[소프트웨어학과]

영상인식을 위한 ViT, CNN 모델 관련 연구

[중간보고서]

Version 1.0
2023. 10. 26

학번: 201920811

이름: 이동현

지도교수: 유종빈

목 차

요약 3
1. 개요 4
2. CNN 실험 4
2-1. Dataset Order 4
2-2. Data Filtering 5
3. ViT 실험 7
3-1. Learnable Class Attention 7
3-2. Knowledge Distillation 개선 8
3-3. Top-5 활용 9
3-4. 부분 고정 학습 9
4. Intentional Forgetting 9
5. 이후 연구 계획 10

요약

자기주도연구 1 에 이어 최신 Vision transformer 와 Convolutional neural networks 기반의 아키텍처에 대한 심화된 이해와 실습을 통해 새로운 모델과 새로운 데이터 증강 기법을 각각 하나씩 제안해보는 것을 목표로 연구를 진행하였다. 중간 연구기간(2023.09.01 - 2023.10.12) 동안에는 최대한 많은 아이디어를 생각해내는 것에 중점을 두어 ResNet50[1]을 바탕으로 Dataset Order, Data Filtering 에 대해 실험하였고, DyTox[2] 모델을 바탕으로 Learnable ClassAttention, gumbel softmax, knowledge distillation 을 실험하였다. 7 번의 실패 끝에 현재는 DyTox 에 매 task 별로 기존 데이터와 새로운 데이터를 번갈아가며 학습시키는 Intentional Forgetting 기법을 발견하였으며 이에 대해 집중적으로 연구하고 있다.

[1] He et al., 2016, Deep Residual Learning for Image Recognition, https://arxiv.org/abs/1512.03385

[2] Arthur Douillard et al., 2021, DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion, https://arxiv.org/abs/2111.11326

1. 개요

본 연구의 목적은 자기주도연구 1 에서 여러 논문을 읽고 축적한 최신 Vision transformer 와 Convolutional neural networks 기반의 아키텍처에 대한 심화된 이해를 바탕으로, 여러가지 실습과 실험을 통해 새로운 학습의 패러다임을 제안해보는 것이다. 여기서 학습의 패러다임은 새로운 모델이 될 수도 있고, 새로운 데이터 증강기법이 될 수도 있고 새로운 학습 방식이 될 수도 있다.

중간 연구기간(2023.09.01 – 2023.10.12) 동안에는 한 가지 방식을 끝까지 고수하기 보다는 문제를 찾아내고 그것을 해결하기 위해 최대한 많은 아이디어를 생각해내고, 직접 구현해보는 것에 집중하였다. 그 결과 7 번의 실패 끝에 Intentional Forgetting 이라는 기법을 찾아냈고, 남은 연구 기간 이 기법에 대해 집중적으로 연구할 예정이다. 매주 연구 진행 과정은 Github Repository3에서 확인할 수 있다. 주기적인 지도교수 면담으로 진행상황을 파악하고 조율하였으며, 대학원생 멘토와 지속적으로 협력하여 연구를 진행하였다.

2. CNN 실험

CNN 실험들은 ResNet 모델과 CIFAR-100 Dataset[4]을 기준으로 이루어졌다.

2-1. Dataset Order

제일 처음 떠올린 개선 방안은 Dataset 의 순서를 정해주는 것이었다. 현재 대부분의 Data Loader 에서는 Data 를 Random 하게 Shuffle 된 순서로 제시한다. 그런데 개인적으로 진행한 Continual Learning 과 관련된 실험[5]에서는 하나의 Class 만을 담은 Batch 를 학습시키면 모델은 금방 그 Class 만을 예측하도록 변경된다.

그렇다면 Shuffle 되었을 때 해당 Batch 에 포함되지 않은 Class 에 대해서 순간적으로 Forgetting 이 일어날 수도 있겠다는 생각이 들었다. 또한 매 학습마다 Shuffle 을 시킨다면 학습마다 결과에 조금씩 차이가 있을 것이고, 그렇다면 최고의 결과가 나왔을 때 학습 순서를 파악하고 분석하면 학습에 도움이 되지 않을까 생각했다.

