SquiggleFilter: An Accelerator for Portable Virus Detection

Tim Dunn*
timdunn@umich.edu
University of Michigan
Ann Arbor, MI, USA

Harisankar Sadasivan*
hariss@umich.edu
University of Michigan
Ann Arbor, MI, USA

Jack Wadden
jackwadden@gmail.com
University of Michigan
Ann Arbor, MI, USA

Kush Goliya
kgoliya@umich.edu
University of Michigan
Ann Arbor, MI, USA

Kuan-Yu Chen
knyuchen@umich.edu
University of Michigan
Ann Arbor, MI, USA

David Blaauw
blaauw@umich.edu
University of Michigan
Ann Arbor, MI, USA

Reetuparna Das
reetudas@umich.edu
University of Michigan
Ann Arbor, MI, USA

Satish Narayanasamy
nsatish@umich.edu
University of Michigan
Ann Arbor, MI, USA

ABSTRACT

The MinION is a recent-to-market handheld nanopore sequencer. It can be used to determine the whole genome of a target virus in a biological sample. Its Read Until feature allows us to skip sequencing a majority of non-target reads (DNA/RNA fragments), which constitutes more than 99% of all reads in a typical sample. However, it does not have any on-board computing, which significantly limits its portability.

We analyze the performance of a Read Until metagenomic pipeline for detecting target viruses and identifying strain-specific mutations. We find new sources of performance bottlenecks (basecaller in classification of a read) that are not addressed by past genomics accelerators.

We present SquiggleFilter, a novel hardware accelerated dynamic time warping (DTW) based filter that directly analyzes MinION's raw squiggles and filters everything except target viral reads, thereby avoiding the expensive basecalling step. We show that our 14.3W 13.25mm^2 accelerator has 274× greater throughput and 3481× lower latency than existing GPU-based solutions while consuming half the power, enabling Read Until for the next generation of nanopore sequencers.

ACM Reference Format:
Tim Dunn, Harisankar Sadasivan, Jack Wadden, Kush Goliya, Kuan-Yu Chen, David Blaauw, Reetuparna Das, and Satish Narayanasamy. 2021. SquiggleFilter: An Accelerator for Portable Virus Detection. In *MICRO'21: 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '21), October 18–22, 2021, Virtual Event, Greece.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3466752.3480117

1 INTRODUCTION

The COVID-19 pandemic caused by the SARS-CoV-2 virus continues on a global scale. Today, diagnostic tests are widely available to detect SARS-CoV-2. Most of these tests involve some form of Polymerase Chain Reaction (PCR), a common technique for exponentially amplifying DNA/RNA. In order to detect a virus such as SARS-CoV-2, custom "primers" are first designed and manufactured which will only attach to and amplify specific regions of DNA/RNA in the target virus's genome. After PCR, the virus's presence or absence can be determined based on whether the amplification was successful or not.

A significant shortcoming of the current approach is that PCR primers are targeted to a specific virus. **Custom primer design is a complex, error-prone, and time-consuming process** [44] [43]. Even though SARS-CoV-2's RNA was sequenced in early January 2020, validated SARS-CoV-2 specific PCR primers took several months to develop [43] [2]. Lack of mass testing capability in the early stages of SARS-CoV-2 made it difficult to detect and control its spread, leading to a catastrophic pandemic. While we now have adequate testing capability for SARS-CoV-2, it is not unlikely for another novel virus like SARS-CoV-2 or its variants to emerge in the near future [40], and if it does, we need to be prepared with adequate testing infrastructure in place to detect and control its spread in the early stages.

We envision a programmable virus detector (one that constructs whole viral genomes) that can be deployed worldwide. As soon as an emerging novel virus is discovered and sequenced, the reference genome of the novel virus would be distributed to all the devices, instantly turning them into targeted detectors.

Our solution uses Oxford Nanopore Technologies' (ONT) MinION Mk1B (henceforth, referred to as the MinION), a new-to-market palm-sized DNA/RNA sequencer. It is fairly low-cost, portable, and can sequence long reads in real time.

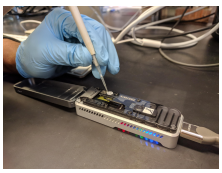

Figure 1: MinION sequencer in our laboratory.

We replace targeted PCR with universal PCR [62], which amplifies *all* DNA/RNA. Thus, it avoids the problem of custom PCR primer design and deployment mentioned earlier. However, this introduces a different problem, as up to 99.99% of the DNA/RNA

*Both authors contributed equally to this research.

MICRO '21, October 18–22, 2021, Virtual Event, Greece

ACM ISBN 978-1-4503-8557-2/21/10.
https://doi.org/10.1145/3466752.3480117

in a typical biological specimen (e.g. saliva) is non-viral [26] (non-target) and most belongs to the host. Amplifying all DNA/RNA preserves this ratio, resulting in the vast majority of sequencing and computing time and cost stemming from processing non-target DNA/RNA.

In order to solve this needle-in-a-haystack problem, ONT sequencers have a feature called **Read Until** [21]. As reads (DNA/RNA fragments) are sequenced, they need to be analyzed in real-time. As soon as the computer classifies that the read is non-viral, the sequencer is instructed to eject it, which saves the time and cost of sequencing non-viral reads (>99% of all reads). Unfiltered viral reads are used to construct the whole virus genome using reference-guided assembly (alignment and variant calling).

The MinION, however, does not have any on-board computing power to perform such secondary analysis. In this paper, **we analyze the performance of the Read Until bioinformatics pipeline** for efficiently sequencing viral pathogens, and realize a portable computing solution that can be integrated with MinION.

We discover new performance bottlenecks that are not addressed by past genomics accelerators [20, 23, 24, 28, 33, 41, 55, 61]. In particular, we find that the Deep Neural Network (DNN) basecaller (software that translates MinION's electrical squiggles to AGTC bases) dominates the computing time (96%). The aligner and variant caller, which have been the targets of recent accelerator research, constitute a much smaller fraction of compute. We also find that a current edge GPU is inadequate to keep up with the throughput of the MinION. Also, its high latency in classifying a read prevents us from taking advantage of the latency-critical Read Until feature of MinION.

Converting squiggles to bases using a compute-intensive basecaller, and then aligning to check if a read belongs to the target virus is needlessly expensive for classifying it. Instead, **we skip the basecaller altogether by directly comparing each read's squiggles to the precomputed expected signal profile of the target virus's entire reference genome** (the "reference squiggle"). By skipping the compute-intensive basecaller step, we improve efficiency significantly.

We present SquiggleFilter, a hardware/software co-designed filter which identifies non-target reads by directly comparing the real-time measured squiggles to the target virus's precomputed reference squiggle. A classification decision is made based on the degree of match. **We develop a custom subsequence dynamic time warping (sDTW) algorithm** [18] to perform this classification. It includes solutions that improve accuracy by adaptively examining longer read prefix lengths when needed. It also includes customizations that result in area efficient hardware.

sDTW-based SquiggleFilter is significantly more efficient than a DNN-based basecaller, and its regular compute-bound characteristic makes it amenable for hardware acceleration. sDTW is a dynamic programming algorithm [49] whose complexity is proportional to the product of the length of the reference (R) and query (Q). Its regular memory access pattern allows us to build a fast and space efficient 1D systolic array accelerator for sDTW with a constant number of processing elements. Fortunately, we find that almost all epidemic viruses have genome references of length 50,000 (R) bases or smaller (see Figure 10) [39]. As a result, our accelerator can easily complete the classification in ~2R cycles (forward and backward of reference strand), and still meet the strict latency requirement for leveraging Read Until.

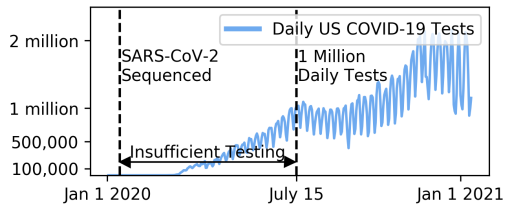

Figure 2: Progression of US COVID-19 testing [30]

Our work makes the following contributions:

- we demonstrate that basecalling is the computational bottleneck in the virus sequencing pipeline. Read alignment and variant calling – targets for prior accelerator work – are not the bottleneck.
- we identify direct squiggle alignment (first proposed in [38]) as a more efficient alternative to basecalling and alignment when enriching low-concentration viral specimens with Read Until.
- we propose multi-stage sDTW and several modifications to vanilla sDTW to realize an accurate and efficient hardware accelerator.
- we co-design a sDTW hardware accelerator to filter non-viral reads, for variable read prefix and almost all infectious viral genome lengths
- we demonstrate that this hardware, unlike current approaches, will enable Read Until to scale with rapidly increasing nanopore sequencing throughput
- we quantify accuracy and efficiency of our classifier using real-world metagenomic datasets, including datasets collected from our wet-lab experiments for Read Until.

Results: We design an edge device with compute capabilities similar to a Jetson Xavier System-on-Chip [7] consisting of SquiggleFilter, an edge GPU, and an 8-core ARM processor. We show that our proposed SquiggleFilter can accurately distinguish target viral DNA/RNA from background human DNA/RNA. We evaluate accuracy using non-contagious lambda phage virus data sequenced in our own lab. In terms of efficiency, we show that our SquiggleFilter accelerator has 274× higher throughput than the conventional software pipeline (using a basecaller) on an edge GPU while only consuming an area of 13.25mm^2 and power of 14.31W. SquiggleFilter's throughput is 233.65M samples/s, which far exceeds the maximum throughput of 2.05M samples/s on a MinION [58], and is adequate to handle up to a 114× increase in MinION's throughput in the future. The latency for classifying any read is 0.043ms, which is insignificant to Read Until decision's critical path.

2 BACKGROUND

2.1 Need for a Virus Detector

While SARS-CoV-2 was discovered – and its RNA genome sequenced – by early January 2020, it was not until several months later that mass testing was available worldwide. Figure 2 shows the steady increase in daily COVID-19 tests performed within the

United States [30]. A widely established global testing infrastructure would have helped control the spread of the virus early on, possibly saving hundreds of thousands of lives.

