Deep Learning Standalone

–

for Chemistry

1. ML Basic
2. Pytorch Basic
3. MLP with Fingerprint Representation
4. CNN with SMILES Representation
5. GNN with Graph Representation
6. Experiment Management and Hyperparameter Tuning with Tensorboard
7. Practical Tips

Practical Tips

1. Baseline을 꼭 먼저 돌려볼 것
→ 구현체가 있다면 보지만 말고 꼭 직접 실행시켜서 보고된 결과가 재현되는지 확인 해볼 것
→ Baseline 하나가 아니라 최대한 다양한 모델들을 미리 실험해볼 것
→ 비교할 만한 Baseline이 없을 경우 ML 알고리즘들을 Baseline 삼을 것

2. Dataset을 준비할 때
→ 각 전처리 과정이 원하는대로 작동하는지 꼭 확인해볼 것
→ 전처리를 미리 진행할지, Dataset 오브젝트를 생성할때 진행할지, 매 iteration마다 진행할지 결정하기 (speed, hard-disk, memory 타협)
→ Data loader의 num_worker 사용해서 multi-processing 할 것

Practical Tips

3. Model Construction
→ 중복 사용될 부분은 module로 빼기
→ Module을 parametric하게 생성할 때는 꼭 nn.ModuleList()에 넣기 (그렇지 않으면 학습에서 제외됨)
→ 처음부터 끝까지 다 짜지 말고, input과 가까운 레이어부터 하나씩 통과시키며 원하는 대로 작동하는지 확인할 것 (나중에 한번에 디버깅하려면 매우 힘듦)
→ 내부 알고리즘에서 for문 사용을 자제할 것 (최대한 행렬 연산으로 대체)
→ Dropout, BN, Activation 순서를 주의해서 사용할 것 (Dropout을 input이나 output layer에 놓으면 안됨) 일반적으로 FC→BN→Act→Dropout 순서
→ 모델 아키텍쳐가 args로부터 인자를 받는 형태로 변수화시킬 것 (실험 관리에 용이)

Practical Tips

4. Training Loop
→ 모델과 X, y 모두 같은 device에 올라가 있는지 확인할 것
→ model.train()과 model.eval()을 train, val, test시 알맞게 꼭 호출
→ optimizer.zero_grad() → loss.backward() → optimizer.step() 순서 헷갈리지 않기
→ iteration별, 혹은 epoch별 Loss 꼭 남기기 (학습 상태 모니터링 위해서)

5. Val, Test Loop
→ with torch.no_grad()를 쓰면 gradient가 남지 않으므로 처리 속도 다소 향상
→ validation set에 대해서 꼭 loss 남기기
→ Test는 매 epoch마다 하는 것이 추후 비교에 용이. 데이터가 너무 크다면 iteration에 따른 로깅도 괜찮음

Practical Tips

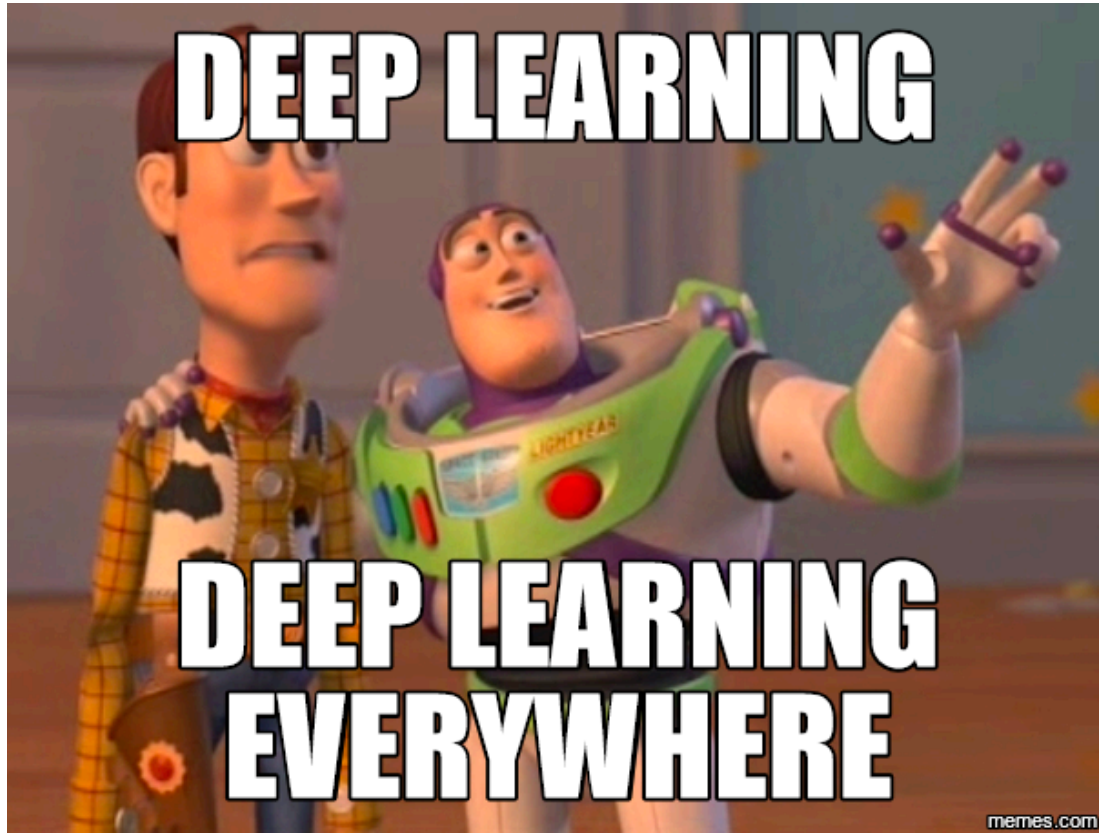
6. Tensorboard
→ 꼭 tensorboard를 활용할 것 (print시킨 loss는 학습이 이루어진다 확인 정도 수준)
→ loss, metric을 tag를 이용해서 group하여 기록하면 보기에 편함
→ hparam 기능도 꼭 사용하여 hyperparameter에 따른 metric 변화 살피기
→ 그 외에도 image나 text 로깅이 모두 가능하니 적절하게 활용!
→ 기회가 된다면 parameter histogram도 활용 추천
→ 만약 train loss대비 val loss가 너무 줄어들지 않는다면 regularization 활용을 검토
→ 처음에는 학습에 영향을 많이 미치는 Model Capacity와 적절한 Lr 값을 찾는데 집중
→ Batch Size의 영향도 확인해볼 것
→ 어느정도 안정화 된 뒤에는 세부적인 rate 및 option들을 튜닝할 것 (l2 weight decay가 너무 큰 경우 학습이 일찍 제한될 수 있음)
→ 일반적으로 hyperparameter tuning에는 한계가 있음. 근본적인 아키텍쳐나 학습방법의 개선을 꾀해볼 것 (loss 그래프가 떡잎부터 달라짐)

