LLAMA Classifier

Code Structure, Pooling, Feature Engineering

Changes

- Embedding Layer Forward 방식 변경

  기존: Llama의 hidden state에서 첫 번째 토큰만 사용 → 수정: Pooling 방식으로 변경.

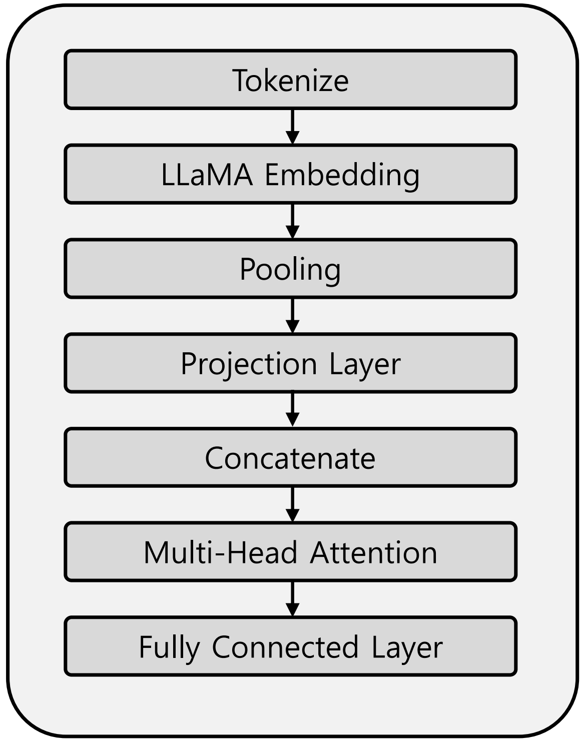
- Fully Connected Layer(fc_layer) 간소화

  fc3 삭제 → **복잡성 최소화** : (1, 448) → fc1(64) → output(2)로 단순화

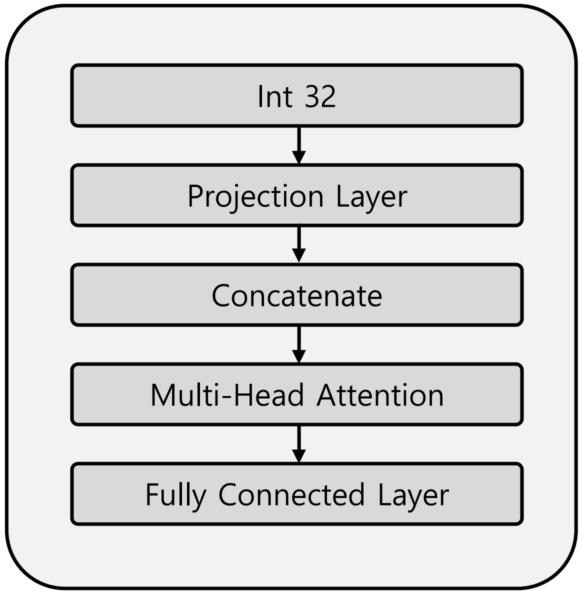
- Validation accuracy를 측정하기 위해, main.py의 로그 및 그래프 생성 수정

- Overfitting 줄이기 위해, Multi-Head Attention 과 FC layer에 dropout 적용

Structure of LLaMA Classify Model

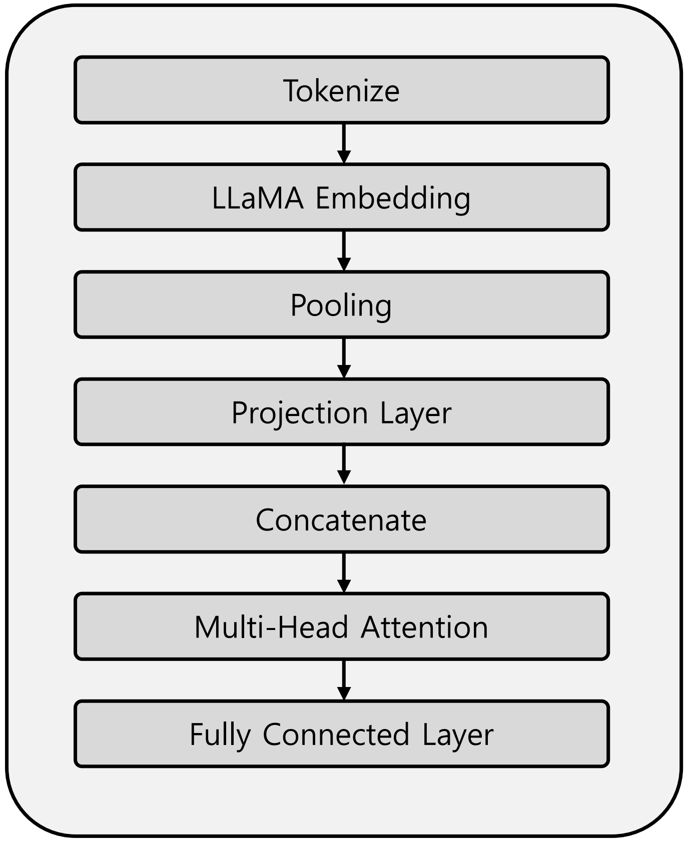


Text Feature



Int Feature

Structure of LLaMA Classify Model



input_ids : (1, 128), attention_mask : (1,128)

{MAX_TOKEN_SIZE = 4096} → hidden_states : (1, 128, 4096)

hidden_states : (1, 128, 4096) → Pooling : (1, 4096)

