시각지능 A – Object Detection

모델 경량화

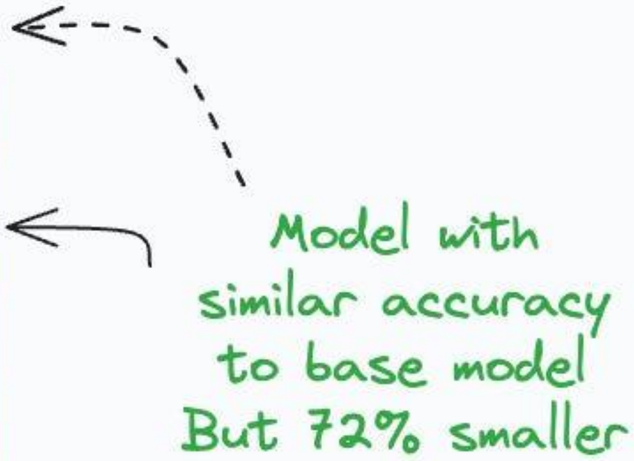
목차 모델 경량화

1. 모델 경량화

모델 경량화

- 모델 경량화(Model compression)
  - 성능을 유지하거나 약간의 성능 저하를 감수하며 모델의 크기와 계산량을 줄이는 방법
  - 작은 기기에 모델을 싣거나 실시간 응용 프로그램을 만들 때 필수
    - On-device AI, 임베디드 시스템 등 모바일 기기
    - 응답 속도, 추론 속도 등 UX 개선

	Threshold	Test Accuracy	Size Reduction
Original Model	-	96.69	-
Reduced Model	0.1	96.62	62%
	0.2	96.41	66%
	0.3	96.31	69%
	0.4	96.08	72%
	0.5	94.88	74%
	0.6	94.49	76%
	0.7	92.19	79%
	0.8	90.65	81%
	0.9	88.39	83%
	1	87.87	84%



https://blog.dailydoseofds.com/p/model-compression-an-overlooked-technique

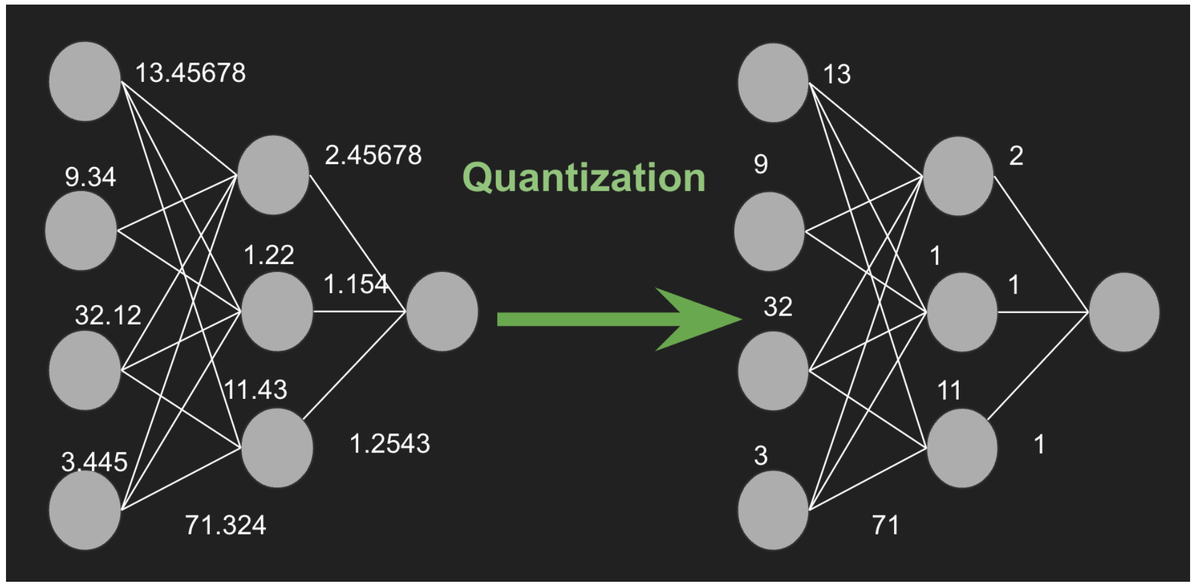
모델 경량화

- 발생 가능한 문제점
  - 일반화 성능의 저하
  - 각 경량화 기법 특징에 따른 고유 이슈(하드웨어 최적화, 정밀도 손실 등)
  - 일부 하드웨어에 사용 불가능한 경량화 기법 존재
  - 모델 단순화 및 근거 우회로 인한 해석 가능성 감소

2. Quantization(양자화)

Quantization

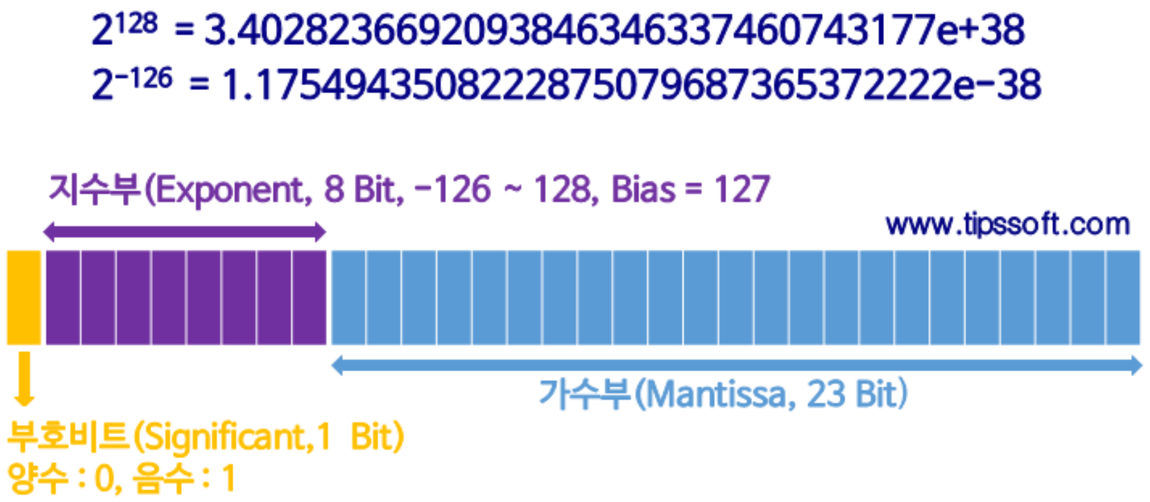
- Quantization(양자화)
  - 모델의 가중치(Weights)나 활성화 값(Activations)을 더 낮은 비트 수로 표현하여 모델의 크기를 줄이고 계산 효율성을 높이는 기법
    - 원래 이미지, 오디오 등이 양자화 대상이었음
    - Weights: 모델 파라미터
    - Activations: wx + b(Feature map, Hidden states, Nodes, QKV)



https://medium.com/@curiositydeck/neural-network-quantization-03ddf6ad6a4f
https://blog.paperspace.com/quantization/

