Incremental Training with Quantized Convolution Backward

May. 31
Jinwook Kim

Introduction

일반적으로 neural network의 training (backward propagation) 과정은 inference (forward propagation) 과정에 비해, computationally expensive 하다. 특히 performance/power가 제한된 mobile device에서는 이미 training된 network에 대해 forward propagation만 수행되곤 한다. 이에 따라, input의 양상이 달라질 때, 적절하게 대응하지 못할 수 있으며, 변화하는 input에 대응하기 위해서는 incremental training (on-line training)이 필요하다. Classification의 class가 늘어나는 것을 가정한 on-line training에 대해서는 기존 연구들이 있으나, 이는 feature에 대한 부분은 그대로 유지하고 마지막 classification layer만 training함으로써, input의 변화에 대응하기에는 부족할 수 있다.

이 문서에서는 변화하는 input에 대응하기 위한 incremental training에 관한 실험 결과와, performance/power가 제한된 mobile device를 위한 convolution layer의 quantized backward propagation에 관해 다룬다.

Method

Incremental training

Train set을 이용한 training과는 달리, incremental (online) training에서는 전체 input을 training할 수 없으며, 현재 input 및 과거 input만 training set으로 활용할 수 있다. 특히, mobile device의 경우, 모든 과거 history를 가져갈 수 없을 것이며, 제한된 history를 통해 제한된 training set만 구성이 가능 할 것이다. 다음과 같은 pseudo-code를 통해 incremental training을 수행 할 수 있다. 실험에서는 현재 input에 대한 정답을 알 수 있는 상황을 가정하였다.

```
def incremental_training(model, loss_function,
                         x: current_input, y: answer_for_x,
                         epochs, optimizer_parameters,
                         num_history, history=[], yhistory=[]):
  history.enqueue(x)
  yhistory.enqueue(y)

  # History가 충분 할 때 (num_history로 주어진 수보다 클 때 training)
  if len(train_set) < num_history:
    return

  # Optimizer는 training시 마다 초기화
  opt = optimizer(optimizer_parameters)

  # History에서 num_history로 주어진 수 만큼 가져와서 training 진행
  train_set = history.dequeue(num_history)
  train_answer = yhistory.dequeue(num_history)

  # 현재 주어진 training set에 대해 주어진 epoch 만큼 training
  for e in range(epochs):
    for batch, batch_answer in batch_loader(train_set, train_answer)
      data_augmentation(batch) # 각 epoch마다, data augmentation 수행
(random crop, flip, ...)
      batch_predict = model(batch)
      loss = loss_function(batch_answer, batch_predict)
      opt.zero_grad()
      loss.backward()
      opt.step()
```

Quantized backward propagation for convolution layer

Convolution layer의 backward propagation 과정은 다음과 같이 backward propagation된 dL/dy로 부터 $dL/dx, dL/dw$를 구하여 진행 할 수 있다.

$$\frac{dL}{dx} = w * \frac{dL}{dy}$$

$$\frac{dL}{dw} = x * \frac{dL}{dy}$$

Quantized convolution layer에서 input x와 weight w는 이미 quantized 된 상태이므로, backward propagation된 dL/dy에 대해서 quantization을 수행하면 전체 과정을 quantization할 수 있다.

Training과정에서는 input x, weight w에 대한 quantization 범위인 α_x, α_w에 대해서도 같이 최적화가 진행 되지만, backward propagation 과정에서는 dL/dy에 대한 quantization 범위 α_{dy}를 adaptive하게 결정하기가 어렵다. 실험에서는 전파 된 dL/dy에 대한 평균/최댓값과 같은 통계치를 구하여 수행하였다. Pseudo-code는 아래와 같다.

```
def conv_backward(x, w, dy, clf = lambda x: x.abs().max(), by):
  a_dy = clf(dy)
  dyq = quantization(dy, a_dy, by) # dy를 범위 a_dy, bitwidth by에 대해 quantization

  dx = conv(w, dy)
  dw = conv(x, dy)
  return dx, dw

```

Experiements

Conditions

실험은 PyTorch를 이용해 구현 되었으며, 실험에 사용된 model 및 image set은 아래와 같다.

1. Train/Test set: CIFAR10 Image Set (Training 50,000, Test 10,000, Class 10)
2. ORG model: CIFAR10 training set으로 pre-trained model
    1. Optimizer and parameters: SGD with learning rate 0.04, momentum 0.9, weight decay 1e-4
    2. 4bit Quantized convolution layer
    3. Epochs: 300
    4. CIFAR10 testset 정확도: 93.01%
    5. ORG (E): Evaluation mode 상태에서의 ORG model
    6. ORG (T): Traininig mode 상태에서의 ORG model

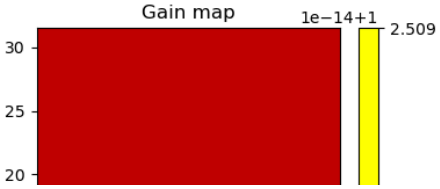
ORG (E)/ORG (T)는 model이 evaluation상태, training 상태에 있을 때를 나타내며, 각각 평가에 사용 되었다. ORG model에 포함된 Batch-norm layer의 경우, PyTorch 구현상에서는 Backward-propagation을 수행하지 않는다 하더라도 Eval/Train mode에 따라 batch-norm layer의 동작이 달라지게 되어 결과에 영향을 주었다. 따라서, Eval/Training 상태를 각각 개별적으로 평가 하였다.

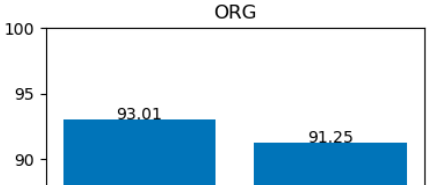
실험에서는 CIFAR10 test-set image를 변형하여, 변형된 image에 대해 incremental training 접근이 정확도 향상에 얼마나 기여하는지를 측정하였다.