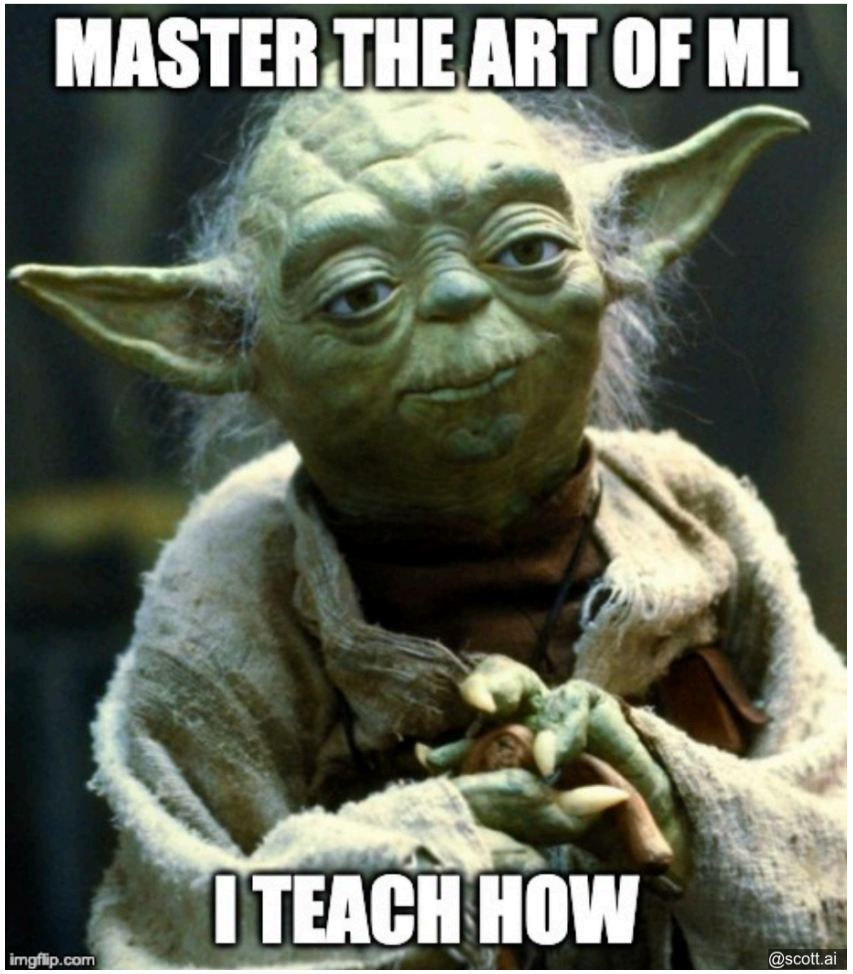
Practical Tips

7. Optimizer and LR Scheduler
→ 요즘에는 다양한 Learning rate scheduler가 사용됨 (cosine, warm-up, ranger, look-ahead) 이들을 활용해서 local-minimum을 벗어나도록 시도
→ Ranger 같은 optimizer는 초기 lr에 robust한 효과를 줌

8. 영혼까지 끌어모아 성능 향상을 짜내야 할 때
→ Ensemble 기법을 활용할 것 (완전히 다른 모델 or seed만 다르게 한 학습 모델)
→ TTA(test time augumentation)도 고려해볼 것

9. 코드를 최적화할 것
→ 불필요한 중복 연산, 알고리즘 효율성을 개선하여 단위 시간당 처리되는 샘플 수를 어떻게해서든 높일 것 (같은 시간 동안 더 많은 실험을 할 수 있음)
→ 새로운 기능을 추가할 때는 git branch를 파서 실험 후 성공적일 때 merge할 것


DEEP LEARNING
DEEP LEARNING EVERYWHERE
memes.com


MASTER THE ART OF ML
I TEACH HOW
imgflip.com
@scott.ai

dingbro

Idea Builder Armed with Software Technology

조재영 Kevin Jo
CEO | Dingbro, Inc

Email. kevin.jo@dingbro.ai
Phone. 010-4102-5358

김승수 Andrew Kim
CTO | Dingbro, Inc

Email. andrew.kim@dingbro.ai
Phone. 010-9196-0940

KAIST Startup Studio W8 2F
291, Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea