<수업 내용 정리 및 소감>

기존에도 AI알고리즘이 요구하는 연산량은 점점 늘어났고 그때까지는 문제가 되지 않았지만, 2018년 transfomer라는 모델을 베이스로 하는 고성능의 AI가 빠르게 증가했고, 그에 필요한 계산량은 급수적으로 늘어났다. 무어의 법칙으로 계산되는 반도체의 발전속도(1년 반에 2배)보다도 훨씬 (2년에 750배)더 빠르게 발전했다. 심지어 무어의 법칙도 틀렸다(점점 발전하면서 생기는 미세공정의 한계)

그렇기에 압축할 필요성이 있다. quantization, pruning등의 압축방법이 있으며, 2개를 주로 사용한다.

연산을 하기 위해서는 컴퓨터가 처리 가능한 0과 1의 바이너리 파일로 원래 값을 변환하는 과정이 필요하다. quantization은 그 과정에 비트를 최대한 줄이며 연산량을 줄이는 과정이다. ex) 64bit의 값을 8bit연산시 메모리 하나당 표현 가능한 값 증가, 연산에 필요한 수치 감소. 처리 효율이 매우 늘어난다. 다만, 여러 bit 계산을 지원하는 연산 기구가 필요하다.

이 과정은 연산기를 쪼개서 사용한다. bit-fusion 방식은 bit를 나누어서 shift를 통해서 연산기를 병렬적으로 사용한다. 다만, 알고리즘이 곤란하다. bit에 따라서 나누는 과정이 힘들다. bitblade 방식은 발전한 방식으로, bit를 나누어 shift하는 과정을 묶어서 스킵한다. shift에 해당하는 연산기를 각각 사용하며, 동일한 shift수치들끼리 연산을 진행한다. 또한 연산기에서 곱셈을 xnor로 대체 가능하고 이는 더 빠른 연산을 만들 수 있다. 그래도 연산기를 마냥 넣으면 노는 친구들이 생겨서 매우 비효율적이다. 고효율을 위해서 이것도 쪼개서 병렬적으로 활용한다. 이걸로 곱셈과 덧셈 연산은 끝이다. 다만 AI가 곱셈 덧셈만 있는 것은 아니다.

신호의 증폭 혹은 무의미한 신호를 제거하는 과정도 필요하다. 이 과정은 Activation Fuction(활성화 함수)를 거쳐서 이루어진다. 원래는 무시했는데, 최대한 줄이다 보니까 집중하게 되었다. 연산이야 연산기를 많이 넣어서 병렬처리하면서 때려막았는데, 이 친구들은 exponential등의 고차원 연산이 필요하다. 특수한 연산기를 도입하거나 샘플테이블을 도입해서 계산을 스킵하는 방법으로 극복한다.

pruning도 좋은데 이 친구는 웨이터 데이터를 최적화하는 방식이다. 쓸모없는 값을 0으로 그냥 바꿔버리는 등의 방식을 통해서 필요한 연산치를 최소로 줄이는 방식이다. separable convolution, depthwise convolution처럼 분리해서 연산하는 알고리즘은 별로 이득이 없기에 비주류이다. shift convolution이나 쉬프트를 통해서 연산을 병렬 처리한다.

인공지능은 당연하게 고성능의 알고리즘이기에 무조건 많은 리소스가 필요하고, 좋은 성능의 컴퓨터 GPU등이 필요하다고 생각했다. 다만, 생각보다도 Ai의 발전속도에 비해서 하드웨어의 발전속도가 너무 느렸다. 생각보다 큰 문제라는 생각이 들었다. 그래서 이번 수업은 상당히 흥미롭게 들었고, 하드웨어보다 소프트웨어가 훨씬 중요하다는 내 생각이 틀렸다는 것을 알게 되었다. 그리고 그 문제에 대한 해결방법이 AI개발에서 중요한 병렬접근을 통한 방법이라는 것이 흥미로웠다. 요새 코딩을 더 배우면서 캡슐화, 모듈화, 그리고 그를 통한 최적화 등의 병렬처리에 대한 공부의 필요성과 관심이 늘어나고 있었다. 이번 수업이 확실히 병렬 처리에 대한 중요성과 문제를 해결하는 여러 방법에 대한 아이디어를 얻게 된 것 같아서 좋았다.