영어에서 한국어로 번역 - www.onlinedoctranslator.com

arXiv:1706.02677v2 [cs.CV] 2018년 4월 30일

정확하고 대규모 미니배치 SGD: 1시간 만에 ImageNet 교육

프리야 고얄 피오트르 달러르 로스 거식 피터 노르트하우스
루카스 웨솔로프스키 아포 키롤라 앤드류 툴로크 양칭 지아 카이밍 헤

페이스북

추상적인

딥 러닝은 대규모 신경망과 대규모 데이터세트를 통해 성공할 수 있습니다. 그러나 더 큰 네트워크와 더 큰 데이터 세트로 인해 훈련 시간이 길어져 연구 개발 진행이 지연됩니다. 분산형 동기식 SGD는 SGD 미니 배치를 병렬 작업자 풀로 나누어 이 문제에 대한 잠재적인 솔루션을 제공합니다. 그러나 이 체계를 효율적으로 만들려면 작업자당 작업 부하가 커야 하며 이는 SGD 미니배치 크기가 적지 않게 증가함을 의미합니다. 이 논문에서는 ImageNet 데이터세트에서 큰 미니 배치가 최적화에 어려움을 야기한다는 것을 경험적으로 보여줍니다. 그러나 이러한 문제가 해결되면 훈련된 네트워크는 좋은 일반화를 나타냅니다. 특히 최대 8192개 이미지의 대규모 미니배치 크기로 학습할 때 정확도 손실이 없음을 보여줍니다. 이 결과를 달성하려면, 우리는 미니배치 크기에 따라 학습률을 조정하기 위한 초매개변수 없는 선형 확장 규칙을 채택하고 훈련 초기에 최적화 문제를 극복하는 새로운 워밍업 방식을 개발했습니다. 이러한 간단한 기술을 사용하여 Caffe2 기반 시스템은 작은 미니배치 정확도를 일치시키면서 1시간 안에 256개의 GPU에서 8192의 미니배치 크기로 ResNet-50을 교육합니다. 상용 하드웨어를 사용하여 구현

달성하다~8GP에서 256GP로 이동할 때 90% 확장 효율성 우리를. 우리의 fi N킹스 활성화 르 훈련 V실제 인식 니션 모델 인터넷에서 - 규모 데이터 높은 ef로 능숙함.

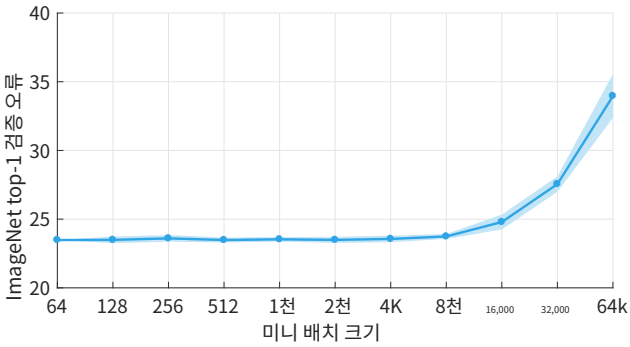


그림 1.ImageNet top-1 검증 오류*대.* 미니 배치 크기. 플러스/마이너스의 오차 범위*둘*표준편차가 표시됩니다. 분산 동기 SGD를 최대 8k 이미지의 미니 배치로 확장하는 간단하고 일반적인 기술을 제시합니다.*Small minibatch training의 top-1 error를 유지하면서.* 모든 미니배치 크기에 대해 학습률을 다음과 같이 설정했습니다.*선의*미니배치 크기의 함수를 사용하고 훈련의 처음 몇 epoch 동안 간단한 준비 단계를 적용합니다. 다른 모든 하이퍼 매개변수는 고정된 상태로 유지됩니다. 이 간단한 접근 방식을 사용하면 모델의 정확도가 미니배치 크기(최대 8k 미니배치 크기)에 따라 변하지 않습니다. 우리의 기술을 사용하면 훈련 시간을 선형적으로 줄일 수 있습니다.-대규모 미니배치 크기로 확장 시 효율성이 90% 향상되어 256 GPU에서 1시간 만에 정확한 8k 미니배치 ResNet-50 모델을 교육할 수 있습니다.

1. 내부 소개 N

스케일 필요해요. 우리는 예상치 못한 찌그러진 시대 AI에서 연구하다 나 역사 나는 N 어느 일 이점 형 데이터와 중오델 스케일 나 빨리 나는 중증명하다 정확도 컴퓨터 V이온 [22,41 , 34,35,삼 6, 16], 스피 채널 [17,40], 그리고 자연 난 란- 게이지 p 처리 [7,38]. 가져가다 프로포우 Nd 영향 com- 푸터 v 으로의 존재 예: v 이슈 대표 엔테이션 르 아르네드 깊은 곳에서 회선 최종 신경 n 에트웍스 [2 삼, 22] 보여주다 에서 형 빌려준 PE 성능 영형 이전 y 도전 N 나음과 같은 작업 이마- 지넷C 분류 N [33] 그리고 새 랜스젠더야 디로 이동 예 의 지각 이온 문제 중와 같은 영형상 감지 tion 그리고 se g남자들- 떠이온 [8,10 , 28]. 더 이상, 이 아빠 일반 세대 리즈: 크다 아르 자형 다taset 및 신경망 승오크 건축가 써 ures consi 열심히 굴복하다 디 네 개선됨 새 라시 아크 영향 모든 작업 에그 베네 fit 사전부터 - 떠이가 [22 , 41,34,삼 5, 36,16]. 하지만 모처럼 다엘과 닷 나 예제일 성장, 트레이도 마찬가지야 N시간; 다발견 중 타 잠재력 엘 nd 한계 o f 대규모 이점 행닝 요구하고 있다 에 발전 중 g N오벨 테크니 키에 대한 질문 피훈련 티 중전자 관리 블레. 목표 이 저장소의 아 데 하는 게 아니야 f를 설명하다 용이성 , com에 nd 중 하나를 하나로 묶다 파 용적인 구 나 에, 큰 - 규모 열차 - 니 ist와 함께 ng 늑골이 있는 신 새 머리의 스토 차스틱그라 다 동양 하강 티 (SGD). 처럼 예를 들어, 우리는 확장한다 아르 esNet-50 [16] 훈련 , 영형 해는 p 이철 형 성했다 그건 미니브야 tch 크기 256 이미지 에스 (8Te를 사용하여 에서 P100G 파 리, 훈련소 N g 시간은 2 9시간), t 영형 엘 르거 미니 버 착 (참조 수치1). 특히 아, 우리 쇼 승 다 자와 *대형 미니 배치 크기 8192, 승* e는 훈련할 수 있다 N 아 르 esNet-50 i n 1시간 너 *256G를 노래하다 PU는 동안 유지하다 g*

256 미니배치 기준선과 동일한 수준의 정확도. 이제 분산형 동기 SGD가 보편화되었지만, 8192만큼 큰 미니배치를 사용하여 일반화 정확도를 유지할 수 있거나 그러한 높은 정확도 모델을 그렇게 짧은 시간에 훈련할 수 있다는 기존 결과는 없습니다.

이 비정상적으로 큰 미니배치 크기를 해결하기 위해 우리는 간단하고 하이퍼 매개변수가 없는 미니 배치를 사용합니다. *선형 스케일링 규칙*학습률을 조정합니다. 이 지침은 이전 연구에서 발견되었지만 [21,4], 그 경험적 한계는 잘 이해되지 않았으며 비공식적으로 우리는 그것이 연구계에 널리 알려지지 않았다는 것을 발견했습니다. 이 규칙을 성공적으로 적용하기 위해 우리는 새로운 방법을 제시합니다. *워밍업*전략, *즉*., 훈련 시작 시 낮은 학습률을 사용하는 전략 [16], 초기 최적화 문제를 극복하기 위해. 중요한 것은 우리의 접근 방식이 기준선과 일치할 뿐만 아니라 *확인*오류, *또한 작은 미니배치 기준선과 거의 일치하는 훈련 오류 곡선을 생성합니다.*자세한 내용은 §에 나와 있습니다.2.

§의 포괄적인 실험5그것을 보여줘 *최적화*어려움은 열악한 것보다는 대규모 미니 배치의 주요 문제입니다. *일반화*(적어도 ImageNet에서는), 일부 최근 연구와는 달리 [20]. 또한 선형 확장 규칙과 워밍업이 객체 감지 및 인스턴스 분할을 포함한 보다 복잡한 작업으로 일반화됨을 보여줍니다.9,31,14,28], 최근 개발된 Mask R-CNN을 통해 시연합니다.14]. 광범위한 미니배치 크기를 다루기 위한 강력하고 성공적인 지침은 이전 연구에서 제시되지 않았습니다.

우리가 제공하는 전략은 간단하지만 이를 성공적으로 적용하려면 딥 러닝 라이브러리 내에서 사소해 보이고 종종 잘 이해되지 않는 구현 세부 사항과 관련하여 올바른 구현이 필요합니다. SGD 구현의 미묘함으로 인해 발견하기 어려운 잘못된 솔루션이 발생할 수 있습니다. 보다 유용한 지침을 제공하기 위해 §에서 이러한 함정을 유발할 수 있는 일반적인 함정과 관련 구현 세부 사항을 설명합니다.삼.

우리의 전략은 프레임워크에 관계없이 적용되지만 효율적인 선형 확장을 달성하려면 사소한 통신 알고리즘이 필요합니다. 우리는 오픈 소스를 사용합니다*카페2*[1]
딥러닝 프레임워크와 *큰 분지*GPU 서버 [24](특수 네트워크 인터페이스와 반대) 표준 이더넷 네트워킹을 사용하여 효율적으로 작동합니다. 우리는 §에서 우리의 접근 방식이 최대한의 잠재력을 발휘할 수 있도록 하는 시스템 알고리즘을 설명합니다.4.

이 보고서에 설명된 실질적인 발전은 다양한 영역에 걸쳐 도움이 됩니다. 산업 영역에서 우리 시스템은 인터넷 규모의 데이터로부터 시각적 모델을 교육할 수 있는 잠재력을 발휘하여 다음과 같은 교육을 가능하게 합니다. *하루에 수십억 개의 이미지*. 마찬가지로 중요한 것은 연구 영역에서 하이퍼 매개변수 검색 없이 단일 GPU에서 다중 GPU 구현으로 알고리즘을 마이그레이션하는 것을 단순화한다는 것입니다. *예를 들어*. Faster R-CNN 마이그레이션 경험에서 [31] 및 ResNets [16] 1~8개의 GPU.

[1] http://www.caffe2.ai

2. 대형 미니배치 SGD

우리는 다음 섹션에서 논의의 기초가 될 확률적 경사하강법(SGD)의 공식을 검토하는 것부터 시작합니다. 손실을 최소화하여 지도 학습을 고려합니다. *엘*(*승*)형식:

$$엘(승) = \frac{1}{|X|}\sum_{엑스\in 엑스} 엘(x, w). \tag{1}$$

여기 *승*는 네트워크의 가중치이고, *엑스*레이블이 지정된 훈련 세트이고, *엘*(*x*, *w*)샘플에서 계산된 손실입니다. *엑스*∈*엑스* 그리고 그들의 라벨 *와이*. 일반적으로 *엘*분류 손실의 합입니다(*예를 들어*., 교차 엔트로피) 및 정규화 손실 *승*.