Given the increasing frequency of viral outbreaks, experts are concerned that it is only a matter of time before a new virus threatens the globe [40]. Thus, we need a virus testing technology that can be widely deployed *ahead-of-time*, and reprogrammed to detect and identify mutations in novel viruses as soon as they emerge.

In this work, we focus on controlling the spread of novel infectious viruses in their early stages, as soon as they are discovered and sequenced. **Our goal is to enable a universal rapid test that can determine the whole genome of a target virus using reference-guided assembly**. Targeting a specific virus enables us to make significant optimizations that help us reduce time and cost of sequencing and compute.

2.2 State-of-the-art Virus Detectors

Tests	Diagnostic Power	Programmable	Time (min)	Cost ($)
Antigen-based test				
Paper [11]	presence		15	5
Non-sequencing molecular test				
RT-LAMP [25][16]	presence		60	15
RT-PCR [1]	presence		120-240	<10
Sequencing based molecular test (30× coverage)				
ARTIC [4][1]	98 targets		305	100
LamPORE [31]	3 targets		<65	-NA-
RNA: 1% virus	whole genome	✓	240	110
0.1% virus [6]	whole genome	✓	1206	190
DNA: 1% virus	whole genome	✓	320	105
0.1% virus [5]	whole genome	✓	470	120

Table 1: A comparison of popular commercial and ONT sequencing-based virus detectors for SARS-CoV-2.

Table 1 lists commonly used tests and ONT-based sequencing solutions for SARS-CoV-2. None of the methods except direct RNA or DNA sequencing are programmable, and therefore, are not effective in controlling the pandemic in its early stages. Antigen (paper) tests detect specific surface proteins on the virus. They are cheap, portable, and fast. However, they have low sensitivity and can only detect viruses present at high concentrations.

Molecular tests identify specific regions of interest in a virus's genome and amplify this DNA if present in the specimen. Polymerase Chain Reaction (PCR) is a common technique used for amplification. It has high sensitivity [42] but requires thermal cycling, which can be slow and expensive. LAMP (Loop Mediated Isothermal Amplification) is a more recent technology that obviates the need for a thermal cycler, but its primers are more complicated to design than PCR.

If amplification was successful (i.e., target DNA is present), it can be detected using fluorometry or colorimetry. Most clinical tests for SARS-CoV-2 stop here. However, by sequencing the amplified specimen, we can assemble portions of virus's genome, depending on the number of targets amplified. ARTIC and LamPORE [31] amplify 98 and 3 genes respectively, and then use ONT's nanopore sequencing.

Current solutions for virus detection use multiplex primer sets specific to a virus. Primer design is a complex, error-prone and time-consuming process [44] [43]. Thus, they are not an effective solution for early pandemic control. The COVID-19 pandemic highlights this problem, where designing and distributing target-specific primers was challenging, especially when supply chains broke amidst the pandemic.

An alternative to developing custom primers is to directly sequence the specimen following amplification with universal primers, which non-selectively amplify all DNA. This amplification step is required to increase the quantity of DNA, which greatly reduces average capture time (the time required for a DNA strand to enter a nanopore) and therefore sequencing time. The wet-lab protocol followed, Sequence Independent Single Primer Amplification (SISPA) [9, 17], is universal and hence can be used on all RNA viruses. SISPA has four major steps: (1) RNA extraction, (2) complementary DNA generation, (3) PCR amplification, and (4) final sequencing specimen preparation.

A significant hurdle to SISPA-based sequencing is that following amplification, the specimen contains the genetic material of the target virus among a sea of human and bacterial DNA/RNA. The proportion of target virus DNA/RNA can be as low as 0.01% percent [26]. As a result, the time and cost of sequencing and data processing for this approach is significantly greater than that of custom primer-based solutions.

If this cost barrier can be overcome, this approach would enable detection of novel viruses without requiring months to develop and distribute virus-specific primers. Read Until can greatly increase the efficiency of sequencing by filtering out non-target reads using the virus's reference genome. Current Read Until approaches are limited by insufficient throughput, but our hardware accelerated SquiggleFilter ensures the future scalability of Read Until on higher throughput sequencers.

2.3 Portable MinION Sequencer

Oxford Nanopore Technology's (ONT) MinION offers multiple benefits that makes it a uniquely attractive solution for mobile and rapid virus detection.

Long reads: MinION sequencers are capable of measuring long strands of DNA, and can theoretically sequence any strand, regardless of length. The current world record stands at over 4 million bases [34].

Cost: The MinION only costs $1,000, and offers affordable specimen preparation kits ($100/use) and flow cells ($125/use assuming 4× re-use). In comparison, it costs $80,000-$100,000 to purchase even the most affordable "Next Generation Sequencing" machines.

Real-time: MinION sequencers provide real-time, streaming output from the device. Streaming signal output enables on-the-fly secondary analyses, and the ability to stop sequencing as soon as the desired coverage is reached.

Portability: A key feature that sets the MinION sequencer apart from all other sequencers in terms of wet-lab, sequencing and compute as shown in Figure 3. The portable compute, however, remains inefficient for real-time sequencing.

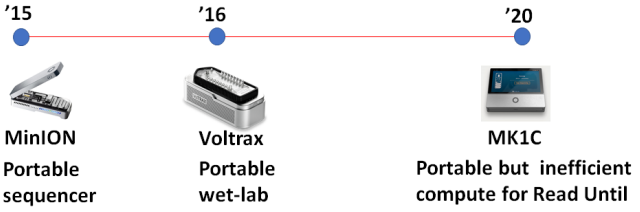

Figure 3: Sequencing and wet-lab is portable. Compute, though portable, is insufficient for Read Until.

Target enrichment: An especially exciting capability of the MinION sequencer is **"Read Until"**, which ejects non-target DNA/RNA strands by reversing the electrical potential across the pore. This effectively enables digital enrichment of target DNA/RNA in low-concentration specimens.

However, a slow read classification results in wasted sequencing time. Currently, the MinION has no inbuilt computing power to make Read Until decisions. We additionally find that commodity GPUs are undesirable in terms of both throughput, latency and power.

3 COMPUTE BOTTLENECKS IN PORTABLE VIRUS DETECTION

Our goal is to build a cost and time efficient sequencing pipeline for determining the whole genome of a targeted virus, but without using custom primers for target amplification. We seek to reduce time and cost using the Read Until feature of Oxford Nanopore (ONT)'s palm-sized MinION sequencer.

To this end, we constructed a software pipeline using state-of-the-art bioinformatics tools and analyzed its performance. Our profiling results expose new performance bottlenecks that are different from those targeted in past accelerators for human genome sequencing [20, 23, 24, 28, 33, 41, 55, 61].

3.1 Bioinformatics Pipeline

The MinION sequencer measures electrical current signals that represent the bases (A, G, T, C) moving through each pore, recording approximately 10 samples for each base. All the active pores (up to 512 in the MinION) concurrently produce squiggles for the reads flowing through them. These squiggles can be analyzed in real-time as the reads flow through the pores.

Figure 4 illustrates the analysis pipeline for the squiggles. A *basecaller* translates squiggles into bases. The latest basecallers (such as ONT's Guppy [59]) use compute-intensive DNNs, which must be large and deep to attain state-of-the-art accuracy. Guppy processes reads in chunks of 2000 samples, and uses five bidirectional LSTM layers for encoding followed by a custom CTC (Connectionist Temporal Classification) decoder. ONT provides two versions of its basecaller: a high-accuracy version (Guppy), and another that trades off accuracy for performance (Guppy-lite).

In our Read Until pipeline, squiggles of a read are basecalled in real-time. After a short prefix of a read has been basecalled, it is then processed by an aligner (MiniMap2 [36]) that aligns the read to the target's reference genome. If a good alignment is found, then the read is classified as a target and passed on to the next stage. Otherwise, a signal is sent to the MinION device, instructing it to eject the non-target read from further sequencing. **Thus, the critical computing path for Read Until includes both the basecaller and aligner**.

The target reads are collected and analyzed by a variant caller (Racon [57] followed by Medaka [8]). We seek to cover every position in the reference genome by 30 reads (30× coverage). The variant caller analyzes the reads piled up at each reference genome location, and identifies any genomic differences ("variants") between the sequenced and reference viruses. **As the variant caller is not involved in Read Until decisions, it is off the critical path**.

3.2 Performance Bottlenecks

Figure 5 shows the performance bottlenecks of the bioinformatics pipeline (Section 3) used to assemble the whole SARS-CoV2 genome, evaluated on the CPU and GPU in Table 3. The results are shown for two representative biological specimens, one where the target viral reads constitute 1% of all the reads, and the other 0.1%.

We observe that a large fraction of computing time (96%) goes towards basecalling. This is in spite of using the more efficient, but less accurate, Guppy-lite.

Compute spent towards aligning (MiniMap2) and variant calling (Racon and Medaka) constitutes significantly smaller fraction, especially for specimens with low viral load (0.1%). In contrast, prior work on genomics accelerators targeted aligners and variant callers used for reference-guided assembly of human DNA [20, 23, 24, 28, 33, 41, 55, 61]. There are several reasons for this significant difference, discussed next.

All the reads are aligned to a target viral genome to classify them as target or non-target. This alignment step, however, is significantly less compute intensive compared to aligning to a human genome, because viral genomes are much shorter (≈30,000 bases) than human DNA (3 billion bases).

Only a small fraction of target reads (1% to 0.1%) need to be processed for reference-guided assembly of a viral genome. Therefore, the variant caller is invoked only for a small fraction of sequenced reads. Also, given that viral genomes are shorter, we find that the variant caller does not consume much compute resources. Furthermore, the variant caller is not on the critical path for using Read Until, as it is not required for classifying reads.

We find that even a 250W Titan GPU has barely enough basecalling throughput (with low accuracy Guppy-lite) to keep up with a MinION's maximum sequencing throughput. An edge GPU (e.g., Jetson Xavier's) is several times slower than that, and therefore it cannot process all the sequenced reads in real-time to exploit the latency sensitive Read Until feature.

Sequencing throughput, however, continues to grow, as shown in Figure 6. Oxford Nanopore Technologies (ONT)'s GridION is only slightly larger, but has 5× the sequencing throughput of a MinION. ONT announced in 2019 that they are working with MinION prototypes that provide 16× sequencing throughput of MinION devices available in the market today. Within the next few years, they plan to release a production flowcell with 100× greater throughput [19].

