TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 05/16 pp665–672
DOI: 10.26599/TST.2022.9010032
Volume 28, Number 4, August 2023

DeepFilter: A Deep Learning Based Variant Filter for VarDict

Hao Zhang, Zekun Yin*, Yanjie Wei, Bertil Schmidt, and Weiguo Liu*

Abstract: With the development of sequencing technologies, somatic mutation analysis has become an important component in cancer research and treatment. VarDict is a commonly used somatic variant caller for this task. Although the heuristic-based VarDict algorithm exhibits high sensitivity and versatility, it may detect higher amounts of false positive variants than callers, limiting its clinical practicality. To address this problem, we propose DeepFilter, a deep-learning based filter for VarDict, which can filter out the false positive variants detected by VarDict effectively. Our approach trains two models for insertion-deletion mutations (InDels) and single nucleotide variants (SNVs), respectively. Experiments show that DeepFilter can filter at least 98.5% of false positive variants and retain 93.5% of true positive variants for InDels and SNVs in the commonly used tumor-normal paired mode. Source code and pre-trained models are available at https://github.com/LeiHaoa/DeepFilter.

Key words: variant filter; deep learning; somatic variant

1 Introduction

Cancer is a lethal genetic disease of the human genome. Finding mutations between cancer and normal tissues (somatic mutations) is the key to understanding the cause of cancers. The rapid progress of next-generation sequencing (NGS) has recently made whole-exome sequencing (WES) and whole-genome sequencing (WGS) feasible and less expensive. Consequently, several somatic variant calling tools for NGS data have been proposed over the last decade[1–6]. However, variability in the accuracy of somatic mutation detection may affect the discovery of alterations and the therapeutic management of cancer patients.

• Hao Zhang, Zekun Yin, and Weiguo Liu are with the School of Software, Shandong University, Jinan 250100, China. E-mail: haoz@mail.sdu.edu.cn; {zekun.yin, weiguo.liu}@sdu.edu.cn.
• Zekun Yin and Weiguo Liu are also with the Shenzhen Research Institute of Shandong University, Shenzhen 518057, China.
• Yanjie Wei is with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: yj.wei@siat.ac.cn.
• Bertil Schmidt is with the Institute for Computer Science, Johannes Gutenberg University, Mainz 55128, Germany. E-mail bertil.schmidt@uni-mainz.de.
* To whom correspondence should be addressed.
Manuscript received: 2022-02-26; revised: 2022-04-27; accepted: 2022-08-22

VarDict[7] is a commonly used somatic variant calling tool used in clinical research. It can call insertion-deletion mutations (InDels), single nucleotide variants (SNVs), and complex variants simultaneously based on a series of heuristic conditions and local realignment. VarDict employs a Fisher-exact test to obtain signature differences between tumor and normal samples. It also performs other heuristic conditions, such as variant allele frequency and mapping quality, to filter false positive variants. An evaluation of eight variant calling tools (HaplotypeCaller, Platypus, VarScan, LoFreq, FreeBayes, SNVer, SAMtools, and VarDict) on non-matched data revealed that VarDict outperforms all other contenders[8]. Reference [9] has reviewed four commonly used variant callers (VarDict, Strelka, Mutect2, and VarScan2) using tumor-normal paired data provided by the National Center for Clinical Laboratories in China (NCCL). Results demonstrate that the heuristic-based caller VarDict achieves the highest recall among all callers in tumor-normal and tumor-only mode. A synthetic data based evaluation[10] also shows that VarDict can detect more true variants than others, but at the same time, VarDict introduces more false positive variants.

In summary, these evaluations indicate that the heuristic-based VarDict is highly sensitive but lacks

precision. In most cases, VarDict introduces more false positives than other callers. Confounding factors, such as the contamination of tumor and normal samples, the sub-clonal heterogeneity of cancer samples, the artifacts introduced by sequencing error, the misalignment in the alignment step, and the incompleteness of reference genomes, lead to many false positive somatic variant calls. Thus, removing large amounts of false positive calls would be highly beneficial. Effective filtering strategies can improve the accuracy and efficiency of cancer diagnosis and treatment.

General-purpose variant filters include SnpEff[11], VCFTools[12], and FiNGS[13]. However, these tools need users to specify filtering conditions. For instance, SnpEff and VCFTools require users to specify their own filtering rules, whereas FiNGS needs a configuration file to specify the filter parameters and thresholds.

Furthermore, many variant calling tools have their own filter modules. For example, Strelka2[3] uses an empirical variant scoring (EVS) step to produce a single aggregate score for the detected variants. This step takes multiple features, such as strand bias, genotype probability, and mapping quality, to train a random forest model. VarDict[7] uses a Perl-based hard filter to remove certain false positive variants and format the detected variants into variant call format (VCF) files. The latest release of VarScan2[4] includes the fpfilter module to filter false positive variants.

