Sparse Convolutions for Faster Object Recognition

Wei Hao, Shivaram Venkataraman
University of Wisconsin, Madison

1 Introduction

With the rapid improvement in accuracy of machine learning models, models developed for image classification [3], object recognition [8] are being deployed in a number of new application scenarios. One particular scenario that is becoming increasingly important is the use case of object detection from video sources and a number of recent projects have proposed new ML models [2, 8, 9] and end-to-end systems for applying ML models to large video datasets [5, 6]. One of the main characteristics of input data in video processing is that a large part of the frame typically contains redundant information (e.g., the background information) and thus video processing frameworks typically can narrow down the area of interest by subtracting the background from each frame [1]. As our study shows (Section 2.2), background subtraction can lead to input images that are up to 97.4% empty on video datasets.

However, current computation libraries for machine learning inference don't differentiate between dense and sparse data (Section 2.3). As a result, even though the input images have a large number of zero entries, unnecessary cycles are spent on computing functions on zero values in the sparse data. These overheads lead to increased latency and are especially significant when ML models are used on devices with limited computing power, e.g. edge devices with only CPUs.

To address this challenge, we are working on a new framework for accelerating ML models applied to sparse input data. As a first in this framework we study convolution operations applied to sparse images. Convolution operations are among the most widely used in object recognition models and acceleration convolutions can enable significant end-to-end speed ups. We improve the convolution implementation in the Intel MKL-DNN [7] library and specifically exploit the structure of the direct convolution algorithm used as a default in MKL-DNN. We design a new technique to check the sparsity of rows and skip sparse rows, while efficiently assigning work among multiple threads. Our results show that the convolution 2D operation speeds up by upto 50% during inference on object recognition tasks run on CPUs. We finally describe work in progress in extending our approach to other workloads and operators.

2 Sparse convolutions

We next describe our approach to improving convolutions for sparse input data. We start by describing commonly used approach for implementing convolution in CPU libraries.

2.1 Direct Convolution

Three convolution algorithms are used in MKL-DNN based on the shape of input data: Direct Algorithm, Winograd Algorithm and Implicit GEMM Algorithm. The Direct Algorithm is used for most input shapes [7] and it computes the convolution by performing operations of the form:

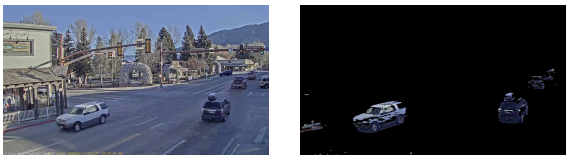
$$dst(n, oc, oh, ow) = \sum_{ic=0}^{IC-1} \sum_{kh=0}^{KH-1} \sum_{kw=0}^{KW-1} src(n, ic, oh \cdot SH + kh - ph_0, ow \cdot SW + kw - pw_0) \cdot weights(oc, ic, kh, kw)$$

where src, weights and dst correspond to the input image, convolution filter weights and output feature map with dimensions of $N \times IC \times IH \times IW$, $OC \times IC \times KH \times KW$ and $N \times OC \times OH \times OW$. And N, IC, IH, IW denote batch size, number of input channels, input height, input width. Correspondingly, OC, OH, OW represent the number of output channels, output height, output width and KH, KW, ph, pw denote the filter height, filter width, padding height and padding width respectively.

The direct convolution is implemented in MKL-DNN to take advantage of multi-core machines and the computation is spread across multiple threads. The assignment of work to threads happens as follows: The output tensor of the convolution (dst) is split by rows and each thread is then assigned a range of rows based on the number of threads available. Each thread is then responsible for computing the convolution for the corresponding rows of the output. For a dense image, the amount of computation in each thread doesn't differ. On the other hand, for a sparse image, a large fraction of computations can be omitted because of the nature of convolution: No matter what values are present in a convolution filter $weights$, the output will be be zero if the corresponding inputs are all zeros. However, given the parallelization strategy described above, the overall time taken for the convolution is bounded by the thread that consumes the most time.

2.2 SparsityCheck

We propose SparsityCheck, a sparse convolution method targeted towards multi-core CPU architectures. Given a convolution operation, SparsityCheck first inspects the values in every batch of input data assigned to each thread. If the values in the input are within the range of error tolerance (δ), then computations corresponding to those inputs are elided.



