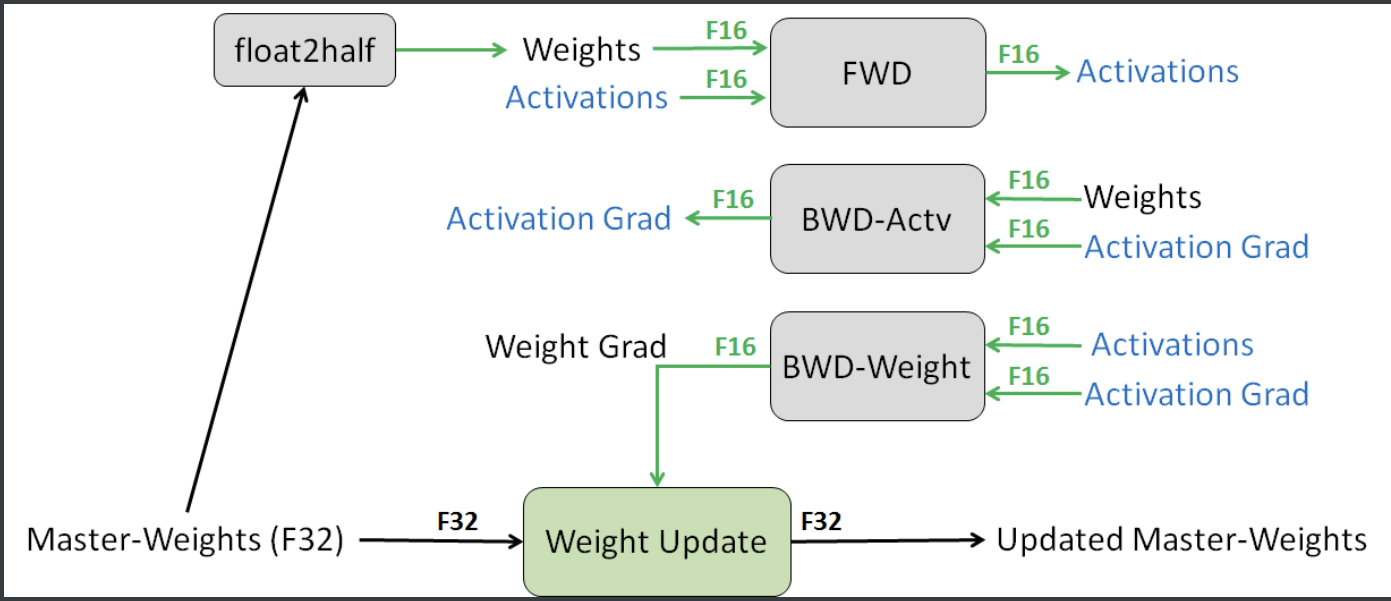
# Mixed Precision Training

딥러닝이 발전함에 따라, 다양한 분야에서 큰 발전이 이루어졌다. 이러한 발전에는 training dataset의 규모와 모델의 복잡도가 증가하는 것이 뒤따라왔다. 복잡하고 큰 모델들은 훈련시키기 위해 더 많은 computing resources를 필요로 하는데, computing resource의 한계로 인해 모델의 성능이 제한되는 결과를 초래할 수 있다.

이러한 computing resource에 대한 필요는 축소된 정밀도 계산으로 보다 완화할 수 있다. 논문에서는 single-precision format(FP32, 32비트 부동소수점)형식을 사용했던 기존 방식과 달리 half-precision format(FP16, 16비트 부동소수점)사용하여 모델의 정확도는 유지하되, 훈련 속도와 필요한 computing resource를 줄이는 방법론을 제시한다.




mixed precision training에서, weight, activations, gradient는 FP16으로 저장된다.

