LLM 경량화 방법들

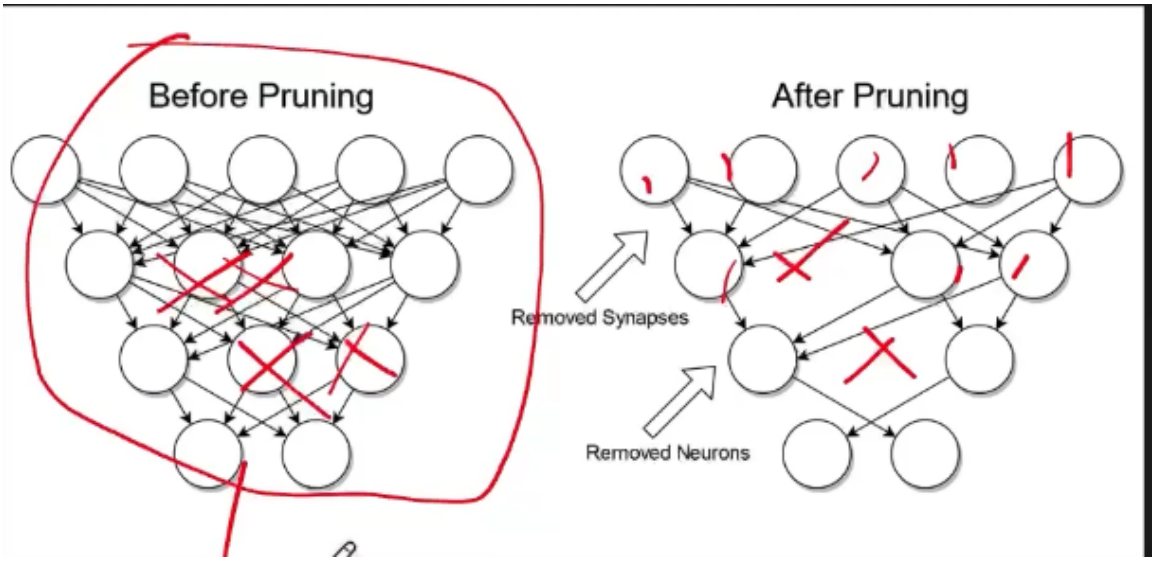
- FP16 → 2byte, FP32 → 4byte

주요 경량화 방법들

- 보통 Pruning + Knowledge Distillation 이 함께 사용된다.

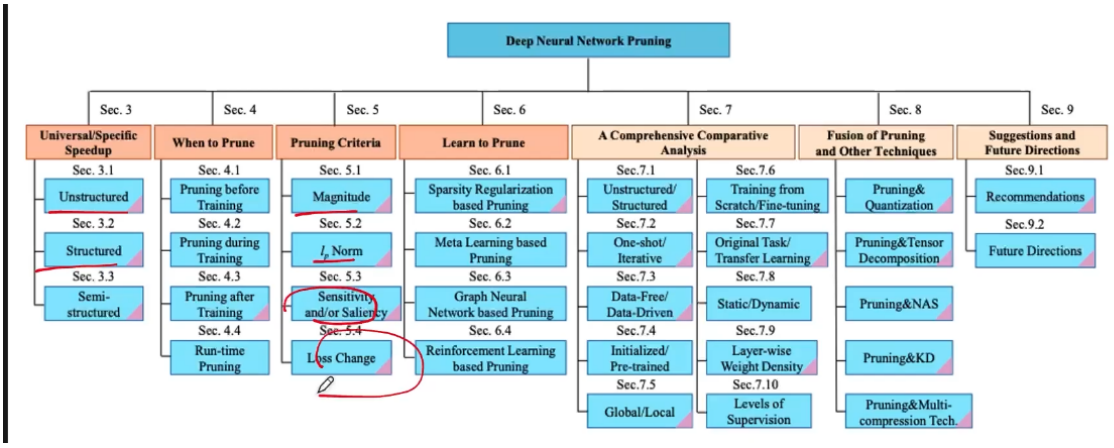
Pruning

- 개요

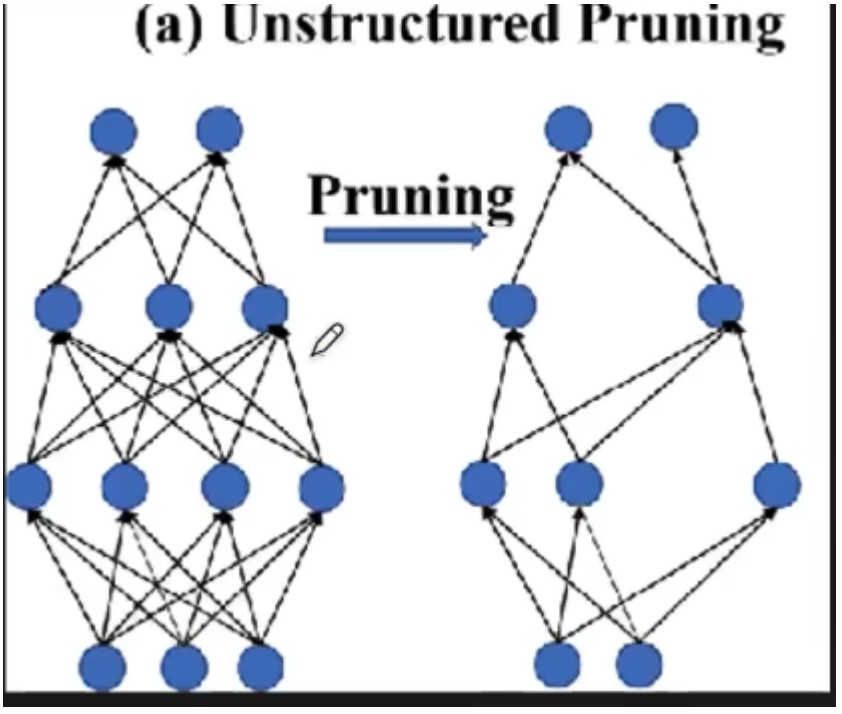


  - 모델을 경량화를 고려할 때 가장 기본적으로 고려하는 방법
  - 기존의 Neural Network의 노드 및 모듈들을 제거하며 전체적인 모델의 파라미터를 제거하는 방법
  - 대부분의 LLM 모델 개발 단계를 Large scale의 Foundation Model을 구축하고 이를 Pruning하여 모델을 경량화하는 방법론임
  - 이 단계에서 특정 기준을 기반으로 Pruning을 진행할 수 있으며, 각 레이어들의 차원을 유지하면서 줄여주어야 함.
  - Pruning이 끝난 단계에서는 Healing 이라는 과정을 통해서 경량화된 모델의 성능을 올려주는 작업이 필수적으로 적용됨.