(a) (b)

Figure 1. Example Frame after Background Subtraction

Dataset	Jackson
Resolution	1920 × 1080 pixels
Frames	20,000
Avg. Sparsity Before Conv2D	97.4%
Avg. Sparsity After Conv2D	96.3%

Table 1. Jackson Dataset Characteristics

Batch Size	Dense (ms)	Sparse (ms)
1	73.8	85.5
4	293.7	285.03
8	597.2	599.9
16	1260.1	1011.1

Table 2. Average Time taken by MKL-DNN for Sparse vs Dense data

Batch Size	Average Improvement
1	36.8%
4	45.7%
8	51%
16	56.3%

Table 3. Average improvement from Sparse Convolution

Additionally to ensure that the overall operation is sped up, SparsityCheck also balances the remaining work uniformly across the threads. However checking the inputs for sparsity might itself lead to some overheads. We find that the data caching and access patterns used by the blocked memory layout [7] in MKL-DNN lead to minimal overheads as seen in our early results. We are also exploring other strategies like randomly sampling rows or randomly sampling entries within each row to further lower overheads.

2.3 Evaluation

The dataset we use is the Jackson dataset [4], an HD video captured from a traffic camera deployment in Jackson Hole, Wyoming. We choose a 12 minutes long time interval which contains 20,000 RGB Frames. Each frame has a resolution of 1920 × 1080 pixels. For pre-processing, we apply background subtraction [1] using the CV2 module from OpenCV to create sparse frames.

Dataset Sparsity We calculate the average sparsity of the video as the ratio of the number of zeros in every frame by the total size of all frames.

$$AverageSparsity = \frac{\sum_{i=1}^{num(frame)} numzero(frame)}{size(frame) \cdot num(frame)}$$

As shown in Table 1, we see that the average sparsity is around 97%. Further we see that applying a 7x7 convolution retains the sparsity.

Performance gains. We next present performance results from running the 7x7 convolution using the conv2D operation in Pytorch on the Jackson dataset. We used the same weights as in the first Conv2d layer in ResNet [3] during inference. We also vary the batch size used to investigate if batching a number of sparse images offers any performance improvement. First we compare the performance of MKL-DNN when presented with dense vs. sparse images in Table 2. We see that MKL-DNN has no speedup for small batch sizes and limited speedup with a batch size of 16.

We integrate SparsityCheck described before with MKL-DNN and Pytorch. Our current configuration performs sparsity checks on entire rows and only skips computation if all pixels in a row are zero ($\delta = 0$). This approach ensures there are no errors in the Conv2d output. Our results showing the speedup with respect to dense convolution in MKL-DNN is shown in Table 3. We find that SparsityCheck provides around 50% improvement compared to the baseline. Our current profiling suggests that our improvements are lower than the average sparsity in each frame due to the fact that our check is conservative and only elides computation when the entire row is empty. We are currently developing techniques to avoid computations for partially sparse rows.

3 Future Work & Conclusion

Sparse input data presents new opportunities for accelerating machine learning workloads. We presented a case study of video datasets with background subtraction and showed how sparse convolutions can improve performance.

We are planning to extend our work in a number of directions to build an efficient end-to-end machine learning engine for sparse data. While we considered convolutions on multi-core CPUs in this work we plan to study techniques for GPUs and other accelerators. We also plan to study how other workloads like natural language processing could benefit from sparse implementation of machine learning operators. Finally we are also looking at how low power edge devices that are often deployed close to a camera source can benefit from better handling of sparse data.

References

[1] S. Brutzer, B. Hoferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *CVPR*, 2011.

[2] P. Burlina. Mrcnn: A stateful fast r-cnn. *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 16)*, 2016.

[4] S. J. Hole. Jackson hole wyoming usa town square live cam - seejh.com, Jan 2018.

[5] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018.

[6] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. NoScope: Optimizing Neural Network Queries Over Video at Scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, 2017.

[7] Deep Neural Network Library (DNNL) . https://intel.github.io/mkl-dnn/. Last accessed 14 September 2019.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, Jan 2017.