딥러닝이론및응용 기말 Project

학번: 20215124

이름: 김수현

**# 서약**

아래 보고서는 **본인의 힘만으로 작성**해야 하며, 다른 학생에게 질문과 다른 학생의 코드를 참고하는 행위는 모두 금지합니다

\* 수업에서 제공한 코드, 노트북은 모두 재활용 가능하며, 카피로 규정하지 않습니다.

\* 수업 자료 이외에 참고자료가 있다면, 출처와 사용 부분에 모두 표시하는 경우는 모두 합당한 자료로 인정하겠습니다.

\* 위에 대해서 모두 이해하고 동의했다면, 아래 `서약글`에 다음을 작성해주세요:

"본인은 위 서약글을 이해하고 동의하며, 프로젝트를 수행하는데 있어서 반칙을 할 경우 (제공자 포함) 본 프로젝트에 대한 점수가 반영되지 않는다는 것에 동의합니다."

학번: 20215124

이름: 김수현

서약글: 본인은 위 서약글을 이해하고 동의하며, 프로젝트를 수행하는데 있어서 부정행위를 할 경우(제공자 포함)본 프로젝트에 대한 점수가 반영되지 않는다는 것에 동의합니다.

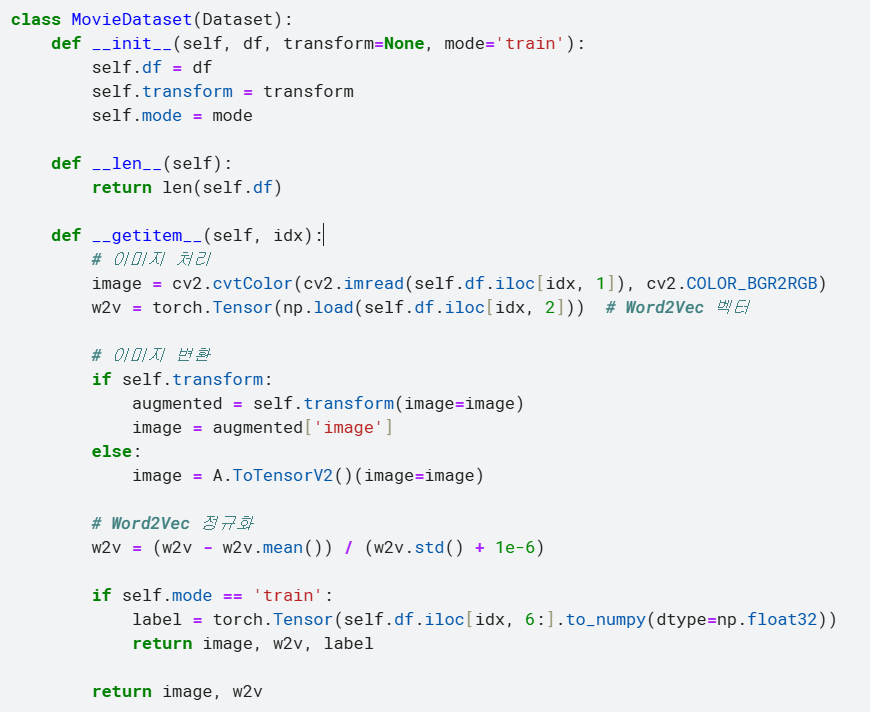
\*모든 코드에는 주석을 작성해 주세요

최종 제출시, 본 보고서와 .ipynb 노트북파일, test에 사용한 모델(.pt)파일을 압축해 제출해 주세요.

중요: 사용한 기법은 자신이 이해한 것 만을 사용하세요. 설명하지 않은 기법을 사용하면 그 부분을 제외하고 채점하겠습니다. 예를 들어서 자신의 힘으로 찾은 코드를 이용하려 하는 경우 내용을 이해하고 보고서에 이해한 내용이 충분히 설명이 되어야만 사용을 허용합니다.

