Groom Project#1

Text Sentiment Classification

< 2조 (2조) >

조현수, 이준엽, 김남준, 김영수, 손동협, 조승연, 정명관

01 PPT PRESENTATION

1. 프로젝트 개요
    - 프로젝트 개요 및 개발환경
    - 프로젝트 진행과정
    - 기대효과
2. 프로젝트 팀 구성 및 역할
    - 프로젝트 팀 구성 및 담당 part
3. 프로젝트 진행 프로세스
    - 전체일정
4. 프로젝트 결과
    - 결과 제시 ① 탐색적 분석 및 전처리
    - 결과 제시 ② 모델 개요
    - 프로젝트 결과 도출 과정
    - 코드 모듈화
    -fine-tuning
5. 자체 평가 및 피드백
    - 평가 및 보완점
    - 조원 간의 피드백
6. Q&A

01 프로젝트 개요

프로젝트 개요 및 개발환경

프로젝트 구현 내용

Sentimental sentence를 BERT 모델을 기반하여 학습시켜 sentimental classification 모델을 구축,
BERT를 fine-tuning 한다

활용 라이브러리, 프레임워크

Huggingface Transformers
BERT모델(BERT-base-uncased , BERT-large-uncased)
XLNet 모델(XLnet-base-cased)
→ GPT 모델과 BERT 모델의 장점들을 활용한 모델

WandB, Sweep

Docker를 이용한 개발환경 구축 및 이미지화
GitHub를 이용한 코드 버전 관리

01 프로젝트 개요

프로젝트 진행 과정

01 프로젝트 개요

프로젝트 기대효과

1. 감정 도메인의 문장에서 높은 성능을 가지는 모델 학습

2. 추가적인 데이터와 모델 세부사항을 더 개선할 경우 추천 시스템, 소비자 반응 분석 시스템에서도 사용 가능

3. BERT 모델 계열의 한국어 모델(KoBERT 등)을 중심으로 한국어 학습 모델까지도 확장

02 프로젝트 팀 구성 및 역할

02 프로젝트 팀 구성 및 담당 part

훈련생	역활	담당업무
조현수	팀장	베이스라인 코드 정리 및 모듈화, 전체part 관리
김남준	팀원	Ensemble 및 data part
김영수	팀원	옵티마이저 분석 및 각종 rate 조정 part
손동협	팀원	옵티마이저 분석 및 각종 rate 조정 part
이준엽	팀원	scheduler 및 sweep 을 통한 탐색 part
정명관	팀원	Ensemble 및 data part
조승연	팀원	scheduler 및 sweep 을 통한 탐색 part

02 프로젝트 팀 구성 및 역할

02 프로젝트 팀 구성 및 담당 part

공통part
1. WandB를 활용한 데이터 분석 2. 코드 분석 및 정리 3. task에 맞게 모델 fine-tuning 4. learning rate 선정

LR scheduler 및 Sweep 을 통한 탐색 part
1. Learning rate scheduler 분석 및 선정 2. WandB Sweep을 통한 최적의 hyper-parameter 탐색

Optimizer part
1. train, valid dataset 비율 조정 및 병합 2. Optimizer들 분석 및 선정

Ensemble 및 Data part
1. 데이터 proprocessing (정규식 표현) 2. 앙상블 기법 적용 (hard voting, soft voting) 3. 결과데이터 시각화 4. 회의 내용 정리

03 프로젝트 진행 프로세스

03 전체일정

첫번째 프로젝트 : Goorm NLP project 1 Text classification

- 3/25 프로젝트 사전공개
- 3/28 ~ 4/1 5일 간 진행
- 주제 선정부터 보고서 작성까지 총 5단계에 걸쳐 프로젝트 진행

구분	기간	활동	비고
주제 선정 및 기획	3/28 월	프로젝트 사전 고지 Kaggle Competition 공개	Baseline 함정 코드 문제
모델 선정	3/29 화 ~ 3/31 목	Baseline-Bert 모델 활용 Bert 외 다른 모델 검토	
모델 튜닝	3/30 수 ~ 3/31 목	Test 코드에서 함정 해결 Baseline 코드 수정	
모델 성능 개선	3/30 수 ~ 4/1 금	Hyperparameter 수정 Fine tunning 지속	
자료 수집 및 보고서 작성	3/31 목 ~ 4/1 금	Wandb를 활용한 시각화 자료 수집	Sweep 기능 활용
총 소요기간	5일		