Recently, deep-learning based methods, such as DeepVariant[6], NeuSoamtic[14], and Clairvoyante[5], have been proposed for variant detection. DeepVariant is based on deep learning technology to detect variants from germline data. It converts sequence alignment data of potential variant candidates into 221×100 "pileup" images and uses convolutional neural networks (CNNs) to process them. NeuSomatic summarizes alignment information around candidate variants by converting them into $k \times 5 \times 32$ feature matrices using k channels representing different features (sequence, quality scores, etc.), and then passes these data into a CNN network. Other methods use ensemble learning to integrate the result from a series of callers. Aside from NeuSomatic, the Roche Sequencing team also provides an ensemble learning caller named SomaticSeq, which ensembles the detection results from a series of callers, including VarDict, Strelka2, and VarScan2, to improve the sensitivity of somatic variant calling. DeepVariant can only detect germline variants, NeuSomatic is designed for somatic mutations, and Cairvoyante is designed for single molecule sequencing. Machine-learning based methods treat the variant calling task as a classification problem. In general, deep-learning based methods have been shown to achieve high precision on certain datasets but may perform worse on specific datasets that are very different from their training data[15].

Some filtering tools are designed for a specific variant caller. For example, FilterMutectCall is designed for filtering variants called from Mutect2, and GARFIELD-NGS[16] is designed to filter variants from GATK HaplotypeCaller. GARFILED-NGS trains four prediction models optimized for InDels and SNVs for Illumina and ION platforms, respectively. Variant caller specific filters can exploit the variant detection information generated by the corresponding caller.

Inspired by these recent results, we introduce DeepFilter, a deep-learning based variant filter designed for VarDict to filter false positive variants effectively while ensuring high calling sensitivity.

2 Method

There are three main steps in DeepFilter:

(1) A hard filter strategy is used for the intermediate results produced by VarDict. Variants that match these conditions are filtered.

(2) The filtered data are input to the neural network for inference.

(3) The filtered data are formatted into VCF files.

The workflow is shown in Fig. 1. Details of each step will now be described in Sections 2.1 and 2.2.

2.1 Hard Filter

To improve the sensitivity, DeepFilter firstly performs

Fig. 1 Workflow of DeepFilter: Raw data are the VarDict intermediate results, which are processed by a hard filter and then input into a deep-learning based filter. The output of the network is converted into VCF format.

a hard filter to remove variants that are highly unlikely to be true variants. We analyzed all training datasets to find the conditions for the hard filter. The statistical results of our analysis are shown in Fig. 2. Based on these statistical results, we chose the following hard filter conditions to remove variants:

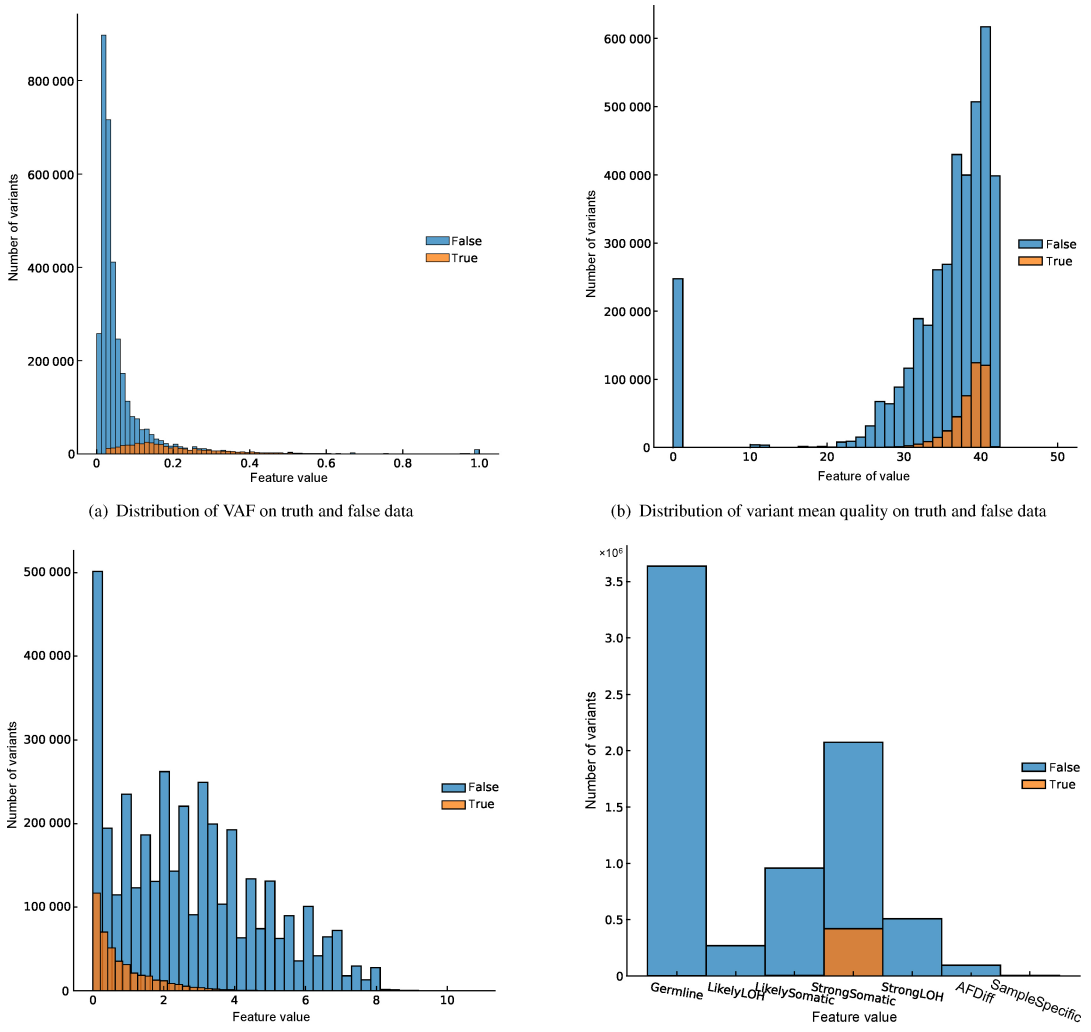
- Tumor sample allele frequency $\leqslant 0.01$;
- Tumor sample mean quality $\leqslant 20$;
- Tumor sample variant mismatch $\geqslant 6$;
- Variant type is Germline.