Quantization

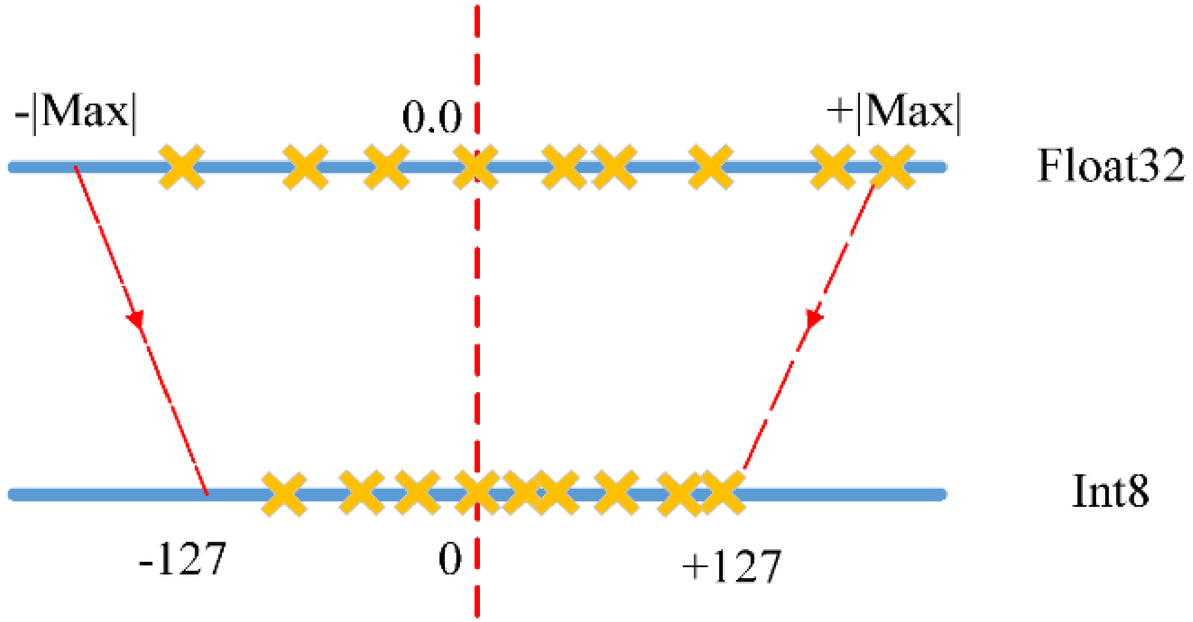
- Bit(BInary digiT)
  - 정보의 가장 작은 단위(0, 1)
  - 모든 컴퓨터의 명령과 데이터를 구성하는 원소
  - 여러 개의 비트를 사용하면 더 넓은 범위의 수를 표현할 수 있음
    - 1bit: 0 or 1
    - 4bit: 0000 to 1111
    - 8bit: 00000000 to 11111111
  - 일반적으로 딥러닝 연산 시 float32 사용
    - 16bit 사용 시 메모리 절반 사용
    - 8bot 사용 시 메모리 ¼ 사용



