Artificial Intelligence in the Life Sciences 1 (2021) 100011

Contents lists available at ScienceDirect

Artificial Intelligence in the Life Sciences

journal homepage: www.elsevier.com/locate/ailsci

Methods & Protocols

Fast-bonito: A faster deep learning based basecaller for nanopore sequencing

Zhimeng Xu[a,#], Yuting Mai[a,#], Denghui Liu[a,#], Wenjun He[a], Xinyuan Lin[a], Chi Xu[a], Lei Zhang[a], Xin Meng[a], Joseph Mafofo[b], Walid Abbas Zaher[b], Ashish Koshy[b], Yi Li[a], Nan Qiao[a,*]

[a] *Laboratory of Health Intelligence, Huawei Technologies Co. Ltd, Shenzhen 518100, China*
[b] *Group42 HealthCare, Abu Dhabi, United Arab Emirates*

ARTICLE INFO

Keywords:
Bonito
Fast-Bonito
Base calling
Nanopore
Ascend chip
Deep neural network
Neural architecture search

ABSTRACT

Nanopore sequencing from Oxford Nanopore Technologies (ONT) is a promising third-generation sequencing (TGS) technology that generates relatively longer sequencing reads compared to the next-generation sequencing (NGS) technology. A basecaller is a piece of software that translates the original electrical current signals into nucleotide sequences. The accuracy of the basecaller is crucially important to downstream analysis. Bonito is a deep learning-based basecaller recently developed by ONT. Its neural network architecture is composed of a single convolutional layer followed by three stacked bidirectional gated recurrent unit (GRU) layers. Although Bonito has achieved state-of-the-art base calling accuracy, its speed is too slow to be used in production. We therefore developed Fast-Bonito, by using the neural architecture search (NAS) technique to search for a brand-new neural network backbone, and trained it from scratch using several advanced deep learning model training techniques. The new Fast-Bonito model balanced performance in terms of speed and accuracy. Fast-Bonito was 153.8% faster than the original Bonito on NVIDIA V100 GPU. When running on HUAWEI Ascend 910 NPU, Fast-Bonito was 565% faster than the original Bonito. The accuracy of Fast-Bonito was also slightly higher than that of Bonito. We have made Fast-Bonito open source, hoping it will boost the adoption of TGS in both academia and industry.

Introduction

In the last several decades, genome sequencing technologies have evolved from Sanger sequencing [1] to massively parallel next-generation sequencing (NGS) [2], and now long-read third-generation sequencing (TGS) [3]. Compared to NGS, TGS can produce longer reads, which makes it a better choice to study complex variations of genomes. Several different TGS technologies have been developed, such as PacBio and Oxford Nanopore Technologies (ONT) platform [3]. Nanopore sequencing powered by ONT can generate very long reads by recording the electrical resistance signals of a single strand of DNA passing through protein nanopores [4]. It has shown great advantages in sequencing long reads and detecting complex genome structure variations [4,5], but the high sequencing error rate slowed down industrial adoption in many areas.

One very critical step that contributes to the high error rate is base calling, which translates raw electronic signals into bases, i.e., ATCG. The electrical signals are supposed to be determined by the nucleotides residing in the nanopores so they can be decoded into nucleotides bases. Unfortunately, they are also vulnerable to sequencing noises and the emergence of DNA changes, making the signals too complicated to be decoded [6] efficiently. Existing machine learning approaches for base calling include Albacore, Guppy, Scappie and Flappie (https://github.com/nanoporetech/flappie) released by ONT; research communities have also contributed various tools, such as Nanocall [7], DeepNano [8], Chrion [9] and causalcall [10]. Although several different approaches have been used for base calling [6–12], it is still quite difficult to balance the speed and accuracy of basecallers. In recent years, deep neural networks have also been used widely for base calling, with popular architectures like convolutional neural network (CNN), recurrent neural network (RNN) and connectionist temporal classification (CTC) decoder [10]. Guppy (https://nanoporetech.com/about-us/news/new-research-algorithms-yield-accuracy-gains-nanopore-sequencing) outperforms other tools in terms of speed and delivers a relatively high accuracy [6]. Bonito (https://github.com/nanoporetech/bonito) achieves state-of-the-art base calling accuracy, representing a significant improvement of over 1% compared to Guppy. However, Bonito is quite slow, which limits its application in practice. Therefore, we developed Fast-Bonito, by using the neural architecture search (NAS) technique to search for a new backbone from scratch, and trained it using several advanced training