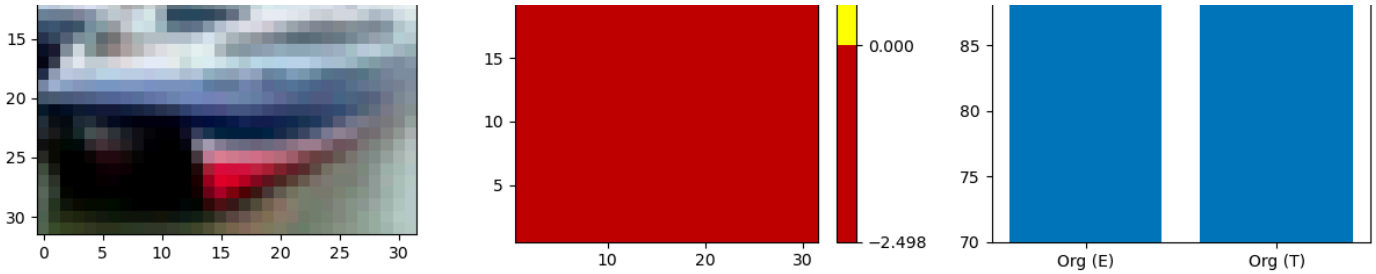
Image 변형 및 parameter 선정

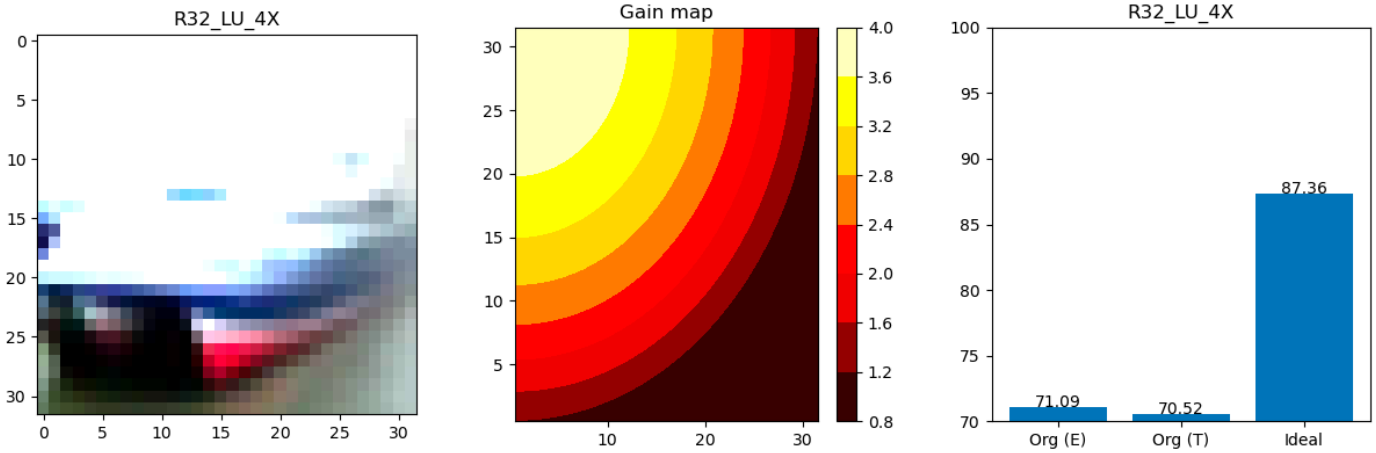
Image 변형은 밝은 광원 (spot light) 혹은 어두운 부분 (dark spot)이 포함되어 Image의 밝기가 균일하지 않은 상황을 가정하였다. 아래 그림과 같은 상황을 가정하여, CIFAR10 test-set을 변형하여 원 model의 성능을 평가하였다. Ideal은 CIFAR10 Training-set (50,000 images)에 대해서도 같은 형태로 변형하여 새로 training 시켰을 때의 결과이다.

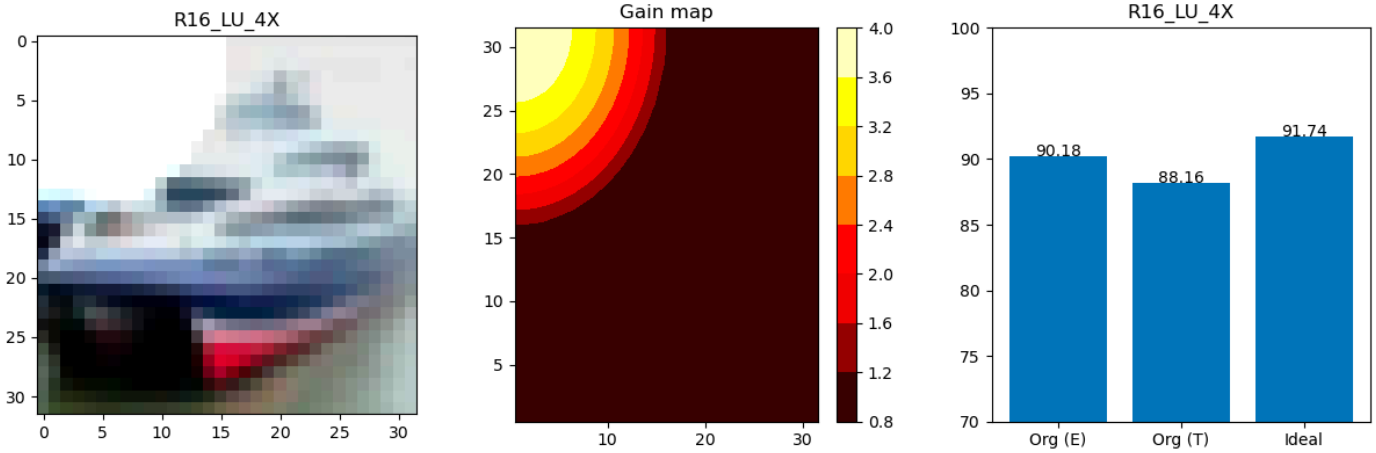
실험에서는 ORG model의 정확도를 고려해 R32_LU_4X (반지름 32px, 좌상단 4배 Gain)형태의 transformer를 사용하였다.

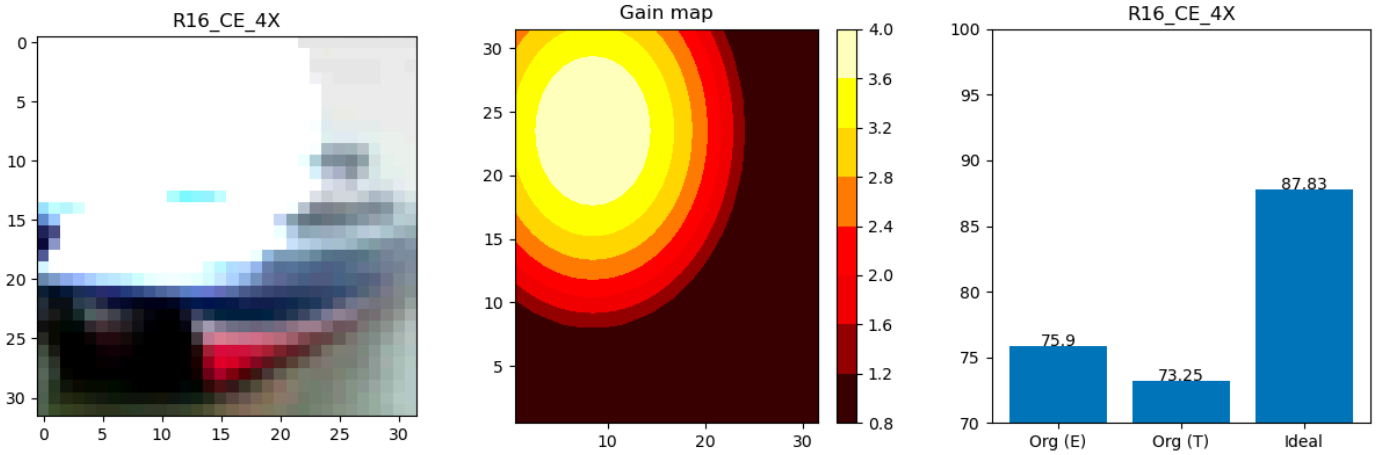


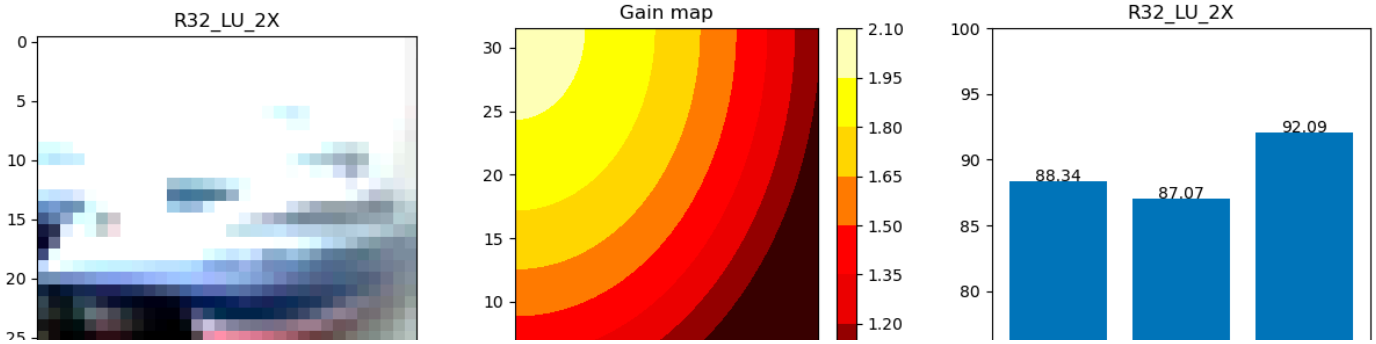


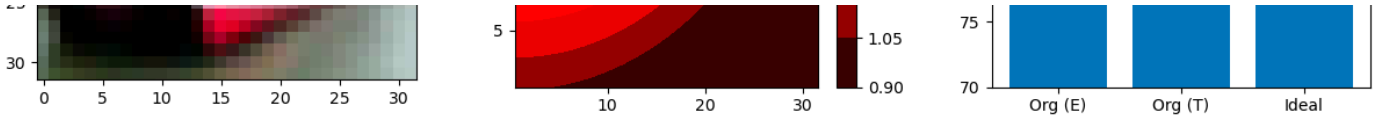

0.000
−2.498
Org (E)
Org (T)


