리뷰 - ImageNet in 1 Hour

임도형
최종 수정: Jul 08, 2018

개요

원제 : Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour

다수의 워커를 사용하여 학습을 빠르게 하는 방법.

분산하지 않을 때 비교하여 성능이 떨어지지 않는다.

hyper parameter 튜닝하지 않아도 된다.

2017년도 6월 논문

이론적인 내용 보다는 실험적인 결과의 내용.

결론

네트워크 구조나 그런것에 대한 것이 아닌, 분산된 woker에서의 학습하는 노하우이다.

복수 GPU에서도 적용 가능하다.

기존 모델에 쉽게 적용할 수 있다.

복수의 서버로 적용하려면 분산컴퓨팅 기술을 사용해야 한다.

고찰

분산된 노드에서 실 학습하는 방법을 제시한다. 특별한 것 없이 가져다 사용할 수 있다.

적용 가능 최대 사이즈는 8K이다. 요건 경험적인 것. 그렇다면 다른 경우라면 다른 제약이 있는 것 아닌가.

학습을 진행해 나가면서 학습율을 줄이는 것이 일반적인 방법과 상반되지 않는다.

다만 기반 학습율을 worker의 수만큼 크게 하라는 것.

관련하여 이런 제목의 논문도 있다. DON’T DECAY THE LEARNING RATE, INCREASE THE BATCH SIZE

방법

- 여러 worker에 분산학습. 각 worker에는 mini batch로.
- 다만 wokrer수만큼 학습율을 크게 한다. Linear Scaling Rule
- gradient를 평균으로 취합해서 웨이트 업데이트
- 기타 기법들 적용
    - L2 정규화
    - 모멘텀
    - gradient 취합시, kn으로 평균.

일반 SGD, 제안하는 SGD

다음이 일반 SGD

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{x \in \mathcal{B}} \nabla l(x, w_t)$$

제안하는 SGD

$$\hat{w}_{t+1} = w_t - \hat{\eta} \frac{1}{kn} \sum_{j<k} \sum_{x \in \mathcal{B}_j} \nabla l(x, w_t)$$

Distributed Synchronous SGD

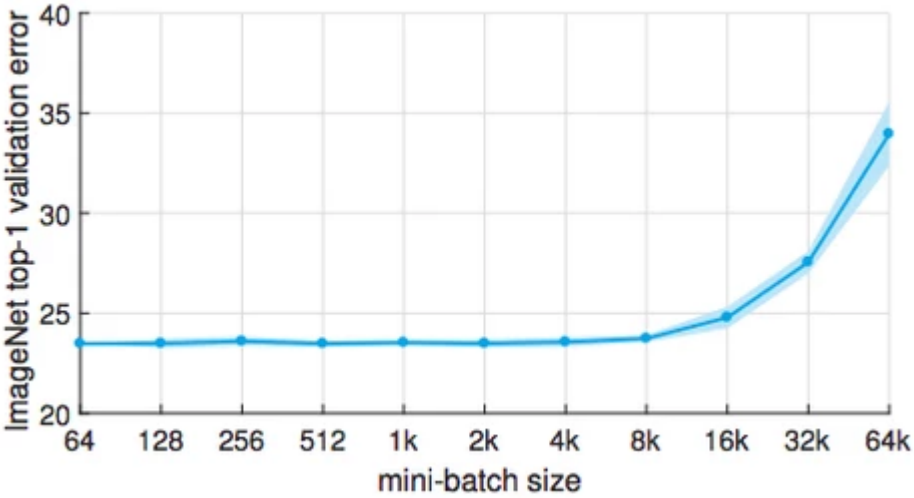
여러 worker에 각 mini batch를 보내고,

각 workersms gradient를 구하고,

이를 전부 취함하여 웨이트를 업데이트 한다.

실험 결과 요약

성능 저하 없음



mini batch 8K까지 성능 하락이 없었음.

시간 단축

ImageNet 데이터를 ResNet-50 모델로 학습.

8000 mini batch 사이즈로 256개 GPU를 사용하여 1시간 걸림.