* Corresponding author.
E-mail address: qiaonan3@huawei.com (N. Qiao).
# These authors contributed equally.

https://doi.org/10.1016/j.ailsci.2021.100011
Received 2 October 2021; Received in revised form 4 November 2021; Accepted 5 November 2021
Available online 8 November 2021

techniques. Now we have a new model that is well balanced in terms of speed and accuracy.

Search for new optimized neural network backbone

The neural network backbone of Bonito is inspired by QuartzNet [13], which was originally developed for speech recognition, so we thought it might not be the best neural network architecture for translating raw electronic signals into bases. We also investigated into the Bonito neural network architecture to find potential bottlenecks limiting its performance. The neural network backbone of Bonito consists of multiple TCSConv-BN-ReLU modules. Each TCSConv-BN-ReLU module consists of several time-channel separable convolutions (TCSConv), a batch normalization (BN) layer, and a ReLU activation function. Although this architecture reduces the number of parameters of the model dramatically, the inference engine library for the hardware is not well written, which slows it down to some extent.

To speed up Bonito, we use neural architecture search (NAS) - a technique for automating the design of artificial neural networks, to search for a new neural network for the same task. Earlier NAS frameworks only focus on searching for higher-performance modules, which might result in a backbone that satisfies accuracy requirements, but takes longer for inference (higher latency). To balance the accuracy and inference latency, a multi-objective and adaptive NAS framework (a so called MnasNet [14]) was used in our work. The automated neural architecture search approach explicitly incorporated latency information into the main objective so that the search can identify a model that achieves a good trade-off between accuracy and latency.

Search space

The NAS search was focused on the architecture of five middle blocks. First, all separable convolution modules in the original network architecture of Bonito were replaced with bottleneck convolution modules from ResNet50 [15], which is better supported by the inference engine library. Then, the search space of each block was defined as follows.

1. The number of modules of each block is searched from 1 to 9.
2. The number of channels of each block is searched from the list [32, 64, 128, 256, 378, 512].
3. The kernel size of the bottleneck convolution operator is searched from the list [3, 5, 7, 9, 11, 17, 29, 31, 47, 53, 69, 73, 83, 91, 107, 115, 123, 129].

New backbone search process

The search space defines the controller module in the NAS approach. A sample-eval-update loop [16] is used to train the controller. At each step, a batch of models are sampled from the controller. Each model is trained for several epochs in the trainer module, and then its inference latency and accuracy will be measured. After computing the multi-objective reward by accuracy and latency, the reward set as the input of the controller and its parameters are updated by maximizing the expected reward.

The best backbone from NAS search is named Fast-Bonito, whose overall architecture is shown in Fig. 1A. Compared with Bonito (Fig. 1B), the major difference is that Fast-Bonito includes a sequential of bottleneck convolution module layers to calculate the probability of nucleotides. A CTC decoder translates this probability into a nucleotide sequence. Considering that electronic signals are diverse in length, we also performed segments-split, which is similar that for Bonito. The input signals are cut into 6000 segments with an overlap of 300 before being fed into the convolutional architecture, with corresponding output segments at a length of 6000. Both ends of the segments were removed by a length of 150, except the first and last one. The back-end and leading-end of the first and last segments were both removed. The remaining parts of the segments were concatenated together and fed into the CTC decoder to produce the final nucleotide sequences.