Step 1: Dataset 준비하기

<코드 캡쳐 첨부>



**MovieDataset 클래스**

1. 이미지를 CNN으로 처리하고 Word2Vec 백터를 추가하여 학습을 수행합니다.
2. 학습 과정에서 transform을 통해 데이터의 다양성을 증가하여 모델의 일반화 성능을 높입니다.
3. Word2Vec 벡터를 정규화하여 계산을 안정적으로 도와줍니다.
4. Mode 파라미터를 사용하여 레이블 반환 여부를 제어합니다.

MovieDataset 클래스에 대한 코드는 기본 제공된 코드에서 **Word2Vec 벡터 정규화** 코드만 추가하였습니다.

1. 벡터의 **평균(mean)을 0**으로, **표준편차(std)를 1**로 만드는 **Z-score 정규화**를 수행합니다.
2. **1e-6**: 계산의 안정성을 위해 추가된 작은 값입니다.

Step 2: Dataset에 대한 Data Loaders 구성

<코드 캡쳐 첨부>

데이터 전처리에 대한 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**train\_transform: 학습용 데이터 증강 (Data Augmentation)**

위 코드는 최종적으로 사용한 전처리 방법입니다.



이미지 크기를 224x224 크기로 조정하여 입력 크기와 일치하도록 맞췄습니다.



데이터 다양성을 높여 모델의 일반화 성능을 향상시키기 위해 사용하였습니다.



수평 뒤집기를 적용했을 때 성능이 올라가는 것을 볼 수 있었습니다.



이미지의 위치, 크기, 각도를 조정하면 데이터 다양성을 높일 수 있다고 생각하여 적용하였습니다.



평균과 표준 편차를 사용하여 이미지를 정규화하였습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

일부 영역을 무작위로 제거하여 특정 픽셀이나 영역에 의존하는 것을 방지하기 위하여 사용하였습니다. 일부 영역에 결함이 있는 경우에도 학습이 진행되는 것을 볼 수 있었습니다. 실제로 이 변환을 사용하기 전과 후의 성능은 **7%**의 차이를 보여주었습니다.

텍스트, 폰트, 라인, 화이트이(가) 표시된 사진

자동 생성된 설명

**test\_transform: 테스트 데이터**

위 코드는 테스트용 변환으로 원본 데이터 보존이 중요하므로 과도한 증강을 피해야 합니다. 따라서 크기 조정, 정규화와 텐서 변환만 수행하였습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

성능 개선에 영향을 주는 batch\_size는 32, 64, 128, 256 순서로 진행했습니다. 진행한 결과, 미세한 차이였지만 128로 설정하였을 때 성능이 좋게 나오는 것을 발견할 수 있었습니다. 따라 최종적인 batch\_size는 128로 설정하여 학습을 진행하였습니다.

Step 3: Neural Network 생성

- Pretrained model을 허용하지 않습니다. (직접 모델을 설계해 주세요)

<코드 캡쳐 첨부>

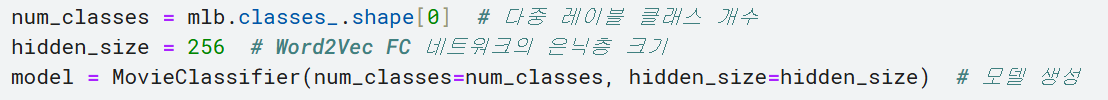
텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명



설계한 모델을 출력 후 네트워크를 구성한 방법과 이유를 각 단계별로 설명

Ex) 차원 분석, 채널 수, 커널 크기, linear layer neuron 수

activation function과 선정 이유

최종 layer에서 사용한 activation function이 무엇인지 왜 사용하였는지

CNN 모델이 무엇을 입력 받고 출력하나요? 등…

<서술형>

**차원 분석**

1. **self.cnn**

|  |  |  |
| --- | --- | --- |
|  | **Input** | **output** |
| **Conv1** | (3, 224, 224) | (64, 224, 224) |
| **MaxPool1** | (64, 224, 224) | (64, 112, 112) |
| **Conv2** | (64, 112, 112) | (128, 112, 112) |
| **MaxPool2** | (128, 112, 112) | (128, 56, 56) |
| **Conv3** | (128, 56, 56) | (256, 56, 56) |
| **Maxpool3** | (256, 56, 56) | (256, 28, 28) |
| **Conv4** | (256, 28, 28) | (128, 28, 28) |
| **Maxpool4** | (128, 28, 28) | (128, 14, 14) |
| **Flatten(평탄화)** | (128, 14, 14) | (25088,) |