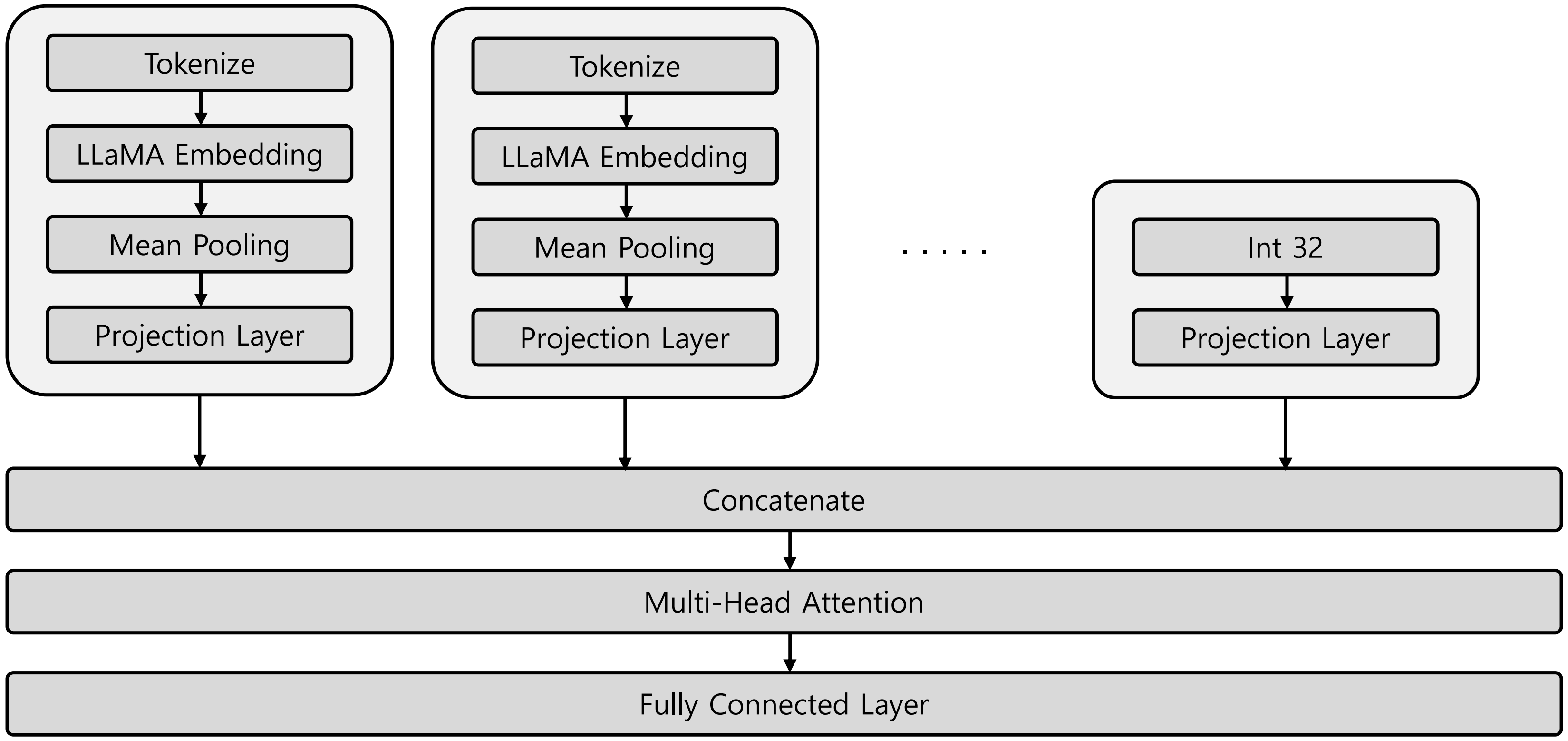
Linear (4096 → 64) : (1, 64) → Unsqueeze (1, 1, 64)

Concatenate Feature : (1, 7, 64)

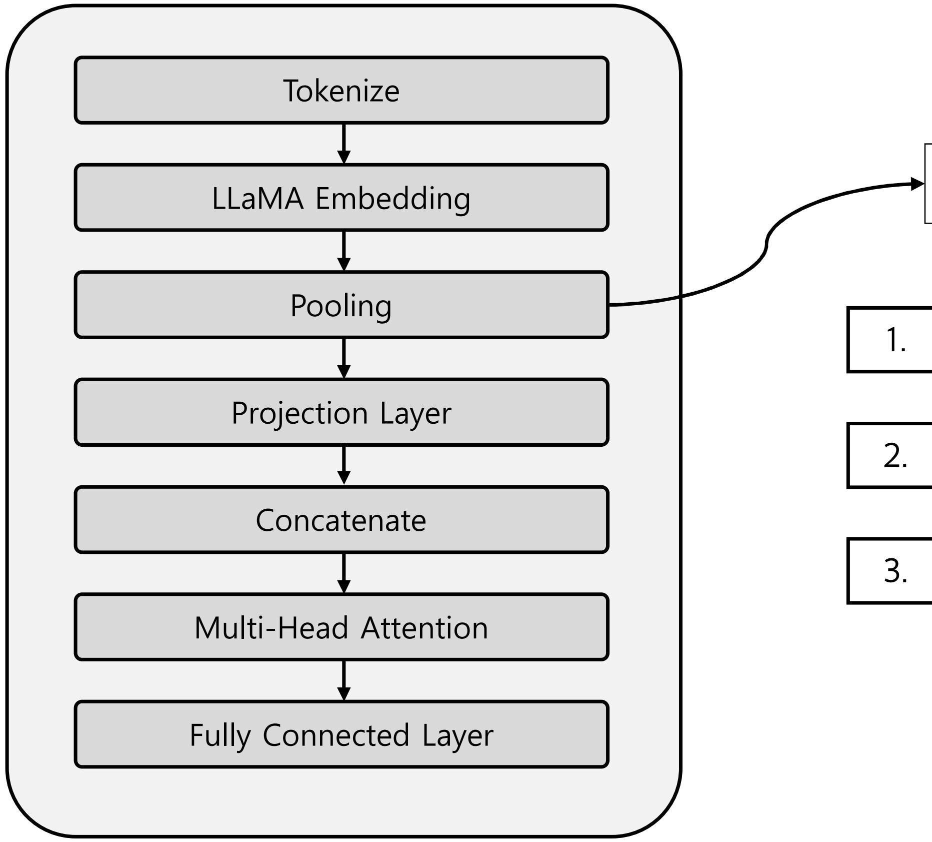
Multi-Head Attention : (1, 7, 64) → Flatten : (1, 448)