04 프로젝트 결과

04 결과 제시 ① 탐색적 분석 및 전처리

▶ 학습 데이터 소개(Train / dev set)

● 데이터 크기 및 길이

- Sentiment.dev0 : 91.61 kB / 2000
  Max_len : 23 / Mean_len : 13

- Sentiment.dev1 : 84.92 kB / 2000
  Max_len : 19 / Mean_len : 11

- Sentiment.Train0 : 8.24 MB / 177218
  Max_len : 24 / Mean_len : 13

- Sentiment.Train1 : 11.37 MB / 266041
  Max_len : 19 / Mean_len : 11

- Tokenizing : BERT
- Regular Expression : 전체적인 자료에서 숫자가 있는 자리에 _num_으로 대체되어 있어 _num_과 ₩n 및 불필요한 특수기호($ 등)를 제거하는 방식 사용

```
["windows have n't been cleaned in years you can see scum on them .₩n',
 'waitresses are slow .₩n',
 'just a mess avoid at all costs !₩n',
 'bad !₩n',
 'now pizza is beyond awful and wings are down there with its level .₩n',
 'walked out of this place after _num_ min of no service .₩n',
 "the place was n't even busy at this time .₩n",
 'never will i be back to this place .₩n',
 'this place is awful !₩n',
 'not the food but the service .₩n']
```

Regular Expression

```
['windows have nt been cleaned in years you can see scum on them',
 'waitresses are slow',
 'just a mess avoid at all costs',
 'bad',
 'now pizza is beyond awful and wings are down there with its level',
 'walked out of this place after min of no service',
 'the place was nt even busy at this time',
 'never will i be back to this place',
 'this place is awful',
 'not the food but the service']
```

04 프로젝트 결과

04 결과 제시 ② 모델 개요

▶ **BERT (**Bidirectional Encoder Representations from Transformers)

- 특정 분야에 국한된 기술이 아닌 fine-tuning을 이용하여 여러 task에서 좋은 성능을 내는 범용 Language Model

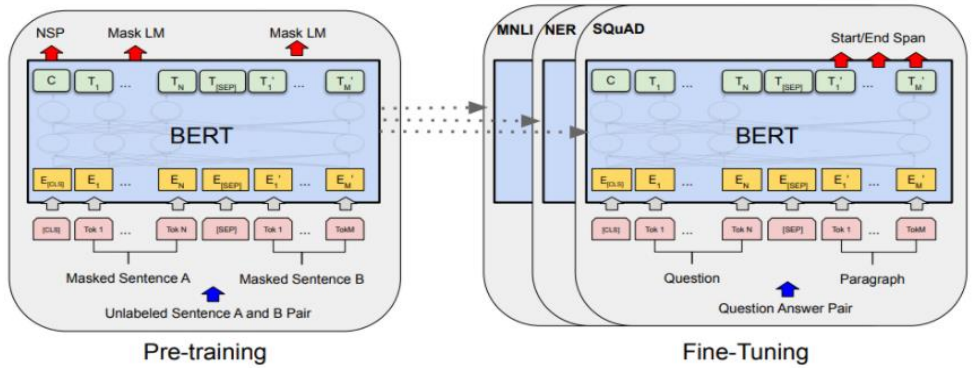
1. Input Representation

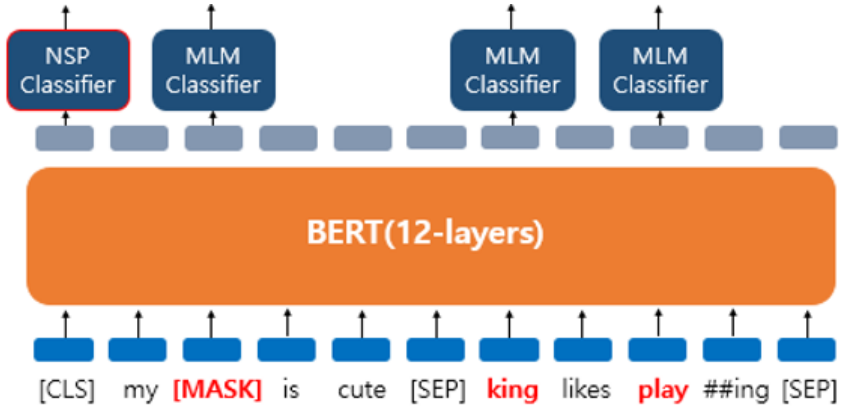
Token Embedding
- WordPiece

Segment Embedding
- 두 개의 문장을 문장 구분자와 함께 입력

Position Embedding
- Sinusoidal function

2. Pre-training
- MLM, NSP





04 프로젝트 결과

04 코드 모듈화

```
20  from compute import compute_acc
21  from visualize_score import plot_graph
22  from dump_datasets import mk_dataset, mk_dataset_xlnet
23  from dump_models import load_model, load_model_xlnet
24  from evaluate import test_model
25  from data_processing import regular
```

