Lean and deep models for more accurate filtering of SNP and INDEL variant calls

Article *in* Bioinformatics · December 2019

DOI: 10.1093/bioinformatics/btz901

4 authors, including:

Sam Freesun Friedman
Broad Institute of MIT and Harvard
27 PUBLICATIONS **165** CITATIONS

Bioinformatics, 2019, 1–8
doi: 10.1093/bioinformatics/btz901
Advance Access Publication Date: 12 December 2019
Original Paper

Sequence analysis

Lean and deep models for more accurate filtering of SNP and INDEL variant calls

Sam Friedman*, Laura Gauthier, Yossi Farjoun and Eric Banks

Data Sciences Platform, Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on August 1, 2019; revised on November 22, 2019; editorial decision on November 25, 2019; accepted on December 10, 2019

Abstract

Summary:
We investigate convolutional neural networks (CNNs) for filtering small genomic variants in short-read DNA sequence data. Errors created during sequencing and library preparation make variant calling a difficult task. Encoding the reference genome and aligned reads covering sites of genetic variation as numeric tensors allows us to leverage CNNs for variant filtration. Convolutions over these tensors learn to detect motifs useful for classifying variants. Variant filtering models are trained to classify variants as artifacts or real variation. Visualizing the learned weights of the CNN confirmed it detects familiar DNA motifs known to correlate with real variation, like homopolymers and short tandem repeats (STR). After confirmation of the biological plausibility of the learned features we compared our model to current state-of-the-art filtration methods like Gaussian Mixture Models, Random Forests and CNNs designed for image classification, like DeepVariant. We demonstrate improvements in both sensitivity and precision. The tensor encoding was carefully tailored for processing genomic data, respecting the qualitative differences in structure between DNA and natural images. Ablation tests quantitatively measured the benefits of our tensor encoding strategy. Bayesian hyper-parameter optimization confirmed our notion that architectures designed with DNA data in mind outperform off-the-shelf image classification models. Our cross-generalization analysis identified idiosyncrasies in truth resources pointing to the need for new methods to construct genomic truth data. Our results show that models trained on heterogenous data types and diverse truth resources generalize well to new datasets, negating the need to train separate models for each data type.

Availability and implementation: This work is available in the Genome Analysis Toolkit (GATK) with the tool name CNNScoreVariants (https://github.com/broadinstitute/gatk).

Contact: sam@broadinstitute.org

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Humans are mostly alike. The 3.2 billion base pairs (bp) of our genome are the same at about 99.9% of genomic sites. Variant calling is finding those rare sites of variation. Next generation sequencing (NGS) reads DNA from a sample by shattering large contiguous pieces of the chromosomes into fragments which are aligned to a human reference sequence. Where these alignments sufficiently differ from the reference, variants are called.

Typically, variant callers are tuned to be very sensitive, more tolerant of false positives than false negatives. The variant filter then scores each variant so analysts can choose a sensitivity threshold appropriate for a given task. Here, we show how variant filtration can be accomplished with convolutional neural networks (CNNs) to improve both the sensitivity and precision of variant calls. We make the best performing models available pre-trained in the Genome Analysis Toolkit (GATK, 2019).

Various machine learning approaches have been applied to genomic sequencing data, sometimes framing the problem in very different ways. Variant calling and genotyping are cast as image classification in DeepVariant (Poplin *et al.*, 2018). The authors apply the innovative inception architecture (Ioffe and Szegedy, 2015; Szegedy *et al.*, 2014) to classify images that encode reads aligned to a reference sequence. The labels correspond to the diploid genotypes: homozygous variant, heterozygous and homozygous reference. The problem of genotyping for somatic variation in cancer samples is tackled with a multi-layer perceptron (MLP) where the input is given by a 1000+ dimensional vector of handcrafted features (Torracinta and Campagne, 2016).

In contrast to those approaches, our input is a multi-dimensional tensor which preserves both read and reference sequence information with the addition of mapping quality and read flags. Base qualities are encoded directly as the probabilities they represent rather

than as separate arbitrarily scaled values. The smoothness prior of natural images (neighboring pixels tend to be similar) justifies extensive max-pooling in the inception architecture used by DeepVariant. In contrast, our architectures avoid max-pooling until deeper layers, because DNA is not smooth but inherently discrete and discontinuous.

2 Materials and methods

Training data have been generated with variants from National Institute of Standards and Technology's (NIST) Genomes in a Bottle (GiaB) project (Zook and Salit, 2011), Illumina's Platinum Genomes (PG; Eberle *et al.*, 2013) and the synthetic diploid (SynDip) dataset (Li *et al.*, 2018). Each resource provides both a validated set of variant calls format (VCF) and a set of intervals over the genome in which the resource is confident all variants have been found. The positive training examples are variants from the validated truth set VCF. Combining the confident regions, the validated VCF, and an unfiltered unvalidated VCF we define negative training examples as variant calls within the confident region, in the unvalidated VCF, but absent from the validated VCF (see Fig. 2).

The three resources used were each constructed differently. The GiaB project created a validated set of variants through consensus of several variant calling algorithms. The validated VCF is made from the sites where most of the callers agreed. Sites without consensus were excluded from the confident region. This method delivers high-quality variants, but it is biased towards the sites that are easiest to call. The PG project used the three-generation CEPH family (pedigree 1463) and curated their calls based on expected Mendelian inheritance, as well as consensus across callers and sequencing chemistries. Both of these truth sources rely entirely on short-read data from Illumina sequencers. In contrast, the recently released SynDip dataset makes use of short-read data as well as high-depth long-read data from a Pacific Biosciences sequencer to help resolve difficult and repetitive parts of the genome and capture larger events. PG-trained models generalize well to other truth resources, but the PG datasets proved difficult for models trained with other truth resources, especially for SNPs. Most of this discordance is caused by sites with extremely high coverage indicative of segmental duplication. The A.S.J. son sample also appears more difficult for generalization, perhaps because the NA12878 truth calls are more reliable, as has been suggested elsewhere (Luo *et al.*, 2018).