R32_LU_4X
Gain map
R32_LU_4X
87.36
71.09
70.52
Org (E)
Org (T)
Ideal

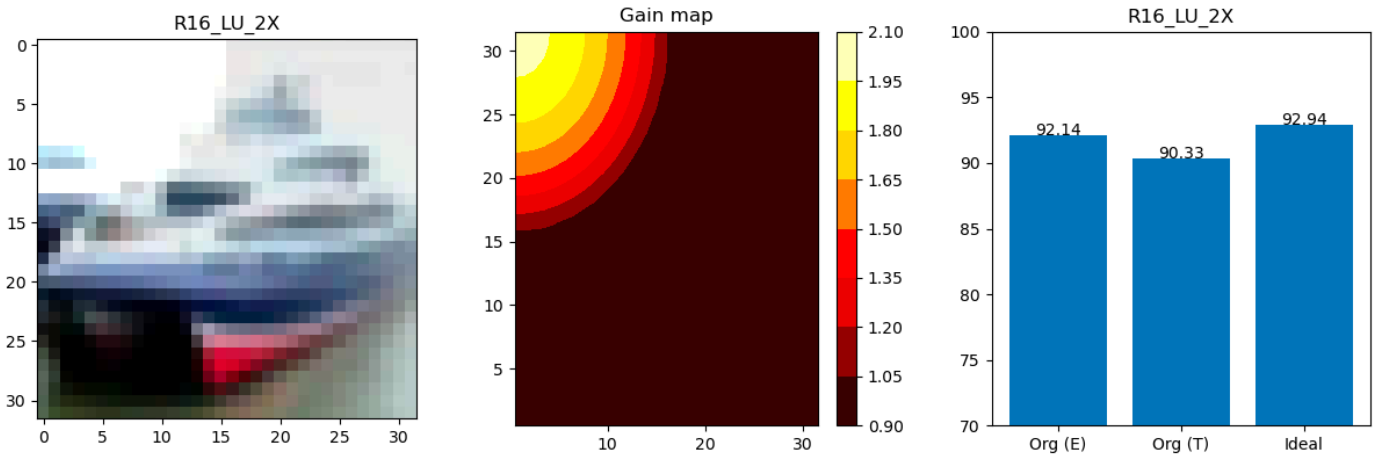

R16_LU_4X
Gain map
R16_LU_4X
90.18
88.16
91.74
Org (E)
Org (T)
Ideal


R16_CE_4X
Gain map
R16_CE_4X
87.83
75.9
73.25
Org (E)
Org (T)
Ideal

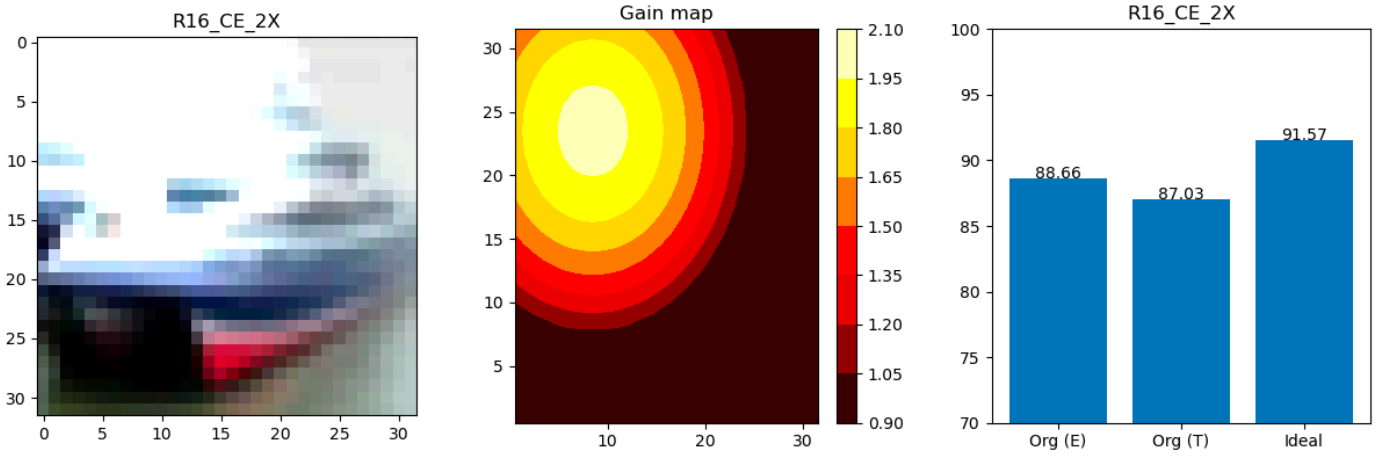

R32_LU_2X
Gain map
R32_LU_2X
92.09
88.34
87.07



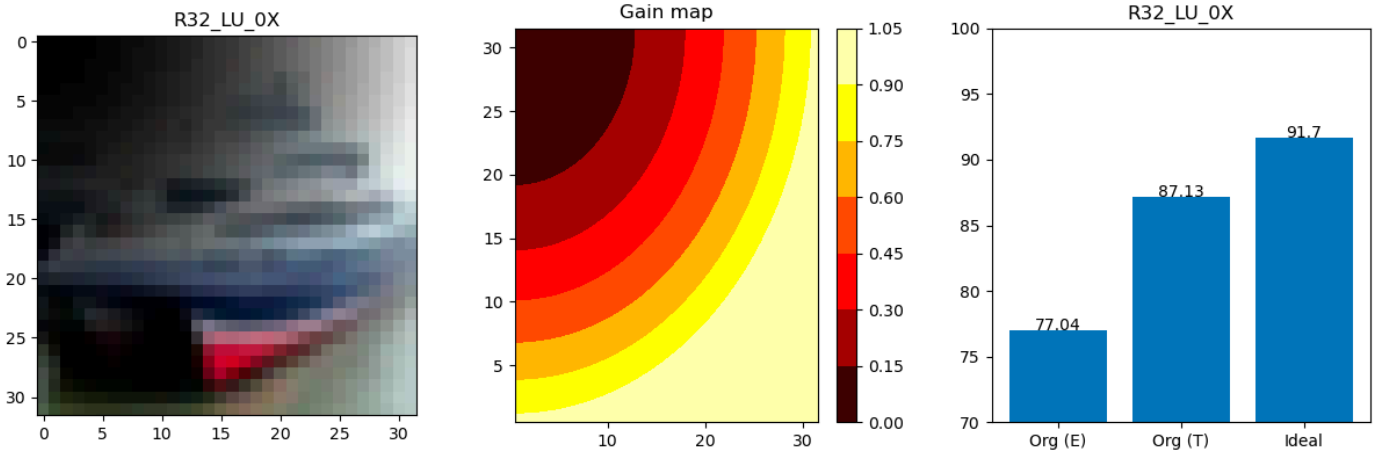
R16_LU_2X



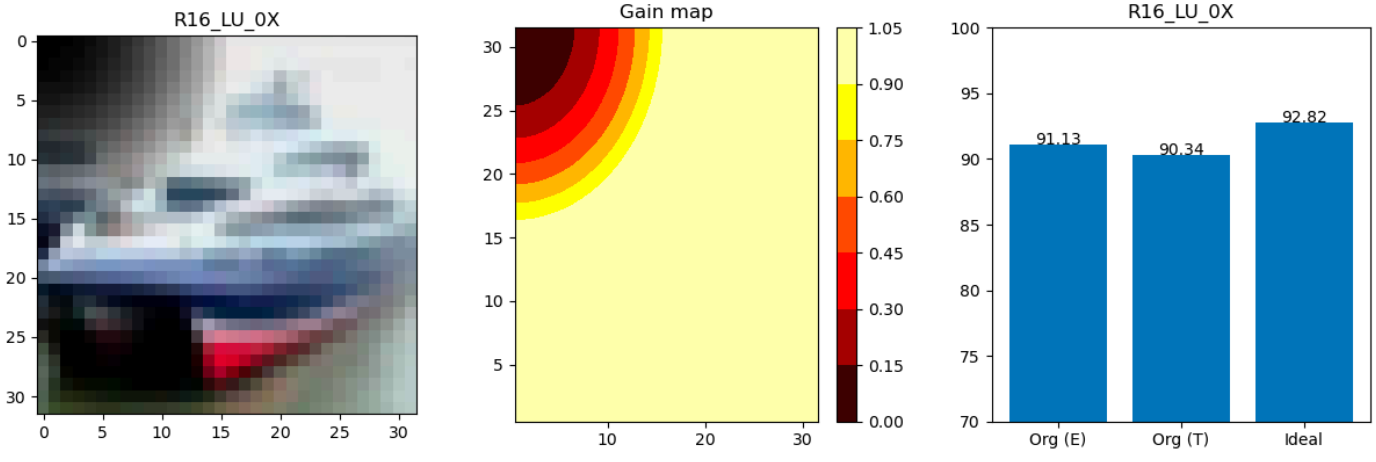
R16_CE_2X



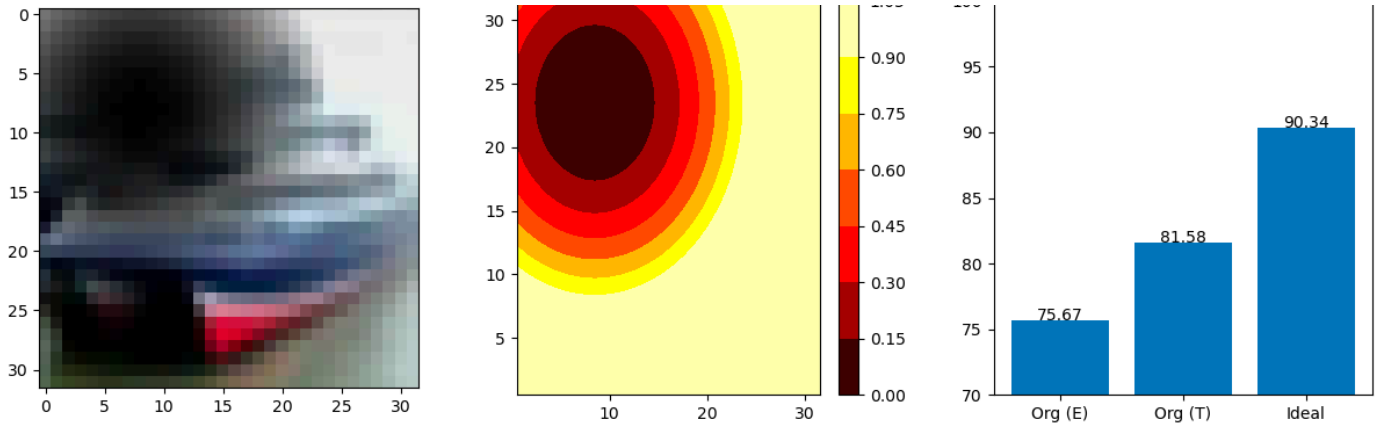
R32_LU_0X



R16_LU_0X



R16_CE_0X



Incremental Training

Incremental training에 대한 실험은 아래 pseudo-code와 같은 형태로 정확도 측정 및 진행 되었다.

