Long-read sequencing has the potential to transform variant detection by reaching currently difficult-to-map regions and routinely linking together adjacent variations to enable read-based phasing. Third-generation nanopore sequence data have demonstrated a long read length, but current interpretation methods for their novel pore-based signal have unique error profiles, making accurate analysis challenging. Here, we introduce a haplotype-aware variant calling pipeline, PEPPER-Margin-DeepVariant, that produces state-of-the-art variant calling results with nanopore data. We show that our nanopore-based method outperforms the short-read-based single-nucleotide-variant identification method at the whole-genome scale and produces high-quality single-nucleotide variants in segmental duplications and low-mappability regions where short-read-based genotyping fails. We show that our pipeline can provide highly contiguous phase blocks across the genome with nanopore reads, contiguously spanning between 85% and 92% of annotated genes across six samples. We also extend PEPPER-Margin-DeepVariant to PacBio HiFi data, providing an efficient solution with superior performance over the current WhatsHap-DeepVariant standard. Finally, we demonstrate de novo assembly polishing methods that use nanopore and PacBio HiFi reads to produce diploid assemblies with high accuracy (Q35+ nanopore-polished and Q40+ PacBio HiFi-polished).

Most existing reference-based small-variant genotyping methods are tuned to work with short-reads.\(^1\) Short-reads have high base-level accuracy but frequently fail to align unambiguously in repetitive regions.\(^1\) Short-reads are also generally unable to provide substantial read-based phasing information, and therefore require using haplotype panels for phasing that provide limited phasing information for rarer variants.

Third-generation sequencing technologies, like linked-reads\(^5\)\(^-\)\(^7\) and long-reads\(^8\)\(^-\)\(^11\), produce sequences that can map more confidently in the repetitive regions of the genome\(^1\), overcoming the fundamental limitations of short-reads. Long-reads can generate highly contiguous de novo assemblies\(^1\)\(^-\)\(^17\), and they are increasingly being used by reference-based analysis methods\(^16\)\(^-\)\(^22\). The Genome In A Bottle Consortium (GIAB)\(^26\) used the additional power of long-reads and linked-reads to expand the small-variant benchmarking set to cover more of the genome\(^27\). This was essential to the PrecisionFDA challenge V2, which quantified the limitations of short-read-based methods to accurately identify small variants in repetitive regions\(^28\).

Oxford Nanopore Technologies (ONT) is a commercial, nanopore-based, high-throughput\(^1\)\(^1\), long-read sequencing platform that can generate 100-kb+ long-reads\(^1\)\(^3\)\(^,\)\(^26\). Nanopore long-reads can confidently map to repetitive regions of the genome\(^1\), including centromeric satellites,acrocentric short arms, and segmental duplications\(^11\)\(^-\)\(^3\)\(^,\)\(^3\)\(^0\)\(^-\)\(^3\)\(^2\). The nanopore sequencing platform promises same-day sequencing and analysis\(^1\),\(^3\)\(^4\), but the base-level error characteristics of the nanopore-reads, being both generally higher and systematic, make small-variant identification challenging\(^1\)\(^4\).

Pacific Biosciences (PacBio) provides a single-molecule real-time (SMRT) sequencing platform that employs circular consensus sequencing to generate highly accurate (99.8%) high-fidelity (PacBio HiFi) reads that are between 15 and 20 kb long\(^1\). The overall accuracy of PacBio HiFi-based variant identification is competitive with short-read-based methods\(^1\). These highly accurate long-reads enabled the small-variant benchmarking of major histocompatibility complex (MHC) region\(^3\)\(^5\) and difficult-to-map regions\(^3\)\(^2\).

In our previous work, we introduced DeepVariant, a universal small-variant calling method based on a deep convolutional neural network (CNN)\(^3\)\(^6\). We showed that by retraining the neural network of DeepVariant, we can generate highly accurate variant calls for various sequencing platforms\(^3\)\(^6\). To limit the computational time, DeepVariant uses only the neural network on candidate sites identified with simple heuristics. However, the higher error rate of nanopore-reads\(^1\)\(^1\)\(^,\)\(^3\)\(^4\) causes too many candidate variants to be picked up by the heuristic-based candidate finder of DeepVariant, limiting the extension of DeepVariant to nanopore-sequencing platform.

Phasing long-reads has been shown to enable or improve methods for small-variant calling, structural-variant calling, and genome assembly\(^1\)\(^2\)\(^1\)\(^3\)\(^,\)\(^2\)\(^3\)\(^,\)\(^2\)\(^4\)\(^-\)\(^4\)\(^1\). Previously, we trained DeepVariant on PacBio HiFi long-read data, and it showed highly competitive performance against short-read-based methods for small-variant identification\(^2\). However, the run time of the haplotype-aware mode of DeepVariant with PacBio HiFi reads remain a bottleneck for production-level scaling.

Here, we present a haplotype-aware genotyping pipeline PEPPER-Margin-DeepVariant that produces state-of-the-art small-variant identification results with nanopore and PacBio HiFi long-reads. PEPPER-Margin-DeepVariant outperforms other existing nanopore-based variant callers like Medaka\(^1\), Clair\(^1\), and

---

1. UC Santa Cruz Genomics Institute, Santa Cruz, CA, USA. 2. Google Inc, Mountain View, CA, USA. 3. Chan Zuckerberg Initiative, Redwood City, CA, USA.
4. These authors contributed equally: Kishwar Shafin, Trevor Pesout, Pi-Chuan Chang. **e-mail:** awcarroll@google.com; bpaten@ucsc.edu
In contrast, PEPPER applies the predictions of the RNN to the haplotype-aware pipeline DeepVariant-WhatsHap-DeepVariant. We analyzed our pipeline in the context of GENCODE genes and report phasing errors in less than 1.5% of genes, with over 88% of all genes being contiguously phased across six samples. Finally, we extended PEPPER-Margin-DeepVariant to polish nanopore-based de novo assemblies with nanopore and PacBio HiFi reads in a diploid manner. We report Q35+ nanopore-based and Q40+ PacBio HiFi-polished assemblies with lower switch error rate compared to the unpolished assemblies.

Results
Haplotype-aware variant calling. PEPPER-Margin-DeepVariant is a haplotype-aware pipeline for identifying small variants against a reference genome with long-reads. The pipeline employs several methods to generate highly accurate variant calls (Fig. 1a). Details of these methods are in the online methods section. An overview is presented here:

1. PEPPER-SNP: PEPPER-SNP finds single-nucleotide polymorphisms (SNPs) from the read alignments to the reference using a recurrent neural network (RNN).

2. Margin: Margin is a phasing and haplotyping method that takes the SNPs reported by PEPPER-SNP and generates a haplotagged alignment file using a hidden Markov Model (HMM).

3. PEPPER-HP: PEPPER-HP takes the haplotted alignment file and finds potential SNP and insertion and deletion (INDEL) candidate variants using an RNN.

4. DeepVariant: DeepVariant takes candidates from PEPPER-HP and an haplotagged alignment file from margin and genotypes the proposed candidates in a haplotype-aware manner. DeepVariant has more total parameters than does PEPPER, models with more parameters, and can generally train to higher accuracy at the cost of increased runtime. By combining PEPPER with DeepVariant in this way, we allow the faster neural network of PEPPER to efficiently scan much more of the genome, and to leverage the larger neural network of DeepVariant to achieve high accuracy on a tractable number of candidates.

5. Margin: Margin takes the output of DeepVariant and the alignment file to generate a phased VCF file using the same HMM.

It is challenging to identify accurate variants with Oxford nanopore reads owing to the error rate. Heuristics-based approaches show robust solutions for highly accurate sequencing platforms but fail when introduced with erroneous reads. For example, in 90x HG003 ONT sequencing reads, at 10% allele frequency, we find 20 times more erroneous variants than true variants (Supplementary Table 4). Existing variant callers, like Clair, that use allele frequency to find a set of candidates often need to set the threshold too high, excluding many true variants from being detected. Our pipeline demonstrates an efficient solution using an RNN to find candidates with PEPPER and to accurately genotype the candidates with DeepVariant.

The use of haplotype information to get better genotyping results with erroneous reads has been demonstrated before. The schema of the PEPPER-Margin-DeepVariant pipeline follows a design similar to that of PacBio HiFi-based DeepVariant and Medaka, which use haplotyping to provide better genotyping results. However, Medaka is a consensus caller that presents as variants the predicted sequence per position that does not match the reference sequence. In contrast, PEPPER applies the predictions of the RNN to the candidates to find likely candidate variants for DeepVariant to accurately genotype. While maintaining similar candidate sensitivity of the heuristic-based approach, PEPPER reduces the number of erroneous homozygous candidate variants substantially (Supplementary Fig. 2).

Nanopore variant-calling performance. We compared the nanopore variant-calling performance of PEPPER-Margin-DeepVariant against that of Medaka, Clair, and Longshot. We called variants on two samples, HG003 and HG004, with 90x coverage. We also compared PEPPER-Margin-DeepVariant’s performance against that of Medaka and Clair for the HG003 sample at various coverages, ranging from 20x to 90x. Finally, we benchmarked the variant-calling performance of PEPPER-Margin-DeepVariant on six GIAB samples.

PEPPER-Margin-DeepVariant produces more accurate nanopore-based SNP calls (F1-scores of 0.9969 and 0.9977) for HG003 and HG004, respectively, than does Medaka (0.9926, 0.9933), Clair (0.9861, 0.9860), and Longshot (0.9775, 0.9776). We also observe better INDEL performance with PEPPER-Margin-DeepVariant (F1-scores of 0.7257 and 0.7128 for HG003 and HG004) than that of Medaka (0.7089, 0.7128) and Clair (0.5352, 0.5260) (Fig. 1b and Supplementary Table 1).

To assess the robustness of our method, we evaluated the variant-calling performance with the HG005 sample on GRCh38 and GRCh37 against two GIAB truth versions (v3.3.2 and v4.2.1). In this comparison, we see that PEPPER-Margin-DeepVariant performs similarly between GRCh38 (F1-score: 0.9971, INDEL F1-score: 0.7629) and GRCh38 v4.2.1 (SNP F1-score: 0.9974, INDEL F1-score: 0.7678) and has higher accuracy than does Medaka (GRCh37: SNP F1-score: 0.9938, INDEL F1-score: 0.7629, GRCh38: SNP F1-score: 0.9927, INDEL F1-score: 0.7406), Clair (GRCh37: SNP F1-score: 0.9789, INDEL F1-score: 0.5666, GRCh38: SNP F1-score: 0.9787, INDEL F1-score: 0.5675), and Longshot (GRCh37: SNP F1-score: 0.9803, GRCh38: SNP F1-score: 0.9767) (Supplementary Table 2). Overall, from the F1-scores, we see PEPPER-Margin-DeepVariant has consistent performance between different samples, reference sequence and truth sets.

We performed a Mendelian concordance analysis of our method with the HG005, HG006, and HG007 trio on GRCh38 inside and outside of the HG005 v4.2.1 high-confidence regions (Supplementary Table 3). In the 2.5-Gb high-confidence region, we observed a paternal and maternal concordance of 99.90%, with overall concordance of 99.75%. In the 315-Mb region outside of high confidence, excluding centromeres, we observed a paternal concordance of 98.20%, maternal concordance of 97.80%, and overall concordance of 95.52%.

