Multi-GPU 환경에서 피라미드넷을 활용한 이미지 분류 분산 처리

한장훈 이호재 윤기영°
서강대학교
hanjh04@naver.com {koo8281, yky9494}@sogang.ac.kr

Distributed Processing of Image Classification using Pyramid Net in Multi-GPU Environment

Janghoon Han, Hojae Lee, Kiyoung Yoon°
Sogang University

요 약

딥 뉴럴 네트워크 모델은 고성능 그래픽 프로세서 유닛(GPU)을 통한 병렬처리로 느린 학습 속도라는 기존의 한계점을 개선하였다. 하지만 DNN 모델이 복잡할 경우 배치 사이즈가 제한되고 속도가 느리다는 문제는 여전히 존재한다. 본 논문에서는 이러한 문제를 해결하기위해 분산처리 환경에서 GPU를 사용하는 딥러닝 모델을 제안한다. 또한 Single-GPU와 Mulit-GPU 환경에서 학습 속도 차이를 비교 분석 한다. 이를 바탕으로 향후 분산 처리 환경에서 효율적인 DNN을 설계하는데 도움이 되고자 한다.

1. 서 론

딥 뉴럴 네트워크(DNN)는 최근 다양한 분야에서 높은 성능을 내고 있다. DNN의 학습 방법 중 대표적인 방법은 Stochastic Gradient Descent(SGD)[1]이다. SGD는 크게 두 단계를 거치는데 첫 번째 단계는 입력을 네트워크에 넣어 출력을 구하고 정답 데이터와 출력데이터의 오차를 구하는 것이다. 두 번째 단계는 앞에서 구한 오차를 바탕으로 네트워크를 뒤로 전파해 가면서 각 노드의 가중치를 갱신하는 backpropagation을 수행하는 것이다. SGD를 batch 마다 수행하면서 네트워크를 학습하게 되는데 두 단계 모두 행렬 곱셈과 같은 많은 양의 연산이 필요하다. 그렇기 때문에 파이토치 같은 딥러닝 프레임 워크들은 이러한 연산을 가속화 하기위해 GPU(Graphics Process Unit) 사용을 지원한다.

배치 사이즈는 학습에 영향을 준다. 배치사이즈가 너무 작으면 상대적으로 부정확한 기울기를 사용하기 때문에 수렴이 어렵고 시간이 오래 걸린다는 문제가 있다. 모델이 복잡해지면 mini batch가 요구하는 GPU 메모리가 증가하게 된다. 따라서 모델이 클 경우 배치 사이즈의 크기에 제한을 받게된다.

이미지 분류(Image classification) 태스크[2]는 딥러닝을 도입하게 되면서 성능이 폭발적으로 증가하였다. 하지만 모델의 성능이 증가할수록 네트워크의 깊이가 증가하는 경향을 띄었다. 모델이 복잡해지면서 학습에 걸리는 시간이 증가하였고 학습 시간을 줄이기 위해 Single-GPU가 아닌 multi-gpu를 사용하여 학습하는 방법이 제안되었다.

본 논문에서는 이미지 분류 알고리즘 중 하나인 PyramidNet[3] 을 구현하고 Single-GPU 환경에서 한정된 배치 사이즈 문제와 학습 속도 문제를 해결하기 위해 multi-GPU 분산 모델을 제안하였다.

2. 관련 연구

이미지 분류는 이미지가 입력으로 들어오면 뉴럴 네트워크가 이미지가 어떤 분류에 해당하는지를 예측하는 것이다. 이 과정을 수행하기 위해 합성곱 계층(convolutional Layer)과 풀링 계층(pooling Layer)을 사용하여, 이미지에 필터링 기법을 적용하고 이미지의 국소적인 부분들을 하나의 스칼라 값으로 변환함으로써 이미지의 크기를 줄이는 CNN[4] 이 주로 사용되었는데, 주요 아키텍처로, AlexNet[5], ResNet[6] 등이 연구되었다.

AlexNet은 convolutional layer, pooling layer 그리고 dropout layer가 각각 5개씩 있고, fully-connected layer가 3개인 간단한 구조이다.

ResNet은 출력 값과 입력 값의 차이를 얻는 Residual Learning[6] 방식을 이용하여 forward와 backward path를 단순화하여 152개의 layer를 쌓아 정확도를 크게 향상시켰다.

