creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사학위논문

8-bit 부동소수점 혼합 정밀도 훈련

Mixed Precision Training with an 8-bit Floating Point Format

2019년 2월

서울대학교 대학원

전기 정보 공학부

조 영 민

공학석사학위논문

8-bit 부동소수점 혼합 정밀도 훈련

Mixed Precision Training with an 8-bit Floating Point Format

2019년 2월

서울대학교 대학원

전기 정보 공학부

조 영 민

8-bit 부동소수점 혼합 정밀도 훈련

Mixed Precision Training with an 8-bit Floating Point Format

지도교수 성 원 용

이 논문을 공학석사 학위논문으로 제출함

2019년 2월

서울대학교 대학원

전기 정보 공학부

조 영 민

조영민의 공학석사 학위 논문을 인준함

2019년 2월

위 원 장: 심병효

부위원장: 성원용

위 원: 유승주

요 약

오늘날, 인공신경망은 다양한 응용 분야에서 우수한 성능을 보인다. 좋은 성능을 얻기 위해서는 복잡한 모델과 큰 사이즈의 데이터가 필요하며, 훈련 과정에 상당히 많은 연산량과 메모리 자원을 요구한다. 본 논문에서는 필요한 연산량과 메모리 자원을 줄이기 위해, 낮은 정밀도 데이터형을 활용한 8-bit 부동소수점(FP8) 혼합 정밀도 훈련을 진행한다. FP8의 지수부, 가수부 bit를 어떻게 구성하는 것이 가장 좋은지 살펴본다. 또한, FP8 혼합 정밀도 훈련에 스토케스틱 가중치 평균(SWA)을 다양하게 적용해보면서 가장 좋은 SWA 적용 방식을 찾는다.

다양한 실험 결과에 의하면, FP8을 부호 1-bit, 지수부 5-bit, 가수부 2-bit(FP8-s1e5m2)으로 구성해서 FP8 혼합 정밀도 훈련을 한 뒤, SWA를 적용하는 것이 가장 적합해 보인다. 본 논문에서는 FP8의 지수부, 가수부 bit 개수를 달리하며 bit 조합에 따른 모델 성능을 검증했다. FP8-s1e4m3은 성능은 우수하지만, 값 표현 범위가 좁아서 복잡한 작업에서 훈련이 되지 않았다. FP8-s1e6m1은 훈련은 되지만 모델이 깊어질수록 성능 저하가 상당히 심했다. FP8-s1e5m2은 약간의 성능 저하를 보였는데, SWA를 적용할 경우 약간의 성능 저하를 보이면서 FP32-Baseline의 성능을 거의 따라갔다. 추가로 본 논문에서는 SWA를 FP8 혼합 정밀도 훈련에 다양하게 적용해보았다. 훈련이 끝난 뒤 SWA를 적용하는 것은 대다수 실험에서 성능 향상을 보였다. 그러나, 훈련 중 SWA를 1회 적용하는 방식과 훈련 중 SWA를 매 에포크마다 적용하는 방식은 성능 향상을 보이지 않았다.

주요어: 혼합 정밀도 훈련(MPT), 8-bit 부동소수점(FP8), 스토케스틱 가중치 평균(SWA)

학 번: 2018-27706

차 례

요 약 **i**

제 1 장 서론 **5**

1.1 낮은 정밀도 데이터형을 활용한 인공신경망 훈련 5

1.2 관련 연구 6

1.2.1 낮은 정밀도 인공신경망 훈련 6

1.2.2 혼합 정밀도 훈련 7

1.3 SWA를 활용한 FP8 혼합 정밀도 훈련 8

1.4 논문의 개요 9

제 2 장 Bit 조합에 따른 FP8 혼합 정밀도 훈련 **10**

2.1 Bit 조합에 따른 값 표현 범위 10

2.2 스토캐스틱 라운딩 12

2.3 실험 결과 13

2.3.1 이미지 분류 작업 14

2.3.2 언어 모델링 작업 16

2.3.3 이미지 생성 작업 18

2.4 결론 21

제 3 장 SWA를 활용한 FP8 혼합 정밀도 훈련 **22**

3.1 스토캐스틱 가중치 평균 22

3.2 SWA 적용 방식 23

3.3 실험 결과 23

3.4 결론 30

제 4 장 결론 **31**

ABSTRACT **35**

표 차례

표 2.1 FP32, FP16, FP8의 값 표현 범위 비교 11
표 2.2 FP8 혼합 정밀도 훈련의 ResNet-CIFAR10 정확도 14
표 2.3 FP8 혼합 정밀도 훈련의 ResNet-ImageNet 정확도 15
표 2.4 FP8 혼합 정밀도 훈련의 WLM-PTB 복잡도 17
표 2.5 FP8 혼합 정밀도 훈련의 DCGAN-CIFAR10 성능 18
표 2.6 FP8 혼합 정밀도 훈련의 DCGAN-CELEBA MS-SSIM 19

표 3.1 다양한 모델 포인트를 활용한 SWA의 ResNet18-ImageNet 정확도 비교 24
표 3.2 훈련 중 SWA를 1회 적용할 때 ResNet20-CIFAR10 정확도 . . 25
표 3.3 훈련 중 SWA를 매 에포크 적용할 때 ResNet20-CIFAR10 정확도 26
표 3.4 가중치 교체 방식별 상수형 러닝 레이트 WLM-PTB 훈련 결과 28
표 3.5 가중치 교체 방식별 주기형 러닝 레이트 WLM-PTB 훈련 결과 30

그림 차례

그림 1.1 FP8 혼합 정밀도 훈련 과정 9

그림 2.1 ResNet20-CIFAR10 그레디언트 히스토그램 11

그림 2.2 FP8 컨버터가 적용된 방식 13

그림 2.3 FP8 혼합 정밀도 훈련의 DCGAN-CELEBA 이미지 생성 . . . 20

그림 3.1 ResNet18-ImageNet 훈련의 러닝 레이트 스케줄 및 다양한 모델 포인트 24

그림 3.2 WLM-PTB의 러닝 레이트 스케줄 27

그림 3.3 가중치 교체 방식별 상수형 러닝 레이트 WLM-PTB 훈련 양상 28

그림 3.4 가중치 교체 방식별 주기형 러닝 레이트 WLM-PTB 훈련 양상 29

제 1 장 서론

1.1 낮은 정밀도 데이터형을 활용한 인공신경망 훈련

오늘날, 인공신경망(Artificial Neural Network)은 이미지 인식, 음성 인식, 기계 번역, 이미지 생성과 같은 다양한 응용 분야에서 우수한 성능을 보인다 [5, 3, 1, 18]. 좋은 성능을 얻기 위해서는 사이즈가 큰 훈련 데이터와 복잡한 모델이 필요하다. 복잡한 모델은 훈련에 상당히 많은 연산량과 메모리 자원을 필요로 한다. 일반적으로 단일-정밀도 부동소수점(Single-precision Floating Point, FP32)을 활용하여 인공신경망을 훈련하는데, 정밀도가 낮은 데이터형을 사용하면 요구되는 연산량과 메모리 자원을 줄여 훈련 시간을 줄일 수 있다. 프로그램의 속도를 결정짓는 요소로 산수 대역폭(Arithmetic Bandwidth), 메모리 대역폭(Memory Bandwidth), 레이턴시(Latency)가 있다. 낮은 정밀도를 사용하면 산수 대역폭과 메모리 대역폭을 줄임으로써 속도를 높일 수 있다. 인공신경망의 훈련을 가속하고, 성능 저하를 최소화하기 위한 낮은 정밀도 데이터형(Reduced Precision Datatype)으로 적어도 16-bit을 활용하는 연구가 진행되어왔다. 널리 사용되는 16-bit 데이터형의 예로 고정소수점 16-bit, IEEE 754 반-정밀도 부동소수점(Half-precision Floating Point, FP16), 브레인 부동소수점(Brain Floating Point, BFP16 [25]) 등이 있다. 16-bit을 활용하면 적어도 2배 이상의 연산 속도 향상과 2배 이상의 적은 메모리 소모를 기대할 수 있다. 그러나, 낮은 정밀도 데이터형으로 훈련할 때 두 가지 문제로 인해 성능 저하가 발생할 수 있다.