Table 1 summarizes the datasets used for training our models. We calculate the transition–transversion rate (Ti/Tv), as well as the insertion–deletion rate for each class in each dataset. Notice how the Ti/Tv for positive SNPs is consistent between datasets and concordant with other analyses of whole exome and whole genome data. In contrast, the negative SNP Ti/Tv ratio is much more variable and in some datasets approaches 0.5, which would indicate no evolutionary bias between transitions and transversions. This indicates a stronger biological signal in the positive class. Similarly, positive INDELs have a much tighter distribution on insertion–deletion rates than negative ones.

The input to our filtering method is one validated truth set of VCF file with a confident region (BED file), an evaluation set of calls with annotations (VCF file) and a reference sequence (FASTA file). Read-level architectures also require aligned reads (SAM file or, more commonly, BAM file). Reads are aligned (and locally realigned) to a reference genome with Burrows-Wheeler Aligner (BWA) and GATK's Haplotype Caller (Li, 2013; McKenna *et al.*, 2010; Poplin *et al.*, 2017).

2.1 Tensor encoding

Reference tensors are 1-hot encodings of reference bases centered at an evaluation variant (see Fig. 1B). The window size around the variant is a hyper-parameter, in the experiments presented here set to 128 bp, unless otherwise noted. Read tensors span different genomic positions in width and different piled up reads in height (see Fig. 1C). Reference tensor channels are the four DNA bases, while

Table 1. This table shows the datasets used in this article

Sample	Reference build	Sequencer	Chemistry	Truth	+SNPs (in thousands)	+INDELs (percent)	−SNPs	−INDELs	+SNP (Ti/Tv)	−SNP (Ti/Tv)	+INDEL (I/D)	−INDEL (I/D)
NA12878	HG19	HiSeqX	PCR−	GiaB	129 (40%)	171 (53%)	13 (4%)	11 (3%)	2.13	0.95	1.03	0.66
NA12878	HG19	HiSeqX	PCR+	GiaB	123 (28%)	160 (37%)	37 (9%)	117 (27%)	2.15	0.54	1.34	6.72
NA12878	HG19	HiSeqX	PCR+	GiaB	142 (21%)	513 (74%)	20 (3%)	12 (2%)	2.10	0.78	1.27	4.92
NA12878	HG19	HiSeqX	PCR−	PG+GiaB	44 (37%)	61 (51%)	10 (8%)	4 (4%)	2.13	0.85	1.21	1.39
NA12878	HG19	HiSeqX	PCR−	PG	59 (27%)	48 (22%)	57 (26%)	54 (25%)	2.11	1.30	1.17	1.40
NA12878	HG38	NovaSeq	PCR−	GiaB	23 (35%)	23 (34%)	16 (24%)	4 (6%)	2.11	0.95	1.30	1.03
NA12878	HG19	HiSeq4000	Exome	GiaB	29 (95%)	1 (3%)	0 (1%)	0 (1%)	2.73	0.65	1.16	0.50
NA12878	HG19	HiSeq4000	Exome	GiaB	26 (96%)	0 (3%)	0 (1%)	0 (0%)	2.85	0.57	1.27	0.25
NA24385	HG38	HiSeqX	PCR−	GiaB	121 (34%)	206 (59%)	17 (5%)	6 (2%)	2.10	0.73	1.28	2.36
CHM	HG38	HiSeqX	PCR−	SynDip	35 (20%)	53 (31%)	44 (26%)	39 (23%)	2.08	1.09	1.33	1.44
CHM	HG19	HiSeq4000	Exome	SynDip	41 (80%)	4 (9%)	1 (2%)	4 (8%)	2.61	1.08	1.53	4.37

Note: The numbers reflect tensors used for training where positive SNPs are downsampled by about 90%. We calculate the Ti/Tv in each dataset for positive and negative SNPs, and the insertion deletion rate for positive and negative INDELs.