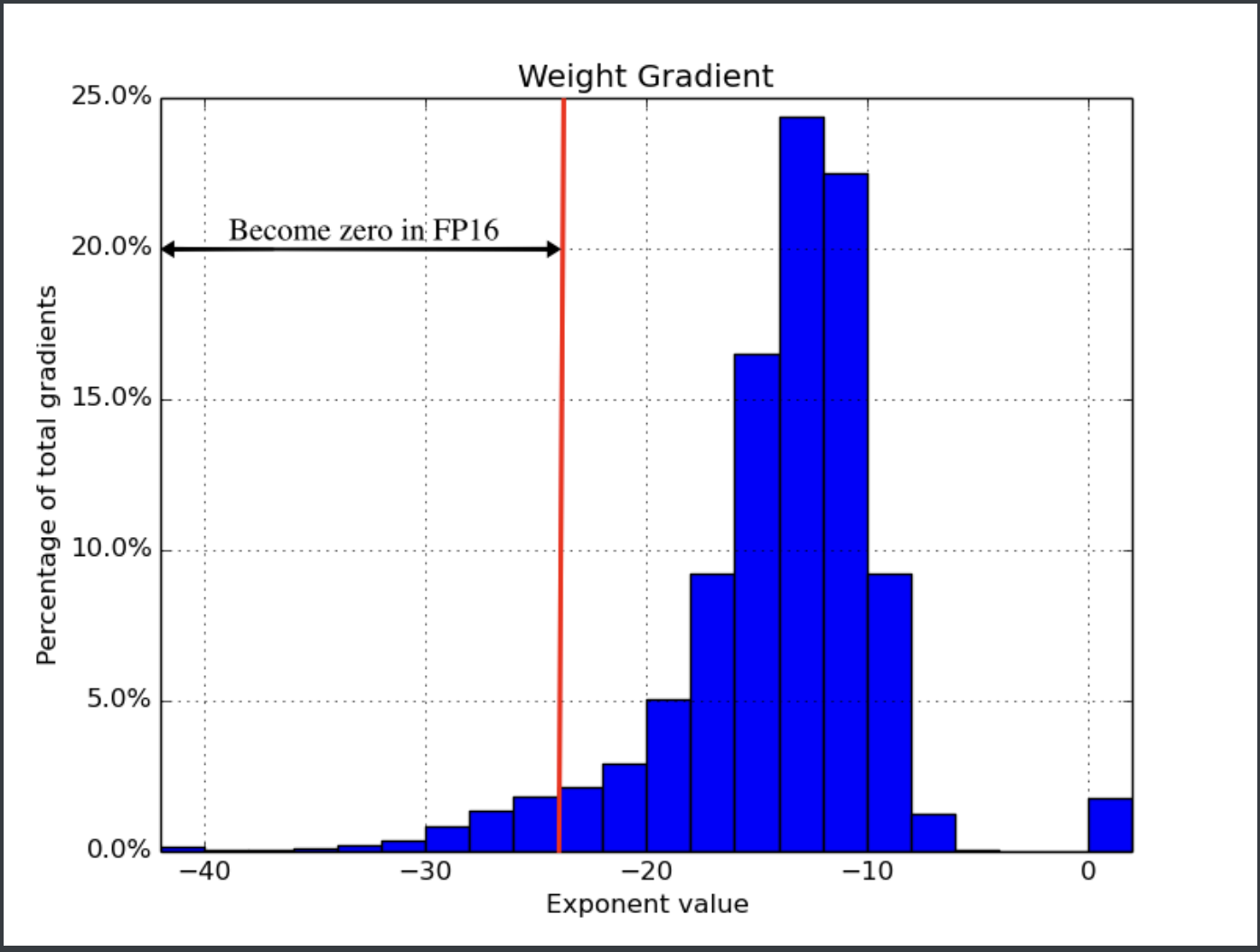
FP32 훈련 세션의 정확도를 일치시키기 위해, weight의 FP32 master copy가 유지되고 이는 optimizer step에 이 weight가 업데이트된다. master weight의 FP16 복사본은 forward와 backward 과정을 거치는데, 이로 인해 computing resource가 절반으로 줄게 된다.

그렇다면, 전체 과정을 FP16으로 진행하면 더 좋을텐데, 왜 굳이 FP32의 master weights를 남겨놓는 이유는 무엇일까?