```
def validate_incremental(testset, model, ...):
  xhistory = []
  yhistory = []
  precision = ...

  for i, (x, y) in enumerate(testset):
    model.eval()
    yt = model(x)
    prec = accuracy(yt, y) # 현재 batch의 accuracy 측정
    precision.update(prec)

    incremental_training(loss, model, x, y, xhistory, yhistory, ...)

  return precision

```

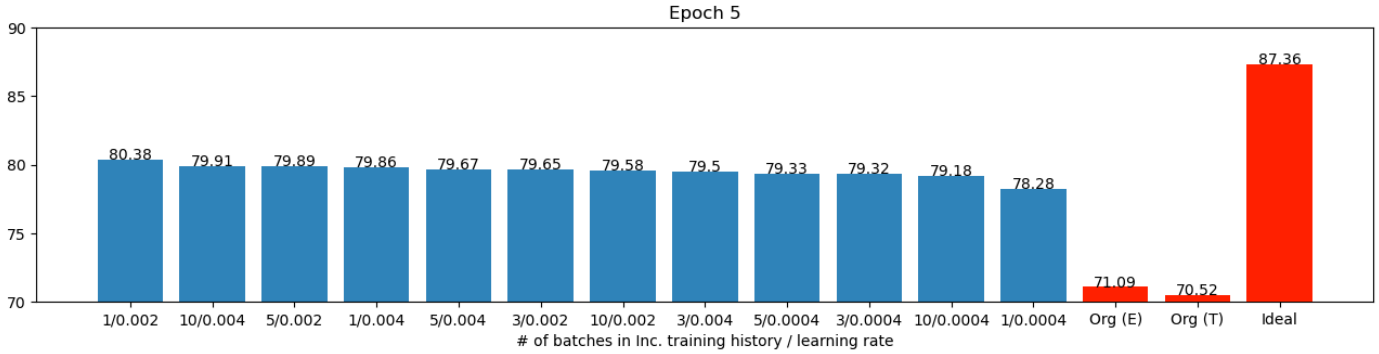
Validation 과정에서는 현재 input에 대한 정확도 측정 후, incremental training을 진행한다. 즉, 정확도 측정에서는 현재 input에 대한 training은 반영되지 않으며, 이전 input history에 대한 training만 반영되어 변경된 model이 사용된다.