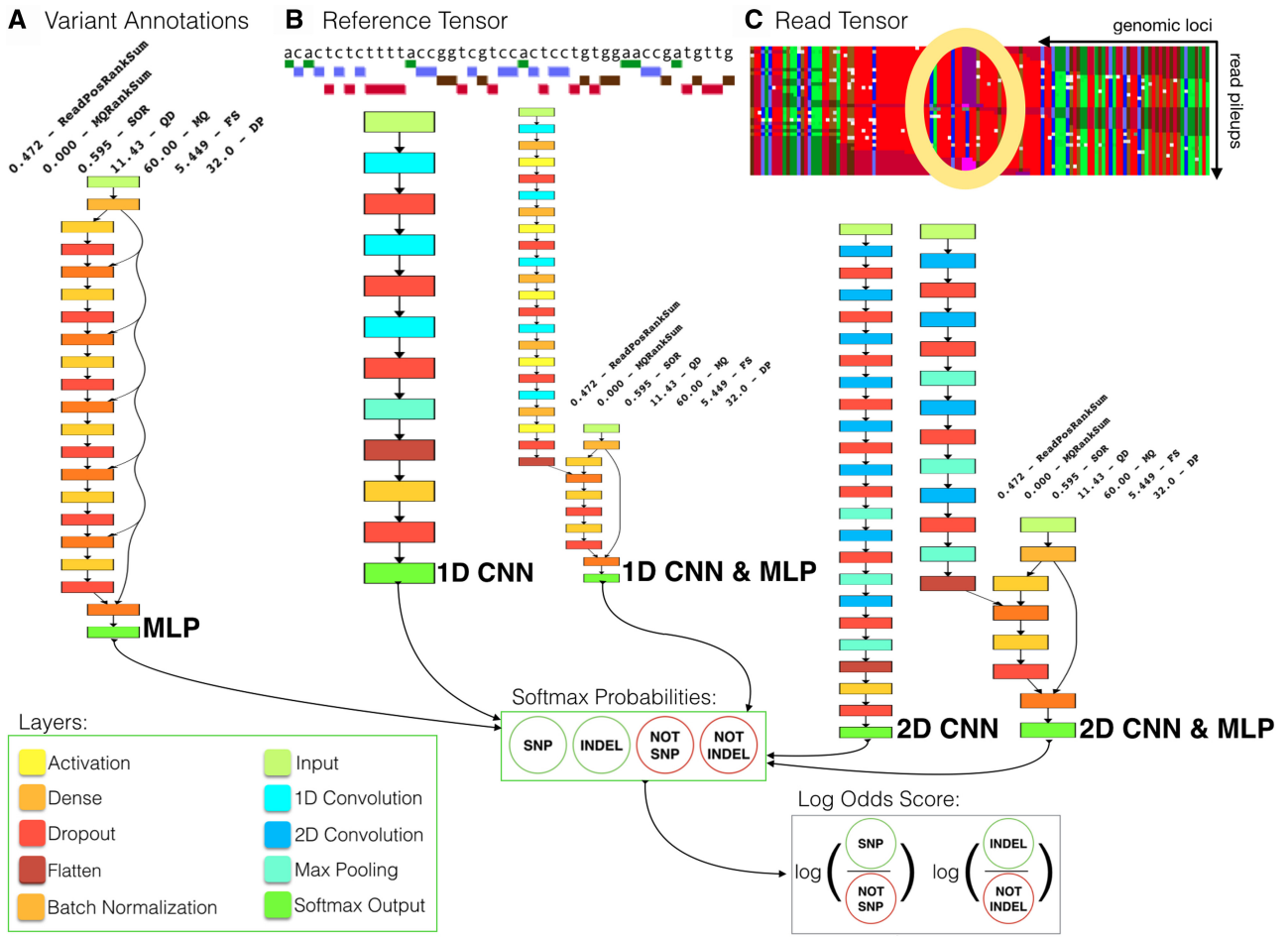

Fig. 1. This figure shows the three types of input data: variant annotations (**A**), 2D 1-hot reference sequence tensors (**B**) and 3D *p*-hot read tensors (**C**). Below the three types of input are architecture topologies which output softmax probabilities. Because we train jointly on SNPs and INDELs, but score them separately, the joint softmax probabilities are transformed into separate log odds scores for each variant type. The read tensor in (C) encodes a heterozygous 3 bp insertion colored purple on the top haplotype. The horizontal direction spans different genomic loci, the vertical direction shows the depth of read coverage, and channel data is indicated by color. The reference data comprises four 1-hot encoded channels which are identical along the vertical axis. The read data is base-quality encoded in the first four channels and is mostly identical along the vertical axis with the exception of errors and true genomic variants. The read flags indicating the strand, pairing and mapping quality are encoded in five channels and they are identical along the horizontal axis. The different model architectures shown here all learn a function mapping their input to the variant classification task illustrated by the softmax and log odds ratio layers

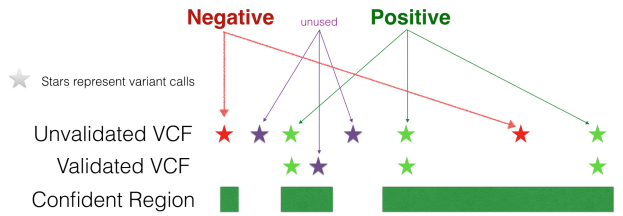

Fig. 2. Diagram showing construction of training data for variant filtering. Variant calling architectures do not require an unvalidated VCF

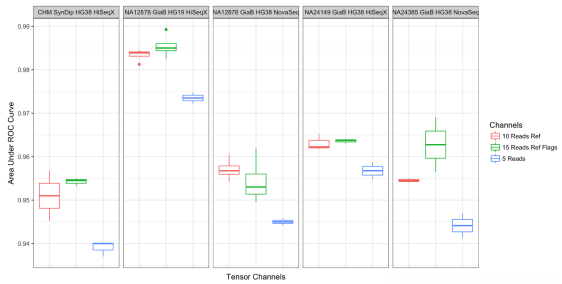

Fig. 3. Tensors with reads reference and flags are compared with tensors without flags and tensors with read sequence alone. The most robust encoding, with reads, reference and flags performs the best

read tensor channels cover the four DNA bases for the reference four more for the reads, two channels for insertions and deletions and an additional channel for metadata (flags) associated with each read.

To evaluate the different choices in tensor encoding we trained identical models on the different data types and sources. We repeated our experiments with different random initializations. Figures 3–8 compare area under the ROC curve on held out test data between different tensor encodings and data sources. Each boxplot shows the average ROC area and the variance across different initializations. Figure 3 compares tensors with 5 channels encoding only the read sequence, 10 channel tensors, which also encode the reference sequence and 15 channel tensors which encode read, reference and the read flag metadata. Though often redundant, encoding both reference and read sequence data allows the model to see the DNA bases removed by deletions, which apparently helps improve predictions. The read flag encoding is also often redundant with separate dedicated channels for both positive and negative strand and first versus second in pair. However, this redundancy allows us to seamlessly encode chimeric reads and extends naturally to encode

paired-end reads as described below. As the figure indicates, this extra information also helps improve predictions.

