Sketching Based Convolution Neural Network

Chen Liao, Terry Junxiang Shi, Yeqing Wang, Mingjie Chen
Professor: David Woodruff

July 18, 2020

Abstract. *This paper describes our proposed method of using the method of Tensor Sketching on the Convolutional layer of the CNN Model. We have 2 methods that will be applied on the CNN model. We will compare the accuracy and running in the experiment section to show the effectiveness of our proposed method.*

1. Introduction

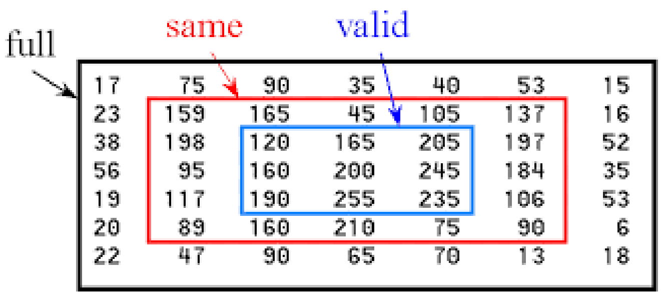
In data-rich fields in artificial intelligence such as image recognition, speech recognition, and computer vision, and many other domains with a heavy emphasis on data processing, especially dealing with different types and modes of data, deep neural networks have become more and more powerful and successful when processing big data. Convolutional neural networks (CNNs) have been essential to the success of deep learning, as in practical situations, the power of these networks can be clearly seen in fields like computer vision. However, as the power of CNNs become more invasive, the amount of data that goes into each network also increases dramatically. With current CNNs processing millions and billions of data, it takes up a lot of running time and storage space to process such intensive calculations. However, some intensive calculations could be simplified to a great extent using network approximation[4] . Expanding on the idea of network approximation using tensor sketch [2], a question that arose was whether or not we could create a smaller network architecture and data processing system, in other words an approximation of an original network, while maintaining similar accuracy results of the original target network architecture. In doing so, the running time, storage space, and GPU capacity would dramatically decrease.

There are of course many methods of network approximation and dimensionality reduction techniques, such as single value decompositions (SVD) and universal network approximation theorems. Many objectives of these techniques apply to CNNs as well, but in this paper, we will focus on implementing tensor sketch techniques to CNNs to different real data networks to evaluate the effectiveness of deep neural network approximation using sketching techniques. Many sketching techniques could be implemented in CNNs for network approximation as well, however, they usually focus on vectors and matrices, which are less applicable for CNNs. We could extend such sketching techniques to higher-order tensors, as previously done in works of [6]. In this paper, we discuss our works and experiments with different sketching techniques including count sketch, gaussian sketch. In addition, we tried random sampling as well, as it often has outstanding practical applications.

Sketching is a randomized dimensionality-reduction method for data sets that aims to keep hold of similar and relevant features in the original data set. A method of sketching, count-sketch utilizes randomized hash functions to achieve such a goal by multiplying the current data set to one with one degree less. It is a probabilistic data structure that generates a table featuring the frequencies or counts of a particular data set and maps them out into a particular data structure. A Count Sketch data structure uses hash functions to map out events to frequencies, and the actual data structure itself is a two-dimensional array of K * N matrix. We define K as the number of Hash functions, and there are N columns. Each Hash function corresponds to a single row. Thus we create a matrix S. So, we ensure the Matrix S is sparse, and the matrix starts out with all 0 values. For each hash function, we compute the corresponding hash value of the element, and mod that by M, the number of columns, and increment the corresponding count in that row for that hash function. A Gaussian Sketch data structure uses a similar approach to Count Sketch, and it is simply a product of a Count Sketch Matrix and a Gaussian Matrix[5].

2. Preliminary

Convolution Convolutions are primary operations on input features and kernels in convolutional layers and pooling layers. There are three kinds of convolution operation known as ‘full’, ‘same’ and ‘valid’ which vary in size of the convolution result.