The hard filter conditions we chose were relatively loose, ensuring that the true variants will most likely not be filtered in this step.

2.2 Network

Filtering false positive variants is important because an error rate of 1:10 000 in a genome with 3 billion positions would result in many false calls, which may lead to positives may also lead to missed diagnoses. Thus, the goal of DeepFilter is to filter false positives effectively while retaining as many as possible true positive variants. In our opinion, "keeping true variants" can be regarded as equally important as "removing false variants". Thus, based on the considerations above, the main principle for designing DeepFilter is to remove as many false positive variants as possible while retaining most true variants.

DeepFilter treats the task of distinguishing whether a variant is true or false as a binary classification problem. We trained two MLP networks for SNV and InDel variants using PyTorch. The structure of the network is illustrated in Fig. 3. The MLP network included one input layer, four hidden layers, and one output layer. The input layer size is the feature number we selected. The hidden layer sizes of the SNV model are 80, 120, 140, 120, and 20, and the hidden layer sizes of the InDel model are 140, 160, 170, 100, and 10. The output



(a) Distribution of VAF on truth and false data

(b) Distribution of variant mean quality on truth and false data

(c) Distribution of number of mismatch on truth and false data

(d) Distribution of variant label on truth and false data

Fig. 2 Distribution of Var1AF, Var1QMean, Var1NM, and VarLabel in all datasets we used.

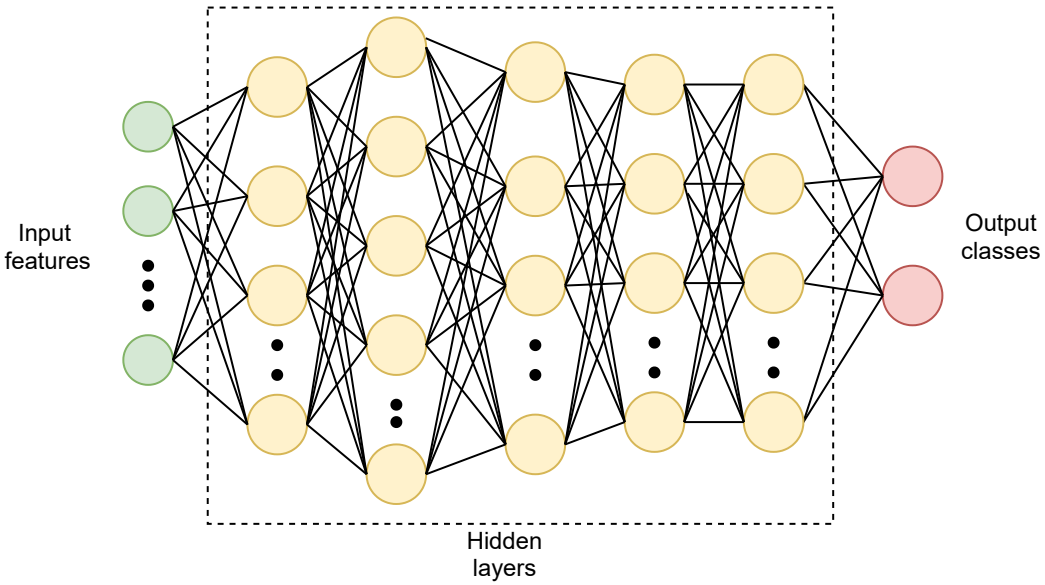


Fig. 3 Network of DeepFilter. We trained two distinct models based on the multi-layer perceptron algorithm. The hidden layer sizes of the SNV model are 80, 120, 140, 120, and 20, and the hidden layer sizes of the InDel model are 140, 160, 170, 100, and 10.

layer size is 2. At the beginning of training, the weights of each layer were initialized using a Xavier uniform distribution[17]. We used leak ReLU as the active function. Each layer was proceeded by a dropout operation with drop rate $p = 0.5$ to avoid overfitting. In addition, we used Adam[18] optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 10^{-9}$. The learning rate is varied over the course of training, according to the formula: $lr = lr \times 0.9^e$, where e is the number of epochs, and the initial learning rate $lr = 0.001$.

We adopted a weighted cross-entropy loss function in this network to balance the filtration intensity and true variant retention rate. The loss of each class is defined as

$$loss(x, class) = weight[class]\left(-x[class] + \log\left(\sum_j \exp(x[j])\right)\right)$$

Thus, the average loss is

$$loss = \frac{\sum_{i=1}^{N} loss(i, class[i])}{\sum_{i=1}^{N} weight(class[i])}$$

In our method, we assigned a higher weight to the positive class. Thus, when the network filters a true variant, it will have a more severe punishment than when it retains a false positive variant. Hence, more true variants will be retained. After careful consideration, we selected a 1:24 ratio of false and true as the default weight to train the network. Nevertheless, users could also define other weights when they re-train the network.

DeepFilter uses different networks in SNV and InDel filtering tasks. VarDict generates many variant features. In the commonly used tumor-normal paired mode, we selected 45 and 42 features for InDels and SNVs, respectively. Each sample included 18 numerical features (totalPosCov, posCoverage, refFwCov, refRvCov, varsFwCount, varsRvCount, frequency, meanPosition, pstd, pvalue, meanQuality, ratio, mapq, qratio, higreq, nm, hicnt, and hicov), five numerical features of the variant (shift3, msi, msint, pvalue, and ratio), and one factor feature (varLabel). The feature varLabel included six classes (including StrongSomatic, LikelySomatic, StrongLOH, LikelyLOH, Germline, and Adiff) to indicate the type of a variant.

We selected more features for the InDel network than for the SNV network. Additional features included the length of reference and alter sequence and varType (Insertion, Deletion, and complex). The description of each feature is listed in Table 1. We performed standardization for numerical features and one-hot encoding for factor features. Thus, the number of features for SNV and InDel networks were changed to 48 and 53, respectively.

3 Result

3.1 Dataset

To ensure the generalization of the neural network, we used three types of datasets to train and test DeepFilter:

Table 1 Description of selected features.

Feature	Description
totalPosCov	Total coverage
posCoverage	Variant coverage
refFwCov	Forward coverage of reference
refRvCov	Reverse coverage of reference
varsFwCount	Forward coverage of variant
varsRvCount	Reverse coverage of variant
frequency	Allele frequency
meanPosition	Mean position in read
pstd	Flag for read position standard deviation (1 if the variant is covered by at least 2 read segments with different positions, otherwise 0).
p-value	The p-value from Fisher test
meanQuality	Mean base quality
ratio	Odd ratio from Fisher test
mapq	Mean mapping quality
qratio	Ratio of high-quality reads to low-quality reads
higreq	Variant frequency for high-quality reads
shift3	Number of bases to be shifted to 3 prime for deletions due to alternative alignment
msi	Microsatellite > 1 indicates MSI
msint	Microsatellite unit length in base pair (bp)
nm	Average number of mismatches for reads containing the variant
hicnt	Number of high-quality reads with the variant
hicov	Position coverage by high quality reads

(1) **Real-world tumor-normal sample data.** We used the SEQC-II data[19] provided by the Somatic Mutation Working Group of SEQC-II consortium. The consortium aims to develop guidelines for somatic mutation detection and provide golden-standard variant sets for widely accepted paired tumor-normal reference samples/materials (HCC1395 and HCC1395BL). The raw sequencing data were downloaded from https://sites.google.com/view/seqc2/home/sequencing.

(2) **Mixture of two golden-standard data.** We used the data obtained by The Hartwig Medical Foundation and Utrecht Medical Center after simulating somatic cancer calling using a mixture of two Genome in a Bottle (GIAB) samples NA12878 and NA24385. NA12878 was considered as the "tumor" mixed with NA24385. The "tumor-normal" pair was generated by physical mixing of samples prior to sequencing. We downloaded the data using scripts provided by bcbio-nextgen[20] tutorial website (https://bcbio-nextgen.readthedocs.io).

(3) **Synthetic data.** We used BamSurgen[21] to generate in silico data. We generated two tumor-normal paired alignment datasets by BamSurgen: one for training and the other for validation. This step ensures that the training data will not be treated as validation data. We generated the synthetic data based on the Fudan University sequencing sample from the SEQC-II project, creating SNV and InDel variants with user-defined parameters. In our experiments, we generated variants with a minimal variant frequency of 0.01 and a maximum variant frequency of 1.0. The distribution of mutation frequency is β-distribution with parameters $\alpha = 2$ and $\beta = 5$. And the number of minimal variant reads is 3.

We used two types of datasets to train DeepFilter. First, we used a mixture of NA12878 and NA24385. Second, we generated a synthetic dataset including 500 000 SNV variants and 400 000 InDel variants. Then, we mixed the two types of datasets as training datasets for DeepFilter training. The detailed training methods and scripts are listed in DeepFilter's GitHub repository (https://github.com/LeiHaoa/DeepFilter).

As for the test datasets, we used the raw sequencing data of the HCC1395 sample from Fudan University (FD2) to evaluate the performance of DeepFilter on real-world tumor-normal data. We also generated 20 000 SNVs and 8000 InDels to evaluate the performance of the network in DeepFilter. Neither of the two testing datasets was trained before.

3.2 Evaluation results

We compared the filtering performance of DeepFilter, VarDict's default hard filter (BuiltinFilter), and the filter in bcbio-nextgen pipeline (bcbio). Bcbio-nextgen is a Python toolkit providing best-practice pipelines for fully automated high throughput sequencing analysis. It also provides the best-practice filter pipeline of VarDict. The pipeline integrates VarDict's builtin hard filter scripts and another carefully tuned hard filter. Then the pipeline uses a call_somatic module to calculate the genotype likelihoods. These likelihoods were compared with user-defined thresholds to filter false variants. In our evaluation, we used the default settings of bcbio-nextgen. VarDict's hard filter supports users to specify extra hard filter conditions according to the sample or sequencing information. However, to be fair, we also used the default parameters provided in VarDict's hard filter.

$$FFR = 1 - \frac{FP_{filtered\ data}}{FP_{raw\ data}} \tag{1}$$

$$TRR = 1 - \frac{TP_{filtered\ data}}{TP_{raw\ data}} \tag{2}$$

The evaluation results are shown in Tables 2–5. We recorded the false positive filtration rate (FFR) and true positive variants retention rate (TRR) of each filter.

Table 2 Performance comparison of DeepFilter, bcbio, and VarDict's default hard filter in filtering SNV variants using the synthetic dataset Syndata.

Tool	TP	FP	FFR	TRR	Recall	Prec	F1-score
Ground truth	18 773	–	–	–	–	–	–
Without filter	17 424	9 765 540	0	1.000	0.928	0.002	0.004
DeepFilter	**16 793**	39 094	**0.996**	**0.964**	**0.895**	0.300	**0.449**
Bcbio	15 711	**36 382**	**0.996**	0.902	0.837	**0.302**	0.444
BuiltinFilter	15 785	5 944 976	0.391	0.906	0.841	0.003	0.006

Table 3 Performance comparison of DeepFilter, bcbio, and VarDict's default hard filter in filtering SNV variants using the real-world dataset FD2.

Tool	TP	FP	FFR	TRR	Recall	Prec	F1-score
Ground truth	39 525	–	–	–	–	–	–
Without filter	37 286	9 806 939	0	1.000	0.943	0.004	0.008
DeepFilter	34 844	**90 559**	**0.991**	0.935	0.882	**0.278**	**0.423**
bcbio	**35 818**	127 750	0.987	**0.961**	**0.906**	0.219	0.353
BuiltinFilter	35 727	6 026 155	0.386	0.958	0.904	0.006	0.012

Table 4 Performance comparison of DeepFilter, bcbio, and VarDict's default hard filter in filtering InDel variants using the synthetic dataset Syndata.