- 보통 sLLM은 큰 Foundation Model의 경량화 버전이라고 볼 수 있다.
- Pruning 방법들



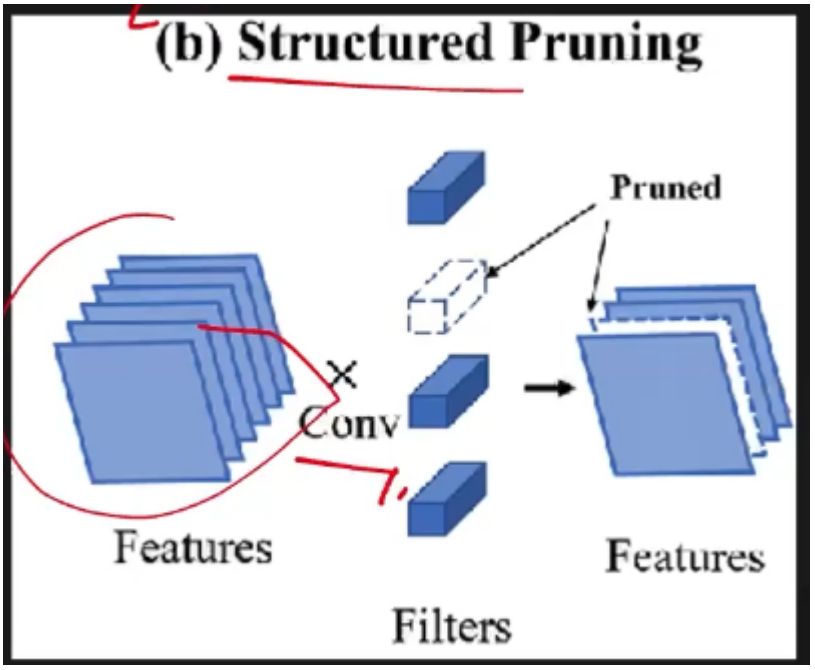
- **Unstructured Pruning (비구조적 프루닝)**



- 개별적인 가중치를 제거하는 방식, 모델 전체에서 특정 가중치 값을 0으로 설정.
- 제거되는 가중치가 개별적이므로 매우 높은 수준의 세밀한 조정이 가능
  (즉, 구조를 생각하지 않고 뉴런을 제거하는 방식)
- 불규칙한 메모리 접근 패턴을 가지므로, 일반적인 하드웨어 최적화가 어렵다.

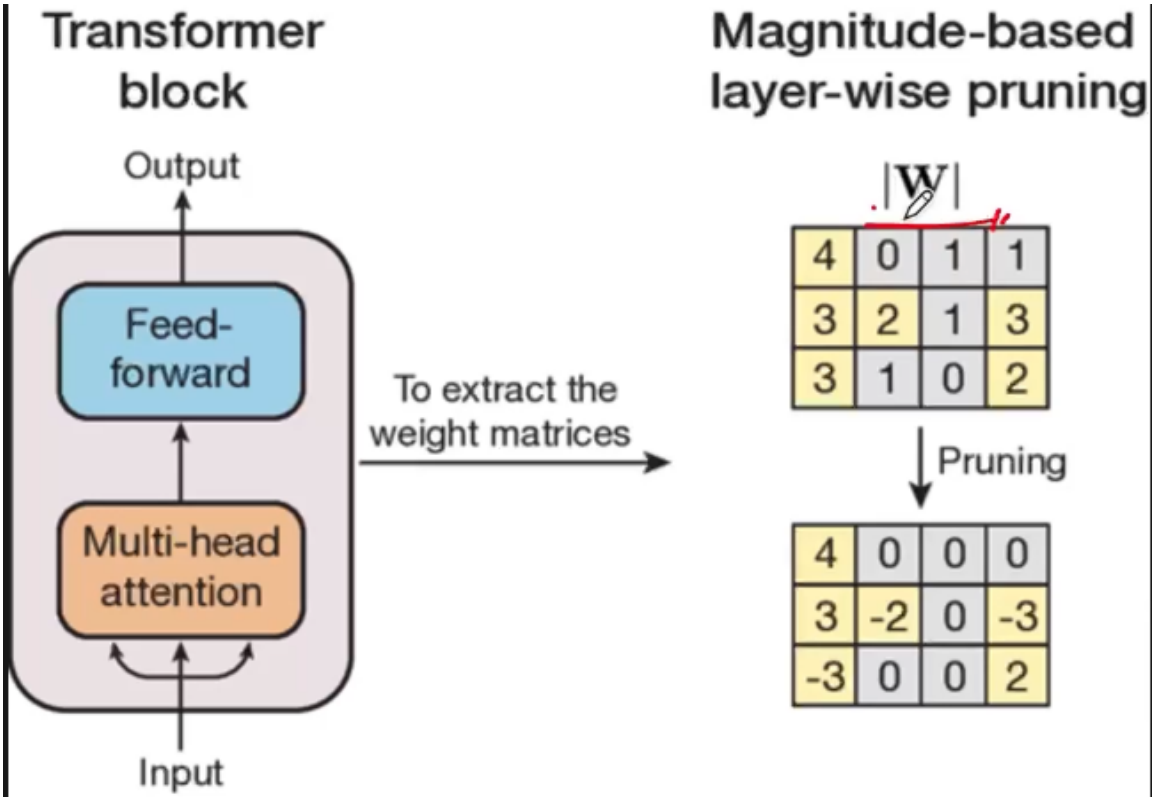
- 전용 하드웨어(e.g. sparse matrix 연산을 가속화하는 하드웨어)가 필요하기도 함.
- **높은 프루닝 비율에서도 모델의 성능 손실이 적지만, 실제 연산 속도를 높이기 위한 가속은 전용 하드웨어에 크게 의존**
- 장점: 성능 손실이 적다
  단점: 하드웨어 의존도가 높다, LLM과 복잡한 모델은 적용하기 어렵다, 구조를 생각하지 않는다. (입/출력의 위치)

