안녕하세요 전공프로젝트 발표를 하게 된 4조 이번생에 공모전은 처음이라의 정원지입니다.

저희는 Computer Vision 이상치 탐지 알고리즘을 주제로 선정하게 되었습니다.

우선 목차를 설명 드리겠습니다.

팀 소개, Computer Vision 이상치 탐지 알고리즘을 주제로 선정하게 된 이유, 수행결과를 보여드리기에 앞서 사용한 몇가지 기법들 설명, 프로젝트 결과, 프로젝트를 하며 있었던 문제점들, 그리고 느낀점 순서로 진행하겠습니다.

저희 조의 이름은 이번생에 공모전은 처음이라 입니다.

저희 3명 모두 공모전을 살면서 처음 해봐서 지은 팀명이며 프로젝트의 방향을 공모전으로 정한 이유기도 합니다.

1차 프로젝트와 달리 2차 프로젝트에서는 기능별로 역할분담을 하여 해오기가 힘든 구조였기에 매일 정규 강의시간 끝나기 전에 꼭 회의를 하였습니다.

그날 서로 새로 찾은 기법들을 발표, 설명해주고 어떤게 더 좋을지 토론하며 모델을 정했으며 작성한 코드들을 같이 보며 리뷰하고 수정 하여 프로젝트를 진행했습니다.

저희 조가 Computer Vison 이상치 탐지 알고리즘을 주제로 선정하게 된 이유를 설명드리겠습니다.

저희 조는 자유주제로 할까, 공모전을 참여해볼까 하다가 3명 모두 공모전에 참여해본 경험이 없는걸 알았고 이번 기회에 공모전에 한번 참여해보자! 하여 공모전 주제를 탐색하던 중 저희가 배웠던 내용을 가장 잘 활용할 수 있는 주제라고 생각해 선택하게 되었습니다.

그래서 dacon에서 주최하는 Computer Vison 이상치 탐지 알고리즘 공모전에 참가하게 되었습니다.

저희가 참여한 이 공모전은 15종류의 사물이 존재하며 사물의 상태에 따라 이미지가 분류되어 있습니다.

이 대회에서는 사물의 종류를 분류하고, 정상 샘플과 비정상(이상치) 샘플을 분류해야 했습니다.

우선 수행결과를 설명드리기에 앞서 저희가 사용한 기법들을 먼저 설명드리겠습니다.

저희는 Regnet 모델을 사용했고 그중에서도 regnet160을 사용했습니다.

옵티마이저는 Lamb를 사용했습니다.

그리고 Early Stopping 기법을 사용했습니다.

자세하게 설명 드리겠습니다.

저희는 처음에 모델을 efficientnetb3(에피션트넷 비3)를 사용하였습니다.

Efficientnet은 강의시간에서도 배웠다 싶이 한정된 자원으로 최대의 효율을 내기 위한 방법으로 모델 스케일링을 시스템적으로 분석하여 더 나은 성능을 얻고자 하는 모델입니다.

하지만 efficientnet은 depth, 네트워크의 깊이가 증가할수록 모델의 용량이 커지고 더 복잡한 특징을 잡아낼수는 있지만 vanishing gradient(소실 경사)의 문제로 학습시키기가 어려웠습니다.

그래서 저희가 찾다가 알게 된 것이 20년 3월에 나왔다는 Regnet(레그넷) 모델입니다.

efficientnet처럼 하나의 베스트 모델을 만드는 것이 중점이 아니라 뛰어난 모델을 어떤 세팅에서도 사용할 수 있도록 디자인 스페이스를 디자인 하는 것이 중심인 모델입니다.

또한 디자인 스페이스의 quantized linear function(양자화 선형 함수)를 통해 더욱 간단하게 스페이스를 정의 할수 있게 되었고 네트워크 구조를 잘 이해할수 있게 되었습니다.

무엇보다 Regnet은 efficientnet 모델보다 gpu에서 속도가 5배 빠름을 증명하였습니다.

그리고 저희는 옵티마이저를 Lamb를 사용하였습니다.

현재 가장 많이 쓰이고 무슨 옵티마이저를 쓸지 모르겠다면 아담을 써라 라는 말도 있는만큼 저희도 처음엔 아담을 썼습니다.

하지만 찾아보니 Lamb(램) 옵티마이저가 adam의 확장버전이며 정확도를 더 높여줄수 있다 하여 램을 사용하게 되었습니다.

학습결과도 아담보다 램이 정확도가 더 높게 나왔었습니다.

원래 large-batch는 몇번 업데이트를 못하니까 큰 learning rate를 가져야 하지만 learning rate가 너무 크면 수렴을 하지 않게 되고 그러면 최대로 늘릴 수 있는 배치 크기가 제한 될 수 밖에 없습니다. 몇몇의 레이어에서 gradient \* learning rate크기가 실제 weight의 크기보다 커지는 불안정한 모습을 확인할수 있었습니다.

하지만 lamb은 large-batch를 안정적으로 학습시켜주므로 동일한 512 batch size에서도 약 1%가량의 성능이 향상됨을 보였습니다.

LAMB (Layer-wise Adaptive Moments optimizer for Batch training)

그리고 Early Stopping 기법을 사용하였습니다.

얼리 스타핑은 모델이 과적합 되기 전 훈련을 멈추는 정규화 기법입니다.

저희는 batch size가 아닌 epoch을 단위로 성능 검증을 하였습니다.

성능의 일시적 변동이 아닌 지속적인 정체 또는 하락에 의한 판단으로 얼리 스타핑의 기준을 정하였으며, patience는 50으로 지정하였습니다.

이제 수행결과를 설명드리겠습니다.

수행결과를 설명드리기에 앞서 저희는 작물 병해 분류 ai 경진대회에서 1위하신 분의 코드를 참조했습니다.

저희는 파이토치를 이용하여 모델링 하였습니다.

처음엔 텐서플로로 구현하려 하였으나 여러가지 방법들을 찾던 중에 딥 러닝 프레임워크인 파이토치를 알게 되었습니다.

파이토치는 익히기 쉽고 간결하며 구현이 빠르게 되며 비교적 빠른 최적화가 가능합니다.

그래프를 만들면서 동시에 값을 할당하는 denine by run 방식으로 코드를 깔끔하게 작성할수 있습니다.

파이토치 단점으로는 학습에 필요한 예제를 구하기 쉽지 않다가 있는데, 저희는 주최측의 공유코드들이 전부 파이토치를 사용하고 있기에 예제 구하기에는 무리가 없다고 판단하였습니다.

이러한 이유들로 저희는 프레임워크를 파이토치로 선택했습니다.

저희는 데이터 전처리에서 이미지 resizing과 이미지 증식을 해주었습니다.

먼저 이미지 리사이징 부분을 보면 원본 이미지 사이즈보다 작은 256,256으로 리사이즈 하여 데이터를 새롭게 저장했습니다.

512, 512 사이즈로 시도 해봤었는데 시간이 너무 오래걸려 저희의 컴퓨터로는 시간을 감당할수 없어 256,256 사이즈로 했습니다.

그리고 exist\_ok=true로 설정을 넣어줘야 exception에러가 발생하지 않습니다.

보간법으로는 주로 이미지 축소시 많이 사용하는 cv2.inter\_atrea를 적용했습니다.

그다음은 이미지가 들어있는 train 데이터들을 증식 시켜주었습니다.

저희의 train 데이터 수가 적어서 데이터 불균형이 일어나기 때문에 랜덤으로 좌우반전, 상하반전, 원근법 조절 등을 해주어 이미지를 증식 시켜주었습니다.

마지막으로는 각 채널에 맞출수 있도록 정규화도 시켜주었습니다.

이 이미지들은 저희의 train 데이터들입니다.

데이터 전처리를 하기 전 원본 이미지 데이터입니다.

방금 말씀드린 이미지 리사이징과 증식을 해주면

이렇게 됩니다.

데이터 전처리가 끝난 후의 이미지 데이터입니다.

트레이닝 부분에선 best epoch을 저장해주어 early stopping에 사용할수 있도록 해주었습니다.

그리고 k-fold 기법도 사용하였는데 저희는 그 중에서도 StratifiedKFold를 사용했습니다.

k-fold와의 차이가 있는데 k-fold는 train, validation셋을 나누는 과정이 랜덤하게 나눠지므로 데이터가 불균형한 경우에는 train, validation셋의 label의 비율이 어긋나게 됩니다.

저희의 데이터는 불균형하다고 dacon측에서 미리 사전에 고지를 해두었기에 좋은 방안이 뭐가 있을까 찾다가 stratifiedKfold를 선택했습니다.

stratifiedKfold는 계층화된 k-fold교차 검증기이며 target에 속성값의 개수를 동일하게 가져감으로써 kfold 같이 데이터가 한곳으로 몰리는것을 방지해주며 값의 분포를 반영해주지 못하는 문제를 해결해 주었습니다.

또한 저희는 ensemble 기법도 사용하였습니다.

Ensemble 기법은 단어의미와 같이 조화 하는것입니다.

여러 모델의 결과를 모아 최종 결과를 내어 성능을 향상시키는 기법입니다.