Tool	TP	FP	FFR	TRR	Recall	Prec	F1-score
Ground truth	7487	–	–	–	–	–	–
Without filter	6227	1 785 748	0	1.000	0.832	0.003	0.006
DeepFilter	**5903**	**9687**	**0.995**	**0.948**	**0.788**	**0.379**	**0.512**
bcbio	4661	23 677	0.987	0.749	0.623	0.164	0.260
BuiltinFilter	4828	1 160 090	0.350	0.775	0.645	0.004	0.008

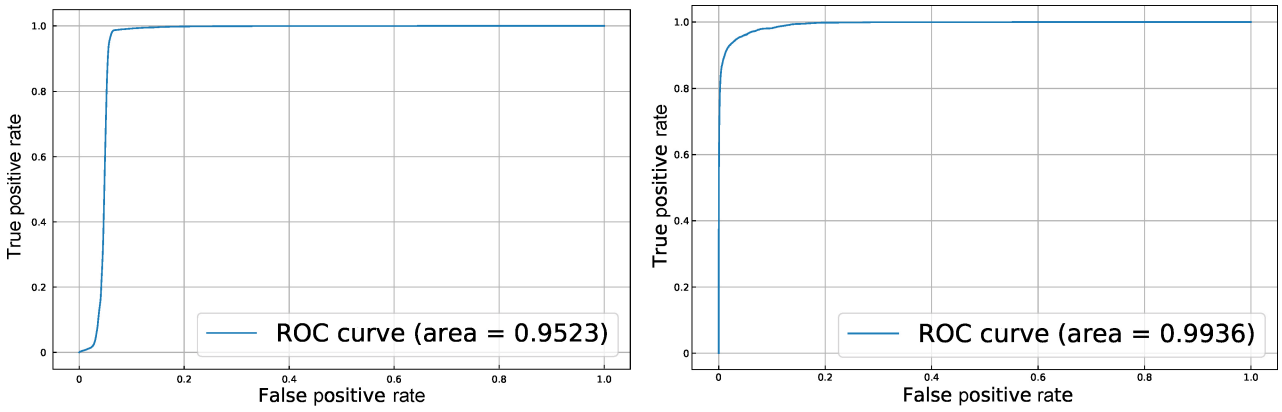
Equations (1) and (2) show the definitions of FFR and TRR. In this evaluation, raw data correspond to the intermediate result of VarDict, and filtered data are the remaining variant calls after filtration. Furthermore, we recorded the overall recall, precision, and F1-score (the harmonic mean of precision and recall) of each variant calling pipeline. In addition, the receiver operating characteristic (ROC) curves on the SNV and InDel modes were drawn to evaluate the pre-trained model (Fig. 4). The area under the curve (AUC) values of the SNV and InDel models were 0.95 and 0.99, respectively.

Table 5 Performance comparison of DeepFilter, bcbio, and VarDict's default hard filter in filtering InDel variants using the real-world dataset FD2.

Tool	TP	FP	FFR	TRR	Recall	Prec	F1-score
Ground Truth	1921	–	–	–	–	–	–
Without filter	1845	1 694 382	0	1.000	0.960	0.001	0.002
DeepFilter	1764	**25 897**	**0.985**	0.956	0.900	**0.064**	**0.120**
Bcbio	**1769**	48 216	0.972	**0.959**	**0.921**	0.035	0.067
BuiltinFilter	1753	1 172 550	0.308	0.950	0.913	0.001	0.002

Tables 2 and 3 show that in the SNV mode, VarDict's default hard filter is only able to filter 39.1% (38.6%) false positive variants on the Syndata (FD2) dataset at the expense of losing 9.4% (4.2%) true variants. The heuristic-based VarDict introduced more false positives than true variants. Thus, the overall variant calling precision only slightly improved when the low FFR hard filter was used. For example, the precision only improved from 0.2% to 0.3% by using the hard filter (Table 2). By combining hard filters and genotype likelihood estimation, the filter in bcbio-nextgen pipeline can filter out 99.6% (98.7%) false positive variants and retain 83.7% (96.1%) true variants. DeepFilter can filter 99.6% (99.1%) on the Syndata (FD2) dataset. Although bcbio-nextgen's filter and DeepFilter have similar filter rates on Syndata, DeepFilter can retain more true variants. Thus, the pipeline with DeepFilter had higher F1-scores than the other pipelines.

In the InDel mode, VarDict's default hard filter has limited improvement to the variant calling results. Compared with this BuiltinFilter, Bcbio-nextgen's filter performed better, and it can filter 98.7% (97.2%) false positive variants. Nevertheless, DeepFilter still



(a) ROC curve and AUC of DeepFilter in SNV mode

(b) ROC curve and AUC of DeepFilter in InDel mode

Fig. 4 ROC curve of DeepFilter models.

outperformed bcbio-nextgen's filter, especially on the Syndata dataset. DeepFilter filtered more false positive variants while retaining more true positive variants than bcbio-nextgen's filter. In summary, the DeepFilter-based pipeline achieved the highest F1-sore on the SNV and InDel modes.