→ Fully Connected Layer: (1,64) → Fully Connected Layer : (1,2)

Structure of LLaMA Classify Model



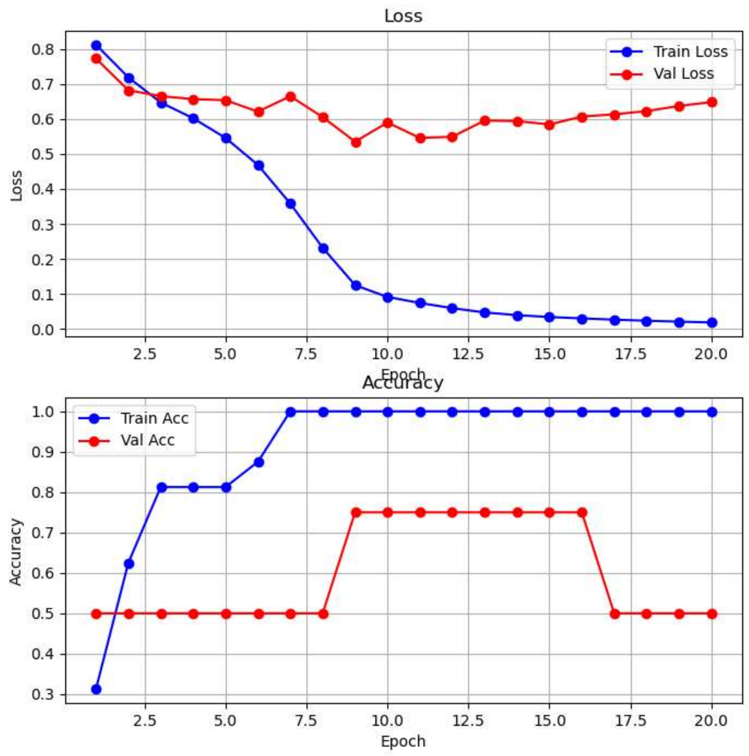
Pooling

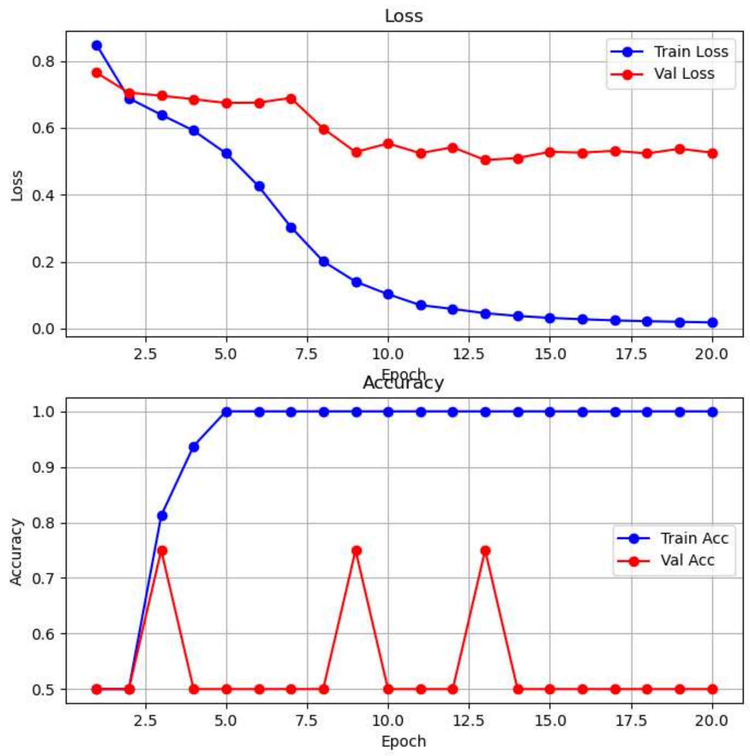


hidden_states : (1, 128, 4096) → Pooling : (1, 4096)

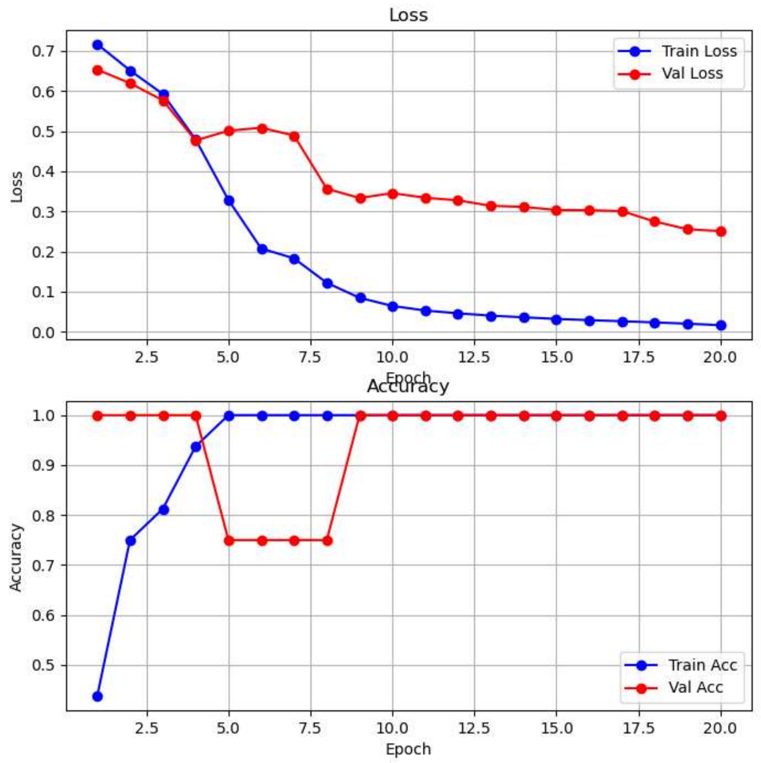
1.	Max Pooling
2.	Avg Pooling
3.	Attention-weight Pooling