1. **self.w2v\_fc**

|  |  |  |
| --- | --- | --- |
|  | **Input** | **output** |
| **Linear1** | (300) | (512) |
| **ReLU** | (512) | (512) |
| **Dropout** | (512) | (512) |
| **Linear2** | (512) | (256) |
| **ReLU** | (256) | (256) |
| **Dropout** | (256) | (256) |

1. **self.fc**

CNN 출력 (25088) + Word2Vec 출력 (256) = 26344

|  |  |  |
| --- | --- | --- |
|  | **Input** | **output** |
| **Linear1** | (25344) | (1024) |
| **ReLU** | (1024) | (1024) |
| **Dropout** | (1024) | (1024) |
| **Linear2** | (1024) | (512) |
| **ReLU** | (512) | (512) |
| **Dropout** | (512) | (512) |
| **Linear3** | (512) | (num\_classes) |

먼저 CNN 부분입니다. 입력 이미지에서 공간적 특징을 학습하고 추출합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

왼쪽과 같이 코드를 작성하지 않고 오른쪽의 코드를 사용한 이유는 더 깊은 네트워크는 복잡한 패턴을 학습할 수 있었기 때문입니다. 왼쪽의 레이어가 적기 때문에 학습 속도가 빠르긴 하지만 깊이가 얕아 복잡한 특징은 제대로 학습하지 못하였기에 오른쪽의 코드를 수행하였습니다.

CNN 모델에서는 이미지 데이터를 입력으로 받습니다. 이미지의 공간적 구조를 학습하기 위해 Convolution과 MaxPooling을 반복합니다. 4개의 Convolution Layer와 MaxPooling을 사용하여 feature map을 줄여나갔습니다. 또한 각 단계마다 Batch Normalization과 ReLU 활성화 함수를 적용하여 학습을 안정화하였습니다.

CNN의 최종 출력을 평탄화(flatten)하여 Linear layer에 입력할 수 있도록 하였습니다.

저는 이미지 데이터를 중심으로 학습하며 텍스트 정보(Word2Vec)는 보조적인 역할을 할 수 있는 방법을 생각해보았습니다. 따라서 이미지의 공간적 특징을 학습하는 CNN 구조에 더 큰 비중을 두었고 사전 학습된 Word2Vec를 사용함으로써 텍스트 의미를 빠르게 모델에 반영하고 모델의 초점을 이미지 학습에 두었습니다. 아래는 Fully Connected Network으로 이미지 특징을 결합하고 최종 분류를 수행한 단계입니다.

Word2Vec FC 이 네트워크는 텍스트 데이터를 수치화합니다. 하지만 300차원의 Word2vec 벡터는 그대로 사용하면 너무 단순합니다. 따라 FC(Fully Connected) 네트워크를 통과시켜 더 유용한 정보를 추출하고 압축시킵니다. 최종적으로 256차원의 벡터가 CNN에서 나온 이미지 feature와 결합할 준비를 합니다. 중간 layer에 ReLu와 Dropout(0.5)을 적용해 overfitting(과적합)을 방지합니다.

결합 FC 부분입니다. CNN 출력과 Word2Vec 출력을 결합하여 처리합니다. 이미지 feature와 텍스트 feature를 결합하면 더 많은 정보를 학습할 수 있습니다. Flatten(평탄화)와 결합을 통해 Fully Connected 네트워크에서 효율적으로 최종 예측을 수행하였습니다.

**Activation function**

ReLu를 사용한 이유는 양수 영역에서 기울기가 1이기 때문에 기울기 소실 문제가 완화됩니다. 깊은 신경망에서도 효과적으로 학습할 수 있습니다. 또한, 단순한 연산으로 계산이 매우 빨라 학습 속도를 크게 향상시킬 수 있습니다. 아래는 MovieClassifier에서 ReLU를 사용한 위치와 이유입니다.