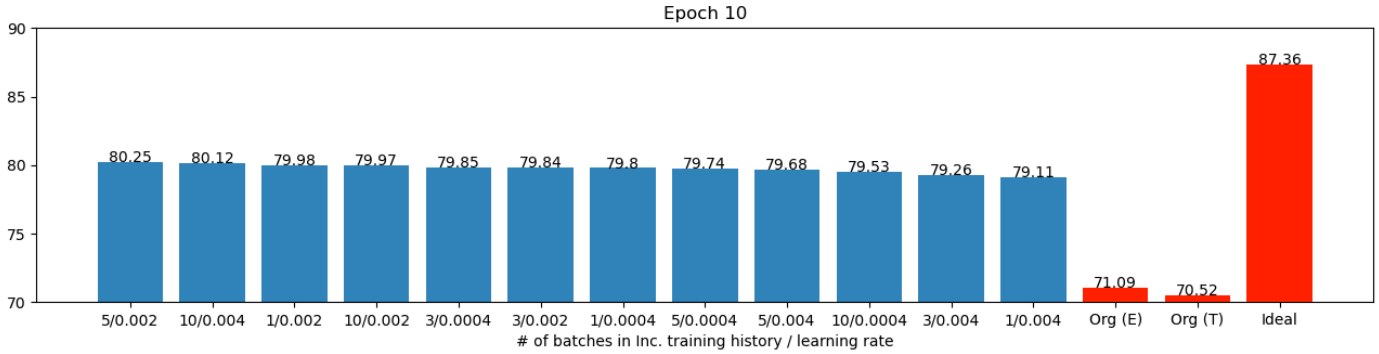
Parameter에 따른 Accuracy

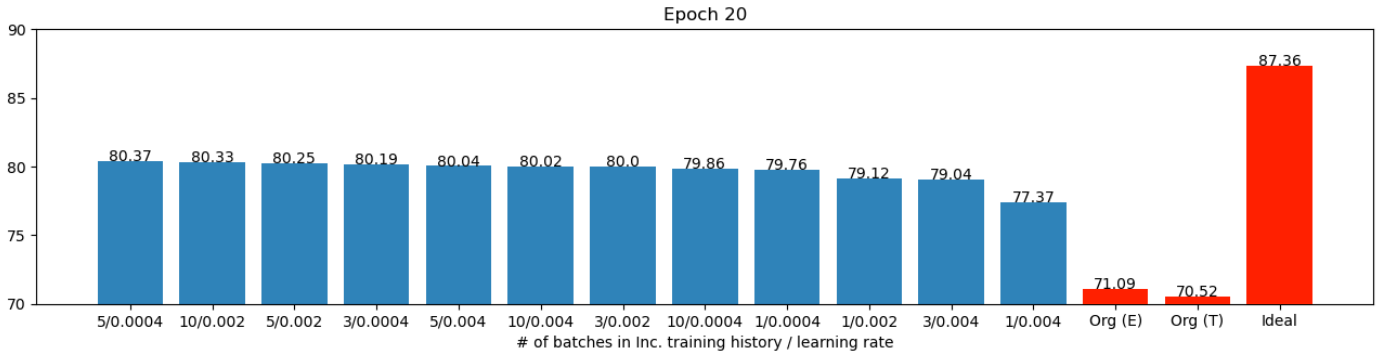
Learning parameter및 history의 크기, epoch에 대한 incremental training의 실험 결과이다. 조건은 다음과 같다.

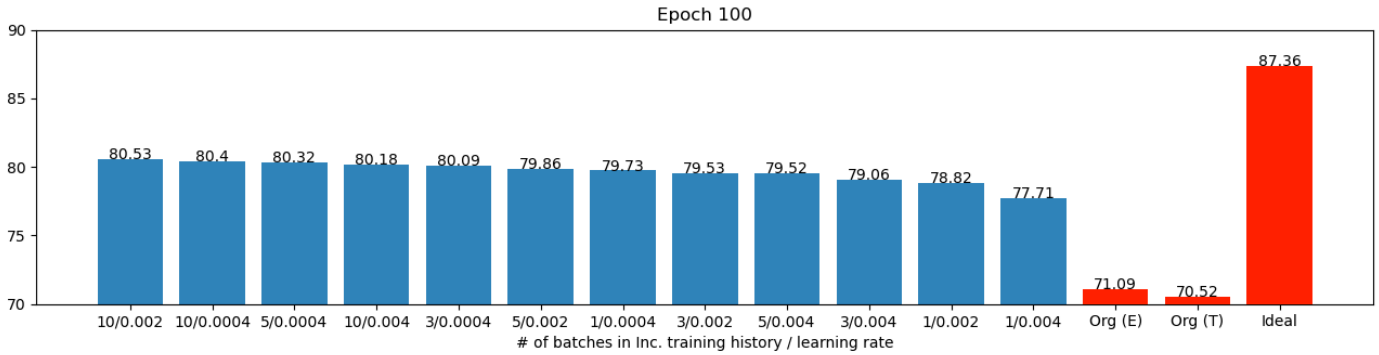
1. Testset: R32_LU_4X로 변형된 CIFAR10 testset
2. Epoch: 5, 10, 20, 100, 200

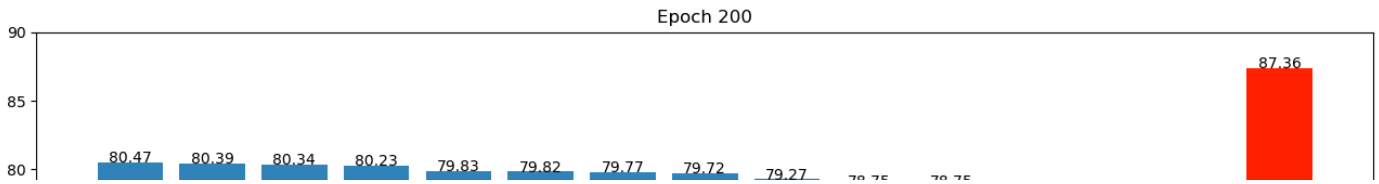
3. Learning rate: 4e-3, 2e-3, 4e-4
4. History: 1, 3, 5, 10 batches (128 per batch) → 128, 384, 640, 1280 images

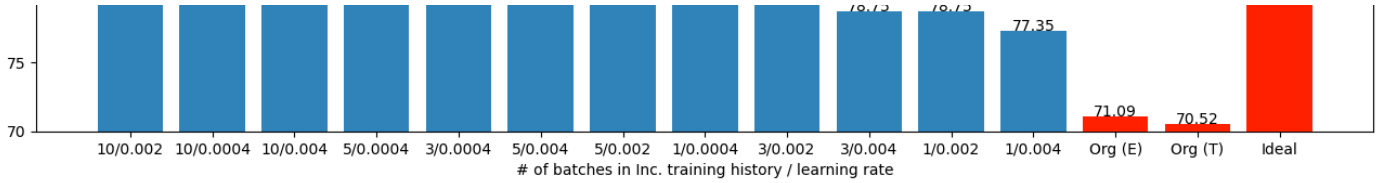












실험상에서는 incremental training을 수행함에 따라, 기존 ORG model대비 약 9% 가량의 정확도 향상을 얻을 수 있었다. Epoch, history, learning rate에 따라 조금씩은 차이를 보이나, 대체로 80% 근방의 결과를 얻을 수 있었다. 정확도는 대략 77~80% 가량을 얻을 수 있었으며, Epoch이 늘어난다고 크게 달라지지는 않았다. Ideal은 50k의 training set을 이용한 반면, incremental training은 10K의 testset을 evaluation하면서 training된 결과이므로 정확도 차이가 training scheme에 따른 것인지, 혹은 data의 수에 따른 것인지 확인 할 필요가 있다.