미니배치 확률적 경사하강법[32](최근 문헌에서는 간단히 SGD라고 함)는 미니배치에서 작동하지만 다음 업데이트를 수행합니다.

$$승_{티+1} = 승_{티} - 에타 \frac{1}{N}\sum_{엑스\in B} \nabla 엘(x, w_{티}) \tag{2}$$

여기 *비*는 *미니배치*샘플링된 *엑스*그리고 *N*=|*비*|은 *미니배치 크기*, *에타*은 *학습률*, 그리고 *티*반복 인덱스입니다. 실제로 우리는 모멘텀 SGD를 사용합니다. §에서 추진력에 대한 논의로 돌아갑니다.삼.

2.1. 대규모 미니배치의 학습률

우리의 목표는 작은 미니배치 대신 큰 미니배치를 사용하는 것입니다. *훈련 및 일반화의 정확성 유지*. 이는 분산 학습에서 특히 흥미롭습니다. 여러 작업자로 확장할 수 있기 때문입니다.[2]작업자당 작업량을 줄이지 않고 모델 정확도를 저하시키지 않고 간단한 데이터 병렬 처리를 사용합니다.

포괄적인 실험에서 보여주겠지만, 다음과 같은 학습 속도 확장 규칙이 광범위한 미니배치 크기에 대해 놀라울 정도로 효과적이라는 것을 발견했습니다.

> *선형 확장 규칙:미니배치 크기에 다음을 곱하면케이, 학습률에 다음을 곱합니다.케이.*

기타 모든 초매개변수(가중치 감소, *등*.)은 변경되지 않고 유지됩니다. §에서 보여주듯이5, *선형 스케일링 규칙*작은 미니배치와 큰 미니배치를 사용하는 것 사이의 정확도를 일치시킬 뿐만 아니라, 마찬가지로 중요하게는 훈련 곡선을 크게 일치시키는 데 도움이 됩니다. 이를 통해 수렴 전에 신속한 디버깅과 실험 비교가 가능합니다.

해석.우리는 선형 스케일링 규칙과 그것이 효과적인 이유에 대한 비공식적 토론을 제시합니다. 반복 시 네트워크 고려 *티*가중치가 있는 *승*$_{티}$, 그리고 일련의 *케이*미니배치 *비*$_{제이}$~을 위한 $0 \le j < k$각각의 크기 *N*. 실행 효과를 비교합니다. *케이*SGD 반복 *작은 미니배치 비*$_{제이}$학습률 *에타*단일 반복과 *대형 미니배치* $\cup_{제이}$ *비*$_{제이}$크기의 *kn*학습률 *etâ*.

[2] 이 작업에서는 '워커'와 'GPU'라는 용어를 같은 의미로 사용하지만 '워커'의 다른 구현도 가능합니다. '서버'는 네트워크를 통한 통신이 필요하지 않은 GPU 8개 세트를 의미합니다.

에 따르면 (2), 후에*케이*학습률을 사용한 SGD 반복*에타*그리고 미니배치 크기는*N*우리는:

$$승_{타+케이}=승_{티}-에타\frac{1}{N}\sum_{j<kx\in B_{제이}}\sum\nabla엘(x,w_{타+제이}). \quad (삼)$$

한편, 대형 미니배치로 한걸음 더 나아가∪제이비제이크기의*kn* 학습률*eta*수익률:

$$\hat{w}_{타+1}=승_{티}-\hat{\theta}\frac{1}{kn}\sum_{j<kx\in B_{제이}}\sum\nabla엘(x,w_{비}). \quad (4)$$

예상대로 업데이트가 다르며~*할 것 같지 않은*저것 $\hat{w}_{타+1}=승_{타+케이}$. 그러나 만일 우리가~*할 수 있었다*추정하다 $\nabla엘(x,w_{티})\ \nabla\nabla엘(x,w_{타+제이})$~을 위한$j<k$, 설정*eta*=*km*양보할 것이다 $\hat{w}_{타+1}\fallingdotseq승_{타+케이}$, 소규모 및 대규모 미니배치 SGD의 업데이트는 유사합니다. 이는 강력한 가정이지만 이것이 사실이라면 두 업데이트가 유사하다는 점을 강조합니다.*오직*우리가 설정하면 *eta*=*km*.

위의 해석은 선형 스케일링 규칙이 적용되기를 바라는 한 가지 경우에 대한 직관을 제공합니다. 우리의 실험에서는*eta*=*km*(및 준비), 소형 및 대형 미니배치 SGD는 최종 정확도가 동일한 모델을 생성할 뿐만 아니라 훈련 곡선도 밀접하게 일치합니다. 우리의 경험적 결과는 위의 근사치가 ~*할 것 같다*대규모의 실제 데이터에서 유효해야 합니다.

그러나 조건이 충족되는 경우는 적어도 두 가지가 있습니다. $\nabla엘(x,w_{티})\ \nabla\ \nabla엘(x,w_{타+제이})$분명히 유지되지 않습니다. 첫째, 네트워크가 빠르게 변화하는 초기 훈련에서는 유지되지 않습니다. 우리는 § 에서 논의된 준비 단계를 사용하여 이 문제를 해결합니다.2.2. 둘째, 미니배치 크기는 무한정 확장될 수 없습니다. 결과는 다양한 크기 범위에서 안정적이지만 특정 지점을 넘으면 정확도가 급격히 저하됩니다. 흥미롭게도 이 점은 다음과 같이 크다.~ ImageNet 실험에서는 8k입니다.

논의.위의 선형 스케일링 규칙은 Krizhevsky에 의해 채택되었습니다.21], 이전이 아니라면. 그러나 Krizhevsky는 미니배치 크기를 128에서 1024로 늘릴 때 오류가 1% 증가한다고 보고한 반면, 우리는 훨씬 더 광범위한 미니배치 크기 체계에서 정확도를 유지하는 방법을 보여줍니다. 첸*외.* [5]에서는 수많은 분산 SGD 변형에 대한 비교를 제시했으며 해당 작업에서도 선형 확장 규칙을 사용했지만 작은 미니배치 기준선을 설정하지 않았습니다. 리 [25] (§ 4.6)은 수렴 후 정확도 손실 없이 최대 5120개의 미니 배치를 사용한 분산 ImageNet 교육을 보여줍니다. 그러나 그들의 작업은 우리 작업의 핵심 기여인 미니 배치 크기의 함수로 학습 속도를 조정하기 위한 하이퍼 매개변수 검색 없는 규칙을 보여주지 못했습니다.

최근 작품에서는 보투(Bottou)*외.* [4] (§ 4.2)는 미니배칭의 이론적 장단점을 검토하고 선형 확장 규칙을 사용하여 솔버가 표시된 예제 수의 함수로 동일한 훈련 곡선을 따르고 학습 속도가 미니배치 크기와 관계없이 최대 속도를 초과해서는 안 된다는 것을 보여줍니다(이는 워밍업을 정당화합니다). 우리의 작업은 전례 없는 미니배치 크기로 이러한 이론을 경험적으로 테스트합니다.

2.2. 워밍업

우리가 논의한 것처럼 대규모 미니배치의 경우(*예를 들어.*, 8k) 선형 확장 규칙은 네트워크가 급격하게 변화할 때 무너지며, 이는 일반적으로 훈련 초기 단계에서 발생합니다. 우리는 이 문제가 적절한 설계를 통해 완화될 수 있음을 발견했습니다. *워밍업*[16], 즉 훈련 시작 시 덜 공격적인 학습 속도를 사용하는 전략입니다.

지속적인 워밍업.[에 제시된 워밍업 전략16]
낮은 것을 사용한다*끊임없는*훈련의 처음 몇 epoch 동안의 학습률입니다. § 에서 보여주듯이5, 우리는 객체 감지 및 분할 방법 프로토타입 제작에 특히 도움이 되는 지속적인 워밍업을 발견했습니다.9,31,26,14] 새로 초기화된 레이어와 함께 사전 훈련된 레이어를 미세 조정합니다.

크기가 큰 미니 배치를 사용한 ImageNet 실험에서*kn*, 우리는 낮은 학습률로 훈련을 시도했습니다. *에타*처음 5개 에포크 동안 목표 학습률로 돌아갑니다.*eta*=*km*. 그러나, 주어진 큰*케이*, 우리는 이 지속적인 준비 단계가 최적화 문제를 해결하는 데 충분하지 않으며 낮은 학습 속도 준비 단계에서 전환하면 훈련 오류가 급증할 수 있음을 발견했습니다. 이로 인해 우리는 다음과 같은 점진적인 워밍업을 제안하게 되었습니다.

점진적인 워밍업.우리는 대안적인 워밍업을 제시합니다. *서서히*학습률을 작은 값에서 큰 값으로 높입니다. 이 램프는 학습 속도의 급격한 증가를 방지하여 훈련 시작 시 건강한 수렴을 가능하게 합니다. 실제로는 크기가 큰 미니배치로*kn*, 우리는 학습률에서 시작합니다*에타*그리고 각 반복마다 일정한 양만큼 증가하여 다음 값에 도달합니다.*eta*=*km*5개 에포크 이후(결과는 정확한 워밍업 기간에 대해 견고함). 워밍업 후에는 원래의 학습률 일정으로 돌아갑니다.

2.3. 대규모 미니배치를 사용한 배치 정규화

배치 정규화(BN) [19]는 미니배치 차원에 따라 통계를 계산합니다. 이는 각 샘플 손실의 독립성을 깨뜨리고 미니배치 크기의 변화는 최적화되는 손실 함수의 기본 정의를 변경합니다. 다음에서는 통신 오버헤드를 피하기 위한 실질적인 고려 사항으로 보일 수 있는 일반적으로 사용되는 '단축키'가 미니배치 크기 변경 시 손실 함수를 보존하기 위해 실제로 필요하다는 것을 보여줍니다.

우리는 (1) 그리고 (2) 샘플당 손실을 가정합니다. *엘*(*x*,*w*) 다른 모든 샘플과 독립적입니다. 이것은~*아니다*BN이 수행되고 활성화가 샘플 전체에 걸쳐 계산되는 경우입니다. 우리는 쓴다*엘*$_{비}$(*x*,*w*)단일 샘플의 손실을 나타냅니다. *엑스*미니배치에 있는 모든 샘플의 통계에 따라 달라집니다.*비*. 우리는 th를 나타냅니다Σ 단일 미니배치에 대한 손실 *비*크기의*N*~처럼 *엘*(*비*,*승*) =$\frac{1}{N}\sum_{엑스\in B}$*엘*$_{비}$(*x*,*w*). BN을 이용하면 훈련 세트는 크기의 모든 개별 하위 집합을 포함하는 것으로 생각할 수 있습니다.*N*원래 훈련 세트에서 가져온 것 *엑스*, 우리는 다음과 같이 표시합니다.*엑스*$_{N}$. 훈련 손실 *엘*(*승*)그러면 다음과 같이 됩니다:

$$엘(승)=\frac{1}{|X_N|}\sum_{B\in 엑스_N}엘(비,승). \quad (5)$$

우리가 보면 *비*'단일 샘플'로 *엑스*$_N$, 각 단일 샘플의 손실 *비*계산된다 *독립적으로*.