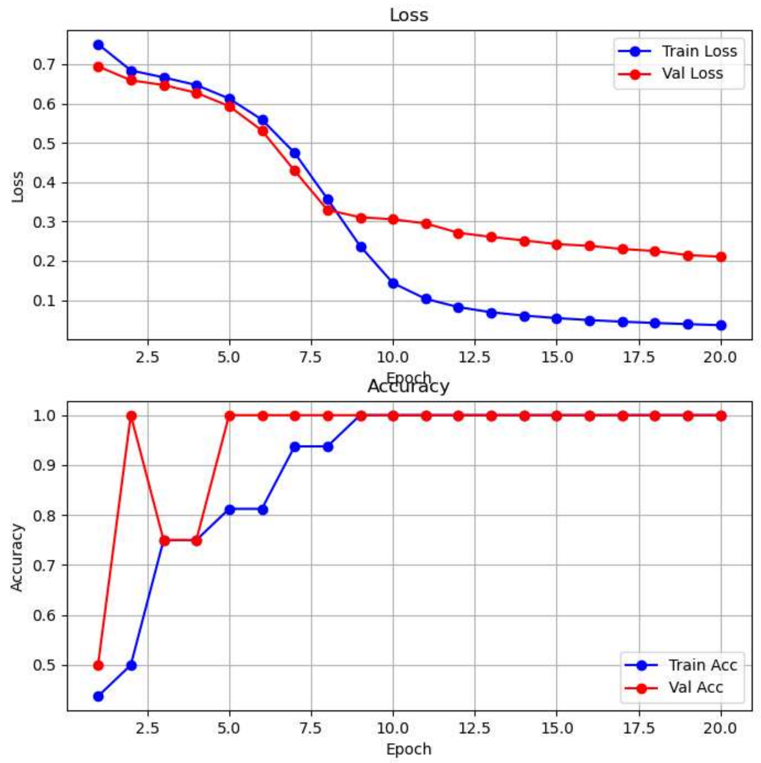
Max Pooling


Loss
Train Loss
Val Loss
Loss
Epoch
Accuracy
Train Acc
Val Acc
Accuracy
Epoch

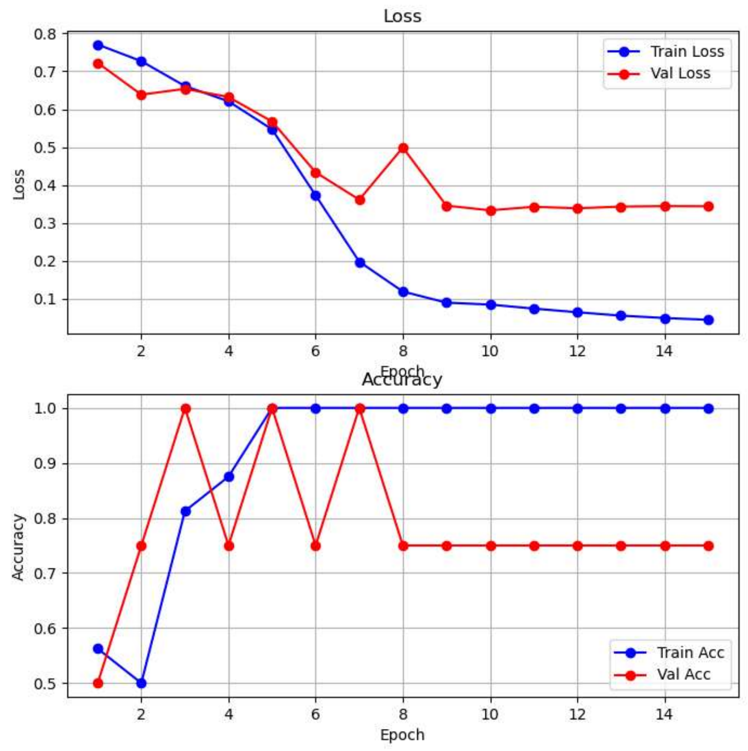

Loss
Train Loss
Val Loss
Loss
Epoch
Accuracy
Train Acc
Val Acc
Accuracy
Epoch

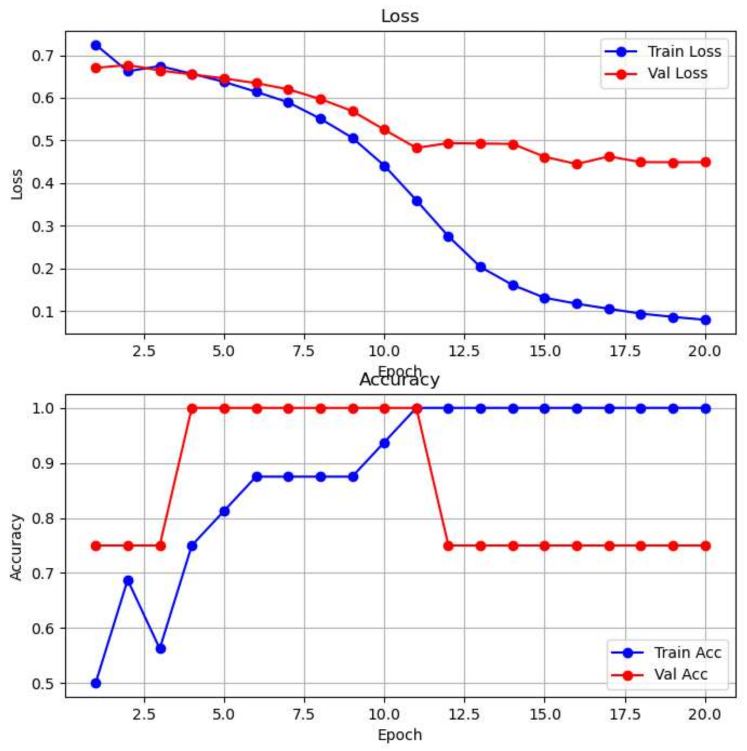
Attention-based weighted Pooling



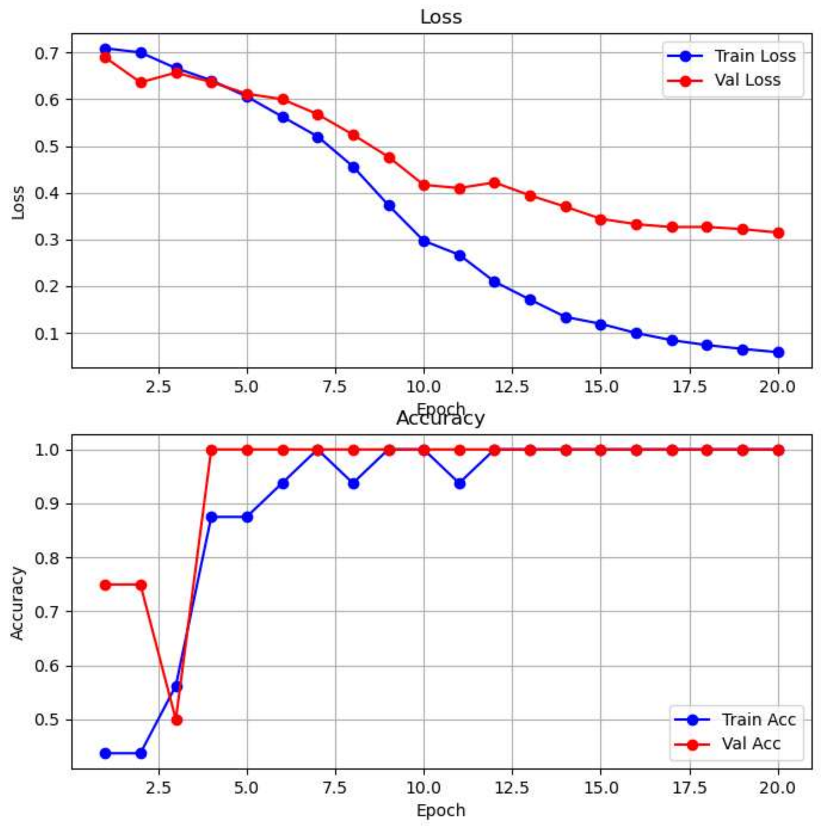