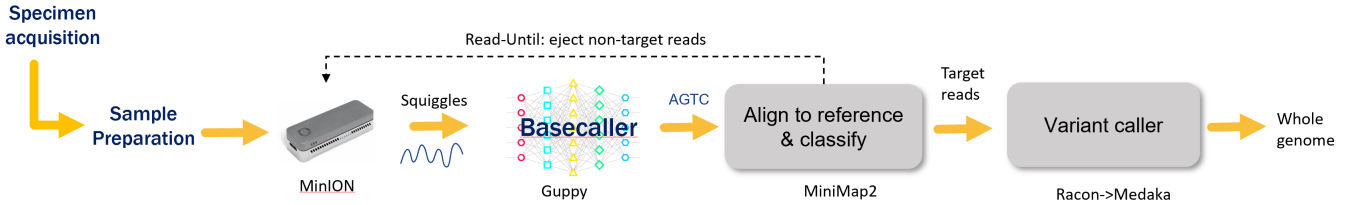

Figure 4: A Read Until pipeline for targeted reference-guided assembly of a virus genome.

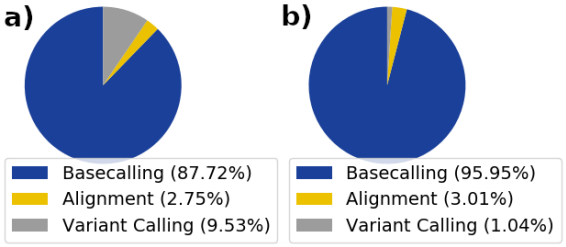

Figure 5: Basecalling is the bottleneck in a Read Until assembly of a SARS-CoV2 genome from specimens with a) 1%, and b) 0.1% viral reads.

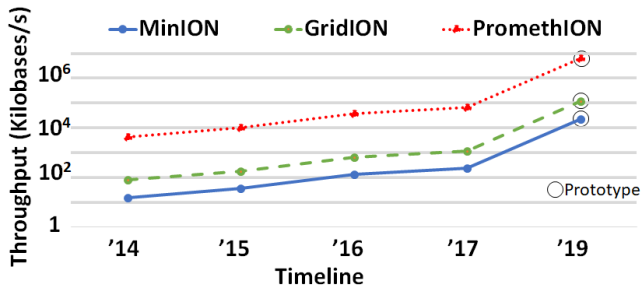

Figure 6: Sequencing throughput is increasing exponentially [48].

Currently, the MinION does not have any on-board compute capability. Our goal is to map all the secondary compute analysis onto an edge system-on-chip so that it can be integrated with the MinION. We address this growing computing need with our small, low-power accelerated SquiggleFilter, which greatly reduces the basecalling and alignment computation required for non-target reads.

4 SQUIGGLEFILTER: A SQUIGGLE-LEVEL TARGETED FILTER USING DYNAMIC TIME WARPING

As discussed in Section 3, classifying a read being sequenced by analyzing its short prefix as target or not, in real-time, is the compute bottleneck. Additionally, basecalling for this classification consumes the most compute time.

Instead of using a basecaller (DNNs) and MiniMap2 aligner to classify a read's prefix, we discuss SquiggleFilter's algorithm that directly aligns each read's electrical signals (query) to the target viral genome's precomputed electrical signal (reference). As a majority of the reads are non-targets, we reduce latency and save much of the work done in basecalling and aligning these non-target reads.

SquiggleFilter aligns the query squiggle with a precomputed reference squiggle of the viral genome using a variant of the dynamic time warping (DTW) algorithm [29]. Recent work has eschewed sDTW due to it's $\Theta(NM)$ complexity [22, 35, 45, 46], but we demonstrate that since both queries (read prefixes) and virus genomes are short, it is a practical solution for viral read enrichment. We further demonstrate its effectiveness on real sequencing data for a SARS-CoV-2 specimen.

Finally, we propose multi-stage sDTW filtering to improve efficiency, and discuss several improvements to conventional sDTW that help realize an efficient hardware accelerator.

4.1 Constructing the Reference Squiggle

In order to align raw signals to a reference genome, the known sequence of bases must first be converted to an expected current profile [37, 38, 51]. As a strand of DNA passes through a nanopore, the current measured is affected by 5-6 adjacent bases simultaneously. A lookup table is provided by ONT which contains the expected current (in pA) for every possible combination of six bases ("6-mer") [53]. This conversion is demonstrated in Figure 7, after which the expected signal is normalized using the mean and standard deviation.

6-mer	current
AAAAAA	86.486
AAAAAC	83.949
AAAAAG	85.475
AAAAAT	84.424
AAAACA	77.097

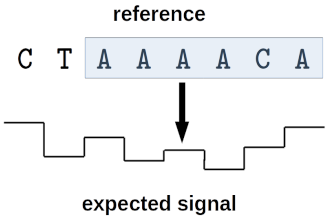

Figure 7: Aligning reference bases to expected currents.

4.2 Normalizing Query Squiggles

Figure 8a shows a contrived minimal example of multiple raw nanopore signals corresponding to the same sequence of bases. Due to a variable rate of DNA/RNA translocation through the nanopore, these signals are out-of-sync (transitions between current levels do not occur simultaneously). Using Dynamic Time Warping (discussed next) solves this issue, and signals are aligned to the expected signal profile (shown in red in Figure 8b). Slight differences in applied bias voltages at each nanopore cause the measured currents to differ significantly, which is why normalization within each read is additionally helpful (Figure 8c).

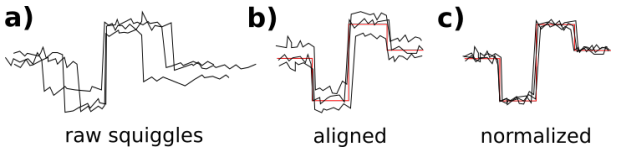

Figure 8: a) Three raw current measurements ("squiggles") for the same sequence of bases. We then show squiggles aligned to the expected signal b) without, and c) with normalization.

4.3 Subsequence Dynamic Time Warping

Dynamic Time Warping (DTW) is a dynamic programming algorithm which is commonly used to align out-of-sync signals [18, 32]. Our filter applies subsequence DTW (sDTW), a slight modification of standard DTW which allows the entire query signal to align to any small portion of the reference, rather than forcing end-to-end alignment of both sequences.

The original sDTW algorithm works as follows for subsequence query Q of length N, reference sequence R of length M, and scoring matrix S:

```
def sDTW(Q,R):
  S = zeros(N,M)
  S[0,0] = (Q[0]-R[0])^2
  for i in range(1,N):
    S[i,0] = S[i-1,0] + (Q[i]-R[0])^2
  for i in range(1,N):
    for j in range(1,M):
      S[i,j] = (Q[i]-R[j])^2 + min(
        S[i-1,j-1], S[i,j-1], S[i-1,j])
  return min(S[N,:])
```

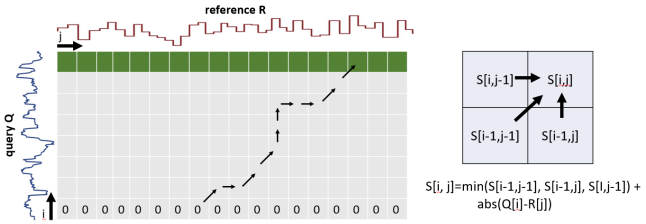

Figure 9: Dynamic time warping algorithm.

The above algorithm dynamically computes all possible alignments of the query Q to reference R (keeping only the best ones) while allowing arbitrary many-to-one or one-to-many mappings between the two signal profiles. It is illustrated in Figure 9. Matrix S records a running tally of the net squared differences between the two signals (using the best alignment of $Q[0:i]$). At the end, $S[N, j]$ (highlighted top row in Figure 9) contains the alignment cost of Q to a subsequence of the reference $R[x:j]$, where x is the start of the best alignment ending at j. The minimum value in this row corresponds to the least squared difference in signal between alignments of the signal to the reference, and thus the cost of the optimal alignment.

4.4 sDTW for Virus Detection

The majority of viruses which are responsible for human epidemics have relatively small single-stranded RNA genomes [39], as is demonstrated in Figure 10. The two notable exceptions are *smallpox*

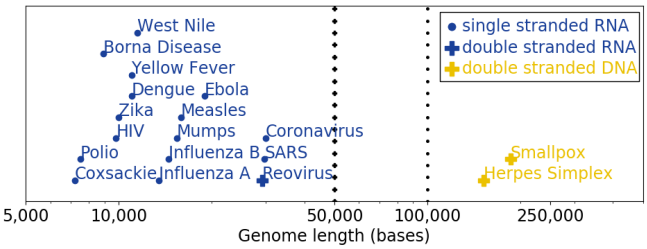

Figure 10: Epidemic virus genome lengths.

and *herpes simplex*, which have larger and more chemically stable double-stranded DNA genomes. Because most viruses have small genomes, we design our filter to operate on viruses with single-stranded genomes of length less than 100,000 bases. Equivalently, the filter works on viruses with double-stranded genomes less than 50,000 bases long. At such short reference genome lengths, it is computationally feasible to compare reads to the entire reference genome for filtering. This would not be a feasible solution for complex organisms such as humans, with genomes approximately 3 billion base pairs long.

4.5 sDTW is an Effective Filter

We seek to design a solution that is capable of detecting all strains of a particular viral species. It is therefore important that our filter is tolerant to variants in the sequenced genome relative to the reference genome used by our filter. We found that reference-guided filtering can be accurate regardless of viral strain, since the number of mutations between different strains is low. Table 2 presents the number of single base mutations between an assembled virus genome for several known SARS-CoV-2 strains, relative to the original Wuhan reference assembly [63]. No insertions or deletions were observed. Strains were defined using NextStrain's [27] classification of all sequenced SARS-CoV-2 genomes into groups of shared ancestors, or "clades", and data was sourced from the GISAID database [50].

Clade	Mut.	GISAID ID	Lab of Origin	Country
19A	23	593737	SE Area Lab Services	Australia
19B	18	614393	Bouake CHU Lab	Ivory Coast
20A	22	644615	Dept. Clinical Microbiology	Belgium
20B	17	602902	NHLS-IALCH	South Africa
20C	17	582807	Public Health Agency	Sweden

Table 2: There are few mutations between SARS-CoV-2 strains, relative to the Wuhan reference genome.