실험을 위해 Batch Size 는 Class 수의 배수 (Cifar-100 에서는 100 의 배수)로 설정하였고, 매 Batch 별로 모든 Class 가 한 번씩 나오도록 구현하였다. 다른 Hyperparameter 들은 모두 똑같이 유지하였다.

먼저 ResNet18, Batch Size 100 에서 좋은 결과[6]가 나왔다. 이에 좀 더 큰 모델인 ResNet50 에서도 잘 작동하는지 확인해보기 위해 추가적인 실험을 진행하였다. 이미지 사이즈를

[3] https://github.com/Chihiro0623/Undergraduate-Research-II

[4] https://www.cs.toronto.edu/~kriz/cifar.html

[5] https://github.com/Chihiro0623/ContinualLearning/tree/main/Experiments/experiment1/experiment1-1

[6] https://api.wandb.ai/links/oso0310/0uzvnf6t

224 와 32 사이에서 바꿔보고, Batch Normalization 에 다양한 Sample 들이 골고루 들어갈 수 있도록 epoch 마다 Class 별 등장 순서를 Shuffle 하기도 하였다. 그 결과[7], Image Size 가 32 일 때에는 성능이 저하되었으나, 224 일 때에는 약 1.7%p 의 성능 향상을 볼 수 있었다.

그런데 Baseline 의 정확도가 76.75 로 너무 낮기 때문에 좀 더 좋은 Hyperparameter 를 Baseline 으로 교체해보라는 대학원생 멘토의 조언을 듣고, Batch 100 에서 79.21 의 정확도를 내는 모델을 Baseline 으로 재설정하였다. 그런데 코드를 새로 만들던 중, 이전의 결과는 학습 데이터를 두 배로 학습하였기 때문에 좋은 결과를 내고 있었다는 점이 발견되었고, 그 점을 수정한 결과 성능은 Baseline 보다 떨어지게 되었다.

그런데 Accuracy 는 떨어지지만 Train Loss 는 더 낮은 점에 착안하여, '제안하는 방법론이 학습은 더 잘 되지만 Robustness 가 부족하여 새로운 Test Data 에 대해 예측을 잘 하지 못하는 것이 아닐까?'라는 생각을 하게 되었고, 이를 해소하기 위해 매 N Epoch 마다 Shuffle 된 Dataset 과 모든 Class 가 한 번씩 나오게 되는 Dataset 을 번갈아가면서 학습시켜 보았으나[8] 성공하지 못했다.

이어서 시작은 정렬된 Dataset 으로 학습하지만, N 번째 Epoch 부터 Shuffle 된 Dataset 으로 학습하는 것을 시도했지만[9] 성공하지 못했다. Optimizer 를 AdamW 로 바꿔보기도 했지만 큰 성과는 없었다. 계속해서 방안을 모색하던 중, Batch Size 100 을 기준으로 매 Batch 마다 Shuffle 시 100 개 중 보통 65-70 개 종류의 Class 들이 포함된다는 것에 착안하여, 인위적으로 Batch 마다 등장하는 Class 의 개수를 N 개로 제한하여 보았지만[10] 실패하였다.

2-2. Data Filtering

우리는 물체를 인식할 때, 빛의 삼원색에 따라 인지하는 것이 아닌 종합하여 인지하게 된다. 따라서 이미지의 세 Layer(R,G,B)를 모델에 들어오는 순간 Convolution 으로 종합하는 방식을 생각해보았는데, 결과는 좋지 않았다.

이어서 이미지 사이즈를 크게 할 수록 성능이 좋아진다는 이미 알려진 사실[11]을 응용해, Linear Layer 를 통해 스스로 입력된 이미지보다 크게 만들어 학습한다면 (예를 들어 32x32 의 입력이 들어왔을 때, 224x224 로 스스로 만들어 학습한다면) 더 성능이 개선될 수 있지 않을까 생각했다.

---

[7] https://api.wandb.ai/links/oso0310/u6s45ncx

[8] https://api.wandb.ai/links/oso0310/g9hmwhgo

[9] https://api.wandb.ai/links/oso0310/4wgpq95b

[10] https://api.wandb.ai/links/oso0310/uztqgvbc

[11] Vajira Thambawita et al., 2021, Impact of Image Resolution on Deep Learning Performance in Endoscopy Image Classification: An Experimental Study Using a Large Dataset of Endoscopic Images, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8700246/

이에는 이미지의 사이즈가 커질 뿐만 아니라, 모델 자체가 Class 별 원형을 학습하는 효과 또한 있을 것이라 예상했다.