< 모듈화 >

재사용성 UP

빠른 Debugging

여러 조건으로 Training 용이

train.py

```
plot_graph(accloss_filename)
```

visualize.py

```
def plot_graph(file_name):
    with open('./scores/' + file_name,'rb') as f:
        train_acc = pickle.load(f)
        train_loss = pickle.load(f)
        valid_acc = pickle.load(f)
        valid_loss = pickle.load(f)

        valid_loss = [float(l.cpu()) for l in valid_loss]

    plt.subplot(2, 1, 1)
    plt.title('Accuracy')
    plt.plot(train_acc, label='train')
    plt.plot(valid_acc, label='valid')
    plt.legend(loc=5)

    plt.subplot(2, 1, 2)
    plt.title('Loss')
    plt.plot(train_loss, label='train')
    plt.plot(valid_loss, label='valid')
    plt.legend(loc=5)

    plt.show()
```

04 프로젝트 결과

04 코드 모듈화

```
20  from compute import compute_acc
21  from visualize_score import plot_graph
22  from dump_datasets import mk_dataset, mk_dataset_xlnet
23  from dump_models import load_model, load_model_xlnet
24  from evaluate import test_model
25  from data_processing import regular
```

< 모듈화 >

재사용성 UP

빠른 Debugging

```
root@78ca778771d1:/workspace# python train.py
Traceback (most recent call last):
  File "train.py", line 24, in <module>
    from evaluate import test_model
  File "/workspace/evaluate.py", line 1
    fwimport pickle
             ^
```

여러 조건으로 Training 용이

04 프로젝트 결과

04 코드 모듈화

```
20  from compute import compute_acc
21  from visualize_score import plot_graph
22  from dump_datasets import mk_dataset, mk_dataset_xlnet
23  from dump_models import load_model, load_model_xlnet
24  from evaluate import test_model
25  from data_processing import regular
```

< 모듈화 >

재사용성 UP

빠른 Debugging

➡ 여러 조건으로 Training 용이

train.py

```
#--------------------------------------DATA PREPROCESSING--------------------------------------
# Remove '_num_', special symbols(!@#%$% ..) in datasets
# train_pos = regular(train_pos)
# train_neg = regular(train_neg)
# dev_pos = regular(dev_pos)
# dev_neg = regular(dev_neg)

# use train, valid datasets for training
# train_pos = train_pos + dev_pos
# train_neg = train_neg + dev_neg
# len(train_pos)
#-----------------------------------------TOKENIZE-----------------------------------------
# seperate encoding to preprocess data before encoding
train_pos = [tokenizer.encode(line) for line in train_pos]
train_neg = [tokenizer.encode(line) for line in train_neg]
dev_pos = [tokenizer.encode(line) for line in dev_pos]
dev_neg = [tokenizer.encode(line) for line in dev_neg]
```

04 프로젝트 결과

04 fine-tuning

- hugging face의 여러 pretrained model 사용
- 여러 fine-tuned model 앙상블(soft voting, hard voting)
- Learning Scheduler 변경
  (get_cosine_schedule_with_warmup)
- optimizer 변경 (AdamW, Adamfactor)
- train, vaild용 데이터 merge 후 학습
- data preprocessing (removing special, unknown toke
- WandB를 이용한 성능 기록 및 시각화
- hyper-parameter tuning
- WandB Sweep을 이용한 hyper-parameter 최적화