전체 과정을 FP16으로 진행하지 않고 FP32의 master weight를 남겨놓는 이유는 다음과 같다

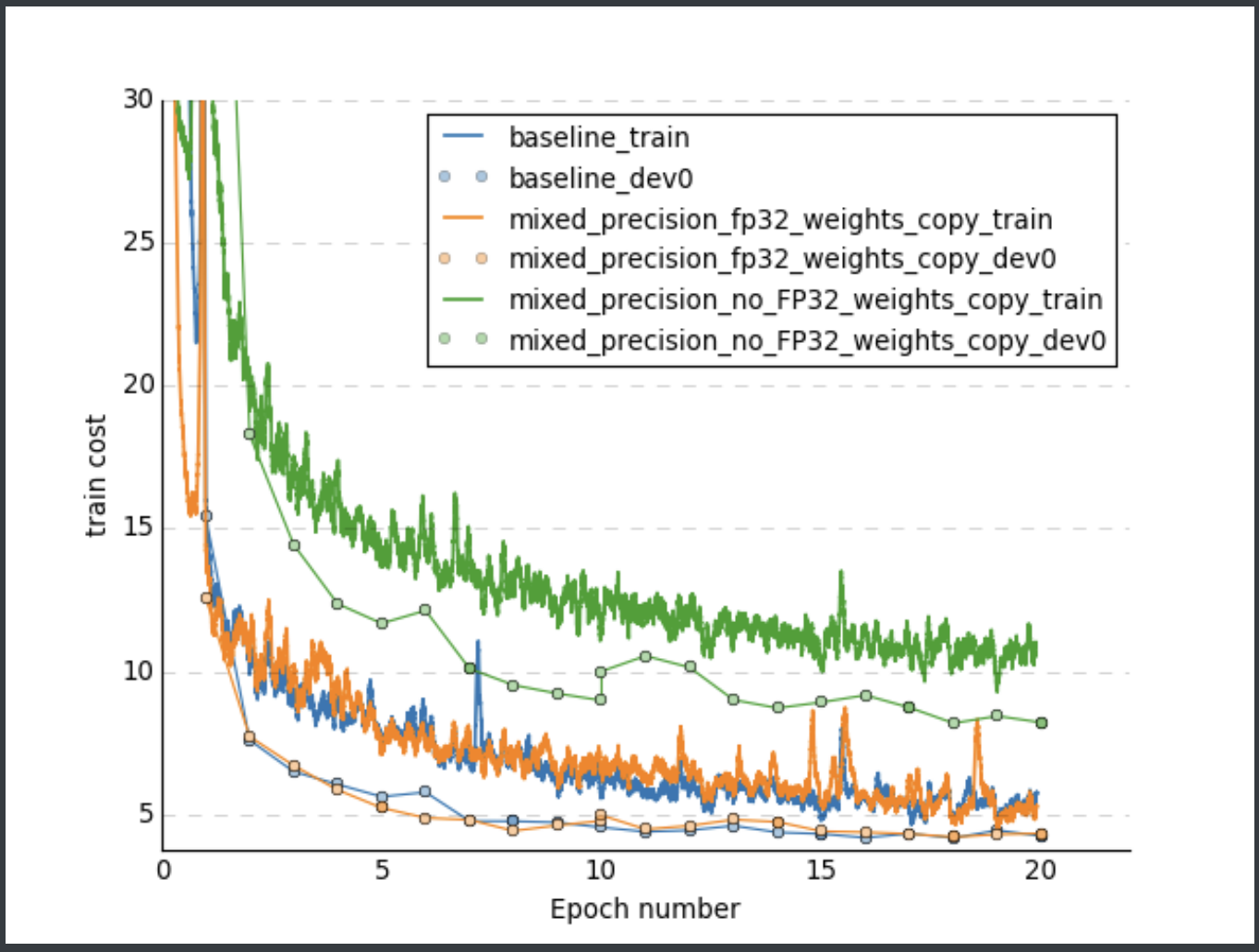
1. **updates(learning rate와 weight gradient를 곱한 값)이 너무 작아서 FP16으로는 표현할 수 없다.**
2. **updates(learning rate와 weight gradient를 곱한 값)이 FP16으로 표현 가능하더라도, 그 뒤에 추가적인 작업에서 weight와 소숫점을 정렬할 때 여전히 0이 되는 문제가 있다.**