미니 배치 크기 *N*BN 통계가 계산되는 것은 손실의 주요 구성 요소입니다. 작업자당 미니배치 샘플 크기가 *N*변경되었습니다, *기본 손실 함수를 변경합니다. 엘그건 최적화된 거야*. 보다 구체적으로 BN이 계산한 평균/분산 통계는 다음과 같습니다. *N*다양한 수준의 무작위 변동을 나타냅니다.

분산(및 다중 GPU) 교육의 경우 작업자당 샘플 크기가 *N*고정된 상태로 유지되며 전체 미니배치 크기는 다음과 같습니다. *kn*, 이는 다음의 미니배치로 볼 수 있습니다. *케이*각 샘플이 포함된 샘플 *비*$_{제이}$독립적으로 선택됨 *엑스*$_N$, 따라서 기본 손실 함수는 변경되지 않고 여전히 정의되어 있습니다. *엑스*$_N$. 이런 관점에서 BN 설정에서는
본 후 *케이*미니배치 *비*$_{제이}$, (삼) 그리고 (4) 이 되다:

$$승_{타+케이}=승_{티}-에타\sum_{j<k}\nabla 엘(비_{제이},w_{티+제이}), \tag{6}$$

$$\hat{w}_{티+1}=승_{티}-\theta\frac{1}{케이}\sum_{j<k}\nabla 엘(비_{제이},w_{티}). \tag{7}$$

§ 과 유사한 논리를 따릅니다.2.1, 우리는 설정했습니다 *eta*$\hat{}$*= km*그리고 *작업자당 표본 크기를 유지합니다. N작업자 수를 변경할 때 상수케이*.

이 작업에서 우리는 *N*=32광범위한 데이터 세트와 네트워크에서 좋은 성능을 발휘했습니다.19,16]. 만약에 *N*조정되면 분산 훈련이 아닌 BN의 하이퍼 매개변수로 보아야 합니다. 우리는 또한 *BN 통계는*~ 아니다 *모든 작업자에 대해 계산됩니다.*, 통신을 줄이는 것뿐만 아니라 최적화된 동일한 기본 손실 기능을 유지하기 위한 것입니다.

3. 분산 SGD의 미묘함과 함정

실제로 분산 구현에는 많은 미묘함이 있습니다. 많은 일반적인 구현 오류로 인해 하이퍼 매개변수의 정의가 변경되어 모델이 학습되지만 오류가 예상보다 높을 수 있으며 이러한 문제를 발견하기 어려울 수 있습니다. 아래 설명은 간단하지만 기본 솔버를 충실하게 구현하려면 명시적으로 고려하는 것이 중요합니다.

체중 감소.가중치 감소는 실제로 L2 정규화 항의 기울기 결과입니다. *손실 함수에서*. 보다 공식적으로, 샘플당 손실은 (1)는 다음과 같이 쓸 수 있습니다.
엘(x,w) =λ ₂*"승"*₂+$\varepsilon(x,w)$. 여기λ ₂*"승"*₂샘플이에요-가중치에 대한 독립적인 L2 정규화 $\varepsilon(x,w)$ 교차 엔트로피 손실과 같은 샘플 종속 항입니다. SGD 업데이트(2)는 다음과 같이 작성할 수 있습니다.

$$승_{티+1}=승_{티}-\tau\lambda 승_{티}-에타\frac{1}{N}\sum_{엑스\in B}\nabla\varepsilon(x,w)_{티} \tag{8}$$

실제로는 일반적으로 표본 종속 항만 사용됩니다.
나₩Σ $\nabla\varepsilon(x,w_{티})$역전파로 계산됩니다. 용어 $\lambda 승_{티}$계산된다 *갈라져* 집계된 그래디언트에 추가됩니다.

기여한 사람 $\varepsilon(x,w_{티})$. 가중치 감소 항이 없는 경우 항 스케일링을 포함하여 학습률을 스케일링하는 여러 가지 동등한 방법이 있습니다. $\varepsilon(x,w_{티})$. 그러나 에서 알 수 있듯이 (8) 일반적으로 이것은 ~ *아니다*경우. 우리는 이러한 관찰을 다음과 같이 요약합니다:

비고 1: 교차 엔트로피 손실을 스케일링하는 것은 ~ 아니다 *학습률을 조정하는 것과 동일합니다.*

모멘텀 수정.Momentum SGD는 바닐라 SGD에 대해 일반적으로 채택되는 수정입니다.2). 모멘텀 SGD의 참조 구현은 다음과 같은 형식을 갖습니다.

$$유_{티+1}=무_{티}+\frac{1}{N}\sum_{엑스\in B}\nabla 엘(x,w_{티}) \tag{9}$$
$$승_{티+1}=승_{티}-에타유_{티+1}.$$

여기 *중*운동량 감소 인자이고 *유*업데이트 텐서입니다. 인기 있는 변형은 학습률을 흡수합니다. *에타*업데이트 텐서에 넣습니다. 대체 *V*$_{티}$~을 위한 *에타유*$_{티}$안에 (9) 결과는 다음과 같습니다.

$$V_{티+1}=mv_{티}+에타\frac{1}{N}\sum_{엑스\in B}\nabla 엘(x,w_{티}) \tag{10}$$
$$승_{티+1}=승_{티}-V_{티+1}.$$

고정의 경우 *에타*, 둘은 동일합니다. 그러나 우리는 *유*그라디언트에만 의존하며 독립입니다. *에타*, *V*얽혀있다 *에타*. 언제 *에타*참조 변형과의 동등성을 유지하기 위해 변경합니다. Σ (9), 업데이트 *V*해야한다: $V_{티+1}=중_{에타티+1}V_{티}+에타_{티+1}\frac{1}{N}\nabla 엘(x,w_{티})$. 우리 요인을 참조 $에타_{티+1}$으로 *운동량 보정*. 이는 훈련을 안정화하는 우리 데 특히 중요하다는 사실을 발견했습니다. *에타*$_{티+1}$ *에타*$_{티}$, 그렇지 않으면 역사 용어 $V_{티}$너무 불안정성을 초래하는 작은 (예: *에타*$_{티+1}$ < *에*$_{티}$운동량 수정은 덜 중요합니다). 이것은 우리의 두 번째 발언으로 이어진다:

참고 2: 다음을 사용하는 경우 학습률을 변경한 후 운동량 보정을 적용합니다(10).

그라데이션 집계.을 위한 *케이*각 작업자는 작업자당 크기의 미니 배치를 가지고 있습니다. *N*, 다음(4), 경사 집계는 pe여야 합니다. Σ rfor전체 세트에 대해 약을 먹였습니다. *kn*예 ~에 따르면$\frac{1}{kn}$ *제이* $_{엑스\in B제이}$ *엘*$(x,w_{티})$. 손실 레이어는 일반적입니다. 평균 손실을 계산하기 위해 구현되었습니다. *봐라-웨*Σ 이는 작업자당 손실을 계산하는 것과 같습니다. *엘*$(x,w_{티})/N$. 이를 감안할 때 올바른 집계에는 다음이 필요합니다. *평균화*그만큼 *케이*누락된 부분을 복구하기 위한 그라데이션1*/케이* 요인. 그러나 allreduce와 같은 표준 통신 기본 요소는 [11] 평균이 아닌 합산을 수행합니다. 그러므로, 흡수하는 것이 더 효율적입니다.1*/케이*손실을 스케일링합니다. 이 경우 입력에 대한 손실의 그래디언트만 스케일링하면 되므로 전체 그래디언트 벡터를 스케일링할 필요가 없습니다. 우리는 이를 다음과 같이 요약합니다.

비고 3: 근로자당 손실을 다음과 같이 정규화합니다. 총 미니배치 크기크네, ~ 아니다 *작업자당 규모N.*

또한 '취소'가 올바르지 않을 수도 있다는 점에 유의하세요. *케이*' 설정으로 *eta*$\hat{}$ *= 에타*(~ 아니다 *km*) 다음과 같이 손실을 정규화합니다.1*/N*(~ 아니다1*/kn*), 이로 인해 잘못된 체중 감소가 발생할 수 있습니다(참조 *비고 1*).

데이터 셔플링.SGD는 일반적으로 데이터를 무작위로 샘플링하는 프로세스로 분석됩니다.*교체로*. 실제로는 일반적인 SGD 구현이 적용됩니다.*무작위 셔플링*더 나은 결과를 제공할 수 있는 각 SGD 에포크 동안 훈련 세트의삼,13]. 셔플링을 사용하는 기준선과 공정한 비교를 제공하려면(*예를 들어*., [16]), 우리는 한 시대의 샘플이 다음과 같이 수행되도록 보장합니다.*케이*작업자는 훈련 세트를 일관되게 무작위로 섞은 것에서 나옵니다. 이를 달성하기 위해 각 에포크에 대해 다음과 같이 분할된 무작위 셔플링을 사용합니다.*케이* 각 부품은 다음 중 하나에 의해 처리됩니다.*케이*노동자. 여러 워커에서 무작위 셔플링을 올바르게 구현하지 못하면 눈에 띄게 다른 동작이 발생하여 결과와 결론이 오염될 수 있습니다. 요약하자면:

참고 4: 모든 데이터에 나누어지는 학습 데이터(에포크당)를 무작위로 무작위로 섞는 단일 사용케이노동자.

4. 의사소통

단일 Big Basin 서버에서 GPU 8개 이상으로 확장하려면 [24], 그래디언트 집계는 네트워크의 서버 전체에 걸쳐 있어야 합니다. 거의 완벽한 선형 확장을 허용하려면 집계를 수행해야 합니다.*병행하여*백프롭으로. 이는 레이어 간 그라디언트 간에 데이터 종속성이 없기 때문에 가능합니다. 따라서 레이어에 대한 그래디언트가 계산되자마자 작업자 전체에 걸쳐 집계되고 다음 레이어에 대한 그래디언트 계산은 계속됩니다([5]). 자세한 내용은 다음에 설명하겠습니다.

4.1. 그라데이션 집계

모든 그래디언트에 대해 집계는 다음을 사용하여 수행됩니다.*모두 감소하다*작업(MPI 집단 작업과 유사) *MPI 올리듀스*[11])._allreduce가 시작되기 전에 모든 GPU에는 로컬로 계산된 기울기가 있고, allreduce가 완료된 후 모든 GPU에는 모든 기울기의 합계가 있습니다.*케이*그라디언트. 매개변수 수가 증가하고 GPU의 컴퓨팅 성능이 향상됨에 따라 역전파 단계에서 집계 비용을 숨기는 것이 더 어려워집니다. 이러한 효과를 극복하기 위한 훈련 기술은 이 작업의 범위를 벗어납니다(*예를 들어*., 양자화된 기울기 [18], 블록 모멘텀 SGD [6]). 그러나 이 작업 규모에서는 최적화된 올리듀스 구현을 사용하여 거의 선형에 가까운 SGD 확장을 달성할 수 있었기 때문에 집단 통신에 병목 현상이 발생하지 않았습니다.

우리의 allreduce 구현은 서버 내 및 서버 간 통신을 위한 세 단계로 구성됩니다. (1) 서버 내 8개 GPU의 버퍼는 각 서버에 대한 단일 버퍼로 합산됩니다. (2) 결과 버퍼는 모든 서버에서 공유되고 합산됩니다. 마지막으로 (3) 결과가 각 GPU에 방송됩니다. 단계 (1)과 (3)의 로컬 축소 및 방송을 위해 NCCL(NVIDIA Collective Communication Library)을 사용했습니다.삼256KB 이상의 버퍼 크기와 다음으로 구성된 간단한 구현의 경우 GPU-호스트 메모리 복사본 수 및 그렇지 않은 경우 CPU 감소. NCCL은 GPU 커널을 사용하여 서버 내 집합체를 가속화하므로 이 접근 방식은 처리량을 향상시키기 위해 유휴 상태였던 CPU 리소스를 사용하는 동시에 GPU에서 역전파에 더 많은 시간을 할애합니다.