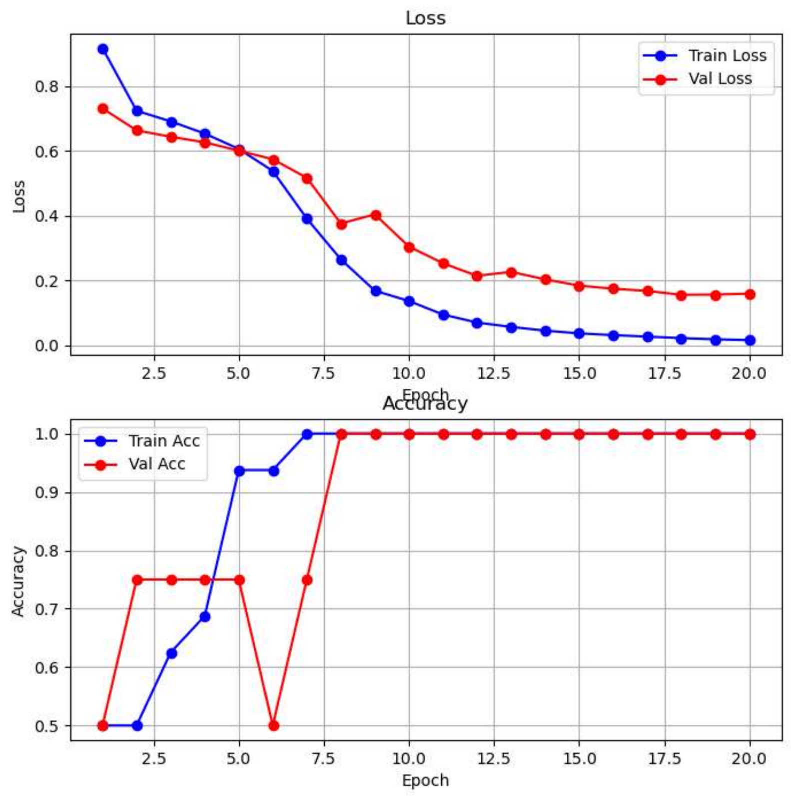
Attention-based weighted Pooling



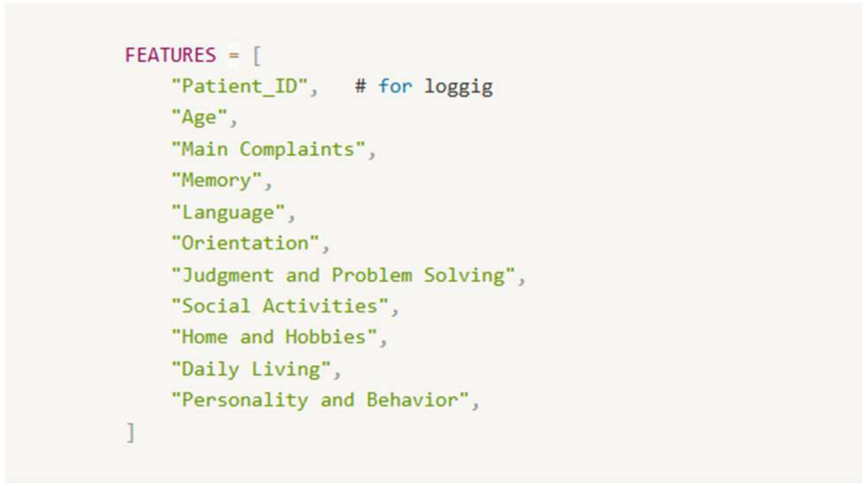


Avg Pooling

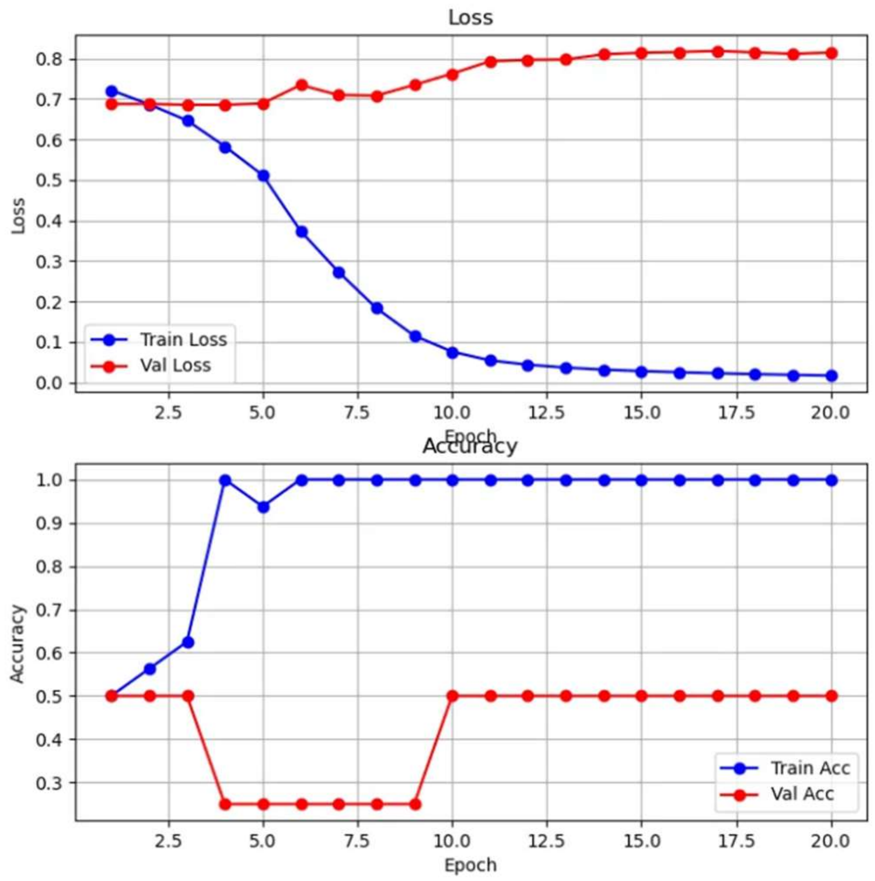

Loss
Train Loss
Val Loss
Loss
Epoch
Accuracy
Train Acc
Val Acc
Accuracy
Epoch


Loss
Train Loss
Val Loss
Loss
Epoch
Accuracy
Train Acc
Val Acc
Accuracy
Epoch

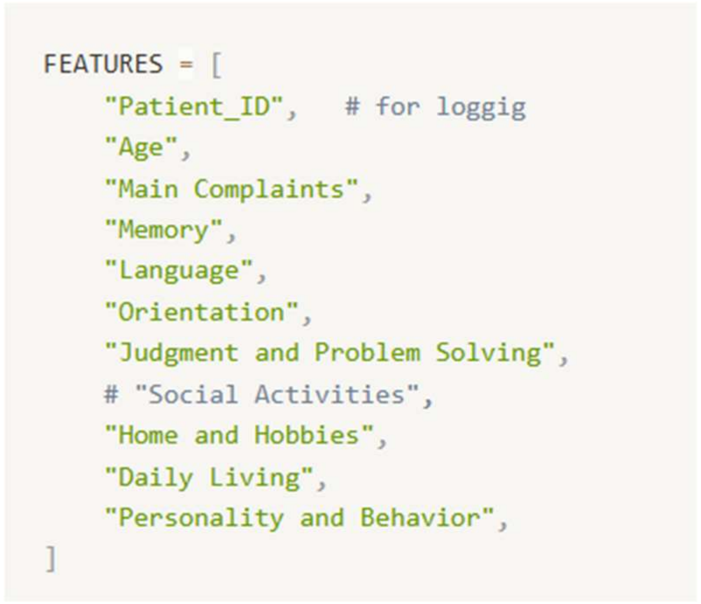