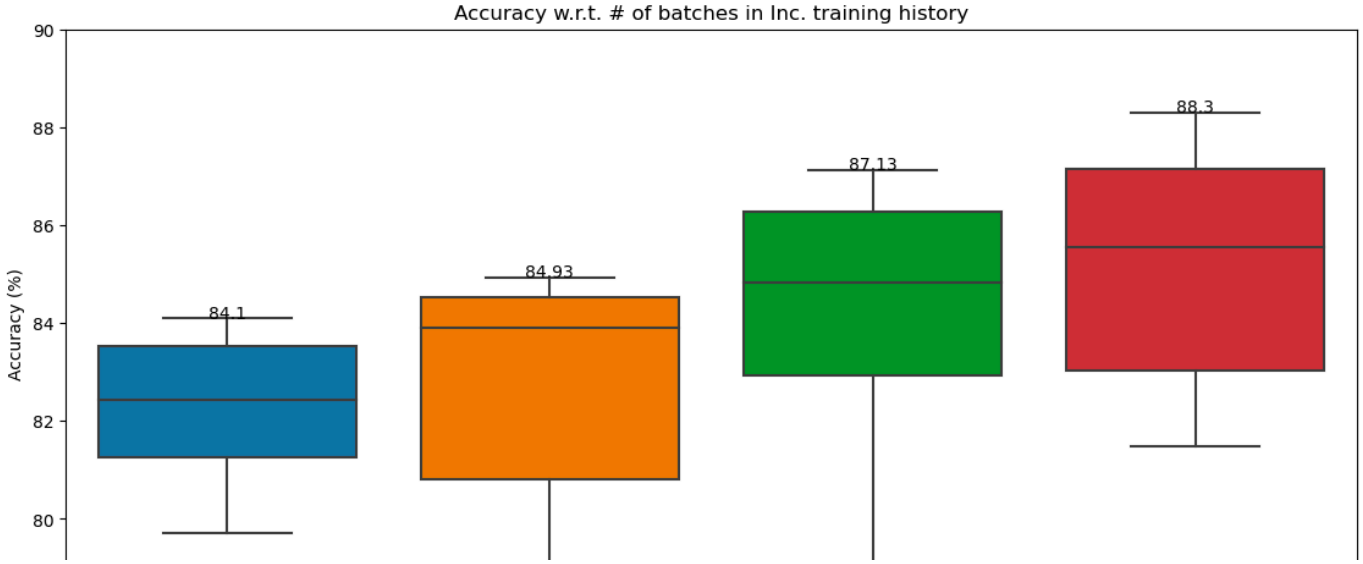
Data/History에 따른 경향성

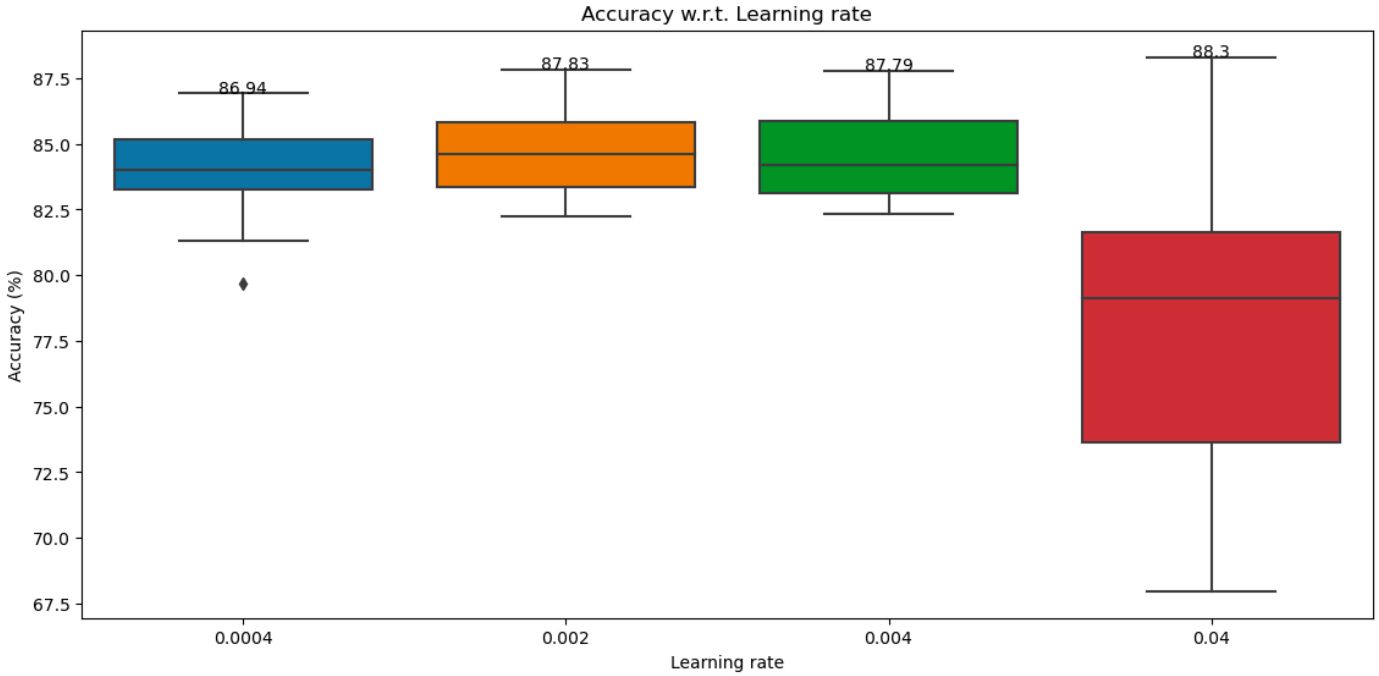
앞선 실험에서 incremental training은 원 model에 비해 정확도 향상을 얻긴 했지만, 50k training set을 변형하여 생성한 Ideal model에 비해서는 7% 가량 뒤지는 결과를 보여주었다. 위 실험에서 incremental training은 10K testset에 대해서만 수행을 하였으며, training시 제한된 수의 data만 대상으로 진행하였다. 10K data가 아닌, 충분한 data가 존재할 경우와, history의 수를 늘릴 때 정확도를 향상 시킬 수 있는지 확인하고자 한다.

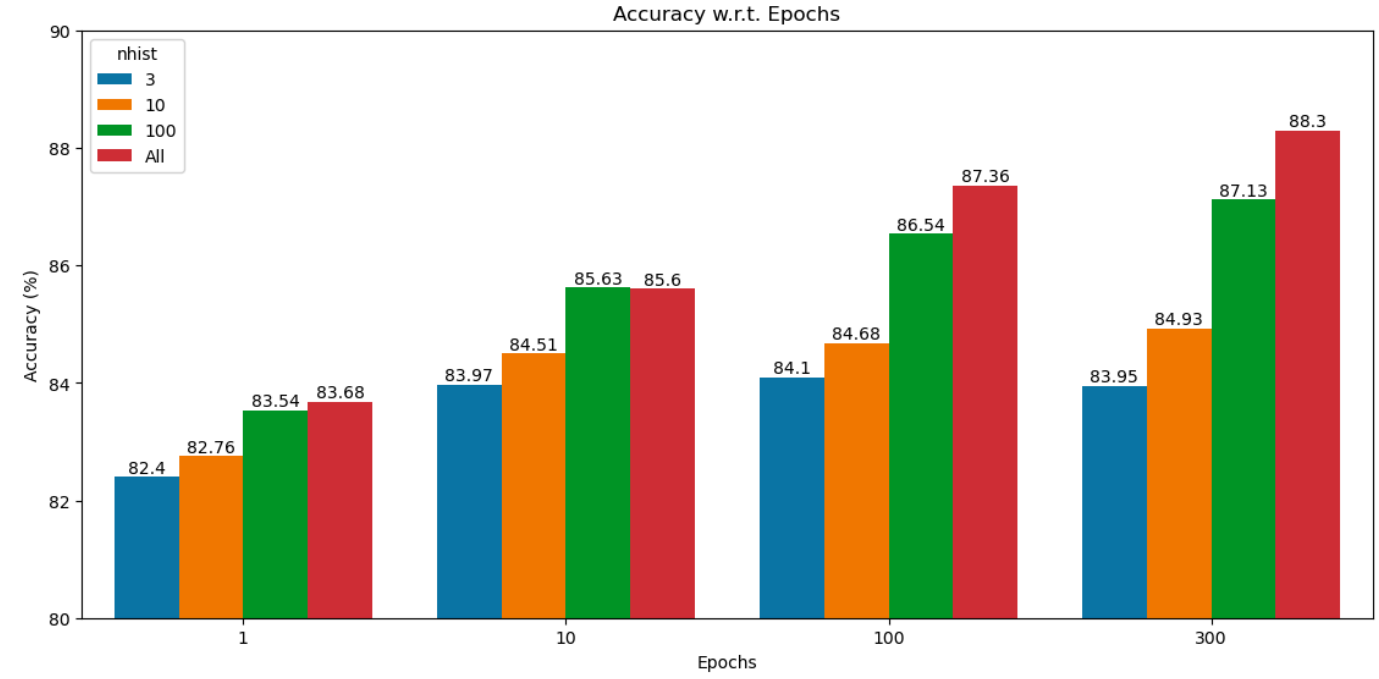
실험 설계

1. Org model (Pre-trained with CIFAR10 train-set)
2. 50K CIFAR10 train-set (128 per batch, overall 391 batches) 을 R32_LU_4X transformer로 변형하여,
3. Incremental training scheme으로 재 training후, 정확도 측정

실험 결과







첫 번째 그래프의 실험 결과를 보면, history의 크기 (한번에 바라볼 수 있는 data의 크기)에 따라 정확도가 향상 되는 것을 확인 할 수 있다. 세 번째 그래프의 결과를 보면, history의 크기가 작을 때, epoch에 따른 정확도 수렴이 일찍 찾아오는 것을 볼 수 있다. History가 작다는 것은 training시 보이는 data set이 작다는 점이며, 현재 주어진 data에 대한 local optimum에 머무르는 것으로 추정해 볼 수 있다.