$2^{-24}$보다 작은 값은 FP16으로는 표현할 수 없고, 0으로 표현되기 때문에 학습이 정상적으로 진행되지 않고, 이는 모델 성능에 악영향을 미친다. 그래서 updates(learning rate와 weight gradient를 곱한 값)에는 FP32로 표현하는 것이다.



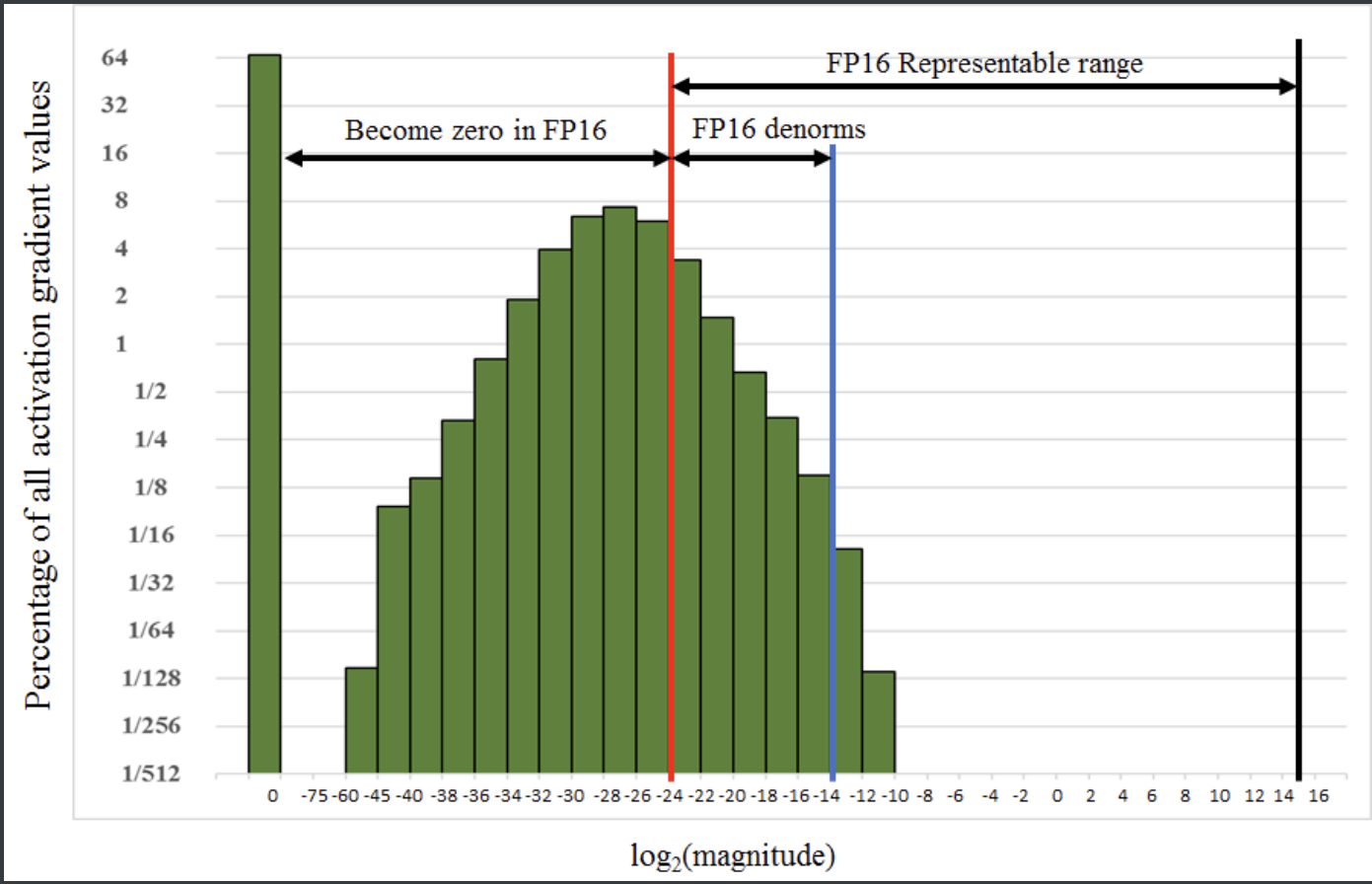

위의 표를 확인하면, 빨간 선 왼쪽 부분의 weight gradient 부분은 FP16의 표현 범위 밖이기 때문에 0으로 표현되어지고, 이는 모델 성능에 악영향을 미치게 된다.

논문에서는 FP32 master copy of weigths의 필요성을 설명하기 위해, 중국어 음성 인식 모델을 기준으로 FP32를 사용한 baseline(파란색)과, 현재 제안하는 방법인 FP16과 FP32를 같이 사용하는 방법(주황색), 마지막으로 FP16만 이용한 방법(초록색) 각각의 training and validation curve를 제시한다. 논문에 따르면, FP16에서 forward와 backward를 거치고 FP32에서 update를 하는 방법은 기존 FP32와 큰 정확도 차이가 없지만, FP16만을 이용한 방법은 상대적으로 80%의 accuracy loss를 초래한다고 한다.




또한, FP16으로 표현되는 weights의 추가적인 사본을 만들고, FP32와 같이 사용하는 것은 weights에 대한 메모리 요구량은 50% 증가하지만, 전체 메모리 요구량은 오히려 줄어들게 된다. deep learning model training에서는 각 layer의 activation이 back-propagation을 위해 저장되는 분량이 메모리 요구량의 상당수를 차지하기 때문이다. 논문에서 제안하는 방법론은 전체 메모리 요구량에서 큰 부분을 차지하는 이 activation 부분을 FP16으로 처리하기 때문에 오히려 전반적인 메모리 요구량은 낮아지게 되는 것이다.

