Speedup Nerual Network Interface

JIN Fenglei

Supervisor: Bei Yu

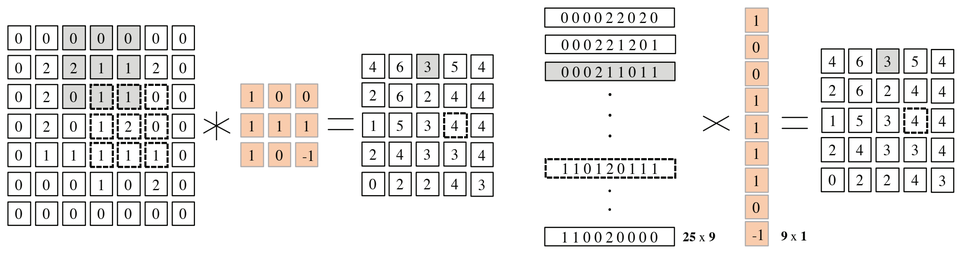
Department of Computer Science and Engineering

Introduction

Convolutional neural networks (CNN) are widely applied in many fields such as image classification and object detection. Though convolution gains a phenomenally success during the development of deep learning, inference compute heavy parts lie in convolution.
In this work, we implement Sparse-sparse Convolution in caffe, and test it with some famous neural network. The experimental result shows that this method leads to good speedup for some layers with sparse inputs and weights. And to reduce matrix format conversion penalty, a Sparse-in-sparse-out Convolution algorithm is proposed.

Previous Work

Thanks to the rapid development of GPU, different methods are proposed to accelerate the intensive computation. However, with the increasing demand of deploying CNN in mobile devices, speedup CNN for CPU computation becomes a popular topic.



(a) direct convolution (b) im2col-based convolution with lowered matrix

Fig. 1: Conventional convolution examples with 7×7 input size, 3×3 kernel size, 1 kernel stride in both dimensions.

Because CPU(GPU) architecture is more friendly to matrix multiplication, in caffe, convolution input is transformed into lowered matrix, i.e., the convolution is performed as fast matrix-matrix multiplication.

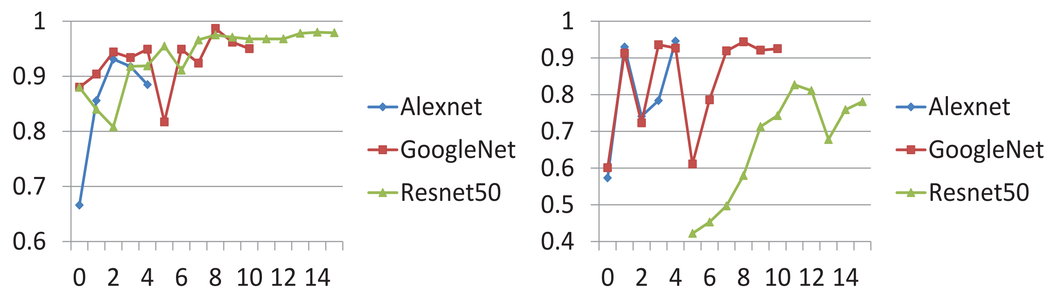


Fig. 2: Weight(left) & input(right) sparsity

Many researches like guided pruning[1] have been done to increase the sparsity of weight. Based on that Intel has proposed Sparse-matrix-dense-matrix Multiplication(SpMDM) and achieves the state of the art speedup.

Sparse-sparse Matrix Multiplication Algorithm

Algorithm 1 sparse-sparse matrix multiplication algorithm

Input: weight M is CSC format with 3 arrays: col_ptr, rowidx and val, and input N is CSR format with 3 arrays: row_ptr, colidx, val
Output: Output O

function CSC_MUL_CSR(M, N)
  allocate O with size $m \times n$
  for row_n in range size($N.row\_ptr$)-1 **do**
    for c in range (N.row_ptr[row_n],N.row_ptr[$row_n + 1$]) **do**
      for r in range (M.col_ptr[row_n],M.col_ptr[$row_n + 1$]) **do**
        $O(M.rowidx[r], N.colidx[c]) += M.val[r] \times N.val[c]$
      end for
    end for
  end for
  return O
end function

Consider sparse matrices M, N of size $m \times k$, $k \times n$ with sparsity α, β respectively.

Time Complexity:

- Input2CSR: N_{size}
- Multiplication: $\frac{N_{size} M_{size}(1-\alpha)(1-\beta)}{k}$
- Total complexity: $N_{size} + \frac{N_{size} M_{size}(1-\alpha)(1-\beta)}{k}$

The time complexity of common matrix multiplication is mkn. Therefore this algorithm achieve $\frac{1}{(1-\alpha)(1-\beta)}\times$ speedup theoretically.

Space Complexity:

- Weight & input: $M_{size}(1-\alpha) + N_{size}$
- Input2CSR: $N_{size}(1-\beta)$
- Total complexity: $M_{size}(1-\alpha) + N_{size}(2-\beta)$

The space complexity of common multiplication is $M_{size} + N_{size}$. When sparsity reaches 0.8, our algorithm only cost 70% space as before.