In the DeepVariant approach the base qualities are treated as a separate channel; however, base qualities are not separate from the bases themselves; they can be directly interpreted as the confidence of each base call. For this reason, our models encode the base qualities into the base channels of the read tensor. Base qualities are stored as phred-scaled values, which map directly to probabilities ($Q = -10\log_{10}P$). Instead of 1-hot tensors, our tensors are P-hot with the error probability shared equally between the three other bases. For example, a Q score of 10 in the A channel would be encoded as a tensor with 0.9 in the A channel and 0.03̄ in the other three channels. We experimented with integrating the qualities as either probabilities or phred-scaled probabilities as they are encoded in the BAM file. The raw phred-scaled values performed the worst. This was likely because these values were not normalized. As a result they required large learning rates and weights in ranges where floating point numbers have more quantization error, which both contribute to less reliable gradients. In general, models trained with probabilities performed slightly better on the test sets than the 1-hot encodings, as shown by Figure 4.

Local re-assembly with De Bruijn graphs helps resolve long insertions and deletions that occur at the ends of reads. Models trained on tensors of re-assembled reads perform better than models trained on the alignment from the original BAM file as shown in Figure 5.

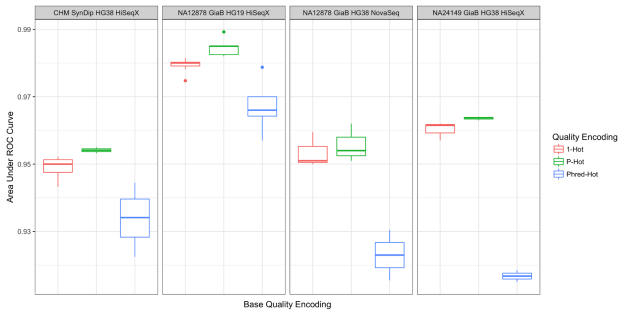

Fig. 4. The P-hot encoding tends to outperforms the other, though less so on NovaSeq where base qualities are aggressively quantized

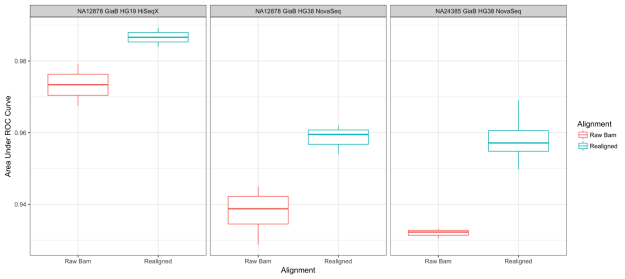

Fig. 5. Local reassembly of reads improves performance

At heterozygous sites, sorting by variant allele clearly reveals the parental haplotypes. For example, at the center of the highlight in Figure 1C, one parental haplotype contains the inserted bases CAA, the other haplotype, like the reference, spans the insertion in purple. Figure 7 compares models trained with allele-sorted read tensors to models with reads sorted by reference start position. Though often comparable, sorting by haplotype sometimes performs better. Ultimately, a more disciplined approach would be to use a permutation invariant model on the pile of reads, rather than imposing a somewhat arbitrary order (Zaheer *et al.*, 2017).

Paired-end sequencing allows for more accurate mapping and resolution of structural variation. It also improves classification performance when pairs are encoded together. Figure 8 compares read tensors where pairs have been grouped by row with a read tensor which treats pairs individually. The tensor types are compared across several different genomic window sizes. For a given genomic window size the paired-end tensors outperformed the single read tensors. In addition, paired-end read tensors perform best with window sizes of 256 bp while single read tensors tend to peak at 128 bp, suggesting that models trained on paired-end read tensors can find more informative features from larger genomic contexts.

2.2 Model architectures

The 1D CNNs perform convolutions over reference tensors which learn DNA motif detectors, i.e. position weight matrices (Rosenblatt, 1960; Stormo *et al.*, 1982). Visualization of these layers after training confirms many correspond to familiar motifs like homo-polymers and short tandem repeats (STR). Subsequent layers learn increasingly abstract grammars over these motifs. Figure 6 shows visualizations of these first layer motif detectors from 1D CNNs.

The 2D CNNs convolve over read tensors, alternating between the axes of genomic position and piled up reads (Szegedy *et al.*, 2016). For read tensors, this separation is natural, because the spatial correlations in sequence context are quite different from spatial correlations across reads in pileups. Max-pooling layers reduce the size of the internal representation and provide translation invariance, such that the variant data remains independent from the exact genomic position. However, because of the discrete nature of DNA sequence data we do not max-pool over the sequence axis, only over the pileups.

After several layers of convolution the spatial dimensions of the tensor are flattened into a single 1D vector. This vector is concatenated with batch-normalized variant annotations Ioffe and Szegedy (2015). The concatenated vector is input to densely connected layers feeding forward to a final softmax layer, which evaluates the

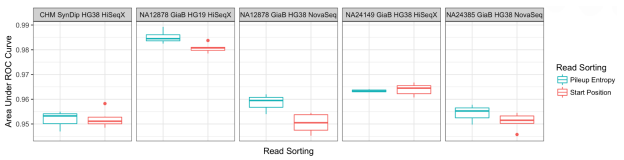

Fig. 7. Reads sorted by pileup entropy compared with reads sorted by reference position. Sorting by pileup entropy tends to improve performance

