<인트로>

1. 위성 영상을 활용한 북극 해빙 예측 발표를 맡게된 김현태입니다.

<목차>

1. 저희는 데이콘, 극지연구소에서 주최하는 북극 해빙 예측 대회에 참가해보고 이를 아마존 클라우드 서비스를 이용하여 웹으로 배포하였습니다. 이 주제를 선정하게 된 배경을 소개하고 저희가 사용한 모델들에 대한 간략한 설명과 구현한 코드를 바탕으로 결과를 보여드리겠습니다. 마지막으로 저희가 느꼈던 부족했던 점과 보완하고 싶었던 점을 알려드리는 순서로 발표를 진행하겠습니다.

<프로젝트 배경>

1. 전 세계적으로 탄소 배출을 줄이기 위한 노력이 눈에 띄게 보이고 있습니다. 실제로 블랙록이라는 세계 최대 규모의 자산 운용사는 ESG라는 기준을 내세우며 환경을 생각하지 않는 기업에 과감히 투자를 하지 않겠다는 의견을 밝혔습니다. 2015년 파리 협정을 통해 대부분의 국가들이 2050년까지 탄소 중립 사회로의 전환 목표를 밝히며 기후 변화에 대한 대응을 하고 있는 추세입니다. 하지만 지구온난화가 직접적으로 우리에게 와 닿지 않는 것이 현실입니다. 그래서 저희는 이번 대회에 참가하여 실제로 근 50년간 빙하의 감소 추세를 확인하고 이를 예측해보는 경험을 통해 심각성을 확인해보고자 이 주제를 선정했습니다.

<프로젝트 팀 구성 및 역할>

프로젝트 팀 구성 및 각자 맡은 역할에 대해 소개하겠습니다.

1. 저희는 2인 1조의 비전공생으로 이번 교육을 통해 데이터 사이언스를 처음 배웠습니다. 기술 구현에 있어서 부족한 부분은 Inflearn 강의와 Youtube 생활코딩, 노마드코더 강의등을 통해 배워서 보충했습니다. 딥러닝 모델은 CNN, VGGNet, RESNet, RNN 총 4가지를 Google Colab을 활용하여 구현해보았고, 이를 Html 템플릿을 활용하여 페이지로 제작했습니다. 서버는 아마존 클라우드 EC2 서비스를 활용하여 Linux Ubuntu OS에서 Apache로 구현하였고 저희가 진행하는 과정은 모두 Github을 활용하여 업데이트하고 공유하며 마크다운 파일로 기록했습니다.