첫 번째, 스웜핑(Swamping [7])으로 인해 훈련이 제대로 되지 않을 수 있다. 스웜핑이란, 두 값을 더할 때 작은 값이 0으로 사라지는 것을 말한다. 스케일(Scale) 차이가 나는 두 값을 더할 때, 지수부(Exponent)를 일치시켜야 한다. 지수부를 일치시키려면, 작은 값의 가수부(Mantissa)를 오른쪽으로 이동(Right Shifting)시키는데, 이때 가수부의 bit 값이 사라질 수 있다. 10-bit 가수부(Mantissa)를 가진 FP16은 두

값의 스케일 차이가 2^{11} 배 이상인 경우, 7-bit 가수부를 가진 BFP16은 두 값의 스케일 차이가 2^{8} 배 이상인 경우, 작은 값이 0으로 사라진다. 또 다른 문제는 훈련 과정에서 작은 값의 그레디언트(Gradients)가 0으로 사라질 수 있다는 것이다. FP32는 지수부에 8-bit을 사용하는데, 지수부에 이보다 적은 bit를 할당하면, 값 표현 범위(Dynamic Range)가 줄어든다. 이에 따라, 작은 값 표현 범위도 줄어들게 되고, FP32에서는 0이 아니었던 작은 값이 낮은 정밀도 데이터형에서는 0로 표현될 수 있다. 인공신경망을 FP32로 훈련할 때, 0이 아니었던 작은 그레디언트 값이 0으로 표현되면서 훈련이 진행되지 않는 문제가 된다. 위 문제들은 가수부, 지수부에 적은 bit 수를 할당할수록 더욱 빈번히 일어난다.

1.2 관련 연구

1.2.1 낮은 정밀도 인공신경망 훈련

그동안 인공신경망 훈련 과정의 연산량과 메모리 요구량을 줄이기 위한 많은 연구가 진행되어왔다. 최근, 가중치(Weights)와 활성화값(Activations)을 1-bit 또는 2-bit 으로 줄이고, 그레디언트(Gradients)는 FP32로 계산하여 인공신경망을 훈련한 연구들이 진행되었다 [10, 3, 9]. DoReFa-Net은 그레디언트까지 낮은 정밀도를 활용하였는데, 1-bit 가중치, 2-bit 활성화값, 6-bit 그레디언트를 활용하여 인공신경망을 훈련하였다 [28]. 이후, 16-bit 고정소수점을 활용하는 연구도 제안되었는데, MNIST, CIFAR10 데이터셋으로 CNN(Convolutional Neural Networks) 모델을 정확도(Accuracy) 손실 없이 훈련했다 [4].

RNN(Recurrent Neural Networks) 훈련 역시 낮은 정밀도를 사용하는 연구가 진행됐다. He et al.(2016)은 GRU [2] 와 LSTM [8]의 가중치와 활성화값에 적은 bit를 할당하여 약간의 성능 저하를 보이며 훈련했다. RNN 모델을 양자화(Quantization)하는 또 다른 방식이 제안되었는데, Ott et al.(2016)은 이진(Binary), 3진(Ternary),

지수식(Exponential) 가중치 양자화를 활용하여 다양한 RNN 모델을 양자화 훈련하였고, 언어 모델링(Language Modeling)과 음성 인식(Speech Recognition)에서의 성능을 평가하였다. 위 접근 방식들은 모두 그레디언트를 FP32로 연산하여 훈련이 진행되었다. 따라서 역전파(Back Propagation)에서의 연산 비용(Computation Cost)은 줄어들지 않는다.

1.2.2 혼합 정밀도 훈련

가중치, 활성화값, 그레디언트 모두 낮은 정밀도 데이터형으로 표현하여 훈련하는 방법도 제안되었다. Micikevicius, et al.(2017)는 낮은 정밀도 데이터형으로 FP16을 활용한 인공신경망 훈련 방법인 **혼합 정밀도 훈련(Mixed Precision Training)**을 제안했다. FP32의 마스터 가중치(Master Copy of Weights)와 손실값 스케일링(Loss Scaling), FP32 accumulator를 활용해 이미지 분류(Image Classification), 언어 모델링(Language Modeling), 이미지 생성(Image Generation) 등 다양한 응용 분야에서 성능을 평가했다. Kalamkar, et al.(2019)은 또 다른 16-bit 부동소수점인 BFP16을 활용해 혼합 정밀도 훈련을 수행했다. BFP16은 FP32와 동일하게 지수부에 8-bit을 할당하여, 값 표현 범위가 FP32와 비슷하다. 따라서, FP32, BFP16 사이 형변환(Casting)이 쉽고, 손실값 스케일링을 사용하지 않아도 되는 장점이 있다. CNN, RNN, 추천 시스템(Recommendation System) 등의 다양한 응용 분야에서 훈련이 잘 되는 것을 보였다.

최근 혼합 정밀도 훈련 방식의 연장선으로, FP8(8-bit Floating Point)을 활용해서 인공신경망을 훈련한 방식이 제안되었다. Wang, et al(2018)은 FP8을 활용하여 성공적으로 ResNet을 훈련했다. 마스터 가중치를 FP32가 아닌 FP16으로 채택했다. 묶음-기반 매트릭스 곱셈(Chunk-based Dot-product)과 FP16 accumulator을 채택했고 나머지 모든 연산(Arithmetics)은 FP8로 진행했다. 스토캐스틱 라운딩(Stochastic Rounding)을 적용하여 약간의 성능 저하를 보이며 ResNet50을 훈련했다. 이후, FP8

에서 잘 동작하는 강화된 손실값 스케일링(Enhanced Loss Scaling)과 스토캐스틱 라운딩을 활용해서 훈련했다. CNN 모델뿐만 아니라 RNN 모델에서도 훈련이 되는 것을 보였다 [19].

1.3 SWA를 활용한 FP8 혼합 정밀도 훈련

그림 1.1은 본 논문에서 사용되는 FP8 혼합 정밀도 훈련 과정을 나타낸다. 가중치 업데이트와 accumulation을 제외한 나머지 모든 연산은 FP8으로 수행한다. 혼합 정밀도 훈련에서 스웜핑(Swamping)은 가중치 업데이트(Weight Update)와 매트릭스 곱셈(Matrix Multiplication) 과정의 accumulation에서 발생할 수 있다. 이를 해결하기 위해, FP32인 마스터 가중치를 도입하고, 여기에 그레디언트를 반영한다. Accumulator의 데이터형으로 FP32를 도입하여 스웜핑으로 인한 언더플로(Underflow)를 방지한다. 손실값 스케일링은 컨스턴트 손실값 스케일링(Constant Loss Scaling), 다이내믹 손실값 스케일링(Dynamic Loss Scaling [15]), 강화된 손실값 스케일링(Enhanced Loss Scaling [19]) 등 다양한 방식이 존재하는데 본 논문에서는 컨스턴트 손실값 스케일링을 활용한다.

본 논문에서는 FP8 혼합 정밀도 훈련을 진행한다. 기존의 FP8 혼합 정밀도 훈련 연구에서는 부호(Sign) 1-bit, 지수부(Exponent) 5-bit, 가수부(Mantissa) 2-bit (s1e5m2)으로 FP8을 구성한다. 본 논문에서는 지수부, 가수부에 할당되는 bit 개수를 달리하며 성능을 검증했다. 가장 우수한 FP8 bit 구성을 찾아서, 스토캐스틱 가중치 평균(Stochastic Weight Averaging, SWA)을 다양한 방식으로 적용했다. SWA는 일반화 성능이 우수한 모델을 얻는 방법 중 하나다. FP8 혼합 정밀도 훈련에 가장 적합한 SWA를 찾고 다양한 실험에서 성능 향상을 보인다. 이미지 분류, 언어 모델링, 이미지 생성 작업에서 실험을 진행했다.