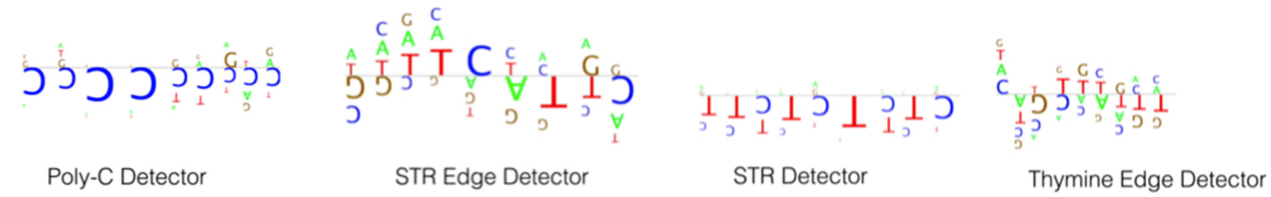

Fig. 6. The size of the weight in each base channel is shown by the size of DNA letter, and negative weights are indicated by upside-down letters. For example, the negative weights on the poly-C detector and STR detector indicate that variants in or near such motifs are less likely to be real. To ease interpretability, the motifs shown here were learned by a model with a single layer of convolution trained with spatial dropout of 50% of the convolutional kernels (Hinton *et al.*, 2012). The single-layer of convolution and spatial dropout encourage motifs that are informative regardless of the state of other parameters in the network

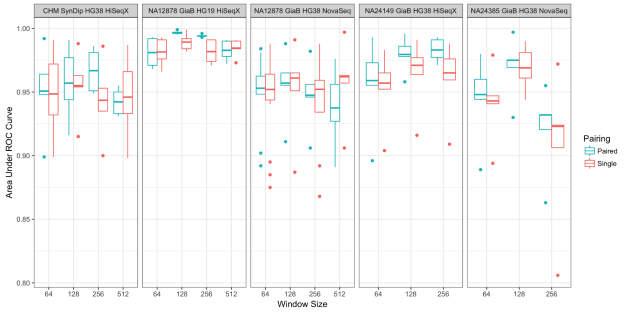

Fig. 8. Paired-end tensors are compared across various window sizes with single read tensors. For single read tensors performance peaks with 128 bp windows, whereas for paired read tensors performance peaks at 256 bp windows

annotations and ultimately classifies the variants as artifact or real (see Fig. 1B and C). Performance improves by including a skip connection, concatenating the normalized annotations with the penultimate dense layer as in Figure 1B and C or with all deeper layers as in Figure 1A. Finally, a softmax layer emits a probability that the variant is real. Since we train jointly over both SNPs and INDELs this softmax has four neurons for the four variant classifications: SNP, INDEL, NOT SNP, NOT INDEL. Because SNPs and INDELs are filtered with different thresholds, we transform these probabilities into separate log odds scores for SNPs:$\log\left(\frac{P_{\text{SNP}}}{P_{\text{NOTSNP}}}\right)$ and INDELs: $\log\left(\frac{P_{\text{INDEL}}}{P_{\text{NOTINDEL}}}\right)$.

2.3 Training

Models are trained by back-propagation of error gradients (Rumelhart *et al.*, 1988). Parameters are optimized with the Adam stochastic gradient descent method (Kingma and Ba, 2014). All architectures are implemented with the Keras deep learning framework (Chollet *et al.*, 2015). Mini-batches of 32 examples are shown to the network which performs inference on them. The categorical cross entropy loss function is computed for the mini-batch and the gradients at each parameter are computed with respect to that loss. The weights and biases are updated according to the learning rate and update procedure of ADAM. Inference is then performed on the next mini-batch and the process repeats. Models were trained with dropout in the fully connected layers of 30% (Hinton *et al.*, 2012).

Throughout training several performance metrics are monitored to ensure the network is learning. Specifically, we compute categorical accuracy, and per label precision and recall after each mini-batch. After each epoch, the validation loss is computed. After several epochs without improvement in the validation set's loss, the learning rate decays by a factor of 0.5 until validation loss stops decreasing, at which point learning is automatically terminated, a technique known as 'Early Stopping' (Caruana *et al.*, 2001). Using these stopping criteria, 1D models converged in 6–12 h, whereas 2D models converged in 24–48 h. All models were trained with a single NVidia K-40 GPU. On a virtual machine equipped with four NVidia V-100 GPUs 2D model inference over a 40× whole genome sample can be accomplished in under 15 min (Sethia, 2019a, b). On a single-core CPU 1D model inference is still under an 1 h but without AVX accelerations 2D model inference can take up to 50 CPU hours.

There is severe class imbalance in this problem. Positive examples outnumber negatives by several orders of magnitude as can be seen in Table 1. To cope with this we oversample from the negative classes when preparing each mini-batch, such that every mini-batch has equal representation from the different classes.

2.4 Hyper-parameter optimization

Several strategies for selecting hyper-parameters were pursued. The most direct method was testing two candidate architectures and comparing performance on a held-out test set, commonly called A/B testing. Although A/B testing gives a clear choice between two architectures, it does not discover new architectures to test. Random search is a simple strategy to implement and completes exponentially faster than grid search, which can give more insight into the total space of potential architectures. Last, we employed Bayesian hyper-parameter optimization via the GPyOpt package to take more disciplined steps over potential architectures (GPyOpt, 2016). The architectures shown in Figure 1 are the results of both random and Bayesian hyper-parameter selection (Snoek *et al.*, 2012). Surprisingly, we found that vastly different architecture topologies and capacities yielded similar classification performance—performance only slightly improved from the first handcrafted architectures we applied to this problem. As a result, the big gain from hyper-parameter optimization was not in accuracy, which changed little, but in the use of computational resources, as we discovered comparably accurate architectures with far fewer parameters.