Since there are only a handful of mutations between various SARS-CoV-2 strains, the final sDTW alignment cost will not be significantly impacted. This cost is used to determine whether a given read aligns to the viral reference genome by comparing it to a

constant threshold. If the alignment cost exceeds the chosen threshold, then the squiggle did not match well with any subsequence of the reference genome's expected current profile, and the read can be discarded. Figure 11 shows that a static threshold can be used to distinguish between viral and human DNA fragments (discarding reads above the threshold and keeping reads below the threshold) even when only a few thousand signals have been captured. Due to the slight overlap in final alignment costs, some reads will be incorrectly classified when using a static threshold.

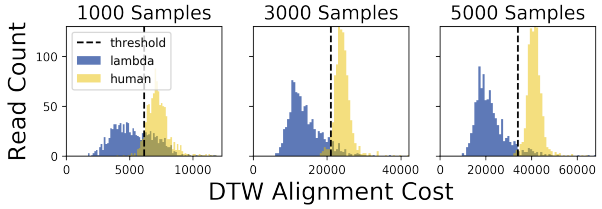

Figure 11: sDTW cost distributions for reads of 3 prefix lengths, aligned to the lambda phage genome.

4.6 Multi-stage sDTW Filtering

We observed that as a read's sequenced prefix length increases, the sDTW alignment cost is more accurately able to distinguish between target and non-target DNA (there is a decrease in overlap between cost distributions in Figure 11). However, waiting to make a Read Until decision increases the proportion of non-target DNA sequenced.

Therefore, instead of a single-stage filter that chooses a constant read length and threshold, we can filter in multiple stages. The first stage examines a shorter read length (e.g. 1000 samples), but chooses a less aggressive threshold that may let many non-target reads through. Non-target reads filtered and ejected using Read Until at this stage would be very short. If a read is retained, it is sequenced further. The second stage then examines the longer read prefix (e.g. 5000 samples), and filters using a more aggressive threshold. Intermediate results can be stored to avoid recomputation. In this way, several stages enable the classifier to filter a majority of non-target reads after seeing only a short prefix. Only reads with initial low-confidence are sequenced more before a decision is made. We have designed our hardware accelerator with this (optional) capability.

4.7 sDTW Algorithm Improvements

We propose several modifications to sDTW which help improve either our accelerator's efficiency or accuracy of non-target read filtering.

Absolute Difference: We reduce hardware area and avoid multiplication by using `abs(Q[i]-R[i])` as our distance metric instead of `(Q[i]-R[j])`^{2}.

Integer Normalization: Our solution uses 8-bit fixed point arithmetic during normalization, with no significant impact to classification accuracy (see Figure 18).

No Reference Deletions: Since the MinION averages 10 samples per base pair, it is unnecessary during sDTW computation for a single squiggle value to be able to align to multiple bases. We removed the possibility of reference deletions entirely from our dynamic programming computation, so that `S[i,j] = abs(Q[i]-R[j]) + min(S[i-1,j-1], S[i-1,j])`.

Match Bonus: This final modification improves filtering accuracy. We found that reads with higher average translocation rates generally have higher alignment costs. To ensure sDTW alignment costs solely represent quality of alignment and are independent of translocation rate, we implemented a "match bonus" that rewards reads for matching additional reference bases, reducing the alignment cost for each matching base by a constant (10) scaled by the number of signals aligned to the previous reference base (thresholded to 10).

4.8 Need for an Accelerator

Despite the reduction in computation when compared to basecalling, sDTW alignment is still too slow to run on commodity hardware. sDTW alignment does avoid expensive floating point operations, instead requiring 8-bit integer comparisons and additions/subtractions. sDTW also has a smaller memory footprint (60,000 reference bases) compared to Guppy-lite (284,000 weights) when filtering SARS-CoV-2. Despite memory and operation complexity advantages, however, the number of operations required for sDTW (1,400 million) is greater than that of Guppy-lite (141 million). This is still more efficient than Guppy (2,412 million). In order to meet current and future MinION device requirements for Read Until, it is necessary to design an accelerator.

5 ACCELERATED SQUIGGLEFILTER

We present a System-on-Chip for reference-guided assembly of target viruses, shown in Figure 12. Its capabilities are similar to a Nvidia Jetson TX2, except for our SquiggleFilter accelerator. Our SquiggleFilter accelerator classifies and filters non-target reads, which constitute >99% of all reads in most biological specimens. Thus, a large fraction of computing identified in Section 3 is handled by our SquiggleFilter accelerator. Furthermore, our accelerator enables low latency read classification, allowing us to use Read Until to eject non-target reads after sequencing only a short prefix.

Target reads (and any false positives) are processed off of Read Until's critical path. Only these small fraction of reads need to be basecalled, aligned, and variant called. We find that we can perform these computations on an edge GPU (basecaller) and ARM processor (aligner and variant caller), and still construct the whole viral genome in approximately 10 minutes. Unfiltered non-target reads (false positives due to sDTW algorithm) will fail to align to the viral reference genome after basecalling, and so they will be discarded without affecting the accuracy of conventional reference-guided assembly. The final assembled genome and raw sequencing data is written to a 32GB eMMC 5.1 flash memory, which is sufficient to store one day's worth of sequencing data.

We now present the 1D systolic array based SquiggleFilter accelerator for our squiggle-level classification algorithm discussed in Section 4. It can be programmed to target any novel viral genomes less than 100K bases. It supports variable query length. That is, it can classify read prefixes of different lengths, and thereby supports multi-stage filtering. The size of the systolic arrays and buffers are derived from our analysis of real-world metagenomic data.

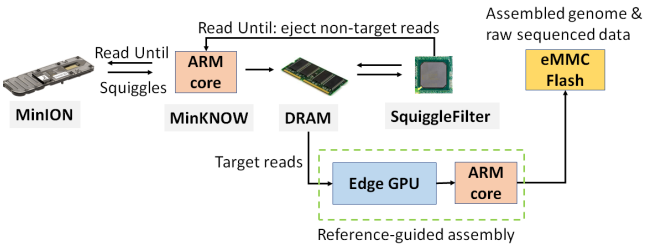

Figure 12: System-on-Chip design with the accelerated hardware filter on ASIC integrated with NVIDIA GPU and 8-core ARM v8.2 64-bit CPU

5.1 SquiggleFilter Design

SquiggleFilter consists of 5 independent tiles (one tile is shown in Figure 13). Each can be individually power-gated based on desired filtering throughput. This number was chosen to meet the expected 100× future increase in sequencing throughput. Each read is assigned to an available tile for classification. As a read is sequenced, squiggles from a MinION R9.4.1 flow cell are streamed into DRAM in real-time. From there, squiggles are fetched into a tile's query buffers. Two ping-pong query buffers enable simultaneous squiggle loading and normalization. Once the desired length of read prefix has been sequenced, the raw squiggles of a query are normalized and then stored across the processing elements connected in a 1D systolic array.

Each tile also stores a copy of the precomputed reference signal (loaded from flash during an initialization phase) in a reference buffer. The reference samples are then streamed into the systolic array. The entire sDTW matrix is computed in a wavefront parallel manner as described in Section 4.7. The final PE determines the final minimum alignment cost, and sends a control signal to the MinION to eject the read if the final cost exceeds a predetermined threshold. Non-ejected reads are sequenced in full and stored in memory.

The number of cycles required to classify a new read is the read prefix length (2000 samples) plus the reference genome length (60,000 samples for SARS-CoV-2).

Reference Buffer: We chose to use a separate buffer (100 KB) for each tile, even though all the reference buffers across the tiles store the same information (viral genome's reference squiggles). This allows us to reduce access latency and provide sustained throughput to each tile with just one read port. The area cost of duplicating the references is negligible, as reference buffers constitute only 6.98% of total tile area.

Furthermore, our design is independent of reference length and limited only by the reference buffer size provisioned. By loading a new precomputed reference signal onto the on-board flash, SquiggleFilter can easily be reprogrammed to detect a novel virus.

Variable Query Length: As discussed in Section 4.6, there exists a trade off between classification accuracy and sequencing length of queries. We find (Section 7.4) that read prefix length of 2000 samples yields the most savings using Read Until, when we use a single threshold. Therefore, we use a 1D systolic array of size 2000 PEs.

Our SquiggleFilter design can handle variable read prefix lengths that are multiples of 2000 squiggle samples. To support query lengths longer than 2000 samples and multi-stage filtering, we configure the last PE such that it can optionally write the sDTW costs every cycle to DRAM. This consumes significant memory bandwidth. However, it enables sDTW computation to continue if greater classification accuracy by analyzing a longer prefix is desired. These intermediate costs are then loaded from DRAM and used to initialize the PEs (similar to initial normalized query) prior to computing the costs for a 4000-sample prefix length.

5.2 Processing Element

Each PE computes a cell in the sDTW matrix every cycle, using the final algorithm described in Section 4.7. At cycle c, each PE (Figure 14) checks for the minimum among its previous neighbor's $c-1$ and $c-2$ cycle's outputs, modified by a bonus which rewards matching new reference bases. This minimum is then added to the absolute difference of the current query and reference values. Each PE stores the resulting costs and bonuses from its last two cycles for the next PE. Additionally, the last PE contains logic to compare its cost to a predefined threshold which determines whether or not to eject the read. This threshold can be reprogrammed on the SquiggleFilter based on software analysis of the target strain, but we have found it to be relatively robust across species and sequencing runs. Each PE is $1203\mu m^2$ and requires 1.92mW when synthesized for a 28nm TSMC chip.

5.3 Normalizer

Normalization rescales the raw signals in order to improve classification accuracy when performing sDTW [47], as discussed in Section 4.2. The normalizer, shown in Figure 15, is a query preprocessor which streams in 10-bit samples from the query buffer for accumulation. After every $n = 2000$ samples, the normalizer updates the mean and Mean Absolute Deviation (MAD), defined as follows:

$$\text{mean} = \bar{x} = \sum_{i=1}^{n} \frac{x_i}{n} \qquad MAD = \sum_{i=1}^{n} \frac{|x_i - \bar{x}|}{n}$$

Thereafter, the streamed-in samples are transformed with mean-MAD normalization. The output normalized value is filtered for outliers and then re-scaled to a reduced precision 8-bit integer which is then fed to the tiles for sDTW classification. We find that 8 bits of precision is sufficient for accurate classification (Figure 18). For efficiency, we do not convert the ADC sample to floating point, but instead use fixed-point values in the range $[-4, 4]$.