http://www.tipssoft.com/bulletin/board.php?bo_table=FAQ&wr_id=177

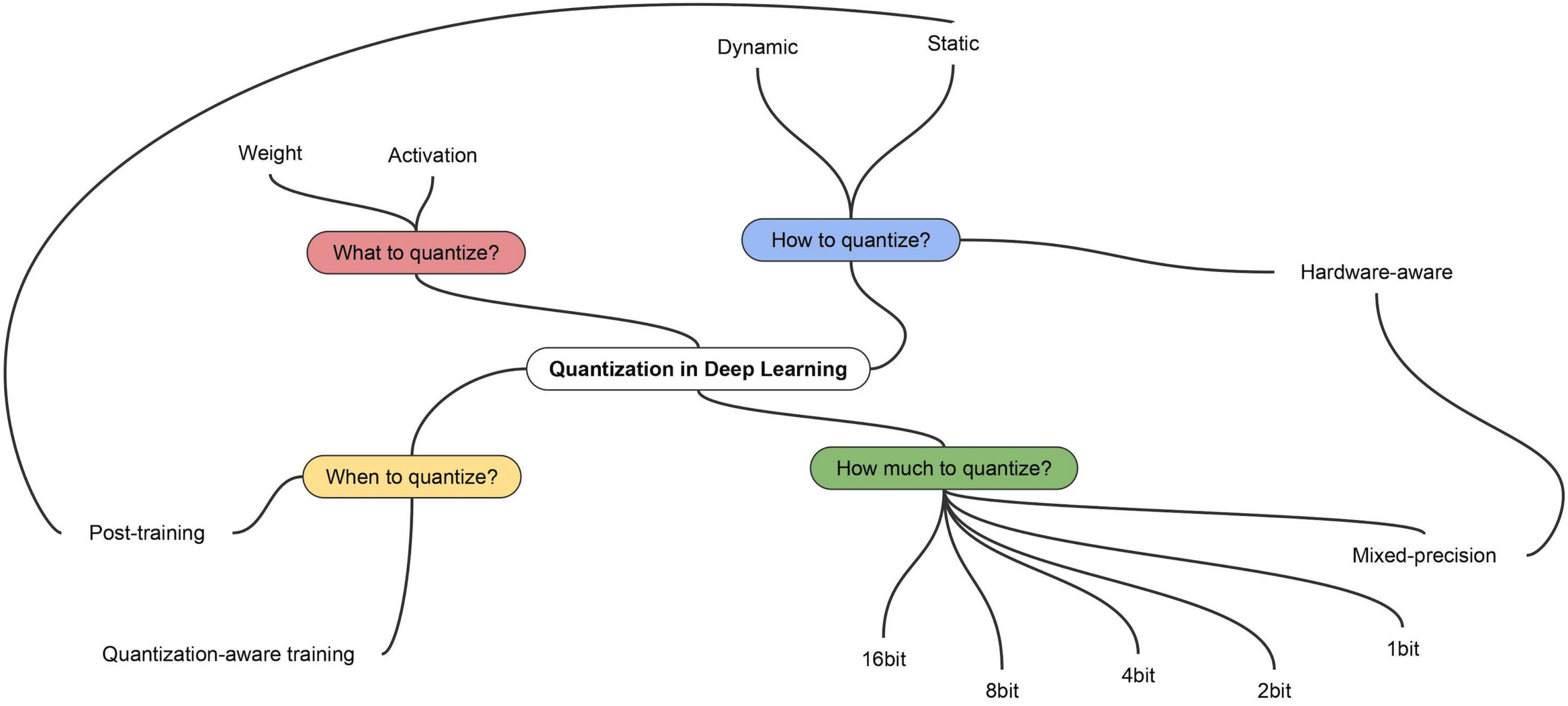
Quantization

- Quantization으로 인한 데이터 범위 변화
  - Float32 -> Int 8 등 용량이 작은 데이터로 대체
  - 모델의 가중치, 모델의 내부 연산 결과 등이 이에 포함됨
  - 데이터의 표현 범위가 바뀌므로 Scaling이 필요



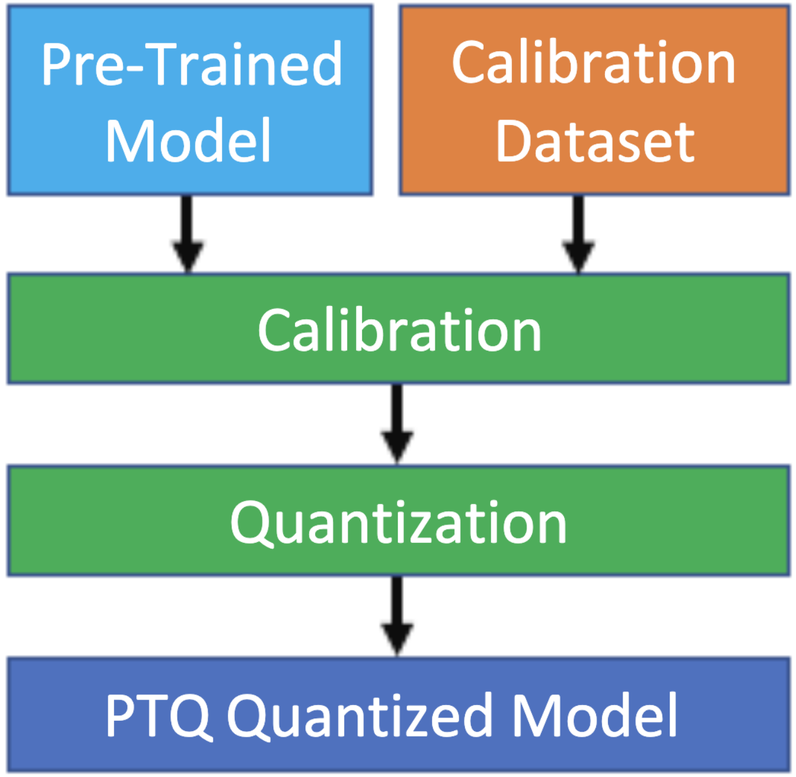
https://ga02-ailab.tistory.com/146

Quantization 방법



PTQ & QAT

- PTQ(Post-Training Quantization)
  - 모델 훈련(또는 Pretraining)을 완료한 후에 적용되는 기법
    - 이미 만들어진/사전 학습된 모델 대상
  - 모델의 가중치, Layer에서 출력되는 activation이 변환 대상
  - 모델 출력 값의 scale을 조정하는 과정(Calibration)이 매우 중요

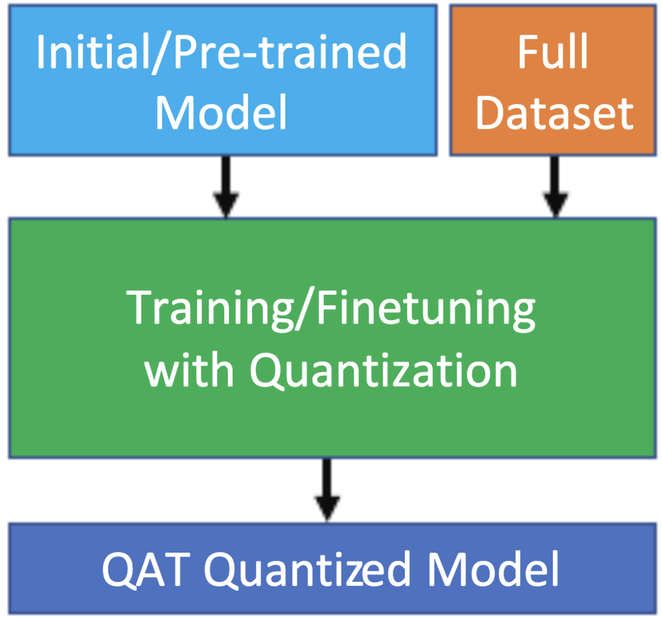


PTQ

O-2A: Ourlier-Aware Compression for 8-bit Post-Training Quantization Model, IEEE Access 2023

PTQ & QAT

- QAT(Quantization-Aware Training)
  - 모델에 양자화를 적용하여 새로 훈련하며, 그 과정에서 발생하는 가중치와 activation을 조정하는 방법
  - Calibration을 통한 Scale 조정이 필요 없지만, 재학습에 소요되는 시간, 리소스를 고려해야 함
  - 양자화를 고려하여 학습시키므로, 성능 하락이 매우 적음
  - 모델이 크거나, 정확도가 중요하다면 적용하기 유리



QAT

O-2A: Ourlier-Aware Compression for 8-bit Post-Training Quantization Model, IEEE Access 2023

Quantization 주의사항

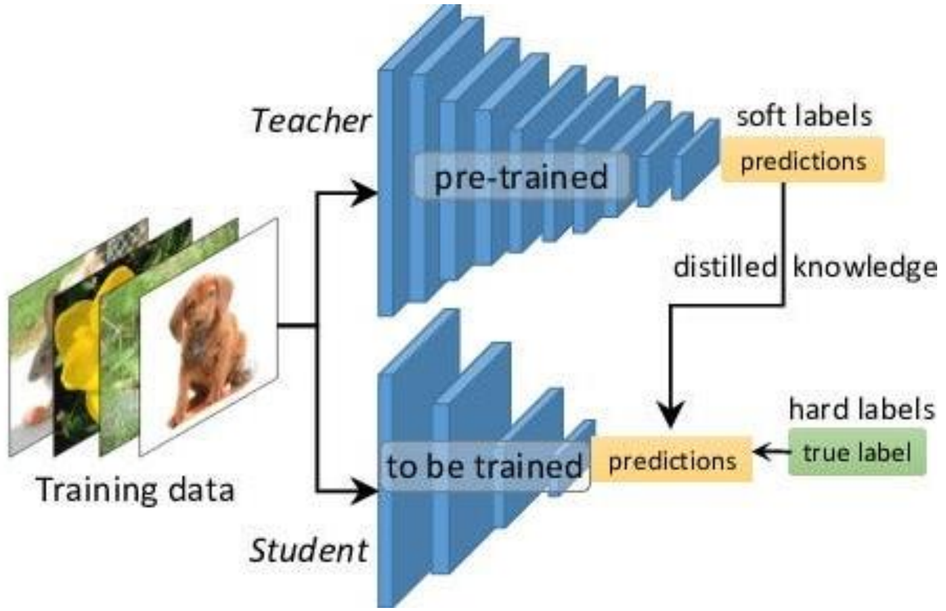
- 모든 Layer가 Quantize될 순 없음
  - Layer 구조가 복잡한 경우
  - 모델이 여러 프레임워크/라이브러리에 의해 결합된 복잡한 구조일 경우
- 모든 Layer을 Quantize할 필요도 없음
  - 경우에 따라 Quantization 과정을 거쳐도 성능만 하락할 뿐 속도가 개선되지 않을 수 있음
- Quantization은 학습 속도를 개선하기 위함이 아닌, Inference 속도를 개선하기 위함
  - 학습에서는 Loss 계산 – Backprop에서 많은 시간을 소모하므로 크게 도움이 되지는 않음

https://gaussian37.github.io/dl-concept-quantization/

3. Knowledge Distillation(지식 증류)

Knowledge Distillation

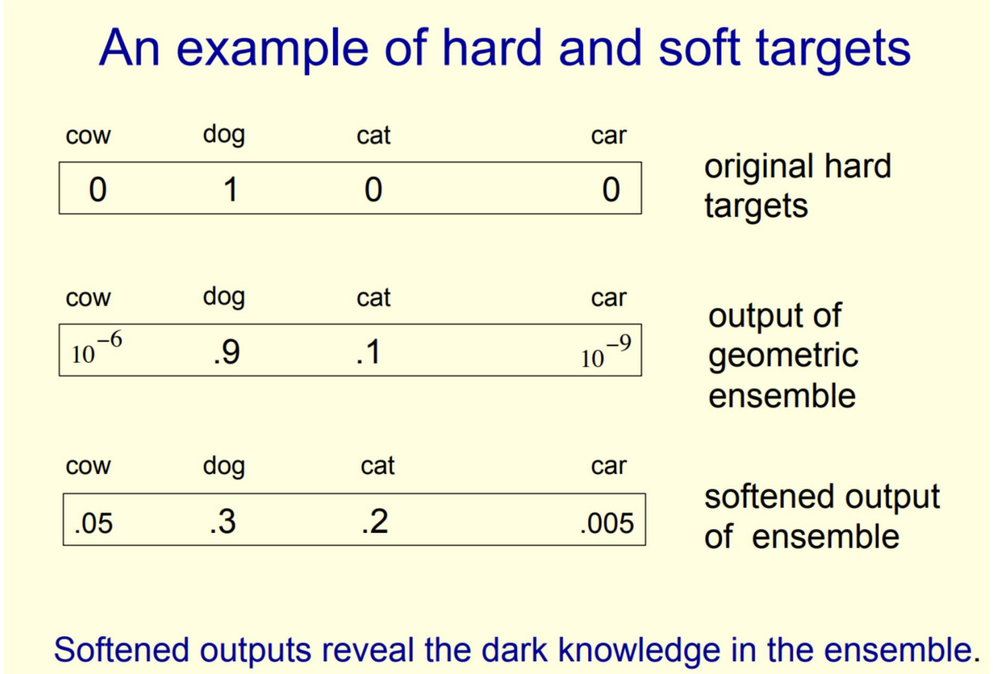
- Knowledge Distillation(KD, 지식 증류)
  - 복잡하고 큰 모델이 학습한 지식을 작은 모델에 전달하는 기술
  - 작은 모델의 성능을 유지하거나 향상시키며 계산 효율성을 높이기 위한 방법
- Teacher
  - Parameter가 크며 복잡한 모델
  - 큰 데이터를 통해 잘 학습되었으나, 계산 비용이 큼
- Student
  - Teacher보다 작으며, 구조를 모사한 모델
  - Teacher가 학습한 정보를 전수받아 저렴하게 계산을 수행