3 Results

We compare CNNs to current state-of-the-art filtering methods, including Gaussian Mixture Models as implemented in GATK version 4.1.4.0 Variant Recalibrator (VQSR) (Van der Auwera *et al.*, 2013), random forest (RF) classifiers from the gnomAD callset version 2.1.1 (Lek *et al.*, 2016) and DeepVariant version 0.7.0 (Poplin *et al.*, 2018). To evaluate our filter against DeepVariant, which is also a genotyper, we rank DeepVariant calls by site-level QUAL scores over the confident region. Table 2 shows area under the precision recall curve (auPRC) tabulated across several different truth sets as well as constituent parts of the CNNVariant 1D and 2D architectures for small and medium versions of the 2D CNN and MLP (500 000 and 3 million parameters, respectively). Higher area scores indicate better accuracy. The row labeled '2D CNN and MLP (all)', which performed the best over the most datasets, was trained on heterogeneous data aggregated across diverse samples, sequencers, references and truth resources. All other neural nets were trained only with GiaB truth labels. The CNNs presented here tended to outperform the simpler Gaussian Mixture Model and RF models and also the more complex DeepVariant model.

We see almost equivalent performance between 2D models trained with or without annotations suggesting the convolutions alone are able to reconstruct the information captured by the summary statistics. On the other hand, models incorporating annotations converged much faster. On whole genome Giab truth sets F1 score is maximized with precision for SNPs 0.999, recall 0.996 and F1 scores 0.997 and for INDELs precision 0.990 recall 0.987 and F1 score of 0.988 depending on the stringency of the filtering threshold. The BAMs included in this analysis averaged read coverage from 2 to 40×. For very low coverage BAMs (5× or less) filtering will still increase precision, but the reduction in recall yields lower F1 scores than with the unfiltered VCF (see the Supplementary Material for more results).

The critical question of any model is: does it generalize? To answer this, datasets from different samples, chemistries, truth resources and reference genomes were generated and are presented in Table 3. Comparing different GiaB samples (e.g. NA12878 versus NA24385) demonstrates how well a model generalizes across samples. Sequencer generalization is shown across HiSeq4000, HiSeqX and NovaSeq machines while chemistry generalization is measured across three different chemistries: whole exome capture (with PCR amplification), whole genome sequence (WGS) with PCR amplification and PCR-free WGS. Baseline models (columns *j* and *k*) were compared with models trained only on a single type of data (columns *a*–*i*). We measure generalization cost (colored red) or benefit (colored green) as the difference in auPRC between models trained and evaluated on the same dataset versus models trained on one dataset and evaluated on another. Predominantly red rows can be thought of as difficult datasets, while predominantly green columns indicate a model that generalizes well. The most generalization benefit came from a model trained on an aggregation of variants from many data types, shown in column *k* of Tables 3 and 4, which was also the best performing model in Table 2. Despite differences in sequencer and protocol, there is only a small generalization cost

Table 2. The architectures from Figure 1 and other published models are evaluated on held-out test sets with approximately balanced representation from each class

Architecture			auPRC								
			INDELs								
Description	Parameters (×1000)	Input tensor dimensions	NA24385 HG38 Zook and Salit (2011)	SynDip HG38 Li *et al.* (2018)	NA24149 HG38 Zook and Salit (2011)	NA12878 HG38 Zook and Salit (2011)	NA12878 HG19 Zook and Salit (2011)	NA12878 HG19 Eberle *et al.* (2013)	NA12878 HG19 Eberle *et al.* (2016)	NA12877 HG19 Eberle *et al.* (2013)	Sample Reference Truth
MLP	172	(7)	0.941	0.701	0.874	0.917	0.724	0.538	0.730	0.542	
1D CNN	576	(128, 4)	0.890	0.822	0.778	0.884	0.885	0.625	0.691	0.584	
1D CNN and MLP	932	(128, 4), (7)	0.959	0.807	0.874	0.959	0.942	0.700	0.817	0.704	
2D CNN	1271	(128, 128, 15)	0.983	0.946	**0.916**	0.983	0.911	0.729	0.868	0.677	
2D CNN and MLP (small)	529	(128, 128, 15), (7)	0.988	0.941	0.899	0.973	0.901	0.709	0.862	0.618	
2D CNN and MLP	3062	(128, 128, 15), (7)	**0.990**	0.956	0.901	**0.992**	0.920	0.749	0.865	0.653	
2D CNN and MLP (all)	529	(128, 128, 15), (7)	0.989	**0.972**	0.907	0.988	**0.952**	**0.770**	**0.915**	**0.713**	
Deep variant (Poplin *et al.*, 2018)	21 644	(299, 299, 3)	0.986	0.940	0.875	**0.992**	0.903	0.666	0.887	0.630	
RF gnomAD (Lek *et al.*, 2016)	25	(11)	0.907	0.824	0.858	0.866	0.840	0.673	0.720	0.650	
VQSR gnomAD (Lek *et al.*, 2016)	<1	(7)	0.934	0.835	0.870	0.911	0.919	0.694	0.704	0.636	
VQSR (Van der Auwera *et al.*, 2013)	<1	(7)	0.909	0.814	0.854	0.902	0.777	0.584	0.671	0.593	
			SNPs								
MLP	172	(7)	0.988	0.972	0.978	0.996	0.985	0.726	0.841	0.747	
1D CNN	576	(128, 4)	0.872	0.922	0.899	0.907	0.824	0.775	0.862	0.734	
1D CNN and MLP	932	(128, 4), (7)	0.990	0.988	0.965	0.995	0.981	0.776	0.879	0.767	
2D CNN	1271	(128, 128, 15)	**1.0**	0.998	0.979	0.999	0.992	0.899	0.968	0.885	
2D CNN and MLP (small)	529	(128, 128, 15), (7)	0.999	**0.999**	**0.989**	**1.0**	0.988	0.830	0.956	0.839	
2D CNN and MLP	3062	(128, 128, 15), (7)	**1.0**	**0.999**	0.972	**1.0**	**0.996**	0.888	0.971	0.893	
2D CNN and MLP (all)	529	(128, 128, 15), (7)	**1.0**	**0.999**	0.968	**1.0**	0.995	0.905	**0.984**	**0.901**	
Deep variant (Poplin *et al.*, 2018)	21 644	(299, 299, 3)	**1.0**	0.998	0.977	0.998	0.988	0.791	0.959	0.817	
RF gnomAD (Lek *et al.*, 2016)	25	(11)	0.993	0.994	0.983	0.988	0.988	0.907	0.943	0.878	
VQSR gnomAD (Lek *et al.*, 2016)	<1	(7)	0.951	0.981	0.941	0.953	0.987	**0.912**	0.939	0.883	
VQSR (Van der Auwera *et al.*, 2013)	<1	(7)	0.986	0.988	0.965	0.906	0.983	0.889	0.939	0.883	

