**데이터사이언스를 위한 머신러닝 및 딥러닝 2**

**HW4 (Self-supervised Learning)**

**서울대학교 산업공학과**

**2021-28863 유재상**

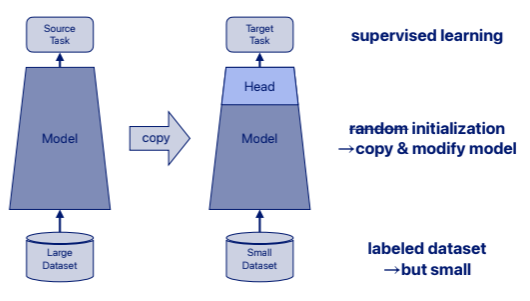
Self-supervised learning method 중 하나인 contrastive learning에 대한 코드를 돌려본 게 저번 3회 과제였습니다. 이번엔, 다른 분류모델인 CIFAR 100의 모델을 학습하는 것을 해보고자 하는데, 처음부터 learning을 하는것과 transfer learning 이 두가지의 비교를 진행하고자 합니다. 다만, 이 역시 google colab의 무료버전을 사용하기엔 시간이 너무 오래 걸려서, 하나의 유로버전 계정과, 다른 기타 무료 계정을 두개를 병렬적으로 돌렸습니다. 따라서, 기존의 결과에서 500 loop 가 도달하지 못한 것들은 무료버전이라고 보시면 됩니다. 하지만 보다 더 나아가서, transfer learning을 하게 될 시에, 기존에 사용되었던 self-supervised learning이 무엇인지에 따라서 모델의 performance가 얼마나 되는지를 비교하는 과정까지 진행해보겠습니다. 이는 단순 block을 활용하여 transfer learning에 활용되는 self-supervised learning의 부분이 얼마나 general 하게 잘 사용될 수 있을까를 반영하고자 하였습니다.

앞단의 내용의 일부는 저번에 제가 작성한 HW3의 내용과 부분적으로 동일함을 먼저 말씀드립니다. 제 자신의 글이기 때문에 “표절”에 대한 논란이 없을까 하여 먼저 언급하겠습니다.

그리고 추가적으로, transfer learning은 크게 2가지를 이루는데, 하나는 source model, 나머지 하나는 task model입니다. Source model의 경우, self-supervised 일수도 있고 supervised learning일 수도 있으나, 본 리포트는 기존 과제 3을 참고로 하기 때문에, self-supervised learning을 통해 학습된 모델임을 가정하고 서술하겠습니다.

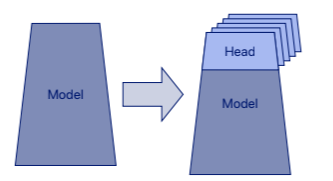
1. **Transfer learning**

기존에 label을 가지고 있는 데이터를 활용하여 모델을 학습하는 것이 일반적이였으나, 모델이 점차 복잡해짐에 따라, 혹은 더 정확한 값을 예측하려고 하는 시도에 따라서, 그에 상응하는 필요로 한 데이터 수의 양이 너무나 많아졌습니다. 이 상황속에서, label이 있는 데이터는 대단히 비용이 많이 들기 때문에, label이 없는 더 큰 데이터셋을 활용하고자 하는 시도들이 시작되었습니다. 이때에 있어 Self-supervised learning은, 기존에 label을 갖고 있던 다른 문제와는 달리, untagged data를 기반으로 하여, 모델 스스로 학습 데이터에 대한 분류문제를 수행한다고 하여, 앞 단에 “self”라는 단어가 붙어졌습니다. 즉, 실제 데이터가 어떠한 범주에 해당하는지를 모르는 과정속에서 data 특징에 따라서 각자의 다른 범주로 묶는 과정을 수행해야 하는데, 이를 좀 더 자세하게 알아보고자 합니다. Self-supervised learning 내에서도 다양한 방법론이 존재하지만, 기본적으로는 두가지 (Pre-trained model and Downstream task)를 수행합니다. 이 때에, 사전 학습된 모델 생성의 경우엔, tag가 없는 상황속에서 대량의 데이터 (언어에서는 일반적인 문장)을 활용하여 다음 문장 혹은 Masking 한 단어를 예측하는 데 활용이 됩니다. 그 뒤에 소량의 tagged data를 활용하여, 앞 단의 pre trained model을 fine tuning 하는데 활용함으로써, 모델의 정확도를 높입니다. 이는 보통 기존 layer에 몇 개의 층을 더 추가하여 특성에 맞는 목적을 수행합니다.



