosen because peaks near the peak being basecalled have been shown to contain information that helps identify the current base [17], [18], [19].

The output vector emitted by the classifier encodes information predicting the state of five adjacent peaks in the centre of the above nine-base window. Each of these five peaks has its own set of outputs that state what to replace an N-label with (one-hot encoding, $/ACGT/$). An N-label occurring on a peak in a tracefile, being scanned from left to right has five opportunities to be viewed and called by the classifier. Having five outputs all attempting to solve the same N-label replacement problem, each with a slightly different view of the input chromatogram is one way to set up a committee of learned predictors whose members evaluate different evidence (Figure 2).

Ensembles of predictors have been shown to outperform single predictors with only minor impact on predictor bias, as seen frequently in the literature [20], [21], [22].

In the present study, two ensemble classifiers are trained, tested and compared. The ensembles are composed of the Support Vector Machine (SVM) [23] [24], and the Artificial

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

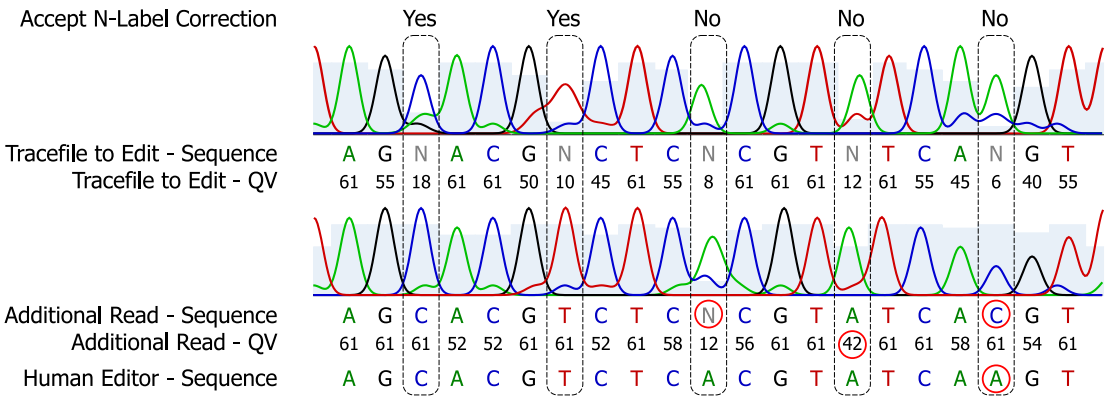

Fig. 1. Schematic examples for accepting N-label corrections as data to use for training and testing. The first two examples (*in dashed lines*) are admissible, since an N-label is being corrected by a base that is agreed upon by the human and a second read with a quality value >60. The last three examples have the circled problems (*resp.*): (1) the basecaller has left the base ambiguous, (2) the second read only has a quality value of 42, and (3) the human and second read disagree on the identity of the correct base.

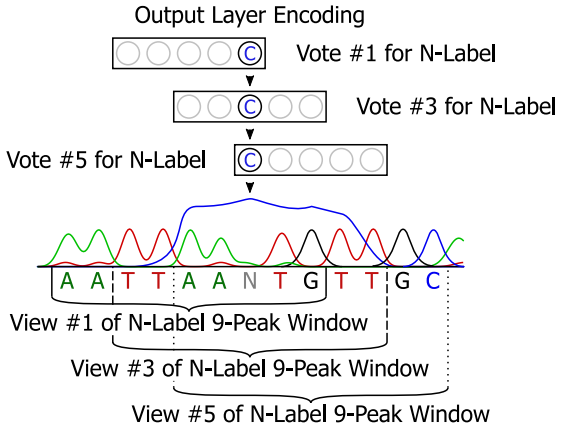

Fig. 2. A schematic of three shifted views of a chromatogram window containing an N-label to be processed by the classifier. There are five possible views (*three shown*), allowing the classifier to accumulate five votes for replacing an N-label. Each circle in the output layer represents the output encodings: (1) whether there is an N-label at a peak (ANN only), and (2) what basecall to label that peak (SVM and ANN).

Neural Network (ANN) [25], [26], [27], [28]. The SVM ensemble achieved a mean error rate of 0.98% and the ANN ensemble achived a mean rate of 0.94%. As the accuracy scores for the two systems are highly similar, we choose our preferred system, the ANN to refine further with the addition of error estimation, given classifier uncertainty.

Some differences exist in the training of the SVM and ANN. For SVM, the implementation used is the Scikit-learn [29] wrapper for libsvm [30] with Radial Basis Function (*RBF*) kernel. The basic SVM classifier component as per libsvm is a binary classifier; to distinguish between four classes, Scikit-learn internally deploys a total of six *one-against-one* classifiers to do multiclass classification when there are four possible N-label replacements A, C, G, T (i.e. $\frac{4\times3}{2}$). Since the ensemble classifiers implemented in this work must process five views to evaluate each N-label, the final SVM ensemble deployed is internally, a total of thirty SVMs.

An SVM using *RBF* kernel has two parameters that can be tuned to improve performance, they are C and γ. Increasing the term C indicates a preference for classification accuracy, over the margin for class separation by the classification hyperplane [23], and γ states the *peakiness* or inverse-spread of the *RBF* kernel [30]. An exhaustive gridsearch is conducted to find the best values for C and γ. A range of values for C and γ are tested for best mean performance using three-fold cross-validation on solely the training set. The entire ensemble is trained during gridsearch. In three-fold cross-validation, two-thirds of the training set are used to train the ensemble, while the remaining third is used to validate performance; this is repeated for every combination of C and γ. The exhastive search took these ranges, in a log-ten scale: $C \in \{10^{-2}-10^{6}\}$ in, and $\gamma \in \{10^{-6}-10^{-2}\}$. The final combination of parameters with the lowest mean cross-validation error are $C = 10^{6}$, and $\gamma = 10^{-4}$. Since the best performance for C occurs in the edge of the gridsearch, a second smaller search is conducted for $C \in \{10^{7}-10^{9}\}$ with $\gamma = 10^{-4}$. This results in identical performance as $C \in 10^{6}$, indicating that further narrowing of the SVM classification margin will not produce any higher an accuracy.

After these parameters are found, one final round of training is conducted using the entirety of the training set. The final trained ensemble is then tested and compared against ANN. For ANN, a complete gridsearch for parameters is not conducted. Some tuning is done to find a set of parameters that would yield a converging network. Since final performance of ANN is comparable to SVM, no additional formal searching is performed.

The implementation used for ANN is an in-house library written in C and Python. Output units are also expanded as follows. The five N-label replacements contribute to only half of the output vector. An additional set of five outputs describe whether there is an N-label to replace (boolean). This boolean is an intermediate training objective that is found to reduce convergence time and improve N-label replacement accuracy in this study. Furthermore, in contrast to the SVM ensemble of thirty – the ANN is one single regression device where the output layer represents the output values for each the five positions represented in the chromatogram in turn; i.e., a single collection of weight layers computes the solution for each of the five views of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

the N-label. This design is motivated by the observation that neural networks perform better in accuracy when they are trained with additional output units that operate on related tasks and subtasks [31], [32]. The general hypothesis behind this observation is that there could be an unseen factorization of the problem that breaks the problem down into simpler, reused components for the ANN to learn. Our ensemble design is similar but not the same as the method of dropout neurons wherein elements of a hidden layer are randomly disconnected during training to achieve an ensemble [33]. The use of an SVM or ANN ensemble to emit basecalls from multiple views of the same chromatogram fragment is a novelty of this work.

2.1 History of Neural Networks in Basecalling

The artificial neural network (ANN) used is the feed-forward backpropagation-of-error artificial neural network with two hidden layers [25], [26], [27], [28]. The artificial neural network was chosen for its capability as a universal function approximator, and ability to learn the highly non-linear data that describes chromatograms. Previous studies have used neural networks as a timeseries feature extractor, or as the classifier component of a basecaller [17], [19], [34], [35], [36], [37], [38].

In early studies [34], [35], a neural network was trained and used as a preprocessor that converts a raw signal to a normalized chromatogram. In these early works, hardware limitations dictated the necessity for cleverly reduced representation of the timeseries. The preprocessor ANN consists of only one layer. The input consists of a span of only three time indices, for each of the four channels A, C, G, T (twelve units). To compute some of the pairwise interactions available only with a hidden layer, the pairwise products of those initial twelve units are concatenated to the input vector ($12 + \frac{12\times 11}{2} = 78$ total input units). The output of the preconditioner is a one-hot encoding in A, C, G, T. Datasets used to train this system consisted of three chromatograms each – while direct performance values were not reported in [34], its was found that the ANN was robust to varied amounts of training iterations (50000 to 100000 epochs) in that accuracy improved without showing signs of overfitting the training data. This is encouraging to our study, as generalizability of a trained ANN is necessary for accurate labelling of chromatograms originating from vastly different genetic markers.