Note: For each model we show the number of parameters, the size of the input and the auPRC for the listed combination of sample, reference, and truth resource, stratified by variant type (SNPs and INDELs). The top set of descriptions is models developed here, whereas the bottom set were published previously. For PG we use the 2017 release, which included a hybrid VCF and confidence region with the GiaB resource. The hybrid VCF is cited by Eberle *et al.* (2016) and the NA12877 and NA12878 PG samples are cited as Eberle *et al.* (2013). All the CNNVariant models evaluated here were trained solely with GiaB truth samples, except the model marked 'all'. Boldface indicates the highest score on each dataset.

Table 3. This table shows the generalization cost (highlighted in red) or benefit (highlighted in green) measured by the difference in auPRC as models trained on one dataset are used to filter INDELs from other datasets

Sample reference sequencer truth	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*
a) CHM HG38 HiSeqX SynDip	0.000	−0.069	−0.127	−0.054	−0.065	−0.040	−0.029	−0.063	−0.242	−0.027	0.003
b) NA24385 HG38 HiSeqX GiaB	−0.020	0.000	−0.369	−0.198	−0.175	−0.117	−0.274	−0.230	−0.286	0.010	0.013
c) NA12878 HG19 HiSeqX GiaB	−0.076	−0.135	0.000	−0.100	−0.133	−0.056	−0.025	−0.082	−0.170	−0.082	−0.034
d) NA12878 HG38 NovaSeq GiaB	−0.041	−0.051	−0.090	0.000	−0.016	−0.069	−0.027	−0.078	−0.232	0.014	0.044
e) NA24385 HG38 NovaSeq GiaB	−0.050	−0.002	−0.129	−0.001	0.000	−0.074	−0.032	−0.118	−0.315	0.016	0.037
f) NA12878 HG19 HiSeqX PG+GiaB	−0.045	−0.084	−0.126	−0.075	−0.089	0.000	−0.078	−0.107	−0.320	−0.056	−0.003
g) NA12878 HG19 HiSeqX PG	0.001	−0.015	−0.007	−0.042	−0.020	0.030	0.000	−0.019	−0.144	−0.022	0.010
h) CHM Exome HG19 HiSeq4000 SynDip	−0.014	−0.030	−0.019	−0.027	−0.029	−0.007	−0.008	0.000	−0.048	−0.024	−0.006
i) NA12878 Exome HG19 HiSeq4000 GiaB	−0.023	−0.051	−0.021	−0.026	−0.094	−0.026	−0.076	0.001	0.000	−0.023	−0.012

Note: Each cell shows the auPRC difference between a model trained on that row's dataset versus a model trained on the corresponding column's dataset. Models from columns *a–i* match the descriptions in rows *a–i*, respectively. Column *j* was trained on the NA12878, NA24143 and NA24149 samples sequenced with the Illumina HiSeqX with GiaB truth labels. Column *k* was trained on NA12878, NA24143, NA24149 and SynDip, aligned to HG19 and HG38, sequenced by both HiSeqX and NovaSeq, with labels from all three truth resources. The model from column *k* is compared against other classifiers in Table 2. The Supplementary Material contains the corresponding table for SNPs.

Table 4. This table shows the generalization cost or benefit measured by the difference in auPRC as models trained on one dataset are used to filter SNPs from other datasets

Sample reference sequencer truth	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*
a) CHM HG38 HiSeqX SynDip	0.000	−0.001	−0.001	−0.001	−0.001	−0.002	−0.001	−0.001	−0.000	−0.000	−0.000
b) NA24385 HG38 HiSeqX GiaB	−0.000	0.000	−0.061	−0.041	−0.036	−0.020	−0.055	−0.053	−0.022	0.001	0.001
c) NA12878 HG19 HiSeqX GiaB	−0.004	−0.002	0.000	−0.003	0.002	−0.005	0.002	−0.004	−0.003	−0.001	0.007
d) NA12878 HG38 NovaSeq GiaB	−0.001	−0.002	−0.001	0.000	−0.001	−0.000	−0.001	−0.003	−0.009	−0.000	0.000
e) NA24385 HG38 NovaSeq GiaB	0.000	0.001	−0.002	0.000	0.000	−0.003	0.001	−0.002	−0.036	0.001	0.002
f) NA12878 HG19 HiSeqX PG+GiaB	−0.053	−0.066	−0.067	−0.061	−0.082	0.000	−0.014	−0.070	−0.067	−0.039	−0.012
g) NA12878 HG19 HiSeqX PG	−0.036	−0.039	−0.056	−0.050	−0.055	0.012	0.000	−0.035	−0.041	−0.029	−0.001
h) CHM Exome HG19 HiSeq4000 SynDip	−0.017	−0.015	−0.002	−0.009	−0.004	−0.009	−0.002	0.000	−0.027	−0.016	−0.004
i) NA12878 Exome HG19 HiSeq4000 GiaB	−0.013	−0.012	−0.004	−0.003	−0.003	−0.003	−0.001	−0.007	0.000	−0.016	−0.001

Note: Models are the same as in the INDEL table in the main text.

to using models trained with genomes on exomes as shown in rows *h* and *i*. Since the exome is only ∼1% of the genome, exome models had much less training data and saw far fewer DNA contexts than whole genome-trained models. As a result exome models overfit much more quickly and tended to generalize poorly, e.g. column *i*.