서버 간 AllReduce의 경우 대역폭 제한 시나리오에 가장 적합한 두 가지 알고리즘을 구현했습니다.*재귀적 반감기 및 배가 알고리즘*[30,37] 그리고 *버킷 알고리즘*(링 알고리즘이라고도 함) [2].
두 경우 모두 각 서버가 보내고 받습니다.2*피*-1 ⁄*피*⁄*비*바이트 데이터, 어디에*비*버퍼 크기(바이트)이고*피*서버 수입니다. 반감기/배기 알고리즘은 다음과 같이 구성됩니다.2 로그2(*피*)통신 단계에서 링 알고리즘은 다음과 같이 구성됩니다.2(*피* -1) 단계. 이는 일반적으로 대기 시간이 제한된 시나리오에서 절반/2배 알고리즘을 더 빠르게 만듭니다(*즉*., 작은 버퍼 크기 및/또는 큰 서버 수의 경우). 실제로 우리는 절반/2배 알고리즘이 최대 백만 개의 요소에 대한 버퍼 크기(대규모 서버 수에서는 훨씬 더 높음)에 대해 링 알고리즘보다 훨씬 더 나은 성능을 발휘한다는 것을 발견했습니다. 32개 서버(256개 GPU)에서 절반/2배로 늘리면 속도가 3만큼 향상되었습니다. *x*링 알고리즘을 통해.

절반/2배 알고리즘은 감소 분산 집합과 전체 수집으로 구성됩니다. Reduce-Scatter의 첫 번째 단계에서 서버는 쌍으로 통신합니다(0순위는 1순위, 2순위는 3순위,*등*.), 입력 버퍼의 서로 다른 절반을 보내고 받습니다. 예를 들어, 랭크 0은 버퍼의 후반부를 1로 보내고, 버퍼의 전반부를 1에서 받습니다. 다음 단계로 진행하기 전에 수신된 데이터에 대한 감소가 수행됩니다. 보내고 받는 데이터가 절반으로 줄어듭니다. 감소-분산 단계가 완료된 후 각 서버는 최종 감소 벡터의 일부를 갖게 됩니다.

그 다음에는 감소-산란에서 통신 패턴을 역으로 되돌리는 allgather 단계가 이어지며, 이번에는 단순히 최종 감소 벡터의 일부를 연결합니다. 각 서버에서는 Reduce-Scatter에서 전송되었던 버퍼 부분이 allgather에서 수신되고, 이제 수신 중이던 부분이 전송됩니다.

2의 거듭제곱이 아닌 서버 수를 지원하기 위해 우리는 다음을 사용했습니다.*바이너리 블록 알고리즘*[30]. 이는 서버가 2의 거듭제곱 블록으로 분할되고 두 개의 추가 통신 단계가 사용되는 절반/2배 알고리즘의 일반화된 버전입니다. 하나는 블록 내 감소-분산 직후와 블록 내 모두 수집 전입니다. 2의 거듭제곱이 아닌 경우에는 2의 거듭제곱에 비해 어느 정도 로드 불균형이 있지만, 실행에서는 심각한 성능 저하가 나타나지 않았습니다.

4.2. 소프트웨어

설명된 allreduce 알고리즘은 다음에서 구현됩니다. *글루*4, 집단소통을 위한 도서관입니다. 지원합니다

삼https://developer.nvidia.com/nccl

4https://github.com/facebookincubator/gloo

다중 통신 컨텍스트. 이는 다중 allreduce 인스턴스를 병렬로 실행하는 데 추가 동기화가 필요하지 않음을 의미합니다. 로컬 축소 및 브로드캐스트(단계 (1) 및 (3)으로 설명됨)는 가능한 경우 서버 간 allreduce를 사용하여 파이프라인됩니다.

*카페2*훈련 반복을 나타내는 컴퓨팅 그래프의 다중 스레드 실행을 지원합니다. 하위 그래프 사이에 데이터 종속성이 없을 때마다 여러 스레드가 해당 하위 그래프를 병렬로 실행할 수 있습니다. 이것을 역전파에 적용하면 전체 감소 또는 가중치 업데이트를 처리하지 않고도 로컬 그래디언트를 순서대로 계산할 수 있습니다. 이는 역전파 중에 다음 세트가 수행됨을 의미합니다.*실행 가능한 하위 그래프* 우리가 실행하는 것보다 더 빠르게 성장할 수 있습니다. allreduce 실행이 포함된 하위 그래프의 경우 모든 서버는 실행 가능한 하위 그래프 세트에서 동일한 하위 그래프를 실행하도록 선택해야 합니다. 그렇지 않으면 서버가 교차하지 않는 하위 그래프 집합을 실행하려고 시도하는 분산 교착 상태가 발생할 위험이 있습니다. allreduce는*집단*작동하지 않으면 서버가 대기 시간을 초과하게 됩니다. 올바른 실행을 보장하기 위해 이러한 하위 그래프에 부분 순서를 적용합니다. 이는 주기적 제어 입력을 사용하여 구현됩니다.*N*-th allreduce는 (의 실행 차단을 해제합니다.*N*+*씨*)-번째 올리듀스, *씨*동시 allreduce 실행의 최대 수입니다. 이 숫자는 전체 컴퓨팅 그래프를 실행하는 데 사용되는 스레드 수보다 낮게 선택되어야 합니다.

4.3. 하드웨어

우리는 Facebook의 Big Basin을 사용했습니다.24] 실험을 위한 GPU 서버입니다. 각 서버에는 NVIDIA NVLink와 상호 연결된 8개의 NVIDIA Tesla P100 GPU가 포함되어 있습니다. 로컬 스토리지의 경우 각 서버에는 3.2TB의 NVMe SSD가 있습니다. 네트워크 연결을 위해 서버에는 Mellanox ConnectX-4 50Gbit 이더넷 네트워크 카드가 있으며 Wedge100에 연결됩니다.1] 이더넷 스위치.

다음 분석에 따르면 ResNet-50용 분산 동기 SGD에 50Gbit의 네트워크 대역폭이 충분한 것으로 나타났습니다. ResNet-50에는 약 2,500만 개의 매개변수가 있습니다. 즉, 매개변수의 전체 크기는 다음과 같습니다. 25 ·10$_6$ ·크기(부동 소수점) =100MB. 단일 NVIDIA Tesla P100 GPU에서 ResNet-50에 대한 역전파에는 120ms가 걸립니다. allreduce가 필요하다는 점을 고려하면 ~2 *X*네트워크의 바이트 수를 작동하는 값과 비교하면 최대 대역폭 요구 사항은 다음과 같습니다.200MB/0.125초 =1600MB/s 또는 12.8 Gbit/s, 통신 오버헤드를 고려하지 않음. 네트워크 오버헤드에 대한 스머지 요소를 추가하면 ResNet-50의 최대 대역폭 요구 사항에 도달합니다. ~15Gbit/s.

이 최대 대역폭 요구 사항은 역전파 중에만 유지되므로 네트워크는 집계보다 대기 시간에 덜 민감한 다양한 작업에 자유롭게 사용할 수 있습니다(*예를 들어*. 데이터 읽기 또는 네트워크 스냅샷 저장).

5. 주요 결과 및 분석

우리의 주요 결과는 ResNet-50을 훈련시킬 수 있다는 것입니다.16] ImageNet에서 [33] 한 시간에 256명의 작업자를 사용하면서 소규모 미니배치 훈련의 정확도와 일치합니다. 준비 전략과 함께 선형 확장 규칙을 적용하면 추가 하이퍼 매개변수를 조정하거나 정확도에 영향을 주지 않고 크고 작은 미니배치(최대 8k 이미지) 간에 원활하게 확장할 수 있습니다. 다음 하위 섹션에서는 (1) 실험 설정을 설명하고, (2) 대규모 미니배치 훈련의 효율성을 확립하고, (3) 더 심층적인 실험 분석을 수행하고, (4) 우리의 결과가 객체 감지/세분화에 일반화되었음을 보여주고, (5) 타이밍을 제공합니다.

5.1. 실험 설정

1000방향 ImageNet 분류 작업 [33]는 주요 실험 벤치마크 역할을 합니다. 모델은 다음에 대해 훈련됩니다.~128만 개의 훈련 이미지와 50,000개의 검증 이미지에 대한 상위 1 오류로 평가되었습니다.

우리는 ResNet-50 [16] 변형 [12], stride-2 컨볼루션이 3에 있다는 점에 유의하세요. *X*1개가 아닌 3개의 레이어 *X*[와 같이 1개 레이어16]. 우리는 Nesterov 추진력을 사용합니다 [29] 와 함께 *중*0.9 중 다음 [12] 그러나 [에서 사용된 표준 운동량에 유의하십시오.16]도 똑같이 효과적입니다. 우리는 체중 감소를 사용합니다*λ*0.0001 이하 [16] 학습 가능한 BN 계수에 가중치 감소를 적용하지 않습니다(즉, *γ*그리고*β*안에 [19]). BN 배치 크기에 따라 달라지는 훈련 목표를 고정된 상태로 유지하기 위해*N*§ 에 설명된 대로2.3, 우리는 사용*N*=32전체 미니배치 크기에 관계없이 전체적으로. 에서와 같이 [12], 실행 평균(모멘텀 0.9)을 사용하여 BN 통계를 계산합니다.

모든 모델은 미니배치 크기에 관계없이 90세대 동안 학습되었습니다. 우리는*선형 스케일링 규칙*에서 § 2.1그리고 학습률을 사용합니다.*에타*=0.1 · *kn*/256이는 미니에서 선형입니다. 배치 크기*kn*. 와 함께*케이*=8작업자(GPU) 및*N*=32 작업자 당 샘플,*에타*=0.1에서와 같이 [16]. 우리는 이것을 숫자라고 부릅니다. 베르 (0.1 · ~~*kn*/256~~)그만큼*참조 학습률*, 그리고 이를 다음과 같이 줄입니다. 1/1030번째, 60번째, 80번째 에포크에서 [16].

우리는 [15] 모든 컨벌루션 레이어에 대해. 1000방향 완전 연결 레이어는 표준 편차가 0.01인 평균 0 가우스에서 가중치를 가져와 초기화됩니다. 우리는 작은 미니배치가 있는 SGD가 BN으로 인한 초기화에 민감하지 않지만 상당히 큰 미니배치의 경우에는 그렇지 않다는 것을 발견했습니다. 또한 초기 교육에서 최적화 문제를 방지하려면 적절한 워밍업 전략이 필요합니다.

BN 레이어의 경우 학습 가능한 스케일링 계수*γ*1로 초기화되며,*γ가 0으로 초기화되는 각 잔차 블록의 마지막 BN을 제외하고.*환경*γ*=0각 잔차 블록의 마지막 BN에서 처음에 순방향/역방향 신호가 ResNets의 ID 지름길을 통해 전파되도록 하며, 이는 훈련 시작 시 최적화를 쉽게 하는 것으로 나타났습니다. 이 초기화는 모든 모델을 향상시키지만 앞으로 설명할 대규모 미니배치 훈련에 특히 유용합니다.