Feature Engineering



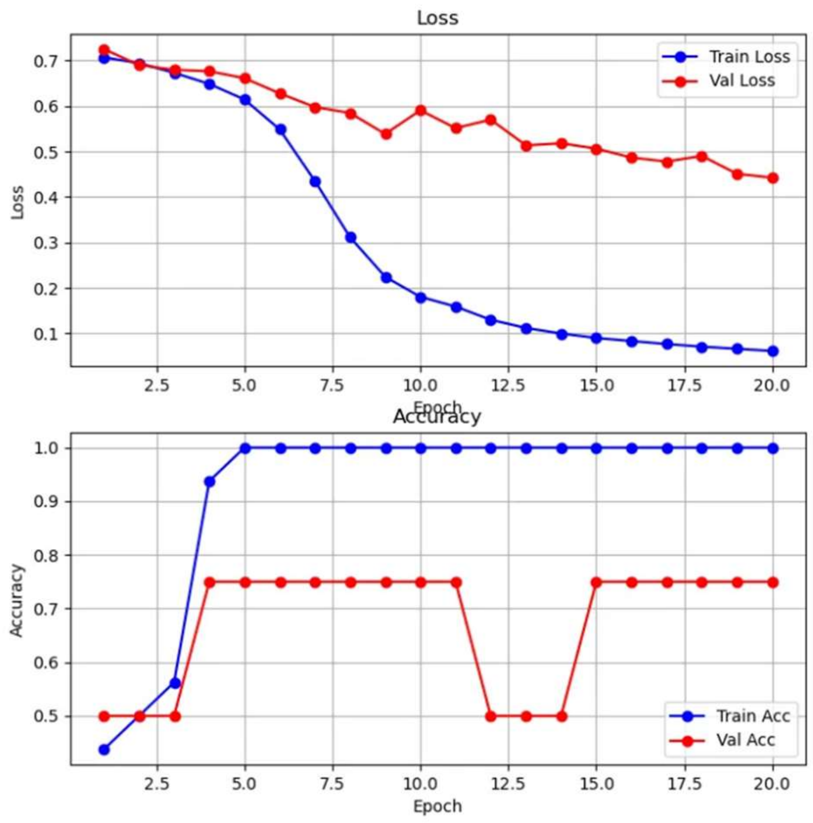
```
FEATURES = [
    "Patient_ID",    # for loggig
    "Age",
    "Main Complaints",
    "Memory",
    "Language",
    "Orientation",
    "Judgment and Problem Solving",
    "Social Activities",
    "Home and Hobbies",
    "Daily Living",
    "Personality and Behavior",
]
```