17	75	90	35	40	53	15
23	159	165	45	105	137	16
38	198	120	165	205	197	52
56	95	160	200	245	184	35
19	117	190	255	235	106	53
20	89	160	210	75	90	6
22	47	90	65	70	13	18

The ‘full’ convolution enables the kernel to stretch out of the valid region of the feature map by zero padding, generating results with ”full” size. The ‘same’ convolution reserves the central part of a full convolution which conform to the size of the larger input. The valid convolution forbids kernels from stretching out of the valid feature region, which delivers outputs with ”valid” size. In our approach, although inputs are padded with zeros during the data preprocessing, valid convolutions are adopted in all pure convolutional operations in the forward propagation phase.

Unrolling convolution Since general convolution operation delivers an asymptotic complexity of O(n^4) whereas matrix multiplication finishes in O(n^3) in the worst case, it is naturally considered conducive to running time efficiency by replacing plain convolution operations with matrix multiplications. Here the idea of unrolling convolutional layers in introduced.According to Chellapilla et al,[1]The central idea of unrolling convolution is an unfolding and duplication of the input and a rearrangement of the kernel parameters that produces a CPU friendly ordering. To depict a traditional convolution to matrix multiplication, all possible unique positions of the kernel on the feature map should be identified as sub matrices. Then these matrices are reshaped into row-wise vectors. Finally, these vectors are stacked together in column-wise fashion. Similar operations are conducted on kernels but in a transposed fashion. A more intuitive illustration of transmitting process is shown in figure 1 by taking sliced 2-D matrices as an example. This set of processes converts original matrices into corresponding convolutional matrices whose multiplications are equivalent to convolutions. A more efficient convolutional layer could be acquired by applying this strategy during forward propagation.[3]

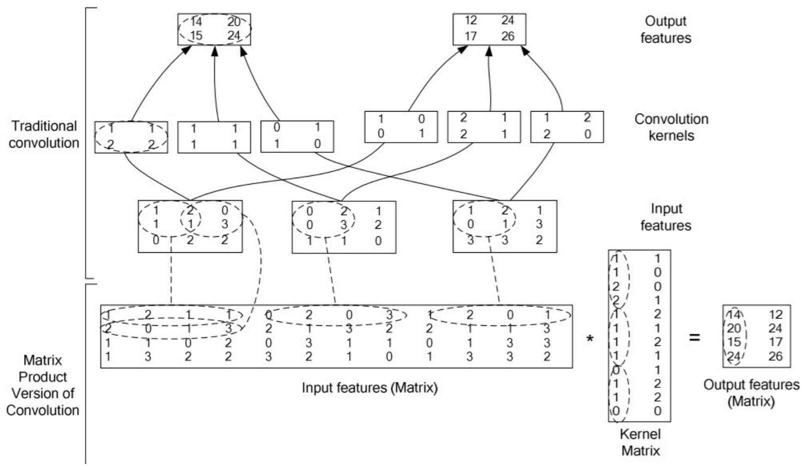


Figure 1. sliced 2-D matrices

Pooling layers Pooling layers serves as compacting features to decrease computational complexity within in a network. Traditional pooling strategies includes average pooling and max pooling, whereas max pooling becomes dominant recently since it underlines most significant features within a small feature neighborhood instead of blurs them (which an average pooling always does). However, the essence of a pooling operation is a convolution with a greater stride (usually greater than one), which also contributes to time redundancy caused by convolutions. Based on the idea of unrolling convolution introduced above, one could naturally propose a method to reduce a pooling convolution to a matrix multiplication. However, we involve an idea of tensor sketch here to replace the whole pooling layer for a further improvement in efficiency.