하나의 GPU에서 모델을 학습시킬 때와는 달리, 여러 개의 GPU에서 하나의 모델을 학습시킬 때에는 추가적인 작업이 필요하다. Data Parallel 방식은 Master GPU가 존재하여, 나머지 GPU들에게 데이터를 분배하며(scatter), 필요한 경우 데이터를 전송받아(gather) 모델에 대한 연

산을 수행한다. Distributed Data Parallel 방식은 Master GPU 없이 각각의 GPU 안의 프로세스가 독립적으로 하나의 모델을 학습시키는 방법이다.

3. 학습 및 평가 데이터

학습 및 평가에는 CIFAR-10[7] 데이터셋을 사용하였다. 데이터셋은 60,000개의 32×32 컬러 이미지이다. 클래스는 총 10개로 한 클래스에 6,000개의 이미지를 가진다. 학습시 50,000개의 이미지로 학습하였으며 10,000개의 이미지를 통해 평가하였다.

4. 제안 방법

4.1 PyramidNet

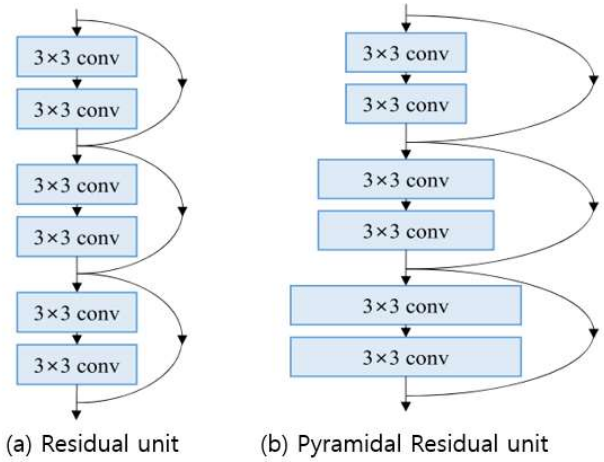

그림 1. ResNet의 Residual unit과 PyramidNet Residual unit 비교.

ResNet은 ILSVRC-2015 에서 우승한 모델로 기존의 degradation 문제를 해결하고 152 개의 layer를 가지는 deep CNN을 구현하였다. ResNet은 residaul block으로 이루어져 있는데 residual block은 gradient의 역전파가 잘 일어날 수 있도록 지름길(skip connection)을 가지고 있다. ResNet은 특정 layer에서 다운 샘플링을 수행하게 되는데 다운 샘플링시 특징 맵의 크기를 반으로 줄이고 특징 맵의 채널을 2배로 증가시키게 된다. 다운 샘플링이 수행되는 layer가 전체 성능에 큰 역할을 하게 되는데 이러한 점에서 착안하여 특정 layer에서 다운 샘플링을 하는 것이 아닌 모든 층에서 다운 샘플링을 하는 PyramidNet이 제안되었다. PyramidNet 은 특정 layer에서 일어난 width의 변화를 전체 네트워크로 분산시킴으로써 성능향상을 이루었다. 그림1에서와 같이 pyramidal residual unit은 layer를 통과할 때 마다 다운 샘플링을 수행한다.

PyramidNet에서 layer의 width를 늘리는 방법은 2가지로, additive 하게 증가시키는 방법과 multiplicative 하게 지수배로 늘리는 방법이 있다. 그림 2에서 (a)와 같이 additive 한 방식이 성능이 좋게 나오는데 초기 layer의 width가 클수록 성능이 향상되었기 때문이다.

본 논문에서 제안한 모델은 PyramidNet의 additive한 방식을 채택하였다. baseline으로 Single-GPU를 사용한 PyramidNet을 사용하였고 비교할 Multi-GPU모델로 Data parallel 기법을 사용한 Data Parallel + PyramidNet과 Distribute data parallel 기법을 사용한 Distribute Data Parallel + PyramidNet을 제안한다.

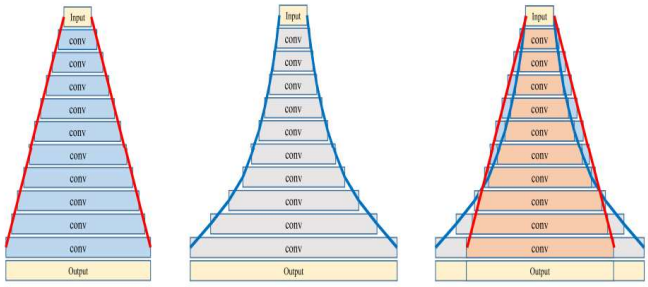

그림 2. layer 수 증가 방법 (a) additive, (b) multiplicative, c) a와 b의 비교.

