# Parameter Efficient Transfer-Learning for NLP. <paper>

## <abstract>

fine-tuning은 다른 다운스트림 테스크가 들어올때마다 기존 parameter/model 재학습시켜야 했음. 이 대신 adapter module을 활용한 transfer을 소개함.
기존의 학습된 모델의 parameter은 그대로 두어 공유하고, 새로운 task를 추가함

## < Intro >

BERT : large text corpora로 비지도 loss와 함께 학습된 Transformer Network. (Text classification / 추출 QA)
목표는 BERT로 다양한 task를 수행할 수 있는 전이학습 application을 만드는것. → Compact / Extensible downstream Model

① Compact : 각 테스크에 적은 파라미터의 추가만으로 많은 테스크 해결가능.
② Extensible : 전의 것을 잊지 않고 새로운 테스크 해결가능.

이전의 전이학습은 fine tuning과 feature-based transfer learning으로 나뉨. 여기서 adapter module을 새로 제안함.
↳ 사전학습된 real-valued embedding vector을 사용.
↳ 사전학습 Network에서 weight를 copy, downstream에 맞게 튜닝.
→ 각 테스크마다 새로운 weight set이 필요함. 하지만 fine tuning보다 adapter은 두배가까이 적은 param으로 비슷한 성능 보임.

Adapter? : 사전 학습된 네트워크의 층사이에 추가되는 새로운 모듈. base model + adapter module

$\phi_w(x)$ (function with parameters $w$)
- FB : $\chi_v(\phi_w(x))$ - 새로운 함수 $\chi$, $v$만 학습.
- FT : compactness를 제한하여 각 테스크에 $w$를 조정.
- AT : $\psi_{w,v}(x)$, 새로운 함수 정의, $w$는 copy, 초기 파라미터 $v_0$는 $\psi_{w,v_0}(x) \approx \phi_w(x)$에서 옴. $v$만이 학습.
  ↳ $w$는 고정되어있기 때문에 전의 모델에 영향주지 X. 보통 기존 network $\phi_w$에 새로운 레이어를 추가하는 방식.
  (param)

① 수식적차이 against Feature-based / fine tuning.

② 방식의차이 against Multitask / Continual
- Multi-task Learning : 모든 task에 동시적인 접근 요구 ⇒ AT는 아님.
- Continual Learning : 끊임없는 task 처리, 하지만 재학습 할때는 전의 task를 잊어버림.

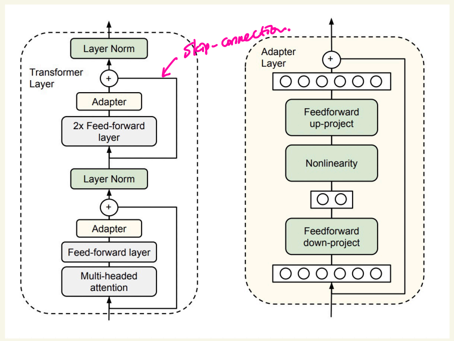
⇒ 반면 AT는 공유된 param에 대해 동결하기 때문에 완벽하게 전 task를 기억함.

## <Adapter tuning for NLP>

Goal : ① 좋은 성능. ② 데이터에 동시적접근이 아닌 순차적 학습 ③ 각 테스크에 적은 수의 파라미터만 추가한 것

Deep network의 vanila fine-tuning에서는 네트워크의 가장 위 layer에 수정이 가해짐. 이는 upstream과 downstream task의 label space와 loss가 다르기 때문에 필요해지게 됨.
하지만 Adapter tuning에서는 가장 위에 layer을 하나 더 추가하고 이 layer의 parameter을 random으로 초기화시켜 학습시킴.
이는 테스크가 추가되어도 총 model size의 증가율을 줄여주고, near-Identity initialization을 Adapter에 적용하여 기존의 network에 영향을 주지 않고 좋은 성능의 학습을 유임.
↳ layer 하나씩만 추가.
↳ I 항등에 가까운 값으로 param 초기화. 기울기 소실, 폭발 문제를 예방.

## <Instantiation for Transformer Networks>



- Feed forward와 Attention layer 모두 projection을 통해 input output size를 맞춤.
- Both use skip-connection. ( x + F(x) / 입력에서 더해야할 차이를 학습 ) → Normalization
- 각 sub-layer 뒤에 adapter을 직접적으로 apply (projection 후, skip-connection 전), 그 뒤 바로 Normalization에 입력됨.
- parameter 수를 제한하기 위해 bottleneck architecture 제안. (특징압축, 차원축소)
  - d-dim의 feature을 더 작은 크기의 m-dim으로 project. ( m-bottleneck dimension)
  - 각 레이어에는 bias를 포함하여 $2md + d + m$ 만큼의 parameter가 추가됨.
  - 결과적으로 기존 모델 파라미터의 $0.5\% \sim 8\%$ 가량을 사용.
  - Adapter은 skip-connection을 이미 포함.
  - projection layer의 파라미터가 0에 가깝게 초기화되면, module은 거의 Identity function으로 초기화됨.
- 각 task마다 new layer normalization parameter을 학습. (conditional batch normalization과 비슷).
  - 각 레이어 마다 2d 만큼의 parameter가 추가됨.

## <Experiments>

GLUE를 포함, 다양한 데이터에 적용
- Adam 최적화
- Pre-trained BERT Transformer network. (24 layer, 330M param)

→ classification task에 있어 최고 성능은 fine-tuning이지만 적은 parameter만으로 적은 성능차이를 이끌어냄.
↳ ex) FT : 9x , AT : 1.3x