Tensor sketch As mentioned, tensor sketch is basically a dimensionality reduction approach by tensor approximation, which represents the original tensor by a smaller one within some tolerable error. This approach compacts essential information by projecting it into proper subspace[7], thus serves for the same function as traditional pooling theoretically. Therefore, max pooling layer is replaced with multiple tensor sketch strategies in our approach.With the definition from David P. Woodruff [7] and Shiva P. Kasiviswanathan[2], tensor sketch could be formally represented as:

Definition 2 (Mode-n Sketch). *Given a tensor, $\mathcal{T} \in \otimes_{i=1}^{p} \mathbb{R}^{d_i}$, the mode-$n$ sketch of $\mathcal{T}$ with respect to a random scaled sign matrix $U_n \in \mathbb{R}^{k \times d_n}$ for $n \in [p]$, is defined as the tensor $\mathcal{S}_n = \mathcal{T} \times_n U_n$.*

A detailed sketching process for an arbitrary tensor along a certain dimension could be calculated with the multiplication results of the sketching tensor with every possible fiber along this dimension in the tensor, where tensors along a certain dimension could be defined as vectors elicited by fixing other dimensions with values of arbitrary random combinations. A more intuitive demonstration of diverse orienting fibers of a 3-D tensor is shown in Figure 2.

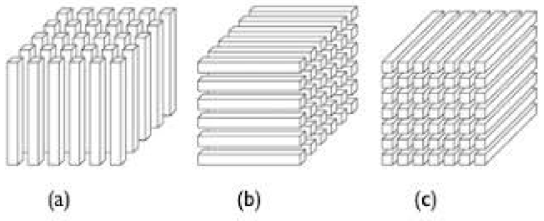


Figure 2. diverse orienting fibers of a 3-D tensor

Various sketching strategies are available and proved effective, but here only Gaussian Sketch (denoted as GS for simplicity) and Count Sketch (denoted as CS for simplicity) are scrutinized and experimented. A very basic difference of CS and GS lies in the sketching matrix. Within the same size as a premise, the sketch matrix of CS consists of columns with one and only one random binary signal of value either 1 or -1 in a random row where as the one of GS is a matrix of normal random variables with an average of zero and a certain standard deviation. Typical representations of sketch matrices of both CS and GS are shown in Figure 3.Comparing to sketch matrix of GS, sketch matrix of CS is priory in its sparsity, which could allow sparse matrix calculation for even a better efficiency than GS. Sparse matrix manipulation is already a mature technique and well supported by various libraries such as Scipy and Matlab sparse library.

$$\begin{bmatrix} 0.54 & 0.32 & 3.58 & 0.73 & -0.12 & 0.67 & 0.49 & 0.29 \\ 1.83 & -1.31 & 2.77 & -0.06 & 1.49 & -1.21 & 1.03 & -0.79 \\ -2.26 & -0.43 & -1.35 & 0.71 & 1.41 & 0.72 & 0.73 & 0.89 \\ 0.86 & 0.34 & 3.03 & -0.20 & 1.42 & 1.63 & -0.3 & -1.15 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3. Sketch Matices of Count Sketch and Gaussian Sketch

3. Proposed Model

The differences of this model comparing to traditional ones comes from two aspects: the replacement of plain convolution operation with matrix multiplication and the replacement of pooling layers with sketching layers. In the unrolling convolutional layer, multi-dimensional input features and kernels are sliced firstly. Then every input feature and kernel pair is converted into a convolutional matrix pair. Subsequently, multiplications are applied and results are stacked. Finally, the result tensor is reorganized to conform to the result of plain convolution thus the spatial property of the original input still holds. The sequence diagram (Figure 4) shows the working flow of our convolutional layer comparing to traditional convolutional layers.