4 Conclusion

We proposed DeepFilter, a deep-learning based filter for removing false positive somatic variants called by VarDict. Our experimental results based on synthetic and real-world NGS data showed that DeepFilter can effectively filter out large amounts of false positive variants while retaining most true positive variants. In addition, we provided multiple pre-trained models for InDel and SNV variants. DeepFilter outperformed VarDict's default builtin hard filter and bcbio-nextgen's filter in terms of FFR and TRR, which made VarDict more valuable in practical clinical research and greatly facilitated downstream analysis in biological research and patient treatment. Source code and pre-trained models are available at https://github.com/LeiHaoa/DeepFilter.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (NSFC) (Nos. 62102231 and 61972231); the Shenzhen Basic Research Fund (No. JCYJ20180507182818013); the Key Project of Joint Fund of Shandong Province (No. ZR2019LZH007); Shandong Provincial Natural Science Foundation (No. ZR2021QF089); the PPP project from CSC and DAAD; and Engineering Research Center of Digital Media Technology, Ministry of Education, China.

References

[1] D. Benjamin, T. Sato, K. Cibulskis, G. Getz, C. Stewart, and L. Lichtenstein, Calling somatic SNVs and indels with mutect2, arXiv preprint arXiv: 10.1101/861054, 2019.

[2] E. Garrison and G. Marth, Haplotype-based variant detection from short-read sequencing, arXiv preprint arXiv: 1207.3907, 2012.

[3] S. Kim, K. Scheffler, A. L. Halpern, M. A. Bekritsky, E. Noh, M. Kallberg, X. Chen, Y. Kim, D. Beyter, P. Krusche，et al., Strelka2: Fast and accurate calling of germline and somatic variants, *Nature Methods*, vol. 15, no. 8, pp. 591–594, 2018.

[4] D. C. Koboldt, Q. Zhang, D. E. Larson, D. Shen, M. D. McLellan, L. Lin, C. A. Miller, E. R. Mardis, L. Ding, and R. K. Wilson, Varscan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing, *Genome Res.*, vol. 22, no. 3, pp. 568–576, 2012.

[5] R. Luo, F. J. Sedlazeck, T. -W. Lam, and M. C. Schatz, A multi-task convolutional deep neural network for variant calling in single molecule sequencing, *Nature Communications*, vol. 10, no. 1, p. 998, 2019.

[6] R. Poplin, P. -C. Chang, D. Alexander, S. Schwartz, T. Colthurst, A. Ku, D. Newburger, J. Dijamco, N. Nguyen, P. T. Afshar, et al., A universal SNP and small-indel variant caller using deep neural networks, *Nature Biotech.*, vol. 36, no. 10, pp. 983–987, 2018.

[7] Z. Lai, A. Markovets, M. Ahdesmaki, B. Chapman, O. Hofmann, R. McEwen, J. Johnson, B. Dougherty, J. C. Barrett, and J. R. Dry, Vardict: A novel and versatile variant caller for next-generation sequencing in cancer research, *Nucleic Acids Research*, vol. 44, no. 11, p. e108, 2016.

[8] S. Sandmann, A. O. D. Graaf, M. Karimi, B. A. V. D. Reijden, E. Hellström-Lindberg, J. H. Jansen, and M. Dugas, Evaluating variant calling tools for non-matched next-generation sequencing data, *Scientific Rep.*, vol. 7, no. 1, p. 43169, 2017.

[9] X. He, S. Chen, R. Li, X. Han, Z. He, D. Yuan, S. Zhang, X. Duan, and B. Niu, Comprehensive fundamental somatic variant calling and quality management strategies for human cancer genomes, *Briefings in Bioinformatics*, vol. 22, no. 3, p. bbaa083, 2021.

[10] X. Bian, B. Zhu, M. Wang, Y. Hu, Q. Chen, C. Nguyen, B. Hicks, and D. Meerzaman, Comparing the performance of selected variant callers using synthetic data and genome segmentation, *BMC Bioinformatics*, vol. 19, no. 1, p. 429, 2018.

[11] P. Cingolani, A. Platts, L. L. Wang, M. Coon, T. Nguyen, L. Wang, S. J. Land, X. Lu, and D. M. Ruden, A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3, *Fly*, vol. 6, no. 2, pp. 80–92, 2012.

[12] P. Danecek, A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. DePristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T. Sherry, et al., The variant call format and VCFtools, *Bioinformatics*, vol. 27, no. 15, pp. 2156–2158, 2011.

[13] C. P. Wardell, C. Ashby, and M. A. Bauer, FiNGS: High quality somatic mutations using filters for next generation sequencing, *BMC Bioinformatics*, vol. 22, no. 1, p. 77, 2021.