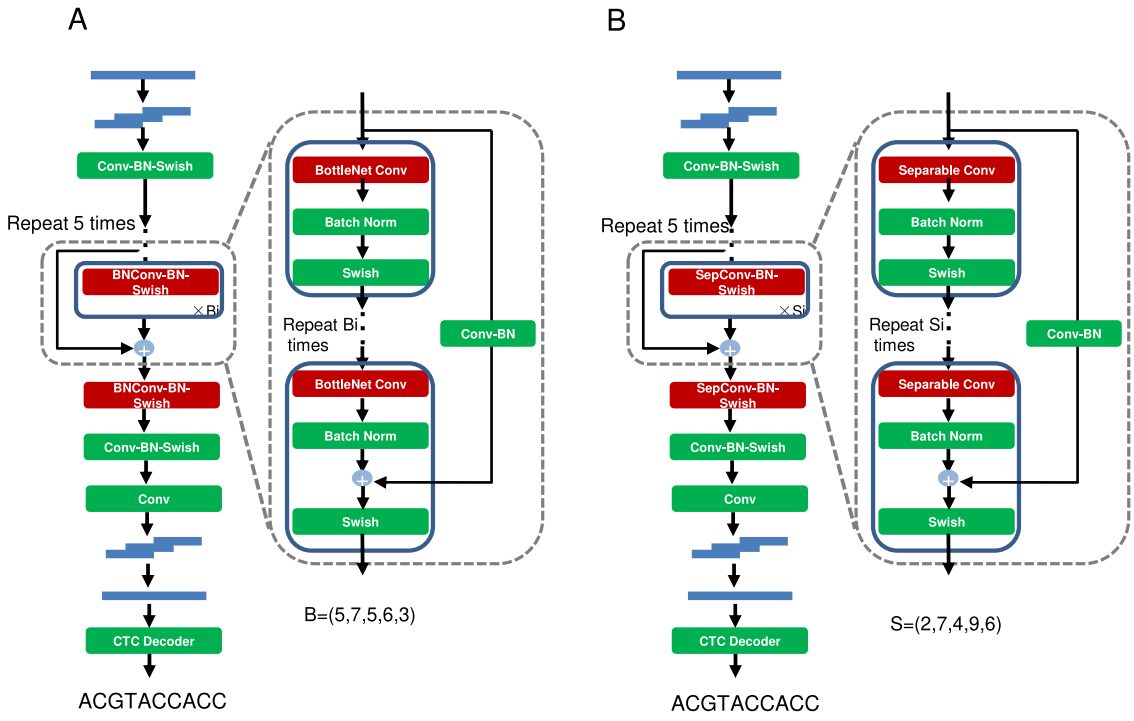

Fig. 1. A schematic comparison of the Fast-Bonito (A) and Bonito (B) architectures. Compared with Bonito, Fast-Bonito replaced the Separable Convolution module in Bonito with the Bottleneck Convolution module for acceleration. The architecture of Fast-Bonito comprises a convolutional architecture, a Convolution module layer in the last layer that calculates the probability of nucleotides, and a CTC decoder that translates the output of the network into a nucleotide sequence.