<프로젝트 수행절차 및 방법-CNN,RNN,VGGNet,RESNet>

1. 저희가 처리해야할 데이터는 이미지 파일이기 때문에 딥러닝 모델을 이용해서 학습시켰습니다. 가장 대표적인 CNN을 바탕으로 개선된 모델인 VGGNet 과 RESNet 그리고 시계열 데이터 분석에 특화된 RNN을 사용하여 총 4개의 모델을 비교해봤습니다.
   1. 처음으로 CNN이란 간단히 말하자면 그림의 전체부분을 보는 것이 아니라 부분을 확인해서 우리가 원하는 값을 찾아내는 것입니다. 이 사진을 보고 새인지 구분하려면 부리를 확인하면 되는데 이때 그림을 전부다 확인하는 것보다 부리부분만 확인하면 수월히 새를 구분할 수 있습니다.
   2. CNN의 전체적인 구조입니다. 이미지를 입력하면 convolution 과정과 pooling 과정을 두번 거치고 flatten 과정과 denser 과정을 통해 원하는 값을 도출해낼 수 있습니다. 각 과정별로 어떤 일이 수행되는지 알아보겠습니다.
   3. Convolution 이란 이미지를 값으로 변환하는 연산을 의미합니다. 이미지를 입력하면 이를 초록색 행렬처럼 수치화하고 노란색 필터값으로 연산과정을 거치게 됩니다. 동일한 3x3행렬의 위치 값끼리 곱한다음 더한 값을 결과값으로 갖게됩니다. 최종 결과는 분홍행렬처럼 바뀌게 됩니다.
   4. Convolution이 마치고 나면 Pooling 과정이 진행되는데 이는 output의 개수가 너무 많아져서 생기는 문제를 방지해 줍니다. 연관성이 낮은 부분을 제거하여 dimension을 축소시킵니다.
   5. Flatten 과정을 통해 받아온 tensor을 1차원 벡터로 변형 시켜줍니다. 이미 pooling layer을 통해서 이미지에서 얻어온 데이터는 특이점이 되어 무관합니다.
   6. 이제 마지막으로 Fully Connected Layer을 통해서 최종 결과물을 출력하게됩니다.
2. 다음으로 소개해드릴 모델은 RNN입니다.
   1. RNN은 CNN보다 시계열 데이터 처리하는데 특화된 모델입니다. 위 그림으로 보면 빨간 노드는 입력값, 노란 노드는 기억하는 역할, 파란 노드는 출력값을 의미하는데 각 입력값에 대한 기억값들이 이후에 이뤄지는 입력에 영향을 미치고 출력결과에 영향을 미칩니다. 입력과 출력값의 시퀀스의 길이에 관계없이 받아들일 수 있는 구조라 유연하게 다룰 수 있습니다.
   2. 다음 그림 처럼 RNN의 기본적인 구조는 CNN과 같습니다. 왼쪽 두번째 그림을 보면 사진을 입력값으로 주었을 때 단어들이 나오게 할 수 있고, 이는 그림속에 새가 있을 때 그림안에 새가 있어요 라는 문장으로 바꿀 수 있음을 뜻합니다. 그 다음은 여러 개의 시퀀스를 통해 값을 도출하는 구조입니다. 예를 들어 문장을 줬을 때 이를 통해 감수성을 판단하여 점수로 변환하는 형태를 말합니다. 마지막 구조는 여러 개의 시퀀스를 입력했을 때 시퀀스로 응답을 하는 흔히 번역기에 사용하는 구조인데 영어로 된 문장을 줬을 때 한글로 변형하는 구조입니다.
   3. RNN에는 장기 의존성의 문제가 있습니다. 예를 들어 I grew up in France…. I speak fluent French라는 문장이 있고 French를 빈칸으로 주고 예측해보라고 했을 때 France라는 정보를 활용하면 쉽게 찾아낼 수 있는데 문장이 멀리 떨어져있을 경우 문제가 발생합니다. 이러한 점을 보완하면서 나온 모델이 LSTM 입니다.
   4. LTSM은 위에 컨베이어 벨트처럼 흐르는 큰 줄기가 있고 안에 있는 각 변수들의 설정을 통해 이러한 문제점을 해결하여 RNN보다 높은 성능을 발휘하고 있어서 보편적인 상황에서 좋은 결과를 가져다줍니다.
3. 다음으로 소개해드릴 VGGNet과 RESNet은 CNN의 신경망 layer 개수를 늘려서 성능을 향상시킨 모델들입니다.
   1. VGGNet 먼저 보면 convolution 과 pooling layer 가 훨씬 많아진 것을 확인할 수 있습니다. 각 convolution은 3x3 kernel을 사용하고 fully connected layer 3번을 진행하면서 파라미터 개수가 급격히 늘어나서 대략 1억개 가량 됩니다.
   2. VGGNet은 학습시에 깊은 layer의 파라미터 초기화 시에 기존에 학습된 결과를 사용하므로 vanishing gradient 문제를 해결합니다.Vanishing gradient 문제는 신경망이 깊어질수록 그래디언트가 점점 작아져 결국 가중치 매개변수가 업데이트 되지 않는 경우를 말합니다.
4. 다음 보여드릴 모델은 RESNet입니다. VGGNet은 14년도 ILSVRC에서 준우승한 모델이고 RESNet은 15년도에 우승한 모델입니다.
   1. RESNet의 가장 큰 특징은 Shortcut Connection을 이용해서 그림처럼 입력값에 출력값을 더해줄 수 있도록 길을 만들어주는 것이다. 이를 통해서 깊은 신경망의 문제점인 vanishing gradient와 exploding gradient는 반대로 그래디언트가 점점 커져서 가중치 매개변수가 기하급수적으로 커지게 되어 발산하므로 두 경우 학습이 제대로 이루어지지 않는다. 를 해결할 수 있게 되었다.
   2. RESNet은 이러한 shortcut connection을 추가하여 이전 값과의 잔차를 계산하여 이를 최소화하는 모델이다.