Ensemble 기법의 특징으로는 대부분 ensemble 사용시 정확도가 높아지며 각 모델별 특징이 두드러지는 즉 over fitting이 잘 되는 모델을 기본적으로 사용합니다.

저희는 pytorch로 dataloader를 생성하여 여러 개의 모델들을 생성하고 각각 학습시킨후 세가지 모델 모두 앙상블로 결합하였습니다.

이렇게 하면 모델이 인스턴스화 되고 가장 좋은 가중치가 로드되게 됩니다.

그 다음 평균을 내주는 걸로 ensemble 하였습니다.

그래서 저희의 수행결과의 최종 정확도는 0.833이 나왔습니다.

다음으로, 저희가 프로젝트를 진행하며 느꼈던 어려웠던 점이나 아쉬웠던점들이 있습니다.

첫번째로는, 수도라벨링을 기법을 사용해보지 못한것입니다.

수도라벨링은 지도학습을 통해 1차적으로 학습된 모델을 이용하여, 태깅이 되지 않은 데이터에 대해 예측을 수행하는 기법입니다. 수행된 예측 결과를 통해 가짜(Pseudo)로 태깅(labeling)한다고 하여 수도라벨링이라고 부릅니다.

수도라벨링 기법을 쓸 때 test dataset을 사용해야 하는데 공모전이기 때문에 사전학습모델은 사용이 불가하여 사용해보지 못하였습니다.

공모전이 끝나고 최종코드에 따로 적용을 해보고 싶었으나 시간이 부족하여 못해봤지만 중간에 수도라벨링 기법을 써보았을 때 기존코드보다 수도라벨링기법을 적용한 코드가 정확도가 약 0.3정도 올라가는 것을 봤었습니다.

아쉬움이 많이 남아 다음 프로젝트때는 수도라벨링을 꼭 써보고 싶습니다.

두번째로는, pytorch가 tensorflow보다 장점도 많고 공모전 공유코드에도 대부분 pytorch로 진행을 하길래 pytorch를 선택하였지만 처음 써보는 것이라 모르는 기능들이 많아 공부하고 적용하는데 어려움이 있었습니다.

Cuda 또한 pytorch와 버전호환이 안되고 리눅스 ubuntu를 이용해야 하는 등 해결해야하는 어려움들이 많았습니다.

세번째로는, 장비의 사양이 저희 셋다 평범한 컴퓨터 수준이라서 학습시간이 오래 걸렸습니다.

그래서 다른 여러가지 경우들을 다 못보고 프로젝트를 끝내야만 했던 것이 아쉽습니다.

마지막으로 저희 팀원들의 느낀점입니다.

김선경님은 수업과 비슷하다고 생각했지만 실제로 코드를 짜는 것 자체가 너무 힘들었고 부족한 점을 확연하게 느낄 수 있었습니다.

심혜주님은 tensorflow가 아닌 pytorch 코드를 오류없이 구현하고 수정하는 과정이었기에 정규시간이 끝난 후에도 하루종일 붙잡고 있을만큼 시간도 많이 쏟았을 뿐만 아니라 어려워서 힘들었지만, 틈틈이 방법을 검색하고 조사하며 해결해나가면서 뿌듯함을 느꼈다.

그리고 그 과정에서 몰랐던 딥러닝 모델과 기법에 대해서도 많이 배울 수 있었고, 코드를 직접 실행하고 수정하면서 어떻게 학습이 이루어지는지 전반적인 흐름을 확인할 수 있었던 좋은 기회라고 생각했다. 또한 강의 내용의 복습과 구글링의 중요성을 다시 한 번 깨달았고 장비(그래픽카드)의 사양의 중요성도 깨달았다.

정원지님은 여러 코드를 참고하여 코드를 작성하는데 환경,경로, 파라미터 등 알맞게 변경하는 과정에서 어려움을 느꼈습니다. 하지만 수많은 구글링, 팀원들과의 적극적인 소통, 브레인스토밍 등으로 해결해나갔고 그 과정에서 뿌듯함을 느꼈습니다.

개인적으로 pseudo-labeling 기법을 사용해보지 못한것이 매우 아쉽지만 다음에 꼭 사용해보고 싶습니다. pseudo-labeling처럼 강의 때 배운것 이외의 새로운 이론, 모델, 기법등을 공부하며 많이 알게 되었기에 뜻 깊은 시간이었다고 느꼈습니다.

마지막으로 질문 있으시면 질문 듣고 끝내도록 하겠습니다

(질문)

이상으로 4조의 발표를 마치도록 하겠습니다. 감사합니다.