The ANN has also been used as a feature extractor for a Hidden Markov Model (HMM) [36]. While the ANN is a competent function approximator on its own, it makes an implied assumption about the conditional independence of its outputs – specifically, there is no explicit modelling of the dependence between adjacent basecalls (although such effects can be made implicit by wide overlapping input windows). In [36], the ANN is used as a preprocessor for an HMM that is then able to model the statistical dependency between adjacent basecalling events. In summary, this is done by treating the ANN as a mapping between the chromatogram real-values and the HMM emission densities, as a posterior to the chain of HMM hidden states. The ANN in that study was given a 33-sample window in each channel $ACGT$ (enough for a window of roughly three-peaks); and it had three hidden layers of sizes 120, 60, and 12. It was found that the HMM performance is comparable with PHRED (using a test set of ten chromatograms). One question raised is whether our study could benefit from explicit state dependency modelling. Given the added complexity to achieve an explicit model of state-dependency, and the comparability of the above study results to PHRED, we actively forgo using HMMs in our present study. Another question raised is whether the ANN itself is too complicated a preprocessor. In an earlier work by the same authors [39], it is demonstrated that a simpler representation, the Gaussian Mixture Model (GMM) does yield worse performance.

While [34], [35] noted that salient information can be gathered about a specific peak to basecall from its adjacent, and nearby peaks – it was not until [17], [19] that the utility of that information was directly tested. In those two studies, 100 tracefiles were used for training, 100 for validation (determining when to stop training), and 1000 were used for testing. By contrast, we used over 160 000 tracefiles in our study. In [17], an ANN was trained (amongst other methods) to perform basecalling using only information extracted in the peaks surrounding a given base. In a window of five peaks, indexed $1-5$, the base to label was at index 2, and contextual information was provided from peaks $1, 3, 4, 5$. That achieved a classification accuracy of 78%. A followup study [19] used a guess-and-check paradigm (*abductive reasoning*) for basecalling. In it, the window included information from nine peaks, with information omitted from the central peak (the peak to label). That design achieved a rate of 86% accuracy using ANN (amongst other methods). This demonstration motivates the use of wide chromatogram windows in the present study – although, we concede to admitting data from the most informative peak of all: the central peak.

Finally, a set of studies was performed that revisits the viability of ANN (amongst other methods) as a basecaller [37], [38], two decades after the first use of ANNs for basecalling [34]. Those two studies are unique in that the role of the ANN is reduced only to classification – all of the preprocessing (e.g. noise reduction, feature extraction) is performed by other learned functions. In the above, a neural network processes only a single time index within the chromatogram at a time and indicates the presence and identity, or absence of a base at that index. Extracted features are (1) the signal strength at the current time index, (2) the numerical gradient for three time indices *before* this time index, and (3) the numerical gradient for three time indices *after* this time index (i.e. 3 values per channel $ACGT$; 12 input elements total). The result is a very light-weight ANN that offloads much of its feature extraction functionality to preprocessors. The final accuracy of the ANN was 98.4%. This result however, was obtained given a dataset of only 22 tracefiles, wherein performance was evaluated using a leave-one-out design using yet smaller subsets of that data.

Our present work is set apart from any of the previous ANN-based basecallers and any other machine learned basecallers for Sanger Sequencing chromatograms that we know of (e.g. [18], [40], [41], [42]), in that our system only processes the peaks labelled as 'N' by KB – a difficult subset of the basecalling problem. By doing so, our baseline in performance accuracy *is* the KB basecaller, and we can only

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

improve upon it by recovering more N-labels. Our study benefits from a dataset that has over a hundred times more chromatograms than previous studies involving ANNs in basecalling. It takes full advantage of the availability of modern hardware by loading both feature extraction and classification onto the ANN, and furthermore represents the N-label replacement problem using a wide input chromatogram window and a five-fold output. The present work is a logical continuation of the use of ANN in basecalling as ANN is here, only a post-processor. While our method benefits from inputting KB basecalls for neighbouring peaks, it is tasked with inferring the identities of the noisiest peaks in a chromatogram. Faster training algorithms have been developed since classic backpropagation for ANNs, that adapt the learning rate (e.g. [43]), descend down higher orders of the error gradient (e.g. [44]), or pretrain weights for deep learning (e.g. [45]). These more complex methods are not tested in this study, as we find that simple gradient descent works surprisingly well.

2.2 Error Estimation with Artificial Neural Networks

One of the greatest advances of modern basecallers is the ability to estimate the probability of error of an emitted basecall; the most well cited example of error estimation is the PHRED score [2]. The PHRED score has been empirically demonstrated to correlate well with observed error [3]. Inspired by this advance, and the need to limit the rate of our own erroneous basecalls, our neural network must provide its own estimation of error. The output elements of most feedforward networks can be considered an estimator for a mean value output vector ($y \in R^n$) conditioned on the input vector ($x \in R^n$) – a conditional mean ($\mu_{y|x}$) [46]. Using a method developed by Zhao [47], it is possible to augment the output vector of a neural network to also estimate the conditional variance (${\sigma^2}_{y|x}$), where each conditional mean output shares an index with one conditional variance unit. Since error and variance both measure the deviation of the mean of the output, it is reasonable to suggest that an error estimation heuristic could be built upon learned output variance. First, we will describe our innovation obtaining an estimation of error from variance, which then motivates a summary for obtaining the variance from a neural network.

The output layer of our network uses a one-hot encoding to label $/ACGT/$, so there are four conditional mean output values for each prediction. Each N-label is called five-times, once for each shifted, partially overlapping chromatogram window. These five calls are consolidated into a single basecall (B) for that N-label. An error occurs when a basecall is not the same as the target base considered correct obtained by alignment ($B^{(c)}$). In total, twenty conditional mean units produce the output of a single N-label. The corresponding twenty conditional variance nodes produce the estimated variance values.

These twenty values are averaged into a single variance value (v), used to create the heuristic error estimation function ($f(r)$). To create an error estimation heuristic, variance values are obtained in association to the basecalls made on a validation set. The variance values of that set are then sorted in ascending order, and assigned an ordinal rank (r). The basecalls associated with each variance value can be correct or incorrect. This allows the proportion of erroneous basecalls to be accumulated on ascending variance. The error estimated is specifically the cumulative error (e_r) associated with each rank of variance. Cumulative error was chosen because it is robust to changes in slope, and conservatively *overestimates* error on average. Formally, the equation to estimate error at a rank of variance is

$$e_r \approx f(r) = \frac{|\{B_i | v_i < r \wedge B_i \neq B_i^{(c)}\}|}{|\{B_i | v_i < r\}|} \quad (1)$$

In summary, the cumulative error (e_r) is the count of incorrect basecalls ($B_i \neq B_i^{(c)}$) emitted with a variance value below the queried rank of variance ($v_i < r$) as a proportion of the total number of basecalls emitted below that same rank of variance. When a rank of variance is queried that is not contained in the data used to build this function, the neighbouring higher error is returned (estimating higher error is safer). Since fitting a curve to predict error on the same data as the training set could potentially lead to a poor heuristic that overfits the training set, the validation set is used instead, as it is disjoint from both the training set and all test sets. While Zhao [47] developed the variance output for neural networks, converting it into an estimate of cumulative error is a novelty of this work.

While we have made the choice to continue on with error estimation using Artificial Neural Networks here, it should be noted that it is entirely possible to complete this exercise using the SVM system instead. The scikit-learn implementation can emit an error probability based upon an internal crossvalidation of the training data. Since this given error probability is based upon fitted data, it may not be immediately suitable for use. To convert the given error probability into a prediction, the same method of treating that probability as a generic measure of uncertainty allows a numerical fit to be performed on a separate set of data (the validation set would be suitable here too, as it is for ANN).

A brief description for using and training the ANN using gradient descent follows, culminating in training the conditional variance outputs. Fuller details including bias units, momentum, and multilayer back propagation are ordinary (e.g. [48]) and will be skipped here (parameters are summarized in Methods, Table 2).