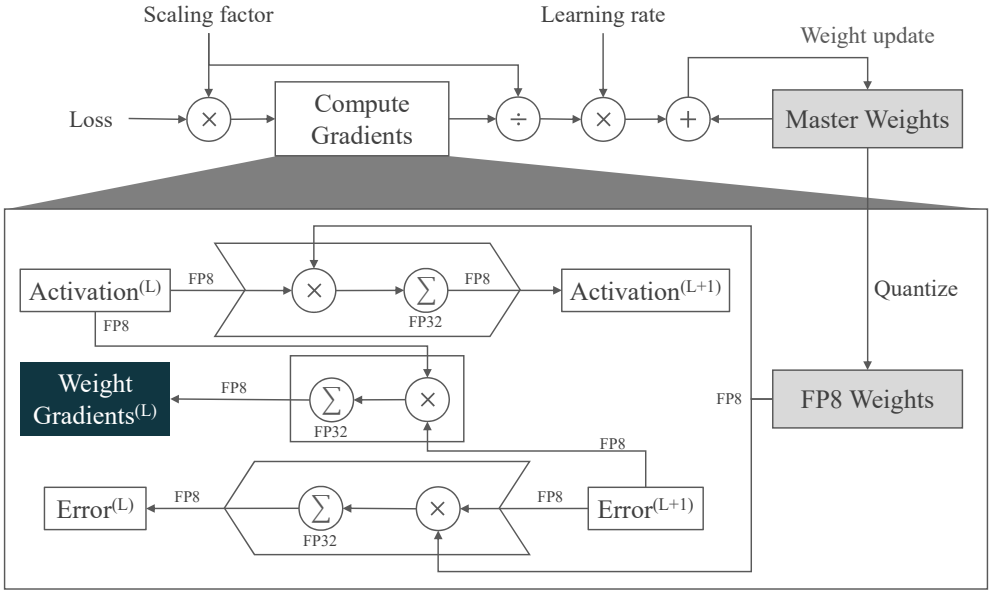


그림 1.1: FP8 혼합 정밀도 훈련 과정

1.4 논문의 개요

본 논문의 구성은 다음과 같다. 챕터 2에서는 bit 조합에 따라 FP8을 다르게 구성하여 인공신경망을 훈련해본다. 지수부, 가수부에 할당하는 bit 개수를 달리하면서 3가지 FP8의 성능을 분석한다. 훈련 종료 후, SWA도 적용해보면서 가장 적절한 FP8 bit 조합을 찾는다. 챕터 3에서는 FP8 혼합 정밀도 훈련에 다양한 방식으로 SWA를 적용해본다. 훈련 종료 후 SWA를 적용하는 방식과 훈련 중 SWA를 적용하는 방식 중에서 적합한 방식을 찾는다. 챕터 4는 본 논문의 결론이다.

제 2 장 Bit 조합에 따른 FP8 혼합 정밀도 훈련

2.1 Bit 조합에 따른 값 표현 범위

낮은 정밀도 데이터형으로 훈련할 때 가장 중요한 요소 중 하나는 값 표현 범위이다. 값 표현 범위는 지수부에 할당하는 bit 개수에 의존한다. 표 2.1은 FP32, FP16, FP8의 값 표현 범위를 나타낸다. FP8을 부호 1-bit, 지수부 5-bit, 가수부 2-bit으로 구성하는 방식(FP8-s1e5m2)은 기존 연구들에서 사용되는 방식이다. 이는 지수부에 5-bit을 사용하는 FP16과 값 표현 범위가 비슷하지만, 가수부의 bit 개수가 적으므로 FP16보다 좁은 최소 준정상 범위(Min Subnormal Range)를 가진다.

FP8의 지수부에 6-bit를 할당하면 값 표현 범위가 FP8-s1e5m2 보다 지수적으로 2배(Exponentially Doubles) 늘어난다. 값 표현 범위는 지수부 바이어스(Exponent Bias)를 달리하면 이동시킬 수 있다. FP8-s1e6m1의 넓어진 값 표현 범위를 이동 시켜 최소 준정상 범위를 늘릴 수 있는데, 이를 통해 더 작은 그레디언트 값들을 보존할 수 있다. 늘어난 최소 준정상 범위는 스케일링 팩터(Scaling Factor)에 덜 민감한 효과를 가져온다. FP8-s1e4m3은 정밀도가 높은 대신, 값 표현 범위가 상당히 좁아진다. 좁아진 값 표현 범위 때문에 인공신경망 모델에 따라서 지수부 바이어스를 조절하는 것이 훈련에 유리하다.

그림 2.1은 CIFAR10을 활용한 ResNet20 모델의 훈련 중 그레디언트 히스토그램(Histogram)을 나타낸다. FP8-s1e6m1의 지수부 바이어스를 46으로 설정하면, FP-s1e6m1 은 $[2^{-45}, 2^{16})$의 수를 표현한다. FP-s1e5m2 는 $[2^{-14}, 2^{15})$의 수를 표현하는데, FP8-s1e5m2 에서 손실값 스케일링을 쓰지 않으면, 약 11%의 그레디언트가 0으로 표현된다. 그러나, 이 그레디언트들은 FP6-s1e6m1 에서 손실값 스케일링을 쓰지 않아도 값이 유지될 수 있다.

표 2.1: FP32, FP16, FP8의 값 표현 범위 비교

Data type	Bit Format (s, e, m)	Exponent Bias	Max Normal	Min Normal	Min Subnormal
FP32	1, 8, 23	127	3.40e38	1.17e-38	1.40e-45
FP16	1, 5, 10	15	6.55e4	6.10e-5	5.96e-8
FP8	1, 6, 1	46	9.83e4	2.84e-14	1.42e-14
FP8	1, 5, 2	15	5.73e4	6.10e-5	1.52e-5
FP8	1, 4, 3	7	2.40e2	1.56e-2	1.95e-3

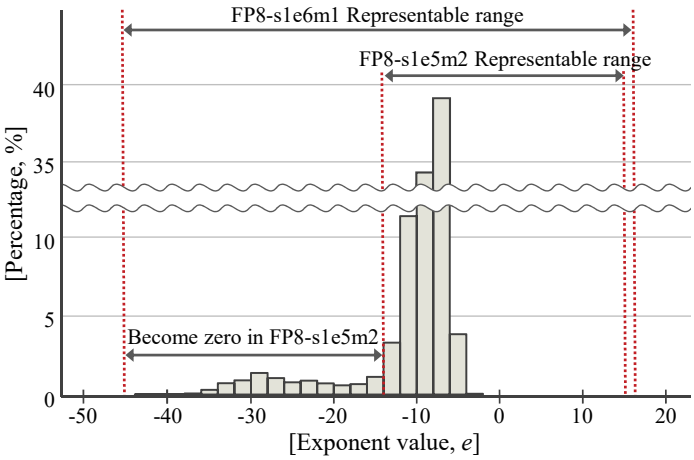


그림 2.1: ResNet20-CIFAR10 그레디언트 히스토그램

2.2 스토캐스틱 라운딩

라운딩 방식은 낮은 정밀도 인공신경망 훈련에서 중요한 역할을 수행한다. 일반적으로 사용되는 라운딩 기법은 RNE(Round to Nearest Even)와 스토캐스틱 라운딩 (Stochastic Rounding, SR [4, 26, 19])이다. 불편 라운딩 기법(Unbiased Rounding Schemes) 중 하나인 RNE는 IEEE 754 부동 연산에 기본적으로 사용되고 있다. 하드웨어에 구현하기 쉬워서 널리 사용되고 있으나, LSB(Least Significant Bit)의 모든 정보를 버린다는 단점이 존재한다. 가수부에 적은 bit 개수가 할당될수록 RNE에 의해 발생하는 정보 손실은 더욱 커진다. 스토캐스틱 라운딩은 RNE에 비해 정보 손실을 상대적으로 줄일 수 있는 효과적인 라운딩 기법이다[26, 19]. 버려지는 LSB 값을 확률값으로 계산하여 라운딩을 수행하므로, 정보 손실을 줄인다.

부호 s, 지수부 e, 가수부 m 라고 나타낸다면, 임의의 부동소수점 x는 $x = (-1)^s \cdot 2^e \cdot m$ 으로 표현할 수 있다. 이때, 라운딩 기법은 아래와 같다:

$$\text{RNE}(x,k) = \begin{cases} (-1)^s \cdot 2^e \cdot \lfloor m \rfloor_k & \text{if } 0 \leq \frac{m - \lfloor m \rfloor_k}{\epsilon} < \frac{1}{2} \\ (-1)^s \cdot 2^e \cdot (\lfloor m \rfloor_k + \epsilon) & \text{elif } \frac{1}{2} < \frac{m - \lfloor m \rfloor_k}{\epsilon} \leq 1 \\ (-1)^s \cdot 2^e \cdot \lfloor m \rfloor_k & \text{elif } \bmod (\lfloor m \rfloor_k, 2\epsilon) = 0 \\ (-1)^s \cdot 2^e \cdot (\lfloor m \rfloor_k + \epsilon) & \text{elif } \bmod (\lfloor m \rfloor_k, 2\epsilon) \neq 0 \end{cases} \tag{2.1}$$

$$\text{SR}(x,k) = \begin{cases} (-1)^s \cdot 2^e \cdot (\lfloor m \rfloor_k + \epsilon) & \text{w.p. } \frac{m - \lfloor m \rfloor_k}{\epsilon} \\ (-1)^s \cdot 2^e \cdot \lfloor m \rfloor_k & \text{w.p. } 1 - \frac{m - \lfloor m \rfloor_k}{\epsilon} \end{cases} \tag{2.2}$$

k 는 가수부의 bit 개수이고, $\epsilon = 2^{-k}$ 이다. 본 논문에서는 스토캐스틱 라운딩을 적용했다.