6 METHODOLOGY

Human DNA datasets containing MinION R9.4 and R9.4.1 flow cells were obtained from the Nanopore Whole-Genome Sequencing Consortium [60] and the ONT Open Datasets [54]. The SARS-CoV-2 dataset contains raw MinION R9.4.1 data available from the Cadde Centre [3]. We sequenced lambda phage DNA in our own laboratory using the ONT Rapid Library Preparation Kit [14] following the Lambda Control protocol with a MinION R9.4.1 flow cell.

We performed basecaller profiling measurements using a Titan XP GPU (server class) and Jetson Xavier GPU (edge

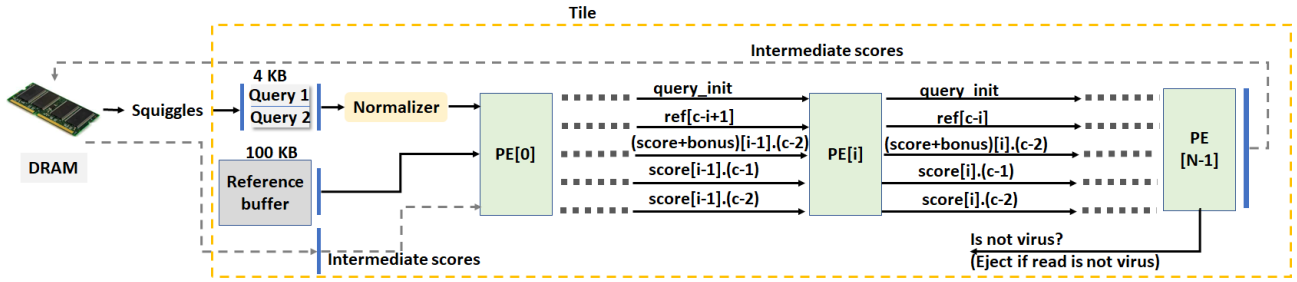

Figure 13: SquiggleFilter Tile. N=2000 PEs are connected with streaming inputs and outputs. The last PE determines the classification by comparing its cost to a threshold every cycle. c is the cycle and i is the PE index.

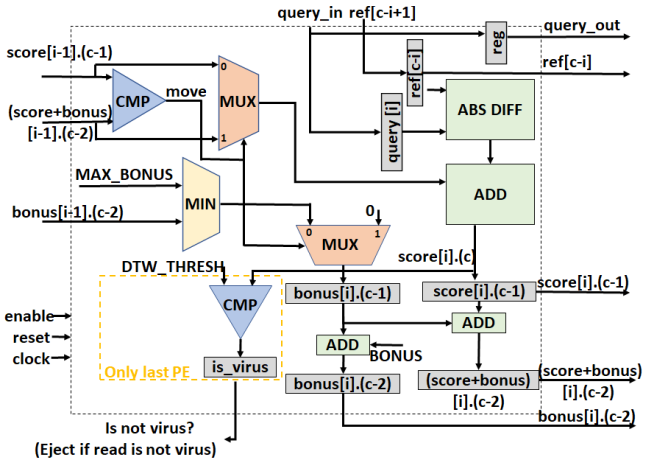

Figure 14: SquiggleFilter Processing Element.

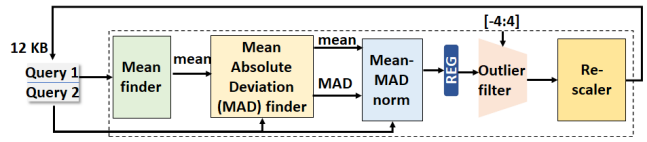

Figure 15: SquiggleFilter Normalizer.

class). Their specifications are provided in Table 3. We evaluated both Guppy (`dna_r9.4.1_450bps_hac.cfg`) and Guppy-lite (`dna_r9.4.1_450bps_fast.cfg`) without modification using Guppy version 4.2.2 [59]. MiniMap2 version 2.17-r954-dirty [36] aligned basecalled reads.

First, we measured the basecalling throughput of Guppy and Guppy-lite on a dataset of 33,004 full-length reads. Next, we used the proprietary Python libraries `ont-fast5-api` version 3.1.6 [12] and `ont-pyguppy-client-lib` 4.2.2 [13] to basecall the same reads in chunks of 2000 signals, thereby simulating Read Until on the same dataset. The Python code was instrumented to record latency information, and we tuned the number of reads simultaneously in-flight to optimize performance. This online Read Until processing (due to smaller batch size) resulted in 4.05× lower throughput for Guppy-lite and 2.85× lower throughput for Guppy on the Titan XP. Using these measurements and the relative peak throughputs of the Jetson and Titan, the Read Until performance of the Jetson Xavier was estimated (necessitated by the unavailability of `ont-pyguppy-client` ARM binaries for fine-grained Read Until control on the Jetson).

	Edge GPU	Edge CPU	GPU	CPU
Model	Jetson AGX Xavier	ARMv8.2	Titan XP	2× Intel Xeon E5-2697v3
Cores	512 Volta	8	3840 Pascal	56
Clock	1377MHz	2265MHz	1582MHz	2600MHz

Table 3: Architectural specifications of evaluated GPUs.

A memory-efficient multi-threaded implementation of sDTW was written in Python for accuracy analysis, and tested on 1000 reads from each of the datasets mentioned above. In order to determine the relative benefits of Read Until using different classification latencies and accuracies, we developed an analytical model to estimate sequencing runtime. This model accounts for factors such as average read length, desired coverage of the reference genome, average DNA capture time, and the Read Until parameters mentioned previously.

The design was first functionally verified via emulation on Amazon Web Service's EC2 F1 instance, which uses a 16nm Xilinx UltraScale+ VU9P FPGA. Further, SquiggleFilter was synthesized using the Synopsys Design compiler for 28nm TSMC HPC and the design is clocked at 2.5GHz. 32GB 256-Bit LPDDR4x is connected to the System-on-Chip along with an 8-core ARM v8.2 64-bit CPU.

7 RESULTS

7.1 SquiggleFilter Hardware Synthesis

ASIC Element	Area (mm^2)	Power (W)
Normalizer	0.014	0.045
Processing Element	0.001	0.002
Tile (1×2000 PEs)	2.423	2.780
Query buffer	0.023	0.009
Reference buffer	0.185	0.028
Complete 1-Tile ASIC	2.65	2.86
Complete 5-Tile ASIC	13.25	14.31

Table 4: SquiggleFilter ASIC synthesis results.

Table 4 shows SquiggleFilter synthesized to a 13.25mm^2 ASIC that consumes 14.21W when performing single-stage filtering and

clocks at 2.5GHz. It contains 5 fully-independent tiles (which could be individually power-gated to improve energy efficiency). The latency for classifying a 2000-sample read from SARS-CoV-2 is 0.027ms, and for lambda phage is 0.043ms, due to its longer reference genome. This adds insignificant latency to each Read Until decision's critical path, since it takes around 500ms to sequence a sufficient number of bases to make an accurate decision. The single-tile classification throughputs for SARS-CoV-2 and lambda phage are 74.63M samples/s and 46.73M samples/s respectively, which are both considerably higher than MinION's current maximum output of 2.05M samples/sec). Additionally, if each tile is configured to perform multi-stage filtering, it will write intermediate results to DRAM, consuming only 10 GB/s main memory bandwidth per tile. Since Jetson Xavier's main memory supports 137 GB/s, our 5 tile design is feasible.

7.2 Performance Analysis

Latency: Figure 16a compares GPU-based basecalling latency to our SquiggleFilter accelerator's latency. Note that we show only basecalling latency as it is the most time consuming step (96% of compute time) of the virus classification pipeline. The measurements demonstrate that it would be impractical to use the high-accuracy Guppy basecaller as its latency is greater than one second, in which time more than 400 bases would have been unnecessarily sequenced for non-target reads. We found that Guppy-lite provides sufficient accuracy for Read Until classification as downstream aligner MiniMap2 is able to account for incorrect basecalls when aligning reads. However, a 149ms basecalling latency for Guppy-lite translates to an additional 60 bases sequenced for each read during classification. Since most non-target reads can be discarded after around 200 bases, this overhead is significant. In comparison, the common-case 0.04ms decision latency of SquiggleFilter ensures that not even a single base pair is unnecessarily sequenced.

Throughput: Figure 16b compares the basecalling throughput of Guppy-lite measured over GPU configurations to SquiggleFilter accelerator's classification throughput. An edge GPU such as the Jetson does not have sufficient compute power to basecall data from all pores in real-time and keep up with the maximum sequencing throughput of the MinION. We calculated that the Jetson's throughput would be approximately 95,700 bases per second, which is only 41.5% of the MinION's maximum output of 230,400 bases per second. In the worst case, Read Until can only be performed using 41.5% of the MinION's pores when basecalling using Guppy-lite on the Jetson. The remaining 59.5% of pores are unable to use Read Until, and will sequence full-length human reads. In contrast, SquiggleFilter's throughput far exceeds MinION's and GridION's sequencing throughputs.

7.3 sDTW Algorithm Accuracy

Figure 17a compares sDTW accuracy to basecalling and alignment on a dataset of 1000 lambda phage and 1000 human reads, with a line plotted for each prefix length. The MiniMap2 alignment quality and sDTW alignment cost thresholds (for determining which reads to sequence and which to reverse) are swept through the range of possible values to show threshold-dependent accuracies. Although the Read Until accuracy obtained by basecalling and aligning slightly outperforms sDTW, this is to be expected since alignment algorithms such as MiniMap2 use numerous scoring heuristics and have matured significantly over the past two decades [36].