To understand performance over realistic coverage ranges, we downsampled the HG003 nanopore sample at coverages varying between 20x and 90x and compared PEPPER-Margin-DeepVariant against Medaka and Clair. The INDEL performance of PEPPER-Margin-DeepVariant achieves the highest F1 score at any coverage compared with those of other tools (Fig. 1c and Supplementary Table 4). At coverage above 30x, PEPPER-Margin-DeepVariant achieves a higher F1 score than does Medaka and Clair (Supplementary Table 5). Overall, we observe that PEPPER-Margin-DeepVariant can yield high-quality variant calls at above 40x coverage on Oxford Nanopore data.

We investigated the nanopore variant-calling performance of PEPPER-Margin-DeepVariant on six GIAB samples (HG001, HG003–HG007), each with various coverage (Supplementary Table 6), and against GRCh37 and GRCh38 reference genomes (Supplementary Table 7). PEPPER-Margin-DeepVariant achieves an SNP F1-score of 0.995 or higher and INDEL F1-score of 0.709 or higher for each sample, demonstrating the ability to generalize variant calling across samples and reference genomes (Fig. 1d and Supplementary Table 8).
We also assessed PEPPER-HP’s use as a variant caller after tuning the method for a balanced precision and recall. We find that PEPPER-HP outperforms Medaka in SNP accuracy while having a comparable INDEL accuracy. However, PEPPER-HP in itself is not able to achieve the genotyping accuracy that DeepVariant provides. As PEPPER-HP uses a compressed representation of nucleotide
bases, it fails to achieve high genotyping accuracy compared with that of DeepVariant's CNN (Supplementary Table 9).

Similar to the nanopore-based haplotype-aware pipeline, the PacBio HiFi-based PEPPER-Margin-DeepVariant pipeline produces highly accurate variant calls. We analyzed PacBio HiFi PEPPER-Margin-DeepVariant variant-calling performance on the 35× HG003 and HG004 data from PrecisionFDA28 against DeepVariant-WhatsHap-DeepVariant (the current state-of-the-art method1,29) and DeepVariant-Margin-DeepVariant. In this comparison we see that DeepVariant-Margin-DeepVariant produces the best performance (HG003 SNP \( F_1 \): 0.9991, INDEL \( F_1 \): 0.9945; HG004 SNP \( F_1 \): 0.9992, INDEL \( F_1 \): 0.9942) compared with DeepVariant-WhatsHap-DeepVariant (HG003 SNP \( F_1 \): 0.9990, INDEL \( F_1 \): 0.9942; HG004 SNP \( F_1 \): 0.9992, INDEL \( F_1 \): 0.9940) and PEPPER-Margin-DeepVariant (HG003 SNP \( F_1 \): 0.9990, INDEL \( F_1 \): 0.9944; HG004 SNP \( F_1 \): 0.9992, INDEL \( F_1 \): 0.9941) (Supplementary Table 10).

We compared the runtime and cost of Oxford nanopore-based variant-calling pipelines on 50× and 75× HG001 data (Supplementary Table 11) using the GCP platform with instance sizes best matching CPU and memory requirements. On a CPU platform, PEPPER-Margin-DeepVariant is eight times faster than Medaka and four times faster than Longshot while providing the best variant-calling performance at a lower cost. Clair is faster and cheaper than PEPPER-Margin-DeepVariant, but fails to generate high-quality variant calls. On PacBio HiFi data, the PEPPER-Margin-DeepVariant pipeline outperforms DeepVariant-WhatsHap-DeepVariant and is three times faster and 1.4 times cheaper, establishing a faster and more accurate solution to haplotype-aware variant calling with PacBio HiFi data (Supplementary Table 12 and Supplementary Table 13).

Nanopore, Illumina, and PacBio HiFi variant calling. We compared the variant-calling performance of Oxford Nanopore and PacBio HiFi long-read-based PEPPER-Margin-DeepVariant against Illumina short-read-based DeepVariant method. We used 35× Illumina NovaSeq, 35× PacBio HiFi, and 90× Oxford Nanopore reads base-called with Guppy v4.2.2 for HG003 and HG004 samples available from PrecisionFDA1,28. We used GIAB v4.2.1 benchmarking data for HG003 and HG004, which is notable for including difficult-to-map regions. Finally, we used GIAB v2.0 stratifications to compare variant-calling performance in difficult-to-map regions and low-complexity regions of the genome.

The SNP \( F_1 \) score of PacBio HiFi (HG003 SNP \( F_1 \): 0.9990, HG004 SNP \( F_1 \): 0.9992) is higher than that of Oxford Nanopore (HG003 SNP \( F_1 \): 0.9969, HG004 SNP \( F_1 \): 0.9977) and Illumina (HG003 SNP \( F_1 \): 0.9963, HG004 SNP \( F_1 \): 0.9962) in all benchmarking regions. Notably, both long-read sequencing platforms outperform the short-read-based method in accurate SNP identification performance. The INDEL \( F_1 \) score of Oxford Nanopore (HG003 INDEL \( F_1 \): 0.7257, HG004 INDEL \( F_1 \): 0.7128) is well below the performance with PacBio HiFi (HG003 INDEL \( F_1 \): 0.9945, HG004 INDEL \( F_1 \): 0.9941) and Illumina (HG003 INDEL \( F_1 \): 0.9959, HG004 INDEL \( F_1 \): 0.9958), suggesting further improvement is required for nanopore-based methods. On HG003 PacBio-CLR data, we observed an SNP \( F_1 \) score of 0.9892 with our method and of 0.9755 with Longshot (Supplementary Table 14). Overall, we find that haplotype-aware long-read-based variant calling produces high-quality SNP variant calls comparable to those produced by short-read-based variant-identification methods (Fig. 2a and Supplementary Table 15). This is the first demonstration that we are aware of in which SNP variant calls with Oxford Nanopore data achieved similar accuracy to Illumina SNP variant calls.

In segmental duplications, 250-bp+ nonunique regions and low-mappability regions, where short-reads have difficulty in mapping, we observe the average SNP \( F_1 \) scores of Illumina (segmental duplication \( F_1 \) score: 0.94, 250 bp+ nonunique: 0.66, low mappability: 0.94) drop sharply for both HG003 and HG004 samples. Long-read-based PacBio HiFi (segmental duplication \( F_1 \) score: 0.99, 250 bp+ nonunique: 0.90, low mappability: 0.99) and Oxford Nanopore (segmental duplication \( F_1 \) score: 0.98, 250 bp+ nonunique: 0.94, low mappability: 0.98) produces more accurate SNP variants. In the major histocompatibility complex region, we see Oxford Nanopore (HG003 SNP \( F_1 \) score: 0.9958, HG004 SNP \( F_1 \) score: 0.9966) achieve best performance, followed by PacBio HiFi (HG003 SNP \( F_1 \) score: 0.9951, HG004 SNP \( F_1 \) score: 0.9955) and Illumina (HG003 SNP \( F_1 \) score: 0.9939, HG004 SNP \( F_1 \) score: 0.9921). In general, the long-read-based haplotype-aware methods outperform short-reads in more repetitive regions of the genome (Fig. 2b and Supplementary Table 16).

In low-complexity regions like homopolymer, dimer and trimer repeat regions of the genome, the average variant-calling performance of Nanopore drops (7–11 bp homopolymer SNP \( F_1 \) score: 0.96, 11 bp+ homopolymer SNP \( F_1 \) score: 0.88), for both HG003 and HG004 samples, compared with that of Illumina (7–11 bp homopolymer SNP \( F_1 \) score: 0.998, 11 bp+ homopolymer SNP \( F_1 \) score: 0.998) and PacBio HiFi (7–11 bp homopolymer SNP \( F_1 \) score: 0.998, 11 bp+ homopolymer SNP \( F_1 \) score: 0.984). In 11- to 50-bp dimer and 15- to 50-bp trimer repeat regions of the genome, we see that the average performance of Oxford Nanopore (dimer SNP \( F_1 \) score: 0.969, trimer SNP \( F_1 \) score: 0.984) is lower than that of PacBio HiFi (dimer SNP \( F_1 \) score: 0.995, trimer SNP \( F_1 \) score: 0.995) and Illumina (dimer SNP \( F_1 \) score: 0.998, trimer SNP \( F_1 \) score: 0.998). Overall, the Illumina short-read-based variant-calling method achieves higher accuracy in low-complexity regions of the genome (Fig. 2c and Supplementary Table 17).

We further compare the variant-calling performance (\( F_1 \) score) of Illumina, PacBio HiFi, and ONT in ‘easy regions,’ which cover 76% of the genome1. In this comparison, we see ONT variant-calling performance (SNP \( F_1 \): 0.9988, INDEL \( F_1 \): 0.9719) is comparable to that of Illumina (SNP \( F_1 \): 0.9997, INDEL \( F_1 \): 0.9996) and HiFi (SNP \( F_1 \): 0.9999, INDEL \( F_1 \): 0.9997), showing that in easy regions, all technologies can generate high-quality variants (Supplementary Table 18). We further look into regions with no tandem repeats (covering 86% of the genome) and see that ONT performance (SNP \( F_1 \): 0.9981, INDEL \( F_1 \): 0.97) is comparable to that of Illumina (SNP \( F_1 \): 0.996, INDEL \( F_1 \): 0.986). However, in tandem repeat and homopolymer regions, the ONT SNP calling \( F_1 \) score drops from 0.998 to 0.9748, and the INDEL calling performance drops from 0.97 to 0.54 (Supplementary Table 19) suggesting that ONT variant calling can generate competitive variant calling in the 86% of the genome outside tandem repeat and homopolymer regions, and it suffers in only the 4% of the genome that is highly repetitive.

Phased haplotype accuracy. We compared phased accuracy for Margin and WhatsHap against GIAB’s phased v3.3.2 variants with 25× nanopore, 50× nanopore, 75× nanopore, and 35× PacBio HiFi data. We generated genotyped variants with PEPPER-Margin-DeepVariant, and used both Margin and WhatsHap to phase the final variant set. The phased sets produced by Margin and WhatsHap were analyzed using whatstat stats and whatstat compare commands against the trio-confirmed truth variants in high-confidence regions.

For all datasets, Margin had a lower switch error rate (0.00875, 0.00857, 0.00816, 0.00895) than did WhatsHap (0.00923, 0.00909, 0.00906, 0.00930), but lower phased N50 (2.07, 6.13, 0.24 Mb; N50 is a weighted median, signifying the length of the sequence in a set for which all sequences of that length or greater sum to 50% of the set’s total size) than did WhatsHap (2.37, 4.90, 8.27, 0.25 Mb) (Fig. 3a and Supplementary Tables 20 and 21). We also compared phased accuracy for Margin and WhatsHap on the same data using a new metric we call local phasing correctness (LPC), which is a generalization of the standard switch error and Hamming rate.
We plot LPC across various length-scale values (Fig. 3b). Margin produced more accurate phasing for all length scales for 25x nanopore and 35x circular consensus sequencin. Margin also produced more accurate phasing for 50x nanopore for length scales up to 128 kb and for 75x nanopore for length scales up to 242 kb.