2.3 실험 결과

ResNet-ImageNet 실험을 제외한 본 논문의 모든 실험은 NVIDIA's Pascal, Volta GPU로 진행되었다. 실험에 사용된 GPU는 Titan Xp, Titan RTX, Titan V100이다. ResNet-ImageNet은 Google's TPU로 실험했다. 다양한 bit 조합에 따른 FP8을 실험하기 위해 TensorFlow [24]를 활용하여 FP8 컨버터(Converter)를 제작하였다. FP8 컨버터를 FP32 산술(FP32 Arithmetics)의 앞, 뒤에 붙여서 FP32 값이 FP8으로 변하도록 했다. 그림 2.2는 FP8 컨버터가 인공신경망의 산술에 적용된 방식을 나타낸다. 그림 2.2의 좌측 그림은 혼합 정밀도 훈련의 FP32 마스터 가중치를 FP8으로 변환하는 과정이고, 우측 그림은 인공신경망 모델 내부 연산의 입력과 출력을 FP8으로 변환하는 과정이다. 연산(Operations)은 컨볼루션(Convolution), 매트릭스 곱셈(Matrix Multiplication), 활성화 함수(Activation Function) 등을 포함한다.

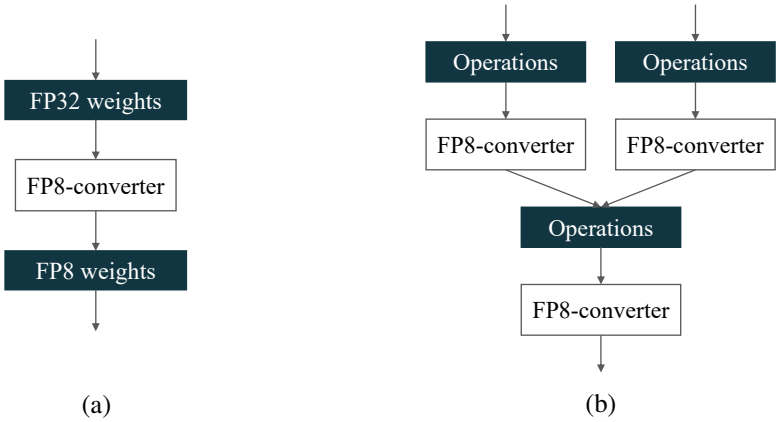


그림 2.2: FP8 컨버터가 적용된 방식

FP8 혼합 정밀도 훈련 방식은 이미지 분류 작업, 언어 모델링, 이미지 생성 작업에 대해서 성능이 평가되었다.

2.3.1 이미지 분류 작업

이미지 분류 작업의 성능을 평가하기 위한 인공신경망 모델로 ResNet [5]을 사용했다. Bit 조합별로 FP8 혼합 정밀도 훈련을 진행했다. 모델이 수렴하고 나면 SWA를 적용해서 성능을 측정했다. 훈련 데이터는 CIFAR10 [14]과 ImageNet [23]이다. 모든 실험에는 컨스턴트 손실값 스케일링을 적용했고, 스케일링 팩터로 $128k$을 사용했다. 모멘텀은 0.9이다. 성능 저하를 최대한 방지하기 위해 ResNet의 입력 레이어와 출력 레이어, 그리고 레지듀얼 연결(Residual Connection)을 FP32로 유지했다.

표 2.2: FP8 혼합 정밀도 훈련의 ResNet-CIFAR10 정확도

(a) ResNet20

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**92.17**	87.13	90.02	91.10
SWA(5x5)	92.08	87.91	90.55	**91.31**
SWA(5x10)	92.04	87.65	90.20	91.25

(b) ResNet56

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**93.46**	89.49	92.45	92.50
SWA(5x5)	93.45	90.24	92.85	92.66
SWA(5x10)	93.47	90.35	**92.86**	92.79

표 2.2는 FP8 혼합 정밀도 훈련의 ResNet20/56-CIFAR10 정확도(%)를 나타낸다. 훈련은 175 에포크 동안 진행했다. 가중치 감소 팩터(Weight Decay Factor)는 $5 \cdot 10^{-4}$이다. 초기 러닝 레이트 값은 0.1로, 75, 125번째 에포크에서 10배 씩 줄였다. 표 2.2의 첫 번째 열 SWA(XxY)는 훈련 종료 후 X 에포크 간격 Y개 모델 포인트를 평균해 SWA를 적용한 것이다. SWA를 적용하기 전, FP32-Baseline의 성능을 가장 잘 따라

가는 것은 FP8-s1e4m3이다. 훈련 종료 후 SWA를 적용하면, FP32는 성능 향상이 거의 없었지만 FP8은 성능 향상을 보였다. 특히, FP8의 가수부에 할당된 bit 개수가 적을수록 성능 향상을 많이 보였다. SWA 적용을 통해 FP8-s1e6m1은 ResNet20에서 최대 0.78%의 성능 향상을 보였고, ResNet56에서 최대 0.86%의 성능 향상을 보였다. FP8-s1e4m3 경우에 비해 성능 향상 폭이 적어도 3배 이상이다. 가수부의 bit 수가 적을수록 SWA가 필요하다.

표 2.3: FP8 혼합 정밀도 훈련의 ResNet-ImageNet 정확도

(a) ResNet18

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**70.14**	63.25	67.37	NaN
SWA(1x5)	70.23	66.78	69.03	NaN
SWA(2x5)	70.19	66.95	69.02	NaN
SWA(1x10)	70.19	67.26	**69.28**	NaN

(b) ResNet50

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**76.10**	71.55	75.17	NaN
SWA(1x5)	76.12	73.38	75.71	NaN
SWA(2x5)	76.08	73.91	75.39	NaN
SWA(1x10)	76.10	73.83	**75.95**	NaN

표 2.3은 FP8 혼합 정밀도 훈련의 ResNet18/50-ImageNet 정확도(%)를 나타낸다. 90 에포크 동안 훈련을 진행했다. 가중치 감소 팩터는 $1{\cdot}10^{-4}$이다. 러닝 레이트는 0.1에서 시작하여, 30, 60, 80번째 에포크에서 10배씩 줄였다. 이전 실험과 마찬가지로 SWA(XxY)는 훈련 종료 후, X 에포크 간격 Y개 모델 포인트를 평균해 SWA를 적용한 것이다. FP8-s1e4m3은 값 표현 범위가 좁아 손실값이 발산하며 훈련이 되지 않았

다. FP8-s1e6m1와 FP8-s1e5m2는 훈련이 가능했다. FP8-s1e6m1은 FP32와의 성능 차이가 ResNet18에서 -6.89%, ResNet50에서 -4.55%로 성능 하락이 상당히 심했다. SWA를 적용하면 전반적으로 성능 향상을 보였다. FP32의 성능 향상은 0.09%로 큰 성능 이득은 없었다. FP8 혼합 정밀도 훈련으로 얻은 모델은 SWA를 사용하면 일반화 성능이 높아졌는데, ResNet18의 경우 FP8-s1e6m1은 최대 4.01%, FP8-s1e5m2은 최대 1.91% 성능 향상이 관찰되었다. SWA를 적용한 ResNet50은 FP32에서 최대 0.02%, FP8-s1e6m1에서 최대 2.36%, FP8-s1e5m2에서 최대 0.78% 성능 향상을 보였다. FP8-s1e5m2 SWA(1x10) 성능은 75.95%로, FP32-Baseline 성능 76.10%을 거의 따라갔다.

ResNet20/56-CIFAR10과 ResNet18/50-ImageNet 실험 결과를 보면, FP8-s1e5m2가 가장 적합한 선택이라고 볼 수 있다. FP8-s1e4m3은 쉬운 작업인 CIFAR10 실험에서는 좋은 성능을 보였지만, 어려운 작업에 속하는 ImageNet 실험에서는 발산하며 훈련이 되지 않았다.

2.3.2 언어 모델링 작업

언어 모델링 작업 인공신경망으로 LSTM 기반 단어 언어 모델을 훈련하였다. 훈련 데이터로는 PTB(Penn Tree Bank [17])를 활용하였다. 모델의 기본적인 구조와 초매개변수(Hyperparameters)는 [27]를 따랐다. 레이어 개수를 1,2,4개로 설정하여 3가지 모델을 훈련했다. 실험할 때, 성능에 영향을 주는 요소를 줄이기 위해 가중치 개수를 제한했다. 총 가중치 개수가 약 $6.7M$ 이 되도록 레이어 개수가 늘어남에 따라 LSTM의 은닉층 차원(Hidden Dimension)을 줄였다. 모든 실험은 39 에포크 동안 훈련되었다. 배치 사이즈(Batch Size)는 20, 시퀀스 길이(Sequence Length)는 35이다. 초기 러닝 레이트는 1에서 시작하여 훈련 6 에포크 이후부터 매 에포크마다 0.8을 곱해서 줄였다. 어느 정도 모델이 수렴한 이후에는 주기형 러닝 레이트를 사용했다. 1xLSTM-920은 주기가 3 에포크, 나머지는 주기가 4 에포크이다. 성능 저하를

줄이기 위해 임베딩 레이어(Embedding Layer)와 출력 레이어는 FP32로 유지했다. FP8-s1e4m3은 지수부 바이어스를 11로 설정했다.

