Nanopore Base Calling on the Edge

Abstract:

DeepNano-coral is a new base caller for nanopore sequencing that has been optimised for use with the Coral Edge Tensor Processing Unit, a small USB-attached hardware accelerator. We designed new versions of two key components used in convolutional neural networks for speech recognition and base calling to achieve this goal. We propose a new way of factoring a full convolution into smaller operations in our components, which reduces memory access operations, which is a bottleneck on this device. DeepNano-coral achieves real-time base calling during sequencing with slightly better accuracy than the Guppy base caller's fast mode, and it is extremely energy efficient, requiring only 10W of power.

Introduction:

MinION is a portable DNA sequencer from Oxford Nanopore Technologies (ONT) that measures electric current as DNA passes through nanopores. The device's electrical signals must be translated into sequences by base caller software. Nanopore read base calling is a difficult task, and existing tools require powerful hardware and a lot of energy to run in real time. We present DeepNano-coral, a new base caller that runs on the Coral accelerator with the Edge tensor processing unit (TPU), a small, energy-efficient, low-cost USB connected device. DeepNano-coral can process 1.5 million signals per second, which is enough for a MinION device to have real-time base calling. This makes our base caller ideal for eld sequencing applications, where low hardware requirements and high power efficiency are essential. Real-time base calling is also required to unlock some of the MinION device's most promising features, such as the ability to adapt the run length to the sample composition or selective sequencing. Deep neural networks are commonly used in today's base callers. Guppy, an ONT base caller, is based on recurrent neural networks (RNN) and has two architectures: a fast base caller that can base call with 85-92 percent median read accuracy in real time when using recent GPU cards, and a high-accuracy base caller (90-96 percent median read accuracy) that is too slow to use in real time without specialised setup. DeepNano-blitz sacrifices some accuracy in order to provide real-time base calling on a common CPU with a custom-engineered RNN, obviating the need for GPUs. Chiron and other RNN-based base callers are too slow for real-time base calling. Convolutional neural networks are another type of nanopore base caller (CNN). Bonito v.0.2, in particular, adapts the Jasper/Quartznet speech recognition architecture to base calling tasks. Bonito provided the most accurate base calling at the time of writing, but the time requirements outstrip even Guppy's high-accuracy mode. The Google Coral Edge TPU accelerator is a limited device designed primarily for vision tasks like image classification. It has only 8 MB of memory (for both model weights and intermediate tensors), only works with 8-bit integers (while GPUs typically work with 32-bit floating point numbers), and the compiler and libraries only provide a limited set of building blocks, optimised mostly for CNNs with small receptiveelds, which are commonly used in image processing. Due to the large size of the network and the use of large receptive elds, such a configuration almost completely eliminates the possibility of adapting RNN-based architectures, and even CNN-based architectures, such as Bonito, are difficult to adapt. DeepNano-coral, our new base caller that runs on the Edge TPU, provides real-time base calling that is significantly more energy efficient than previous methods.

Related work:

The technique presented in this paper is based on the QuartzNet speech recognition architecture, which was also employed in ONT's Bonito base caller. Briey, a window of the raw signal of length T is utilised as an input to a deep CNN, which processes the data using various sorts of blocks (see Figure 1). The network generates a tensor with 5 output channels in the nal decoder block. The softmax function converts the ve channels into probability distributions across the possible outputs A,C,G,T,- at each point, with a dash indicating an empty output. The CTC layer then selects the DNA sequence with the highest posterior probability. Convolutions are

grouped into two sorts of building blocks in the QuartzNet / Bonito architecture: B and C. A C-type block is made up of three layers: a convolutional layer, a batch normalisation layer (which renormalizes channel values and stabilises gradients to improve training), and an activation function (Bonito uses Swish). Residual skip connections are used in B-type blocks. Two branches are created from the input signal. The main branch is made up of R copies of a C-type sub-block, with the activation function omitted from the last copy. The second branch, known as skip connection, comprises of batch normalisation and pointwise convolution. The output is activated when the two branches have been added together. Bonito makes advantage of the resulting network, which is huge and computationally costly. Some intermediate results can be as large as B x T / 3 x 464, where B is the number of sequences in a batch and T is the sequence length. The network consists of 36 convolutional layers with a total of 6.6 million parameters, needing around 2.2 million multiplications per sample.
https://nanoporetech.com/about-us/news/new-research-algorithms-yield-accuracy-gains-nanopore-sequencing
https://openreview.net/forum?id=Hkuq2EkPf

Method:

we present the architecture of our new base caller designed for the Edge TPU. Our architecture is inspired by the Bonito CNN, which was drastically scaled down and key components were replaced by the enhancements described here. Further technical details regarding adapting Bonito-like architecture to the Edge TPU are described in the Supplement. The k-blueprint-separable convolutions. A convolutional layer is the dominant building block of many neural network architectures, mostly in the domain of image recognition, but recently also for automated speech recognition.In this paper, we consider 1D convolutions, which take as an input a tensor X of dimensions (T;Cin) representing a data stream of length T, each data point containing Cin values called channels. To apply a convolution with odd depth D, the input tensor X is rst padded with bD 2 c zeros at the beginning and at the end. Then the output tensor Y with dimensions (T; Cout) is computed as follows: Yt;j =P 0<=d<D;0<=i<C ,Xt+d;iWj;d;i + Bj ; where W and B are trained weights representing convolution kernel weights and bias.An obvious drawback of full convolutions is a large number of parameters(CoutDCin) and required ops (TCoutDCin). A standard solution is to use a separable convolution , which is an approximation of the full convolution by a composition of two operations: depthwise and pointwise. The depthwise operation works on each channel separately: Zt;j =P 0□d<D Xt+d;jW(D) d;j +B(D) j . This is followed by the pointwise operation, which mixes the channels at each timepoint: Yt;j =P 0□i<Cin Zt;iW(P) j;i +B(P)j . This reduces the ops from TCoutDCin to T(DCin +CoutCin). The ordering of pointwise and depthwise operations was chosen somewhat arbitrarily, and reversing it may improve the accuracy . The variant with the reversed order is called a blueprint-separable convolution.Recent works indicate that separable convolutions do not always improve the speed on non-CPU architectures, because the depthwise operation requires a smaller ratio of ops to memory operations, which are generally slow.A full convolution with depth D = 3 can be faster than a separable convolution with the same depth. Full convolutions with small depths are thus feasible in image recognition, while in base calling, the kernels need to be much larger. Our design of k-separable convolutions is heavily in uenced by this observation. Our goal is to reduce the time-consuming depthwise operations using dilation with step size k, and compensate by replacing the pointwise operation by a convolution operating on a window of size k instead of a single point. Namely, we start with what we call a fat-pointwise operation, which is a standard convolution of depth k: Zt;j =P 0□d<k;0□i<C Xt+d;iW(P) j;d;i + B(P) j .The second step uses a dilated depthwise operation with depth D=k, which skips points by using dilation k: Yt;j =P 0□d<D=k Zt+dk;jW(D) d;j + B(D) j . This reduces the depthwise kernel (and thus memory I/O) by a factor of k, while retaining the receptiveeld D of the whole layer.Note that the special case of k = 1 leads to a standard blueprint convolution, while we typically use k = 3, which on the Edge TPU roughly maintains the same computation time as separable convolutions, while increasing the accuracy. Our k-separable

convolutions oer running times comparable to separable convolutions, while providing roughly k times more parameters, which increases their expressive power.Residual block with depth-to-space compression. Our second change also targets reduction of the depthwise convolution. As shown in Figure 4, when we apply convolutions on a shorter input, we can use more channels in a comparable time.Our idea is to redesign the residual block of the CNN (B-type block in Figure 1) so that we compress its depth and increase the number of channels. In particular, compression with depth-to-space ratio x : y means converting input tensor (T;C) to tensor (T=x;Cy) using a strided convolution with both depth and stride set to x (see Figure 2). This convolution takes x consecutive data samples of C channels and converts them into a single compressed sample of Cy channels. At the end of the residual block, we restore the original dimensions with a strided transposed convolution. This makes the new block a drop-in replacement for the original B-type block design Compression ratio x=y < 1 saves memory, which is essential due to limited Coral resources. While compression may sometimes decrease accuracy, the network may learn to de-duplicate information from consecutive data samples, and thus prevent data loss. In fact, any subsequent pointwise operations effectively operate on x original samples, yielding increased receptiveelds. Thus, we can further lower the depth of the depthwise operation in the block, o setting larger computation of pointwise operations, which were increased by a factor of y2=x. In our experiments, compression ratio 3:2 works well on Coral. To complete the residual block, we add the depthwise operation before the decompression. While the original B-type block repeats separable convolutions R times, we repeat them R □ 2 times, since we consider the compression and decompression blocks as replacements for two separable convolutions. Identity initialization. A proper neural network initialization can a ect both trainability and nal accuracy of models

Result:

We tested the performance and energy usage of two machines with different configurations: a desktop (i7-7700k 4 core CPU; NVIDIA GTX 1650 GPU) and a laptop (i7-7700HQ 4 core CPU) that couldn't run Guppy's GPU version. We used a USB 3.0 interface to connect the Coral Edge TPU device to DeepNano-coral. DeepNano-coral reached the required speed for live base calling (1.5M signals per second) on both computers while using less than 11W. (computed as a difference between the idle energy consumption and the consumption during base calling). The overall energy used on base calling on our testing set was 0.58-0.68Wh, which is nearly half of the energy required by Guppy fast on the desktop. Although Guppy fast used less energy when the baseline was included due to its shorter running time, this would not translate to energy savings in a practical setting because the computer would need to run throughout the sequencing. DeepNano-coral uses more energy and runs at a slower pace on the GPU than GPU- and CPU-optimized applications. This emphasises the significance of network design optimization for a certain platform. To demonstrate the impact of our new network designs on base calling accuracy, consider the following example. We began with the modest Bonito architecture, replacing key components with our new designs described in the Methods section. Only B-type (residual) blocks are changed in these studies, but standalone C-type blocks remain unchanged. However, we found that changing the configuration of these C-type blocks has no discernible effect on accuracy. Starting with the little Bonito the accuracy and speed by adding the following features: 3-blueprint-separable convolutions, 3:2 compression, a mixture of the two, and finally identity initialization .We evaluate a variety of kernel depths for each option. Only at depths of size k = 3(2n + 1) do 3-separable convolutions have a symmetrical receptive eld. We stop at kernel size 21 in most studies because larger kernels result in base calling speeds that are slower than sequencing speeds. In general, adding our modifications improves accuracy at a similar rate, and the most accurate version is the one that includes all of our changes.