BERT-base-uncased: 110M parameters

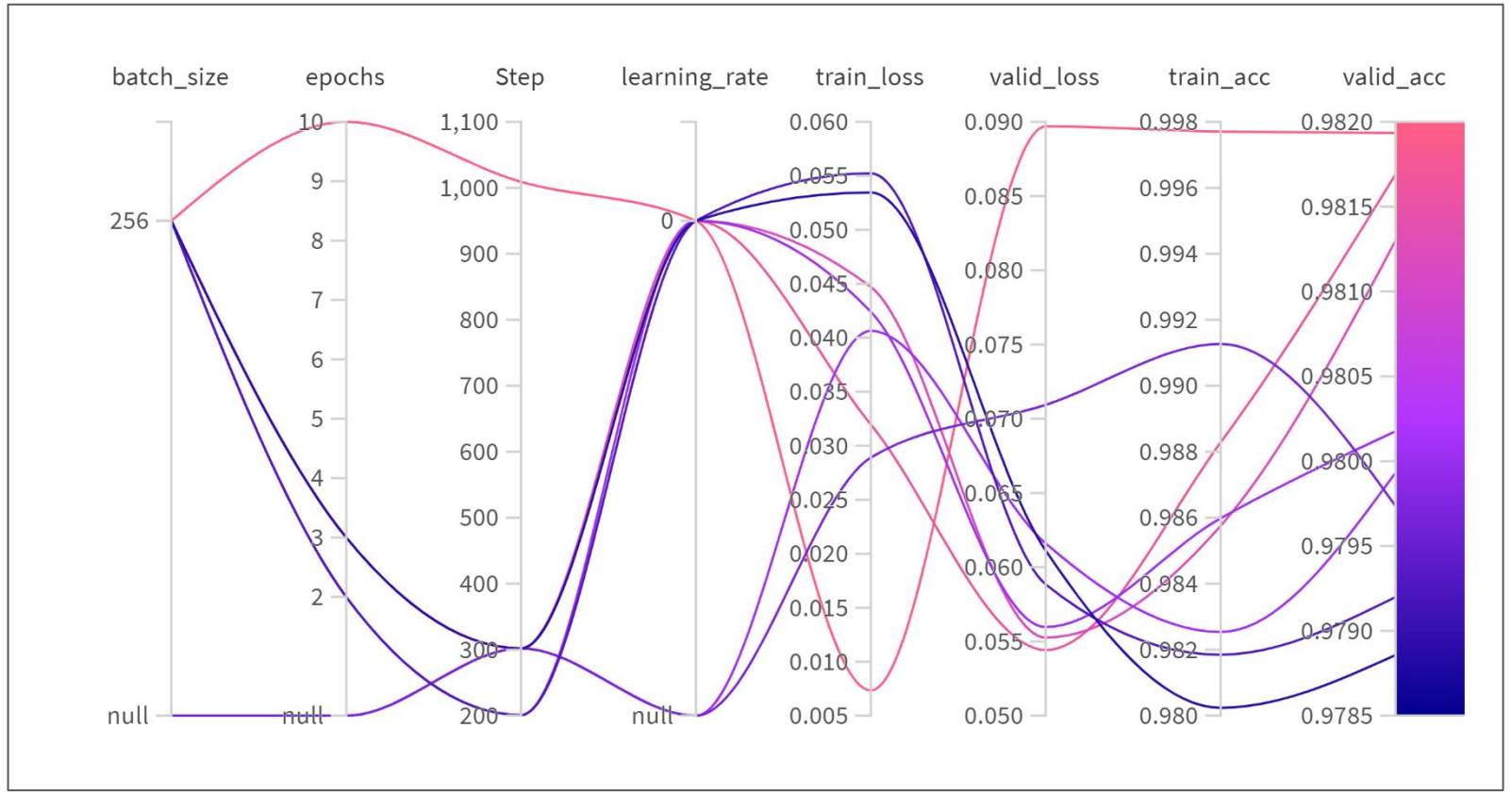
XLNet-base-cased: 110M parameters

RoBERTa-base: 125M
BERT-large-uncased: 340M parameters

```
optimizer = AdamW(model.parameters(), lr=LEARNING_RATE)
# optimizer = Adafactor(model.parameters(), lr=LEARNING_RATE,
#                       eps=(1e-30, 1e-3), decay_rate=-0.8, clip_threshold=1.0,
#                       beta1=None,
#                       weight_decay=0.0,
#                       relative_step=False,
#                       scale_parameter=False,
#                       warmup_init=False,)
scheduler = get_cosine_schedule_with_warmup(optimizer,
                                            num_warmup_steps=int(1732*TRAIN_EPOCH*0.1),
                                            num_training_steps=1732*TRAIN_EPOCH)
```