256 mini batch 사이즈인 경우 8 GPU를 사용하여 29시간 걸림.

hyper parameter free

hyper parmaeter 를 튜닝하지 않음.

validation error, training error

256 mini batch 사이즈 수준의 validation error와 training error를 보임

Linear Scaling Rule

mini batch 사이즈를 n배 하면, 학습율을 n배 한다.

논문에서는 놀랍도록 효과가 좋다고 한다.

다른 연구에서 제안한 방법.

2가지 예외

학습 초기에 네트워크가 급격히 업데이트될 때. 이 경우 Warumup 전략을 사용.

무제한의 mini batch 사이즈가 될 수 없다. ImageNet의 경우 실험적으로 8k가 최대.

Warmup Strategy

초기 최적화 어려움(early optimization difficulties)를 극복하기 위해 warmup 전략을 사용함.

학습 시작 시에 낮은 학습율을 사용는 것.

constant warmpup과 gradual wrampup이 있다 하는데, 실험적으로 constant는 안좋다고.

기반 mini batch사이즈의 k배의 사이즈일 경우(8k인경우 k는40(=8K/256)) 40 epoch에 걸쳐 학습률을 증가시키다.

이후 Linear Scaling Rule을 사용.

일반화 가능

다양한 ML 프레임웤에 사용 가능

다양한 분야에 사용 가능.

기존 알고리즘의 복수 GPU 사용이 용이

Batch Normalization with Large MiniBatch

실 구체 사용 방법

Weight Decay

L2 Regularization을 적용한다.

$$l(x, w) = \frac{\lambda}{2}\|w\|^2 + \varepsilon(x, w)$$

모멘텀

모멘텀을 사용한다. η는 학습율

$$v_{t+1} = m\frac{\eta_{t+1}}{\eta_t}v_t + \eta_{t+1}\frac{1}{n}\sum \nabla l(x, w_t)$$

Gradient 취합

worker들의 gradient를 평균내어 사용.

이 때 worker들의 갯수 n이 아닌, mini batch 사이즈 k까지 포함한 kn으로 나누어라.

$$\frac{1}{kn}\sum_j \sum_{x\in\mathcal{B}_j} l(x, w_t)$$

Data 섞기

하나의 epoch에 전체 데이타에 대하여 단일 셔플링을 하라.

각 worker별로 셔플링하면 안된다.

Communication Cost

MPI allreduce가 무시할 수 없는 버틀넥.

- MPI : Message Pass Interface, 분산 계산에서 사용되는 표준 메시지 전달 방식
- allreduce : 분산 노드의 계산 결과를 수집하고 그결과를 각 노드에 전달하는 오퍼레이션

다음 방식으로 극복.

- 같은 물리적 서버에 잇는 GPU의 결과를 1개의 버퍼에 모음.
- 이를 중앙 서버로 보내서 allreduce 연산 실행
- 받은 결과를 각 GPU에 전달

intra server

GPU와 CPU간의 카피시에,

256K 미만의 경우 NCCL(NVIDIA Collective Communication Library)를 사용.

아닌 경우 CPU reduction(?) 사용.

inter server

- recursive halving and doubling 알고리즘과 bucket 알고리즘 사용.
- 서버 수가 2의 자승이 아닌 경우 binay block 알고리즘 사용