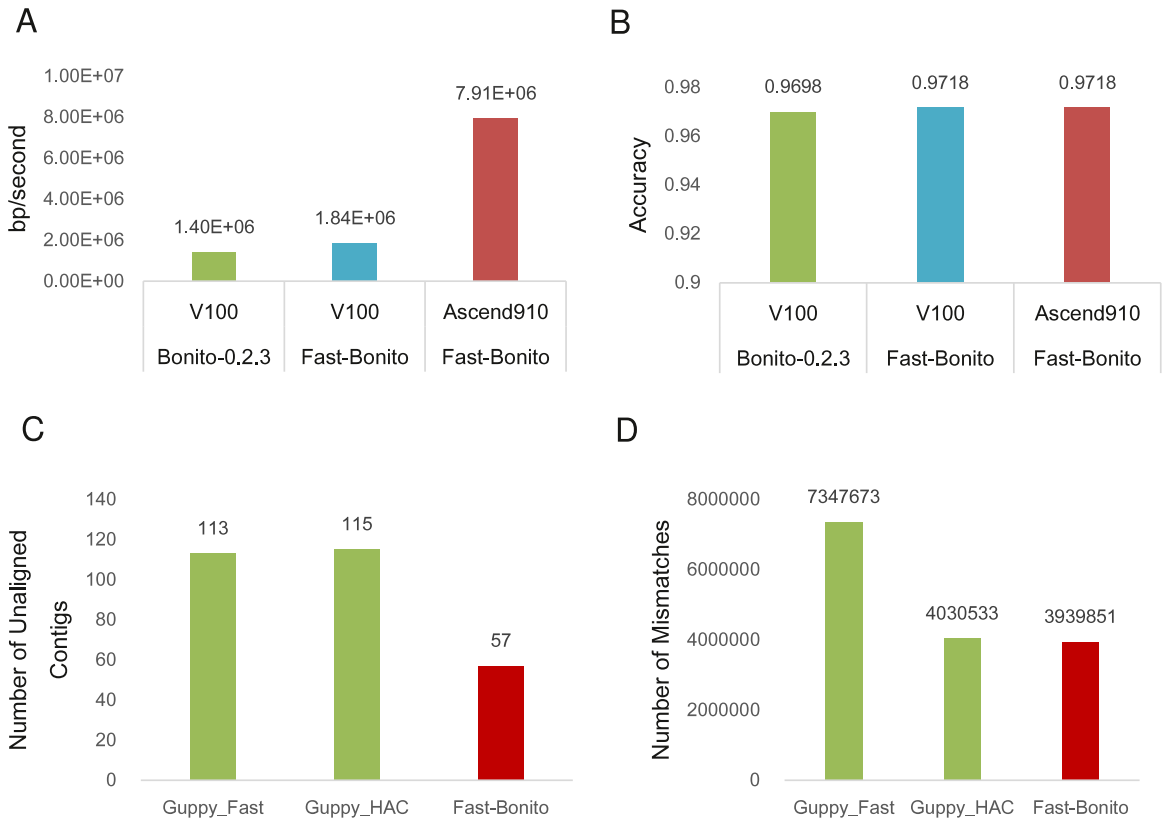

Fig. 2. Performance of Fast-Bonito. A) On NVIDIA V100, Fast-Bonito is 153.8% faster than the original version. While on HUAWEI Ascend 910 NPUs, Fast-Bonito is 565% times faster. **B)** The base calling accuracy of Fast-Bonito is slightly higher than the original Bonito (by 0.2%). Fast-Bonito has the least number of unaligned contigs **C)** and mismatches **D)**, when the assembly contigs are aligned with the reference genome, compared with "Guppy Fast" and "Guppy HAC".

Train the new backbone from scratch

After the best backbone of Fast-Bonito was obtained, we began to train it from scratch as described below.

1) Data Augmentation is a popular method that is used to improve model performance [17–19]. Commonly used augmentation approaches, such as cutout [20], rotate, flip [21], shearX [22], are not very suitable for 1-dimensional DNA sequence. We therefore used SpecAugment [19], a data augmentation method for speech recognition, which is a similar task to a DNA sequence decoder.
2) Label Smoothing [23,24] is a method to prevent the network from becoming over-confident and has been utilized in many state-of-the-art models. Here we used this method to prevent over-fitting.
3) Knowledge Distillation [25,26] is a method to migrate knowledge from a well-trained "teacher" model to a "student" model, thus to improve the performance of the "student" model. Here we took the pre-trained model of the original Bonito as "teacher" and our own model as "student".

Evaluation of the new backbone

Our primary benchmark dataset was obtained from the dataset provided by Bonito. This dataset contains 100,000 reads for evaluation. On NVIDIA V100 GPU, the speed of the original Bonito model was 1400,000 bp/s (Fig. 2A), and that of Fast-Bonito was 1840,000 bp/s — 153.8% faster than Bonito.