To analyze haplotagging accuracy, we artificially constructed an admixture sample by trio-binning reads from HG005 and HG004.
HGO02723 and combining an equal amount of maternal reads from each sample, resulting in a 55× nanopore alignment and a 35× PacBio HiFi alignment. We ran PEPPER-SNP, haplotagged each alignment with Margin using these variants, and compared the number of direct-matched reads $R_1$ (truth H1 to tagged H1 or truth H2 to tagged H2) and crossmatched reads $R_2$ (truth H1 to tagged H2 or truth H2 to tagged H1) of the output. In Fig. 3c, for each 10-kb bucket in chr1, we plot the number of reads that were direct-matched (top, red) and crossmatched (bottom, blue) for both data types, with phasesets plotted in black alternating between top and bottom. This natural switch plot identifies consistent phasing as regions where the majority of reads are either direct- or cross-matched, and haplotagging switch errors as regions where the majority of reads transition between the two. As the plot shows, nanopore reads allow us to haplotag consistently with phase sets in the range of tens of megabases, whereas PacBio HiFi reads cannot be used for long-range haplotagging. For each bucket, we can calculate a local haplotagging accuracy using the ratio: \[ \text{max}(R_1, R_2) / (R_1 + R_2). \]

On average the haplotagging accuracy is 0.9626 for ONT data and 0.9800 for HiFi data using Margin with reduced performance for WhatsHap (Supplementary Table 22 and Supplementary Figs. 3–6). As Margin has higher haplotagging accuracy than does WhatsHap, we see that the variant calling with Margin exhibits higher accuracy than does WhatsHap for both Oxford Nanopore and PacBio HiFi data (Supplementary Table 23 and Supplementary Table 12).

Lastly, we compare the runtime and cost for the haplotag and phase actions on the four HG001 datasets using Margin and WhatsHap. For Margin, we calculate a cost of $1.35 (36 m) for 25× ONT, $3.17 (84 m) for 50× ONT, $4.64 (123 m) for 75× ONT, $1.23 (33 m) for 35× PacBio HiFi, and for WhatsHap $1.48 (941 m), $2.10 (1336 m), $2.66 (1688 m), and $1.20 (764 m), respectively (Supplementary Table 24).

Gene analysis. We performed an analysis of Margin’s phasing over genic regions to understand its utility for functional studies. With 75× nanopore data from HG001 on GRCh37, we classified each of the GENCODE v35 genes\(^1\) (coding and noncoding) as wholly, partially, or not spanning the GIAB v3.3.2 high-confidence regions, the number of gene bodies as spanned by high-confidence regions as regions where the majority of reads transition between the two. As the plot shows, nanopore reads allow us to haplotag consistently with phase sets in the range of tens of megabases, whereas PacBio HiFi reads cannot be used for long-range haplotagging. For each bucket, we can calculate a local haplotagging accuracy using the ratio: \[ \text{max}(R_1, R_2) / (R_1 + R_2). \]

On average the haplotagging accuracy is 0.9626 for ONT data and 0.9800 for HiFi data using Margin with reduced performance for WhatsHap (Supplementary Table 22 and Supplementary Figs. 3–6). As Margin has higher haplotagging accuracy than does WhatsHap, we see that the variant calling with Margin exhibits higher accuracy than does WhatsHap for both Oxford Nanopore and PacBio HiFi data (Supplementary Table 23 and Supplementary Table 12).

Lastly, we compare the runtime and cost for the haplotag and phase actions on the four HG001 datasets using Margin and WhatsHap. For Margin, we calculate a cost of $1.35 (36 m) for 25× ONT, $3.17 (84 m) for 50× ONT, $4.64 (123 m) for 75× ONT, $1.23 (33 m) for 35× PacBio HiFi, and for WhatsHap $1.48 (941 m), $2.10 (1336 m), $2.66 (1688 m), and $1.20 (764 m), respectively (Supplementary Table 24).

Gene analysis. We performed an analysis of Margin’s phasing over genic regions to understand its utility for functional studies. With 75× nanopore data from HG001 on GRCh37, we classified each of the GENCODE v35 genes\(^1\) (coding and noncoding) as wholly, partially, or not spanning the GIAB v3.2 high-confidence regions, the phasesets proposed by Margin, and the switch errors between the two as determined by WhatsHap compare. In Fig. 4a, we first plot the number of gene bodies as spanned by high-confidence regions (23,712 wholly, 26,770 partly, 11,372 not), then further compare how many of each division were wholly spanned by Margin’s phasesets (23,712 wholly, 26,770 partly, 11,372 not), then further compare how many of these gene regions were perfectly captured (wholly phased with no switch errors and having no SNP or INDEL miscalls) for the entire gene, all annotated exons (1,738 for nanopore, 2,086 for PacBio HiFi), for all annotated exons (3,121, 2,446), and for all coding sequences (3,481, 2,471). For nanopore data, we find that for 91.8% of genes, the CDS is fully phased and genotyped without error, and for 82.3% of genes, all exons are fully phased and genotyped without error.

Diploid polishing of de novo assemblies. Oxford Nanopore-based assemblers like Flye\(^1\) and Shasta\(^1\) generate haploid assemblies of diploid genomes. By calling and phasing variants against the haploid contigs they produce, it is possible to polish the haploid assembly into a diploid assembly. We implemented such a diploid de novo assembly-polishing method with PEPPER-Margin-DeepVariant (Fig. 5a). It can polish haploid Oxford Nanopore-based assemblies with either Nanopore or PacBio HiFi reads. A detailed description of this method is presented in the Methods.

We generated haploid assemblies using Shasta\(^1\) and Flye\(^1\) for diploid samples HG005, HG00733, and HG02723, and haploid sample CHM13 (chrX) using nanopore reads, and we polished the Shasta assemblies using ONT and PacBio HiFi reads. To evaluate the base-level accuracy of the assemblies we use the k-mer-based tool YAK\(^5\), which uses Illumina trio data to estimate sequence quality, switch error rates, and Hammering error rates. We compare the haploid assemblies, polished diploid assemblies, and trio-aware diploid assemblies generated with hifiasm\(^1\). Hifiasm uses parental short-read data to generate maternal and paternal assemblies.

The estimated quality values (QV) of nanopore-based assemblies with Shasta assembler are higher than the nanopore-based Flye assemblies. (Supplementary Table 29). As Shasta generated higher-quality assemblies than did Flye, we polished the Shasta assemblies with the PEPPER-Margin-DeepVariant diploid polisher. The nanopore-polished assemblies achieve Q35+ estimated quality and PacBio HiFi-polished assemblies achieve Q40+ estimated quality for all three diploid samples. Finally, we show that the unpolished CHM13-chrX Shasta assembly (QV34.6) can be improved to QV36.9 with nanopore-based and QV42.7 PacBio HiFi-based assembly polishing with PEPPER-Margin-DeepVariant. Compared with the nanopore-based Shasta assemblies, the trio-aware PacBio HiFi assembler hifiasm achieves higher-quality assemblies with respect to base-level accuracy, but the NG50 of the hifiasm assemblies are lower for HG00733 and HG02723 samples. In summary, PEPPER-Margin-DeepVariant achieves Q35+ ONT-based assembly polishing and Q40+ PacBio HiFi-based assembly polishing of ONT assemblies (Fig. 5b and Supplementary Tables 29 and 30).

The dominant error modality for ONT data are homopolymers\(^1\). In Fig. 5c, we show the run-length confusion matrix of PacBio HiFi read alignments to four chrX assemblies of CHM13-chrX. The Shasta assembly starts to lose resolution at run-lengths
greater than 7 (RL-7) and loses all resolution around RL-25. The nanopore-polished assembly improves homopolymer resolution up to RL-10, but also fails to resolve run-lengths greater than RL-25. The PacBio HiFi polished assembly has fair resolution up to RL-25. The trio-hifiasm assembly shows accurate homopolymer resolution up to and beyond RL-50.
Figure 5d shows the switch error rate of the assemblies. The switch error rate and Hamming error rate of haploid Shasta assemblies are reduced after polishing with PEPPER-Margin-DeepVariant using ONT data. The trio-hifiasm assembly that uses maternal and paternal short-reads to resolve haplotypes has a much lower switch error rate and Hamming error rate (Fig. 5d and Supplementary Table 30).

In Table 1, we compare HG005 assemblies at the small-variant level. The analysis shows that the $F_1$ score of the unpolished Shasta...
Fig. 5 | Diploid assembly-polishing results. a, Illustration of the diploid assembly-polishing pipeline. b, Estimated quality values of assemblies using YAK. c, CHM13 chrX run-length confusion matrix between different assemblies and PacBio HiFi reads aligned to the corresponding assembly. d, Switch error and Hamming error comparison between assemblies.
assembly (INDEL: 0.1203, SNP: 0.4928) improves significantly after polishing with nanopore reads using PEPPER-Margin-DeepVariant (INDEL: 0.3611, SNP: 0.9825). The PacBio HiFi-polished Shasta assembly achieves a similar F₁ score (INDEL F₁: 0.9565, SNP F₁: 0.9976) than does the trio-hifiasm assembly (INDEL: 0.9733, SNP: 0.9988). This analysis provides evidence that PEPPER-Margin-DeepVariant can effectively improve the assembly quality at the small-variant level.

The current version of the PEPPER-Margin-DeepVariant pipeline does not attempt to polish structural variants (SVs, >50 bp in size). The resulting haplotypes preserve all SVs initially contained in the input assembly. Since the input assemblies are haploid, only one (randomly assembled) allele for each heterozygous SV is retained within the pair of output haplotypes. To benchmark SV recall and precision, we first called SVs from the assemblies using svm-asm and then validated the reconstructed SV sets using the previously described approach. Our benchmarks using HG002, HG005, HG0073, and HG02733 genomes show that input Shasta assemblies on average contained signatures of 94.6% and 48.3% of homozygous and heterozygous SVs, respectively. After polishing using PEPPER-Margin-DeepVariant, the average reconstruction rate slightly increased to 95.7% and 50.9% for homozygous and heterozygous SVs, respectively. The average SV precision was 81.6% before and 83.2% after polishing (Supplementary Table 31).

Discussion

Long-read sequencing technology is allowing gapless human genome assembly and enabling investigations in the most repetitive regions of the genome. In this work, we present PEPPER-Margin-DeepVariant, a state-of-the-art long-read variant-calling pipeline for Oxford nanopore data. For the first time, we show that nanopore-based SNP identification outperforms a state-of-the-art short-read-based method at whole-genome scale. Particularly in segmental duplication and difficult-to-map regions, the nanopore-based method outshines the short-read-based method. It seems likely that the anticipated widespread application of long-read variant calling will for the first time accurately illuminate variation in these previously inaccessible regions of the genome.

The genomic contexts where nanopore SNP accuracy suffers for our pipeline are identifiable, meaning that variant calls in these regions can be treated with skepticism while calls outside these contexts can be handled with confidence. The one obvious area that Nanopore variant calling lags is in INDEL accuracy. While the results achieved here are to our knowledge the best shown so far, we believe it is likely that further technological innovations at the platform level will be required to make nanopore INDEL accuracy on par with that of other technologies in all genomic contexts. However, we find that in the 86% of the genome without tandem repeats or homopolymers, INDEL calls from our method are already of high quality.

PEPPER-Margin-DeepVariant is designed for whole-genome sequencing analysis. Although targeted sequencing with the Oxford Nanopore platform is reasonably popular, several issues may limit the application. For example, read length, read quality, coverage, and heterozygosity of the target region are expected to be different than whole-genome sequencing. Further investigation and benchmarking are required to extend support for variant calling on amplicon sequencing data.

Oxford Nanopore provides a highly multiplexed sequencing solution with its PromethION device. With this device and the PEPPER-Margin-DeepVariant pipeline described here, it should be comfortably possible to go from biosample collection to complete genome inferences in under half a day. This fast turnaround should enable its use in a medical context, where diagnosis for acute disease situations requires speed.

We have demonstrated that our nanopore-based phasing is able to wholly phase 85% of all genes with only 1.3% exhibiting a switch error. This phasing ability could play a useful role in population genetics studies and clinical genomics. For clinical applications, the accurate identification of compound-heterozygotes should be particularly valuable.