Experimental Results

We test the speedup on some convolution layers of Caffenet and Resnet50. The trained weights can be easily download at SkimCaffe. The test input is 1000 images downloaded at ImageNet.

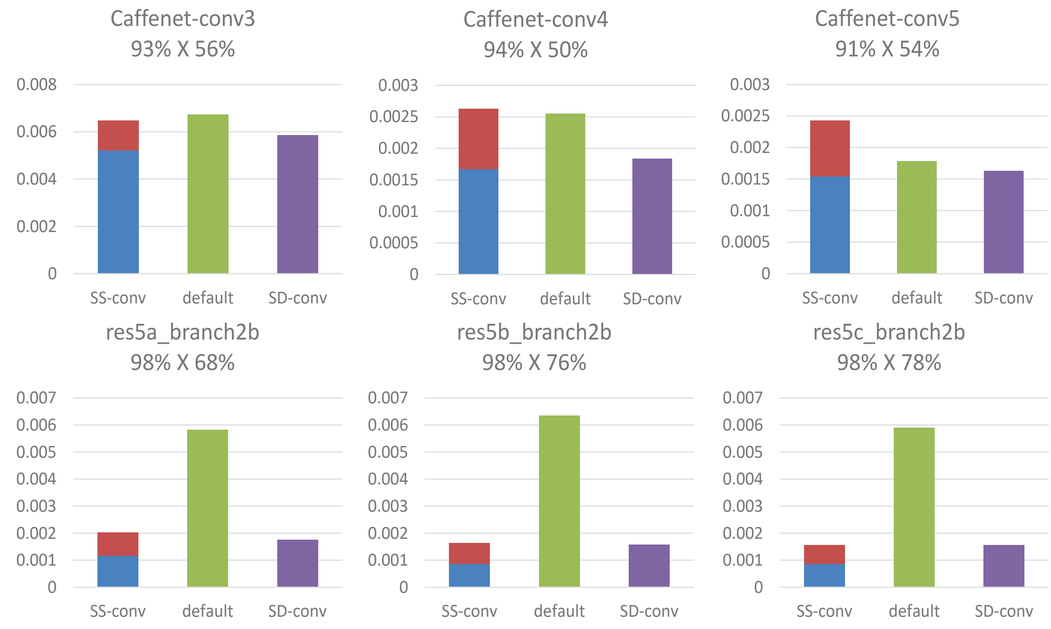


Fig. 3: The left percentage is the sparsity of weight matrix, while the right is input sparsity. The left cylinder represents the time of Sparse-sparse Convolution: input2CSR(red) plus Sparse-sparse Matrix Multiplication(blue), while the middle and right represents default and SpMDM time.

- Because of the low sparsity of Caffenet convolution layers, the speedup is not obsvious.
- Sparse-sparse Convolution achieves $2.9\times \sim 3.9\times$ speedup in Resnet50, reaches the state of the art speed.
- The penalty of input2CSR stunts the performance.

Advancement

As showed in the result, the penalty of input2CSR limits the performance of Sparse-sparse Convolution. To eliminate this penalty, a new algorithm called Sparse-in-sparse-out Convolution is proposed. The key points of this algorithm are:

- Output a matrix in CSR format, which can be directly used as the input sparse matrix of the next convolution layer.
- One sparse conversion is needed, and all matrices in the back convolutions flow in sparse format.
- The time complexity reduce to $N_{size}M_{size}(1-\alpha)(1-\beta)/k$ in behind convolution layers.

Algorithm 2 Sparse-sparse Matrix Multiplication(output CSR)

Input: Weight M and Input N in CSR format with 3 arrays: row_ptr, colidx, val
Output: Output O in CSR format with 3 arrays: row_ptr, colidx and val

function CSR_MUL_CSR(M, N)
  for row_m in range size($M.row\_ptr$)-1 **do**
    Initialize temp[$N.col.size$]
    for c in range (M.row_ptr[row_m],M.row_ptr[$row_m + 1$]) **do**
      for r in range (N.col_ptr[$M.colidx[c]$],M.col_ptr[$M.colidx[c] + 1$]) **do**
        $temp(N.colidx[r]) += M.val[c] \times N.val[r]$
      end for
    end for
    Detect whether the elements in $temp$ equals to zero, update 3 arrays of O
  end for
  return O
end function

Since the sparse conversion is eliminated, the speedup can increase to $7\times$ maximum theoretically.

Conclusions

Sparse-sparse Convolution can reduces memory consumption with good speedup of sparse input and weight convolution layer. It is suitable for CPU-only devices like mobile phones.

References

[1] Park, Jongsoo, et al. "Faster CNNs with Direct Sparse Convolutions and Guided Pruning." arXiv preprint arXiv:1608.01409 (2016).

[2] Wen, Wei, et al. "Learning structured sparsity in deep neural networks." Advances in Neural Information Processing Systems. 2016.

Acknowledgement

My deeper gratitude goes to my summer research advisor Prof. Yu and PhD mentor Mr. Yuzhe Ma. Their guidance and experience help a lot to finish my research. And thanks Engineering Faculty for providing me this chance which encourages me to think and study in a researcher's perspective.