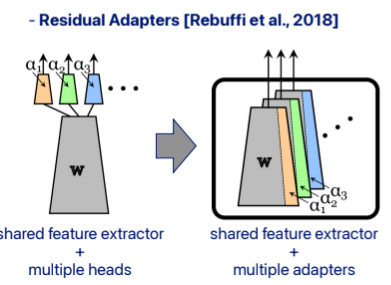
이렇게 앞 단의 모델에서, paramter를 학습한 다음엔, 그 학습된 결과를 바탕으로, 새로운 task를 하고자 하는 것이 바로 transfer learning입니다. 이 때, 기존의 pre task를 가지고 학습했던 맨 윗단의 결과부분을 수정 및 head를 추가함으로써, 새로운 task에 대한 prediction을 수행합니다. 기존과 가장 큰 차이점은, 기존엔 다른 task를 함에 따라 모델을 end-to-end 방식으로 처음엔 각 모델의 parameter를 random initialization 해서 최종 결과값을 나올 때까지 오랜 시간의 학습을 재시작해야 하는 반면, transfer learning은 기존에 학습된 parameter를 그대로 가져옴으로써, 모델의 학습 속도를 훨씬 효과적으로 늘릴 수 있습니다. 이는 역시 우리가 다른 도메인에서의 target task의 수가 현저히 적을 경우 더욱 빛을 발휘할 수 있습니다. 또한 각자의 필요함에 따라, 기존의 model에 해당하는 파라미터를 고정한 체, head 부분만 바꿀 수 있고, 모델 전체를 바꾸게 할 수 있고, 마지막으로는 모델의 일부분만 수정할 수 있는 adaptor 방법론이 있습니다. 이에 대해서 간단히 설명하고 모델의 결과값을 설명하겠습니다.

**1.2 Fix model parameter**



Model의 파라미터를 고정한 체, 다른 task에 해당하는 prediction을 하는 head 부분만 추가하여, 그 부분만 파라미터를 변경하는 경우를 의미합니다. 이는, 하나의 source model을 가지고, 여러 다른 task를 수행할 경우, memory efficiency를 증가하기 위해 사용됩니다. 만일, 여러 개의 다중 task를 수행할 때, 모델의 전체를 임의로 수정하게 될 경우, 우리는 n개의 개별적인 모델이 다 저장이 필요하기 때문에, 이를 모두 관리하게 될 시엔 많은 메모리 capacity를 차지하게 됩니다. 이를 방지하고자, model 부분의 파라미터를 고정한 체, head 부분만 바꾼다면, 실질적으론 model parameter (1개) + head (n개) 만 필요로 하게 되므로, 효율적인 관리가 가능하게 됩니다. 이를 model freeze라고 부릅니다. 하지만, 이렇게 했을 경우, performance 측면에서 많은 악영향을 주기 때문에, 이러한 trade off를 잘 반영해야 합니다.

**1.3 Adapter model**



이를 해결하기 위해서, 나온 방법이 adapter를 사용하는 겁니다. 전체 모델을 수정하게 되므로 인한 memory 문제 vs freeze로 인한 performance 성능 저하 이 두가지를 잡기 위해, 각 layer에 소수의 추가 parameter를 배치함으로써 이를 해결하고자 합니다.

즉, 각 layer에서의 일부분만 사용하기 때문에, 모델 전체를 변경하지 않게 되므로 사진과 같이 모델부분의 대다수를 freeze 하게 이어갈 수 있고, 그래도 일부분의 수정이 이루어지기 때문에, performance를 증가할 수 있습니다. 실제로 이렇게 adapter model를 하게 될 시에, 전체 메모리 측면에서는 2개 정도의 full model를 차지함으로써 여러 다중 모델을 관리할 수 있고, 모델의 성능은 상당히 높은 결과값을 이룹니다.

* 1. **Contrastive Learning**

그 중, contrastive learning은 입력 샘플 간의 “비교 (대조)”를 통해, 모델을 학습하는 방법론을 의미합니다. 따라서 위에서 언급하였듯이, 사전에 정답 데이터 (label)을 구축하지 않은 classification 모델에서, 데이터 양이 많고 비용이 들지 않는 비정형데이터를 활용하는 방법을 “입력 샘플 간의 비교”를 통해 학습하고자 하는 것을 의미합니다. 더 나아가서, 이는 representation learning을 수행하기 위한 하나의 방법론으로 볼 수 있습니다. 즉, 학습된 표현 공간 내에서 “Intra 평균은 가깝게, Inter 평균은 멀게”함으로써, 표현 상에서 각 데이터의 특성에 맞게 clustering을 하고자 합니다. 또 다른 특징으로는 contrastive learning은 다른 문제를 다루고자 fine tuning을 할 때, 모델의 아키텍쳐를 수정할 필요가 없다는 점에서 장점이 있습니다.