우리는 규모와 종횡비 데이터 확대를 사용합니다.36] 에서와 같이 [12]. 네트워크 입력 이미지는 224입니다. *X*증강 이미지 또는 수평 뒤집기에서 224픽셀 무작위 자르기. 입력 이미지는 다음과 같이 색상별 평균과 표준편차로 정규화됩니다.12].

무작위 변형 처리.모델은 훈련 시 무작위로 변하기 때문에 모델의 오류율을 다음과 같이 계산합니다.*중앙값*마지막 5개 시대의 오류. 또한, 우리는 오류의 평균과 표준편차(std)를 보고합니다.*5개의 독립적인 실행*. 이를 통해 결과에 대한 신뢰도가 높아지고 모델 안정성에 대한 척도도 제공됩니다.

ImageNet 모델의 무작위 변형은 일반적으로 이전 연구에서 보고되지 않았습니다(주로 리소스 제한으로 인해). 우리는 무작위 변화를 무시하면 신뢰할 수 없는 결론을 초래할 수 있다는 점을 강조합니다. 특히 결과가 단일 임상시험 또는 다수의 임상시험 중 최고인 경우 더욱 그렇습니다.

기준선.이러한 설정에서 우리는 다음을 사용하여 ResNet-50 기준을 설정합니다.*케이*=8 (서버 한 대에 GPU 8개) 및*N*=32 작업자당 이미지(미니배치 크기*kn*=256),에서와 같이 [16]. 우리의 기준선에는 23.60%의 상위 1개 검증 오류가 있습니다.*±*0.12. 참고로 ResNet-50은fb.resnet.torch [12]에는 24.01%의 오류가 있고 원본 ResNet 논문의 오류는 [16]는 약한 데이터 증대 하에서 24.7%를 나타냅니다.

5.2. 최적화 또는 일반화 문제?

최적화 및 일반화 동작을 탐색하여 대규모 미니배치 훈련에 대한 주요 결과를 설정합니다. 적절한 준비 전략을 사용하면 대규모 미니배치 SGD가 두 가지 모두 일치할 수 있음을 보여 드리겠습니다.*훈련 곡선*작은 미니배치 SGD로 구성되어 있으며*확인*오류. 즉, 우리의 실험에서는 두 가지 모두*최적화*그리고*일반화*대규모 미니배치 훈련은 소규모 미니배치 훈련과 일치합니다. 게다가 § 에서는5.4우리는 이러한 모델이 객체 감지/분할 전송 작업에 대해 좋은 일반화 동작을 보여 작은 미니배치 모델의 전송 품질과 일치한다는 것을 보여줄 것입니다.

다음 결과를 위해 우리는*케이*=256그리고*N*=32, 그 결과 미니배치 크기가 생성됩니다.*kn*=8k(1024를 나타내기 위해 '1k'를 사용함). 논의한 바와 같이, 우리의 기준선은 다음과 같은 미니배치 크기를 갖습니다.*kn*=256참조 학습률은 다음과 같습니다. *에타*=0.1.선형 스케일링 규칙을 적용하면*에타*=삼.2 대규모 미니배치 실행에 대한 참조 학습률로 사용됩니다. § 에서 설명한 세 가지 준비 전략을 테스트합니다.2.2:*워밍업 없음*, *지속적인 워밍업*~와 함께*에타*=0.15개 에포크 동안,*점진적인 워밍업*다음으로 시작하는*에타*=0.1그리고 선형적으로 증가하여*에타*=삼.25시대 이상. 모든 모델은 처음부터 학습되며 다른 모든 하이퍼 매개변수는 고정된 상태로 유지됩니다. 특정 미니배치 크기에 대해 더 나은 결과는 해당 경우에 대한 하이퍼 매개변수를 최적화하여 얻을 수 있음을 강조합니다.*우리의 목표는 각 미니배치 크기에 대한 하이퍼 매개변수 조정을 방지하는 일반적인 전략을 사용하여 미니배치 크기 전반에 걸쳐 오류를 일치시키는 것입니다.*.

	케이	*N*	*kn*	*에타* 상위 1개 오류(%)	
기준(단일 서버)	8	32	256	0.1	23.60 ±0.12
워밍업 없음, 그림2a 지속적	256	32	8천	3.2	24.84 ±0.37
인 워밍업, 그림2b 점진적인	256	32	8천	3.2	25.88 ±0.56
워밍업, 그림2c	256	32	8천	3.2	23.74 ±0.09

1 번 테이블.ResNet-50을 사용하는 ImageNet의 유효성 검사 오류(평균 및 표준은 5회 시행에 걸쳐 계산됨). 우리는 작은 미니배치 모델을 비교합니다(*kn*=256) 대규모 미니배치 모델 사용(*kn*=8k) 다양한 준비 전략을 사용합니다. 소규모 및 대규모 미니배치 훈련(점진적 준비 포함)에 대한 상위 1 검증 오류는 23.60%로 매우 가깝습니다.*±* 0.12*대*. 23.74% *±*각각 0.09.

훈련 오류.훈련 곡선은 그림에 나와 있습니다.2. 워밍업 없이(2a), 대규모 미니 배치에 대한 훈련 곡선*kn*=8k는 작은 미니 배치를 사용한 훈련보다 열등합니다. *kn*=256모든 시대에 걸쳐. 지속적인 준비 전략(2b) 실제로 결과가 저하됩니다. 작고 일정한 학습 속도가 준비 중 오류를 줄일 수 있지만 직후 오류가 급증하고 훈련이 완전히 복구되지 않습니다.

우리의 주요 결과는 점진적인 워밍업을 통해 큰 미니배치 훈련 오류가 작은 미니배치로 얻은 기본 훈련 곡선과 일치한다는 것입니다. 그림 참조2c. 큰 미니배치 곡선은 낮은 값으로 인해 더 높게 시작되지만*에타* 워밍업 단계에서는 곧 따라잡습니다. 약 20개의 Epoch 후에는 소규모 및 대규모 미니배치 훈련 곡선이 거의 일치합니다. 워밍업을 하지 않는 것과 점진적인 워밍업을 비교하면 다음과 같습니다.*대규모 미니 배치 크기는 초기 훈련의 최적화 어려움으로 인해 어려움을 겪습니다.*이러한 어려움이 해결되면 훈련 오류와 해당 곡선이 작은 미니배치 기준과 거의 일치할 수 있습니다.

유효성 검사 오류입니다.테이블1보여줍니다*유효성 검사 오류* 세 가지 준비 전략을 위해. 워밍업이 없는 변형에는 ~기준선보다 1.2% 더 높은 검증 오류로 인해 발생할 수 있습니다.~2.1%학습 오류 증가(그림2a), 과적합이나 기타 일반화 불량의 원인이 아닙니다. 이 주장은 점진적인 워밍업 실험을 통해 더욱 뒷받침됩니다. 점진적인 워밍업 변형에는*확인*기준선의 0.14% 이내의 오류(이러한 추정치의 std는~0.1%). 최종 훈련 오류를 고려하면(그림2c) 이 경우에는 잘 일치합니다. *최적화 문제가 해결되면 대규모 미니배치 훈련을 사용해도 명백한 일반화 저하가 관찰되지 않습니다.*, 미니배치 크기가 256에서 8k로 변경되더라도 마찬가지입니다.

마지막으로, 그림4점진적인 워밍업이 포함된 대규모 미니배치 훈련에 대한 훈련 및 검증 곡선을 모두 보여줍니다. 볼 수 있듯이 검증 오류는 두 번째 학습률 하락 이후 기준선과 거의 일치하기 시작합니다. 실제로 실행 평균을 사용하는 대신 오류를 평가하기 전에 BN 통계를 다시 계산하면 검증 곡선이 더 일찍 일치할 수 있습니다(그림의 캡션도 참조).4).

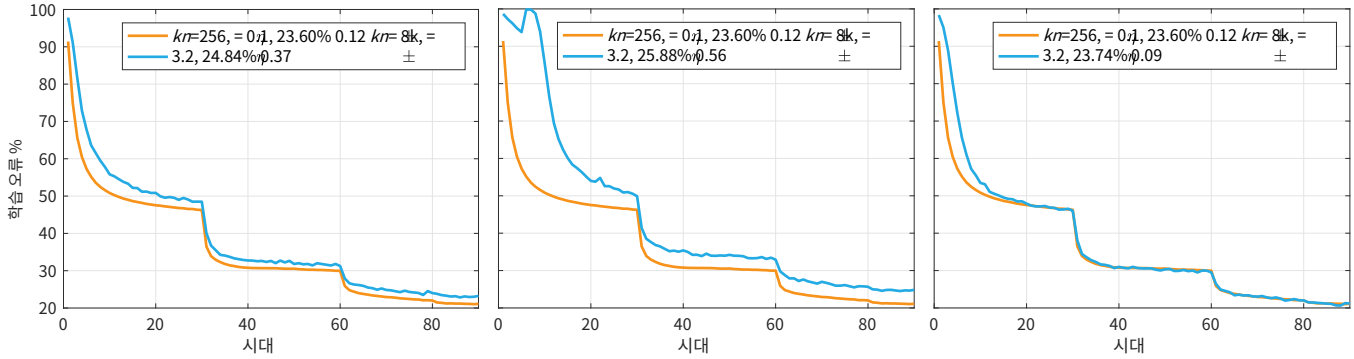
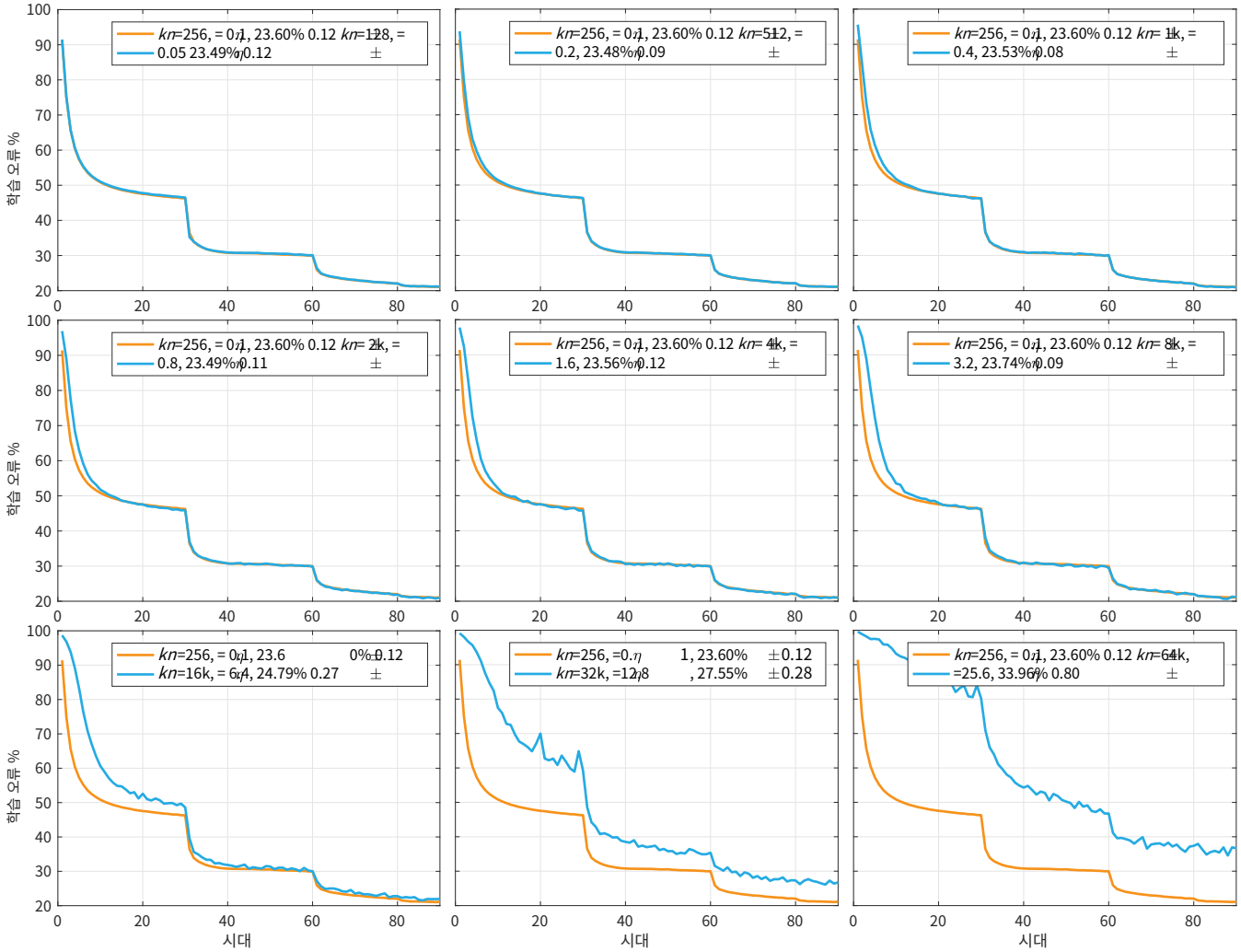