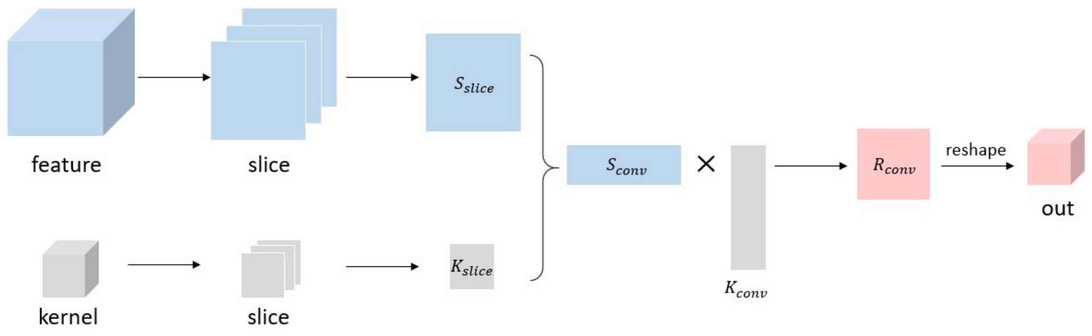


Figure 4. Diagram of sketch pooling layer

In the second refinement, traditional pooling layers in CNN are replaced with tensor sketches as both compress the input and make approximations but tensor sketch is prior in asymptotic complexity. Another technique applied in sketching layers is multi-way sketch. Since single sketch operation only compresses information along single dimension, the ratio of the input is modified and some of the spatial information is lost if we only apply single sketch on the input tensor. Therefore, multiple sketches with the same sketch matrix along diverse directions are applied to maintain the original ratio of the input.The sequence diagram (Figure 5) shows the working flow of our sketching layer comparing to traditional pooling layers. Generally, the scheme of our model follows an add-on and replicable fashion as refinements of this model could be transplanted to all convolutional neuron networks because modifications lie in convolutional layers and pooling layers — the most essential modules in CNNs.

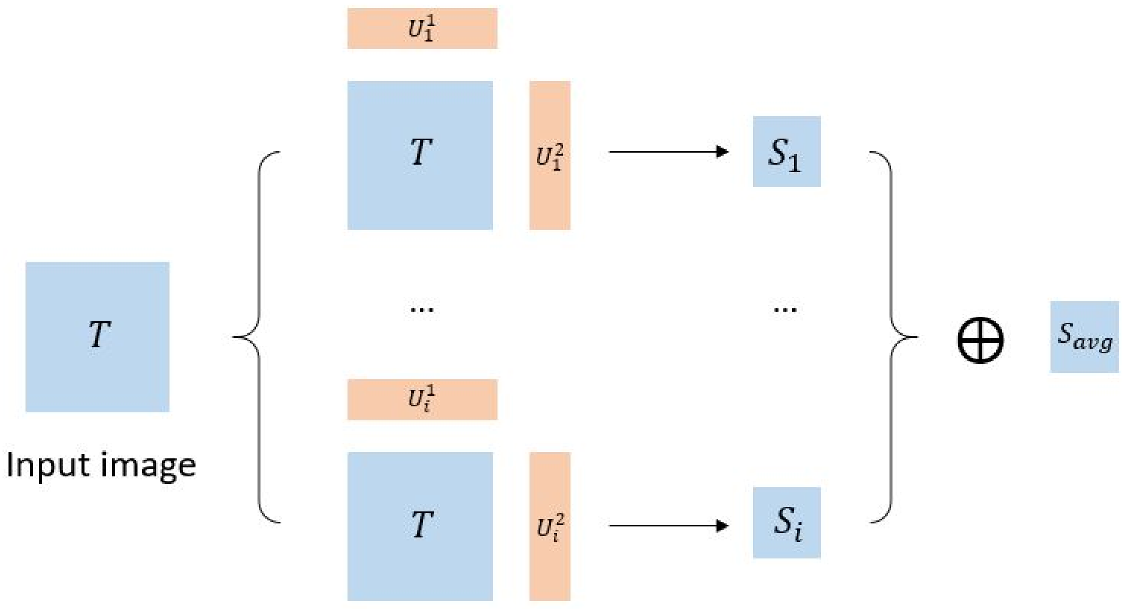


Figure 5. Diagram of fast conv layer

3.1. Method

We use MINST database to do multi-class classification of the 28×28 pixel black-and-white image into one of the ten digit classes: 0-9. The convolutional neural network model we use here is LeNet. The classification neural network consists of the single hidden layer and output layer. This hidden layer has 100 nodes that use the ReLU activation function. Since we have 10 classes to classify, the output layer is constructed with 10 nodes. Only three layers have weight respectively: W_1, W_5, W_o. Softmax function was used as activation function for the output nodes. The following figure summarizes the architecture. Multiple Tensor Sketch algorithm is fused into pooling layer to speed up the whole algorithm.