그러다 이미지에 직접 만든 필터를 씌우고 정해진 Epoch 마다 그 필터 부분만을 초기화시켜 스스로 이미지를 증강시키면 좋겠다는 생각을 하게 되었다. 더 나아가 이미지를 처음 모델에 넣은 뒤 나온 예측 결과와 입력을 바탕으로 필터를 만들면 해당 Class 의 원형을 더 잘 학습시킬 수 있다는 생각이 들어 적용시켜 보았고, 결과는 아래 그림 1 과 같다. Accuracy 결과는 현재 로그가 소실되어 확인할 수 없지만, Baseline 보다 좋은 결과를 기록한 적은 없었다.

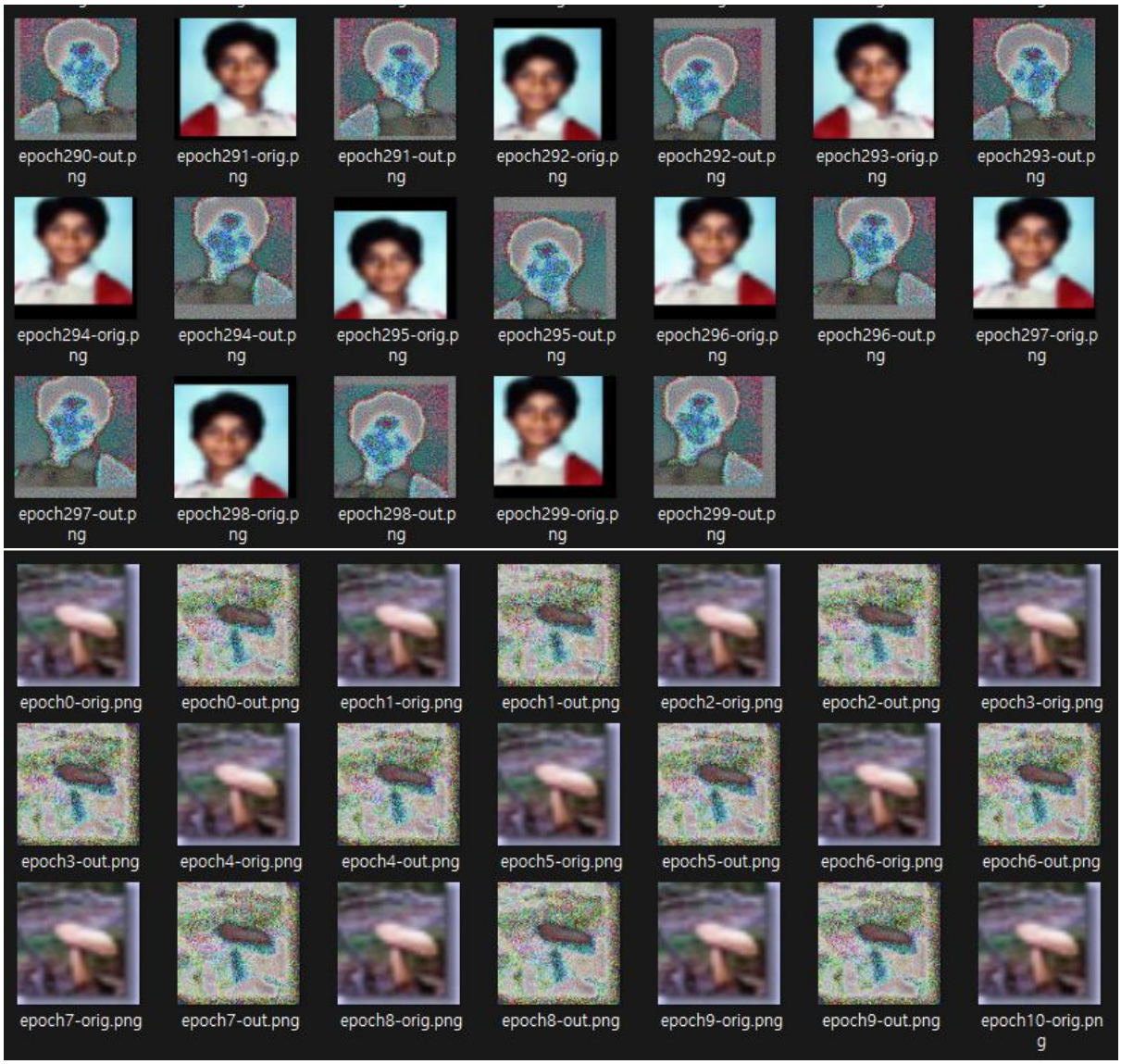

그림 1. Epoch 별 원본 데이터와 필터를 씌운 데이터의 스크린샷. 위 그림은 Train Data 중 무작위로 고른 Class 이고, 아래 그림은 Test Data 중 무작위로 고른 Class 이다. 두 그림 모두에서 모델이 물체의 윤곽을 효과적으로 파악해내고 있음을 확인할 수 있다.