- **Structured Pruning**



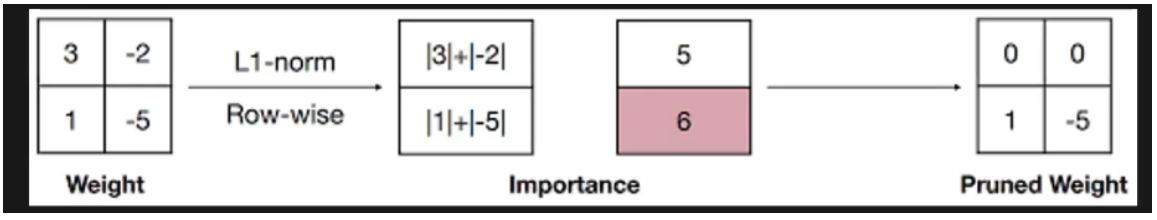
- 네트워크의 필터, 채널, 뉴런 또는 레이어 전체를 제거하는 방식
- 특정 구조를 통째로 제거하므로, 모델 구조가 단순해지고 메모리 사용량의 이점과 적용성에서 매우 간편함.
- 일반 하드웨어에서도 연산 속도 향상을 얻을 수 있으며 적용성이 쉽다는 점에서 차원맞추고 적용하는데 매우 효과적이다.
- 같은 프루닝 비율에서 비구조적 프루닝에 비해 성능 손실이 발생할 가능성이 크다.
  - 개별 가중치가 아닌 네트워크 구조 전체를 제거하기 때문
  - 필요한 레이어 안의 필요한 뉴런이 제거될 수 있다.
  - 요즘은 Criteria(기준)를 정의해서 제거하는 방식으로 진행되고 있다.

- **Magnitude Pruning**



  - 가중치의 절대값 크기를 기준으로 가중치의 중요도를 판단.
  - 크기가 작은 가중치는 네트워크 성능에 미치는 영향이 적다고 가정하여 제거를 진행
  - 단순하고 계산이 간단하여 적용이 쉬우며, 많은 프루닝 기법에서 기본적으로 사용
  - 다만, 구조적으로 적합한 뉴런 혹은 가중치를 선택해야 하기에,
    이를 칼럼 기준으로 제거하거나 layer 기준으로 적용하는 방법론으로 활용.

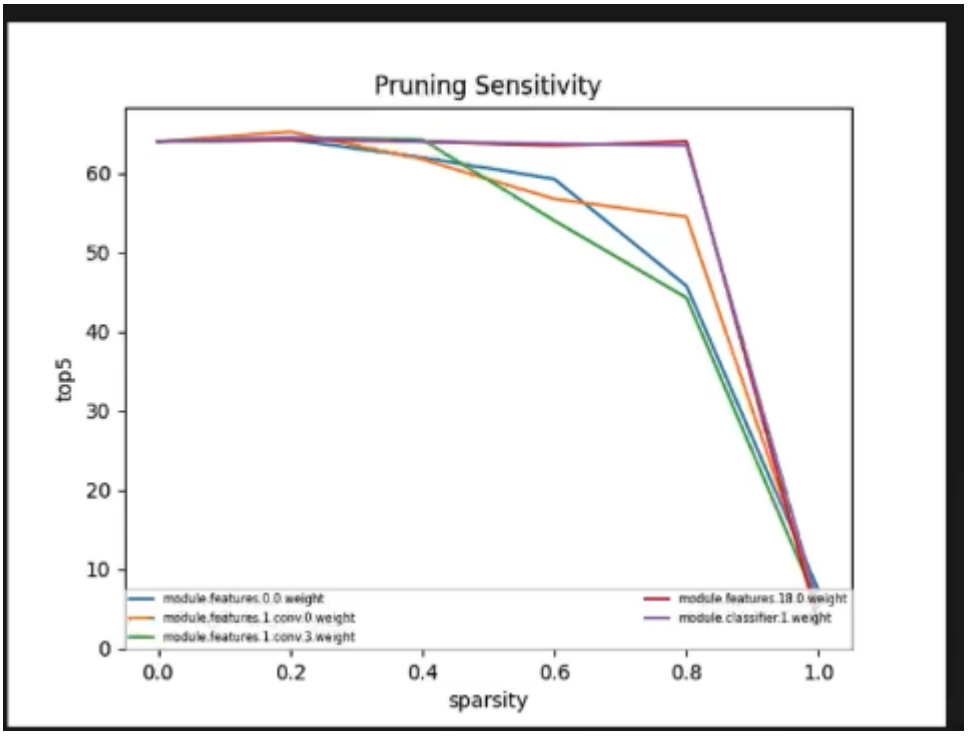
- **Lp norm based pruning**



  - 가중치 혹은 필터를 lp 노름 기반으로 중요도를 측정함.

- Norm 값이 낮은 가중치 혹은 필터를 우선적으로 제거하게 되며,
  일반적으로 P=1, 2를 적용해 사용됨
- 가중치의 크기 뿐만 아니라, 가중치 벡터의 전체적인 크기와 밀접한 연관이 있어 필터나 채널 단위(CNN일때)의 구조적 프루닝에 적합
- 다만, 구조적으로 적합한 뉴런 혹은 가중치를 선택해야 하기에, 이를 칼럼 기준으로 제거하거나 layer 기준으로 적용하는 방법론을 활용.