4.2 Data Parallel + PyramidNet

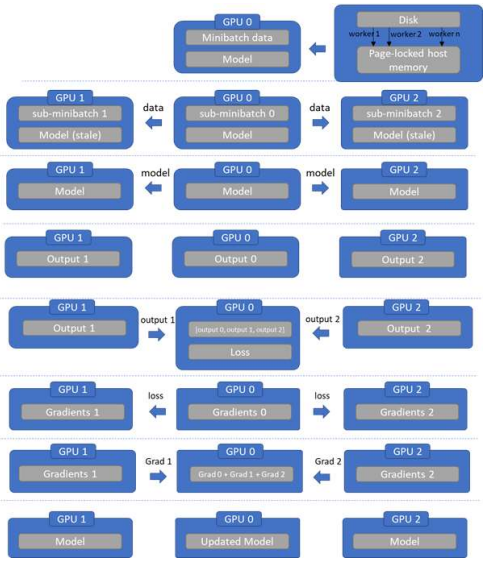

그림 3. Data Parallel 방식의 모델 학습 과정

Data Parallel + PyramidNet 방법은 PyramidNet 모델을 Multi-GPU 환경에서 학습시킨다. 0번 GPU가 Master GPU 역할을 한다. Master GPU는 학습 과정에서 생기는 계산의 분배와 데이터 전송을 관리한다. Master GPU는 나머지 GPU들에게 mini-batch 를 분배하고 모델을 복사하여 전송한다. 각 GPU들은 forward 연산을 각자 수행한 후, 계산된 output값을 Master GPU에게 전송한다. Master GPU는 전송받은 output값으로 loss값을 계산하고 다시 나머지 GPU들에게 전송한다. 각 GPU는 loss값을 이용해 backward 연산을 각자 수행한 후, 계산된 gradient값을 Master GPU에게 전송한다. Master GPU는 전송받은 gradient값으로 모델을 update한다.

4.3 Distributed Data Parallel + PyramidNet

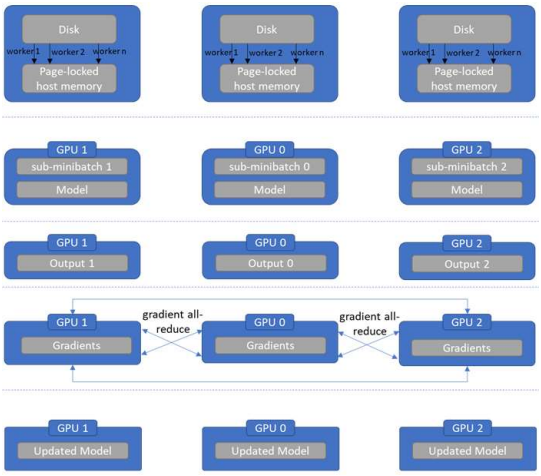

그림 4. Distributed Data Parallel 방식의 모델 학습 과정

Distributed Data Parallel + PyramidNet 방법은 Data Parallel + PyramidNet 방법처럼 PyramidNet 모델을 Multi-GPU 환경에서 학습시킨다. 그러나 Data Parallel 방법과는 다르게 Master 역할을 하는 GPU가 존재하지 않는다. 데이터를 GPU에 로드하는 것부터 forward, backward 연산까지 각각의 GPU가 병렬적으로 수행한다. 수행 과정은 다음과 같다. 먼저 각 GPU들에서 실행되는 프로세스들은 각자 호스트에서 데이터를 병렬적으로 로드하고 모델을 복사한다. 각 GPU는 다른 GPU와 상관없이 독립적으로 forward 연산을 수행하고 loss와 gradient 값을 계산한다. gradient값을 구하면 동기화 과정을 통해 다른 GPU들에게 gradient값을 넘겨준다. 각 GPU는 모두 동일한 gradient값을 가지므로, 동기화하는 과정없이 각각 모델을 업데이트한다.

5. 실험 및 결과

PyramidNet 모델을 Single-GPU를 이용해 학습한 환경, 2개의 GPU와 data parallel 이용해 학습한 multi-GPU 환경, 2개의 GPU와 distributed-data parallel을 이용해 학습한 multi-GPU 환경으로 세 가지로 분류하여 배치 사이즈를 바꾸어가며 성능 비교를 수행하였다.