Figure 18 shows the maximal F-score for all of our algorithm modifications and standard sDTW on the same dataset. As expected, accuracy generally increases along with sample prefix length. We found that using both integer normalization and absolute difference for our distance metric reduce filtering accuracy slightly, a compromise which was expected. Eliminating reference deletions results in a slight accuracy improvement. Combining all three of these optimizations results in the lowest accuracy (but most efficient) of all configurations tested. We find that by including our "match bonus", we can recover lost accuracy and outperform the baseline, with a minor performance penalty. Figure 19 furthermore demonstrates that there is no a significant loss in filter accuracy until there is more than a 1,000 base difference between the reference genome and viral strain sequenced.

7.4 Benefits of Read Until

Read Until not only saves sequencing time, but also cost. Figure 20 shows our wet-lab experiment. After sequencing for a while, washing the flow cell with nuclease and re-multiplexing (rapid alternations of pore voltage bias direction, shown with dotted black line) leads to control and Read Until pores having the same number of active channels. This means that Read Until does not damage the flow cell any more than normal sequencing, but enables more experiments to be run over the lifetime of any flow cell.

The single-threshold Read Until design space was first explored for our lambda phage dataset. Figure 17a shows the accuracy of SquiggleFilter for a variety of Read Until prefix lengths (each line), and for all reasonable sDTW alignment cost thresholds (points on each line). Given this experimentally measured accuracy, the total expected sequencing time to perform Read Until for lambda phage was calculated using our analytical model, and is shown in Figure 17b. We found that the best single-threshold configuration for SquiggleFilter outperforms Guppy-lite on this dataset by 12.9% in terms of Read Until runtime. By using multiple thresholds, we can reduce runtime by a further 13.3%.

A similar analysis was then performed for the SARS-CoV-2 dataset, and the results are shown in Figure 17c. Optimal sDTW alignment cost thresholds were taken from the Read Until runtime minima from Figure 17b, and the corresponding Read Until runtimes using those thresholds are marked for the SARS-CoV-2 dataset.

7.5 Looking Forwards: Scalability

Sequencing throughput is expected to increase by $10 - 100\times$ within the next few years, due to new nanopore chemistry enabling a denser configuration with many more channels per flow cell [19]. Figure 21 shows that without further improvements to basecalling throughput, current GPUs will be unable to keep pace with new sequencing technology. As a result, the time and cost savings gained through Read Until will be largely lost. We can see that Guppy-lite's slight edge over SquiggleFilter in terms of accuracy has already been lost due to its inability to perform Read Until on 512 pores. In

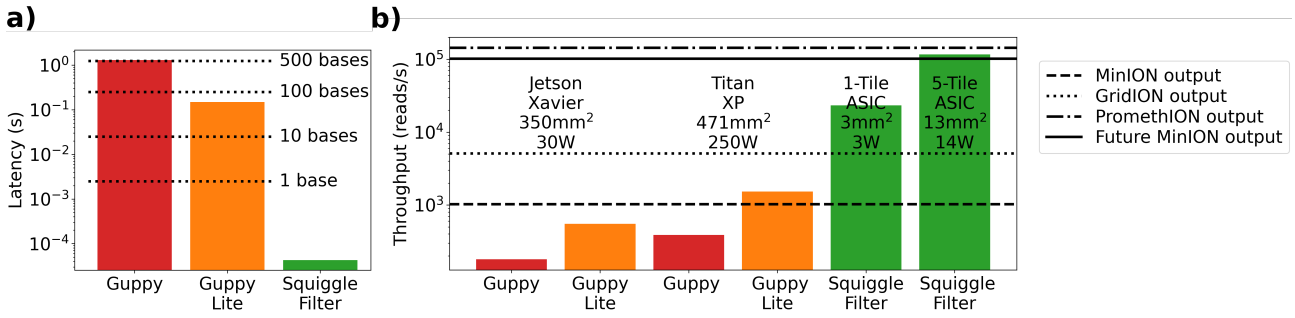

Figure 16: a) Latency, and b) throughput of Guppy, Guppy-lite and SquiggleFilter during Read Until.

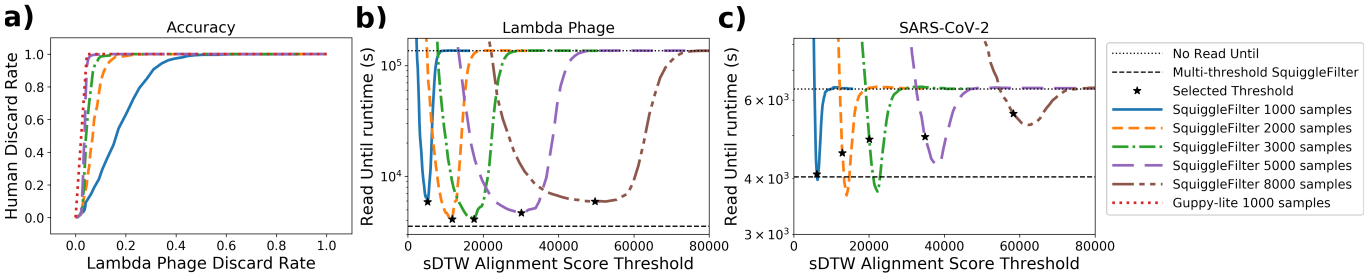

Figure 17: SquiggleFilter Read Until a) accuracy, and performance on b) lambda phage and c) SARS-CoV-2 datasets.

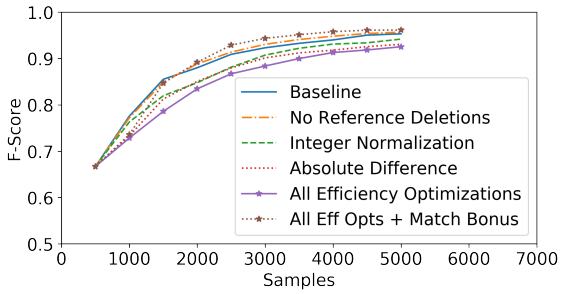

Figure 18: Accuracy results for modifications to the standard sDTW algorithm.

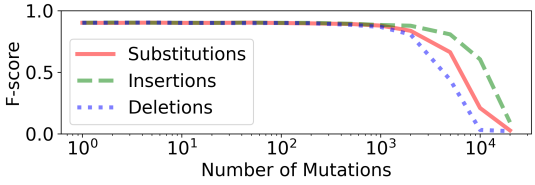

Figure 19: SquiggleFilter accuracy is robust against random (lambda phage) reference mutations.

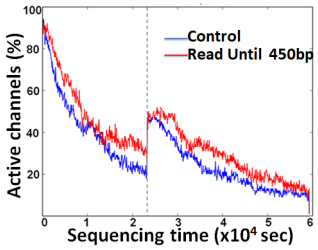

Figure 20: Time saved is cost saved for sequencing.

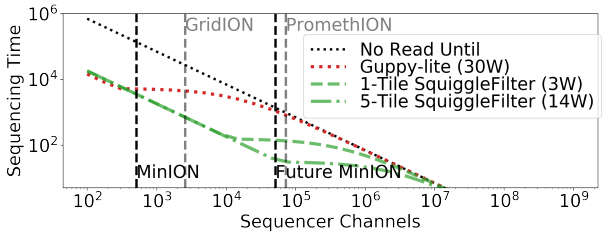

Figure 21: Future SquiggleFilter Read Until benefits.

contrast, our SquiggleFilter accelerator can tolerate a 114× increase in sequencing throughput.

8 RELATED WORK

The MinION was released in 2014 as the first commercially available nanopore-based DNA/RNA sequencing device [10]. The first Read Until software pipeline was developed two years later, in 2016 [38]. In this seminal work, raw nanopore signal was first segmented into events, and then events were aligned to a lambda phage reference using subsequence Dynamic Time Warping (described in Section 4.3). Event segmentation is used to detect the most likely positions in the raw signal where a new base has entered the pore, and could be considered a rudimentary form of basecalling. In fact, it has been used as an essential preprocessing step in several older basecallers [59]. Unfortunately, the throughput measured by this original work on an 8-core ARM processor is 40× lower than the current maximum MinION output.

As basecalling throughput and accuracy has gradually increased over the last few years, the standard approach for Read Until pipelines has been to basecall the signal and use an aligner to determine if each read aligns to the target genome [15, 22, 45, 46]. This method achieves the highest accuracy, but is not scalable. When

pairing a server-class GPU with a handheld MinION device, it is just able to perform Read Until with the required throughput, albeit with significant latency (as shown in Section 7.2).

UNCALLED, a more recent work, skips basecalling by doing approximate alignments in 3 steps: event segmentation, FM-index look-ups, and seed clustering [35]. However, we evaluated UNCALLED and observed that it requires longer prefix lengths for accurate alignment. 23.63% of 2000-sample long chunks from our lambda phage dataset were not alignable. After segmentation, UNCALLED uses an FM-index to filter reads. UNCALLED aligns only ~76% of the lambda reads of 2000 samples on a modern Intel i7-7700 desktop processor taking 16ms per read. Moreover, ~14% of reads take 353ms per read to be aligned as more samples are required for a decision. ~10% of the reads, however, are left unaligned. On an edge device with an ARM core and lower memory bandwidth, performance would be worse. No existing software-only solution has adequate throughput and low enough latency to effectively perform Read Until on an edge device.

In contrast, our approach shifts to a minimalistic sDTW alignment algorithm, and by designing hardware to accelerate the simple and regular sDTW computation, we can easily meet the desired throughput and latency requirements on an edge device. General purpose DTW accelerators have already been designed to solve alignment problems in other domains such as audio signal processing [52] and astronomy [47], but nanopore viral DNA/RNA filtering required several application-specific optimizations to meet the desired latency, throughput and accuracy requirements. Our design involves several algorithmic modifications to vanilla sDTW (described in Section 4.7), uses an on-chip buffer for efficient repeated alignments to the same reference, replaces all floating-point computation with integer arithmetic for increased efficiency, uses multi-stage filtering for optimal Read Until results, and has been evaluated on a novel virus (SARS-CoV-2).

There has recently been significant work on designing hard-ware accelerators for genomics applications [20, 23, 24, 28, 33, 41, 55, 61], but these accelerators focus on human genome sequencing. As a result, they efficiently align many (usually short) basecalled reads to a long reference genome with high throughput and accuracy. As noted previously in Section 3.2, our problem has very different computational needs. We must selectively filter short noisy raw signals (squiggles) with sufficiently high throughput and low latency to effectively exploit Read Until. We achieve this by replacing the basecaller and aligner with SquiggleFilter.

9 CONCLUSION

In designing a universal virus detector, we identify the basecaller to be a significant bottleneck in filtering non-target reads. This compute problem is only going to get worse, as the throughput of nanopore sequencers is expected to increase by 10-100× in the near future. We address this problem using hardware-accelerated SquiggleFilter for filtering non-target reads without basecalling them. We show that our 14.3W 13.25mm^2 accelerator has 274× greater throughput and 3481× lower latency than existing approaches while consuming half the power, enabling Read Until for the next generation of nanopore sequencers.

ACKNOWLEDGMENTS

We would like to thank Robert Dickson and John Erb-Downward for their valuable pathology-related insights and borrowed use of their Jetson AGX Xavier. We would additionally like to thank Jenna Wiens, Piyush Ranjan, Arun Subramaniyan, and Yichen Gu for their input and feedback at various stages of this project.

A ARTIFACT APPENDIX

A.1 Abstract

Our artifact contains the RTL and testbench SystemVerilog code for our SquiggleFilter accelerator in the `design/` subdirectory. Additionally, `sdtw_analysis.ipynb` is a full Jupyter Notebook pipeline containing our software sDTW algorithm implementation and our Read Until runtime model, along with scripts for generating multiple figures from our paper.

A.2 Artifact check-list (meta-information)

- **Algorithm:** Hardware and software implementation of custom subsequence Dynamic Time Warping (sDTW) algorithm for filtering non-viral DNA reads in real time.
- **Program:** RTL and SystemVerilog testbench code for SquiggleFilter accelerator. Jupyter Notebook containing Python sDTW implementation and runtime model.
- **Data set:** Raw human, lambda phage, and SARS-CoV-2 FAST5 data from several public sources [3, 54].
- **Run-time environment:** Vivado 2019.1 and Jupyter Notebook. Build instructions targeted to Ubuntu 18.
- **Hardware:** At least one CPU core and 10GB RAM for the notebook. Recommended requirements for Xilinx Vivado based on Xilinx SDK: min 2.2GHz, Intel Pentium 4, Intel Core Duo, or Xeon Processors; SSE2 minimum.
- **Output:** Software regeneration of multiple figures from the paper. Verification of hardware using SystemVerilog testbench.
- **How much disk space required (approximately)?:** 40GB public dataset download. 40GB for public dataset download. Xilinx Vivado requires upto 30GB of diskspace for installation and an additional 2.5GB if Vivado simulation is started.
- **How much time is needed to complete experiments (approximately)?:** Jupyter Notebook requires 10 minutes with 56 cores. Vivado simulation on the SARS-CoV-2 reads can take 1-21 minutes on a Quadcore 8th Gen i5 with 8GB RAM depending on the number of test-cases anyone may wish to run.
- **Publicly available?:** Yes.
- **Archived (provide DOI)?:** https://doi.org/10.5281/zenodo.5150973

A.3 Description

A.3.1 How to access. All of the source code is open source, and can be obtained either through GitHub[1] or Zenodo[2].

A.3.2 Hardware dependencies. The SquiggleFilter code requires approximately 10GB of RAM, and the datasets used require approximately 40GB of disk space. Xilinx Vivado comes with the following additional requirements on the processor: minimum 2.2GHz, Intel Pentium 4, Intel Core Duo, or Xeon Processors; SSE2 minimum.

A.3.3 Software dependencies. Any Linux OS can be used, but a recent Ubuntu release is recommended for ease of installation. The Jupyter Notebook has multiple Python package dependencies, which will be installed by the `setup.sh` script. For hardware evaluation, a recent installation of the licensed Vivado Design Suite is recommended; we used release 2019.1. Further details on the installation can be found on https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html.

A.3.4 Data sets. Our artifact uses three raw nanopore signal (FAST5) datasets:

- **lambda**: This dataset of 21,000 lambda phage reads was generated in our laboratory, and is included in our GitHub repository at `data/lambda/fast5`.
- **covid**: This dataset of 1.2 million SARS-CoV-2 reads is downloaded from the CADDE Centre [3] to `data/covid/fast5` by the `setup.sh` script.
- **human**: This dataset of 65,000 huan reads is downloaded from ONT Open Datasets [3] to `data/human/fast5` by the `setup.sh` script.

A.4 Installation

All source code is available in either our GitHub[1] or Zenodo[2] repositories.

- **README.md** contains instructions for evaluating the artifacts
- **design/** contains the SystemVerilog RTL and testbench. `testbench_top.sv` is the top file of the testbench for behavioral simulation. `normalizer_top.v` is the top file for the normalizer and it's sub-modules. `warper_top.sv` is the top file for the systolic array.
- **sdtw_analysis.ipynb** contains our software pipeline, Python sDTW implementation, and runtime model.
- **setup.sh** is the setup script
- **data/** contains all three datasets
- **scripts/** contains all scripts used for data analysis

Please follow all instructions from `README.md` to evaluate the artifacts.

A.5 Evaluation and expected results

A.5.1 Hardware. After installing and running Vivado, go under settings and change the simulation run time to 18ms for complete simulation. On the flow navigator, pressing the run simulation option would start the simulation and messages would start appearing on the tcl console printing whether the testcases passed or failed. We observe and expect all the testcases to pass. Additionally, the waveform may be viewed as the simulation begins. Please find detailed instructions in `README.md`.

A.5.2 Software. After the Jupyter Notebook is running, please select the `sf-venv3` kernel (`Kernel -> Change Kernel`) created by the `setup.sh` script. Then, run all cells in order (`Kernel -> Restart and Run All`). The entire pipeline should run successfully, computing the sDTW scores on the datasets selected and regenerating most of the figures in our paper.

[1] https://github.com/TimD1/SquiggleFilter
[2] https://doi.org/10.5281/zenodo.5150973

A.6 Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-badging
- http://cTuning.org/ae/submission-20201122.html
- http://cTuning.org/ae/reviewing-20201122.html

REFERENCES

[1] 2019-nCoV CDC-qualified Probe and Primer Kits for SARS-CoV-2. LGC Biosearch Technologies. [Online]. Available: https://www.biosearchtech.com/products/pcr-kits-and-reagents/pathogen-detection/2019-ncov-cdc-probe-and-primer-kit-for-sars-cov-2

[2] "ARTIC V3 Update Notes," the ARTIC Network. [Online]. Available: https://artic.network/resources/ncov/ncov-amplicon-v3.pdf

[3] Brazil-uk centre for arbovirus discovery, diagnosis, genomics and epidemiology. [Online]. Available: https://cadde.s3.climb.ac.uk/SP1-raw.tgz

[4] cDNA PCR Sequencing Kit. Oxford Nanopore Technologies. [Online]. Available: https://store.nanoporetech.com/us/sample-prep/cdna-pcr-sequencing-kit.html

[5] Direct cDNA Sequencing Kit. Oxford Nanopore Technologies. [Online]. Available: https://store.nanoporetech.com/us/sample-prep/direct-cdna-sequencing-kit.html

[6] Direct RNA Sequencing Kit. Oxford Nanopore Technologies. [Online]. Available: https://store.nanoporetech.com/us/catalog/product/view/id/297/s/direct-rna-sequencing-kit/category/28/

[7] Jetson agx xavier developer kit. NVIDIA. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

[8] Medaka. Medaka - Medaka 1.2.0 documentation. [Online]. Available: https://nanoporetech.github.io/medaka/

[9] Metagenomic analysis of SARS-CoV-2 respiratory samples via Sequence-Independent Single Primer Amplification (SISPA) and nanopore sequencing. Oxford Nanopore Technologies. [Online]. Available: https://nanoporetech.com/sites/default/files/s3/literature/COVID-19_metagenomic_sequencing.pdf

[10] MinION DNA Sequencer. Oxford Nanopore Technologies. [Online]. Available: https://nanoporetech.com/products/minion

[11] Navica App and BinaxNOW COVID-19 Ag Test Card. Abbott Point of Care Testing. [Online]. Available: https://www.globalpointofcare.abbott/en/product-details/navica-binaxnow-covid-19-us.html

[12] ont-fast5-api. FAST5 API: a simple interface to HDF5 files of the Oxford Nanopore .fast5 file format. [Online]. Available: https://pypi.org/project/ont-fast5-api/

[13] ont-pyguppy-client-lib. PyGuppy: Python bindings for the GuppyClient library. [Online]. Available: https://pypi.org/project/ont-pyguppy-client-lib/

[14] Rapid Library Preparation Kit (SQK-RAD004). Oxford Nanopore Technologies. [Online]. Available: https://store.nanoporetech.com/us/sample-prep/rapid-sequencing-kit.html

[15] Read until api. Oxford Nanopore Technologies. [Online]. Available: https://github.com/nanoporetech/read_until_api

[16] SARS-CoV-2 Rapid Colorimetric LAMP Assay Kit. New England Biolabs. [Online]. Available: https://www.neb.com/products/e2019-sars-cov-2-rapid-colorimetric-lamp-assay-kit

[17] Sequence-Independent, Single-Primer Amplification of RNA viruses V.3. University of Wisconsin-Madison. [Online]. Available: https://www.protocols.io/view/sequence-independent-single-primer-amplification-o-bckxiuxn.html

[18] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370.

[19] C. Brown, "Technology update," 2019, nanopore Community Meeting. [Online]. Available: https://nanoporetech.com/resource-centre/nanopore-community-meeting-2019-technology-update

[20] D. S. Cali, G. S. Kalsi, Z. Bingöl, C. Firtina, L. Subramanian, J. S. Kim, R. Ausavarungnirun, M. Alser, J. Gomez-Luna, A. Boroumand *et al.*, "Genasm: A high-performance, low-power approximate string matching acceleration framework for genome sequence analysis," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 951–966.

[21] H. S. Edwards, R. Krishnakumar, A. Sinha, S. W. Bird, K. D. Patel, and M. S. Bartsch, "Real-time selective sequencing with rubric: Read until with basecall and reference-informed criteria," *Scientific Reports*, vol. 9, no. 1, pp. 1–11, 2019.

[22] ——, "Real-time selective sequencing with rubric: read until with basecall and reference-informed criteria," *Scientific reports*, vol. 9, no. 1, pp. 1–11, 2019.