표 2.4는 FP8 혼합 정밀도 훈련의 LSTM 기반 단어 언어 모델의 테스트 복잡도를 나타낸다. 전반적으로 모델의 깊이가 얕은 경우, FP8이 FP32의 성능을 잘 따라갔다. 그러나 레이어가 깊어질수록 FP32와 비교하여 FP8의 성능 저하가 관찰되는데, 특히 가수부가 적은 FP8-s1e6m1의 성능 저하가 가장 심했다. 훈련이 종료된 뒤 SWA를 적용하면 대다수 성능 향상을 보였다. FP8-s1e5m2는 SWA를 적용하면 1xLSTM-920, 2xLSTM-650에서는 FP32-Baseline의 성능을 추월했으나, 4xLSTM-460에서는 성능 하락(-2.53)을 보였다.

표 2.4: FP8 혼합 정밀도 훈련의 WLM-PTB 복잡도

(a) 1xLSTM-920

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**80.74**	83.52	80.93	80.44
SWA(3x4)	79.48	82.84	80.03	**79.64**

(b) 2xLSTM-650

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**85.30**	87.18	86.29	85.58
SWA(4x4)	84.17	85.21	**84.38**	84.53

(c) 4xLSTM-460

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**94.50**	102.21	98.20	96.27
SWA(4x4)	93.85	101.56	97.03	**95.74**

2.3.3 이미지 생성 작업

이미지 생성 작업에 대해서 FP8 혼합 정밀도 훈련을 진행했다. DCGAN [22]을 CIFAR10 데이터로 훈련하고 성능을 측정했다. 생성자(Generator)은 1개의 완전 연결 레이어(Fully Connected Layer), 3개의 디컨볼루션 레이어(Deconvolutional Layer)와 1개의 Tanh 레이어 및 ReLU로 이루어졌다. 판별자(Discriminator)는 3개의 컨볼루션 레이어(Convolutional Layer)와 1개의 완전 연결 레이어 및 Leaky ReLU로 구성되었다. 아담(Adam [13])으로 50000 스텝(Steps) 훈련했으며, 성능 저하를 줄이기 위해 입력 레이어와 출력 레이어는 FP32로 유지했다. 성능을 평가할 때는 생성 이미지 50000개를 사용했다. 성능 지표로 인셉션 스코어(Inception Score, IS)와 프레체 인셉션 디스턴스(Frechet Inception Distance, FID)를 사용했다. 표 2.5는 FP8 혼합 정밀도 훈련의 DCGAN-CIFAR10 성능을 보인다. FP8-s1e4m3의 성능이 가장 좋았다. FP8-s1e5m2의 성능을 FP32-Baseline과 비교하면, SWA를 적용하기 전에는 IS -0.06, FID -0.64의 성능 저하를 보였지만, SWA를 사용하는 경우 성능을 추월했다.

표 2.5: FP8 혼합 정밀도 훈련의 DCGAN-CIFAR10 성능

(a) Inception Score

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**4.20**	4.05	4.14	4.25
SWA(1x5)	4.20	4.02	4.19	4.24
SWA(2x5)	4.24	4.03	4.25	**4.29**

(b) Frechet Inception Distance

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	**6.37**	6.84	7.01	5.93
SWA(1x5)	6.24	6.71	6.59	5.65
SWA(2x5)	5.62	6.58	5.90	**5.61**

다음으로 DCGAN을 CELEBA [16] 데이터로 훈련했다. 생성자은 5개의 디컨볼루션 레이어와 1개의 Tanh 레이어 및 ReLU로 이루어졌다. 판별자는 5개의 컨볼루션 레이어와 Leaky ReLU로 구성되었다. 아담으로 30000 스텝 훈련을 진행했다. 입력 레이어와 출력 레이어는 FP32로 유지했다. 표 2.6은 FP8 혼합 정밀도 훈련의 DCGAN-CELEBA MS-SSIM(Multi Scale SSIM)을 나타낸 것이다. MS-SSIM 값이 작으면 다양한 이미지를 생성한다는 것을 의미한다. 50000개 이미지 샘플로 MS-SSIM을 측정했다. 실제 CELEBA 데이터의 MS-SSIM 값은 약 0.22이다. 본 논문에서 훈련한 DCGAN-CELEBA의 MS-SSIM 값은 0.22를 상회했는데, 이는 생성되는 이미지들이 실제 데이터보다 서로 유사하다는 것을 의미한다. FP8 DCGAN 실험의 MS-SSIM 값은 약 0.03 이하의 측정 오차를 보였는데, 이는 FP32의 성능에 잘 도달했다고 볼 수 있을 것이다. 이를 확인하기 위해 각 모델에서 이미지를 생성했다.

표 2.6: FP8 혼합 정밀도 훈련의 DCGAN-CELEBA MS-SSIM

	FP32	FP8 (s1e6m1)	FP8 (s1e5m2)	FP8 (s1e4m3)
Baseline	0.270	0.270	0.262	0.261
SWA(1x5)	0.263	0.288	0.264	0.271

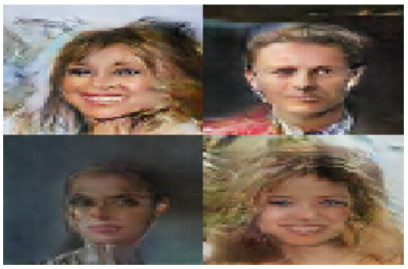
그림 2.3은 FP8 혼합 정밀도 훈련으로 얻은 DCGAN-CELEBA 이미지 생성을 나타낸다. 좌측 생성 이미지들은 데이터형별 훈련 종료 후 생성된 이미지를 나타낸다. 우측의 생성 이미지들은 데이터형별 훈련 종료 후 SWA를 적용하여 생성된 이미지를 나타낸다. 전반적으로 SWA를 활용하면 눈동자, 코, 입술, 머릿결 그리고 배경과의 경계선 등이 선명해지는 양상이 나타났다.

(a) FP32

(b) FP32(SWA)

(c) FP8-s1e6m1

(d) FP8-s1e6m1(SWA)



(e) FP8-s1e5m2

(f) FP8-s1e5m2(SWA)

(g) FP8-s1e4m3

(h) FP8-s1e4m3(SWA)

그림 2.3: FP8 혼합 정밀도 훈련의 DCGAN-CELEBA 이미지 생성

2.4 결론

본 챕터에서는 FP8의 지수부, 가수부에 할당되는 bit 개수를 달리하며, 혼합 정밀도 훈련을 수행했다. 실험 결과에 의하면 FP8을 구성하는 bit 조합으로 FP8-s1e5m2가 가장 적절해 보인다. 훈련이 성공적으로 되는 경우, FP8-s1e4m3의 성능이 가장 우수했지만, ImageNet 이미지 분류와 같이 복잡한 작업에서는 값 표현 범위가 좁아서 FP8-s1e4m3이 훈련되지 않았다. FP8-s1e6m1은 모든 실험에서 훈련은 됐지만, 성능 하락이 심했다. 특히 ImageNet 데이터 이미지 분류와 같이 복잡한 작업이나, 레이어가 깊은 복잡한 모델에서 성능 저하는 더욱 심해졌다. FP8-s1e5m2는 수행한 실험에서 훈련이 모두 성공적으로 됐다. 약간의 성능 하락이 존재하지만, FP8-s1e6m1에 비해 성능 저하가 덜했다.

FP8 혼합 정밀도 훈련 종료 후 SWA를 적용했는데, 대다수 성능 향상을 보였다. SWA 적용을 통한 성능 향상은 FP8에서 두드러졌고, FP8의 가수부 bit 개수가 적을수록 성능 향상은 커졌다. 절대적인 성능은 FP8-s1e4m3이 우수했지만, 성능 향상 폭은 FP-s1e6m1이 가장 컸다. 가수부 bit 개수가 적으면 SWA를 반드시 적용해야 하는 것으로 보인다. FP8-s1e5m2는 SWA를 적용할 경우, FP32-Baseline의 성능을 거의 추적했다.