그러나, 이렇게 FP16으로 activation gradient를 처리하는 방법은 문제가 있다. 바로, 대부분의 gradient값이 FP16이 표현할 수 있는 범위를 벗어나버리는 경우에는, 모두 0이 되어버리는 것이다.




만약 위의 도표처럼 대부분의 값이 FP16 범위의 밖으로 나가버리면 어떻게 해야할까? 논문에서는 Loss Scaling 방법을 제안한다.

바로, backpropagation이 시작되기 전에 forward에서 계산된 loss값을 Scaling하는 것이다. Chain-rule에 의해 모든 gradient value가 같은 양만큼 scaling되도록 보장되며, backpropagation 도중에 추가적인 작업을 필요로 하지 않는다. 이렇게 scaling하여 FP16범위로 조정하여 계산한 다음, FP32으로 update되기 전에 다시 unscaling하여 결과적으로 문제가 발생하지 않게끔 해야 한다.

저자들은 이러한 FP32와 FP16을 혼합하여 모델 훈련을 하는 Mixed Precision Training 기법을 다양한 task에 적용해보았고, 실제로 모델에 정확도에 큰 영향을 미치지 않고 성공적으로 메모리 요구량과 훈련 시간을 단축시키는데에 성공하였다. 다만, loss scaling을 하지 않아도 성공적인 학습이 가능했던 task도 있었지만, loss scaling 없이는 학습 성능이 매우 낮아지거나 학습에 실패한 경우가 발생함을 확인할 수

있었다.

보통 deep learning을 공부하는 입장에서는 computing resource가 부족한 경우가 대부분인데, 해당 기법을 이용하면 한정된 환경에서 보다 원활한 학습이 가능할 것으로 생각되어진다. 다만, loss scaling을 염두에 두고 해당 기법을 진행해야 할것으로 보인다.