04 프로젝트 결과
04 fine-tuning

- **Wandb sweep을 사용한 모델 결과**



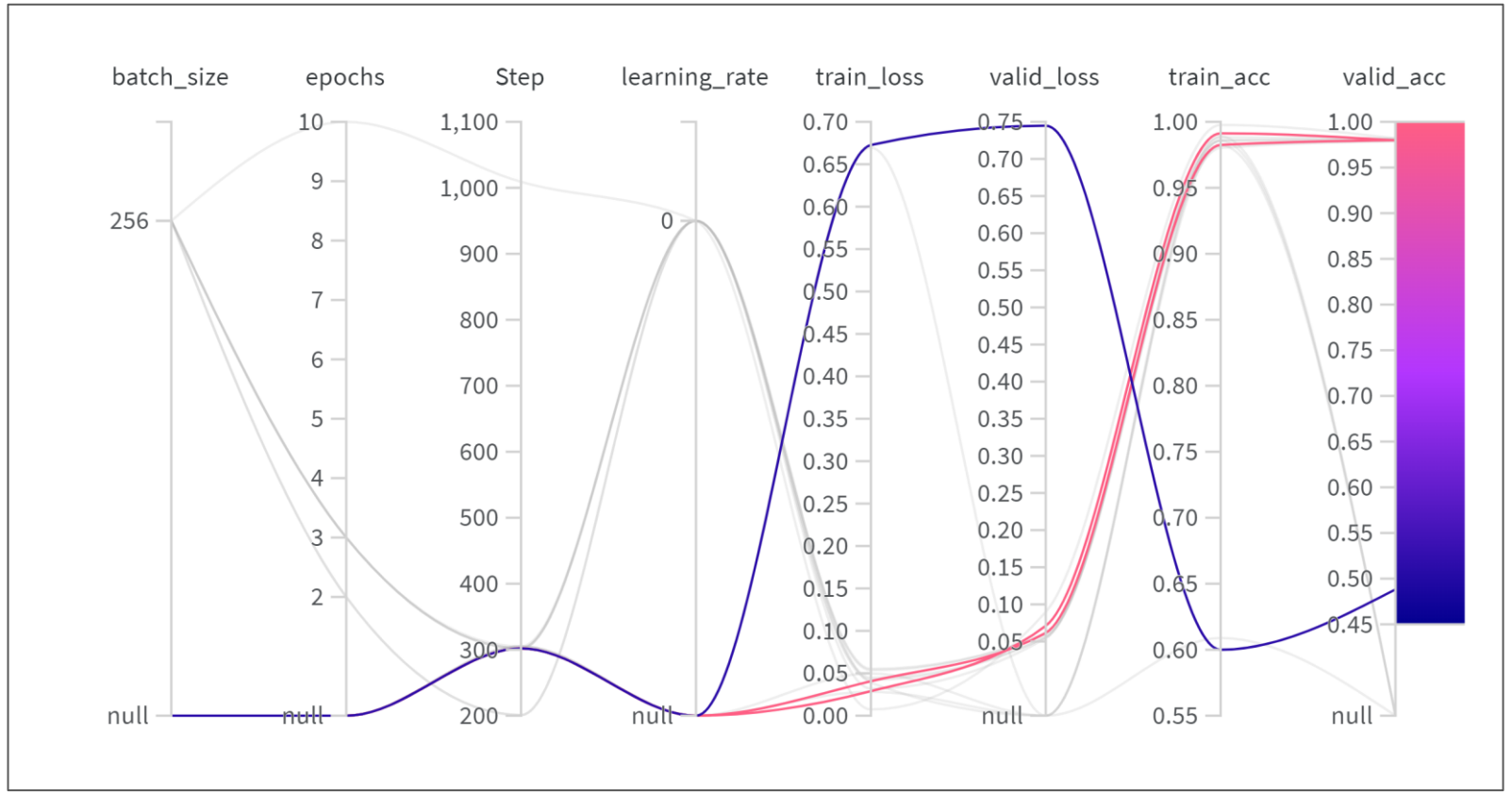
- **BERT**
Learning rate를 변화
Train + Valid
Epoch 변화
정규화 추가
LR Scheduler

- **Xlnet**
Epoch 변화

https://wandb.ai/goorm_team_2/project1/reports/Shared-panel-22-04-01-12-04-14--VmlldzoxNzcyNDMz