4 Discussion

The best performing models were trained on heterogenous data from various samples, truth resources and sequencing chemistries. Once such a model is trained, we have found it tends to generalize well across datatypes. This allows model training to be performed sporadically, generating a pipeline resource that can be used for a wide variety of applications. This could be particularly useful for experimental designs using small capture panels that might not contain enough variants overlapping with the truth resources to effectively train a model. The models labeled 1 D CNN and MLP and 2D CNN and MLP (small) in Table 2 are pre-trained and packaged in the GATK. Inference on new variants is performed by the tool CNNScoreVariants. Utilities to tensorize BAM and VCF files and to train custom neural network architectures are available via the tools CNNVariantWriteTensors, CNNVariantTrain (GATK, 2019).

The models presented here filter variants from a single sample of NGS data. Future work includes extending them to filter jointly called cohorts. This requires a different method for evaluating variants given that cohorts may not include samples for which truth data are available. Another challenge is that annotation distributions have been found to vary with the number of samples containing the variant in question, which could require additional complexity for the MLP portion of the model. Another extension could incorporate more steps of the pipeline into the classification model. For example, a 1D segmenting architecture can be trained that labels the sequence of genomic positions with their genotypes for a given sample, similar to DeepVariant. The neural network is a powerful tool in the machine learning toolbox that has been successfully applied here to SNP and INDEL filtration, but promises much more extensive applications in the field of genomics.

Conflict of Interest: none declared.

References

Caruana,R. *et al.* (2001) Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In: *Advances in Neural Information Processing Systems*, pp. 402–408.

Chollet,F. *et al.* (2015). Keras. https://github.com/keras-team/keras.

Eberle,M. *et al.* (2013) Platinum genomes: a systematic assessment of variant accuracy using a large family pedigree. In: *60th Annual Meeting of the American Society of Human Genetics*, pp. 22–26.

Eberle,M.A. *et al.* (2016) A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree. *Genome Res.*

GATK. (2019). https://github.com/broadinstitute/gatk.

GPyOpt. (2016). A Bayesian optimization framework in python. https://sheffieldml.github.io/GPyOpt/.

Hinton,G.E. *et al.* (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv: 1207.0580.*

Ioffe,S. and Szegedy,C. (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. CoRR, abs/1502.03167.

Kingma,D. and Ba,J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv: 1412.6980.*

Lek,M. *et al.* (2016) Analysis of protein-coding genetic variation in 60, 706 humans. *Nature*, **536**, 285–291.

Li,H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv: 1303.3997*.

Li,H. *et al.* (2018) A synthetic-diploid benchmark for accurate variant-calling evaluation. *Nat. Methods*, **15**, 595–597. doi: 10.1038/s41592-018-0054-7.

Luo,R. *et al.* (2018). Clairvoyante: a multi-task convolutional deep neural network for variant calling in single molecule sequencing. *bioRxiv*, 310458.

McKenna,A. *et al.* (2010) The genome analysis toolkit: a map reduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, **20**, 1297–1303.

Poplin,R. *et al.* (2017). Scaling accurate genetic variant discovery to tens of thousands of samples. *bioRxiv*, 201178.

Poplin,R. *et al.* (2018) A universal SNP and small-indel variant caller using deep neural networks. *Nat. Biotechnol.*, **36**, 983.

Rosenblatt,F. (1960) Perceptron simulation experiments. *Proc. IRE*, **48**, 301–309.

Rumelhart,D.E. *et al.* (1988) Learning representations by back-propagating errors. *Cogn. Model.*, **5**, 1.

Sethia,A. (2019a). Parabricks. https://www.parabricks.com.

Sethia,A. (2019b). Parabricks. https://console.cloud.google.com/marketplace/details/parabricks/parabricks.

Snoek,J. *et al.* (2012). Practical Bayesian optimization of machine learning algorithms. In: *Advances in Neural Information Processing Systems*, pp. 2951–2959.

Stormo,G.D. *et al.* (1982) Use of the perceptron algorithm to distinguish translational initiation sites in *E. coli*. *Nucleic Acids Res.*, **10**, 2997–3011.

Szegedy,C. *et al.* (2014). Going deeper with convolutions. *arXiv preprint arXiv: 1409.4842*.

Szegedy,C. *et al.* (2016). Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.

Torracinta,R. and Campagne,F. (2016). Training genotype callers with neural networks. *bioRxiv*, 097469.

Van der Auwera,G.A. *et al.* (2013) From FASTQ data to high-confidence variant calls: the genome analysis toolkit best practices pipeline. *Curr. Protoc. Bioinformatics*, **43**, 11–10.

Zaheer,M. *et al.* (2017) Deep sets. In: *Advances in Neural Information Processing Systems*, pp. 3391–3401.

Zook,J.M. and Salit,M. (2011) Genomes in a bottle: creating standard reference materials for genomic variation-why, what and how? *Genome Biol.*, **12**, P31.