We have extended PEPPER-Margin-DeepVariant to PacBio HiFi reads and demonstrated a more accurate and cheaper solution to the existingWhatsHap-DipVariant variant calling methods, making cohort-wide variant calling and phasing with PacBio HiFi more accessible. Currently, we find PacBio HiFi sequencing analyzed with our method has the best performance, but we expect that improvements to nanopore core technology and basecalling may close this gap.

We have demonstrated diploid polishing of nanopore-based haploid assemblies with PEPPER-Margin-DeepVariant. We achieve Q35+ nanopore polished assemblies and Q40+ PacBio HiFi polished assemblies. We observe that our polishing method can resolve homopolymer errors up to 20 bp with PacBio HiFi data. However, our polishing method fails to resolve 25-bp+ long homopolymers, indicating that they need to be resolved during the consensus generation of the de novo assembly methods. As nanopore assembly methods like Shasta move toward generating fully resolved diploid genome assemblies like trio-hifiasm, our polishing method can enable nanopore-only Q40+ polished diploid assemblies.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of
data and code availability are available at https://doi.org/10.1038/s41592-021-01299-w.

Received: 8 March 2021; Accepted: 6 September 2021; Published online: 1 November 2021

References

1. Altshuler, D. M. et al. An integrated map of genetic variation from 1,092 human genomes. Nature 491, 56–65 (2012).
2. McKenna, A. et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20, 1297–1303 (2010).
3. Li, W. & Freemdenberg, J. Mappability and read length. Front. Genet. 5, 381 (2014).
4. Chaisson, M. J. P. et al. Multi-platform discovery of haplotype-resolved structural variation in human genomes. Nat. Commun. 10, 1784 (2019).
5. Belton, J. M. et al. Hi-C: a comprehensive technique to capture the conformation of genomes. Methods 58, 268–276 (2012).
6. Falconer, E. & Lansdorp, P. M. Strand-seq: a unifying tool for studies of chromosome segregation. Semin. Cell Developmental Biol. 24, 643–652 (2013).
7. Weisenfeld, N. I., Kumar, V., Shah, P., Church, D. M. & Jaffe, D. B. Direct determination of diploid genome sequences. Genome Res. 27, 757–767 (2017).
8. Jain, M. et al. Improved data analysis for the MinION nanopore sequencer. Nat. Methods 12, 351 (2015).
9. Eid, J. et al. Real-time DNA sequencing from single polymerase molecules. Science 323, 133–138 (2009).
10. Jain, C., Rhie, A., Hansen, N., Koren, S. & Phillippy, A. M. A long read mapping method for highly repetitive reference sequences. Preprint at https://doi.org/10.1101/2020.11.01.363887 (2020).
11. Miga, K. H. et al. Telomere-to-telomere assembly of a complete human X chromosome. Nature 585, 79–84 (2020).
12. Logsdon, G. A. et al. The structure, function and evolution of a complete human chromosome 8. Nature 593, 7857 (2021).
13. Shafin, K. et al. Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes. Nat. Biotechnol. 38, 1044–1053 (2020).
14. Cheng, H., Concepcion, G. T., Feng, X., Zhang, H. & Li, H. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. Nat. Methods 18, 170–175 (2021).
15. Nurk, S. et al. HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads. Genome Res. 30, 1291–1305 (2020).
16. Kolmogorov, M., Yuan, J., Lin, Y. & Pevzner, P. A. Assembly of long, error-prone reads using repeat graphs. Nat. Biotechnol. 37, 540–546 (2019).
17. Ruan, J. & Li, H. Fast and accurate long-read assembly with wtdbg2. Nat. Methods 17, 155–158 (2020).
18. nanoporetech/medaka: sequence correction provided by ONT Research, https://github.com/nanoporetech/medaka (Oxford Nanopore Technologies, 2018).
19. Luo, R. et al. Exploring the limit of using a deep neural network on pileup data for germline variant calling. Nat. Mach. Intell. 2, 220–227 (2020).
20. Edge, P. & Bansal, V. Longhot enables accurate variant calling in diploid genomes from single-molecule long read sequencing. Nat. Commun. 10, 1–10 (2019).
21. Wenger, A. M. et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. Nat. Biotechnol. 37, 1155–1162 (2019).
22. Ehler, J., Haukness, M., Pesout, T., Marschall, T. & Paten, B. Haplotype-aware de novo assembly from noisy long reads. Genome Biol. 20, 116 (2019).
23. Huddleston, J. et al. Discovery and genotyping of structural variation from long-read haploid genome sequence data. Genome Res. 27, 677–685 (2017).
24. Sedlazeck, F. J. et al. Accurate detection of complex structural variations using single-molecule sequencing. Nat. Methods 15, 461–468 (2018).
25. Patterson, M. D. et al. WhatsHap: weighted haplotype assembly for future-generation sequencing reads. J. Comput. Biol. 22, 498–509 (2015).
26. Zook, J. M. et al. Extensive sequencing of seven human genomes to characterize benchmark reference materials. Sci. Data 3, 160025 (2016).
27. Wagner, J. et al. Benchmarking challenging small variants with linked and long reads. Preprint at https://doi.org/10.1101/2020.07.24.212712 (2020).
28. Olson, N. D. et al. precisionFDA Truth Challenge V2: calling variants from short-and-long-reads in difficult-to-map regions. Preprint at https://doi.org/10.1101/2020.11.13.380741 (2020).
29. Jain, M. et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. Nat. Biotechnol. 36, 338 (2018).
30. Jain, M. et al. Linear assembly of a human centromere on the Y chromosome. Nat. Biotechnol. 36, 321 (2018).
31. Fiddes, I. T. et al. Comparative Annotation Toolkit (CAT)—simultaneous clade and personal genome annotation. Genome Res. 28, 1029–1038 (2018).
32. Eichler, E. E., Clark, R. A. & She, X. An assessment of the sequence gaps: unfinished business in a finished human genome. Nat. Rev. Genet. 5, 345 (2004).
33. Euskirchen, P. et al. Same-day genomic and epigenomic diagnosis of brain tumors using real-time nanopore sequencing. Acta Neuropathol. 134, 691–703 (2017).
34. Rang, F. J., Kloosterman, W. P. & de Ridder, J. From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. Genome Biol. 19, 90 (2018).
35. Chin, C.-S. et al. A diploid assembly-based benchmark for variants in the major histocompatibility complex. Nat. Commun. 11, 1–9 (2020).
36. Poplin, R. et al. A universal SNP and small-indel variant caller using deep neural networks. Nat. Biotechnol. 36, 983 (2018).
37. Edge, P., Bafna, V. & Bansal, V. HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies. Genome Res. 27, 801–812 (2017).
38. Rodriguez, O. L. et al. A novel framework for characterizing genomic haplotype diversity in the human immunoglobulin heavy chain locus. Front. Immunol. 11, 2136 (2020).
39. Chin, C.-S. et al. Phased diploid genome assembly with single-molecule real-time sequencing. Nat. Methods 13, 1050 (2016).
40. Koren, S. et al. De novo assembly of haplotype-resolved genomes with trio haploid and diploid genome assemblies. Bioinformatics 36, 1174 (2018).
41. Porubsky, D. et al. Fully phased human genome assembly without parental data using single-cell strand sequencing and long reads. Nat. Biotechnol. 39, 302–308 (2021).
42. Harrow, J. et al. GENCODE: The reference human genome annotation for The ENCODE Project. Genome Res. 22, 1760–1774 (2012).
43. Zook, J. M. et al. An open resource for accurately benchmarking small variant and reference calls. Nat. Biotechnol. 37, 561 (2019).
44. Baid, G. et al. An extensive sequence dataset of gold-standard samples for benchmarking and development. Preprint at https://doi.org/10.1101/2020.12.11.420220 (2020).
45. Frankish, A. et al. GENCODE reference annotation for the human and mouse genomes. Nucleic Acids Res. 47, D766–D773 (2019).
46. Keller, D. & Vingron, M. SVIM-asn: Structural variant detection from haploid and diploid genome assemblies. Bioinformatics 36, 22–23 (2020).
47. Zook, J. M. et al. A robust benchmark for detection of germline large deletions and insertions. Nat. Biotechnol. 38, 1347–1355 (2020).
48. Tewhey, R., Bansal, V., Torkamani, A., Topol, E. J. & Schork, N. J. The importance of phase information for human genomics. Nat. Rev. Genet. 12, 215–223 (2011).
49. Browning, S. R. & Browning, B. L. Haplotype phasing: existing methods and new developments. Nat. Rev. Genet. 12, 703–714 (2011).
50. Glusman, G., Cox, H. C. & Rouch, J. C. Whole-genome haplotyping approaches and genomic medicine. Genome Med. 6, 1–16 (2014).
51. Li, H. et al. A synthetic-diploid benchmark for accurate variant-calling evaluation. Nat. Methods 15, 595–597 (2018).

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© The Author(s), under exclusive licence to Springer Nature America, Inc. 2021
Methods

Analysis methods and data pre-processing. Read alignment. We used minimap2 (ref. 53) version 2.17-r941 and pbmm2 version 1.4.0 to align reads to a reference genome. The Supplementary Notes have details on execution parameters.

Subsampling alignment files to different coverages. We used samtools53 version 1.10 to align coverage files of different coverages. The Supplementary Notes have details on execution parameters.

Variant calling. We used the following methods to call variants with nanopore data:

- PEPPER-Margin-DeepVariant version r0.4
- Medaka version v1.2.1
- Clair version v2.1.1
- Longshot version v0.4.2

For Illumina short-reads and PacBio HiFi we used DeepVariant version v1.1.0. The details on execution parameters are available in Supplementary Notes.

Benchmarking variant calls. We used hap.py54 version v0.3.12 to assess the variant calls against GIAB truth set. The hap.py program is available via jmdani/20/hap.py:v0.3.12 docker image. The command used for the assessment is described in the Supplementary Notes.

For HG002, HG003, HG004, and HG005 we used GIAB v4.2.1 truth set52 against GRCh38 reference and for HG001, HG005, HG006, and HG007 samples, we used v3.2 variant benchmarks52 against GRCh37 reference genome. We used GIAB stratification v2.0 files with hap.py to derive stratified variant-calling results. The GIAB benchmarking data availability is listed in the data availability section of the Supplementary Notes.

Mendelian analysis. For our Mendelian Analysis we used RTG version 3.12 (ref. 55). We used Margin version v2.0 and WhatsHap22 version v1.0 to haplotag and phase the variants.

Margin is available at https://github.com/UCSC-nanopore-cgl/margin and WhatsHap is available at https://github.com/whatsnap/whatsnap. The details on how we ran these tools is described in Supplementary Notes.

Small-variant switch error rate and Hamming error rate determination. We used a Workflow Description Language (WDL)-based analysis pipeline whaphap.wdl, available at https://github.com/tpeou/slideromics_scripts, to derive the switch error rate and Hamming error rate compared to the GIAB truth set. The whaphap workflow invokes the stats and compare submodules available in whaphap version v1.0.

In our analysis, we compared phased variants against GRCh37 reference against the GIAB v3.3.2 truth set to derive switch error rate and Hamming error rate. We considered only variants that have PATMAT annotation in the truth set and that fall in the high-confidence region defined by GIAB benchmarking set. The non-PATMAT annotated variants in the GIAB benchmarking variant set are not trio-confirmed, so we did not use those to benchmarking our phasing methods. We used WhatsHap compare command to generate the whole-genome switch error rate and used a custom script defined in whaphap.wdl to derive the Hamming error rates.

Local phasing correctness calculation. For the LPC analysis, we used the calcLocalPhasingCorrectness executable found in the https://github.com/UCSC-nanopore-cgl/margin repository. The LPC analysis requires a truth variant set and a query variant set. We used GIAB benchmarking set as truth. The calcLocalPhasingCorrectness generates a tov file describing the results.