그림을 보면 모델이 중요한 부분을 제외한 다른 부분들은 배경으로 처리하여 무시하는 것처럼 보인다. 그러나 모델은 중요한 부분과 배경을 따로 지시 받아 설정하는 것이 아니기 때문에, 그림

2 와 같은 경우에는 중요한 부분을 한 번 배경으로 오인하여 제거하기 시작하자 계속해서 중요한 부분을 무시하게 되어 결국 잘 학습하지 못하는 것을 확인하였다. 이것이 현재 CNN 모델 학습법의 근본적인 문제일 수도 있겠다는 생각이 들었다.

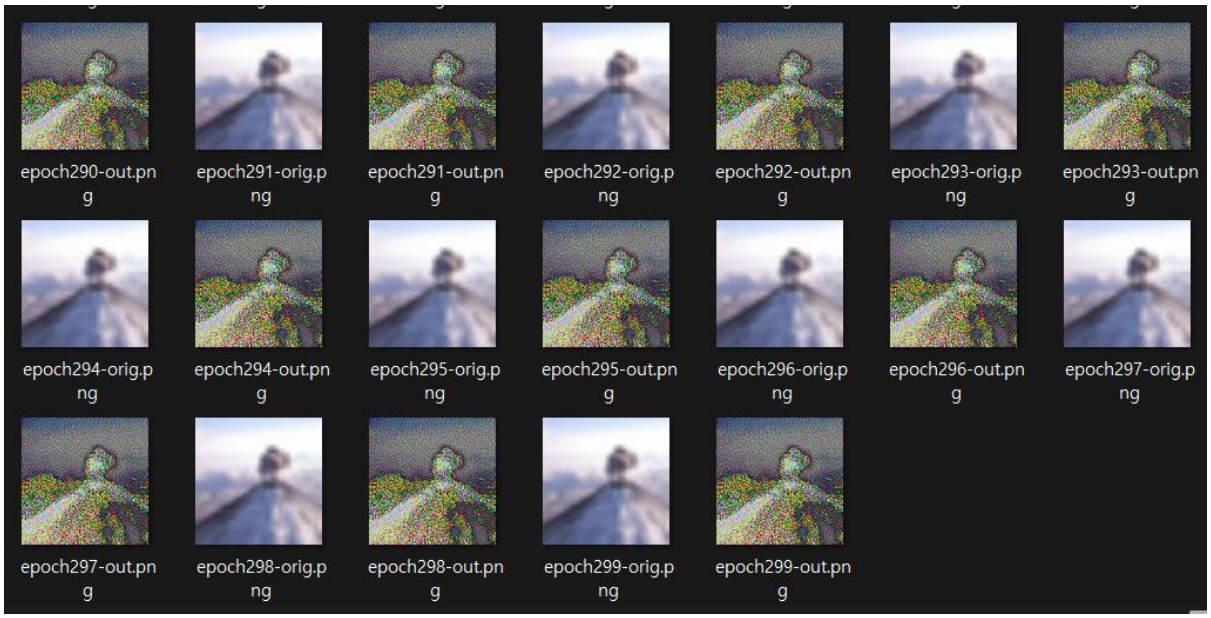

그림 2. Epoch 별 원본 데이터와 필터를 씌운 데이터의 스크린샷. Test Data 중 무작위로 고른 Class 이다. 주목해야 할 목표가 아닌 오른쪽 아래에 집중하는 것을 볼 수 있다.