https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

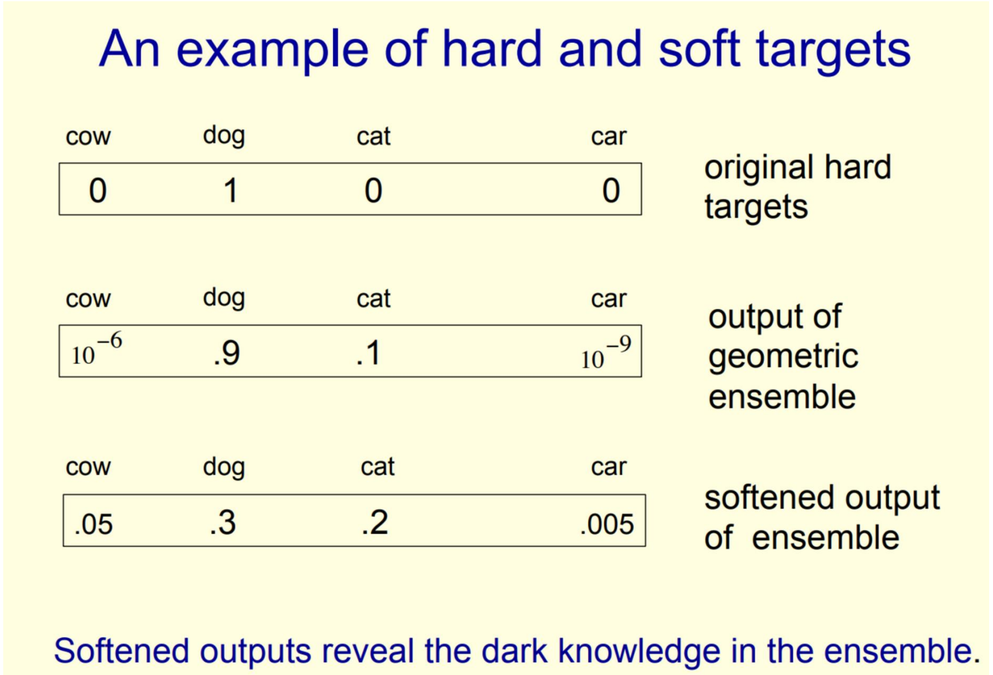
Knowledge Distillation

- KD 원리
  - Student 모델이 Teacher 모델에서 출력된 Soft label을 바탕으로 학습
  - Soft label에는 Teacher 모델이 학습한 지식의 정수(Dark knowledge)가 들어있음
- Hard labels v.s. Soft labels
  - Hard labels
    - 일반적으로 학습할 때 사용하는 데이터셋의 정답 라벨
    - One-hot vector(Sparse vector)
  - Soft labels
    - Teacher 모델이 제공하는 예측 확률 분포
    - Classifier와 Softmax를 통과한 값



Knowledge Distillation

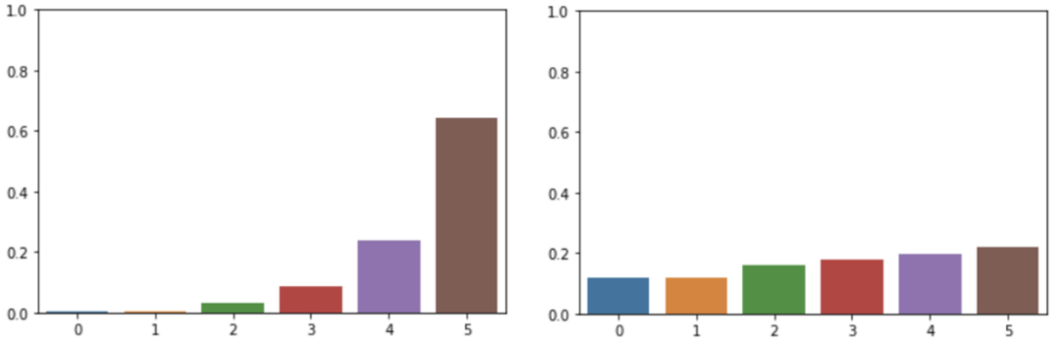
- Why Soft labels?
  - 확률 분포에는 정답 클래스 뿐만 아니라 다른 클래스와의 관계나 불확실성까지 포함
  - Teacher 모델이 학습한 데이터의 패턴이나 클래스 간 관계를 잘 표현
  - Soft label을 통해 정답과 오답에 대한 Teacher 모델의 Generalization 능력이 전달
  - 일반화 능력을 향상시키기 위한 기존의 방법
    - Ensemble
    - Dropout
    - Regularization(L1, L2)



Knowledge Distillation

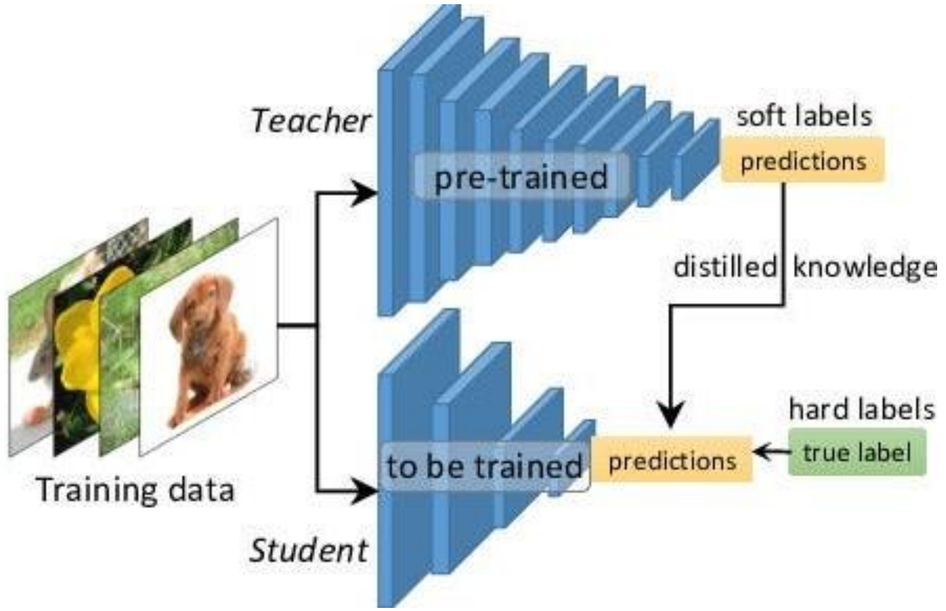
- Softer softmax
  - Teacher 모델에서 soft label을 더 잘 뽑아내기 위하여 softer softmax
  - 일반화 성능을 높이기 위하여(학습의 난이도를 높이기 위하여)
  - T(Temperature)
    - Teacher 모델이 생성한 확률 분포를 부드럽게 만드는 계수
    - 2~4 사이 값