그림 2.워밍업.미니배치 크기 256과 비교하여 다양한 준비 전략을 사용하는 미니배치 크기 8192에 대한 훈련 오류 곡선. *확인*오류(평균 ±5회 실행의 표준)이 미니배치 크기와 함께 범례에 표시됩니다.*kn*및 참조 학습률*에타*.



에포크 3.훈련 오류*대*. 미니배트 h 사이즈.훈련 어 에 대한 ror 곡선 256 미니브 8k 미니 배치 크기까지 점진적인 eline(준비 기간 제
승armup과 린 이니 일자형 5회 실행의 훈련 td 곡선이 밀접하게 매트 새n 베이스 외)을 사용하여 기준선 및 더 큰 미니 배치를 배치합니
중배치.*발리다* *tion*오류(평균 ±에스 가 어떻게 되나요? 승n 전설에서 알 함께 다.*kn*및 참조 학습률*에타*.

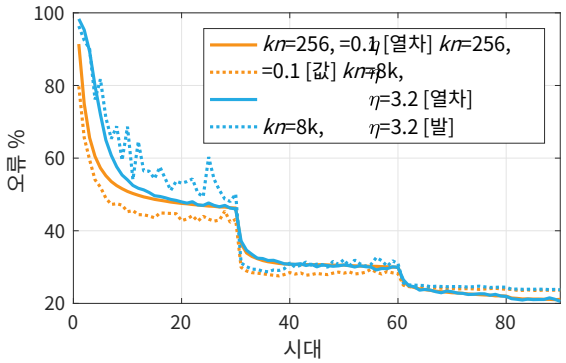


그림 4.훈련 및 검증 곡선점진적인 준비를 갖춘 대규모 미니배치 SGD 용*대*. 소형 미니배치 SGD. 충분한 에포크 동안 훈련한 후에 두 곡선 세트 모두 밀접하게 일치합니다. BN 통계(추론 전용)는 다음을 사용하여 계산됩니다.*달리기*평균은 대규모 미니배치에서 덜 빈번하게 업데이트되므로 초기 훈련에서 더 노이즈가 많습니다(이는 초기 에포크에서 검증 오류의 더 큰 변동을 설명합니다).

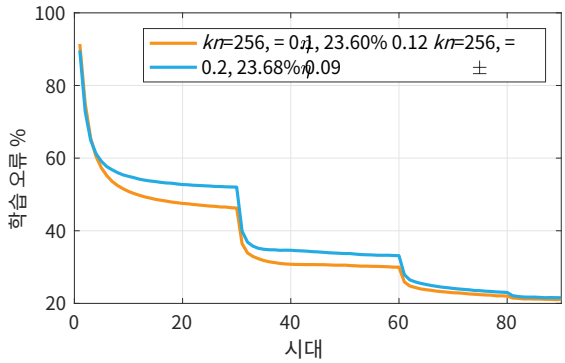


그림 5.학습률이 다른 작은 미니배치에 대한 훈련 곡선*에타*.예상대로 변화*에타*결과는 다음과 같습니다.*일치하지 않는*. 이는 배치 크기를 변경하고 선형적으로 확장하는 것과 대조됩니다.*에타*), 결과적으로 다음과 같은 곡선이 생성됩니다.*일치합니까*,*예를 들어*. 그림 참조삼.

5.3. 분석 실험

미니배치 크기*대*. 오류.수치1(1페이지)은 64~65536(64k) 범위의 미니 배치 크기로 학습된 모델에 대한 상위 1개 검증 오류를 보여줍니다. 모든 모델에 대해 선형 스케일링 규칙을 사용하고 참조 학습률을 설정했습니다.

~처럼*에타*=0.1 ·*kn*256.다음을 갖춘 모델의 경우*크네*>256,우리는 사용했었다 점진적인 워밍업 전략은 항상*에타*=0.1 5 epoch 후에 기준 학습률에 대해 선형적으로 증가합니다. 수치1은 검증 오류가 64에서 8k까지 광범위한 미니배치 크기에서 안정적으로 유지되다가 이후 증가하기 시작한다는 것을 보여줍니다. 64k를 초과하는 훈련은 선형 학습률 확장 규칙을 사용할 때 발산됩니다.[5]

다양한 미니배치 크기에 대한 훈련 곡선.그림의 9개 플롯 각각 삼256개 미니배치 기준선(주황색)에 대한 상위 1개 훈련 오류 곡선과 다양한 크기의 미니배치(파란색)에 해당하는 두 번째 곡선을 보여줍니다. 검증 오류는 플롯 범례에 표시됩니다. 미니배치 크기가 증가함에 따라 모든 훈련 곡선은 훈련 시작 시 기준선과 약간의 차이를 보입니다. 그러나 최종 검증 오류가 기준선과 거의 일치하는 경우(*kn*≤8k) 훈련 곡선은 이후에도 거의 일치합니다.

티 유효성 검사 시 오류는 발생하지 않습니다. 성냥
(*kn*≥16케이), 거기 아 에 띄는 g 비교할 때 훈련의 곡선
에프 면 모든 시대. 티 그의 다음과 같이 제안한다 AP 새로운
에서, 더 트레이닝 g 우베스는 될 수 있다 신뢰할 수 있는 도구로 사용됨 이 대행
성공을 위해 훨씬 전에 e 트레이닝 핀 녀.

대안 학습 요금 규칙. 티 할 수 있는2a해상도를 보여줍니다 궁극기
다중 학습률 에스. 소규모의 경우 중초기 배치(*kn* = 256),

5우리는 verng 다중 가용성 y 대규모 미니 하드웨어 분야에서 우리는*시뮬레이션* *먹은*디스
그래디언트의 리벳된 후 배치 사용(누적 단계는≥12k) 단일 서비스 트 어, 우리가-
녀 때문에 주목합니다. 그래디언트 적산에서 윈 SGD 업데이트. 우리는
분산 트레이에 ive 단일 서비스에 대한 어, 수확량
Nng.

에타=0.1최상의 오류를 제공하지만 약간 더 작거나 더 큼*에타* 또한 잘 작동합니다. 8k 이미지의 미니배치에 선형 스케일링 규칙을 적용하면 최적의 오류도 다음과 같이 달성됩니다.*에타* =0.1 ·32,선형 스케일링 규칙의 성공적인 적용을 보여줍니다. 그러나 이 경우 결과는 변화에 더 민감합니다.*에타*. 실제로는 한계점에 가깝지 않은 미니배치 크기를 사용하는 것이 좋습니다.

수치5다음을 사용하여 256 미니배치의 훈련 곡선을 보여줍니다.*에타*=0.1또는0.2.학습률이 바뀌는 것을 보여줍니다.*에타* 일반적으로 최종 오류가 유사하더라도 훈련 곡선의 전체 모양이 변경됩니다. 이 결과를 선형 스케일링 규칙(미니배치 크기가 변경될 때 최종 오류와 훈련 곡선을 모두 일치시킬 수 있음)의 성공과 대조하면 작은 미니배치와 큰 미니배치 간에 유지되는 기본 불변성이 드러날 수 있습니다.

우리는 또한 tw를 보여줍니다√ o 대안 전략: 유지*에타*0.1로 고정하거나0.1 ·32[에서 이론적으로 정당화된 제곱근 스케일링 규칙에 따라21] 규모가 커진다는 이유로*에타*기울기 추정기의 표준편차 감소량의 역수입니다.√

공정한 비교를 위해 우리는 또한 점진적인 워밍업을 사용하여0.1 ·32.결과에서 알 수 있듯이 두 정책 모두 실제로는 제대로 작동하지 않습니다.

배치 정규화γ초기화.테이블2b새로운 BN의 영향에 대한 통제 γ초기화가 도입되었습니다.

§ 5.1우리는 쇼 BN 초기화를 사용한 미니배치 크기 256 및 8k 티 그 앤드 rd 에 대한 결과(γ=1모든 BN 레이어에 대해) 그리고 ~와 함께 우리 최기화(γ=0최종 BN 레이어 ual 블록의 경우). 전자의 h re 시 결과는 향상된 퍼스를 보여줍니다γ=0두 미니배 형태 앤스 위 치 크기 모두에 대해 노출 에도 8k 미니배치 크기의 경우 약간 더 큽니다. r은 이것 행동 또한 최적화 문제로 인해 대규모 미니배치가 더 쉽게 당신은 영향을 미칩니까 어렵다고 제안합니다. 우리는 imization과 초기 노출 택했어요 화 방법이 대규모 미니배치 훈련에 도움이 될 것 푸시 보우 으로 기대합니다.

해상도 et-101. ResNet-101에 대한 결과 [16]는 Taing 블라 2씨. 기차 ResNet-101에 배치 크기로 표시됩니다.*kn*=8케이

kn	*에타*	상위 1개 오류(%)
256	0.05	23.92 ±0.10
256	0.10	23.60 ±0.12
256	0.20	23.68 ±0.09
8천	0.05 ·32	24.27 ±0.08
8천	0.10 ·32	23.74 ±0.09
8천	0.20 ·32	24.05 ±0.18
8천	0.10√	41.67 ±0.10
8천	0.10 ·32 —	26.22 ±0.03

(ㅏ)학습률 확장 규칙 비교.참조 학습률은 다음과 같습니다. *에타*=0.1가장 잘 작동합니다 *kn*=256 (23.68% 오류). 선형 스케일링 규칙은 다음을 제안합니다. *에타*=0.1 ·32언제 *kn*=8k는 다시 최고의 성능을 제공합니다(23.74% 오류). 다른 확장 방법 *에타*더 나쁜 결과를 낳습니다.

kn	*에타*	γ-초기화	상위 1개 오류(%)
256	0.1	1.0	23.84 ±0.18
256	0.1	0.0	23.60 ±0.12
8천	3.2	1.0	11.24 ±0.07
8천	3.2	0.0	23.74 ±0.09

(비)일괄 정규화 γ초기화.초기화 중 γ=0에서 *마지막* 각 잔여 블록의 BN 레이어는 작은 미니배치와 큰 미니배치 모두에 대한 결과를 향상시킵니다. 이러한 초기화는 더 나은 최적화 동작으로 이어지며, 이는 대규모 미니배치로 훈련할 때 더 큰 긍정적인 영향을 미칩니다.