To begin, training for either the conditional mean or conditional variance output elements only involves changing the equation needed to update the weight layer W. Training the conditional variance elements is similar to the mean elements in that a gradient descent can be used to reduce output error at each training iteration [47]. The activation vector a at the output layer of an ANN is parameterized by a weight matrix ($W \in R^n \times R^n$) by $a = f(\text{net})$ and $\text{net} = Wa^{(h)}$ where the function f is a transfer function ($f : R^n \rightarrow R^n$), and net is the matrix product of a weight matrix W and the activation of the connected hidden layer ($a^{(h)} \in R^n$). To train the network, the weight matrix W is updated by adding differences ΔW that reduce the error of network output ($W \leftarrow W + \Delta W$). The corresponding delta rule is $\Delta W = \eta \delta a^{(h)}$ where η is the learning rate ($\eta \in [0.0, 1.0]$), and δ is the change needed to reduce error ($\delta \in R^n$). The output vector of the network is a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

concatenation of a vector of *conditional mean* outputs and *conditional variance* outputs $a = \langle a^{(\mu)}, a^{(\sigma^2)} \rangle$ (Figure 3).

The *conditional mean* outputs $a^{(\mu)}$ then contribute to their weight updates $\delta^{(\mu)}$ using

$$\delta^{(\mu)} = f'(\text{net}^{(\mu)})(t - a^{(\mu)}) \tag{2}$$

where t is the target output to train toward. The derivative $f'(\text{net})$ is the element-wise derivative of the network activation vector with respect to its matrix product, net; $f'(\text{net}) = \frac{\partial f(\text{net})}{\partial \text{net}}$. The *conditional variance* outputs then contribute their weight updates $\delta^{(\sigma^2)}$ with reference to $\delta^{(\mu)}$ using

$$\delta^{(\sigma^2)} = f'(\text{net}^{(\sigma^2)})((t - a^{(\mu)})^2 - a^{(\sigma^2)}) \tag{3}$$

where the target output (t) to train toward is in this formula, replaced by the variance of the *conditional mean* $(t - a^{(\mu)})^2$. The ANN in this study uses the hyperbolic tangent as the activation function ($f \leftarrow \tanh$) since zero-centred activation functions are known to converge faster [49].

3 Methods

In this section, the data treatment is summarized, neural network training parameters are listed, and the experimental design is described. Training data and testing data are treated differently, in that the neural network is presented with a far more pristine set of data for training, so that we intentionally bias it toward learning how to make accurate predictions without being distracted by noisier patterns of input. By contrast, testing is performed on all available patterns of input regardless of noise. Testing data is thus much more reflective of real-life operational conditions.

3.1 Data Treatment for Training Neural Networks

In the data used in this study, different tracefiles had different QV thresholds for marking a basecall as 'N'. To make this study consistent, all peaks with basecalls of QV < 20 are converted to 'N' at the time data is sampled. As well, any N-labels that are contiguous at the very start or the very end of the tracefile to edit are not included for correction, as the alignment can become unreliable in those cases. Finally, a trimming rule is applied to the beginning and end of the basecalled sequences, such that any window of twenty bases, may contain at most six Ns. The presence of consecutive N-labels at the start and end of a chromatogram are due to physical conditions present in the sequencing reaction or electrophoresis machinery.

To train the network, only the cleanest data is used. That is, all nine bases of the input must have been aligned as either a pair of matching bases, or an N-label to a base (but not another N-label). The training set consists of the 6000 randomly selected tracefiles from each the *Lepidoptera*, *Insecta*, and *Animalia* subsets containing N-labels that meet inclusion criteria (i.e. the aligned sequence from the partner tracefile has a base with QV>60 – and that base is concordant with the aligned human sequence). To evaluate when to stop training, an additional 2000 tracefiles are randomly selected from each of those three subsets to be used as a validation set. The training and validation sets are disjoint from one another. Validation data is used to incrementally monitor the progress of training by testing prediction accuracy, without itself contributing to the tuning of parameters. Training stops when 96% of validation predictions are *perfect* matches with the desired output window – that is, every base in the five-base output window is correctly stated. Table 1 summarizes the training parameters and layer dimensions for the neural network.

3.2 Testing Data Treatment and Interpreting Output

While training the network on pristine data is ideal, the cleanest windows are not a realistic representation of the data observed in real life. Instead, it is practical to know the actual prediction performance of the network when exposed to windows containing the noise observed in real life. For testing, the N-labels meeting criteria for inclusion are presented in five shifted positions, allowing for the emission of an output of that N-label by all five output nodes. The surrounding bases in the remainder of the input 9-base window are not filtered in any way, they can correspond to alignments with substitutions, insertions, deletions, and other N-labels that *do not* meet inclusion criteria. Here, only the output unit involved in predicting the N-label of interest is read. The windows containing the N-labels at each of the five central positions are scanned into the network, and the resulting basecalls are accumulated. In this study, three rules are tested for interpreting the output: *Centre-of-Five*, *Majority Rule*, and *Unanimous*. In the Centre-of-Five (*Centre*) rule, only the middle of the five outputs is used to determine the correct basecall. *Centre* is analogous to using just a single output unit to perform classification. In the *Majority* rule, the basecall made most frequently by the five outputs is accepted. In *Unanimous*, basecalls are only accepted when all five outputs agree. When the five outputs do not agree, then the *N*-label is abandoned and remains an 'N' in the sequence. The proportion of *N*-labels abandoned increases as rules become stricter. *Majority* and *Unanimous* are both examples of ensemble methods.

4 Application

A *DNA Barcode* is a short segment of DNA from a standard gene region, or a non-coding region of DNA, used to identify species of life [5]. The creation of a comprehensive set of barcodes for all species of Eukarya on the planet is the goal of the Barcode of Life project. The most recent estimate in the number of Eukarya on the planet is 8.7 million species [50]. In order to accomplish this goal, methods to improve the rate of DNA sequencing are needed. While Next Generation Sequencing (NGS) [51] methods are one avenue to increase the rate of sequencing, they are more appropriate for genomics applications where high coverage is practical. Applications that require longer read lengths, and faster analysis pipelines benefit more from Sanger Sequencing [52].

This study focuses on automating the finishing activity of replacing N-labels with A, C, G, T labels. The reason why the study focus is set on the replacement of only N-labels as opposed to performing a complete additional round of basecalling is because the vast majority of A, C, G, T labels

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

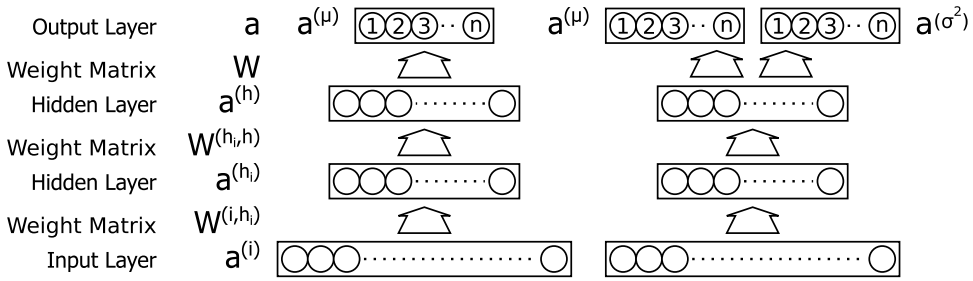

Fig. 3. Schematics for a standard feed forward neural network with only mean output nodes $\mu_{y|x}$ (*left*), and a network with additional conditional variance nodes $\sigma^2_{y|x}$ – one for each conditional mean node (*right*). The schematic shows networks with two hidden layers. Only variables involved with computing output are described in the text, $f(Wa^{(h)}) \rightarrow a$. Other variables describe the input layer $a^{(i)}$, the lower hidden layer $a^{(h_i)}$, and their connections $W^{(i,h_i)}$, $W^{(h_i,h)}$.

TABLE 1
Data representation, training parameters, and layer dimensions for the neural network component.

Parameter	Description	Setting
KB basecall	a six-element one-hot encoding for /ACGTN-/; {OFF,ON} set to:	{-.8,+.8}
KB quality value	an integer value scaled from [0, 100] to:	[-.8,+.8]
chromatogram	a real value scaled from the chromatogram window range [min, max] to the range:	[-.8,+.8]
learning rate (η)	proportion of the gradient to use in a weight update	.0001
momentum (α)	proportion of previous weight update to repeat each update	.9
initialization	range from which to draw random initial weight and bias values	$[-.005, .005]$
training epochs	number of times to loop through all of the training data	200
input layer size	the total number of input units encoding 9-bases of input	499
hidden layers	the number of processing layers between input and output layers	2
hidden layer size	set to one-quarter of the input layer size	124
output layer size	the total number of output units encoding 5-bases of output	120

assigned by the underlying KB basecaller are already assigned with high confidence. Therefore, this study proposes to address only the more difficult subproblem seen when those confident base labels could not be assigned by KB. This takes care of the issue of class imbalance on the input-side of the problem, in that there is always only the N-label class to process, instead of the entire KB basecalled sequence (we go further into detail about class imbalance on the input-side in §4.1, Table 2; and class imbalance on the output-side – i.e. the answer set in A, C, G, T in §4.1, Table 3).