We used https://github.com/tpeou/slideromics_scripts/plot_haplotagging_lpc.py script to visualize the results. The details of parameters are described in the Supplementary Notes. The methods used in local phasing correctness (in ‘Phasestat and haplotagging accuracy’) as a metric are presented separately in the methods description.

Haplotagging accuracy and natural switch detection. We used https://github.com/tpeou/slideromics_scripts/haplotagging_stats.py to calculate the haplotagging accuracy. The script calculates average haplotagging accuracy and average tagged reads per 10kb. Details on execution parameters are available in the Supplementary Notes.

We visualized the natural switch error using compare_read_phasing_hapBam.py available in https://github.com/tpeou/slideromics_scripts. Details on execution parameters are available in the Supplementary Notes.

Phaselineblock N50 calculation. An N50 value is a weighted median; it is the length of the sequence in a set for which all sequences of that length or greater sum to 50% of the set’s total size. We used the ngx_plot.py available from https://github.com/rtlorigo/nanopore_assembly_and_polishing_assessment/ to plot phaselineblock N50. From a phased VCF file, we extracted the phaseblock name, contig, start position, and end position to create the input file. We used 3,272,116,950 as the size of the genome to maintain consistency with previous work54. Variant calling and phasing analysis on Gencode annotated regions. We used Gencode v35 (ref. 56) to determine the variant calling and phasing accuracy in gene regions. The Gencode data is publicly available and can be found in the data availability section of the Supplementary Notes.

We used https://github.com/tpeou/slideromics_scripts/gencode_to_stratification.py script to convert the Gencode regions to a bed file that is acceptable to hap.py. With the newly defined stratified regions from Gencode, we ran hap.py to determine the variant calling accuracy in gene regions.

Assembly QV and switch error rate analysis. We assessed the assemblies with yack version 0.1 available from https://github.com/lb3/yack. YACK is a short-read k-mer-based assembly quality estimator. We use short-reads for each sample to estimate the quality of the assembly with k-mer size of 31. With parental short-reads, YACK can also estimate the switch error rate in the assembly. WDL version of the pipeline standard_qc_haploid.wdl is available in https://github.com/human-pangenomics/hpp_production_workflows/.

Homopolymer run-length analysis. We used runLengthMatrix module of margin to derive the homopolymer run-length analysis between assembly and reads. In runLengthMatrix, we convert each read sequence into RLE form and track a map of raw positions to RLE positions. We convert from a raw alignment to RLE alignment by traversing the matches in raw space and tracking the previous RLE match indices. From this set of matched read and reference RLE positions, we construct a confusion matrix. Details of the command are provided in the execution parameters section of the Supplementary Notes.

Small-variant accuracy evaluation of assemblies. We used dipcall version 1.0.2 to identify the small variants from the assemblies. The dipcall variant identification takes the maternal and paternal haplotypes generated by a phased assembly and a reference genome sequence. It maps the haplotypes to the reference and generates a VCF file containing all small variants identified in the assembly. For the haploid assembly, we provided the haploid assembly as both maternal and paternal haplotypes to dipcall, dipcall also generates a bed file containing regions where the assembly maps to the reference. We intersected the bed files to get regions that are assembled by all assembly methods and intersected with GIAB high-confidence region.

Finally, we used hap.py to compare the variant calls derived from the assemblies against GIAB benchmarking VCF to get the accuracy statistics. The Supplementary Notes contain the dipcall parameters.

Structural-variant accuracy evaluation of assemblies. We evaluated SV precision and recall for each assembly as follows. We aligned each set of contigs (either haploid or diploid) against the reference with minimap2 v2.18 using default parameters and the ‘-a’ switch. We have selected the hg19 reference (instead of the November 2018 version) to be able to compare against the curated set of SVs in the HG002 genome54 that was initially produced using the hg19 reference. Given the reference alignment, we used svm-asm 1.0.2 (ref. 55) in the respective (haploid or diploid) mode to call SV of size more than 50 bp. Given two sets of SV calls, we used the SVBenchmark tool from the SVanalyzer package v0.36 with default parameters to estimate recall and precision. To estimate recall for homozygous and heterozygous SVs separately, we split each truth set into two respective parts. The statistics were evaluated within the GIAB Tier 1 high-confidence regions that cover 2.51 Gb of the human genome54.

The assemblies produced by hifiasm had high recall (97.8% homozygous, 97.3% heterozygous) and precision (98%) against the HG002 curated SVs set (Supplementary Table 31). In addition, the recall and precision of the Shasta and PEPPER-Margin-Diploid variant assemblies measured against the HG002 curated SVs were highly correlated with recall and precision measured against the hifiasm SV calls. This allowed us to estimate the SV recovery accuracy of the Shasta and PEPPER-Margin-Diploid variant assemblies for the HG005, HG007, and HG0273 genomes, for which curated sets of SVs were not available.

Method description PEPPER. PEPPER is a recurrent neural-network-based sequence-prediction tool. In PEPPER, we use summary statistics derived from reads aligned to a reference to produce base probabilities for each genomic location using a neural network. We translate the position-specific base probabilities to the likelihood values of candidate variants observed from the read alignments. We propose candidate variants with likelihood value above a set threshold to DeepVariant for genotyping. Candidate prefiltering with PEPPER ensures a balanced classification problem for DeepVariant and achieves high-quality variant calling from erroneous long-reads.
We use PEPPER in two steps in the variant-calling pipeline. Initially, we use the PEPPER-SNP submodule to find SNPs from the initial unphased alignment file. In this setup, we tune PEPPER-SNP to have high precision, so Margin can use the SNPs confidently to phase the genome. To this end, we also exclude INDELs from the callset as they have notably worse performance for nanopore reads. Margin can then tag reads in the alignment file with predicted haplotypes.

After Margin, we use the PEPPER-HP submodule on the phased alignment file to generate haplotype-specific likelihoods for each candidate variant observed from the read alignments. In PEPPER-HP, we consider SNPs, insertions, and deletions as potential candidate variants. We propose the candidate variants with likelihood values higher than a set threshold to DeepVariant for genotyping with a more extensive convolutional neural network (CNN). We tune PEPPER-HP to achieve high-sensitivity but low-precision during candidate finding. The PEPPER-Margin-DeepVariant suite can identify small variants with high-quality from erroneous reads.

**PEPPER-SNP: PEPPER-SNP is a submodule of PEPPER used to identify single-nucleotide polymorphisms from reads aligned to a reference sequence.** PEPPER-SNP works in three steps: image generation, inference, and candidate finding. First, we generate summary statistics from reads aligned to a reference sequence. We encode basic alignment statistics at each genomic location in an image-like tensor format. Second, we apply a recurrent neural network to predict the two most likely bases at each genomic location. Finally, we use the base predictions from each genomic location to compute the likelihoods of SNPs we observe from the reads. We filter candidate variants with a likelihood threshold below a set threshold to find a set of SNPs. Likelihood thresholds for PEPPER-SNP were determined by training the model on HG002 chr1–chr19, selecting an appropriate threshold, and verifying on chr20. The SNP set we get from PEPPER-SNP is used by Margin to phase the alignment file.

**PEPPER-SNP: image generation.** In the image generation step of PEPPER-SNP, we generate summary statistics of base-level information per genomic location. The summary provides weighted observation of bases from all reads divided into nucleotide and orientation.

In PEPPER-SNP, we do not encode insert bases observed in reads to reference alignment, as we only look for SNPs. We use a position value to represent a nucleotide and orientation in the reference sequence. For each genomic location, we iterate over all reads that align to that genomic location and encode ten features to encode base-level information: \([A,C,G,T,Gap(*)]\) divided into two read orientations: forward, reverse. Finally, we normalize the weights of each genomic position based on the read coverage.

Supplementary Fig. 7 describes the feature encoding scheme we use in the image generation step of PEPPER-SNP. The top row of the image, annotated as REF, describes the reference base observed at each genomic location. The colors describing the bases are: A: Blue, C: Red, G: Green, T: Yellow. Each row after REF describes a feature; each feature encodes an observation of nucleotide bases from a forward, or a reverse-strand read. We use ten features to encode the base-level information: \([A,C,G,T,Gap(*)]\) divided into two read orientations: forward, reverse. For example, \(A_1\) encodes the observations of base A from forward-strand reads, and \(A_2\) encodes observations of base A from reverse-strand reads. The columns describe genomic locations to the reference sequence.

In each column, we encode each observation as weights, which we show as alpha of each base. The gray weights are zero weights. At position 23, the weight distribution of \([C,G]\) bases indicates a potential heterozygous variant at that position. The REF row is shown in the figure to describe the scheme; in practice, we do not encode the REF row.

In the inference step of PEPPER-SNP, we use a recurrent neural network for sequence prediction. The network architecture consists of two bidirectional gated recurrent unit (GRU) layers and a linear transformation layer. The linear transformation layer produces a prediction of two bases for each genomic location present in the summary image. To identify potential variants, we use 15 class-labels for base prediction: \([A,A,C,G,A,T,A,*,C,C,G,T,T,C,T,C,*,G,G,T,G,G,T,G,*,T,T,T,T,*,+\)]\).

We do not use colinear classes like CA and AC as two separate classes, as it is not possible to differentiate between these two classes from the summary observations.

From the image-generation module of PEPPER-SNP, we get summary images in 1-kb chunks. We use a sliding-window method during inference and chunk the 1-kb segments into multiple overlapping windows of 100-bp segments. We first run inference on the leftmost window and go to the next window with 50-bp overlapping bases. We pass the hidden state output from the left window to the next window and keep a global inference counter to record base-predictions.

**PEPPER-SNP: inference model.** Supplementary Fig. 8 describes the neural-network-based inference scheme. The two dotted boxes indicate two adjacent windows with a sliding window. The top panel shows the inference scheme. We start with the first window and produce base predictions for each genomic location present in that window; then, we slide the window to the right predictions from all windows in a global counter and report them to the candidate finder to calculate candidate likelihoods.

We trained the inference model using a gradient descent method. We use adaptive moment estimation (Adam) to compute gradients on the basis of a cross-entropy loss function. The loss function is defined to calculate the prediction performance at each genomic location against a labeled set of expected base observations derived from the Genome-In-A-Bottle (GIAB) truth set. The gradient optimization attempts to minimize the loss function by tuning the parameters of the neural network.

We trained PEPPER-SNP with 100× coverage of HG002 data subsampled at different coverage values (20×–100×). We split the training sets into three sets: train, test, and holdout. We use chromosome 1 to 19 for training, chromosome 20 for testing, and we keep 21 and 22 as holdout sets. We train the models for several epochs and test after each epoch. Finally, we pick a model that performs the best on the holdout dataset.

**PEPPER-SNP: candidate finding.** In the candidate finding step of PEPPER-SNP, we take the base-predictions and derive likelihoods of SNPs we observe from the read alignments. If the likelihood value of a variant is above a set threshold, we pick that allele to be a real variant.