모델 유형	*kn*	*에타*	상위 1개 오류(%)
ResNet-101	256	0.1	8.22 ±0.06
ResNet-101	8천	3.2	22.36 ±0.09

(씨)ResNet-101에 적용된 선형 스케일링 규칙.소규모 미니배치 훈련과 대규모 미니배치 훈련 간의 오류 차이는 약 0.3%입니다.

표 2.ImageNet 분류 실험.별도의 언급이 없는 한 모든 실험은 ResNet-50을 사용하며 평균 5회 이상 시도되었습니다.

그리고 선형적으로 스케일링된 *에타*=삼.222.36%의 오류가 발생합니다. *대*. 그만큼 *kn*=25622.08%를 달성하는 기준선 *에타* =0.1.즉, 미니배치 8k로 훈련된 ResNet-101은 오류가 0.28% 약간 증가했습니다. *대*. 기준선. 8k의 미니배치 크기는 ResNet-50과 유사하게 ResNet-101에 대한 유용한 미니배치 훈련 방식의 가장자리에 있을 가능성이 높습니다(그림 참조).1).

그만큼련 시간 ResNet-101의 92.5분입니다 에서현의 구현하대용하여 설명 g 256 테슬라 P 100 GPU 및 미니- 배치 s 8k의 크기. 우리 난 이걸 믿어 에설득력 있는 결과 특검 ed-정확도 tr ResN의 해제 et-101이 더 좋습니다 빨간색.

이미지Net-5k.관찰자 샤를 운전하다 피v의 증가 동맹 오류 b에트윈 미니바 tch 크기 8k 그리고 난 16,000원 중ageNet- 1k(그림 응?1), 자연 랄 질문은 만약 위치가 N 이의 '팔꿈치 오류에 곡선은 푸 N데이터의 액션 et 정보- 친구 콘텐츠. 에게 조사하다 질문입니다, 승 이마 geNet-5k 데이터 제안된 세트 by 시에 *외*. [39] 저것 연장하다ImageNet-1 케이 080만명까지 n개 이미지(루 글리 5 ×더 큰) 4k를 추가하여 추가 케이트 나로부터의 괴로움 중ageNet- 22k [33]. 우리는 평가한다 이때 way cl 인증 오류 아니면 원산지ㅣ ImageNet-1k 검증 세트 에서와 같이 [39].

그만큼미니배치 화하다 *대*. 유효한 오류 c 유열망하다 이미지Net-5k가 표시됩니다. 그림의 n6. 질적으로, t 그는 곡선

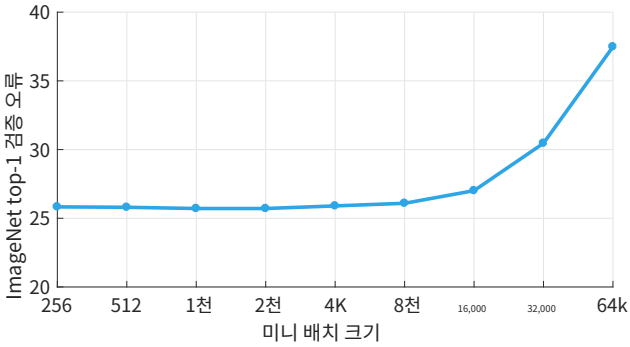


그림 6.ImageNet-5ktop-1 검증 오류 *대*. 고정된 90세대 훈련 일정을 갖춘 미니배치 크기입니다. 곡선은 ImageNet-1k의 결과와 질적으로 유사합니다(그림1) 5를 보여줌 ×훈련 데이터가 증가해도 최대 유효 미니배치 크기가 크게 변경되지는 않습니다.

ImageNet 사전 훈련			머리	
kn	*에타*	상위 1개 오류(%)	박스AP(%)	마스크 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
1천	0.4	23.53 ±0.08	35.9 ±0.2	33.9 ±0.2
2천	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
8천	3.2	23.74 ±0.09	35.8 ±0.1	33.9 ±0.2
16,000	6.4	24.79 ±0.27	35.1 ±0.3	33.2 ±0.3

(ㅏ)대규모 미니배치 사전 훈련 학습을 Mask R-CNN으로 전환합니다. 박스 및 마스크 AP(COCO에서최소)256개에서 8,000개 예제의 미니배치로 사전 훈련된 ResNet-50 모델과 거의 동일합니다. 16k의 미니배치 사전 훈련 크기를 사용하면 ImageNet 검증 오류가 모두 발생합니다. *그리고* COCO AP가 저하됩니다. 이는 ImageNet 오류가 일치하는 한 대규모 미니 배치가 전이 학습 성능을 저하시키지 않음을 나타냅니다.

# GPU	*kn θ* ·1000	반복	박스AP(%)	마스크 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

(비)Mask R-CNN에 선형 학습률 스케일링을 적용했습니다.[의 단일 ResNet-50 모델 사용16] (따라서 성병이 보고되지 않음) 마스크를 훈련시킵니다.
선형 학습률 s에 따라 1~8개의 GPU를 사용하는 R-CNN
칼링 규칙. 성공을 두 번째 마스크 AP는 일반 완전히 동일한 아크로 ss 모든 구성 ns
에서하는 상자 에서필요가 없습니다. N규칙 너머 라 분류.
타가능하다 3.물체 C에서의 탐지 영역와 Mas 케R-CNN [14].

내우 비슷해요 이미지로 Net-1k 곡선, s 시간적으로 분에 아르 자형
피실무자 그것 그럴 것 같지 않다 심지어 5 ×Inc 데이터세트에 넣다 티
에스 적으로 이어지다 의미있는 내사용 증가 -
ㅏ미니배치 크기. 수량 ㅏ적극적으로 n 8k 미니배트 시간
내를 증가시킨다 앨리데이션 오류 이보다 0.26% 중 25.83% ㅏ
256개의 미니배치 26.09%로. ㅏ N이해하다 g 정밀한 이자형
아≠재료이 좋다 트윈 일반 화 오류, 중이니배치 크기 ,
ㅏnd 데이터세트 inf 영향을 콘테 nt는 f에 대해 열려 있습니다. 미래의 일.

5. 4. 일반 De로의 전환 테션과 S 분할

낮은 오류 아르ma의 평가 geNet은 그렇지 않습니다 와 으로는 엔 디
g올. 대신에, t 그는 나에게 유용하다 중ageNet 훈련 N 는 학습에 거짓말을 -

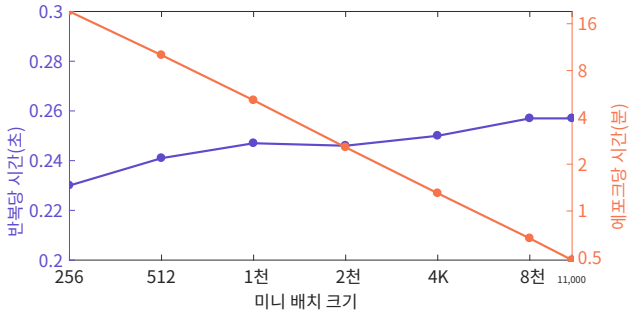


그림 7.분산 동기 SGD 타이밍.다양한 미니배치 크기로 훈련하는 경우 반복당 시간(초) 및 ImageNet epoch당 시간(분)입니다. 기준선(*kn* =256)단일 서버에서 8개의 GPU를 사용하는 반면, 다른 모든 훈련 실행은 훈련을 다음에 분산합니다(*kn*/256)섬기는 사람. 352개의 GPU(44개 서버)를 사용하여 우리의 구현은 전체에 대해 한 번의 패스를 완료합니다.~약 30초 안에 128만 개의 ImageNet 훈련 이미지.

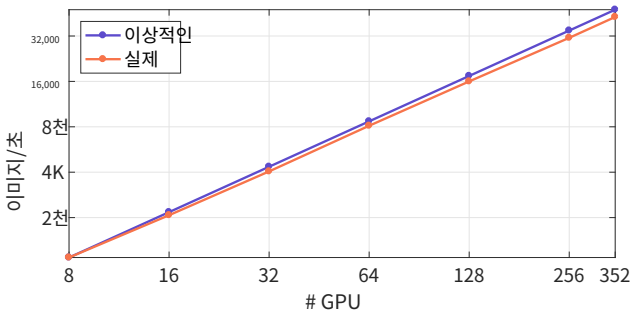


그림 8.분산 동기 SGD 처리량.8개의 GPU가 있는 단일 서버에서 다중 서버 분산 교육으로 이동할 때 발생하는 작은 오버헤드(그림7, 파란색 곡선)은 이상적인 확장보다 약간 낮은 선형 처리량 확장을 초래합니다 (~효율성 90%). AllReduce 통신 시간의 대부분은 기울기 계산을 통해 AllReduce 작업을 파이프라인함으로써 숨겨집니다. 또한 이는 상용 이더넷 하드웨어를 사용하여 달성됩니다.

관련 작업으로 전환하거나 잘 일반화하는 좋은 기능을 제공합니다. 가장 중요한 질문은 큰 미니배치로 학습한 기능이 작은 미니배치로 학습한 기능과 마찬가지로 일반화되는지 여부입니다.

이를 테스트하기 위해 COCO에서 객체 감지 및 인스턴스 분할 작업을 채택합니다.27] 이러한 고급 인식 작업은 ImageNet 사전 훈련으로부터 상당한 이점을 얻습니다.10]. 최근 개발된 Mask R-CNN을 사용합니다.14] 객체 인스턴스를 감지하고 분할하는 방법을 학습할 수 있는 시스템입니다. 우리는 [에서 사용된 모든 하이퍼 매개변수 설정을 따릅니다.14] Mask R-CNN 훈련을 초기화하는 데 사용되는 ResNet-50 모델만 변경합니다. COCO에서 Mask R-CNN을 훈련시킵니다. 기차발35k5K 이미지 분할 및 결과 보고최소의[에 사용된 분할 14].

Mask R-CNN의 미니배치 크기 개념이 분류 설정과 다르다는 점이 흥미롭습니다. 의 확장으로서 *이미지 중심*빠르고/빠른 R-CNN [9,31], Mask R-CNN 전시 *다양한 레이어에 대한 다양한 미니배치 크기*: 네트워크 백본은 두 가지를 사용합니다.

이미지 (GPU당), 그러나 각 이미지는 512개 지역에 기여합니다. of-Inte분류 계산을 위한 나머지(다항 교차- 엔트로피, 경계 상자 등록 세션(부드러운-L1/H 유 베르) 그리고 픽셀-w이세 마스크(28 *×*28 바이항 교차 엔트로프 y) 손실. 이번 다이빙 미니배치 od 테스트 시간 리 앱의 크기와 손 데온 프로 한번 가보세요 케이스의 rse 세트 로실 기능의 흉상 바퀴벌레.