The objective of our training regimen is to optimize the number of basecalls possible by making these replacements subject to the constraint that the rate of error must not be increased above 1%. As the objective in DNA barcoding is to curate a reliable database for species identification, a standard is stipulated that requires sequences submitted must be at least 500 characters in length, with fewer than 1% N-labels (i.e. < 5 N-labels) [4]. The present study takes the above 1% standard for error as inspiration for its own target threshold for error, as we would expect that each base should contribute a probability of error less than or equal to the mean proportion of error over the length of a submitted sequence. While this stipulation is for the human edited and finished sequences submitted to BOLD, increasing the total number of reliable bases from each tracefile used to assemble a finished sequence improves the probability that this standard can be met. Since the quality value of 20 accompanying a PHRED or KB basecall indicates an error rate of 1%, the QV threshold of 20 is used as a parameter to determine when basecalls from the instrument basecaller should be considered an N-label. This ensures that any N-label replacements made in the present work can only reduce rate of error below 1%. The present work can be seen as a part of basecalling, as an additional automated pass on the chromatogram. This work can also be seen as part of sequence finishing, wherein some of the sequence editing performed by the human has been completed by the system.

4.1 Data Selection

The system developed must be tested on a wide variety of different genetic markers. It must also be validated on an extensive set composed of many thousands of tracefiles. Fulfilling these needs improves our confidence about performance on data to be seen by a production system. The data is selected from five subsets of sequences in BOLD: *Lepidoptera*, *Insecta*, *Animalia*, *Plantae*, and nonprotein. Data drawn into the taxonomic rank subset *Insecta* intentionally exclude *Lepidoptera*; and data drawn into *Animalia* exclude *Insecta*. Each subset is randomly sampled without replacement. The datasets *Lepidoptera*, *Insecta*, and *Animalia* were randomly sampled for 55000 sequences each. All available *Plantae* (66172) and non-protein-coding (17445) sequences were included at the time of sampling (August 2014).

Pairs of tracefiles and the corresponding human finished sequence for each sample are exhaustively aligned, such that every tracefile can participate as the sequence containing N-labels to be corrected, as well as the sequence providing corrections. After alignment, the resulting set of sequences with admissible N-labels for training and testing a system is obtained (Table 2).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

As mentioned in the introduction, the variation in the N-labels of the dataset must be reasonably represented in order that a trained system is capable of generalizing well onto unseen data. Here, the data are plotted along histograms describing three properties of chromatograms that are related to the noise in the data. The plots show that a spread in noise is covered by the sampled data. Figure 4 shows the distribution of the data across: (1) the base position within the chromatogram timeseries that the N-label peak occurs, (2) the ratio of the primary peak to the secondary peak amplitude for the N-label, and (3) the QV of the N-labels emitted by KB.

The second peak ratio is explained as follows. A single chromatogram may exhibit biological or physical uncertainty, when the tallest peak (the primary peak) at a time index of the chromatogram is accompanied by a shorter peak (the secondary peak) at that same time index. The second peak ratio is the height of the secondary peak in proportion to the height of the primary peak. This could be explained by either the presence of a mixture of sequences where one base differs at that location (e.g. heteroplasmy, heterozygosity), or it can be due to the mechanical properties of the electrophoresis or fluorescent dye and its detection (e.g. old reagents, dye blobs). The quality value is emitted by KB as an estimation of inverse error. The spreads in peak locations, second peak ratios, and quality value ensure that the method is trained against a wide variety of N-labels. This spread is reflective of the chance encounters that our system may have in real world operation, and is intended to reduce the bias of the system so that it does not overfit any specific type of N-label.

The test set should also be inspected for its distribution of nucleotide A, C, G, and T basecalls for corrected N-labels. This allows the determination of whether the resulting proportion of bases recovered by our system improves the tracefiles, beyond what is possible using a flat or random assignment. Table 3 is such a breakdown of the testset, and the meaning of its values in context with performance values are examined in the section Application Results.

In this study, the primary focus is on the *COI* data contained in *Lepidoptera*, *Insecta*, and *Animalia*, as this genetic marker is the longest established [5] and constitutes the greatest number of DNA barcodes indexed by BOLD Systems (http://boldsystems.org) [4]. As we will demonstrate in this work, our system is capable of improving the number of tracefiles that can be used for *COI* DNA barcodes. In an ideal situation, a system could be trained on each of the different genetic markers in BOLD, to take advantage of the biases in each of the datasets. Doing so however, could be potentially detrimental to a production system, as (1) it is uncertain whether intentionally biasing the system is actually good for classification, and (2) the incoming sequence may come from a new genetic marker, for which the system has no parameterization. Here, a more robust approach is taken. After the system is trained on *COI* training data, it is tested not only on *COI* data, but also on additional subsets of different genetic markers. The additional subsets *Plantae*, and nonprotein are used in this study to demonstrate the versatility and generality of the current method, and to test the feasibility of this method on the barcodes of plants, fungi, and other potentially unseen genetic markers and sequences of the future.

5 Application Results

There are three major components of the results. First, frequency of testing error is shown for each of the three network output voting schemes (Centre, Majority, and Unanimous). Second, averaged *conditional variance* output values obtained in Unanimous voting are furthermore processed, and numerically fitted to a frequency of error curve on the validation set. This curve then predicts probability of error in the testing set. This error prediction curve is plotted to determine its usefulness as a filter to accept N-label replacements below 1% error. Third, the final proportion of recovered N-labels below 1% error is reported.

5.1 Voting Scheme Performance

Table 4 (SVM) and Table 5 (ANN) summarize overall test set accuracy of the method in replacing N-labels with bases using *Centre*, *Majority*, and *Unanimous* voting rules. In these tables, an N-label must be seen in all five output positions to be included. Some N-labels are too close to the ends of the chromatogram to be seen in all five positions, so they are not included (and are left as N-labels). The rules determine when to *abandon* upgrading an N-label to a basecall. In *Majority*, an N-label is abandoned if there is a tie for the top voted bases. In *Unanimous*, an N-label is abandoned if the upgraded label cannot be agreed upon in all five output positions.

In general, the number of N-labels recovered is higher for ANN than SVM when comparing between like voting schemes and test sets. In *Center* and *Majority* rules, ANN performs consistently better than SVM in error rate, but in *Unanimous*, performance is nearly identical, with SVM and ANN switching back and forth for best error rate. In the rules that abandon N-labels (*Majority*, *Unanimous*), ANN consistently rejects fewer N-labels.

Table 6 shows the overlapping set of N-label replacements between SVM and ANN under *Unanimous* rule.

In the vast majority of the cases (90.7%), when a correct N-label replacement is produced, it is a label that is shared between SVM and ANN ($\cap$). This is expected, as correct classifications are likely to be the most regular and easiest to classify. In the more difficult cases, the intersection is not as big. Only 49.0% of erroneous replacements are shared, and only 40.6% of abandoned N-labels are shared. This indicates each ensemble has found a different subset of the data difficult to classify. While a grand ensemble consisting of both SVM and ANN parts is possible, this avenue is saved for future works.

In both SVM and ANN, *Unanimous* rule has the best error rate, and comes closest to a consistent rate of error below 1% for each test set. As this is a direct comparison of the two systems, it is fair to say that in *Unanimous* rule, SVM and ANN have about the same numerical performance. In *Unanimous*, the rates in error for total *COI* are different by 0.01% (favours SVM), and for overall total are different by 0.04% (favours ANN). Since performance is close, we elect to perform error estimation with our preferred system, ANN.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

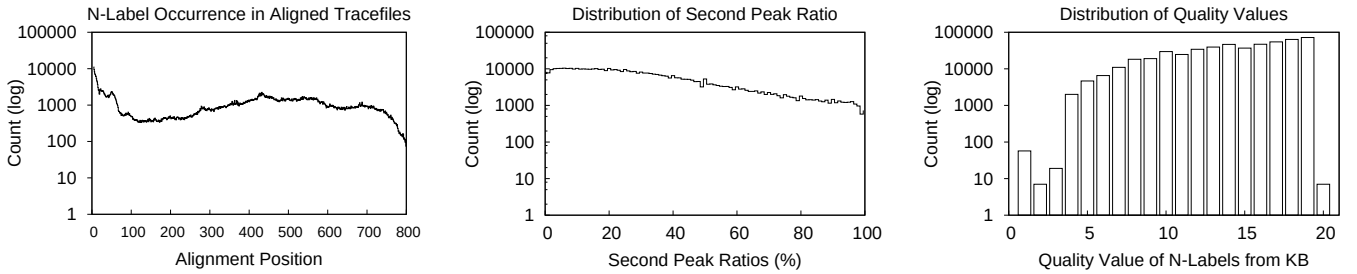

Fig. 4. Histograms showing the spread of the data along: sequence position in the basecalled sequence for N-labels (*left*), the ratio of the secondary to primary peak amplitudes for N-calls (*centre*), and quality value of N-calls emitted by KB basecaller (*right*).