- **Sensitivity (민감도 기반 프루닝)**

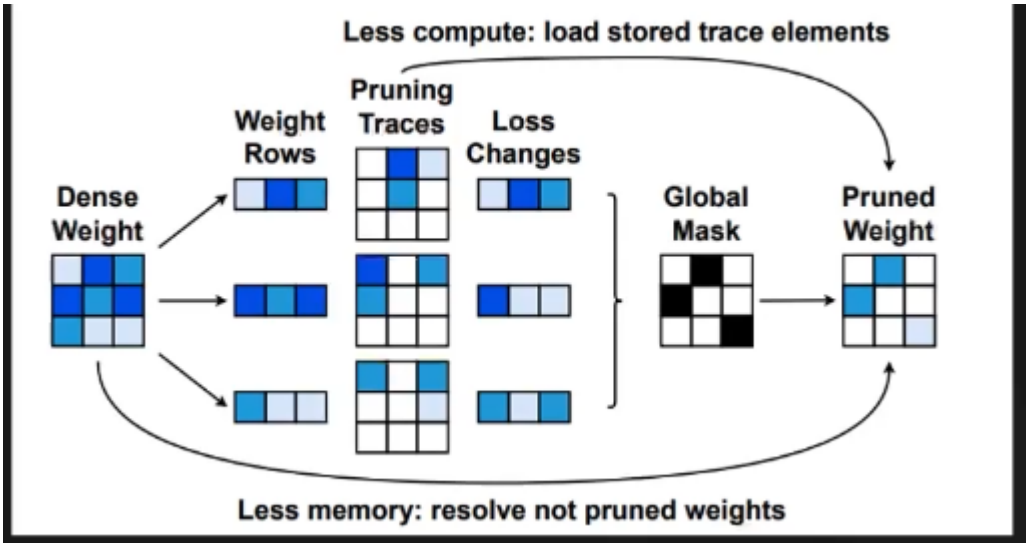


- 가중치의 중요도를 해당 가중치가 네트워크 손실에 미치는 영향도를 평가
- 손실 함수의 가중치에 대한 기울기(민감도)를 활용하여 민감도가 낮은 가중치를 우선적으로 제거
- 특정 가중치가 네트워크 성능에 실제로 미치는 영향을 고려하므로, 보다 정교한 프루닝이 가능

가중치 w_j 민감도는 다음과 같이 계산이 가능 $s_j(W;\ D) = \frac{|g_j(W;D)|}{\sum_{k=1}^{M}|g_k(W;D)|}$

- g_i → loss func
- 전체적인 가중치의 합에서 얼마만큼의 영향을 미치는 지를 계산함.

- **Loss Change (손실 변화 기반 프루닝)**
  - 직접적인 Loss가 얼마나 변화하는지를 확인함.



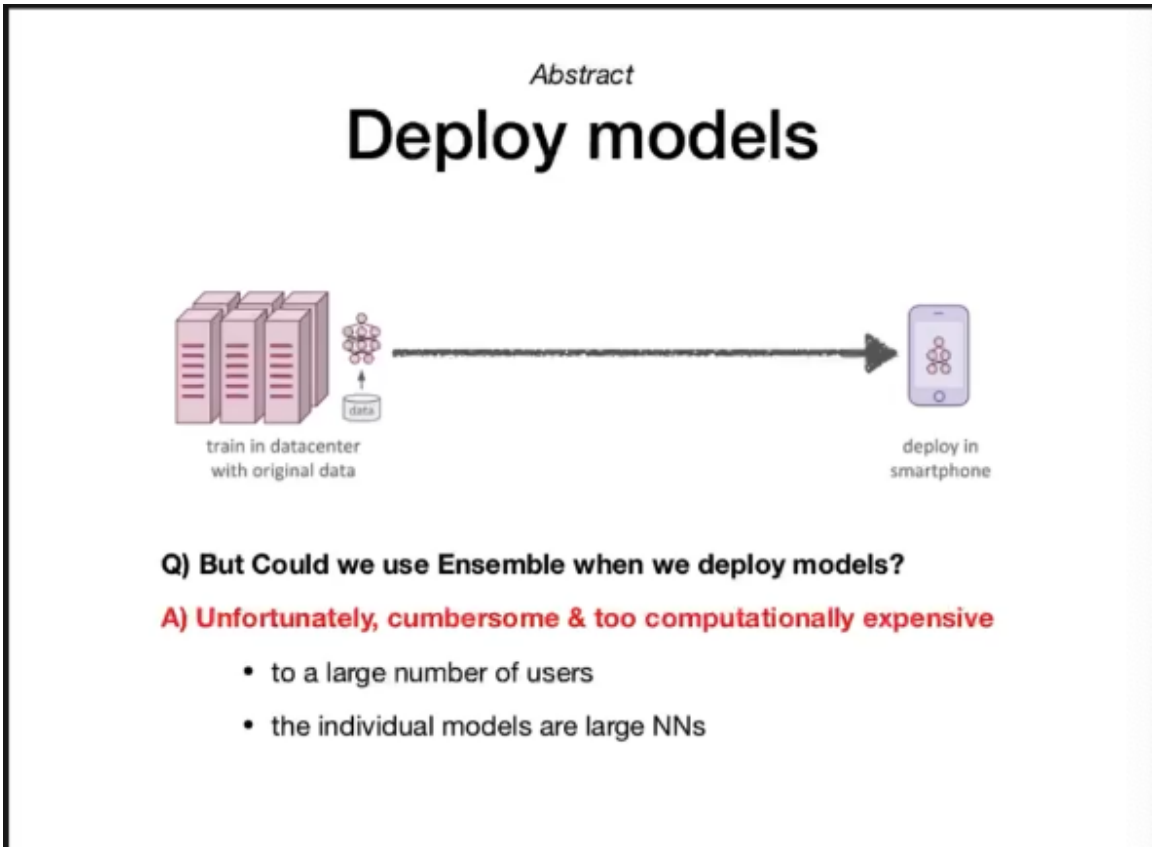
  - 가중치를 제거했을 때, 네트워크의 손실 변화량을 기반으로 중요도를 평가.
  - 손실 함수의 테일러 급수 전개를 사용하여 특정 가중치가 0이 되었을 때 발생하는 손실의 변화를 추정하고, 손실 변화가 적은 가중치를 제거
  - 이 수식은 기계학습과 딥러닝에서 테일러 급수를 사용하여 손실 함수의 변화를 근사
  - 특히, 가중치를 프루닝할 때 손실 함수에 미치는 영향을 평가하기 위해 사용
  - 네트워크 성능에 가장 덜 영향을 미치는 가중치를 선택적으로 제거할 수 있어 성능 저하를 최소화