<프로젝트 수행절차 및 방법 -데이터 전처리>

1. 다음은 저희가 참여한 대회에서 제공하는 데이터에 대해 살펴보고 전처리 과정과 제출해야하는 파일 형태에 대해 간략히 소개하겠습니다.
2. 저희가 사용가능한 데이터는 주최측으로 제공받은
3. 1978년부터 2018년까지 월별로 482개의 데이터 입니다. 그림에서 보이듯 순서대로 해빙농도, 북극점, 해안선 마스크, 육지마스크, 결측값 5개의 체널로 구성된 데이터였습니다.
4. 제출해야하는 submission 파일은 총 136,192의 열에는 0부터 250사이의 농도 값이 있는 형태로 제출해야했습니다.
5. 저희는 파일을 불러와 오름차순으로 정렬하고 북극점의 값을 해빙농도와 합친형태로 총 4개의 체널을 가진 이미지 리스트로 변환했습니다.
6. 저희가 제출해야하는 결과값은 19년도와 20년도 해빙농도를 예측해야하므로 24개월 주기를 갖고 학습을 시켜주었고 최종 결과를 확인할 수 있는 데이터로 48개를 X\_test와 y\_test 배열안에 넣어둔뒤 이를 제외한 나머지로 학습을 시켰습니다.

<프로젝트 수행결과>

저희가 구현한 모델들의 성능에 대해서

1. 각 모델별로 학습시킨 결과입니다. 가장 왼쪽에 있는 그림이 원본 사진이고 오른쪽 4 그림이 각 모델별로 학습시킨 결과입니다. 주최측에서 평가 지표로 활용한 Mean Absolute Error은 평균절대오차로 일반적인 회귀 모델 성능을 측정하는 지표로 활용됩니다.
2. RNN 결과값이 제일 좋게 나왔습니다. 아무래도 저희가 예측하려고 하는 데이터가 시계열 데이터를 바탕으로 진행되다 보니 LSTM을 활용한 RNN모델에서 가장 좋은 성적을 거둘 수 있었던 것 같습니다.
3. 저희는 구현한 모델과 공부한 내용을 웹페이지를 통해 배포하기 위해 간단히 AWS를 활용해서 Linux 아파치 서버를 구축했습니다. Windows 대신 Linux를 활용한 이유는 보편적인 웹 서버에서 주류로 이용하고 있어서 구현해보았습니다.
4. Apahce 서버로 구동을 했는데 기본적으로 아파치는 스레드/프로세스 기반으로 클라이언트 요청 하나당 스레드 하나가 처리하는 구조입니다. 저희는 default setting인 preFork방식을 사용했는데 worker MPM 방식으로 각 프로세스가 최대 64개의 스레드처리를 통해 동시 접속자가 많은 사이트에 적합한 방식도 있습니다.
5. Apahce와 비슷한 수준으로 많이 사용되는 Nginx도 있는데 이는 비동기 방식으로 순서의 영향을 받지 않고 event driven 처리 기반 구조를 갖고 있습니다. 클라이언트의 요청을 event handler를 통해 받아서 먼저 처리되는 것부터 진행됩니다.
6. 저희가 구현한 사이트이고 아래 링크는 깃헙주소입니다. 깃헙을 활용해서 코드 관리를 하고 파일을 공유했습니다.

<한계점,보완점>

딥러닝은 데이터의 개수가 많으면 많을수록 성능이 좋아집니다. 실제로 얼마나 데이터가 많아야하는지에 대한 연구에서 확인할 수 있듯이, 대략 25000개 이상의 데이터가 있을 때 성능이 어느정도 궤도에 올라오게 되는데 저희는 480개 밖에 없기 때문에 너무 모자랐습니다. 이를 보완하기 위해 데이터 증강을 해야합니다. 방법으로는 clustering으로 북극을 지역별로 나눠서 학습시키는 시도라든지 개인적인 생각으로는 월별로 나뉘어진 데이터를 더 잘게 쪼개서 주별로 학습시키게되면 데이터가 훨씬 늘어나서 좋은 모델이 학습될 것 같습니다.

이상 발표를 마치도록 하겠습니다.