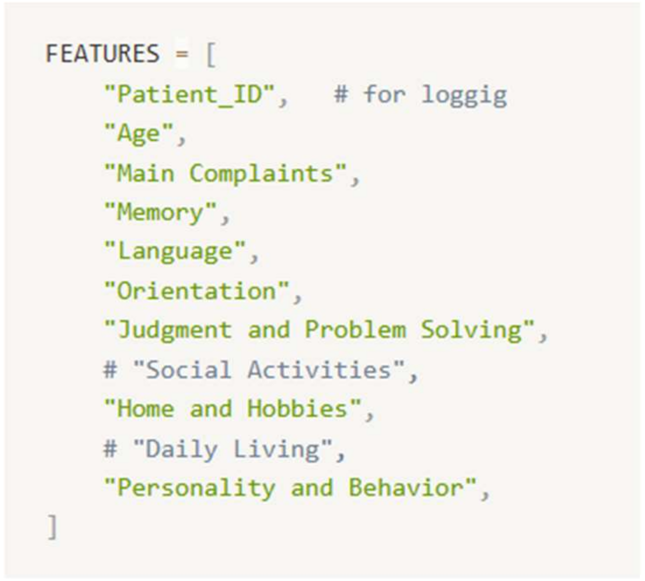
Feature Engineering



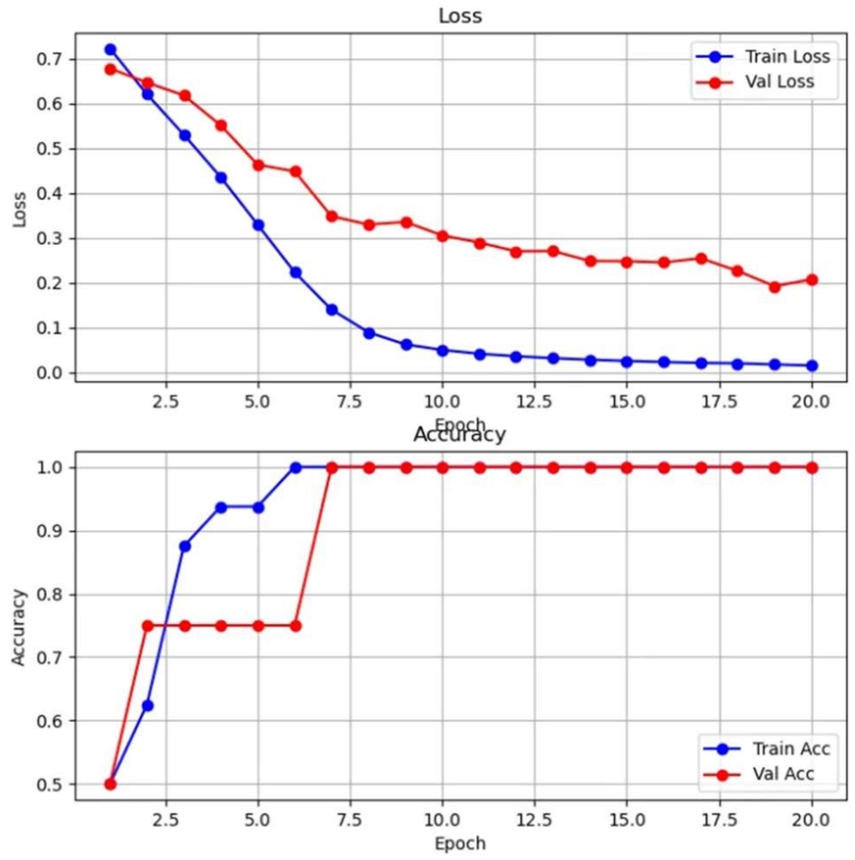
```
FEATURES = [
    "Patient_ID",   # for loggig
    "Age",
    "Main Complaints",
    "Memory",
    "Language",
    "Orientation",
    "Judgment and Problem Solving",
    # "Social Activities",
    "Home and Hobbies",
    "Daily Living",
    "Personality and Behavior",
]
```