3. ViT 실험

계속해서 실패를 경험하던 중, 담당교수님으로부터 ViT 을 활용한 Continual Learning 에서 Task 별로 새로운 Token 을 추가하면 적은 비용으로도 성능을 개선할 수 있을지도 모른다는 조언을 듣고, 이에 대한 연구를 시작하게 되었다. 그런데 선행 연구를 탐색하던 중 DyTox 라는 유사한 모델이 이미 있다는 것을 발견하였고, 따라서 DyTox 를 Base 로 하여 성능을 개선하는 연구를 진행하기로 선회하였다.

학습 방식은 Class Incremental Learning 으로, Cifar-100 Dataset 을 바탕으로 Task 별로 10 개의 Class 씩 증강하는 10 Steps, 5 개의 Class 씩 증강하는 20 Steps, 2 개의 Class 씩 증강하는 50 Steps 로 나뉘어져 있다.

3-1. Learnable Class Attention

Task 가 지나 새로운 학습 데이터들이 들어오더라도 기존의 데이터들에 대해서는 중요하게 여기는 부분들이 계속해서 유지된다면 Continual Learning 의 문제인 Catastrophic Forgetting 을 방지할 수 있을 것이라고 생각했다. Class Attention 을 시각화한 결과는 그림 3 과 같고, Task 가 지날수록 중요하게 보는 부분이 옅어진다는 것을 알 수 있다.

이를 개선하기 위한 방법으로 가장 간단하게 Attention 값에 2 를 곱하는 방식으로 값에 할당되는 가중치를 두 배로 향상시켜 보았다. 그러자 Average 기준 10 Steps 에서는 1.5%p 라는 큰 성능 향상이 있었으며, 50 Steps 에서도 소소한 성능 향상을 보였다.

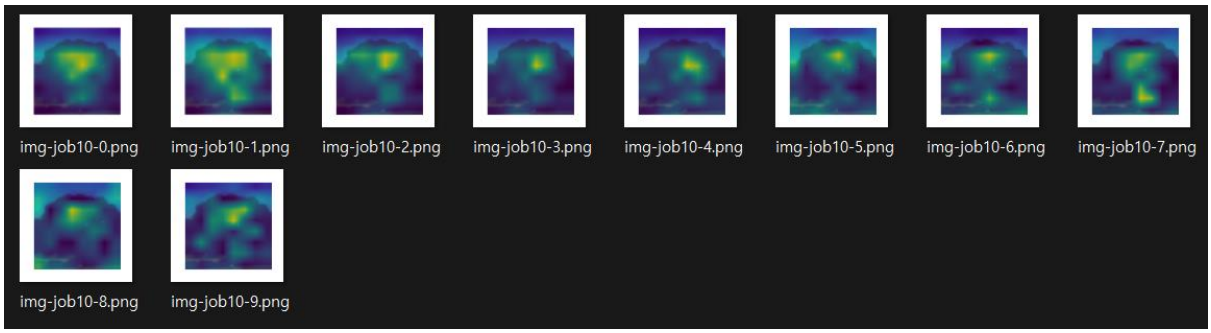

그림 3. DyTox 의 10 Steps 에서 Task 별로 ClassAttention 을 시각화한 스크린샷. Task 가 지날수록 중요하게 보는 부분(노란색)이 옅어지고 있다.

그러나 20 Steps 에서는 큰 성능 향상을 보이지 않았기 때문에 20 Steps 의 성능을 향상시키기 위한 개선법에 대해 생각하던 중, 단순히 2 를 곱하는 것이 아닌 Learnable Parameter 를 곱하는 Learnable Class Attention 방식을 생각해보았다. 그러나 이 방식 또한 성능의 개선을 보이지 않았고, 오히려 10 Steps 의 성능도 저하시켰다. 결과는 표 1 및 WandB Report[12]에서 확인할 수 있다.