예시: 손실 함수의 1차 테일러 전개로 손실 변화를 근사 $\Delta L = L(W + \Delta W) - L(W) \approx \nabla_W L \cdot \Delta W$

Knowledge Distillation

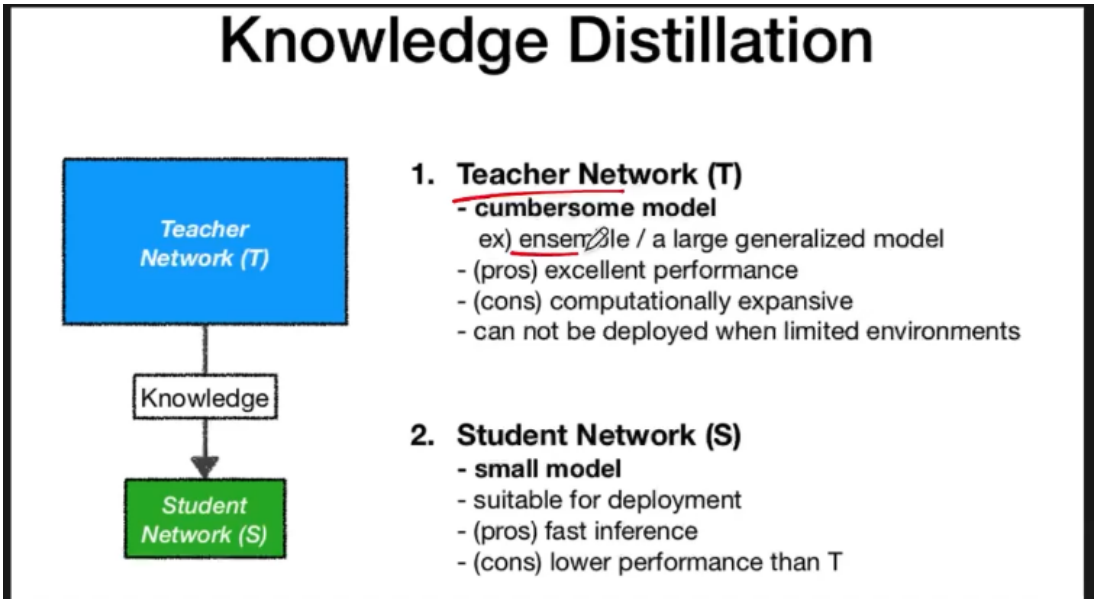
- **개요**
  - Knowledge
    - 모델이 가지고 있는 지식이나 정보 혹은 Task에 대한 이해
  - 딥러닝에서는 KD는 Teacher Model로 부터 증류한 지식은 Student Model로 전달하거나
    주입하는 일련의 procedure로 볼 수 있음
  - 즉, Teacher Network의 방대하거나 잘 훈련된 지식을 이 보다 더 작은 Network인 Students Network에게 전달하는 방법론으로 볼 수 있음
  - 다만 KD를 하기 위해서는 well-defined된 Teacher Model이 필수적으로 필요함.

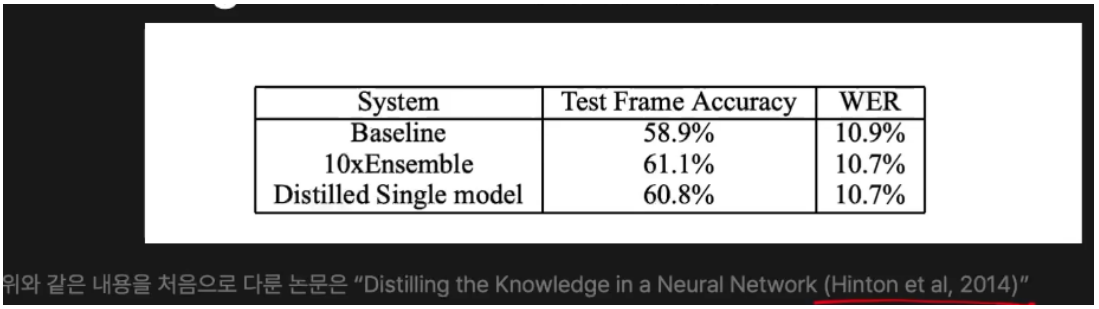
- **등장 배경**



  - 모델을 Deploy 하기 위해서는 많은 자원과 지속적인 feeding 과정이 필요
  - 다만 모델의 size가 커지면 배포와 각각의 쿼리를 처리하는 데,
    매우 복잡도가 높아지며, 이를 해결하기 위해서는 다량의 자원과 처리를 위한 시스템 설계 및 관리가 필수적으로 이루어져야 함.

- A 모델의 정확도가 95% 이고, B 모델의 정확도가 90% 이지만
  A 모델의 추론속도가 10분, B모델이 1분이라면?
- **이때, A의 지식을 B에 넘겨줄 수 있지 않을까?**