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$



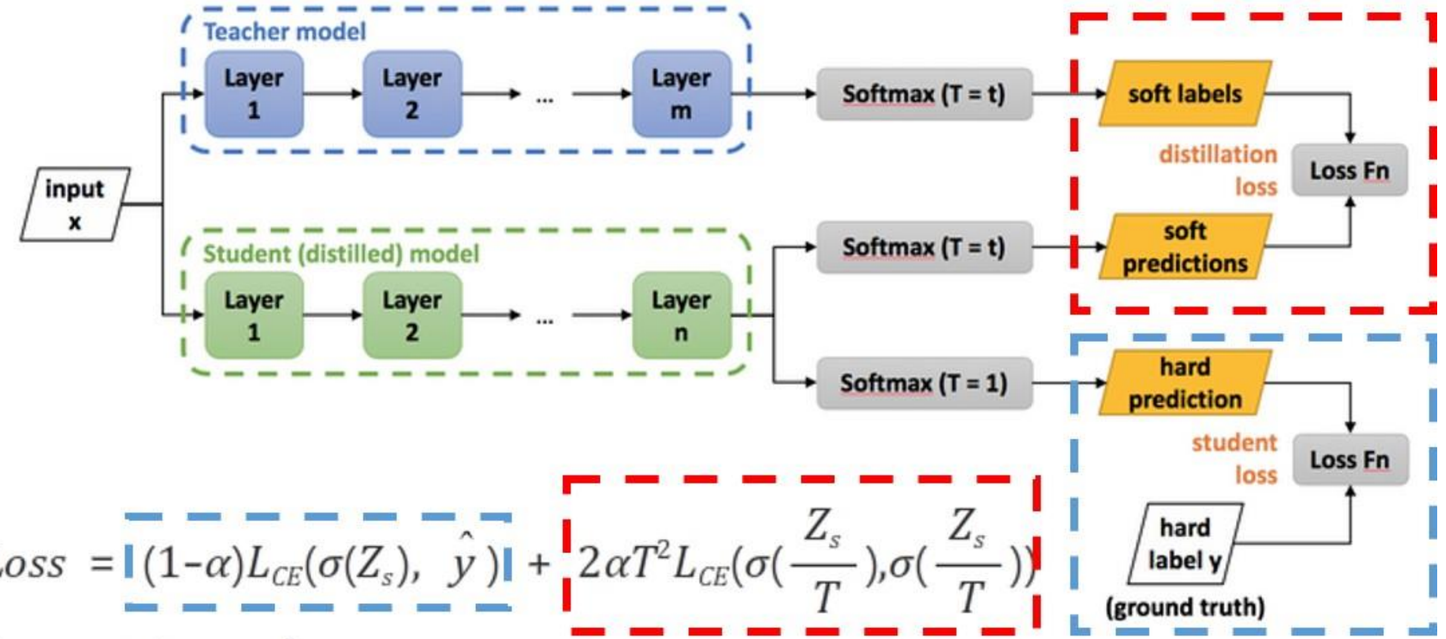
Knowledge Distillation

- 학습 과정
  - Teacher 모델을 학습
  - Teacher 모델의 Prediction 결과를 Soft labels로 변환
  - Student 모델로 두 가지 학습을 진행
    - Student 모델의 Prediction과 Soft labels 사이의 손실 계산
    - Student 모델의 Prediction과 Hard labels 사이의 손실 계산
  - 두 손실 값을 합산하여 모델 Backpropagation 및 최적화



Knowledge Distillation

- 손실 계산과 최적화



$$Total\ Loss = (1-\alpha)L_{CE}(\sigma(Z_s),\ \hat{y}) + 2\alpha T^2 L_{CE}(\sigma(\frac{Z_s}{T}),\sigma(\frac{Z_s}{T}))$$

$L_{CE}()$: *Cross entropy loss*

$\sigma()$: *Softmax*

Z_s: *Output logits of Student network*

Z_t: *Output logits of Teacher network*

$\hat{y}$: *Ground truth(one-hot)*

α: *Balancing parameter*

T: *Temperature hyperparameter*

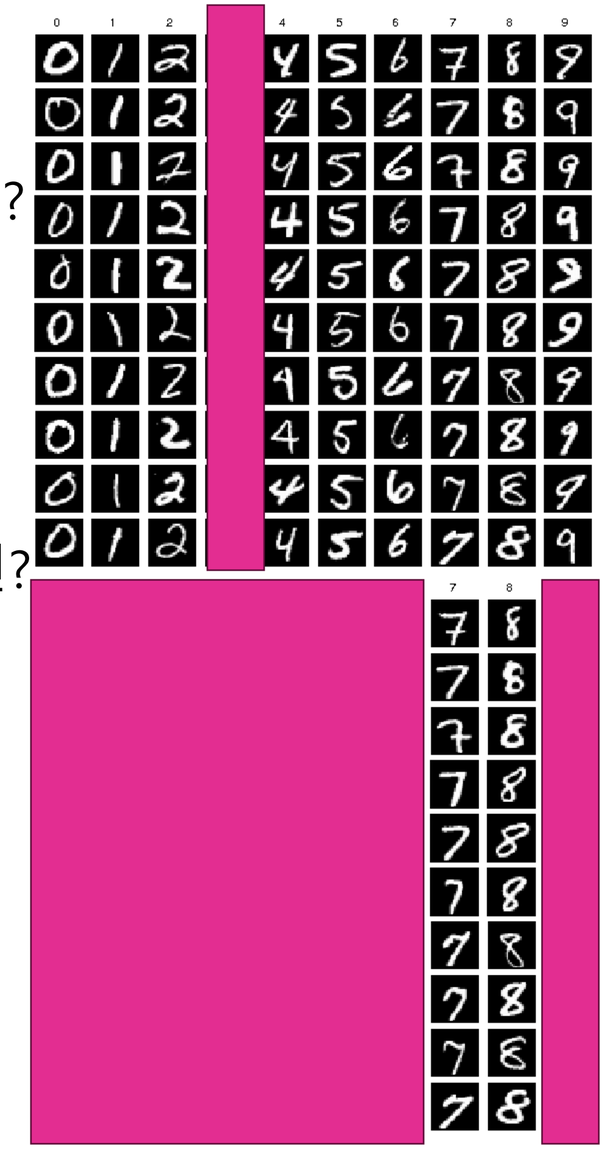
https://junsk1016.github.io/deeplearning/Knowledge-Distillation/