제 3 장 SWA를 활용한 FP8 혼합 정밀도 훈련

3.1 스토캐스틱 가중치 평균

Izmailov, et al(2018)은 SGD(Stochastic Gradient Descent)로 얻은 손실값 함수(Loss Function)의 기하학적 구조를 보고, 스토캐스틱 가중치 평균(Stochastic Weight Averaging, SWA)를 제안했다. 훈련 중에 얻은 여러 모델의 가중치 평균(Averaging of Weights)이 구하면 기존 훈련 방식 SGD로 얻은 가중치(W)보다 일반화 성능이 높다고 주장했다. 알고리즘 1은 가중치 평균(W_{SWA})을 구하는 전체 과정을 나타낸다. 러닝 레이트(Learning Rate)는 상수형 러닝 레이트(Constant Learning Rate) 또는 주기형 러닝 레이트(Cyclical Learning Rate)를 사용한다. 알고리즘 1의 13번째 줄과 같이 가중치들을 훈련 도중에 업데이트하면, W_{SWA}를 계산하는데 요구되는 메모리 오버헤드(Memory Overhead)를 줄일 수 있다.

Algorithm 1 Stochastic Weight Averaging

Require:
1: initial weight $\hat{w}$, cycle length c, learning rate schedule $\alpha \in R^c$,
2: number of epochs N, steps per an epoch T

Ensure: w_{SWA}
3: $w \leftarrow \hat{w}$, $w_{SWA} \leftarrow w$, $n_{models} \leftarrow 1$
4: **for** $i \leftarrow 1$ to N **do** ▷ train for N epochs
5: $\alpha \leftarrow \alpha(i)$
6: **for** $j \leftarrow 1$ to T **do** ▷ update weight T times per an epoch
7: $w \leftarrow w - \alpha \bigtriangledown L_j(w)$ ▷ stochastic gradient descent
8: **end for**
9: **if** mod(i, c) = 0 **then**
10: $n_{models} \leftarrow i/c$
11: $w_{SWA} \leftarrow \frac{w_{SWA} \cdot n_{models} + w}{n_{models}+1}$ ▷ update weight average
12: **end if**
13: **end for**

3.2 SWA 적용 방식

SWA를 혼합 정밀도 훈련에 활용하는 방식은 크게 두 가지로 나눌 수 있다. 첫 번째는 훈련이 종료되고 나서 SWA를 적용하는 것이다. 본 논문에서는 훈련이 끝난 뒤, 다양한 모델 포인트로부터 W_{SWA}를 구했고 이들의 성능을 비교했다. 두 번째는 훈련 중 SWA를 적용하면서 훈련을 진행하는 것이다. 본 논문에서는 훈련 중 모델 가중치 (W)를 W_{SWA}로 1회 교체하는 방식과 훈련 중 W를 W_{SWA}로 매 에포크마다 교체하는 방식을 비교했다. 이러한 SWA 적용 방식은 ResNet-ImageNet, ResNet-CIFAR10, LSTM 기반 WLM-PTB에서 실험되었다.

3.3 실험 결과

첫 번째 방식으로, 훈련 종료 후 W_{SWA}를 적용해보았다. 훈련 데이터는 ImageNet 이고, 인공신경망 모델은 ResNet18이다. 그림 3.1은 러닝 레이트 스케줄과 W_{SWA}을 구하는데 사용된 모델 포인트를 나타낸다. SWA(XxY)는 W_{SWA}를 계산할 때, X 에포크(Epoch) 간격으로 Y 개의 모델을 사용한 것을 의미한다. 표 3.1은 다양한 모델 포인트로부터 구한 W_{SWA} 별 정확도(%)를 나타낸다. 훈련 도중에는 W_{SWA}를 활용하지 않고, 훈련이 끝나면 W_{SWA}를 적용해 성능을 측정했다. Baseline은 기존 훈련 방식인 SGD를 통해 얻은 모델로, SWA를 적용하지 않은 것이다. FP32는 SWA를 적용했을 때 SWA(1x5)에서 최대 성능 70.23%을 얻었다. 이를 FP32-Baseline 성능 70.14%와 비교하면 최대 성능 향상 폭으로 약 0.07%을 얻는다. FP32-SWA(5x10)은 69.70%으로 오히려 성능 저하를 보였다. 반면, FP8-s1e5m2는 모든 경우 성능 향상을 보였다. 최대 성능 향상인 FP8-s1e5m2-SWA(1x10)를 FP8-s1e5m2-Baseline과 비교하면 약 1.91% 성능 향상을 얻었다. FP8-s1e5m2 실험 결과 중 SWA(1x5), SWA(2x5), SWA(1x10)의 정확도가 다른 것보다 성능이 좋다. 이들은 동일한 러닝 레이트 구간의 모델들로부터 W_{SWA}를 구한 것이다. 이를 통해 W_{SWA}는 동일한 러

닝 레이트 구간의 모델 포인트로 계산해야 함을 알 수 있다.

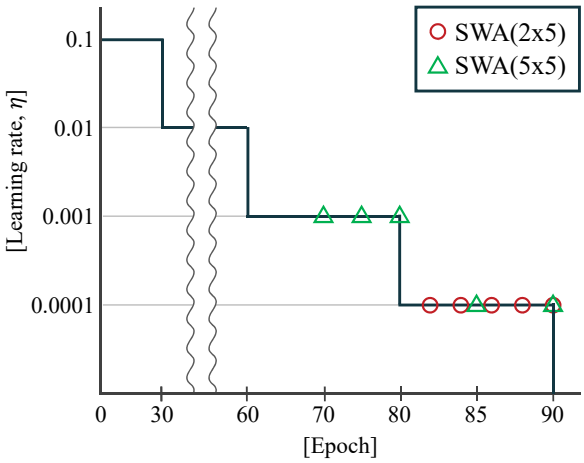


그림 3.1: ResNet18-ImageNet 훈련의 러닝 레이트 스케줄 및 다양한 모델 포인트

표 3.1: 다양한 모델 포인트를 활용한 SWA의 ResNet18-ImageNet 정확도 비교

	FP32	FP8-s1e5m2
Baseline	70.14	67.37
SWA(1x5)	**70.23**	**69.03**
SWA(2x5)	**70.19**	**69.02**
SWA(1x10)	**70.19**	**69.28**
SWA(3x5)	70.16	68.98
SWA(4x5)	70.18	68.87
SWA(5x5)	70.10	68.72
SWA(5x10)	69.70	68.60

두 번째 방식으로, 훈련 중 모델의 가중치(W)를 W_{SWA}로 교체($W \leftarrow W_{SWA}$)하며 훈련하는 방식을 분석했다. 본 논문에서는 훈련 중에 W를 W_{SWA}로 1회 교체하는 실험과, 훈련 중에 W를 W_{SWA}로 매 에포크마다 교체하는 실험을 비교했다. 모델 성능은 ResNet20-CIFAR10과 PTB-WLM에서 검증되었다.

표 3.2는 훈련 중 SWA를 1회 적용할 때 ResNet20-CIFAR10 정확도(%)를 나타낸다. 모델 훈련 특정 에포크에서 모델의 가중치 W를 W_{SWA}로 1회 교체한다. 2번째 열의 FP8은 훈련 중 W를 W_{SWA}으로 교체하지 않은 것이다. 3∼4번째 열인 FP8($W \leftarrow W_{SWA}$)는 훈련 중 W를 W_{SWA}로 교체한 것이다. 3번째 열(Once, 75, 1x10)은 모델 훈련 75번째 에포크에서 W를 W_{SWA}으로 1회 교체한 것이고, 4번째 열(Once, 125, 1x10)은 모델 훈련 125번째 에포크에서 W를 W_{SWA}으로 1회 교체한 것이다. (Once, 75, 1x10), (Once, 125, 1x10) 실험 모두, 교체할 때 사용하는 W_{SWA}로 1 에포크 간격 10개 모델 포인트를 평균내서 사용했다. 실험 결과를 보면, 훈련 중 SWA를 적용하지 않은 것이 성능이 우수했다. FP8-Baseline의 성능은 90.02%로, SWA를 1회 적용한 3∼4번째 열의 Baseline 결과보다 좋다. 성능 향상을 보인 SWA(5x5), SWA(5x10)는 훈련이 끝나고 나서 SWA를 적용한 결과였다.

