Neural Architecture Search 의문점 정리

1. **Figure2에 있는 일련의 RNN구조는 인간이 설계**
   - 이 논문에서는 일반적인 CNN 구성을 따름, Layer 별 구조가 다를 수도 있지만, 해당 논문에서는 동일한 구조를 반복함.(구현 단순화, 탐색 공간 증가 억제, 일정한 구조 하에서 성능 차이를 더 명확히 측정 가능)
2. **모든 controller replica는 같은 RNN구조를 공유한다.**

   - 같은 층 수
   - 같은 hidden state 차원 수
   - 같은 action sequence 정의

   하지만 같은 RNN 구조로 만든 child replica 라고 하더라도 내부에서 파라미터를 확률적으로 선택하고, 어떤 이전 layer 와 연결할지 선택하는 Skip connections 마저 확률적으로 선택하므로 **모든 child replica 의 구조는 다 다를 수밖에 없음.**

3. **모든 controller replica는 같은 파라미터 θc를 공유하며, parameter server에서 업데이트된 최신 값을 주기적으로 동기화(sync)**
   - 각자 독립적으로 샘플을 생성하고 평가
   - reward로부터 계산한 gradient를 다시 parameter server로 전송
   - parameter server는 이를 기반으로 θc를 업데이트
   - 업데이트된 θc는 다시 모든 controller replica에 반영

   이 θc는 각 layer에서 어떤 선택을 할지 결정할 확률분포를 조정하는 파라미터. 여러 child 중 성능이 좋은 것들: reward (validation accuracy 등)를 받음 → 그 reward를 기반으로 해당 구조를 생성했던 확률 경로(=log prob)를 강화 (REINFORCE) → 그 결과 θc는 그런 구조를 더 자주 생성하도록 업데이트됨

   Ex) 초기상태(filter height): 3 (0.33), 5 (0.33), 7 (0.33)

   여기서 filter width = 5가 좋은 결과를 자주 냈다면, controller는 다음 번에도 그것을 더 높은 확률로 샘플링하게끔 θc를 업데이트

   업데이트된 상태: 3 (0.10), 5 (0.75), 7 (0.15)

4. **복잡성을 늘리기 위해서 skip connections 추가**

   구조적 표현력을 증가시키기 위해

   set-selection type attention: 현재 layer에서 이전 layer들과의 skip connection을 만들지 말지를 결정하기 위해 각 이전 layer를 하나씩 independent하게 판단하는 방식, 선택된 layer는 모두 현재 시점의 입력으로

   Anchor point: controller RNN이 과거에 생성한 layer들의 hidden state를 저장해 두는 지점, 각 layer를 생성한 시점의 hidden state h_j를 anchor point로 저장

   - 정보 보존: 깊은 layer에서도 초기 layer의 정보가 직접 전달되어 gradient 흐름과 표현력을 보완
   - 다양성 증가: 구조적으로 더 복잡하고 풍부한 표현 가능 (DenseNet, HighwayNet 등의 성능 향상 요인)
   - 탐색 공간 확장: 여러 skip 연결 조합을 통해 훨씬 다양한 아키텍처 가능

5. **3.4에서는 NAS로 고성능 RNN을 생성하는 방식을 고안**
   - 순차적인 layer의 출력을 다음 layer의 입력으로 받는 방식 유지.(figure5에서 트리구조로 비유)
   - cellstate를 어디에 삽입할 것인지도 강화학습을 통해 학습.