TABLE 2
Count of filtered data that contain N-labels meeting criteria for inclusion (*left*), and their taxonomic relationship along with the genetic marker sequences used in this study (*right*). The sequences in *nonprotein* are various non-protein coding genetic markers sampled across all of *Eukarya*.

Subset	Alignments	N-labels
Lepidoptera	32715	160772
Insecta	34159	151025
Animalia	32707	177666
Plantae-matK	24928	203163
Plantae-rbcL	26860	104320
nonprotein	12043	52772
total	163412	849718

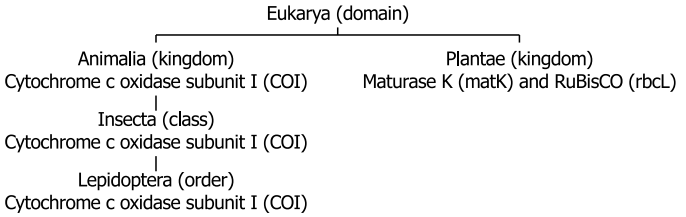

TABLE 3
Nucleotide composition (%) of the N-label answers in the test sets. The highest proportion base is bolded for each set of data.

Subset	A	C	G	T
Lepidoptera	36.91	11.44	13.86	**37.79**
Insecta	33.59	10.15	17.94	**38.31**
Animalia	27.70	15.69	26.34	**30.27**
Plantae-matK	28.83	9.18	23.95	**38.04**
Plantae-rbcL	27.49	14.86	21.36	**36.29**
nonprotein	24.77	25.10	**31.47**	18.66

5.2 Using Estimated Error Values

To further reduce error beyond what can be achieved by unanimous voting, more of the N-labels that are unreliable to recover must be abandoned. Using the error estimation function developed in §2.2, a plot of expected error and rank of emitted variance is produced. Figure 5 shows fitting error to variance output. The error here is replacing an N-label with incorrect A, C, G, T (*false positive*). The other error possible is incorrectly allowing an N-label to persist, despite the correct answer being part of the answer set (*false negative*). This quantity can be expressed as the proportion of abandoned N-labels observed, as it responds to the threshold for error in the present error estimation function, rather than directly to variance (addressed later in §5.3; plotted as Figure 7). Finally, there are the *true positive* and *true negative* cases, which would correspond to the correct assignment of an N-label to A, C, G, T, and an N-label to itself (a *correct abandon*), respectively. The former is the complement to the plotted error rate (so its data is implicitly included), and the latter is intentionally excluded from the study design. We want the system to learn to model its own uncertainty – in this regard, all *abandon* actions performed by the system are *incorrect*. Suppose the data from the four cases above are plotted together, that would yield a plot for *classification accuracy* given variance. As only the *false positive* case of error is meaningful, this motivates the present error function plot. The data used for fitting the error prediction curve is the validation set, as it is disjoint from the training set.

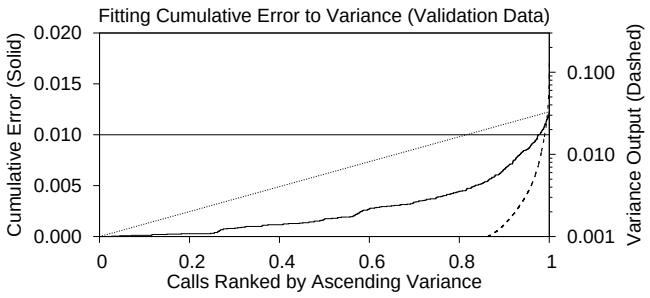

Fig. 5. Nonparametrically fitting the *variance output* to cumulative error. Cumulative error is shown given basecalls ranked in ascending variance on left-y-axis (*solid*); the ranking is shown as a quantile in x. The actual variance values are on right-y-axis (*dashed*). The ranked left-y-axis shows that the vast majority of the results data ($> 90\%$) is below 1% error. The long diagonal guideline shows the hypothetical plot for error, had it been randomly distributed with respect to variance (*dotted*).

In Figure 5, basecalls are plotted by their variance values – ranked from lowest to highest (*x-axis*). As the plot is read from left to right, errors resulting from incorrect N-label replacements can be counted, and accumulated. This results in the solid curve (*left-y-axis*). Formally, the curve in the plot is given by

$$f(r) = \frac{1}{r} \sum_{i=1}^{r} |B_i \neq B_i^{(c)}| \tag{4}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

TABLE 4
Rate of error of SVM ensemble for N-label corrections made to the test sets, and proportion of N-labels abandoned. Accuracy is computed over only the N-labels that a given voting scheme includes. Bold values indicate better performance than ANN (Table 5).

Data	*Error Rate (%)*			*N-labels Abandoned (%)*		
	Centre	Majority	Unanimous	Centre	Majority	Unanimous
Lepidoptera	4.84	3.39	**1.22**	0.00	0.38	11.02
Insecta	4.68	3.37	1.40	0.00	0.41	9.80
Animalia	4.62	2.91	**0.97**	0.00	0.40	10.83
Plantae-matK	4.40	2.22	**0.58**	0.00	0.27	12.62
Plantae-rbcL	4.79	2.57	0.90	0.00	0.63	13.91
nonprotein	8.44	5.67	1.30	0.00	1.13	21.70
total *COI*	4.71	3.20	**1.18**	0.00	0.40	10.57
total *Plantae*	4.52	2.33	0.68	0.00	0.38	13.02
total	4.87	2.99	0.98	0.00	0.44	12.32

TABLE 5
ANN ensemble performance in rate of error for N-label corrections made to the test sets, and proportion abandoned. Accuracy is computed over only the N-labels that a given voting scheme includes. Bold values indicate better performance than SVM (Table 4).

Data	*Error Rate (%)*			*N-labels Abandoned (%)*		
	Centre	Majority	Unanimous	Centre	Majority	Unanimous
Lepidoptera	**4.11**	**3.14**	1.23	0.00	**0.37**	**9.17**
Insecta	**4.04**	**3.14**	**1.39**	0.00	**0.36**	**8.20**
Animalia	**3.54**	**2.56**	1.00	0.00	**0.32**	**8.29**
Plantae-matK	**2.56**	**1.66**	0.60	0.00	**0.14**	**7.65**
Plantae-rbcL	**3.09**	**2.28**	**0.73**	0.00	**0.51**	**12.01**
nonprotein	**6.07**	**4.56**	**0.95**	0.00	**1.27**	**18.12**
total *COI*	**3.87**	**2.93**	1.19	0.00	**0.35**	**8.55**
total *Plantae*	**2.73**	**1.85**	**0.64**	0.00	**0.25**	**9.01**
total	**3.53**	**2.57**	**0.94**	0.00	**0.37**	**9.35**

TABLE 6
N-label replacements are expressed as the proportions found in set differences (SVM, ANN) and in set intersections (∩), obtained by *Unanimous* ensembles for each test set.

Data	*Correct (%)*			*Errors (%)*			*Abandoned (%)*		
	SVM	∩	ANN	SVM	∩	ANN	SVM	∩	ANN
Lepidoptera	2.9	92.2	4.9	23.8	50.3	25.9	34.7	43.7	21.6
Insecta	2.5	93.3	4.2	23.0	53.6	23.5	34.4	44.0	21.6
Animalia	2.8	91.8	5.4	23.5	49.2	27.4	39.4	39.7	20.9
Plantae-matK	2.4	89.9	7.7	21.7	50.0	28.3	49.1	34.8	16.1
Plantae-rbcL	4.4	89.1	6.6	33.2	47.1	19.8	34.0	42.4	23.6
nonprotein	5.9	83.8	10.3	42.9	32.0	25.0	33.6	46.0	20.5
total *COI*	2.7	92.4	4.9	23.4	51.1	25.5	36.4	42.3	21.3
total *Plantae*	3.0	89.7	7.3	26.1	48.9	25.0	43.8	37.5	18.7
total	3.0	90.7	6.3	25.7	49.0	25.3	39.3	40.6	20.1

which is §2.2 Equation 1 rewritten as a summation. The cumulative error function $f(r)$ is given by the count of incorrect basecalls $|B_i \neq B_i^{(c)}|$ as a proportion of all basecalls up to a certain ordinal rank r for a given value of variance. This plot is not the same as a *response-operator-curve* because no parameters are being changed; rather this curve reports error on variance after training has already completed.