CIFAR	10 steps (10-10)			20 steps (5-5)			50 steps (2-2)		
	#P	Avg	Last	#P	Avg	Last	#P	Avg	Last
DyTox	10.73	71.50	57.76	10.74	68.86	51.47	10.77	64.82	45.61
재현		71.47	58.01		68.72	50.51		64.76	44.31
Attn*2		73.00	59.67		68.87	52.24		64.94	44.36
Temperature		71.33	57.78		68.61	51.53			
Temperature2					68.68	50.52			
Temperature4					68.49	50.82			

표 1. Learnable Class Attention 의 Step 별 결과. 첫 번째 행은 DyTox 논문에서의 결과, 두 번째 행은 자체 재현 결과, 세 번째 행은 Attention 에 2 배를 곱한 결과, 네 번째에서 여섯 번째 행은 Learnable Class Attention 을 각각 1, 2, 4 로 초기화한 결과를 나타낸다. 각 Step 별 열은 왼쪽부터 파라미터의 수, 평균 정확도, 마지막 Task 정확도를 나타낸 것이다. 파라미터의 수는 매 시행이 논문의 적힌 값과 같기 때문에 표기를 생략했다.

3-2. Knowledge Distillation 개선

DyTox 에서 Knowledge Distillation 을 위해 사용하는 Teacher Model 이 이전 Task 까지의 학습이 완료된 Model 이었다. 그러나 각 Task 에서 사용하는 학습 데이터에서 이전 Task 까지의 데이터가 차지하는 비중은 몹시 작기 때문에 Teacher Model 은 대부분이 자신도 잘 모르는 새로운 데이터에 대해 Teacher 역할을 수행하고 있었다는 모순적인 상황을 확인할 수 있었다. 그러나 Knowledge Distillation 을 아예 사용하지 않는 것보다는 훨씬 좋은 성능을 보여주었지만, 이 부분을 더욱 개선시킬 수 있을 것이라는 생각이 들었다.

[12] https://api.wandb.ai/links/oso0310/s9lp7w0x

각 Task 만을 학습하는 경우 대개 80 대 후반에서 90 대 초반 정도의 높은 정확도를 보여준다. 이를 활용해 새로운 Task 만 학습한 전문적인 Teacher 모델을 만들어 새롭게 들어오는 데이터에는 새로운 Teacher 가, 옛날 데이터에는 옛날 Teacher 가 각각 Knowledge Distillation 을 수행하는 방식을 구현해보았으나 성능이 좋지 않았다. 추가적으로 각 Task 만을 학습한 모델들을 만들어 각 Weight 들을 더한 후 학습을 시작해보았으나 좋은 결과가 나오지 않았다. 이 실험들의 결과는 WandB Report[13]에서 확인할 수 있다.

3-3. Top-5 활용

Top-5 정확도는 항상 90% 이상의 높은 성능을 보인다. 따라서 이를 활용하여 Top-1 의 성능을 높일 수 있지 않을까 생각했다. Top-5 정확도가 90%라는 것은 5 개 안에 정답이 있을 확률이 90%라는 뜻이기 때문이다. 모델이 Top-5 만을 정교하게 보게 하기 위해서 Loss Function 에 들어가기 전 예측 값에서 상위 5 개를 제외한 모든 값들을 전부 최하위 값으로 만들어 보았다. 해당 실험 결과는 WandB Report[14]에서 확인할 수 있다.

3-4. 부분 고정 학습

DyTox 의 Finetuning 과정에서는 이미지를 Token 으로 변환하는 부분인 Self-Attention Blocks (SAB)만을 Freeze 시킨 뒤 Token 으로 Class 를 예측하는 부분인 Task-Attention Blocks (TAB)와 Classifier Head 만을 학습시키고 있다. Finetuning 에 사용되는 데이터는 지금까지 있었던 모든 Task 의 데이터들 중 각 Class 의 평균에 가까운 1000 장을 Class 별로 균등한 수로 뽑아 사용하고 있다.

DyTox 학습 시 항상 Finetuning 과정에서 성능이 많이 향상되는 것을 확인할 수 있다. 따라서 이 Finetuning 과정을 응용하여 Finetuning 이 아닌 본래 학습 때 SAB, TAB, Head 를 일정 Epoch 마다 Freeze 시켜 학습시킨다면 각 모듈이 각자에 맞게 잘 학습할 것이라고 생각했다. 해당 실험 결과는 WandB Report[15]에서 확인할 수 있다.