Knowledge Distillation

- MNIST 실험
  - Vanilla(대조군)
    - 784 → 800 → 800 → 10(ReLu)
    - Test error = 146
  - Teacher network
    - 784 → 1200 → 1200 → 10
    - Dropout, jittering
    - Test error = 67
  - KD(실험군)
    - Teacher model: Enhanced vanilla
    - Student model: 784 → 800 → 800 → 10
    - Test error = 74

Knowledge Distillation

- 추가로 알게 된 사실
  - 만약 MNIST 데이터 중 3에 해당하는 모든 이미지를 제거하여 학습한다면?
    - Test error = 109
    - Error on 3 = 14/1010 (accuracy = 98.6%)
    - Soft label의 영향
  - 만약 MNIST 데이터 중 7, 8을 제외한 모든 이미지를 제거하여 학습한다면?
    - Test accuracy: 87%



Knowledge Distillation

- 장점
  - Teacher 모델의 성능과 특징을 모방한 작은 모델을 만들 수 있음
  - 추론 속도 및 학습 속도 향상
  - Teacher 모델을 통해 데이터를 올바르게 판단하는 법(Dark/distilled knowledge)를 배우므로 일반화 성능의 하락이 적음
  - 아키텍쳐나 하드웨어를 가리지 않고 사용할 수 있음
- 단점
  - Teacher 모델의 품질에 좌우됨
  - 최적화 과정이 복잡함(T, α 등 하이퍼 파라미터 조정이 어려움)

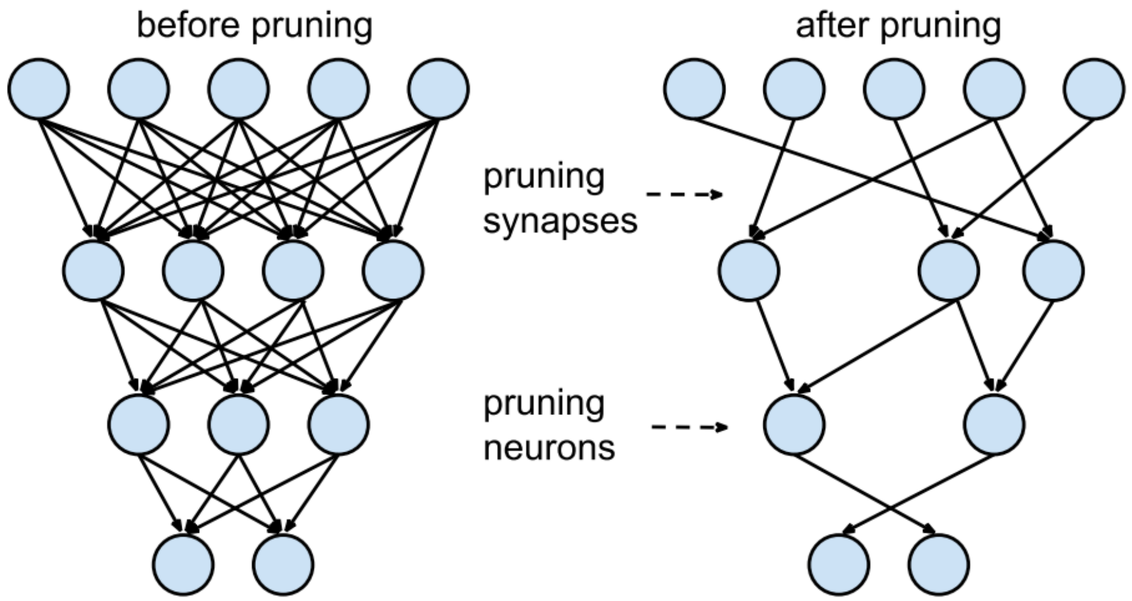
Knowledge Distillation

- Transfer learning과 KD의 차이점
  - Transfer learning
    - 대규모의 벤치마크 데이터셋을 통해 Foundation 모델에 사전학습 수행
    - 벤치마크 데이터와 무관한(다른 도메인의) 데이터로 fine-tuning
    - 이러한 경우를 Domain adaptation이라 함
  - Knowledge distillation
    - 유사한 도메인이나 동일한 데이터를 통해 Teacher 모델 학습
    - 동일한 데이터를 통하여 Student 모델 학습

4. Pruning(가지치기)

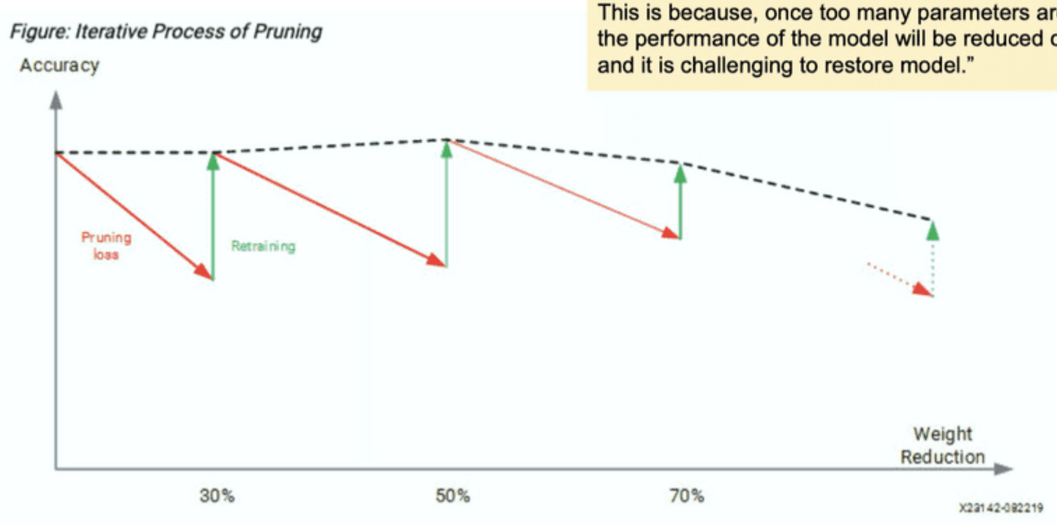
Pruning

- Pruning(가지치기)
  - 딥러닝 모델의 불필요한 부분을 제거하여, 모델의 크기를 줄이고 효율성을 높이는 기법
  - 파라미터 수나 계산량을 줄이며 복잡도를 줄임
  - 학습된 모델의 물리적 규모를 축소시키는 방법
- Dropout과 유사하지만
  - Dropout은 학습 시에만 적용되며
  - 계산에서 배제된 뉴런을 제거하지는 않음
  - Pruning은 몇몇 뉴런을 제거