The error accumulation demonstrates that most errors occurs when variance is high – in the right of the plot. This results in the vast majority of basecalls occurring below the 1% error mark (*the horizontal line*). If the error did not correspond with the variance, then we would instead see a straight line connecting the origin to the maximum error, indicating a uniform distribution of error (*dotted diagonal line, left-y-axis*). As the plot is concave, our expectation that error and variance vary together is supported. While this plot does not demonstrate that error and variance are asymptotic, it does show that the two vary together within the domain of ranked variance observed in the validation set. This trend may not continue for higher quantities of error and variance, but since this is the total amount of error accumulated by the neural network, it cannot be seen in present study whether error and variance eventually diverge. For reference, the raw estimated and averaged variance values emitted by the ANN are shown as the dashed curve (*right-y-axis*), which, like error, increases to the right in a concave. It was found that using the variance without transforming the x-axis to ranked variance resulted in poorer discrimination, as the few high-variance (and high-error) basecalls dominated the active range of the function, leading to poorer sensitivity in the highly populous low-error portion of the plot. These raw variance values should not be used directly in a production system without being transformed, as they actually only describe the uncertainty in the training set, and have been empirically found to be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

poor for generalization.

Having found the *observed error* as a function in rank of variance in the *validation set*, finding the corresponding *predicted error* as a function in rank of variance in the *test set* is needed to ensure that the cumulative error function is generalizable onto unseen data. The correlation between the two functions is given as Figure 6, showing *observed* validation set error as a function of *predicted* test set error.

While the error curves are highly correlated, never exceeding 0.2% difference in the COI test sets, they are imperfect matches. To ensure observed error is below 1%, the threshold for predicted error can be set at $.8\%$. This is discussed in the final tally of recovered N-labels, next.

5.3 Final Tally of Recovered N-labels

Table 7 tallies the total proportion of N-labels recovered using the Neural Network method. First, the statistics from *Unanimous Vote* method are shown. Then the application of two error thresholds are used to reduce error rate, at the cost that fewer N-labels are recovered.

When resulting performance in accuracy is compared against the nucleotide composition of correct answer N-labels (Table 3), it is clear that the proportion of corrections generated by the system are far above a flat assignment of labels. For instance, *Lepidoptera* contains the highest number of N-labels that should be corrected to a single basecall, the base T (Table 3). A flat assignment of the basecall T to every N-label of the test set would at most yield a correct call for 38% of N-labels. However, in Table 7, the final tally of recovered basecalls is 76%, making the system better than flat assignment by a difference of 38%. The least difference in improvement is in the *nonprotein* test set, as a flat assignment of G yields a recovery of 31%, as compared to the system final tally of 59% – a difference of 28%. Since *nonprotein* did not contribute to training data, and is also the least similar of the test data, this result is surprisingly good. It translates to over half of incoming N-labels replaced without human intervention.

Finally, it is useful to diagnose if the proportion of N-labels called responds as a concave function to the predicted and observed probabilities of error. Such a response would support the use of neural network conditional variance as an indicator of error because the number of abandoned N-labels are bunched to the left of the curve. This bunching indicates that the conditional variance has high discriminatory power for the most error prone data, and quickly rejects that data first, while becoming more reluctant as variance reduces. Figure 7 shows amount of N-labels called as a response to amount of observed error. By examining the curves, we see that the N-label recovery response is indeed concave to observed error. We can also verify visually that our threshold of 0.8% predicted error is also in a fairly good position (*vertical guideline*) as the rate of recovery begins to saturate and flatten out beyond that point.

In summary, the neural network method is capable of recovering an average of 83.96% of N-labels with an error rate of 0.94% when the *unanimous* ensemble rule is used. Interestingly, the non-COI subsets have an error rate of below 1% in *unanimous*. The overall error rate drops to 0.62% on average, when a threshold for predicted error is set to 0.8%, with a reduced rate of N-label recovery of 78.40%. When using this predicted error threshold, *all subsets of data* have an observed error rate less than 1%. The threshold of 0.8% is conservative, and is intended to ensure unseen data presented during production continues to have an error rate below 1%. As compared with a flat assignment of bases, in the worst case (*nonprotein*), this method is still able to recover a difference of 28% more N-labels correctly. Figure 7 confirms that the selection of 0.8% as the threshold is appropriate, as the vast majority of recoverable N-labels have been obtained by that predicted level of error. This result satisfies the need to recover the vast majority of N-labels, with low impact to the rate of error in the overall basecalled sequence.

6 Discussion: Significance to BOLD

To estimate the impact of this automation, a random sample of tracefiles was taken from BOLD to find the number of tracefiles that would be affected by N-label recovery (given the stipulation of length-500 sequences with $< 1\%$ ambiguity). A random sample of 448540 tracefiles containing a DNA sequence of at least 500 basepairs after trimming was included to make an estimation. For tracefiles longer than 500 basepairs, the window within that contains the fewest N-labels was taken for analysis. In the random sample, 76% (338995) of the tracefiles are already barcode compliant. From the results of the present study, an average of 80% of N-labels will be replaced with base labels by the ANN for COI, matK, and rbcL genetic markers. Assuming this proportion holds, then 4.6% (20696) of tracefiles can be made admissible by the system. This translates to an actual savings in that those incoming chromatograms that are *just* below standard can be automatically recovered without human intervention. Additionally, tracefiles that already meet the 1% limit of ambiguity for BOLD can be further improved. Of the sampled data, 20% (88265) of tracefiles fall into this category. Replacing N-labels in such sequences can be beneficial if an N-label happens to occur on a diagnostic character.

The ambitious nature of the Barcode of Life project requires as much reduction in human labour as possible in order to accelerate toward its goal to have a set of DNA Barcodes for every species of life on the planet. A benefit of an automated system like the one proposed is that the results are completely reproducible. Reproducibility is needed for consistent identification of species. Identical N-labels will be recovered every time the system is run, in contrast to human editing, which varies depending on individual, fatigue, etc.. For applications like Barcode of Life that continue to use Sanger Sequencing due to its favourable properties, the present automation can increase the speed that sequences are recovered, by offloading more work onto this algorithm.

7 Conclusion

In this study, a difficult subset of the basecalling problem is targeted, by directly leveraging the expertise of the KB Basecaller. The proposed solution benefits from ignoring the vast majority of the basecalling process, and focuses on a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2016.2598752, IEEE/ACM Transactions on Computational Biology and Bioinformatics

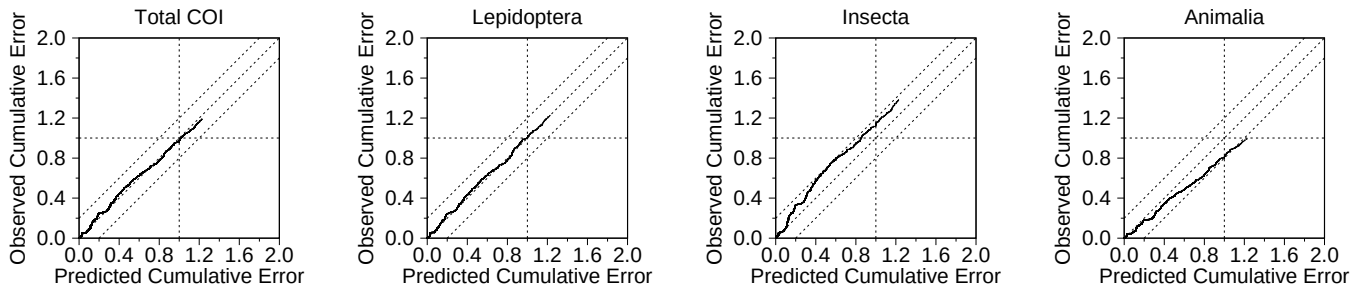

Fig. 6. Observed cumulative error from the COI test sets as a function of predicted cumulative error. Errors are mapped together using corresponding rank in variance in the two datasets. The far left plot shows the correlation for all COI test sets, and the other three plots show each of the COI test sets, *Lepidoptera*, *Insecta*, and *Animalia*. The far left curve shows high agreement between observed and predicted error – the average case. Breaking down the data, variance output under predicts error in *Lepidoptera* and *Insecta*, and over predicts error in *Animalia*. In all cases, the absolute difference is less than 0.2% in error at the 1% predicted error mark.

TABLE 7
Tally of the proportion of N-labels recovered correctly in the test set. Each step to reduce error comes with a reduced total count of N-labels recovered. Columns: *N-labels* is the total proportion of N-labels remaining from the entire test set after tracefiles have only been aligned and trimmed; *Error* is the observed total error at the current step of processing as a fraction of the unanimously voted data. The two predicted error thresholds shown are 1%, corresponding to the desired error rate if observed error had perfect concordance; and $.8\%$, corresponding to an arbitrary buffer to ensure the final error rate is below 1% observed error.