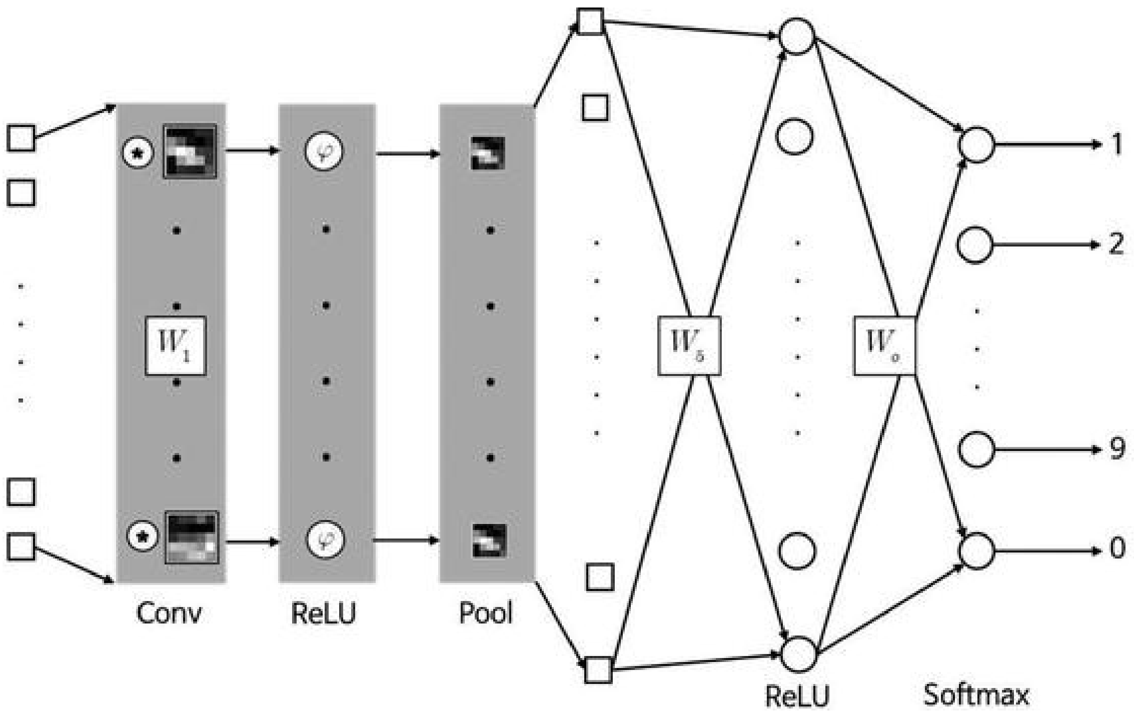


Figure 6. Architecture of neural network

4. Experiment

In this section, we experimentally demonstrate the effectiveness of our proposed method of applying different sketching methods on the convolutional layers of the CNN Models. Our goals through the experiments is to test the limits of the running time of Sketch Applied ConvNet and design a substantially smaller network that achieves almost the same performance as the original network on a wide range of datasets.[1]

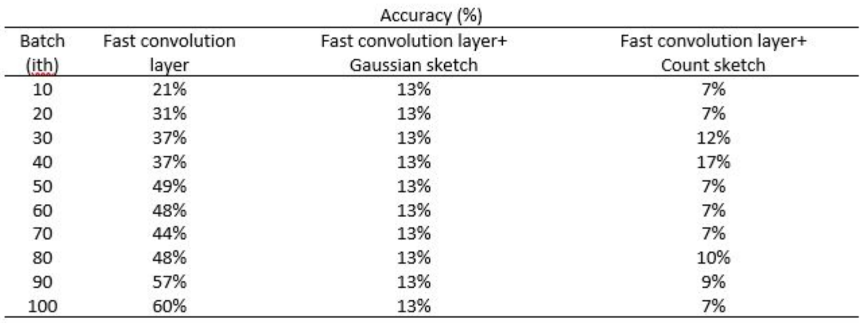
Comparing Groups We have 3 comparing groups in total, two methods were mentioned in the Background session, the count sketch and gaussian sketch, and one last comparing group is the plane CNN model. By doing this, we can compare the differences of running time and accuracy of our proposed methods to the plain CNN model.