4. Intentional Forgetting

위 3-2 의 결과에서 10 개의 Class 만을 학습시키더라도 90%의 정확도를 넘지 못하는 몇 Task 를 확인할 수 있었다. 그래서 DyTox 를 학습시키는 동안 Class 별 결과를 확인해보니, 어떤 Class 는 자신이 처음 도입된 Task 에서 얼마 지나지 않았는데도 나쁜 결과를 내는 경우도 있고 어떤 Class 는 자신이 처음 도입된 Task 에서 많이 지났음에도 높은 결과를 내는 경우도 있었다.

---

[13] https://api.wandb.ai/links/oso0310/kgtzsczr

[14] https://api.wandb.ai/links/oso0310/848pli0j

[15] https://api.wandb.ai/links/oso0310/k0zog2nn

이를 개선하기 위해 여러가지 방안을 생각해 보았다. 가장 먼저 떠오른 것은 Class Incremental Learning 에서 재현용으로 사용되는 Memory 의 구성을 개선하는 것이다. 그러다 문득 에빙하우스의 망각곡선에 따르면 '7 번 잊고 난 후 외우는 것은 긴 시간동안 잊지 않는다는 것'이 떠올랐다. 실험으로 이를 구현해보기 위해 현재 Task 의 데이터만을 포함한 데이터셋 A 와 이전 Task 들의 데이터도 함께 포함한 데이터셋 B 를 번갈아가면서 학습시켜 보았다.

그 결과, A 로 학습시키는 경우 이전 Task 의 Class 들은 잊혀지게 되어 정확도는 떨어지지만, 그 후 B 로 학습시키는 경우 A 로 향상된 성능이 거의 유지되면서 이전 Task 의 Class 들의 성능이 오르게 되어 결국 전체적인 성능도 오르는 것을 확인할 수 있었다. 실험 결과는 아래 표 3 및 WandB Report[16]에서 확인할 수 있다. 10 Steps 과 20 Steps 에서는 좋은 개선을 보였는데, 이는 해당 방법론이 추가적인 메모리와 시간을 요구하지 않는다는 점을 고려하면 꽤나 큰 개선으로 보인다.

그러나 50 Steps 에서는 딱히 좋은 성능을 보여주지 않고 있기 때문에, 50 Steps 에서도 성능을 개선시킬 수 있도록 Epoch 주기를 Tuning 하는 등의 방식을 통해 Forgetting 방법론을 개선시켜야 할 것 같다.

CIFAR	10 steps (10-10)			20 steps (5-5)			50 steps (2-2)		
	#P	Avg	Last	#P	Avg	Last	#P	Avg	Last
DyTox	10.73	71.50	57.76	10.74	68.86	51.47	10.77	64.82	45.61
재현		71.47	58.01		68.72	50.51		64.76	44.31
exp		72.81	60.47		69.79	52.55		64.72	43.36

표 2. Intentional Forgetting 의 Step 별 결과. 첫 번째 행은 DyTox 논문에서의 결과, 두 번째 행은 자체 재현 결과, 세 번째 행은 Intentional Forgetting 의 결과를 나타낸다. 각 Step 별 열은 왼쪽부터 파라미터의 수, 평균 정확도, 마지막 Task 정확도를 나타낸 것이다. 파라미터의 수는 매 시행이 논문의 적힌 값과 같기 때문에 표기를 생략했다.

5. 이후 연구 계획

지금까지의 연구는 관찰을 통해 문제를 찾아내고 그 문제를 해결하기 위한 방안들을 찾기 위해 최대한 다양한 아이디어들을 Brainstorming 하고 직접 구현해보는 방식으로 CNN 의 개선 방안을 찾기 위한 연구 과정을 단련하기 위해 노력하였다. 그 결과, 여러 번의 실패 끝에 Intentional Forgetting 이라는 하나의 가능성을 찾아냈다.

남은 연구 기간 동안에는 Intentional Forgetting 을 심화적으로 연구하여, Intentional Forgetting 이 어떻게 성능의 개선을 가져오는지, CNN 을 포함한 다른 모델들 및 Class Incremental Learning 외의 다른 방식의 학습에서도 성능의 개선을 가져오는지 등을 파악하고 방법론을 일반화 및 최적화 할 예정이다.

[16] https://api.wandb.ai/links/oso0310/bly07y98