실험에 사용한 GPU 환경은 NVIDIA TITAN Xp 이며, 사용한 PyramidNet 모델 학습의 경우 배치 사이즈가 256일 경우 GPU 메모리 최대 용량인 11GB를 차지한다. 만약 배치 사이즈가 256을 초과하면 Out of memory error가 발생하여 GPU를 이용한 학습이 제한되었다. 하지만 2개의 동일한 GPU와 Data parallel 혹은 Distributed data parallel 방법을 사용해 2개의 GPU 메모리를 사용함으로써 PyramidNet 모델을 배치 사이즈를 256이상으로 학습시켜보았다.

이 세 가지 방법을 배치 사이즈 별로 비교해 보면 다음 그림 5와 같다.

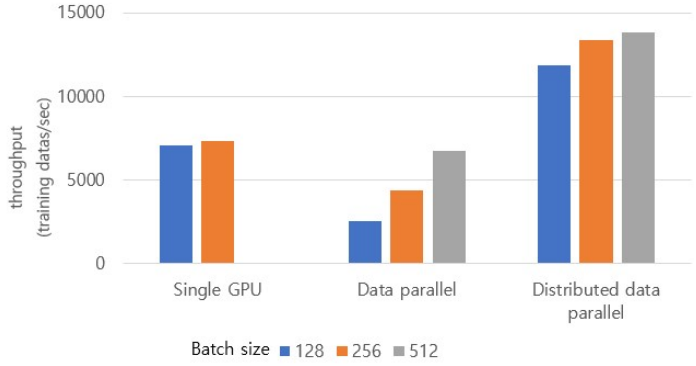

그림 5. Single GPU, Data parallel multi-GPU, Distributed parallel multi-GPU의 배치 사이즈에 따른 throughput 측정

Single-GPU 환경에서는 배치 사이즈가 증가하면 throughput이 조금 증가했으며 PyramidNet 모델의 경우에는 배치 사이즈가 512 이상이 되면 Out of memory error가 발생하여 수행할 수 없었다.

Data parallel multi-GPU의 경우에는 2개의 GPU가 batch를 둘로 나누어 수행하지만 master GPU에게만 메모리 할당이 많이 되고 master GPU에게 데이터 처리가 편중되어 throughput이 잘 나오지 않았다. batch가 512 정도 되어야 Single-GPU에서 batch 256과 비슷한 throughput을 내는 것을 관찰할 수 있었다.

Distributed data parallel multi-GPU의 경우에는 배치 사이즈가 작을 때와 클 때 모두 2개의 GPU에서 균등하게 나누어 병렬적으로 연산하므로 throughput이 이전의 경우들 보다 훨씬 높게 나왔으며 배치 사이즈가 커질수록 성능이 더 잘나오는 경향을 보였다.

6. 결 론

본 논문에서는 single-GPU와 Multi-GPU 환경에서 Data Parallel과 Distributed Data Parallel 방법으로 PyramidNet 모델을 학습시켜보며 학습 환경에 따른 학습 속도 차이를 분석해보았다. Distributed Data Parallel 방법은 Data Parallel 방법과 달리, 병렬적으로 처리하는 부분을 늘리고 속도 저하에 주요한 요인인 프로세스 간 통신 및 동기화 과정이 적어서 학습 속도가 더 빨랐다. 향후 연구 주제로는 단일 서버가 아닌 멀티 서버의 멀티 GPU를 이용한 속도 향상 방법 등이 있다.

참 고 문 헌

[1] Bottou, Léon. “Large-scale machine learning with stochastic gradient descent.“ Proceedings of COMPSTAT’2010. Physica-Verlag HD, 2010. 177-186

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3):211-252, 2015

[3] Han, Dongyoon, Jiwhan Kim, and Junmo Kim. “Deep pyramidal residual networks.“ Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[4] Behnke, Sven. Hierarchical neural networks for image interpretation. Vol. 2766. Springer, 2003.

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet classification with deep convolutional neural networks.“ Advances in neural information processing systems. 2012.

[6] He, Kaiming, et al. “Deep residual learning for image recognition.“ Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[7] Krizhevsky, Alex, and Geoffrey Hinton. Learning multiple layers of features from tiny images. Vol. 1. No. 4. Technical report, University of Toronto, 2009.