[14] S. M. E. Sahraeian, R. Liu, B. Lau, K. Podesta, M. Mohiyuddin, and H. Y. K. Lam, Deep convolutional neural networks for accurate somatic mutation detection, *Nature Communications*, vol. 10, no. 1, p. 1041, 2019.

[15] M. Wang, W. Luo, K. Jones, X. Bian, R. Williams, H. Higson, D. Wu, B. Hicks, M. Yeager, and B. Zhu, SomaticCombiner: Improving the performance of somatic variant calling based on evaluation tests and a consensus approach, *Scientific Reports*, vol. 10, no. 1, p. 12898, 2020.

[16] V. Ravasio, M. Ritelli, A. Legati, and E. Giacopuzzi, Garfield-NGS: Genomic variants filtering by deep learning models in NGS, *Bioinformatics*, vol. 34, no. 17, pp. 3038–3040, 2018.

[17] X. Glorot and Y. Bengio, Understanding the difficulty

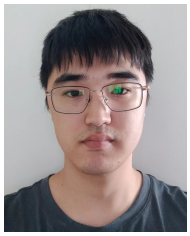
of training deep feedforward neural networks, in *Proc. 13th International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, 2010, pp. 249–256.

[18] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv: 1412.6980, 2014.

[19] W. Chen, Y. Zhao, X. Chen, Z. Yang, X. Xu, Y. Bi, V. Chen, J. Li, H. Choi, B. Ernest, et al., A multicenter study benchmarking single-cell RNA sequencing technologies using reference samples, *Nature Biotechnology*, vol. 39, no. 9, pp. 1103–1114, 2020.

[20] R. V. Guimera, Bcbio-nextgen: Automated, distributed next-gen sequencing pipeline, *Embnet Journal*, vol. 17, no. B, p. 30, 2011.

[21] A. D. Ewing, K. E. Houlahan, Y. Hu, K. Ellrott, C. Caloian, T. N. Yamaguchi, J. C. Bare, C. P'ng, D. Waggott, V. Y. Sabelnykova, et al., Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection, *Nature Methods*, vol. 12, no. 7, pp. 623–630, 2015.



Hao Zhang received the bachelor degree from Shandong University in 2017. He is currently pursuing the PhD degree with the School of Software, Shandong University, Jinan, China. His major research interests are high performance computing and its application in life science.

Zekun Yin is a postdoctoral researcher at Shandong University. He received the bachelor and PhD degrees from Shandong University in 2014 and 2020, respectively. His major research interests are high performance computing and its application in life science and geoscience. He is an expert in the design, implementation, and deployment of parallel bioinformatics algorithms, applications, and libraries on multi- and many-core HPC platforms. He won ACM Gordon Bell Prize in 2017 for the research work on Tangshan earthquake simulation on the Sunway TaihuLight supercomputer.

Yanjie Wei is a professor and the executive director in Center for High Performance Computing, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. He received the PhD degree from Michigan Tech University in the field of computational biophysics in 2007. From 2008 to 2011, he worked as a postdoctoral research associate at Princeton University. His research focuses on high performance computing and bioinformatics. He is serving as the editorial member for *Future Generation Computer Systems* and *Interdisciplinary Sciences: Computational Life Sciences*. He has published more than 100 peer reviewed journal/conference papers, including *Nucleic Acids Research*, *PloS Computational Biology*, *Briefings in Bioinformatics*, *Bioinformatics*, *Cell Research*, *BMC Bioinformatics*, *Proteins*, *Journal of Chemical Theory and Computation*, *Journal of Physical Chemistry B*, ICPP2016, ICPP2018, and PPoPP2015.

Bertil Schmidt is currently a tenured full professor and the chair of parallel and distributed architectures with Johannes Gutenberg University, Germany. Prior to that, he was a faculty member with Nanyang Technological University, Singapore, and with University of New South Wales. His research group has designed a variety of algorithms and tools for bioinformatics, mainly focusing on the analysis of large-scale sequence and short read datasets, and data mining. He is a senior member of IEEE. For his research work, he was the recipient of CUDA Research Center Award, CUDA Academic Partnership Award, CUDA Professor Partnership Award, and Best Paper Award at IEEE ASAP 2009 and IEEE ASAP 2015.

Weiguo Liu received the bachelor and master degrees from Xi'an Jiaotong University, China, and the PhD degree from Nanyang Technological University, Singapore. He is currently a full professor and the director of High-Performance Computing Lab at School of Software, Shandong University. His research interests include high-performance computing, bioinformatics, and data mining. His research group has designed tools and algorithms for applications in data processing and computational science using parallel computing technologies such as CUDA-enabled GPUs, CPU or GPU or Xeon Phi clusters, and supercomputers. He was the recipient of numerous awards, including the ACM Gordon Bell Prize Award, Fraunhofer IGD Best Paper Award, and CCF HPC China Best Paper Award.