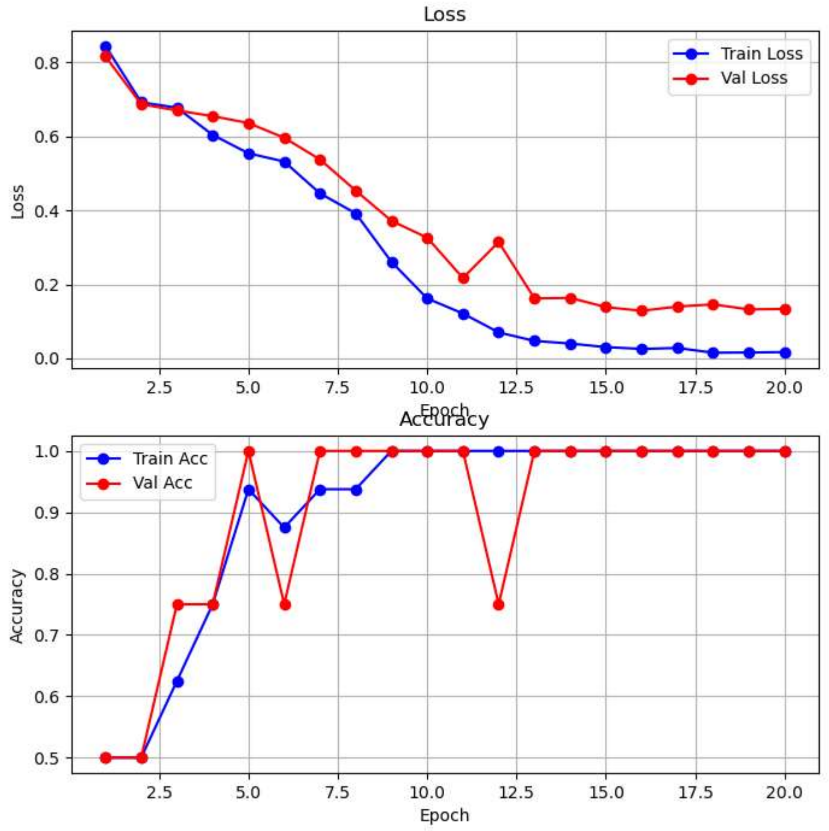
Feature Engineering



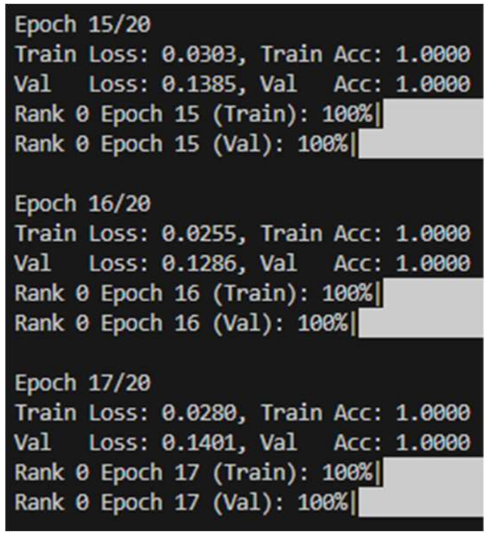
```
FEATURES = [
    "Patient_ID",   # for loggig
    "Age",
    "Main Complaints",
    "Memory",
    "Language",
    "Orientation",
    "Judgment and Problem Solving",
    # "Social Activities",
    "Home and Hobbies",
    # "Daily Living",
    "Personality and Behavior",
]
```



Optimized Model : Dropout(0.2)



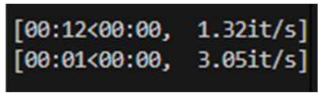
Result



```
Epoch 15/20
Train Loss: 0.0303, Train Acc: 1.0000
Val   Loss: 0.1385, Val   Acc: 1.0000
Rank 0 Epoch 15 (Train): 100%|
Rank 0 Epoch 15 (Val): 100%|

Epoch 16/20
Train Loss: 0.0255, Train Acc: 1.0000
Val   Loss: 0.1286, Val   Acc: 1.0000
Rank 0 Epoch 16 (Train): 100%|
Rank 0 Epoch 16 (Val): 100%|

Epoch 17/20
Train Loss: 0.0280, Train Acc: 1.0000
Val   Loss: 0.1401, Val   Acc: 1.0000
Rank 0 Epoch 17 (Train): 100%|
Rank 0 Epoch 17 (Val): 100%|
```



```
[00:12<00:00,  1.32it/s]
[00:01<00:00,  3.05it/s]
```

학습 속도 : 1.32it/s → 1 Epoch에 12sec 소요

예측 속도 : 3.05it/s → 1 Epoch에 1sec 소요

→ Data 1000개로 증강 시 1 Epoch 20~30min 예상

→ 적은 Epoch 수로 결과 확인 가능할 것이라 판단

Batch Size를 늘리면, GPU Out of Memory(OOM) 발생.