In PEPPER-SNP, we derive a allele probability (AP) and a nonreference observation likelihood (NR) for each observed SNP. First, at each genomic location \(pos\) we use the prediction vector of baseclasses \([A,A,C,G,A,T,A,*,C,C,G,T,T,C,T,C,*,G,G,T,G,G,T,G,*,T,T,T,T,*,+\)]\) to derive two prediction vectors \(V_1[pos] = [A_1,C_1,G_1,T_1,Gap(*)]\) and \(V_2[pos] = [A_2,C_2,G_2,T_2,Gap(*)]\). For example, the predicted value of class \([GT]\) contributes to \(V_1[pos][GT]\) and \(V_2[pos][GT]\) values. Then, we iterate over all reads that align to find potential SNPs by recording each read base that does not match the reference base. We calculate the likelihood of candidate base \(b\), observed at position \(pos\), by taking the maximum likelihood from the prediction vectors \(V_1[pos]\) and \(V_2[pos]\) as shown in Eq. 1 where we denote allele likelihood as AP:

\[
P \left[ b, pos \right] = max \left( V_1[pos] [b], V_2[pos] [b] \right)
\] (1)

We also calculate the likelihood of nonreference base observation (NR) to estimate the likelihood of observing any allele other than the reference allele at a location. We derive nonreference base observation likelihood from prediction vectors \(V_1\) and \(V_2\) independently and take the maximum value between two values. For each prediction vector, we take the sum of the values, subtract the observation likelihood of the reference base, and divide by the sum of the prediction vector. Equation 2 describes the calculation NR of PEPPER-SNP:

\[
NR[pos] = \frac{\max \left( \sum_{b=1}^{4} V_1[pos][b], \sum_{b=1}^{4} V_2[pos][b] \right) - \sum_{b=1}^{4} V_1[pos][Ref]}{\sum_{b=1}^{4} V_1[pos][b]}
\] (2)

Finally, we derive a genotype for the variant from the prediction vector \(V_1\) and \(V_2\). If a variant has a likelihood above a set threshold observed in both \(V_1\) and \(V_2\), we set the genotype to be a homozygous alternate (\(1/1\)). If the likelihood is above the threshold in one vector but not in the other, we call it a heterozygous variant (\(0/1\)). For each variant, we use NR value of that position to be the genotyping quality.

**PEPPER-SNP: code availability.** PEPPER-SNP is available at https://github.com/kishwarshafin/pepper.

**PEPPER-HP: haplotype-aware sequence prediction.** PEPPER-HP is a haplotype-aware sequence prediction tool designed to find candidate variants from read alignments. In PEPPER-HP, we take a phased alignment file as input where each read has a haplotag of [0,1,2] indicating which haplotype the read represents (or lack of haplotype information). and we output a set of SNP insertion, and deletion candidates for genotyping using DeepVariant.

Similar to the PEPPER-SNP submodule, PEPPER-HP has three steps: image generation, inference, and candidate finding. In the image-generation step, we generate two sets of summary statistics, one per haplotype, and save them as image-like tensors. We use a recurrent neural network to predict bases on each haplotype for each genomic location in the inference step. Finally, we calculate likelihood values for SNP and INDEL candidates on the basis of base-prediction of each haplotype. We compare candidates with likelihood values over a certain threshold to be candidate variants and propose them to DeepVariant for genotyping. Similarly, likelihood thresholds for PEPPER-HP were determined by training the model on HG002 chr1–chr19, selecting an appropriate threshold, and verifying on chr20.

**PEPPER-HP: image generation.** In the image-generation step of PEPPER-HP, the input is an alignment file with phased reads, and we generate image-like summary statistics of base-level information per genomic location for each haplotype \([0,1]\). The summary of haplotype-1 provides weighted observation of bases from reads with haplotag 0 and haplotype-2 provides weighted observations of bases from reads with haplotag 2. Reads that are unphased or have haplotag 0 contribute to summary statistics for both haplotypes.
In PEPPER-HP, we represent position in reference sequence using two values: position and index. The position value indicates a location in the reference sequence, and we use the index to accommodate insert alleles anchored to a position. All reference sequence positions have an index of 0. On each haplotype [0,1], we iterate over all haplotype associated reads that align a genomic location and encode ten features to encode base-level information: \([A,C,G,T,Gap(*)]\) divided into two read orientations: \([forward,reverse]\). The weights depend on the mapping quality and base quality of the associated reads. Finally, we normalize the weights of each genomic position based on the haplotype associated read coverage.

**PEPPER-HP: Inference model.** In Supplementary Fig. 9, we describe the feature encoding scheme of PEPPER-HP. We derive two summary statistics on the basis of the haplotype association of the reads. The top row of the image, annotated as REF, describes the reference base observed at each genomic location. The colors describe the bases at: \([A:Blue,C:Red,G:Green,T:Yellow,Gap:White]\). Each column represents a reference position with two values (pos,index). For example, \((14,0)\) is the reference position 14 and \((14,1)\) is the insert base anchored in position 14.0. For each haplotype, we use ten features to encode base-level information: \([A,C,G,T,Gap(*)]\) divided into two read orientations: \([forward,reverse]\). From the summary statistics, we see that at location \((23,0)\) the PEPPER-HP measures the base coverage of \(2,3\) bases that match with the reference, denoting a heterozygous variant present in haplotype-1 sequence. We use a recurrent neural network for sequence prediction on haplotype-specific images. The PEPPER-HP network architecture consists of two bidirectional gated recurrent unit (GRU) layers and a linear transformation layer. For each haplotype, the linear transformation layer predicts a base for each genomic location present in the image-like tensor. We use five class-labels for base prediction: \([A,C,G,T,Gap(*)]\). The haplotype-specific images in 1-kb chunks, and we use a sliding window method to slide the window to 50 bp to the right for the next step. We use two global counters from the base predictions, one per haplotype, to record the haplotype-specific base predictions.

The inference scheme of PEPPER-HP is shown in Supplementary Fig. 10. We have two haplotype-specific images for each genomic region representing two haplotypes. For each haplotype, we start from the leftmost window and generate haplotype-specific base predictions. The base predictions are recorded in two global counters.

We train the PEPPER-HP inference model using a gradient descent method. We use adaptive moment estimation (Adam) to compute gradients on the basis of a cross-entropy loss function. The loss function is defined to calculate the prediction performance at each genomic position against a labeled set of expected base observations derived from the GIAB truth set. We use GIAB v3.3.2 truth set as the variants in v3.3.2 are phased. We use phase-specific base predictions to optimize PEPPER-HP model for each haplotype.

We train PEPPER-HP with three sets of HG002 data with 50X, 80x, and 100x coverage. We further generate multiple train sets by arbitrarily downsampling the three training sets at different fractions. We split the training sets into three sets: train, test, and holdout. We use \(1\)–\(19\) for training, \(20\) for testing, and we keep \(21\) and \(22\) for holdout sets. We train the models for several epochs and test after each epoch. Finally, we pick the model that performs the best on the holdout dataset.

**PEPPER-HP: candidate finding.** In the candidate-finding step of PEPPER-HP, we evaluate variants observed from the read alignments. We use the base predictions from the neural network to calculate the likelihood of an observed allele. If the likelihood value of a candidate variant is above a set threshold, we pick that variant as a potential candidate for DeepVariant to assess.

We evaluate the SNPs and INDELs observed in read alignments to find potential candidate variants. First, at each genomic location (pos,index) we use the haplotype-specific base prediction values \(V_{pos, index} = [A,C,G,T,Gap(*)]\) and \(V_{pos, index} = [A,C,G,T,Gap(*)]\). We iterate over all the reads to find potential variants by recording the read base that does not match the reference base. We calculate the allele likelihood of a SNP candidate \(AP_{SMO} \) with base observation \(b\), observed at position \(pos, index\) by taking the maximum likelihood from the prediction vectors \(V_{pos, index}\) and \(V_{pos, index}\) as described in Eq. 3. For inserts and deletes we extend the likelihood calculation to cover the length of the allele.

\[
AP_{SMO} [b, pos] = \max \left( V_{11} \{pos, 0\} [b], V_{12} \{pos, 0\} [b] \right) \tag{3}
\]

We also calculate the likelihood of observing a variant other than the reference allele at any location. In Eq. 4, let \(R(pos,index)\) be the reference base at location \((pos,index)\) and \(N\) be the maximum index value observed in position \(pos\). The reference base at any position with \(index > 0\) is \(gap(*)\). We take the maximum value of observing a nonreference base between \((0,0)\) and \((pos, max, index[pos])\). For each index, we calculate the total value of the prediction vector, subtract the observation likelihood of the reference base and divide by the sum of the nonreference observation likelihood \(NR(pos)\) is the maximum value we observe across index values. For insertion and deletion alleles, we cover the allele length and take the maximum value as the \(NR\) for those candidates.

\[
NR(pos) = \max_{0} \left( \sum_{pos} \frac{V_{pos,pos} - V_{pos,0}}{\sum_{pos} V_{pos,0}} \right) \tag{4}
\]

On the basis of the allele likelihood and nonreference observation likelihood, we calculate a likelihood value for each type of candidate SNP, insertion, and deletion. Then for each type, we set a threshold value and if a candidate passes the threshold value, we propose the candidate to DeepVariant for genotyping.

**PEPPER-HP: code availability.** PEPPER-HP is available at https://github.com/krishashavan/pepper as a submodule.

**Margin.** Margin is a suite of tools employing Hidden Markov Models (HMMs) to perform genomic analysis with long reads. MarginPHASE was introduced as a graph-based assembly polisher which can do standalone polishing and is the first step in a two-part polishing framework MarginPolish-HELEN13. Release 2.0 of Margin incorporates both tools into one suite, including diploid-aware polishing in MarginPolish which has informed improvements in a new iteration of MarginPhase.

In this paper we focus exclusively on improvements made to the phasing submodule. The core partitioning algorithm is described in our previous work13, but we provide a summary here of the previous methodology, followed by a description of the modifications made in the current iteration. First, we give a high-level overview of the phasing workflow.

**Overview.** Margin takes as input an alignment (BAM), reference (FASTA), and variant set (VCF). It determines regions to run on from the alignment and variant set, and it breaks the input into chunks to enable multiprocessing. For each chunk, it extracts reference substrings around each variant site, and read substrings aligning around each variant site. For each variant site, it calculates the alignment likelihood between all substrings and all variant alleles. These likelihoods are used in the core phasing algorithm which bipartitions reads and assigns alleles to haplotypes. After all chunks have been analyzed, we stitch the chunks together to produce results across whole contigs. Last, we output a copy of the input BAM with haplotagged reads and a copy of the input VCF with phased variants.

**Parameterization.** Margin is parameterized with a configuration file. For each configuration, parameters are grouped into polish and phase sections. There is also a default configuration which implements the default parameter values and thresholds determined by experimentation on the HG002 sample.

**Core phasing algorithm.** For the core phasing algorithm, we construct a graph \(G = (V,E)\) describing all possible bipartitions of reads with positions as variant sites, vertices as a combination of position and read bipartitions, and edges as possible transitions between bipartitions for adjacent positions. For example, at position \(P\), with aligned reads \(R\) and \(R\), we have the possible vertices \(V_1\) with haplotypes \(H_1 = (R,R)\) and \(H_2 = (R,R)\) (hereafter described as ‘\(R/R\)’ with the ‘/’ symbol separating the partitions and the ‘\(1\)’ symbol denoting an empty set), \(V_2 = R/R\), \(V_3 = R/R\), and \(V_4 = R/R\). At position \(P\), with the same aligned reads \(R\) and \(R\) and a new read \(R\), vertex \(V_5\) will be split in two parts: \(V_5 = R/R\) and \(V_6 = R/R\), which are both connected to vertex \(V_6\). However, because all reads shared between vertices are the same haplotype assignment, \(V_5 = R/R\) and \(V_6 = R/R\), and is not connected to \(V_6\) because read \(R\) has different haplotype assignments in the two vertices. We extend each vertex description above as previously represent all possible genotypes. After running the forward–backward algorithm on this graph, at each position the posterior distribution over states describing read bipartitions and genotypes can be marginalized to determine the most likely genotype.