SW, HW

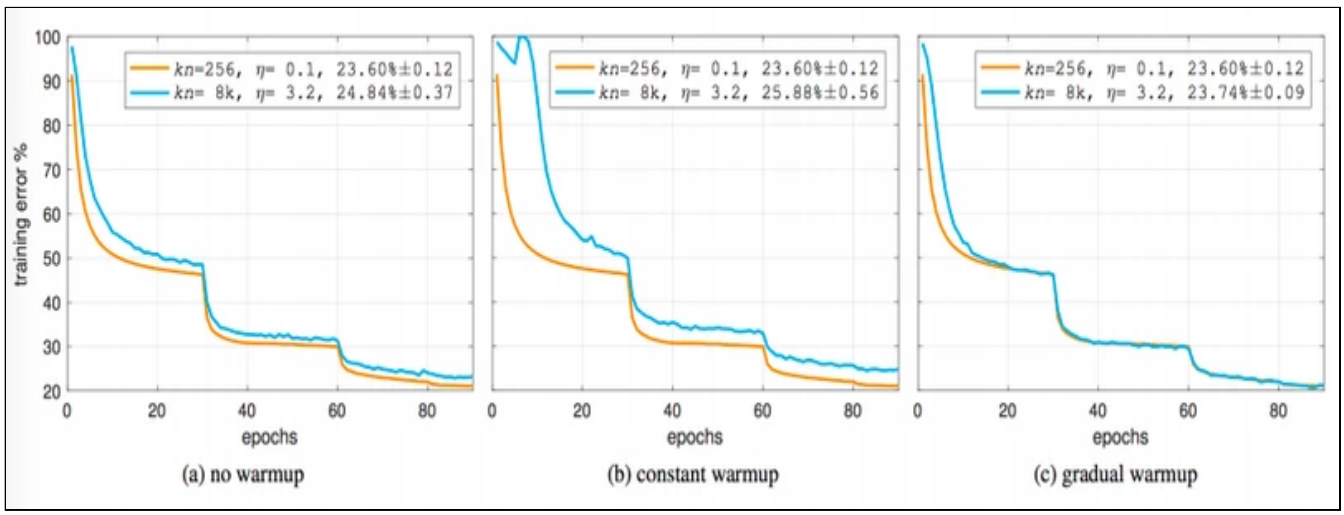
- Gloo로 구현한 allreduce 라이브러리 사용.
- Caffe2 사용. multi thread 지원
- Facebook의 8GPU가 달린 Big Basin 사용.
- ResNet ImageNet 일경우 15G이상의 네트웤 필요.

실험 상세

- ResNet50 train on ImageNet-1k (1.28 million images)
- Momentum SGD, batch size n=32
- 학습율 = 0.1 * kn / 256 (linear scaling rule)
- Baseline: k=8GPU, n=32, top-1 validation error=23.6%
- 서버 수 : 8 ~ 256(1 to 32 Big Basins)
- median error 사용

실험 결과

학습 초기



gradual wramup으로 초기 학습의 문제 없다. 그런데 사용하지 않았을 때도 별 문제 없는 것 같은데.

성능 저하

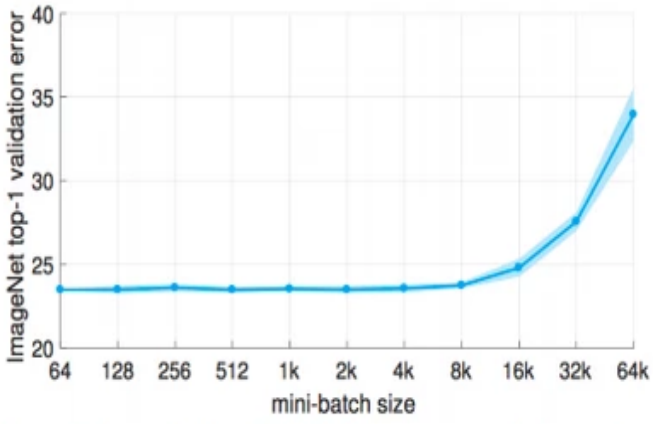


Figure 1. **ImageNet top-1 validation error *vs.* minibatch size.**

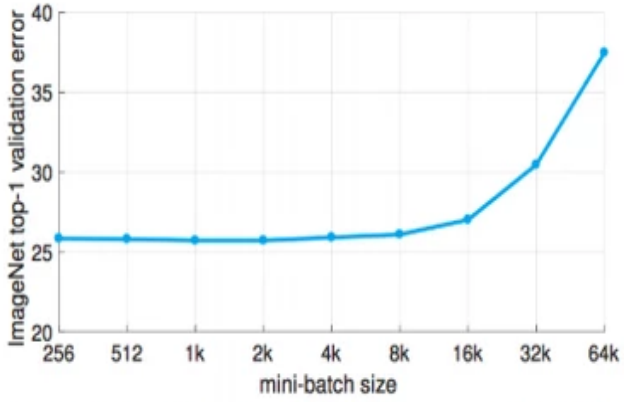


Figure 6. **ImageNet-5k** top-1 validation error *vs.* minibatch size

mini batch 사이즈 8K까지는 성능 저하 없다.