Dataset	Unanimously Voted		at $< 1\%$ Predicted Error		at $< .8\%$ Predicted Error	
	N-labels	Error	N-labels	Error	N-labels	Error
Lepidoptera	81.78	1.23	79.82	1.01	76.22	0.78
Insecta	84.97	1.39	82.68	1.13	79.24	0.94
Animalia	89.06	1.00	87.07	0.81	83.59	0.63
Plantae-matK	89.98	0.60	88.60	0.52	85.58	0.41
Plantae-rbcL	75.68	0.73	73.66	0.63	69.32	0.53
nonprotein	67.10	0.95	63.54	0.60	58.82	0.43
total *COI*	85.38	1.19	83.31	0.97	79.80	0.78
total *Plantae*	85.13	0.64	83.53	0.55	80.07	0.45
total	83.96	0.94	81.98	0.77	78.40	0.62

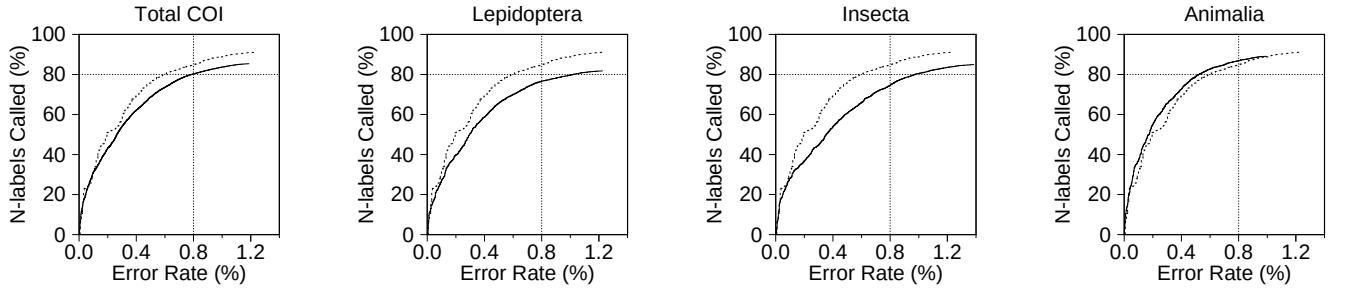

Fig. 7. The amount of N-labels recovered by this method, as observed error increases. The dashed line is the predicted error provided by the validation set, while the solid line is the amount of error observed in the COI test sets. The vertical dashed line corresponds to the 0.8% predicted error in the validation set. The horizontal dashed line corresponds to 80% of N-labels recovered.

specific, nuanced task. Our system is able to recover the majority of N-labels (78.40%) occurring internally within a basecalled sequence, while maintaining error below the target 1% (0.62%). The present method uses all the data from a chromatogram, and data emitted by KB basecaller to make its decisions. As the proposed system is constructed on the artificial neural network, there is the drawback that the system is a black-box classifier, in that it is impossible to know *how* the system performs classifications. There are two answers to this limitation. First, performance of the system in the target problem meets the target rate of error on a diverse set of test data; while the inner workings of the device are not known, we have established confidence in its performance on future data. Second, black-boxes are somewhat tolerated in the target community of the proposed system; the KB Basecaller is itself proprietary software, whose implementation details are not published.

The present system can be considered part of sequence finishing, as it performs one type of editing that a human would commit – the replacement of N-labels with bases. While this system is capable of fixing N-labels with a single tracefile, a human editor would generally use at least two aligned sequences, or even up to 96-aligned chromatograms resulting from a complete plate of capillary electrophoresis sequencing runs. Future works will need to focus on the other editing actions that a human editor commits. These include *insertion* of base labels at missed peaks in a chromatogram or *deletion* of excess base labels at spurious

peaks. For workflows that automatically accept or reject tracefiles based on the number of unambiguous bases, this present system reduces work by increasing the proportion of tracefiles that can move onto assembly and analysis, thus reducing the number of samples that need be resequenced.

To address the insertions and deletions, a different strategy would need be proposed. In the present system, we benefit from the presence of N-labels in the KB basecalled sequence to call the attention of our system. In the more subtle problems of insertions and deletions, these flags are unavailable and locating editing actions appropriately would require a confident pre-processing step. Furthermore, adapting the nature of the problem requires more thought, as it is a problem of variable length representation i.e. the count of tokens in the input and output will necessarily not match. This is incompatible with the fixed-length representation used by the present SVM/ANN. This would require that either the problem is rephrased in a form addressable by SVM/ANN etc., or that a system that can accommodate variable length representations be used, such as HMM. It is however, useful to know the impact of addressing insertions and deletions by quantifying their occurrence.

To estimate their abundance relative to that of N-labels, a random sample of alignments between tracefiles and DNA barcode sequences is obtained. Only alignments corresponding to tracefiles with at least 200 basepairs after trimming are included, and only if 90% of the aligned bases are matched between the tracefile and the DNA barcode sequence. These two requirements are set in place to reduce the impact of low quality tracefiles from interfering with estimation. This results in a set of 440884 alignments. Of these, the mean sequence length is 587 base pairs, with a mean of 14.05 N-label replacements per alignment. In proportion, for every 1000 N-label replacements in the data (6198299 total), there are 12.6 insertions (78230 total), and 7.1 deletions (44234 total). While these edits are sparse, resolving them is important depending on future application and data quality needs.

It is likely that the present method of automated N-label replacement can be extended into other applications beyond DNA Barcoding, with proper consideration of what error rates and types of errors are stipulated in the standards of those applications. For instance, this study paid less attention to the occurrence of heterozygous peaks, when two or more chromatogram channels peak at the same spot. For applications requiring specificity in this type of uncertainty, a filter for peak height can be fitted, or an expression of abundance of the two competing bases can be estimated. Finally, it is possible that DNA Barcoding, along with other applications can benefit from a solution similar to the present system, capable of correcting ambiguous homopolymers, where the count of a long stretch of the same base is uncertain. This would require establishing the correct count of that base to use as the answer set of the data (similar to establishing what is considered the correction for an N-label in this study). In principle, a machine learned classifier for future tasks can benefit from the same error estimation using variance, as the present system.

Acknowledgments

This research program was supported by the Ontario Ministry of Research and Innovation, by the Canada Foundation for Innovation, by NSERC, and by the government of Canada through Genome Canada and the Ontario Genomics Institute in support of the International Barcode of Life project.

References

[1] F. Sanger, S. Nicklen, and A. Coulson, "DNA sequencing with chain-terminating inhibitors," *Proc. Natl. Acad. Sci.*, vol. 74, no. 12, pp. 5463–7, December 1977.

[2] B. Ewing, L. Hillier, M. C. Wendl, and P. Green, "Base-calling of automated sequencer traces using Phred – i. accuracy assessment," *Genome Res.*, vol. 8, pp. 175–85, 1998.

[3] B. Ewing and P. Green, "Base-calling of automated sequencer traces using Phred – ii. error probabilities," *Genome Res.*, vol. 8, pp. 186–94, 1998.

[4] S. Ratnasingham and P. D. Hebert, "BOLD: The barcode of life data system (www.barcodinglife.org)," *Molecular Ecology Notes*, 2007.

[5] P. D. N. Hebert, A. Cywinska, S. L. Ball, and J. R. deWaard, "Biological identifications through DNA barcodes," *Proc Biol Sci*, vol. 270, no. 1512, pp. 313–21, Feb 2003.

[6] R. W. Hyman, H. Jiang, M. Fukushima, and R. W. Davis, "A direct comparison of the KB Basecaller and Phred for identifying the bases from DNA sequencing using chain termination chemistry," *BMC Research Notes*, vol. 3, no. 257, p. 4 pages, 2010.

[7] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J Mol Biol*, vol. 48, no. 3, pp. 443–53, Mar 1970.

[8] E. Lyons, M. Scheible, K. S. Andreaggi, J. Irwin, and R. Just, "A high-throughput Sanger strategy for human mitochondrial genome sequencing," *BMC Genomics*, vol. 14, no. 1, pp. 881+, 2013. [Online]. Available: http://dx.doi.org/10.1186/1471-2164-14-881

[9] C. Berger, B. Berger, and W. Parson, "Sequence analysis of the canine mitochondrial DNA control region from shed hair samples in criminal investigations." *Methods Mol Biol*, vol. 830, pp. 331–48, 2012.