The state space for this algorithm increases exponentially with the number of reads at each position. To account for this, Margin implements a pruning and merging heuristic where the input is divided into smaller pieces, unlikely states are pruned, and the resulting graphs are merged before running the full forward–backward algorithm.

**Improved functionality.** One of the most significant changes to the Margin workflow is that we now only analyze sites proposed by the input VCF. Previously, we considered any reference position where less than 80% of the nucleotides agreed with the reference base as a candidate variant site. To determine which proposed variants are considered, we read the input VCF and remove all SNPs with \(\text{PASS} = \text{false}\) or \(\text{VCFEntries} = \text{false}\). We also excluded any nonsynonymous variants and all non-PASS variants. We then ranked the remaining variants using the IMPROVE algorithm.
score below phase.minVariantQuality = 10. For each chunk, we perform an adaptive sampling of variants (phase.useVariantSelectionAdaptiveSampling = true) where we start by taking all variants with a quality score above phase.variantSelectionAdaptiveSamplingMinQualityThreshold = 20. If the average distance between these variants for the chunk is greater than phase.variantSelectionAdaptiveSamplingDesiredBasepairsPerVariant = 2000, we take variants ordered by quality score descending until we achieve the desired number of variants. These values were determined after experimentation on HG002.

Instead of considering only the nucleotide aligned directly to the variant position as the first position of Margin had done, we now extract substrings from the reference and reads and perform an alignment to determine which allele the read most likely originated from. In theory, this allows Margin to use INDELs during phasing, although for our current evaluations we do not test this functionality. When extracting reference bases, we take phase.columnAnchorTrim = 12 bp from the reference boundary (the variant position). The length of the chunk.

To stitch two chunks together, we need to determine whether the two previous chunks were in cis or trans. To do this, we compare the number of reads that are in cis and in trans between the two chunks; if there are more reads in trans, we switch the haplotypes of the current chunk’s reads and variants. To multithread this process, Margin separates all the chunks in a contig into numThreads contiguous groups. The chunks in each of the groups are stitched together by a single thread, and then the same stitching process is used to stitch each of these groups together to complete the whole contig.

The final assignment of a read to a haplotype is determined by the haplotype it was assigned to in the first chunk for the contig. The final assignment of an allele to a haplotype is determined by the chunk it falls within (boundary region excluded).

**Phase set determination.** When writing the output VCF, Margin makes predictions about which sets of variants are confidently inherited together and annotates the output with phaseSet (PS) tags. Margin will assign a phaseSet to a variant if it is heterozygous, a SNP (phase.onlyUseSNPVCFSets = true), and if it agrees with the genotype from the input VCF (phase.updateAlloutputVCFformatFields = false). For variants meeting this criteria, as Margin iterates through the VCF it will extend the current phaseSet unless (1) the variant is the first in the contig, (2) there are no reads spanning between the current variant and the previous variant, (3) there is an unlikely division of reads for the variant (explained below), or (4) the reads spanning the current variant and the previous variant are discordant above some threshold (explained below). The values described below were determined after experimentation on HG002.

To identify unlikely divisions of reads (which we take as potential evidence there is an error in the phasing), we take the number of primary reads assigned to each haplotype and find the binomial p value for that division of reads. If that probability is less than the threshold phaseSetMinBinomialReadSplitLikelihood = 0.0000001, we create a new phase set fr this variant.

Within each chunk region (boundary region excluded) and after determining haplotype assignment for the reads, we track which primary reads were used for phasing and to which haplotype they were assigned in the chunk. This serves as a check against poorly-phased or poorly-stitched chunks. To determine discordancy in the phasing between variants, we compare the number of reads which are in cis or ‘discordant’ (C) given the read assignment to adjacent variants, and the number in trans or ‘discordant’ (C) between variants. If the discordancy ratio C / (C + C) is greater than phaseSetMaxDiscordantRatio = 0.5, we create a new phase set for this variant.

Margin: code availability. Margin is available at https://github.com/UCSC-nanopore-cgl/margin.

**Local phasing correctness.** The LPC is a novel metric for measuring phasing accuracy that we developed for this study. More precisely, the LPC is a family of metrics parameterized by a varying parameter \( \rho \in [0,1] \), which controls the degree of locality. The LPC can be seen as a generalization of the two most common metrics used to evaluate phasing accuracy: the switch error rate and the Hamming rate. The switch error rate corresponds to the LPC with \( \rho = 0 \) (fully local), and the Hamming rate is closely related to the LPC with \( \rho = 1 \) (fully global). With intermediate values of \( \rho \), the LPC can measure meso-level phasing accuracy that the two existing metrics cannot quantify.

The LPC consists of a set of pairs of heterozygous variants where each pair contributes an amount that decays with greater genomic distance. If the variants are incorrectly phased relative to each other, the pair contributes 0. This sum is normalized by its maximum value so that the LPC is always a value between 0 and 1. In mathematical notation,

\[
\text{LPC}_\rho = \frac{\sum_{j \neq i}^{N} \rho^{\delta(i,j)}}{\sum_{j \neq i}^{N} 1}
\]

where \( \delta(i,j) \) is the distance between variants \( i \) and \( j \), and \( \delta(i,j) \) is an indicator for whether variants \( i \) and \( j \) are correctly phased relative to each other. In the case that \( \rho = 0 \) the above formula is undefined, and we instead take the limit of \( \text{LPC}_\rho \) as \( \rho \to 0 \).

If we take \( \delta(i,j) \) to be \( |j-i| \), then it can be shown that \( \text{LPC}_0 \) is equivalent to the complement of the switch error rate (that is the ‘switch correctness’ rate). \( \text{LPC}_0 \) is not equivalent to the Hamming rate, but they are monotonic functions of each other. Thus, \( \text{LPC}_0 \) and Hamming rate always produce the same relative ranking but not the same numerical value. Alternatively, we can take \( \delta(i,j) \) to be the genomic distance between the variants, measured in base pairs. In doing so, the LPC no longer has the provable relationships to switch error rate and Hamming rate, neither of which has any mechanism to incorporate genomic distance. However, we believe the LPC to be a more robust and discriminative metric for phasing accuracy than the variants’ ordinal numbers. Accordingly, we believe that this amounts to a further strength of the LPC over existing metrics, and all LPC values reported in this work use this definition of distance.
The $\rho$ parameter is mathematically convenient but difficult to interpret. To improve interpretability, we can reparameterize the LRC with $\lambda = -\log(\alpha_0) / \rho$, which we call the ‘length scale’. This is the distance (measured either in base pairs or number of variants) at which a pair of variants has 1/2 of the maximum weight. The length scale gives an approximate sense of the scale of distances that the LRC incorporates, although it is worth stressing that pairs that are closer together always receive more weight than pairs that are further apart.

**Local phasing correctness: code availability.** The code used to calculate LRC is available at [https://github.com/UCSC-nanopore-cgl/margin](https://github.com/UCSC-nanopore-cgl/margin).

**DeepVariant.** DeepVariant is a small-variant identification method based on deep neural networks that achieve high performance on different short-read and long-read sequencing platforms. DeepVariant has released models for different datatypes and a details of the methods implemented in DeepVariant can be found in associated releases. Here we present an overview of the methods we implemented to achieve high performance with nanopore data.

**Adapting DeepVariant to Oxford Nanopore reads.** DeepVariant performs variant calling in three stages, ‘make examples’, ‘call variants’ and ‘postprocess variants’. In ‘make examples’, potential variant positions are identified by applying a minimal threshold for evidence. In positions meeting the candidate generation criteria, the reads overlapping the position are converted into a pileup of a 221-bp window centered at the variant. Multiple features of the reads are represented as different dimensions in the pileup, including the read length, base quality, mapping quality, strand, whether the read supports the variant, and whether the base matches the reference. Prior to this work, the heuristics for candidate generation were simple (at least 2 reads supporting a variant allele and a variant allele fraction at least 0.12). However, the higher error rate of Oxford Nanopore data generated far too many candidates for DeepVariant to call variants in a genome in a reasonable time. To integrate DeepVariant with PEPPEr, we developed the ability to import candidates directly from a VCF to replace the logic in ‘make examples.’ This allows DeepVariant to read the output of PEPPEr, and in theory, can be used in a similar manner with the outputs of other methods.

DeepVariant has released models for different datatypes. In order to achieve high perfomance on Oxford Nanopore data, a new DeepVariant model trained for this dataype was required. For this, we modified the training process for DeepVariant. The training process for DeepVariant is very similar to the variant calling process (generating candidates from make examples in a similar manner), but with the addition of a step which reconciles a candidate variant with the truth label from GIAB. We adapted this process to use the VCFCandidateConverter to propose PEPPEr candidates. Because the representations of variants can diverge from the representation in GIAB (even for the same variant), modifying the process for training required multiple rounds of iteration to identify mislabeled edge cases and to modify the proposed candidate representation for training.

With these modifications, training of DeepVariant models using the existing machine could proceed. These modifications also allowed existing logic for the stages ‘call variants’ and ‘postprocess variants’ to work directly with the trained Oxford Nanopore model.

**DeepVariant: code availability.** All code in the DeepVariant repository incorporates the implementations made in this paper (https://github.com/google/deepvariant). This repository also contains a model retraining tutorial ([https://github.com/google/deepvariant/blob/1.1/docs/deepvariant-training-case-study.md](https://github.com/google/deepvariant/blob/1.1/docs/deepvariant-training-case-study.md)).

**Training PEPPEr-DeepVariant for ONT bacinellers.** It is necessary to train new models once the underlying read quality of Oxford Nanopore changes (that is bacinellar update). It is possible to take the existing PEPPEr model and re-train it with 100X HG002 on one gpu for a few hours to get a model for the newer chemistry. Similarly, it is possible to take the existing DeepVariant model and train the model for a few hours on one GPU to get a model for the new bacinellar. A case study of retraining an Illumina model for BGISEQ is available here ([https://github.com/google/deepvariant/blob/1.1/docs/deepvariant-training-case-study.md](https://github.com/google/deepvariant/blob/1.1/docs/deepvariant-training-case-study.md)).

As the bacineller updates of ONT are incremental, we believe retraining to newer models once the underlying read quality of Oxford Nanopore changes (that is with 100X HG002 on one gpu for a few hours to get a model for the newer chemistry. Similarly, it is possible to take the existing DeepVariant model and train the model for a few hours on one GPU to get a model for the new bacinellar. A case study of retraining an Illumina model for BGISEQ is available here ([https://github.com/google/deepvariant/blob/1.1/docs/deepvariant-training-case-study.md](https://github.com/google/deepvariant/blob/1.1/docs/deepvariant-training-case-study.md)).

**Assembly polishing with PEPPEr-Margin-DeepVariant.** The assembly polishing method of PEPPEr-Margin-DeepVariant is described below:

1. **PEPPER-SNP:** PEPPEr-SNP finds SNPs from the read alignments to a haploid assembly using a RNN. For assembly polishing, we use the same infrastructure described in the haplotype-aware variant calling section.

2. **Margin:** Margin takes the SNPs reported by PEPPEr-SNP and generates a haplotaged alignment file using a HMM. For assembly polishing, we use the same infrastructure described in the haplotype-aware variant calling section.

3. **PEPPER-HP:** PEPPEr-HP takes the hapltagged alignment file and evaluates each haplotype independently and produces haplotype-specific candidate SNP and INDEL-like errors present in the assembly.