표 3.2: 훈련 중 SWA를 1회 적용할 때 ResNet20-CIFAR10 정확도

	FP8	FP8($W \leftarrow W_{SWA}$)	
		(Once, 75, 1x10)	(Once, 125, 1x10)
Baseline	**90.02**	89.87	89.10
SWA(5x5)	90.20	90.40	90.23
SWA(5x10)	90.55	90.71	90.94

표 3.3은 훈련 중 SWA를 매 에포크 적용할 때 ResNet20-CIFAR10 정확도(%)를 나타낸다. 3번째 열(All, 75∼, 1x5)은 모델 훈련 75번째 에포크 이후부터 SWA

를 적용한다. SWA 적용에 사용하는 W_{SWA}는 1 에포크 간격 5개 모델(1x5)을 평균 내서 구한다. 4번째 열(All, 75∼, 1x10)은 모델 훈련 75번째 에포크 이후부터 SWA를 적용한다. 1 에포크 간격 10개 모델(1x10)을 평균해서 매 에포크마다 SWA를 적용한다. 실험 결과를 보면, (All, 75∼, 1x5), (All, 75∼, 1x10)의 Baseline 성능은 각각 89.19%, 89.36%으로 FP8-Baseline 성능인 90.02%를 넘지 못했다. 반면, 훈련 종료 후에 SWA를 적용하면 소폭의 성능 이득이 존재했다. (All, 75∼, 1x5)은 약 0.55%, (All, 75∼, 1x10)은 약 0.11% 성능 향상을 보였다.

표 3.3: 훈련 중 SWA를 매 에포크 적용할 때 ResNet20-CIFAR10 정확도

	FP8	FP8($W \leftarrow W_{SWA}$)	
		(All, 75∼, 1x5)	(All, 75∼, 1x10)
Baseline	**90.02**	89.19	89.36
SWA(5x5)	90.20	89.49	89.47
SWA(5x10)	90.55	89.74	89.47

LSTM 기반의 단어 언어 모델 (Word level Language Model, WLM) 에 대해서도 성능을 검증했다. PTB 데이터로 모델을 39 에포크 훈련했다. 그림 3.2는 단어 언어 모델 훈련에 사용된 2가지 러닝 레이트 스케줄로 상수형 러닝 레이트와 주기형 러닝 레이트를 나타낸다. 모델이 어느 정도 수렴한 15 에포크 이후, 상수형 및 주기형 러닝 레이트를 적용했다. 상수형 러닝 레이트 값은 약 0.203이다. 주기형 러닝 레이트의 주기는 3 에포크이고, 1주기의 러닝 레이트는 에포크 간격으로 1, 0.5, 0.25로 바뀐다.

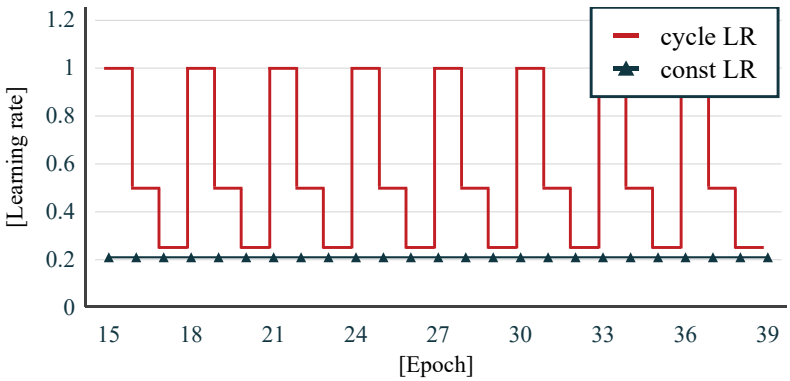


그림 3.2: WLM-PTB의 러닝 레이트 스케줄

그림 3.3은 상수형 러닝 레이트를 사용할 때 가중치 교체 방식별 WLM-PTB 밸리데이션 복잡도(Validation Perplexity) 양상을 나타낸다. Baseline은 훈련 중 W를 교체하지 않은 것이다. $W \leftarrow W_{SWA}$(Once, X)는 X 에포크에서 1회 W를 W_{SWA}로 교체한 것이다. $W \leftarrow W_{SWA}$(All, 20∼)은, 20 에포크 이후로 매 에포크마다 W를 W_{SWA}로 교체한 것이다. SWA 적용에 사용되는 W_{SWA}는 1 에포크 간격 5개 모델(1x5)을 평균화했다. 그림 3.3에서 매 에포크 W를 W_{SWA}로 교체하는 방식은 모델의 수렴 속도를 낮추는 양상을 보인다. 훈련 중 SWA를 1회 적용하는 것은 모델이

수렴하는 데 큰 영향을 미치지 않았다.

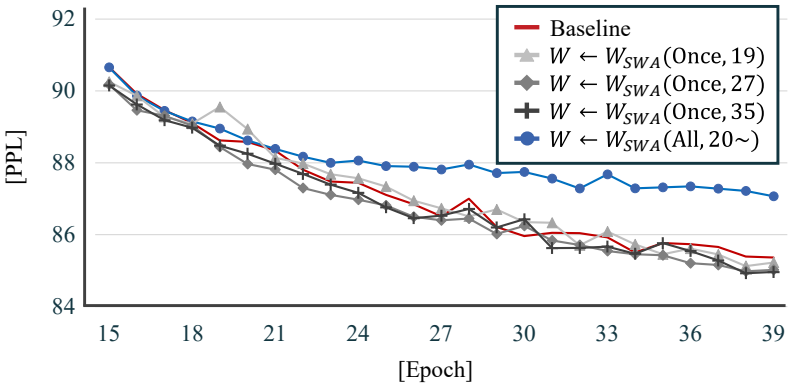


그림 3.3: 가중치 교체 방식별 상수형 러닝 레이트 WLM-PTB 훈련 양상

표 3.4는 상수형 러닝 레이트를 사용할 때 가중치 교체 방식별 WLM-PTB 훈련 결과를 나타낸다. 훈련 중 SWA를 적용하지 않은 FP8-Baseline의 복잡도가 81.91로, 가장 성능이 좋았으며, 매 에포크 SWA를 적용하는 (All, 20~)은 가장 큰 성능 저하를 보였다. W를 W_{SWA}로 1회 교체하는 방식은 성능 이득이 없었는데, (All, 20~)과 비교하면 성능 저하가 크지 않았다. 훈련이 끝난 뒤, SWA를 적용했는데, 3 에포크 간격 4개 모델을 평균한 SWA(3x4)의 방식은 모든 경우 성능 향상을 보였다.

표 3.4: 가중치 교체 방식별 상수형 러닝 레이트 WLM-PTB 훈련 결과

	FP8	FP8($W \leftarrow W_{SWA}(1 \times 5)$)			
		(Once, 19)	(Once, 27)	(Once, 35)	(All, 20~)
Baseline	**81.91**	82.08	82.63	81.93	84.29
SWA(3x4)	81.69	81.98	82.28	81.78	83.83

그림 3.4는 주기형 러닝 레이트를 사용할 때 가중치 교체 방식별 WLM-PTB 밸리데이션 복잡도 양상을 나타낸다. Baseline은 훈련 중 SWA를 적용하지 않은 것이다. $W \leftarrow W_{SWA}$(Once, 18)은, 모델 훈련 18번째 에포크에서 1회 W를 W_{SWA}로 교체하고 훈련을 진행한다. $W \leftarrow W_{SWA}$(All, 18∼)은 모델 훈련 18번째 에포크 이후, 매 에포크마다 W를 W_{SWA}로 교체하며 훈련을 진행한다. 사용되는 W_{SWA}는 1 에포크 간격 5개 모델을 평균화해서 구했다. 상수형 러닝 레이트 훈련에서와 유사하게, 주기형 러닝 레이트 훈련 방식에서도 매 에포크 W를 W_{SWA}로 교체하는 방식은 모델의 수렴 속도를 늦추었다.

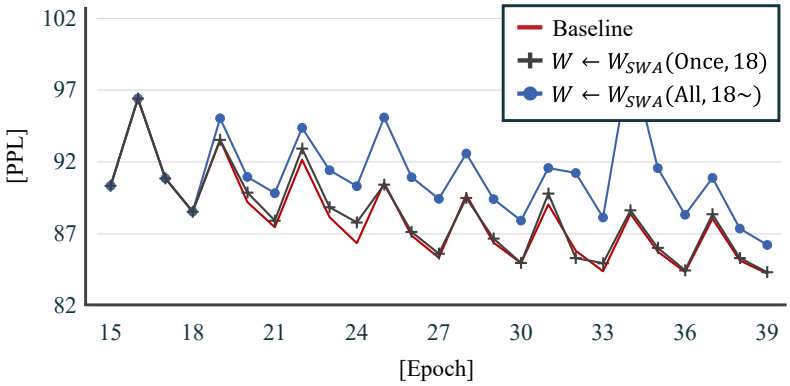


그림 3.4: 가중치 교체 방식별 주기형 러닝 레이트 WLM-PTB 훈련 양상

표 3.5는 주기형 러닝 레이트를 사용할 때 가중치 교체 방식별 WLM-PTB 훈련 결과를 나타낸다. 훈련 중 SWA를 적용하지 않는 것의 복잡도가 80.03으로, 성능이 가장 좋았다. W를 W_{SWA}로 1회 교체하는 방식은 성능 이득을 보이지 않았고, (All,

18∼)은 가장 큰 성능 저하를 보였다.