4.1. Datasets and Network Architecture

Datasets We can classify our data sets into three group according to their image sizes. The large datasets we prepared are ImageNet10, the medium size datasets we prepared

CIFAR 10 and SVHN, the small size datasets were MNIST.
CNN Architecture We prepared 2 CNN Models to test our proposed method on. By using the skethching method on the convolutional layer of the prepared model, we can compare the result with the original CNN Model.

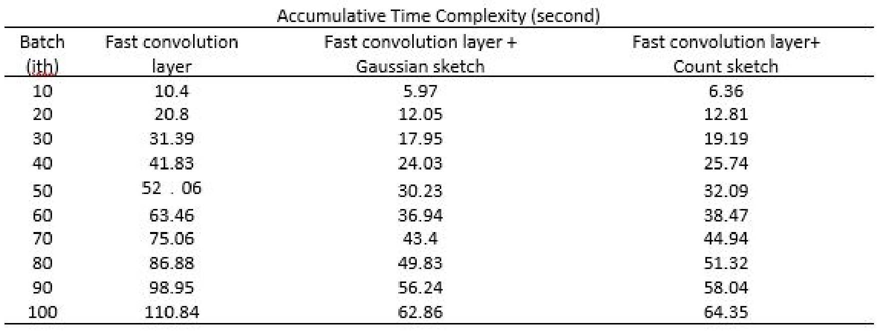
4.2. Results



Accuracy (%)

Batch (ith)	Fast convolution layer	Fast convolution layer+ Gaussian sketch	Fast convolution layer+ Count sketch
10	21%	13%	7%
20	31%	13%	7%
30	37%	13%	12%
40	37%	13%	17%
50	49%	13%	7%
60	48%	13%	7%
70	44%	13%	7%
80	48%	13%	10%
90	57%	13%	9%
100	60%	13%	7%

Figure 7. Accuracy



Accumulative Time Complexity (second)

Batch (ith)	Fast convolution layer	Fast convolution layer + Gaussian sketch	Fast convolution layer+ Count sketch
10	10.4	5.97	6.36
20	20.8	12.05	12.81
30	31.39	17.95	19.19
40	41.83	24.03	25.74
50	52 . 06	30.23	32.09
60	63.46	36.94	38.47
70	75.06	43.4	44.94
80	86.88	49.83	51.32
90	98.95	56.24	58.04
100	110.84	62.86	64.35

Figure 8. Accumulative time complexity

The results are shown in the graph above, we ran our proposed method on the convolutional layer on the CNN model ten times to compare the total final running time, . In Figure 7, it shows the accuracy result we get when we compare the three methods, Figure 8 shows the result of total running time. In Figure 8, you can see the fast convolution operation running time take to over 100 seconds. The convolutional layer, the running time greatly reduce to 65 seconds. The Gaussian sketch, the running reduce to around 63 seconds. In our proposed method, each run takes less time as well for our 2 proposed methods. Aside from the running time, the accuracy are another important part to consider. As Figure 7 shows, our proposed methods did not achieve the same accuracy comparing to the Fast Convolution its self. We believe if we train them enough, the accuracy will rise to a better level, however, our proposed method will not achieve to an acceptable performance.