결론적으로, incremental training에서 정확도 향상을 위해서는 충분한 data가 주어지는 것도 중요하지만, training시 사용되는 history의 크기가 중요할 것으로 생각할 수 있다. 다만, mobile device에서는 history를 충분히 가지기 어려울 것이며, history가 쌓이기 전에는 model update가 발생하지 않으므로, 적절한 parameter setting이 중요할 것이다.

Quantization results

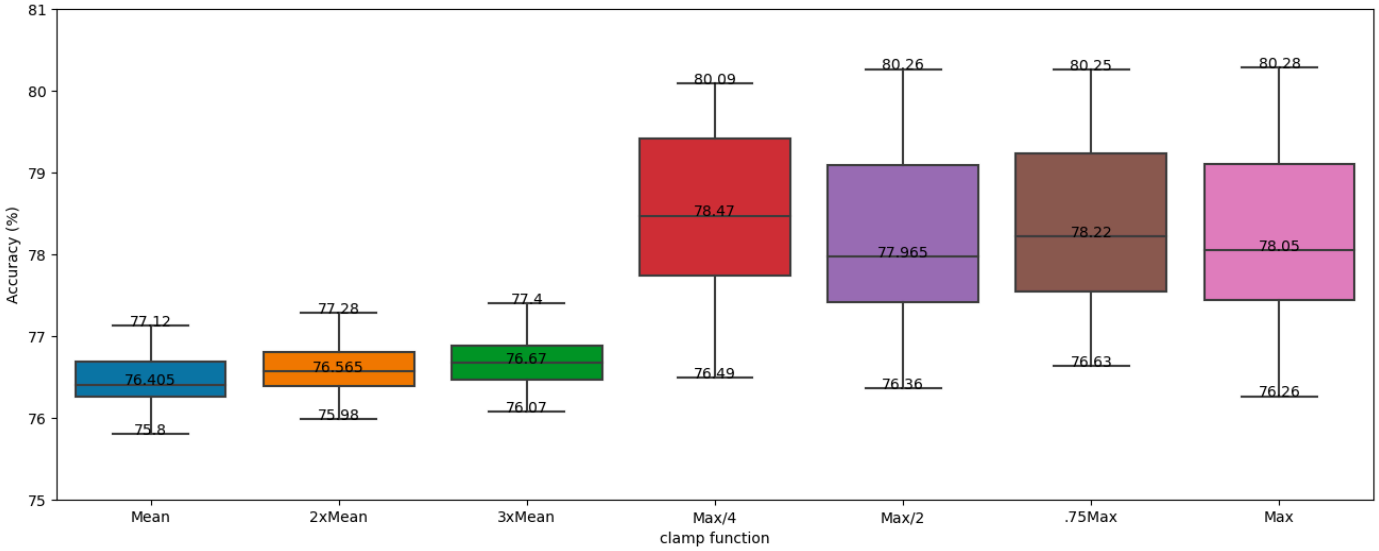
Convolution layer의 quantization 실험 결과는 incremental training 실험과 동일한 조건에서 convolution layer의 backward propagation만 수정하여 수행 하였다. 즉, CIFAR10 50K training set으로 training한 Org model은 원래대로 backward propagation수행 되었으며, incremental training 과정에서 convolution layer에 대한 backward propagation만 quantization 하여 수행하였다.

Optimizer parameter 및 history 크기는 위 incremental training 실험 결과에서 각 epoch별 가장 좋은 결과를 주었던 parameter를 선택하여 수행하였다. Quantization에서의 실험 parameter는 아래와 같다.

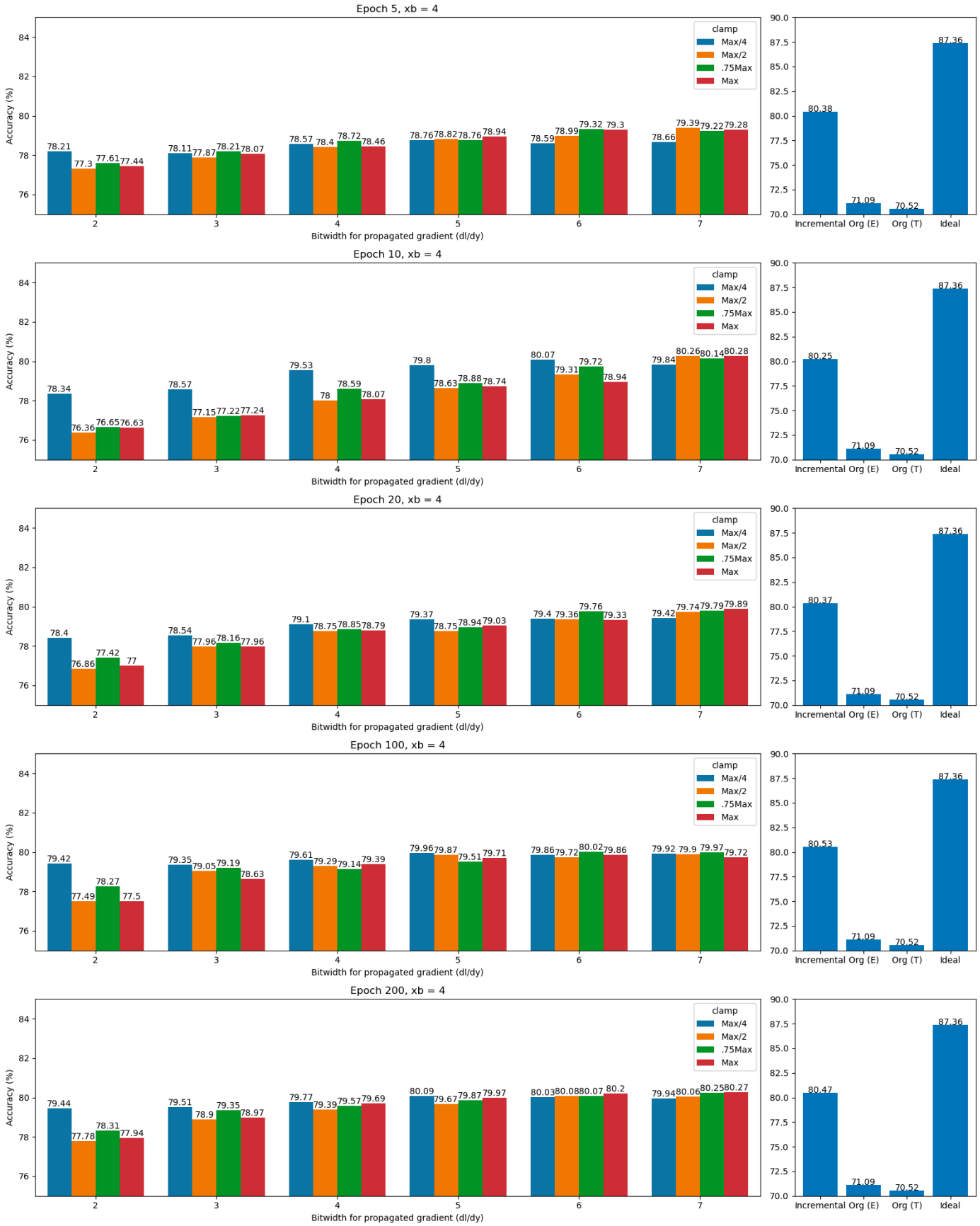
1. dL/dy bitwidth - 2, 3, 4, 5, 6, and 7bit
2. input x, convolution weight w bitwidth - 2, 3, 4 bit
    1. Org model 4bit에 대해서, backward propagation 시 단순 shift operation으로 bit truncation
3. dL/dy 의 quantization range
    1. dL/dy.abs().mean(), $2 \times dL/dy$.abs().mean(), $3 \times dL/dy$.abs().mean()
    2. dL/dy.abs().max() $/4$, dL/dy.abs().max() $/2$, dL/dy.abs().max() $\times 3/4$, dL/dy.abs().max()

Quantization range

Quantization range에 대한 실험 결과는 아래와 같다.