   - Haplotype-1 and Haplotype-2 candidate finding: For haplotype-1, we take reads with HP-1 and HP-0 and generate base-level summary statistics. Then we use a RNN to produce nucleotide base predictions for each location of the genome. Finally, we find all the observed SNP or INDEL-like candidates in the reads with HP-1 and HP-0 tag and calculate a likelihood of each candidate being an error in the assembly. If the likelihood of the candidate is above a set threshold, we propose that candidate as a potential edit for haplotype-1 (HP-1) of the assembly. For haplotype-2, we take reads with HP-2 and HP-0 and find SNP and INDEL-like candidate errors following the same process described in haplotype-1 candidate finding.

4. **DeepVariant:** DeepVariant takes the haplotype-specific candidate set from PEPPEr-HP and the hapltagged alignment from Margin to identify errors present in the assembly using a convolutional neural network (CNN).

   - Haplotype-1 and haplotype-2 polishing: For each candidate error from haplotype-1 set proposed by PEPPEr-HP, we generate a feature set representing base, base-quality, mapping quality, and so on, in a tensor-like format. In haplotype-specific polishing setup, reads are sorted by their haplotags in the order HP-1, HP-0, HP-2. Then we use a pre-trained inception_v3 CNN to generate a prediction of [0/0, 1/1] for each candidate. Candidates in haplotype-1 that are classified as [1/1] are considered as missing heterozygous variants from the haploid assembly or an error present in the assembly. We apply the candidates of haplotype-1 classified as [1/1] to the haploid assembly using befoots consensus to get a polished assembly haplotype_1 fasta representing one haplotype of the sample.

   - For haplotype-2, we sort the reads by their haplotags in the order (HP-2, HP-0, HP-1). Candidates in haplotype-2 that are classified as [1/1] are similarly considered to be a missing heterozygous variant or an error present in the assembly, and are applied to the haploid assembly to get the second polished haplotype (haplotype_2 fasta).

   - We trained the PEPPEr-Margin-DeepVariant assembly polishing method with HG002 assembly generated by the Shasta assembler. We first aligned the HG002 Shasta assembly to GRCh38 reference with dipcall to associate assembly contigs with GRCh38 reference. Then we aligned HG002 GIAB v4.2.1 small-variant benchmarking set to the assembly and marked the high-confidence regions to restrict training only in regions that GIAB notes as high quality. The alignment between HG002 GIAB v4.2.1 to the assembly produced the training set. We trained PEPPEr and DeepVariant on contigs that associate to chr1–chr19 and used chr20 as a holdout set. We used the same approach for Oxford Nanopore and PacBio HiFi-based models.

**Reporting summary.** Further information on research design is available in the Nature Research Reporting Summary linked to this article.

**Data availability.** We have made the analysis data available publicly (variant calling outputs, genome assemblies) in: [https://console.cloud.google.com/storage/browser/pepper-deepvariant-public/analysis_data](https://console.cloud.google.com/storage/browser/pepper-deepvariant-public/analysis_data). The source data for the main figures can be found in: [https://console.cloud.google.com/storage/browser/pepper-deepvariant-public/figure_source_data/figure_source_data/](https://console.cloud.google.com/storage/browser/pepper-deepvariant-public/figure_source_data/figure_source_data/).

For sequencing data, we used several publicly available datasets:

- GIAB consortium: [https://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/](https://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/)
- Human Pangenome Reference Consortium (HPRC): [https://s3-us-west-2.amazonaws.com/human-pangenomics/index.html](https://s3-us-west-2.amazonaws.com/human-pangenomics/index.html)
- T deline-to-telomere consortium: [https://github.com/google/deepvariant/blob/1.1/docs/deepvariant-training-case-study.md](https://github.com/google/deepvariant/blob/1.1/docs/deepvariant-training-case-study.md)

Please see the Supplementary Notes to find specific links to the sequencing data that we used for our analysis. Source data are provided with this paper.

**Code availability.** The modules of PEPPEr-Margin-DeepVariant are publicly available in these repositories on GitHub with associated issue trackers:

- PEPPEr: [https://github.com/kishwarshafin/pepper](https://github.com/kishwarshafin/pepper)
- Margin: [https://github.com/UCSC-nanopore-cgl/margin](https://github.com/UCSC-nanopore-cgl/margin)
- DeepVariant: [https://github.com/google/deepvariant](https://github.com/google/deepvariant)

The PEPPEr-Margin-DeepVariant software is available at [https://doi.org/10.5281/zenodo.5275510](https://doi.org/10.5281/zenodo.5275510), and we used r0.4 version for the evaluation presented in this manuscript. For simpler use, we have also created a publicly available docker container, kishwars/pepper_deeprvariant:r0.4, that can run our variant-calling and polishing pipelines.

**References.**

52. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 34, 3491–3499 (2018).
53. Li, H. et al. The sequence alignment/map format and SAMtools. *Bioinformatics* 25, 2078–2079 (2009).
54. Krusche, P. et al. Best practices for benchmarking germline small-variant calls in human genomes. *Nat. Biotechnol.* 37, 555–560 (2019).
55. Cleary, J. G. et al. Joint variant and de novo mutation identification on pedigrees from high-throughput sequencing data. *J. Comput. Biol.* **21**, 405–419 (2014).
56. Newey, W. K. Adaptive estimation of regression models via moment restrictions. *J. Econom.* **38**, 301–339 (1988).
57. K. Shafin, et al. PEPPER-Margin-DeepVariant (version r0.4), https://doi.org/10.5281/zenodo.5275510 (Zenodo, 2021).

**Acknowledgements**

Research reported in this publication was supported by the National Human Genome Research Institute of the National Institutes of Health under award numbers U41HG010972, R01HG010485, U01HG010961, and OT2OD026682 (K.S., T.P., M.K., J.M.E, K.H.M., M.J., B.P.). We thank Circulomics Inc. for sharing HG001 Nanopore data. We thank J. Zook and J. Wagner from the National Institute of Standards and Technology (NIST) for providing a draft version of the HG005 v4.2.1 benchmark. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

**Author contributions**

B.P. and A.C. designed and executed the study. K.S. developed PEPPER. T.P. developed Margin. P.-C.C. designed candidate import functionality in DeepVariant. K.S., T.P., and P.-P.C. contributed equally to the methods development and core analysis presented. M.N. designed alt-event alignment in DeepVariant; A.K. contributed to haplotype sorting and improvements on DeepVariant runtime; S.G. contributed to candidate import module of DeepVariant; G.B. designed and executed the post-processing model to improve multiallelic variant accuracy; M.K. designed and evaluated assembly polishing; J.M.E. designed local phasing metric and contributed to phasing evaluation. K.H.M. provided experimental design guidance, and P.C. generated assemblies and provided guidance on assembly polishing. M.J. performed nanopore sequencing, quality control and helped to design and execute analysis. All authors approve of the final manuscript.

**Competing interests**

K.S. has performed paid internships at NVIDIA Corp and Google. P.C., M.N., A.K., S.G., G.B., and A.C. are employees of Google and own Alphabet stock as part of the standard compensation package. M.J. has received reimbursement for travel, accommodation, and conference fees to speak at events organized by ONT. The remaining authors declare no competing interests.

**Additional information**

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41592-021-01299-w.

Correspondence and requests for materials should be addressed to Andrew Carroll or Benedict Paten.

Peer review information Nature Methods thanks Ruibang Luo and the other, anonymous, reviewers for their contribution to the peer review of this work. Peer reviewer reports are available. Lin Tang was the primary editor on this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Reprints and permissions information is available at www.nature.com/reprints.
Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

- The exact sample size (n) for each experimental group/condition, given as a discrete number and unit of measurement
- A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
- The statistical test(s) used AND whether they are one- or two-sided
- Only common tests should be described solely by name; describe more complex techniques in the Methods section.
- A description of all covariates tested
- A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
- A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
- For null hypothesis testing, the test statistic (e.g. F, t, r) with confidence intervals, effect sizes, degrees of freedom and P value noted. Give P values as exact values whenever suitable.
- For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
- For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
- Estimates of effect sizes (e.g. Cohen’s d, Pearson’s r), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection: No software was used for data collection.

Data analysis:
- Base calling: Oxford Nanopore Technology provided Guppy basecaller v4.2.2 (commercial product)
- PEPPER v0.4: https://github.com/kishwarshaflinerpepper (MIT License)
- Margin v2.0: https://github.com/UCSC-nanopore-cgl/margin (MIT License)
- DeepVariant v1.1.0: https://github.com/google/deepvariant (BSD-3-Clause License)
- Minimap2 2.17-r941: https://github.com/lh3/minimap2 (MIT License)
- Pbmm2 1.4.0: https://github.com/PacificBiosciences/pbmm2 (BSD-3-Clause-Clear License)
- Samtools 1.10: https://github.com/samtools/samtools (MIT/Expat License)
- Medaka v1.2.1: https://github.com/nanoporetech/medaka (MPL-2.0 License)
- Clair v2.1.1: https://github.com/HKU-BAL/Clair (BSD-3-Clause License)
- Longshot v0.4.2: https://github.com/pjedge/longshot (MIT License)
- Hap.py v0.3.12: https://github.com/Illumina/hap.py (simplified BSD license)
- WhatsApp v1.0: https://github.com/whatsap/whatsap (MIT License)
- Canu (Trio-binning) v2.1.1: https://github.com/marbl/canu (GNU General Public License, version 2)
- Natural switch error visualization: https://github.com/tpesout/generomics_scripts (MIT License)
- Local Phasing Correctness visualization: https://github.com/tpesout/generomics_scripts (MIT License)
- hifiasm 0.14: https://github.com/chyylp123/hifiasm (MIT License)
- Shasta 0.7.0: https://github.com/chunzuckerberg/shasta (MIT License)
- Flye 2.8.2: https://github.com/fenderglass/Flye (BSD-3-Clause License)
- K-mer analysis with YAK 0.1: https://github.com/lh3/yak (MIT License)
- dipcall v0.2: https://github.com/lh3/dipcall (No License/CC0)
- RTG 3.12: https://github.com/RealTimeGenomics/rtg-tools (BSD-2-Clause)
Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:
- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

We have made the analysis data that we used to generate the figures available publicly (variant calling outputs, runtime, genome assemblies etc.) in:
https://console.cloud.google.com/storage/browser/pepper-deepvariant-public/analysis_data.

Sequencing data availability:
Genome-In-a-Bottle consortium: https://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/
Human Pangenome Reference Consortium (HPRC): https://s3-us-west-2.amazonaws.com/human-pangenomics/index.html
Telomere-to-telomere consortium: https://github.com/nanopore-wgs-consortium/CHM13

In the supplementary data section, we have listed the URLs of the FASTQ files used for each sample that are publicly available via different consortium.

There are no restrictions on any of the data we presented in our study.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size: No sample size calculation was performed as we used all available sample data for which GIAB has truth variants (training on HG002, evaluating on HG001, HG003-HG007).

Data exclusions: No data was excluded from the study.

Replication: Each sample was sequenced using a consistent sequencing protocol. We evaluated on all available data for which GIAB has truth variants.

Randomization: Randomization was not relevant to our study.

Blinding: Investigators were not blind to group, as all data was pooled together and publicly available.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

| Materials & experimental systems | Methods |
|---------------------------------|---------|
| n/a | n/a |
| ☒ | ☐ |
| Antibodies | Involved in the study |
| ☐ | ☐ |
| Eukaryotic cell lines | ChiP-seq |
| ☒ | ☐ |
| Palaeontology and archaeology | Flow cytometry |
| ☒ | ☐ |
| Animals and other organisms | MRI-based neuroimaging |
| ☒ | ☐ |
| Human research participants | |
| ☒ | |
| Clinical data | |
| ☒ | |
| Dual use research of concern | |