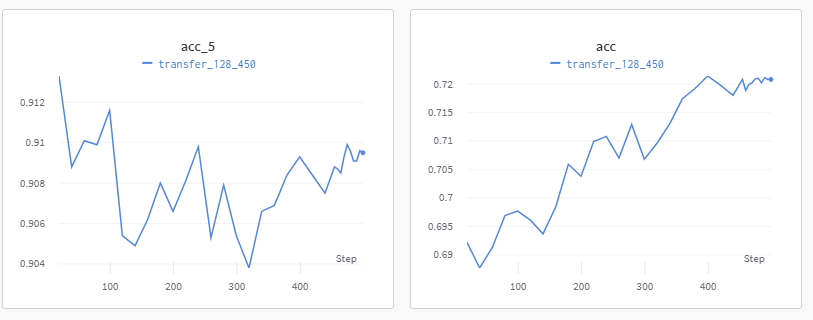
본 연구에서는 기존에 contrastive learning을 통해 학습된 ‘CIFAR10’ self-supervised learning을 source model로 활용하여 ‘CIFAR 100’ transfer learning을 한 모델과, 처음부터 end-to-end CIFAR 100 모델을 학습한 내용과의 결과값 차이를 보도록 하겠습니다.

제가 사용한 source model의 파라미터는 bs를 128로 했을 경우와 64로 했을 경우 크게 두가지를 사용하였고, 더 나아가 450 loop를 학습한 parameter와 150개의 loop만 학습한 parameter를 비교하여, 두가지 추가질문을 하고자 했습니다.

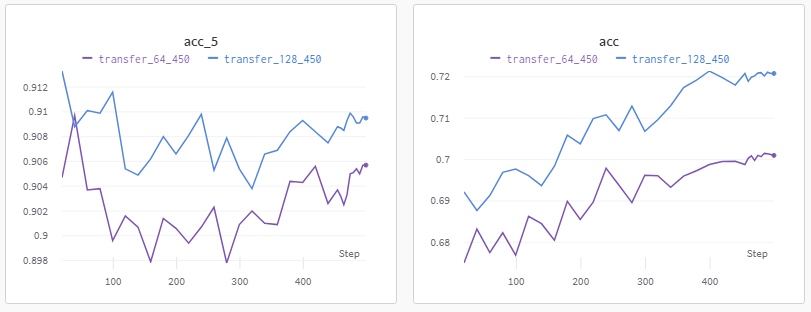
1. Negative samples의 수가 달라질수록 self-supervised learning의 효과가 좋아진다고 했는데, 이게 transfer learning에서도 이어질 수 있을까? (bs 128 vs 64)
2. Source model의 충분한 학습이 뒷 단에 transfer learning에서도 영향을 줄까? (loop 150 vs 450)

따라서, 동일한 데이터를 어떻게 활용하는지에 따라 transfer learning에 사용되는 model의 parameter에 영향을 줄 수 있고, 이가 transfer learning의 performance에도 영향을 줄 수 있습니다. 본 과제 2은 end-to-end vs transfer learning에 대해서 비교하고자 합니다. 3장에서는 negative samples와 관련된 parameter를 변경함에 따른 성능 차이를 보여주고, 4장에서는 학습한 loop에 따른 transfer learning에 성능차이를 서술하겠습니다.

**2. End to end vs transfer learning (CIFAR 100)** ****  비교한 transfer learning은 배치 사이즈를 128로 하고 450 루프동안 학습한 모델의 파라미터입니다. 모델이 보여준 성능은 확연히 차이가 났습니다. 특히 맨 오른쪽 사진인 loss 그래프를 봤을 경우, 맨 처음부터, 전이학습을 한 모델이 낮았으나, 특정 중간에 어떠한 변곡점을 기준으로 더 크게 차이가 났습니다. 전이학습을 한 경우, 대략 17번째에서 튀어 오르는 경우가 발생하였지만, 그 이후로는 계속해서 loss를 최소화하는 방향으로 이어졌습니다. 하지만, 기존에 end-to-end의 경우엔, 18번째에서 더욱 현저하게 튀어 오르고, 그 이후로 내려갔긴 했으나, loss가 특정 구간에 static 하게 유지되는 것을 볼 수 있습니다. 실제로 accuracy 부분을 봐도, 17번쨰 이후로, accuracy 와 acc\_5가 모두 0.01 정도로 이어진 것으로 보아, 학습이 그 이후로 되지 않은 것으로 보입니다. 이는 제가 모델을 잘못 세웠을 수도 있으나, 모델이 중간에 break이 되는 경우를 볼 수 있습니다.