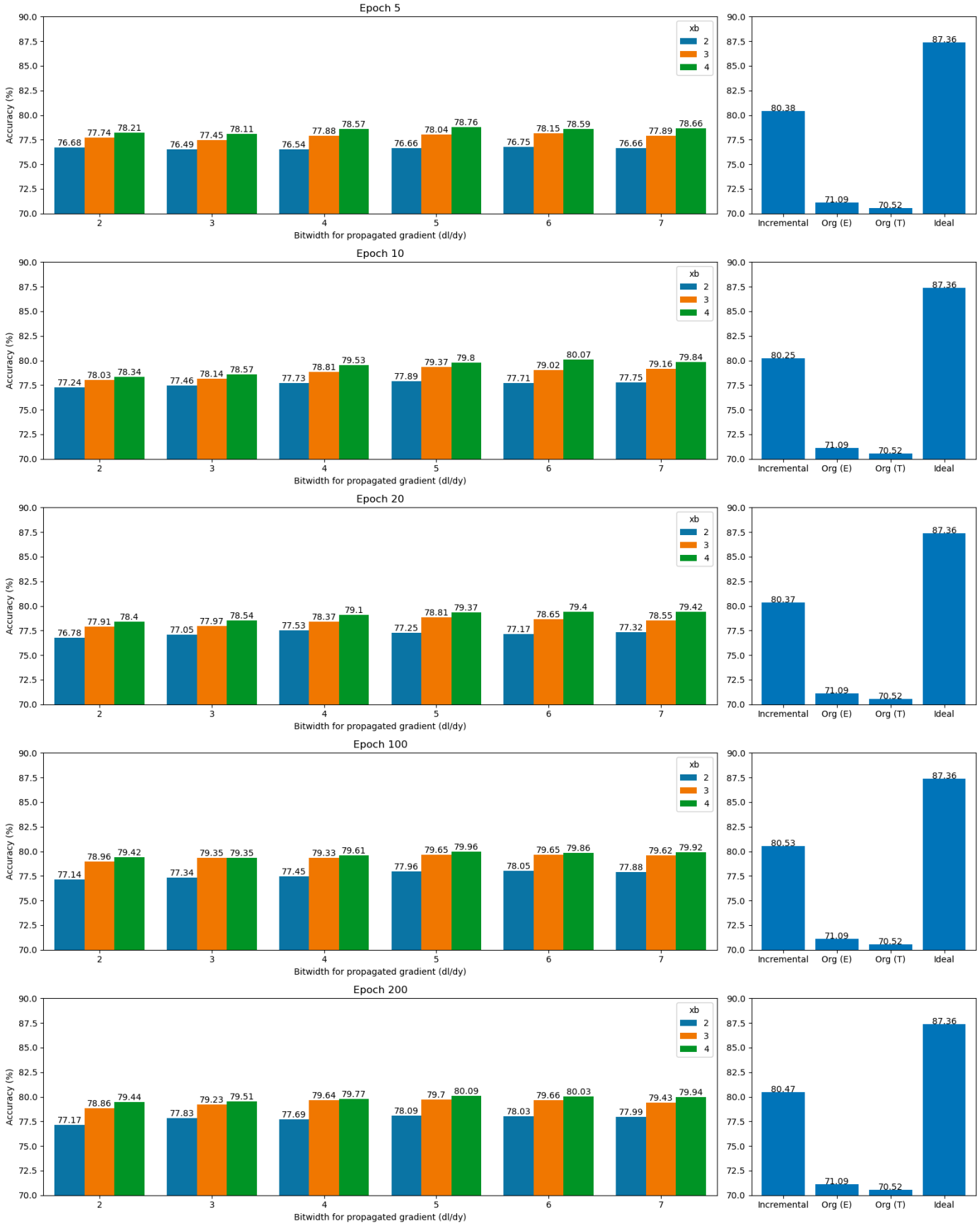


Mean을 기준으로 range를 잡은 것에 비해서, Max기준으로 range를 잡은 경우가 전체적으로 좋은 결과를 보여주었으며, Max기반은 bitwidth등에 따라서 결과가 달라졌다. Max 기반 quantization 범위 계수에 대한 결과는 아래와 같다.



위 결과는 input x, weight w에 대한 quantization bitwidth를 Org model과 같은 4bit으로 고정했을 때의 결과이다. 대체적으로 max/4가 우세하나, 당연하게도 dL/dy bitwidth가 증가함에 따라 좀더 범위를 넓힌 Max/2, Max등이 좀더 좋은 결과를 보였다. 그러나, 그 차이가 크지는 않았다.

Quantization bitwidth



위 실험은 input x, weight w의 bitwidth의 변화를 포함한 결과이다. 이때 quantization 범위 계수를 구하는 함수는 Max/4를 사용하였다. 그림에서 xb는 x, w의 bitwidth를 나타낸다. 실험 결과로는 정확도는 xb에 큰 영향을 받았으며, dL/dy의 bitwidth에는 크게 영향을 받지 않았다. 원래의 Org model bitwidth가 4임을 감안할 때, xb를 낮춰서 얻는 이득은 크지 않을 것으로 보이며, dL/dy역시 같은 수준으로 bitwidth를 정해도 무방할 것으로 보인다.

전체적인 결과로는 quantization하기 전 (Incremental)과 결과를 비교하였을 때, 4bit quantization 기준 크게는 약 1.8% 수준, 작게는 0.8% 수준의 정확도감소를 보였으나, Org model의 결과보다는 크게 상승한 수치로, Quantized backward propagation역시 의미있는 접근이 될 수 있음을 보였다.

Discussion

본 문서에서는 mobile device등 제한된 computational resource를 가진 end-user device에서 neural network의 update 방안인 incremental training 방법과 convolution layer의 quantized backward propagation 방법 및 그에 대한 실험 결과를 다루었다.

Incremental training의 경우, input이 변화된 환경에서 적은 수의 data set 과 5~10 내외의 epoch 에 대한 training만으로도 Org model 대비 약 8~9% 가량의 성능향상을 얻을 수 있었다. 비록 training 시 바라보는 data set의 크기가 제한됨에 따라 local optimum point에 빠지는 경향을 보여 정확도 향상에 한계를 보였으나, computational resource가 제한되는 상황에서는 이 역시도 의미를 가질 수 있다고 생각 된다. 적절한 history의 크기와 epoch은 application에 따라 다를 수 있겠으나, CIFAR10 대상으로한 실험에서는 상대적으로 적은 epoch과 history만으로도 Org model 대비 상당한 정확도 향상을 얻을 수 있었다.

Convolution layer의 quantized backward propagation은 floating point를 이용한 기존의 backward propagation대비 약 1~2% 의 성능 하락 만으로도 bitwidth를 제한할 수 있었다. Quantization range에 대해서는 backward propagation된 dL/dy의 간단한 통계 (Maximum of absolute value)를 활용하여 결정할 수 있었으며, layer별로 특정 상수를 지정하는 부분에 대해서도 필요하다면 추가적으로 시도 해 볼 수 있을 것이다.