[10] J. Naue, T. Sanger, U. Schmidt, R. Klein, and S. Lutz-Bonengel, "Factors affecting the detection and quantification of mitochondrial point heteroplasmy using sanger sequencing and snapshot minisequencing." *Int J Legal Med*, vol. 125, no. 3, pp. 427–36, 2011.

[11] P. H. Fernandes, J. Saam, J. Peterson, E. Hughes, R. Kaldate, S. Cummings, A. Theisen, S. Chen, J. Trost, and B. B. Roa, "Comprehensive sequencing of PALB2 in patients with breast cancer suggests PALB2 mutations explain a subset of hereditary breast cancer." *Cancer*, 2014.

[12] F. Mertens and J. Tayebwa, "Evolving techniques for gene fusion detection in soft tissue tumours." *Histopathology*, vol. 64, no. 1, pp. 151–62, 2014.

[13] K. Qu, Q. Pan, X. Zhang, L. Rodriguez, K. Zhang, H. Li, A. Ho, H. Sanders, A. Sferruzza, S.-M. Cheng, D. Nguyen, D. Jones, and F. Waldman, "Detection of BRAF V600 mutations in metastatic melanoma: Comparison of the cobas 4800 and sanger sequencing assays," *The Journal of Molecular Diagnostics*, vol. 15, no. 6, pp. 790 – 795, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1525157813001335

[14] D. French, A. Smith, M. P. Powers, and A. H. Wu, "KRAS mutation detection in colorectal cancer by a commercially available gene chip array compares well with sanger sequencing," *Clinica Chimica Acta*, vol. 412, no. 17-18, pp. 1578 – 1581, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0009898111002579

[15] S. H. Zhan, R. T. Abboud, B. Jung, B. Kuchinka, D. Ralston, B. Casey, and A. Mattman, "Sanger sequencing solved a cryptic case of severe alpha1-antitrypsin deficiency," *Clinical Biochemistry*, vol. 45, no. 6, pp. 499 – 501, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0009912012000562

[16] J. Traeger-Synodinos and C. L. Harteveld, "Advances in technologies for screening and diagnosis of hemoglobinopathies." *Biomark Med*, vol. 8, no. 1, pp. 119–31, 2014.

[17] D. Thornley and S. Petridis, "Machine learning in basecalling – decoding trace peak behaviour," in *Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB '06. 2006 IEEE Symposium on*, 2006, pp. 1–8.

[18] D. Thornley and S. Petridis, "Decoding trace peak behaviour - a neuro-fuzzy approach," in *IEEE International Conference on Fuzzy Systems*, July 2007. [Online]. Available: http://pubs.doc.ic.ac.uk/neuro-fuzzy-dna-basecaller/

[19] D. Thornley, M. Zverev, and S. Petridis, "Machine learned regression for abductive DNA sequencing," in *The 2007 International Conference on Machine Learning and Applications*, December 2007, nominated for best paper. [Online]. Available: http://pubs.doc.ic.ac.uk/machine-learned-abduction-DNA/

[20] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1, Aug 1995, pp. 278 – 282.

[21] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[22] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: http://dx.doi.org/10.1023/A:1022627411411

[24] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, Jun. 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1009715923555

[25] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115 – 133, 1943.

[26] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioural sciences," *Ph.D. dissertation, Committee in Appl. Math. Harvard Univ., Cambridge, MA,*, Nov. 1974.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagation errors," *Letters to Nature*, vol. 323, pp. 533 – 536, October 1986.

[28] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359 – 366, 1989.

[29] 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*, vol. 12, pp. 2825–2830, 2011.

[30] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[31] R. Caruana, "A dozen tricks with multitask learning," in *Neural Networks: Tricks of the Trade (2nd ed.)*, ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K. Müller, Eds., vol. 7700. Springer, 2012.

[32] R. Neuneier and H. Zimmermann, "How to train neural networks," in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. B. Orr and K. Müller, Eds., vol. 1524. Springer, 1998, pp. 373–423.

[33] P. Baldi and P. Sadowski, "The dropout learning algorithm," *Artificial Intelligence*, vol. 210, pp. 78 – 122, 2014.

[34] J. B. G. III, D. Torgersen, and C. Tibbetts, "Pattern recognition for automated DNA sequencing: I. on-line signal conditioning and feature extraction for basecalling." in *ISMB*, L. Hunter, D. B. Searls, and J. W. Shavlik, Eds. AAAI, 1993, pp. 136–144.

[35] C. Tibbetts, J. Bowling, and J. G. III, *Automated DNA Sequencing and Analysis*. San Diego, CA: Academic Press, 1994, ch. Neural networks for automated base-calling of gel-based DNA sequencing ladders, pp. 219 – 230.

[36] P. Boufounos, S. El-Difrawy, and D. Ehrlich, "Basecalling using hidden markov models," *Journal of the Franklin Institute*, vol. 341, no. 1–2, pp. 23 – 36, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0016003203000875

[37] O. Khan, K. Assaleh, G. Husseini, A. Majdalawieh, and S. Woodward, "DNA base-calling using artificial neural networks," in *Biomedical Engineering (MECBME), 2011 1st Middle East Conference on*, 2011, pp. 96–99.

[38] O. Mohammed, K. Assaleh, G. Husseini, A. Majdalawieh, and S. Woodward, "Novel algorithms for accurate DNA base-calling," *Journal of Biomedical Science and Engineering*, vol. 6, pp. 165 – 174, 2013.

[39] P. Boufounos, S. El-Difrawy, and D. Ehrlich, "Hidden markov models for DNA sequencing," in *Proceedings of Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, 2002.

[40] M. S. Pereira, L. Andrade, S. E. Difrawy, B. L. Karger, and E. S. Manolakos, "Statistical learning formulation of the DNA basecalling problem and its solution in a bayesian em framework," *Discrete Applied Mathematics*, vol. 104, pp. 1–3, 2000.

[41] L. Andrade-Cetto and E. Manolakos, "A graphical model formulation of the DNA base-calling problem," in *Machine Learning for Signal Processing, 2005 IEEE Workshop on*, 2005, pp. 369–374.

[42] K.-c. Liang, X. Wang, and D. Anastassiou, "Bayesian basecalling for DNA sequence analysis using hidden markov models," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 4, no. 3, pp. 430–440, 2007.

[43] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS*, 1993, pp. 586–591.

[44] J. Martens, "Deep learning via hessian-free optimization," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, J. Furnkranz and T. Joachims, Eds. Haifa, Israel: Omnipress, June 2010, pp. 735–742.

[45] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[46] H. White, "Learning in artificial networks: A statistical perspective," *Neural Computation*, vol. 1, no. 4, pp. 425 – 464, 1989.

[47] L. Zhao, "Uncertainty prediction with multi-layer perceptrons," Master's thesis, The University of Guelph, June 2000.

[48] F. Karray and C. De Silva, *Soft Computing and Intelligent Systems Design: Theory, Tools, and Applications*. Pearson/Addison Wesley, 2004. [Online]. Available: http://books.google.ca/books?id=mqYw-Xig0IsC

[49] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, G. B. Orr and K. Müller, Eds. Springer, 1998.

[50] C. Mora, D. P. Tittensor, S. Adl, A. G. B. Simpson, and B. Worm, "How many species are there on earth and in the ocean?" *PLoS Biol*, vol. 9, no. 8, p. e1001127, 08 2011.

[51] W. J. Ansorge, "Next-generation DNA sequencing techniques," *N Biotechnol*, vol. 25, no. 4, pp. 195–203, Apr 2009.

[52] A. Grada and K. Weinbrecht, "Next-generation sequencing: Methodology and application," *Journal of Investigative Dermatology*, vol. 133, no. 8, p. e11, 2013.

Eddie Y. T. Ma received the B.S. degree from University of Guelph, Guelph, Ontario, Canada in 2007, and the M.S. degree from University of Guelph, Guelph, Ontario, Canada in 2009. He is an analyst and developer of the Biodiversity Institute of Ontario at the University of Guelph. His research interests are machine learning and bioinformatics.

Sujeevan Ratnasingham received the B.S. degree from University of Guelph, Guelph, Ontario, Canada in 2000. He is the Director of Informatics of the Biodiversity Institute of Ontario, at the University of Guelph. He is interested in biodiversity knowledge discovery, phylogenetics, machine Learning, high performance computing, and the integration of biological data from the molecular scale to the ecological scale.

Stefan C. Kremer received the B.S. degree from the University of Guelph, Guelph, Ontario, Canada in 1991, and the Ph.D. degree at the University of Alberta, Edmonton, Alberta, Canada in 1995. He is the Director of the School of Computer Science at the University of Guelph and inaugural Director of the Bioinformatics Program at the same university. His interests are in machine learning and structural pattern recognition applied to biological problems.