1. CNN : 이미지의 특징을 추출하는 과정에서 ReLU를 적용합니다.

합성곱 계층의 결과에 비선형성을 추가하여 다양한 특징을 학습할 수 있습니다.

1. Word2Vec FC

입력 벡터를 비선형 변환하여 더 추상화된 표현을 학습할 수 있습니다.

1. 결합 FC

CNN 출력과 Word2Vec 출력을 결합한 후 비선형 변환을 통해 복잡한 관계를 학습할 수 있습니다.

최종 레이어에는 활성화 함수가 존재하지 않습니다. 그 이유는 **BCEWithLogitsLoss** 손실 함수를 사용하였기 때문입니다. (step 4) 따라 내부적으로 sigmoid 함수를 적용하여 확률 값을 계산합니다. 따라서 최종 출력 레이어는 logits 형태로 반환됩니다.

**MovieClassifier 모델 정의 & 초기화**

* **num\_classes**

mlb.classes\_.shape[0]을 통해 다중 레이블 이진화된 클래스 수를 가져옵니다.

* **hidden\_size**

모델 내부에서 사용할 Fully Connected Layer의 숨겨진 노드 크기를 나타냅니다.

제 모델에서는 256을 설정하여 적당한 크기의 특징 표현을 학습하도록 하였습니다.

* **model 초기화**

num\_classes와 hidden\_size를 인자로 전달하여 모델이 올바르게 동작하도록 설정하였습니다.

Step 4: Cost (Loss) Function과 Optimizer 선택

<코드 캡쳐 첨부>



* **BCEWithLogitsLoss**:

Binary Cross-Entropy Loss와 Sigmoid 함수를 함께 계산하는 손실 함수입니다.

* **Adam (Adaptive Moment Estimation)**:

학습 속도를 조절하고 모델의 가중치들을 업데이트하여 손실 함수 값을 최소화합니다.

Optimizer와 Cost 함수를 선정한 이유와 선정하는데 중요하다고 생각하는 내용을 모두 작성합니다.

<서술형>

Cost 함수로 **BCEWithLogitsLoss**를 선정한 이유는 다중 레이블 분류 문제에 적합하다고 생각이 들었기 때문에 선정하였습니다. BCEWithLogitsLoss는 각 클래스별로 독립적인 확률을 계산하고 Binary Cross-Entropy를 적용하기 때문에 다중 레이블 분류에 적합하다고 생각이 들었습니다.

Optimizer에서 **Adam**을 선정한 이유는 대부분의 최적화 문제에 대해 좋은 성능을 보여주기 때문에 선정하였습니다. 초기 학습률 설정에 대한 민감도를 줄여주고 weight\_decay=1e-4 라는 가중치의 크기를 추가해 과적합(overfitting)을 방지해 줄 수 있기에 선정하였습니다.

Step 5: 구성한 모델에 대한 Train and Validate 진행

<코드 캡쳐 첨부>

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

*  입력 데이터와 레이블을 GPU 장치로 이동시켰습니다.
* 

이전 배치의 기울기의 초기화를 통해 이전 기울기가 영향을 주지 않도록 합니다.

* 텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

  자동 생성된 설명

출력(outputs)은 logit 값이며 Sigmoid 활성화 함수가 적용되기 전입니다.

* 텍스트, 폰트, 화이트, 라인이(가) 표시된 사진

  자동 생성된 설명

BCEWithLogitsLoss를 사용해 손실을 계산합니다.

**train\_df**는 모델 학습을 수행하면서 각 batch 단위로 손실과 정확도를 계산하고, 최적화를 통해 모델 파라미터를 업데이트합니다.

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

* 

검증 과정에서는 기울기를 계산하거나 파라미터를 업데이트할 필요가 없습니다. 따라 비활성화합니다. 그렇게 된다면 메모리 사용량이 줄어들고 계산 속도가 빨라질 수 있습니다.

* 텍스트, 폰트, 화이트, 라인이(가) 표시된 사진

  자동 생성된 설명

모델에 입력 데이터를 전달하여 예측 결과를 얻습니다. 출력값 outputs은 logit 형태이고 각 클래스의 확률과 비슷한 값입니다.