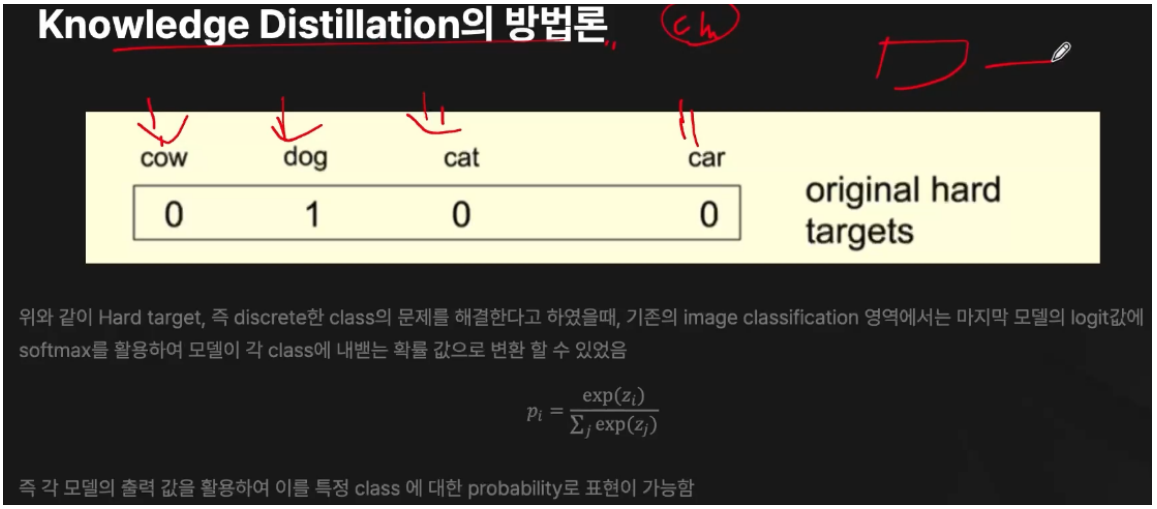


System	Test Frame Accuracy	WER
Baseline	58.9%	10.9%
10xEnsemble	61.1%	10.7%
Distilled Single model	60.8%	10.7%

위와 같은 내용을 처음으로 다룬 논문은 "Distilling the Knowledge in a Neural Network (Hinton et al, 2014)"

- 복잡하고 헤비한 모델을 다수의 유저에게 디플로이 하는것은 자원적으로 어려움
- 특히, 앙상블과 같은 모델을 고려했을 때, 이 모델이 가지는 지식을 단일 모델에 전달하는 방법론은 있으나, 좀 더 일반화하여 큰 모델이 가진 지식을 증류하여 단일의 작은 모델에 전달하였을 시 효과적으로 전달이 되는 것을 확인
- 앙상블과 같은 모델의 학습시간을 단축시키는 새로운 방식의 앙상블 기법을 연구 및 제시

- **방법론**



- Hard Target: 클래스에 대해 명시적으로 주어진 것
- 만약 학습이 잘된 모델에 강아진 사진을 넣게 된다면, 실제 모델의 출력값은 cow, dog, cat, ..., car에 대한 Logit 값이 될 것이고, 이를 softmax를 취하게 되면 위와 같은 probability 값으로 활용 할 수 있음.
- 그러면 위와 같은 dog를 제외한 class 의 값들은 매우 적은 값들이 되지만 이 값들은 의미 없는 값들일까?
  - 이 값들은 겉으로 보기에는 의미가 없다고 볼 수 있지만, 모델이 가지고 있는 지식으로 해석할 수 있음
  - Cow나 car에 대한 값들은 매우 적지만, 그나마 cat에 대한 확률 값이 가장 높다!
    - 이를 확장하면 강아지라는 사진은 car나 cow보다는 고양이에 대해서 더 높은 값을 가지고 있는 지식이다.

- 그러나 Cow, car에 대한 값은 매우 적은 값을 가지고 있어, 하나의 정보 혹은 지식으로 사용하기 어려움
- 이를 soft하게 만들면 이 분포가 조금 더 고르게 펴지게 되고, 이를 활용하면 모델의 지식으로 활용가능한 정보가 됨
  - 이게 바로 KD의 시초이며, soft ouptut이라 할 수 있으며, 논문에서는 dark knowledge라고 할 수 있음
  - T: Temperature (얼마나 soft하게 만들 것인가?)

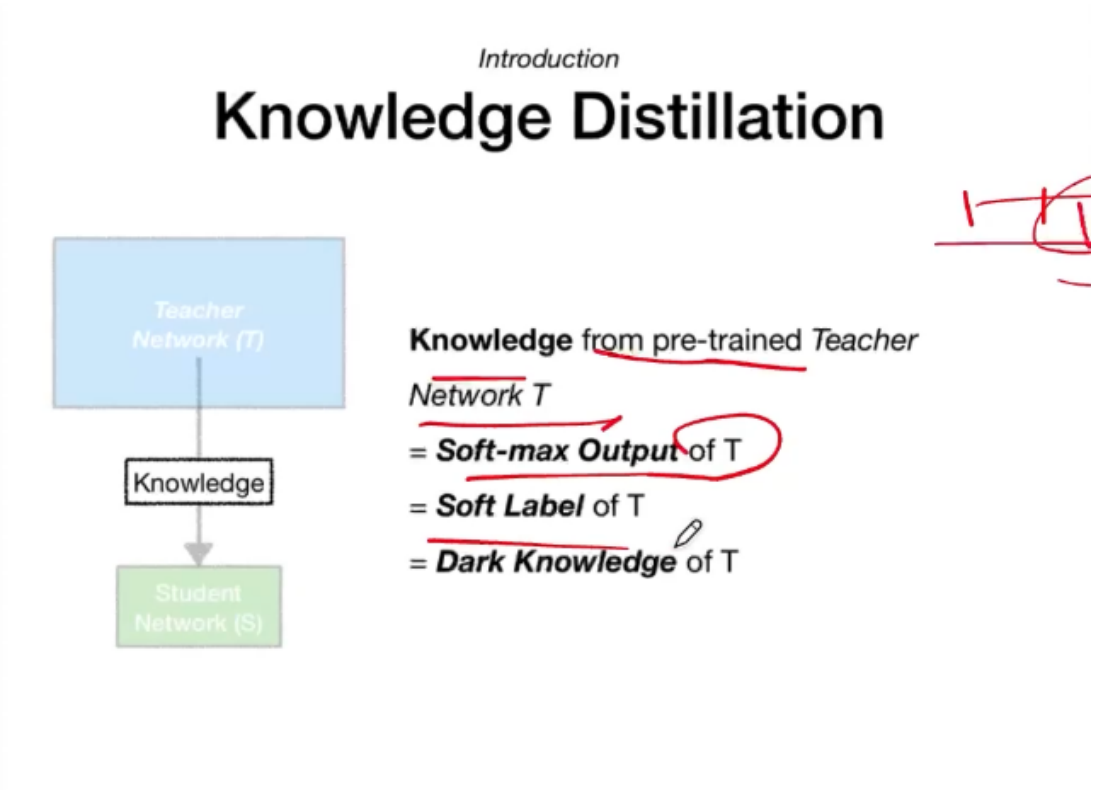
$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