학습율에 따라

kn	η	top-1 error (%)
256	0.05	23.92 ±0.10
256	0.10	23.60 ±0.12
256	0.20	23.68 ±0.09
8k	$0.05 \cdot 32$	24.27 ±0.08
8k	$0.10 \cdot 32$	23.74 ±0.09
8k	$0.20 \cdot 32$	24.05 ±0.18
8k	0.10	41.67 ±0.10
8k	$0.10 \cdot \sqrt{32}$	26.22 ±0.03

학습율 자체는 경험적으로 찾아야 한다.

다른 태스크

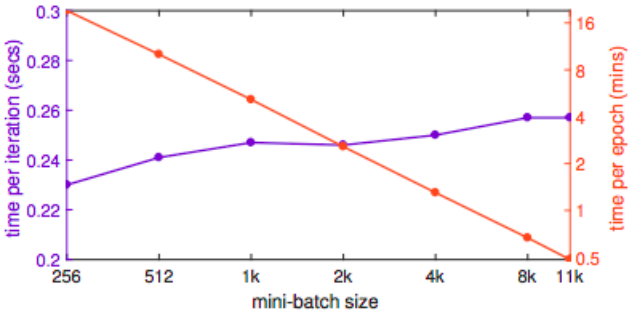
ImageNet pre-training			COCO	
kn	η	top-1 error (%)	box AP (%)	mask AP (%)
256	0.1	23.60 ±0.12	35.9 ±0.1	33.9 ±0.1
512	0.2	23.48 ±0.09	35.8 ±0.1	33.8 ±0.2
1k	0.4	23.53 ±0.08	35.9 ±0.2	33.9 ±0.2
2k	0.8	23.49 ±0.11	35.9 ±0.1	33.9 ±0.1
4k	1.6	23.56 ±0.12	35.8 ±0.1	33.8 ±0.1
8k	3.2	23.74 ±0.09	35.8 ±0.1	33.9 ±0.2
16k	6.4	24.79 ±0.27	35.1 ±0.3	33.2 ±0.3

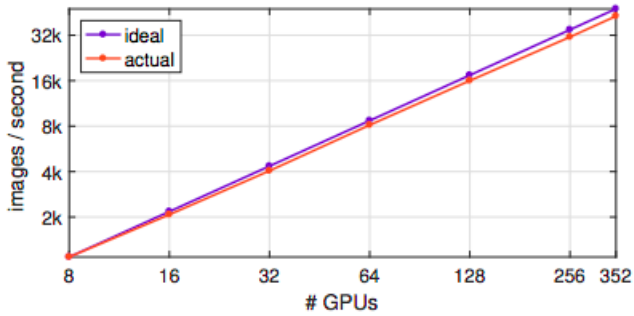
Transfer Learning에도 잘된다.

# GPUs	kn	$\eta \cdot 1000$	iterations	box AP (%)	mask AP (%)
1	2	2.5	1,280,000	35.7	33.6
2	4	5.0	640,000	35.7	33.7
4	8	10.0	320,000	35.7	33.5
8	16	20.0	160,000	35.6	33.6

segmentation도 잘된다.

속도





1 iteration이 0.22 ~ 0.26. mini batch 사이즈가 커질 수록 커지는 경향.

1 epoch당 시간은 선형적으로 감소. 8K의 경우 0.5분 소요.

GPU 갯수와 처리 이미지 수는 거의 선형적.8K(=256=8*32)인 경우 초당 32K개 처리.

Reference

- paper : https://arxiv.org/pdf/1706.02677.pdf
- 관련 아티클 : https://towardsdatascience.com/deep-learning-at-scale-accurate-large-mini-batch-sgd-8207d54bfe02. 논문 읽다 만듯.
- 설명 자료 : https://hma02.github.io/AllanMa/assets/pdf/2017-7-31-imagenet-in-1h.pdf

- 이후 논문
  - Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes, https://www.preferred-networks.jp/docs/imagenet_in_15min.pdf
  - DON'T DECAY THE LEARNING RATE, INCREASE THE BATCH SIZE : https://openreview.net/pdf?id=B1Yy1BxCZ