* 텍스트, 폰트, 화이트, 라인이(가) 표시된 사진

  자동 생성된 설명

BCEWithLogitsLoss를 사용해 손실을 계산합니다.

**val\_df**는 검증 데이터셋을 사용하여 평균 손실과 정확도를 계산합니다. model.eval()과 torch.no\_grad()를 통하여 평가 모드에서 기울기 계산을 비활성화합니다. 반환된 평균 손실과 정확도는 모델의 검증 성능을 평가기에 중요합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

학습(train\_df)과 검증(val\_df) 함수를 반복 호출하며 손실(loss)과 정확도(accuracy)를 계산하여 모니터링합니다.

Step 6: CNN model training/validation 분석

<코드 캡쳐 첨부>

















































































위에서 수행한 training + validation 과정을 설명하세요

training loss와 validation loss 그래프를 통해서 분석

Ex) hyper-parameter, model을 변경하면서 성능 개선한 과정을 최대한 설명하세요

overfitting, underfitting 분석 등..

<서술형>

텍스트, 그래프, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

**그래프 분석**

1. Training Loss (파란색)

학습 과정에서 train loss는 지속적으로 감소하고 있습니다. 모델이 훈련 데이터셋에 최적화되고 있습니다.

1. Validation Loss (주황색)

초반 Epoch에는 Validation Loss가 Train Loss 같이 감소하면서 모델이 일반화되고 있다는 것을 볼 수 있었습니다.

중반 Epoch에는 Validation Loss가 감소되지 않으면서 Train Loss와의 차이가 벌어지는 것을 볼 수 있었습니다.

후반 Epoch에는 Train Loss는 계속해서 감소하지만 Validation Loss는 더 이상 감소하지 않고 상승하다 머물렀다 반복하는 것을 볼 수 있었습니다. 이는 모델이 훈련 데이터에 과적합(Overfitting) 되고 있다는 것을 의미합니다.

**hyper-parameter 및 model 변경을 통한 개선 과정**

1. Regularization

weight\_decay = 1e-4 를 적용하였지만 미세한 변화만 있었을 뿐 확연하게 개선되었다는 점은 발견하지 못하였습니다. 좀 더 높게 설정하여 과적합을 완화할 예정입니다.

1. 데이터 증강 (Data Augmentation)

CoareDropout의 기법을 추가하여 모델의 일반화 성능을 높일 수 있었습니다.

1. Batch Size

Batch Size를 조정하여 학습 안정성에 영향을 줄 수 있었습니다.

Step 7: Predict with Test Data

<코드 캡쳐 첨부>

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명 ( batch\_size = 64, num\_epochs = 20 )

Word2Vec 벡터를 사용하기 전의 Confusion matrix 분석입니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명 ( batch\_size = 64, num\_epochs = 20 )

Word2Vec 벡터를 사용하여 CNN과 결합한 후의 Confusion matrix 분석입니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명 ( batch\_size = 128, num\_epochs = 40 )

batch\_size와 num\_epochs을 증가시킨 후의 Confusion matrix 분석입니다.

텍스트, 스크린샷, 도표, 다채로움이(가) 표시된 사진

자동 생성된 설명 ( batch\_size = 128, num\_epochs = 40 )

그래프에서 epoch이 증가할 때 Overfitting(과적합)이 일어나는 것을 확인하여 **weight\_decay=1e-4**의 코드를 추가한 이후의 Confusion matrix 분석입니다. 과적합이 해결되지 않았지만 성능은 조금 오른 결과를 볼 수 있었습니다. 다른 방안이 있다면 같이 적용하여 과적합을 방지하고자 하려고 합니다.

Step 8: Training Techniques

성능 개선을 위해서 사용한 기법 중에서 특별히 효과적이었던 부분이나 강조하고자 하는 내용을 작성해주세요.

**1. 학습용 데이터 증강 (Data Augmentation)**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 (전)

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명 (후)

RGBShift 대신 CoarseDropout을 사용한 결과 모델의 성능이 향상했습니다.

**2. Word2Vec 결합**

Word2Vec 벡터를 CNN 기반의 이미지 feature와 결합한 결과 모델의 성능이 향상되었습니다.