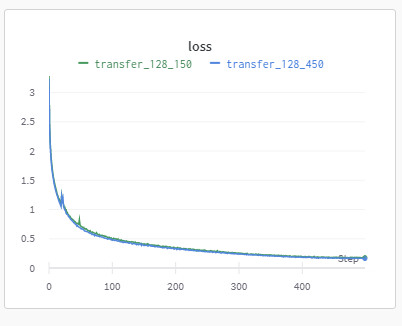


좀 더 자세하게 봤을 경우, accuracy는 전반적으로 우 상향하는 모습을 볼 수 있습니다. 종합적으로 봤을 경우, 전이학습을 통해, loss 와 accuracy가 상당부분 높은 성과를 보이고 있음을 볼 수 있어, 전이학습의 효과를 엿볼 수 있습니다.

**3. transfer learning’s batch size (128 vs 64)  
**

Batch-size에 따른 transfer learning의 효과는 있었음을 알 수 있습니다. 이는 앞서 1장에서 언급한, pre-trained and use as a block 형식을 사용에 적합한 source model block을 어떤 경우 더 잘 보여주고 있는지를 보여줌과 동시에, 3번쨰 과제에서 언급한 negative sample의 수에 따른 모델의 성능과도 연결될 수 있습니다. Loss와 accuracy 모두 128개의 batch size를 활용한 모델이 근소하게 앞서고 있습니다. 특히 두 모델 모두, 문제가 있었던 loss의 17번째 루프에서 조금의 튀어 오름이 있었으나 그 이후로 잘 잡아나가는 모습을 보이고 있고, accuracy도 우상향하는 것을 볼 수 있습니다. 다만, 앞단의 self-supervised learning에서의 negative sample이 많을수록, contrastive learning의 효과가 커진다는 내용과 일맥상통하여, 이러한 효과가 전이학습에서의 performance에서도 영향을 줬지 않았나 싶습니다.

**4. transfer learning’s trained loop (150 vs 450)**

비교에 사용한 loop는 동일하게 batch size를 128로 해 둔 상황속에서, self-supervised에서 loop가 각 150,450이였을때의 모델의 파라미터를 비교집단으로 삼았습니다. 결과가 이 역시 loop가 높으면 높을수록, accuracy 와 loss 부분에서 근소하게 나마 더 좋은 성능을 보이고 있음을 볼 수 있습니다. 허나, 엄청 유의미할 정도의 차이를 보이는지에 대해서는 의문을 갖고 있습니다. 실제 accuracy 부분이 대략 0.005 차이이면, 추가 300 루프를 학습하기 위한 시간 및 모델의 학습에 필요한 effort를 고려했을 때에 그리 효과적이진 않아 보입니다.

**Cf) 종합적인 판단**

**** ****

제가 실험해본 전체 전이학습 모델에 대한 전체 plotting 결과입니다. 전체적으로 봤을 경우, 어떤 경우가 앞도적으로 좋다 라는 등에 대한 평가를 내지 못하고 있으나, 가장 뚜렷한 점은, 기존 end-to-end 모델보다는 이러한 4가지의 모델 모두가 특징을 잘 잡고 있으며 결과값도 유의미하게 차이가 난다는 점입니다. 허나, 그 4가지 모델 사이에서는 유의미한 차이가 보이지 않습니다. 따라서, 앞 단의 source model의 성능을 최대화하고, 그것이 가장 좋은 전이학습의 성능을 볼 수 있는지를 유도하기 위해, pretrained model을 어떻게 할지에 대해서 루프 수를 높게 가져가거나, parameter수를 엄청 비교해 나가면서 할 필요성에 대해서는 잘 느끼지 못했습니다. 제가 생각하기엔 가장 중요한 전이학습의 포인트는, ‘전이학습에 활용될 source model의 최고의 performance를 내기 위해 한땀한땀 수정하기 보다는, **전체 트랜드를 따라갈 수 있도록, 전체적인 틀을 잘 만들어 주는 방법이 종합적인 판단(메모리 사용량, 모델학습에 필요한 시간 등)으로는 더욱 효과적일 수 있다.**” 라는 점입니다. 왜냐하면, 지금은 CIFAR 100에만 test해봤던 모델이지만, 앞서 1장에서처럼, 이 source model이 다양한 task에 사용될 수 있고, 각 task 마다 원하는 최적의 파라미터가 다룰 수 있기 때문에, 전체적인 방향성을 잡아주는 용도로 source model의 학습이 필요하다는 점이라고 생각합니다.