[23] D. Fujiki, A. Subramaniyan, T. Zhang, Y. Zeng, R. Das, D. Blaauw, and S. Narayanasamy, "Genax: a genome sequencing accelerator," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 69–82.

[24] D. Fujiki, S. Wu, N. Ozog, K. Goliya, D. Blaauw, S. Narayanasamy, and R. Das, "Seedex: A genome sequencing accelerator for optimal alignments in subminimal space," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 937–950.

[25] M. A. GOUILH, R. CASSIER, E. MAILLE, C. Schanen, L.-M. ROCQUE, and A. VABRET, "An easy, reliable and rapid sars-cov2 rt-lamp based test for point-of-care and diagnostic lab," *medRxiv*, 2020.

[26] A. L. Greninger, S. N. Naccache, S. Federman, G. Yu, P. Mbala, V. Bres, D. Stryke, J. Bouquet, S. Somasekar, J. M. Linnen *et al.*, "Rapid metagenomic identification of viral pathogens in clinical samples by real-time nanopore sequencing analysis," *Genome medicine*, vol. 7, no. 1, p. 99, 2015.

[27] J. Hadfield, C. Megill, S. M. Bell, J. Huddleston, B. Potter, C. Callender, P. Sagulenko, T. Bedford, and R. A. Neher, "Nextstrain: real-time tracking of pathogen evolution," *Bioinformatics*, vol. 34, no. 23, pp. 4121–4123, 2018.

[28] T. J. Ham, D. Bruns-Smith, B. Sweeney, Y. Lee, S. H. Seo, U. G. Song, Y. H. Oh, K. Asanovic, J. W. Lee, and L. W. Wills, "Genesis: a hardware acceleration framework for genomic data analysis," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 254–267.

[29] R. Han, Y. Li, X. Gao, and S. Wang, "An accurate and rapid continuous wavelet dynamic time warping algorithm for end-to-end mapping in ultra-long nanopore sequencing," *Bioinformatics*, vol. 34, no. 17, pp. i722–i731, 2018.

[30] J. Hasell, E. Mathieu, D. Beltekian, B. Macdonald, C. Giattino, E. Ortiz-Ospina, M. Roser, and H. Ritchie, "A cross-country database of covid-19 testing," *Scientific data*, vol. 7, no. 1, pp. 1–7, 2020.

[31] P. James, D. Stoddart, E. D. Harrington, J. Beaulaurier, L. Ly, S. Reid, D. J. Turner, and S. Juul, "Lampore: rapid, accurate and highly scalable molecular screening for sars-cov-2 infection, based on nanopore sequencing," *medRxiv*, 2020.

[32] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: a survey and empirical demonstration," vol. 7, no. 4. Springer, 2003, pp. 349–371.

[33] S. K. Khatamifard, Z. Chowdhury, N. Pande, M. Razaviyayn, C. Kim, and U. R. Karpuzcu, "A non-volatile near-memory read mapping accelerator," *arXiv preprint arXiv:1709.02381*, 2017.

[34] D. Kilburn, J. Burke, R. Fedak, H. Olsen, M. Jain, K. Miga, S. Mayes, and K. Liu. High Data Throughput and Low Cost Ultra Long Nanopore Sequencing. [Online]. Available: https://15a13b02-7dac-4315-baa5-b3ced1ea969d.filesusr.com/ugd/5518db_164bac27f4654b1f94d3472f09372498.pdf

[35] S. Kovaka, Y. Fan, B. Ni, W. Timp, and M. C. Schatz, "Targeted nanopore sequencing by real-time mapping of raw electrical signal with uncalled," *BioRxiv*, 2020.

[36] H. Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinformatics*, vol. 34, no. 18, pp. 3094–3100, 2018.

[37] N. J. Loman, J. Quick, and J. T. Simpson, "A complete bacterial genome assembled de novo using only nanopore sequencing data," *Nature methods*, vol. 12, no. 8, pp. 733–735, 2015.

[38] M. Loose, S. Malla, and M. Stout, "Real-time selective sequencing using nanopore technology," *Nature methods*, vol. 13, no. 9, p. 751, 2016.

[39] G. Mahmoudabadi and R. Phillips, "A comprehensive and quantitative exploration of thousands of viral genomes," *Elife*, vol. 7, p. e31955, 2018.

[40] A. J. McMichael, "Environmental and social influences on emerging infectious diseases: past, present and future," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 359, no. 1447, pp. 1049–1058, 2004.

[41] A. Nag, C. Ramachandra, R. Balasubramonian, R. Stutsman, E. Giacomin, H. Kambalasubramanyam, and P.-E. Gaillardon, "Gencache: Leveraging in-cache operators for efficient sequence alignment," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 334–346.

[42] M. Nagura-Ikeda, K. Imai, S. Tabata, K. Miyoshi, N. Murahara, T. Mizuno, M. Horiuchi, K. Kato, Y. Imoto, M. Iwata *et al.*, "Clinical evaluation of self-collected saliva by rt-qpcr, direct rt-qpcr, rt-lamp, and a rapid antigen test to diagnose covid-19," *Journal of Clinical Microbiology*, 2020.

[43] M. Park, J. Won, B. Y. Choi, and C. J. Lee, "Optimization of primer sets and detection protocols for sars-cov-2 of coronavirus disease 2019 (covid-19) using pcr and real-time pcr," *Experimental & molecular medicine*, vol. 52, no. 6, pp. 963–977, 2020.

[44] N. V. Patel. Why the CDC Botched Its Coronavirus Testing. MIT Technology Review. [Online]. Available: https://www.technologyreview.com/2020/03/05/905484/why-the-cdc-botched-its-coronavirus-testing/

[45] A. Payne, N. Holmes, T. Clarke, R. Munro, B. Debebe, and M. W. Loose, "Nanopore adaptive sequencing for mixed samples, whole exome capture and targeted panels." *BioRxiv*, 2020.

[46] R. Ronan. Read until adaptive sampling. Oxford Nanopore Technologies. [Online]. Available: https://nanoporetech.com/resource-centre/read-until-adaptive-sampling

[47] D. Sart, A. Mueen, W. Najjar, E. Keogh, and V. Niennattrakul, "Accelerating dynamic time warping subsequence search with gpus and fpgas," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 1001–1006.

[48] T. Sauvage, W. E. Schmidt, H. S. Yoon, V. J. Paul, and S. Fredericq, "Promising prospects of nanopore sequencing for algal hologenomics and structural variation discovery," *BMC genomics*, vol. 20, no. 1, pp. 1–17, 2019.

[49] P. Senin, "Dynamic time warping algorithm review," *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, vol. 855, no. 1-23, p. 40, 2008.

[50] Y. Shu and J. McCauley, "Gisaid: Global initiative on sharing all influenza data–from vision to reality," *Eurosurveillance*, vol. 22, no. 13, p. 30494, 2017.
[51] M. Stoiber, J. Quick, R. Egan, J. Eun Lee, S. Celniker, R. K. Neely, N. Loman, L. A. Pennacchio, and J. Brown, "De novo identification of dna modifications enabled by genome-guided nanopore signal processing," *bioRxiv*, 2017. [Online]. Available: https://www.biorxiv.org/content/early/2017/04/10/094672
[52] V. Sundaresan, S. Nichani, N. Ranganathan, and R. Sankar, "A vlsi hardware accelerator for dynamic time warping," in *11th IAPR International Conference on Pattern Recognition. Vol. IV. Conference D: Architectures for Vision and Pattern Recognition,*, vol. 1. IEEE Computer Society, 1992, pp. 27–30.
[53] O. N. Technologies, "kmer_models," *GitHub repository*, 2017.
[54] ——. (2020) Ont open datasets: Gm24385 dataset release. [Online]. Available: https://nanoporetech.github.io/ont-open-datasets/gm24385_2020.09/
[55] Y. Turakhia, G. Bejerano, and W. J. Dally, "Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 199–213, 2018.
[56] J. R. Tyson, P. James, D. Stoddart, N. Sparks, A. Wickenhagen, G. Hall, J. H. Choi, H. Lapointe, K. Kamelian, A. D. Smith *et al.*, "Improvements to the artic multiplex pcr method for sars-cov-2 genome sequencing using nanopore," *bioRxiv*.
[57] R. Vaser, I. Sović, N. Nagarajan, and M. Šikić, "Fast and accurate de novo genome assembly from long uncorrected reads," *Genome research*, vol. 27, no. 5, pp. 737–746, 2017.
[58] S. Wei, Z. R. Weiss, and Z. Williams, "Rapid multiplex small dna sequencing on the minion nanopore sequencing platform," *G3: Genes, Genomes, Genetics*, vol. 8, no. 5, pp. 1649–1657, 2018.
[59] R. R. Wick, L. M. Judd, and K. E. Holt, "Performance of neural network basecalling tools for oxford nanopore sequencing," *Genome biology*, vol. 20, no. 1, p. 129, 2019.
[60] R. E. Workman, A. D. Tang, P. S. Tang, M. Jain, J. R. Tyson, P. C. Zuzarte, T. Gilpatrick, R. Razaghi, J. Quick, N. Sadowski *et al.*, "Nanopore native rna sequencing of a human poly (a) transcriptome," *BioRxiv*, p. 459529, 2018.
[61] L. Wu, D. Bruns-Smith, F. A. Nothaft, Q. Huang, S. Karandikar, J. Le, A. Lin, H. Mao, B. Sweeney, K. Asanović *et al.*, "Fpga accelerated indel realignment in the cloud," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2019, pp. 277–290.
[62] L. Zhang, X. Cui, K. Schmitt, R. Hubert, W. Navidi, and N. Arnheim, "Whole genome amplification from a single cell: implications for genetic analysis," *Proceedings of the National Academy of Sciences*, vol. 89, no. 13, pp. 5847–5851, 1992.
[63] P. Zhou, X.-L. Yang, X.-G. Wang, B. Hu, L. Zhang, W. Zhang, H.-R. Si, Y. Zhu, B. Li, C.-L. Huang *et al.*, "A pneumonia outbreak associated with a new coronavirus of probable bat origin," *nature*, vol. 579, no. 7798, pp. 270–273, 2020.