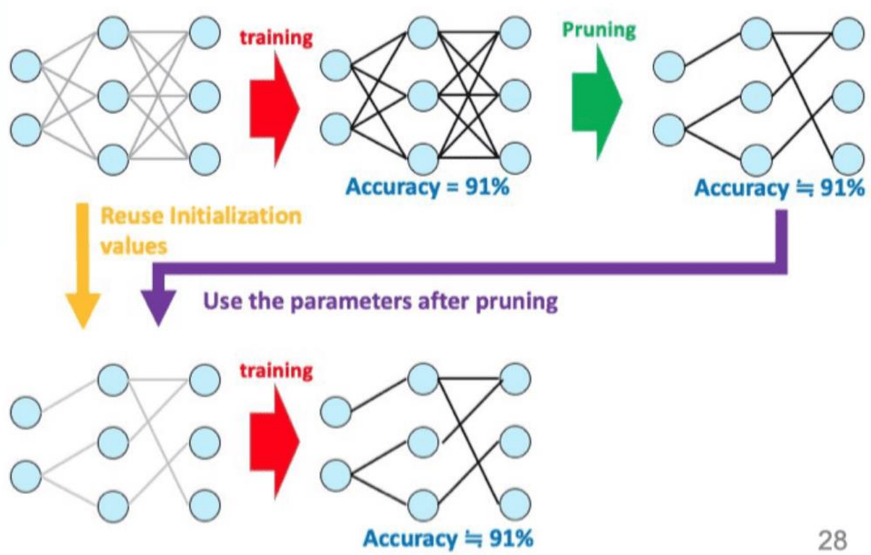


Pruning

- 기본 원리
  - 상대적으로 기여도가 낮거나 거의 사용되지 않는 가중치를 찾아야 함
    - 모델에 중요한 가중치와 그렇지 않은 것은 따로 있으므로
  - 다만 일부 가중치가 삭제될 경우 모델의 성능 하락은 필연적
  - 이 피해를 줄이기 위하여 제거-재학습 과정이 반복적으로 필요



"Pruning cannot be done to a smaller size at once.
This is because, once too many parameters are removed from the model, the performance of the model will be reduced obviously, and it is challenging to restore model."



https://blogik.netlify.app/BoostCamp/U_stage/45_pruning/

Pruning

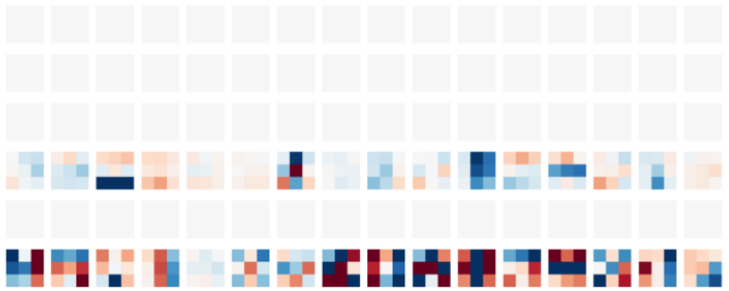
- Pruning 과정
  - 기본 모델을 데이터셋에 대해 학습
  - 학습이 완료된 후, 각 파라미터의 중요도를 측정
    - 중요도는 파라미터의 절대값 크기 또는 출력하는 Activation 값의 크기로 평가
  - 중요도가 낮은 가중치나 Layer 제거
    - 다만 지나치게 많은 부분을 제거할 경우, 유용한 파라미터까지 영향을 받을 수 있음
  - 모델이 여전히 좋은 성능을 유지할 수 있도록 재학습
  - 위 과정을 반복

Pruning

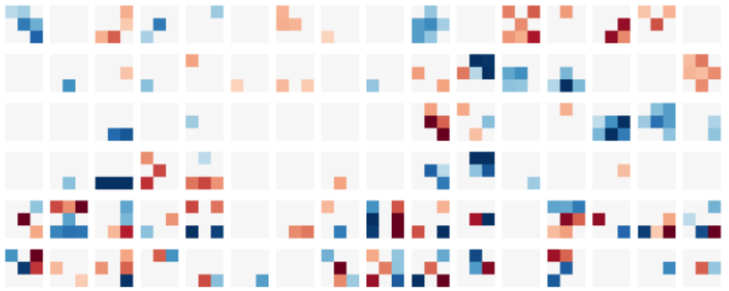
- 방식
  - Unstructured Pruning
    - 가중치 단위로 중요도를 평가하여 낮은 개별 파라미터를 제거
    - 이 과정이 반복되며 파라미터가 Sparse하게 변함
    - 많이 Sparse할 경우 특정 하드웨어에서 속도 개선이 이루어지기 힘듦(메모리만 절약

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 2 \\ 9 & 0 & 0 & 6 \\ 7 & 0 & 0 & 0 \end{bmatrix}$$

  - Structured Pruning
    - 모델의 필터, 채널, 뉴런 또는 레이어 전체를 제거
    - CNN: Filter, Channel



Structured: remove entire channels



Unstructured: remove individual connections

https://simpling.tistory.com/50

Pruning

- 장점
  - 모델의 파라미터 수를 줄여, 메모리 사용량과 스토리지 사용량을 감소
  - 특히 구조적 Pruning을 적용한 경우 추론 속도가 크게 향상
  - Unstructured Pruning을 통해 모델의 Sparsity을 증가시킬 수 있으며, 메모리 또한 절약
- 단점
  - 하드웨어에 따라 Sparse vector의 이점을 활용하기 어려울 수 있음
  - 신경망은 복잡한 구조를 띠고 있으므로, 불필요한 파라미터만 선택적으로 제거하기 힘듦
  - 재학습으로 인한 시간과 리소스 소요