- 보정 전과 후

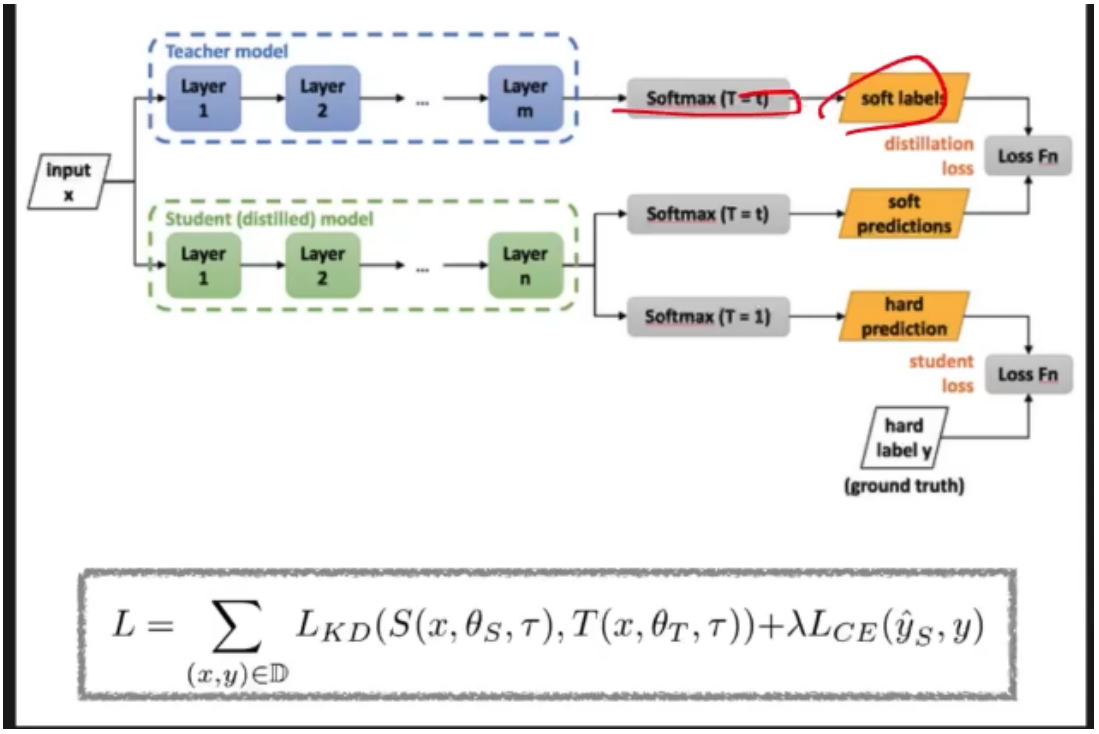
cow	dog	cat	car	
10^{-6}	.9	.1	10^{-9}	output of geometric ensemble

cow	dog	cat	car	
.05	.3	.2	.005	softened output of ensemble

Softened outputs reveal the dark knowledge in the ensemble.



- **distillation loss**
  - Soft하게 만든 Soft Target은 Teacher 모델의 지식이라고 할 수 있음
  - 이 지식을 Student model에 전달하는 방법은 다음과 같음.



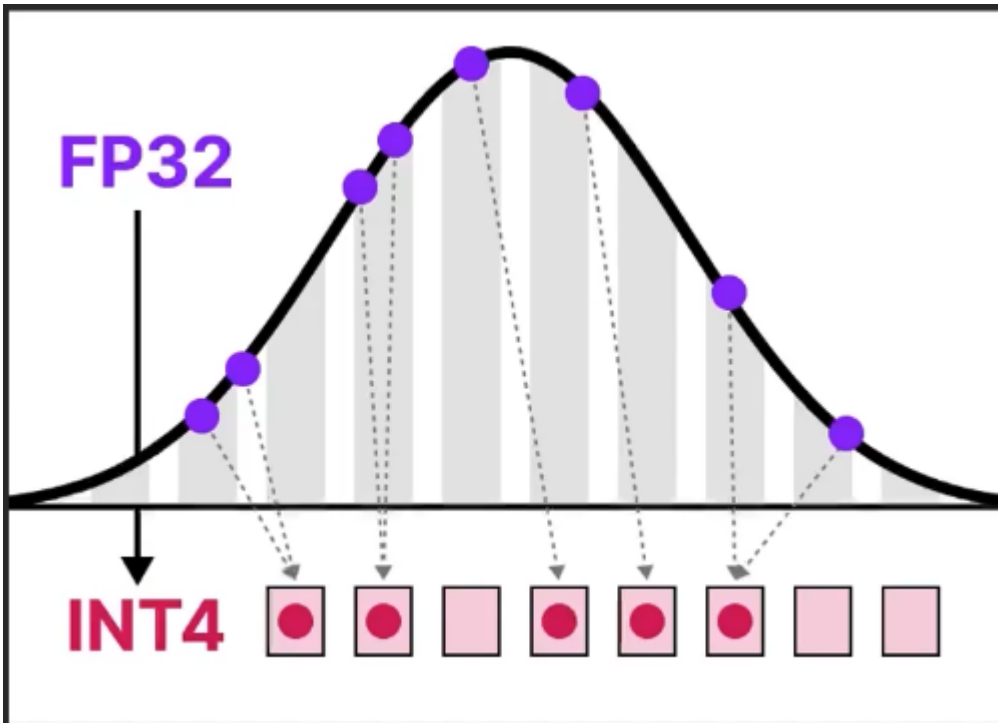
- Teacher 모델을 학습하고, 손실함수를 통해서 작은 Student 모델을 학습시키는 방식으로 적용할 수 있음
  - 즉, Teacher를 학습, Student 모델을 학습하게 적용
- Student 모델을 학습시, Student 모델의 soft prediction과 teacher model의 soft label을 사용하여 distillation loss를 활용
- Student Network (hard) prediction과 Original (hard) label을 활용하여 classification loss 구성해서 학습을 시킴

- **KD 와 Transfer learning?**
  - Transfer learning은 다른 domain adaption 영역에서 적용하는 방식이며, 새로운 domain에서 지식을 전달하는 방식
  - KD는 도메인 차이점이 없으며, 같은 도메인 내의 Teacher 모델에서 Student 모델로 지식을 전달하는 과정을 의미할 수 있음
    - **즉, 모델을 압축하고 간소화 하는 적용에 있어 KD를 활용**
  - 대형 모델에서는 KD가 가장 효과적인 방법론이며,
    LLM에서 모델의 가중치를 가지치기 하고, 이를 적용해서 훈련시킬 수 있음.

