LITERATURE REVIEW: — Parallelizing the Interpolation between Latent Space of Autoencoder Networks to Introduce Novelty in 3D Object Reconstruction

Tansin Jahan
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
tansinjahan@cmail.carleton.ca

October 5,2018

1 Introduction

The idea of parallel computing infers to execute more than one tasks simultaneously so that the complexity(ex- time, space etc.) in computation can be carried out smoothly. Though this is the simplest definition of parallel computing, in the real world, parallelization means a lot more than just handling the complexity in the computation of algorithms. With the development of Deep Learning Networks, parallel computing has become the essential choice for the implementation of these huge networks.Typically there are million of parameters(weights, biases) and a large amount of data involves in a CNN model. This large amount of data produces as output at each layer is known as Forwarding propagation. The loss is calculated at the same time and updated parameters are backpropagated to the layers to minimize the loss. For this, the different optimization technique is applied such as Stochastic Gradient Descent or ADAM etc. This whole process is computationally expensive and therefore needs to be parallelized so that the training time minimizes and the network can run faster than usual times.

So, this project involves building a simple Autoencoder neural network which is popular to reconstruct 2D or 3D object given as an input. Autoencoder, being a symmetrical deep network involves several convolutional layer and parameters mentioned above. Typically this network feeds on 2D images involving higher dimension and then reduces that into lower dimension [6] which is called as latent space. In latent space (represents as Z vector) the input object has minimum dimension with maximum features. From the lower dimension, the Z vector passes through deconvolution and therefore produces the same 2D images.

2 Literature Review

With the recent development of Convolutional Neural Network, it has been used to solve several Computer Vision problem. For example, object detection or reconstruction from input image, semantic information extraction from a scenario, object segmentation- these are all recent computer vision application where CNN has been used to produce better result. Likewise, 3D reconstruction of an object is one of the Computer Vision problem and recently multiple approaches (ex: 3D-GAN, 3D-shapenets) has been proposed as a solution to the problem [4]. Previous work shows that given a depth image as input, the volumetric representation can be produced [5]. Following these works, an approach to reconstruct 3D voxelized object from different viewpoint of one or more images of that object (i.e. single-view or multi-view) is proposed where recurrent neural network has been used [1]. In total 50,000 model is used to train and test the proposed network. Training the network with this large amount of data is really time consuming and therefore introducing parallelism between the layers of the model can help to improve the performance of the network.

2.1 GPU implementation

GPU implementation for convolutional operation of deep networks has become very popular to improve state-of-the-art results. In the paper, 'Imagenet classification with deep convolutional neural networks' - two GTX 580 3GB GPU has been used to train the network which took five to six days to complete the whole training [3]. In this parallelization scheme, they kept half of the kernels (or neurons) on each GPU, where the GPUs communicate only in certain layers.

Though a single GPU can greatly accelerate neural network training and testing, sometimes it is desirable for muptiple GPUs to be used at once because the neural network model and data may not fit onto a single GPU or training with one GPU will simply take too long. Many researchers have approached the question of how to best use multiple GPUs to train deep networks more effectively. So, here I discuss one such method to parallelize convolutional neural networks described by Krizhevsky in his article One weird trick for parallelizing convolutional neural networks [2]. Many convolutional neural networks comprise both convolutional layers, where the linear function f(x) is a convolution of the input and a set of learned weights, and fully connected layers, where the linear function f(x) is an inner product between the input and a set of learned weights. Krizhevskys key insight is that different types of parallelization are better for different types of layers. Convolutional layers generally require a large amount of computation, but have fewer parameters than fully connected layers. Thus, sharing updated parameters between GPUs is less burdensome than for fully connected layers, and splitting data amongst multiple GPUs helps speed computation. To this end, Krizevsky proposed parallelizing the convolutions with data parallelization, where the same convolutional layers are on each GPU, but the GPU process different batches of the data, and parallelizing the fully connected layers with model parallelism, where the model is split amongst GPUs, but each GPU sees the same data [2].

References

[1] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. *CoRR*, abs/1604.00449, 2016.

[2] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *CoRR*, abs/1610.07584, 2016.

[5] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[6] Zhuotun Zhu, Xinggang Wang, Song Bai, Cong Yao, and Xiang Bai. Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 204:41–50, 2016.