표 3.5: 가중치 교체 방식별 주기형 러닝 레이트 WLM-PTB 훈련 결과

	FP8	FP8($W \leftarrow W_{SWA}(1 \times 5)$)	
		(Once, 18)	(All, 18∼)
Baseline	**80.93**	81.07	83.15
SWA(3x4)	80.03	80.62	83.62

3.4 결론

지금까지 FP8 혼합 정밀도 훈련에 SWA를 적용하는 방식에 대해 살펴보았다. ResNet-ImageNet, ResNet-CIFAR10과 LSTM 기반 WLM-PTB에서 실험했다. 우선 훈련 종료 후 SWA를 적용하는 방식을 분석했는데, 대다수 실험에서 일반화 성능을 높이는 것으로 나타났다. 훈련 도중에 SWA를 적용하는 방식으로는 두 가지 방식을 실험했다. 훈련 중 1회 SWA를 적용하는 방식과 훈련 중 매 에포크 SWA를 적용하는 방식을 분석했다. 실험 결과에 의하면, 훈련 중 SWA를 적용하는 두 방식 모두 성능 향상이 없었다. 특히 매 에포크 SWA를 적용하는 방식은 모델의 수렴 속도를 낮춰서 큰 성능 저하를 보였다. 따라서, 본 논문에서는 FP8 혼합 정밀도 훈련에 SWA를 적용할 때, 훈련 중에는 사용하지 않고 훈련이 끝난 후에 W_{SWA}를 사용할 것을 제안한다.

제 4 장 결론

본 논문에서는 8-bit 부동소수점(FP8)의 가수부, 지수부 bit를 달리하며 혼합 정밀도 훈련을 진행했다. 또한, FP8 혼합 정밀도 훈련에 다양한 방식으로 SWA를 적용해 보았다. FP8을 부호 1-bit, 지수부 5-bit, 가수부 2-bit(s1e5m2)로 구성하는 것이 가장 우수한 것으로 확인됐다. 한편, 훈련 중 SWA를 적용하는 것은 성능 이득이 없었고, 훈련 종료 후 SWA를 적용하는 것이 우수한 성능을 보인다는 것을 확인했다.

먼저, 8-bit 부동소수점의 가수부, 지수부에 bit 개수를 다르게 할당하며 인공신경망 모델에서 성능을 살펴보았다. FP8-s1e4m3은 훈련이 가능한 작은 모델에서는 가장 우수한 성능을 보이지만, 값 표현 범위가 좁아서 큰 모델에서는 훈련이 되지 않았다. FP8-s1e6m1은 가장 큰 성능 저하를 보였는데, 모델의 층수가 깊어질수록 성능 하락은 심해졌다. FP8-s1e5m2은 큰 모델에서도 훈련이 가능했고, 약간의 성능 저하를 보이며 FP32 성능을 대부분 추적했다. 각각의 데이터형에 SWA를 적용해 성능을 분석한 결과, 거의 모든 경우 훈련 종료 후 SWA를 적용하면 성능 향상을 보였다. FP8 가수부에 할당된 bit 개수가 적을수록 SWA 활용으로 인한 성능 향상 폭이 컸다. SWA를 적용한 FP8-s1e5m2는 SWA를 적용하지 않은 FP32-Baseline의 성능을 거의 따라갔다.

다음으로, FP8 혼합 정밀도 훈련에 SWA를 활용하는 방식을 알아보았다. 훈련이 끝나고 나서 SWA를 적용하는 방식으로 다양한 모델 포인트의 SWA를 적용해보았다. 동일 러닝 레이트 구간에서 모델 포인트를 선택하는 것이 그렇지 않은 것보다 성능 향상이 우수했다. 또한, 훈련 중 SWA를 활용하는 방안에 대해서 살펴보았다. 훈련 중 1회 SWA를 적용하는 방식과 매 에포크 SWA를 적용하는 방식을 분석했다. 대다수 실험 결과에 의하면 훈련 중 SWA를 적용하지 않는 것이 그렇지 않은 경우보다 좋았다. 특히, 매 에포크마다 SWA를 활용하는 것은 훈련 속도를 낮춰서 모델이 수렴하지 않는 결과를 낳았다.

참고 문헌

[1] Amodei, Dario, et al, “Deep speech 2: End-to-end speech recognition in english and mandarin,” *International conference on machine learning*, 2016.

[2] Cho, Kyunghyun, et al, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[3] Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” *Advances in neural information processing systems*, 2015.

[4] Gupta, Suyog, et al, “Deep learning with limited numerical precision,” *International Conference on Machine Learning*, 2015.

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

[6] He, Qinyao, et al, “Effective quantization methods for recurrent neural networks,” *arXiv preprint arXiv:1611.10176*, 2016.

[7] Higham, Nicholas J, “The accuracy of floating point summation,” *SIAM Journal on Scientific Computing*, vol. 14, no. 3, pp. 783-799, 1993

[8] Hochreiter, Sepp, and Jürgen Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[9] Hubara, Itay, et al, “Binarized neural networks,” *Advances in neural information processing systems*, 2016.

[10] Hwang, Kyuyeon and Sung, Wonyong, "Fixed-point feedforward deep neural network design using weights+ 1, 0, and 1," *2014 IEEE Workshop on Signal Processing Systems*, 2014.

[11] Izmailov, Pavel, et al, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[12] Kalamkar, Dhiraj, et al, "A Study of BFLOAT16 for Deep Learning Training," *arXiv preprint arXiv:1905.12322*, 2019.

[13] Kingma, Diederik P., and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] Krizhevsky, Alex, and Geoffrey Hinton, "Learning multiple layers of features from tiny images," vol. 1, no. 4, Technical report, University of Toronto, 2009.

[15] Kuchaiev, Oleksii, et al, "Mixed-precision training for nlp and speech recognition with openseq2seq," *arXiv preprint arXiv:1805.10387*, 2018.

[16] Liu, Ziwei, et al, "Deep learning face attributes in the wild," *Proceedings of the IEEE international conference on computer vision*, 2015.

[17] Marcus, Mitchell, et al, "The Penn Treebank: annotating predicate argument structure," *Proceedings of the workshop on Human Language Technology. Association for Computational Linguistics*, 1994.

[18] McCann, Bryan, et al, "Learned in translation: Contextualized word vectors," *Advances in Neural Information Processing Systems*, 2017.

[19] Mellempudi, Naveen, et al, "Mixed Precision Training With 8-bit Floating Point," *arXiv preprint arXiv:1905.12334*, 2019.

[20] Micikevicius, Paulius, et al, "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

[21] Ott, Joachim, et al, "Recurrent neural networks with limited numerical precision," *arXiv preprint arXiv:1608.06902*, 2016.

[22] Radford, Alec, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[23] Russakovsky, Olga, et al, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 221-252, 2015.

[24] TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/?hl=en

[25] Using bfloat16 with tensorflow models. https://cloud.google.com/tpu/docs/bfloat16

[26] Wang, Naigang, et al, "Training deep neural networks with 8-bit floating point numbers," *Advances in neural information processing systems*, 2018.

[27] Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[28] Zhou, Shuchang, et al, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

ABSTRACT

In recent years, Artificial Neural Networks(ANN) perform well in a variety of applications. To achieve better performance, complex models and large data are required, which necessitates a significant amount of computation and memory resources during training. In this paper, mixed precision training with 8-bit floating point format(FP8) is performed to reduce the amount of computation and memory resources. We analyzed how many bits are best for exponent and mantissa of the FP8. We also explored various ways of applying Stochastic Weighted Averaging(SWA) to FP8 mixed precision training and found the best way of them.

According to various experimental results, it is most suitable to apply SWA after FP8 mixed precision training with FP8 of sign 1-bit, exponent 5-bit and mantissa 2-bit(FP8-s1e5m2). In this paper, we verified the model performance according to the combination of bits with different numbers of exponent and mantissa bits of FP8. The FP8-s1e4m3 has good performance, but its narrow value representation makes it difficult to train in complex tasks. A neural network with FP8-s1e6m1 can be trained, but the deeper the model, the more severe the performance degradation. FP8-s1e5m2 showed some performance degradation. The applying SWA to FP8-s1e5m2 almost followed the performance of the FP32-Baseline. In addition, this paper has applied SWA in various ways to FP8 mixed precision training. Applying SWA after training has improved performance in most experiments. However, applying SWA once during training or applying SWA at every epoch during training have not shown any improvement.

주요어: Mixed Precision Training, 8-bit Floating Point(FP8), Stochastic Weight Averaging(SWA)

학번: 2018-27706