전송 어 나한테 배우고 있어 lrge 미니배치 프리 -훈련. 테스트하려면 미니백이 얼마나 큰지 시간*전 훈련*효과 마스크R-CNN, 승 나는 ResNet-50을 사용한다 중ma에 대한 훈련을 받은 모델 geNet-1k 25로 6~16,000개의 미니배치 그리고 이를 미니배치 크기를 초기화 리즈 마스크 R-CNN 훈련. 각각 하는 데 사용합니다. 우리는 5개 모두 재훈련 5 모델 그런 다음 마스크를 훈련시킵니다. 를 사용하여 R-CNN을 사용합니다. 중에 델 머리 (총 35개 모델) 여 이평균박스 신고 그리고 마스크 AP, 5t 이상 평균 아ials, 표에 있음3a. T et 유효성 시간 결과 보여줘 이미지만큼은 N검사 오류는 다음과 같습니다. 낮게 유지하고, 나는 어느 최대 8,000개 배치까지 true입니다. 일반화하다, 일반화하다 객체 드

섹션은 작은 미니배치 기준선의 AP와 일치합니다. 관찰했다는 점을 강조합니다. *일반화 문제 없음*대규모 미니배치로 훈련된 모델을 사용하여 데이터 세트(ImageNet에서 COCO로) 및 작업(분류에서 탐지/세분화까지) 간에 전송할 때.

Mask R-CNN에 적용된 선형 스케일링 규칙.또한 Mask R-CNN을 사용하여 선형 스케일링 규칙의 일반성에 대한 증거를 보여줍니다. 실제로 이 규칙은 이미 [[]에서 명시적인 논의 없이 사용되었습니다.16] 8개의 GPU를 사용할 때 기본 Mask R-CNN 훈련 방식으로 효과적으로 적용되었습니다. 테이블3b은 1, 2, 4 또는 8개의 GPU로 훈련할 때 선형 학습률 규칙이 상수 상자 및 마스크 AP를 생성한다는 것을 보여주는 실험 결과입니다. 이러한 실험을 위해 [에서 수행된 것처럼 출시된 MSRA ResNet-50 모델에서 Mask R-CNN을 초기화합니다. 14].

5.5. 런타임

수치7런타임 특성의 두 가지 시각화를 보여줍니다. 우리 시스템의 행위. 파란색 곡선은 시간당 미니 배치 크기가 256에서 11264(11k)까지 다양하므로 반복 . 특히 이 곡선은 상대적으로 평평하며 반복당 시간은 안으로12%만 주름이 집니다. 미니바 스케일링 tch 사이즈 b 와이 44 *×*. 다른 방식으로 시각화하면 주황색 곡선은 대략 엑스실상 선형 감소 l에포크 ds당 시간에 오버에서 아르 자형 16마일30초만 있으면 돼 N따른다. 런타임 수행 중캘리포니아 N 또한 b이것은에서 t의 관점에 처리량(이미지/초 두 번째), 에스 보여주다 본8. 8 GPU 기준 완벽하게 효율적으로N 넥스트라포 - 관계 에 상대적 구현하려면 n 달성하다 에스 ~90% 확장 효율성.

승인 영향을 .우리는 w Leon tical 배경, Jerr에게 보투포 아르 자형 도움이 되는 이론에 대한 토론 이재형 표하고 싶습니다. y 팬 디 크리스토론을 위한 Puhrsch 효율적인 데이터에 관한 사항 l 영향 이딩, 안- 그렸다나 디버그에 대한 도움이 필요하신가요 g분산 훈련을 하고 있어요, 그리고 케비 N 이B 요? rian Dodds, Jia Ning, 케O유튼, 미가 해리스, 디 존 5세 Big Basin과 h는 olk입니다. l하드웨어 지원.

참고자료

[1] J. Bagga, H. Morsy, 그리고 Z. 야오. 열리는 6팩 및 웨지 100용 디자인. https://code.facebook.com/posts/203733993317833/ opening-designs-for-6-pack-and-wedge-100, 2016.

[2] M. Barnett, L. Shuler, R. van De Geijn, S. Gupta, DG Payne 및 J. Watts. 인터프로세서 집단 통신 라이브러리(인터콤). ~ 안에 *확장 가능한 고성능 컴퓨팅 컨퍼런스*, 1994.

[3] L. 보투. 일부 확률적 경사하강법 알고리즘의 신기할 정도로 빠른 수렴. 2009년 SLDS 2009 컨퍼런스 참석 시 제공되는 미공개 공개 문제입니다.

[4] L. Bottou, FE Curtis 및 J. Nocedal. 고르다. 대규모 기계 학습 방법. *arXiv:1606.04838*, 2016.

[5] J. Chen, X. Pan, R. Monga, S. Bengio 및 R. Jozefowicz. 분산 동기 SGD를 다시 살펴봅니다. *arXiv:1604.00981*, 2016.

[6] K. Chen 및 Q. Huo. 블록 내 병렬 최적화 및 블록별 모델 업데이트 필터링을 사용한 증분 블록 교육을 통해 딥 러닝 머신의 확장 가능한 교육을 제공합니다. ~ 안에 *ICASSP*, 2016.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa. 처음부터 (거의) 자연어 처리. *JMLR*, 2011.

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell. Decaf: 일반적인 시각적 인식을 위한 심층 컨벌루션 활성화 기능입니다. ~ 안에 *ICML*, 2014.

[9] R. 거식. 빠른 R-CNN. ~ 안에 *ICCV*, 2015.

[10] R. Girshick, J. Donahue, T. Darrell 및 J. Malik. 정확한 객체 감지 및 의미론적 분할을 위한 풍부한 기능 계층. ~ 안에 *CVPR*, 2014.

[11] W. Gropp, E. Lusk 및 A. Skjellum. *MPI 사용: 메시지 전달 인터페이스를 사용한 휴대용 병렬 프로그래밍*. MIT 출판부, 매사추세츠주 케임브리지, 1999.

[12] S. 그로스(Gross)와 M. 윌버(M. Wilber). 잔여 그물을 훈련하고 조사합니다. https://github.com/facebook/fb.resnet.torch, 2016.

[13] M. Gürbüzbalaban, A. Ozdaglar 및 P. Parrilo. 무작위 재편성이 확률적 경사하강법을 능가하는 이유 *arXiv:1510.08560*, 2015.

[14] K. He, G. Gkioxari, P. Dollár 및 R. Girshick. 마스크 R-CNN. *arXiv:1703.06870*, 2017.

[15] K. He, X. Zhang, S. Ren 및 J. Sun. 정류기에 대한 심층 탐구: imagenet 분류에서 인간 수준의 성능을 능가합니다. ~ 안에 *ICCV*, 2015.

[16] K. He, X. Zhang, S. Ren 및 J. Sun. 이미지 인식을 위한 심층 잔여 학습. ~ 안에 *CVPR*, 2016.

[17] G. Hinton, L. Deng, D. Yu, GE Dahl, A.-r. 모하메드, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, TN Sainath 등 음성 인식의 음향 모델링을 위한 심층 신경망: 4개 연구 그룹의 공유된 견해. *IEEE 신호 처리 매거진*, 2012.

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv 및 Y. 벤지오. 양자화된 신경망: 낮은 정밀도의 가중치와 활성화를 사용하여 신경망을 훈련합니다. *arXiv:1510.08560*, 2016.

[19] S. Ioffe 및 C. Szegedy. 배치 정규화: 내부 공변량 이동을 줄여 심층 네트워크 훈련을 가속화합니다. ~ 안에 *ICML*, 2015.

[20] NS Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy 및 PTP 탱. 딥 러닝을 위한 대규모 배치 훈련: 일반화 격차와 날카로운 최소값. *ICLR*, 2017.

[21] A. Krizhevsky. 컨벌루션 신경망을 병렬화하는 이상한 방법 중 하나입니다. *arXiv:1404.5997*, 2014.

[22] A. Krizhevsky, I. Sutskever 및 G. Hinton. 심층 컨벌루션 신경망을 사용한 ImageNet 분류. ~ 안에 *닙스*, 2012.

[23] Y. LeCun, B. Boser, JS Denker, D. Henderson, RE Howard, W. Hubbard 및 LD Jackel. 손으로 쓴 우편번호 인식에 역전파를 적용했습니다. *신경 계산*, 1989.

[24] 이경. Big Basin을 소개합니다: 차세대 AI 하드웨어. https://code.facebook.com/posts/1835166200089399/소개-대분지, 2017.

[25] M. 리. *시스템 및 알고리즘 공동 설계를 통한 분산 기계 학습 확장*. 카네기멜론대학교 박사학위 논문, 2017.

[26] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan 및 S. 벨롱지. 객체 감지를 위한 피라미드 네트워크 기능을 제공합니다. ~ 안에 *CVPR*, 2017.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár 및 CL Zitnick. Microsoft COCO: 컨텍스트 내 공통 개체. ~ 안에 *ECCV*. 2014.

[28] J. Long, E. Shelhamer 및 T. Darrell. 의미론적 분할을 위한 완전 컨벌루션 네트워크. ~ 안에 *CVPR*, 2015.

[29] Y. 네스테로프. *볼록 최적화 입문 강의: 기본 과정*. 스프링거, 2004.

[30] R. Rabenseifner. 집단적 환원 작업의 최적화. ~ 안에 *ICCS*. 스프링거, 2004.

[31] S. Ren, K. He, R. Girshick 및 J. Sun. 더 빠른 R-CNN: 지역 제안 네트워크를 통한 실시간 객체 감지를 지향합니다. ~ 안에 *닙스*, 2015.

[32] H. 로빈스 및 S. 먼로. 확률론적 근사 방법. *수학적 통계의 연대기*, 1951.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, AC Berg, L. Fei-Fei. ImageNet 대규모 시각적 인식 챌린지. *IJCV*, 2015.

[34] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus 및 Y. LeCun. Overfeat: 컨볼루셔널 네트워크를 사용한 통합 인식, 위치 파악 및 감지. ~ 안에 *ICLR*, 2014.

[35] K. Simonyan 및 A. Zisserman. 대규모 이미지 인식을 위한 매우 깊은 컨벌루션 네트워크. ~ 안에 *ICLR*, 2015.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke 및 A. Rabinovich. 컨볼루션을 통해 더 깊이 들어가 보세요. ~ 안에 *CVPR*, 2015.

[37] R. Thakur, R. Rabenseifner 및 W. Gropp. 집단통신 최적화 MPICH에서의 작업. *IJHPCA*, 2005.

[38] Y. Wu, M. Schuster, Z. Chen, QV Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey 등 Google의 신경 기계 번역 시스템: 인간 번역과 기계 번역 간의 격차를 해소합니다. *arXiv:1609.08144*, 2016.

[39] S. Xie, R. Girshick, P. Dollár, Z. Tu 및 K. He. 심층 신경망에 대한 집계된 잔차 변환입니다. ~ 안에 *CVPR*, 2017.

[40] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu 및 G. Zweig. Microsoft 2016 대화식 음성 인식 시스템. *arXiv:1609.03528*, 2016.

[41] MD Zeiler 및 R. Fergus. 컨벌루션 신경망을 시각화하고 이해합니다. ~ 안에 *ECCV*, 2014.