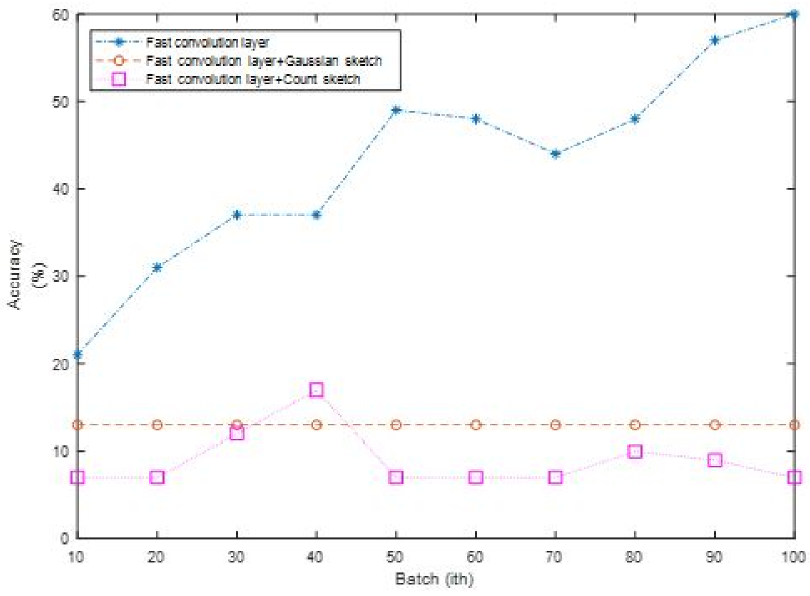


Figure 9. Accuracy

4.3. Evaluation

In conclusion, as we mentioned before, our goal throughout the experiment is to test the limits of the running time of Sketch Applied ConvNet and design a substantially smaller network that achieves almost the same performance as the original network on a wide range of datasets. As our result displayed, however, our proposed method did accelerate the operation time efficiently, but did not achieves almost the same performance. The accuracy were way off from our expectation of good performance. Using our 2 proposed method of sketching the convolutional layer of the CNN Model, the running time were greatly improved but at the same time, we loss a great amount of data by sketching the layer. Also, there are some limitation of our methods and some future works we can do.

5. Limitation

From the result, in terms of time complexity, we have improved greatly in running time of group Fast convolution layer+Count sketch than that of group Fast convolution layer+Gaussian sketch. However, the time difference between group Fast convolution layer+Count sketch and is not so significant. As aspect of accuracy, the pure fast convolution layer is much higher than the other two groups during all training batch, which indicates inefficiency of new fusion of tensor sketch method into LeNet model.

5.1. Future Works

Another intriguing algorithm—OSNAP and random samples could also be put into comparison with Sketch methods. Also, we could adjust our algorithm to solve other database that could be solved using LeNet model, for example, Fashion-MNIST, CIFAR-10, CIFAR-100. Furthermore, we could apply the innovative tensor sketch to different convolutional neural network model, for example, VGG-16. In this way, a more comprehensive understanding that whether tensor sketch method can be applied and successfully improved in CNN could be achieved.

References

[1] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High Performance Convolutional Neural Networks for Document Processing. In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), October 2006. Université de Rennes 1, Suvisoft. http://www.suvisoft.com.

[2] Shiva Prasad Kasiviswanathan, Nina Narodytska, and Hongxia Jin. Deep neural network approximation using tensor sketching, 2017.

[3] Zewen Li, Wenjie Yang, Shouheng Peng, and Fan Liu. A survey of convolutional neural networks: Analysis, applications, and prospects, 2020.

[4] Jelani Nelson and Huy L. Nguyen. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. *arXiv e-prints*, page arXiv:1211.1002, November 2012.

[5] Douglas J. Orr, André Alcântara, Maxim V. Kapralov, P. John Andralojc, Elizabete Carmo-Silva, and Martin A.J. Parry. Surveying rubisco diversity and temperature response to improve crop photosynthetic efficiency. *Plant Physiology*, 172(2):707–717, 2016.

[6] Weicai Wang, Yang Gao, Pablo Iribarren, Yanbin Lei, Yang Xiang, Guoqing Zhang, Li Shenghai, and Anxin Lu. Wang et al. 2015. 09 2015.

[7] David P. Woodruff. Computational advertising: Techniques for targeting relevant ads. *Foundations and Trends® in Theoretical Computer Science*, 10(1-2):1–157, 2014.