As neural processing units (NPU) are more efficient for deep learning tasks, we also performed further testing on HUAWEI Ascend 910 NPUs, which are powered by a scalable neural network computing architecture named DaVinci [27], where Fast-Bonito could reach 7910,000 bp/s in speed — 565% faster than Bonito on NVIDIA V100 GPU (Fig. 2A).

"bonito evaluate dna_r9.4.1 –chunks 100,000" was used to evaluate the read accuracy of base calling. Compared with Bonito, Fast-Bonito also achieved a slightly higher median in accuracy, which was 97.18% on both NVIDIA V100 and HUAWEI Ascend 910 (Fig. 2B), compared to 96.98% for Bonito. To estimate the impact on downstream analysis from the improved base calling accuracy, we performed "Shasta[28]-Racon[29]" contigs assembly pipeline and estimated the assembly accuracy by Quast [30] based on a previously published dataset [28]. To compare the assembly accuracy, base-called FASTQ/FASTA files were first pre-assembled by Shasta [28]. Then the assembled contigs were fed to Racon [29] for further polishing. Quast [30] was used to estimate the assembled contigs accuracy between "Guppy HAC", "Guppy Fast", and Fast-Bonito, against the reference genome "GRCh38". Compared to Guppy ("Guppy HAC" and "Guppy Fast" modes), Fast-Bonito also had less unaligned contigs (Fig. 2C) and mismatches (Fig. 2D).

Conclusion

Base calling is one of the key steps in the nanopore sequencing analysis workflow. Fast and accurate base calling is still a challenging problem. Although Bonito is able to achieve state-of-the-art accuracy, the speed of Bonito limits its application in production systems. Base calling tools require huge computational resources, especially GPU resources. Our study demonstrates that NPUs can be used to accelerate base calling as well, with a speed 4.3 times faster than on NVIDIA V100 GPUs. This shows a promising potential for the use of NPUs in genomic research.

The Bonito version used in this study was 0.2.3. Fast-Bonito was also developed using the same training and validation datasets. Bonito is still an active project that keeps releasing new features. We will also continuously update Fast-Bonito with the new Bonito features in the future.

Data and software statement

The training and validation datasets we used for Fast-Bonito were downloaded from the Bonito project on GitHub (https://github.com/nanoporetech/bonito). Fast-Bonito is publicly available under https://github.com/EIHealth-Lab/fast-bonito. The datasets used to estimate assembly accuracy were downloaded from https://s3-us-west-2.amazonaws.com/human-pangenomics/index.html?prefix=HG002/nanopore/. Due to the huge amounts of data, only the "12_16_19_R941_GM24385_13.fast5.tar" subset was used for downstream base calling and assembly.

Test environment

The test environment for the NVIDIA Tesla V100 used 16 GB GPU plus 8 CPUs, and the HUAWEI Ascend 910 environment used 16 GB*8 NPUs with 192 CPUs, but only 1 NPU and 24 CPUs were specified for the current task.

Author contributions

N.Q. and Y.L. designed and conceived the project. Y.M. performed the optimization of Bonito and model training. Z.X. and D.L. implemented the Fast-Bonito package and performed data experiment with the help from W.H, X.L., C.X., L.Z. and N.Q. D.L. wrote the manuscript. X.M., J.M., W.A.Z. and A.K. provided insightful suggestions and revised the manuscript. We also thank Dr. Min Sung Park for the discussion of using NPU to accelerate genomic research at very early stage. All authors read and approved the final manuscript.

Declaration of Competing Interest

The authors declare no competing interests.