- 관련 논문
  - NVIDA → LLM Pruning and Distillation in Practice: The Minitron Approach

---

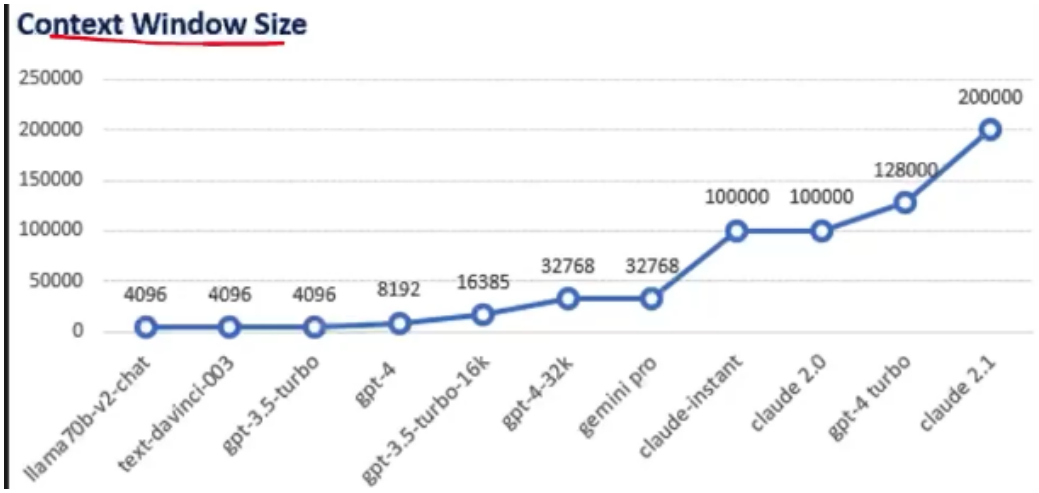
Quantization



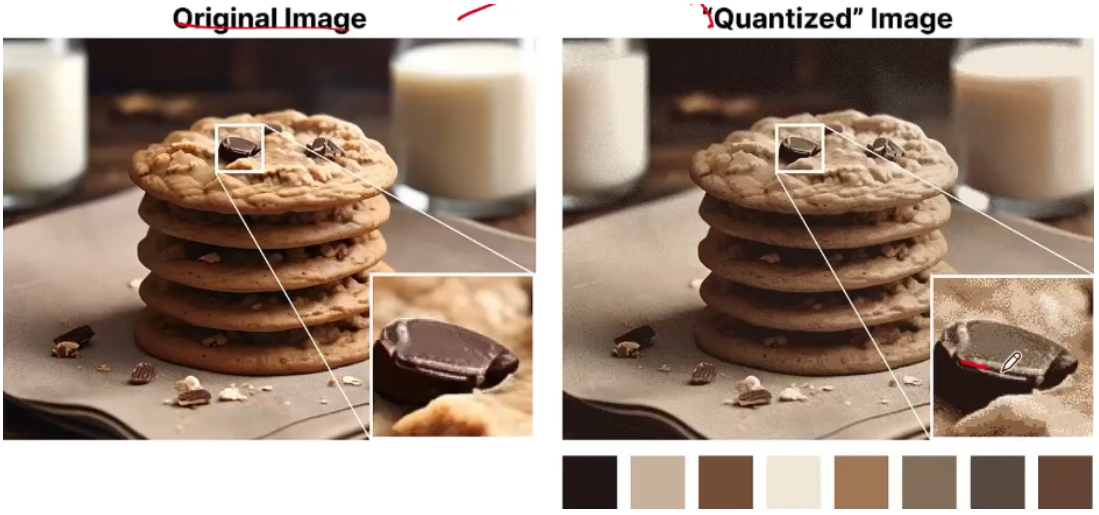
- **등장 배경**
  - 딥러닝 기술의 발전
    - 딥러닝 기술의 발전과 함께, 특히 LLM이나 컴퓨터 비전 모델에서 수십억 개 이상의 파라미터를 가지는 모델이 등장
  - 모델 효율성 및 배포문제
    - 대규모 모델을 실시간 애플리케이션이나 배포 가능한 상태로 만들기 위해서는 모델의 크기를 줄이고 계산량을 최소화해야함
    - 프루닝은 모델의 웨이트를 직접적으로 제거, 조정하는 방식이다보니 항상 힐링과정을 거쳐주어야 함.
      - 즉, 매핑을 잘 시켜서 효율성을 높이겠다.
  - LLM에서 모델의 발전은 수많은 리소스 자원을 필요로함.

- 특히, Tansformer의 확장성으로 LLM의 파라미터 수가 증가할수록 모델의 성능이 증가하였고, 이를 통해 대규모 파운데이션 모델들이 등장
- 아무리 프루닝을 하여도 구조적인 한계점이 명확

- 메모리를 줄이면서 성능을 최대한 유지하는 방법이 무엇일까?
  - 양자화를 통해 더 적은 메모리로 모델을 표현하고,
    이를 통해 값들을 표현해 나가면 되지 않을까?
- 이러한 이유로 LLM에서 다양한 양자화 기법들이 등장하였음.
  - 또한, Context Window Size가 점점 커지고 있다.



- Data Type을 줄여도 어느정도 표현력을 유지!



- Quantization은 딥러닝에서만 사용되지 않는다.
  - 컴퓨터과학, 신호처리에서도 사용됨

- 예: Product Quantization