References

[1] Sanger F, Coulson AR. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. J Mol Biol 1975;94:441–8.
[2] Behjati S, Tarpey PS. What is next generation sequencing? Arch Dis Child - Educ Pract Ed 2013;98:236–8.
[3] Lee H, et al. Third-generation sequencing and the future of genomics 2016. doi:10.1101/048603.
[4] Mikheyev AS, Tin MMY. A first look at the Oxford Nanopore MinION sequencer. Mol Ecol Resour 2014;14:1097–102.
[5] Ho SS, Urban AE, Mills RE. Structural variation in the sequencing era. Nat Rev Genet 2020;21:171–89.
[6] Wick RR, Judd LM, Holt KE. Performance of neural network basecalling tools for Oxford Nanopore sequencing. Genome Biol 2019;20:129.
[7] David M, Dursi LJ, Yao D, Boutros PC, Simpson JT. Nanocall: an open source basecaller for Oxford Nanopore sequencing data. Bioinformatics 2017;33:49–55.
[8] Boža V, Brejová B, Vinař TDeepNano. Deep recurrent neural networks for base calling in MinION nanopore reads. PLoS One 2017;12:e0178751.
[9] Teng H, et al. Chiron: translating nanopore raw signal directly into nucleotide sequence using deep learning. GigaScience 2018;7:giy037.
[10] Zeng J, et al. Causalcall: Nanopore Basecalling Using a Temporal Convolutional Network. Front Genet 2020;10:1332.
[11] Huang N, Nie F, Ni P, Luo F, Wang J. An attention-based neural network basecaller for Oxford Nanopore sequencing data. In: 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE; 2019. p. 390–4. doi:10.1109/BIBM47256.2019.8983231.
[12] Silvestre-Ryan J, Holmes I. Pair consensus decoding improves accuracy of neural network basecallers for nanopore sequencing 2020. doi:10.1101/2020.02.25.956771.
[13] Kriman S, et al. QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. ArXiv191010261 Eess 2019.
[14] Tan M, et al. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2019. p. 2815–23. doi:10.1109/CVPR.2019.00293.
[15] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. ArXiv151203385 Cs 2015.
[16] Zoph B, Le QV. Neural Architecture Search with Reinforcement Learning. ArXiv161101578 Cs 2017.
[17] Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QVAutoAugment. Learning Augmentation Strategies From Data. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2019. p. 113–23. doi:10.1109/CVPR.2019.00020.
[18] Zoph B, et al. Learning Data Augmentation Strategies for Object Detection. ArXiv190611172 Cs 2019.
[19] Park DS, et al. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. Interspeech 2019:2613–17 (ISCA, 2019). doi:10.21437/Interspeech.2019-2680.
[20] DeVries T, Taylor GW. Improved Regularization of Convolutional Neural Networks with Cutout. ArXiv170804552 Cs 2017.
[21] Shorten C, Khoshgoftaar TM. A survey on Image Data Augmentation for Deep Learning. J Big Data 2019;6:60.
[22] Hu B, Lei C, Wang D, Zhang S, Chen Z. A Preliminary Study on Data Augmentation of Deep Learning for Image Classification. ArXiv190611887 Cs Eess 2019.
[23] Kim S, Seltzer ML, Li J, Zhao R. Improved training for online end-to-end speech recognition systems. ArXiv171102212 Cs 2018.
[24] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2016. p. 2818–26. doi:10.1109/CVPR.2016.308.
[25] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network. ArXiv150302531 Cs Stat 2015.
[26] Wei L, et al. Circumventing Outliers of AutoAugment with Knowledge Distillation. ArXiv200311342 Cs 2020.
[27] Liao H, et al. Ascend: a Scalable and Unified Architecture for Ubiquitous Deep Neural Network Computing 2021 IEEE International Symposium on High-Performance Computer Architecture; 2021.
[28] Shafin K, et al. Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes. Nat Biotechnol 2020;38:1044–53.
[29] Vaser R, Sović I, Nagarajan N, Šikić M. Fast and accurate de novo genome assembly from long uncorrected reads. Genome Res 2017;27:737–46.
[30] Gurevich A, Saveliev V, Vyahhi N, Tesler G. QUAST: quality assessment tool for genome assemblies. Bioinformatics 2013;29:1072–5.