04 프로젝트 결과
04 fine-tuning

- Learning Rate 변화

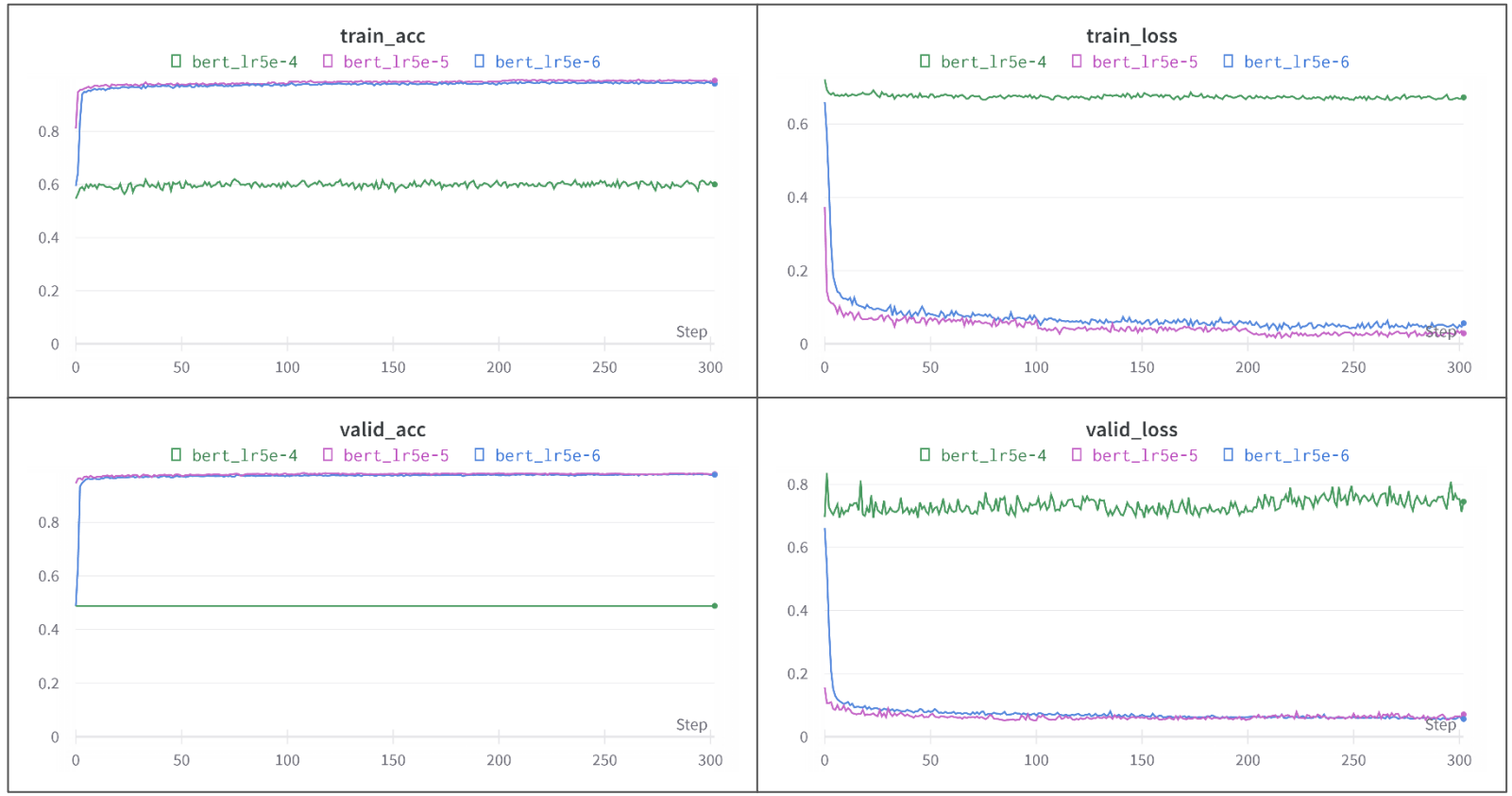


Learning Rate를 5e-4, 5e-5, 5e-6 으로 하여 실행

https://wandb.ai/goorm_team_2/project1/reports/Shared-panel-22-04-01-12-04-25--VmlldzoxNzcyNDkz

04 프로젝트 결과
04 fine-tuning

- **Learning Rate 변화**



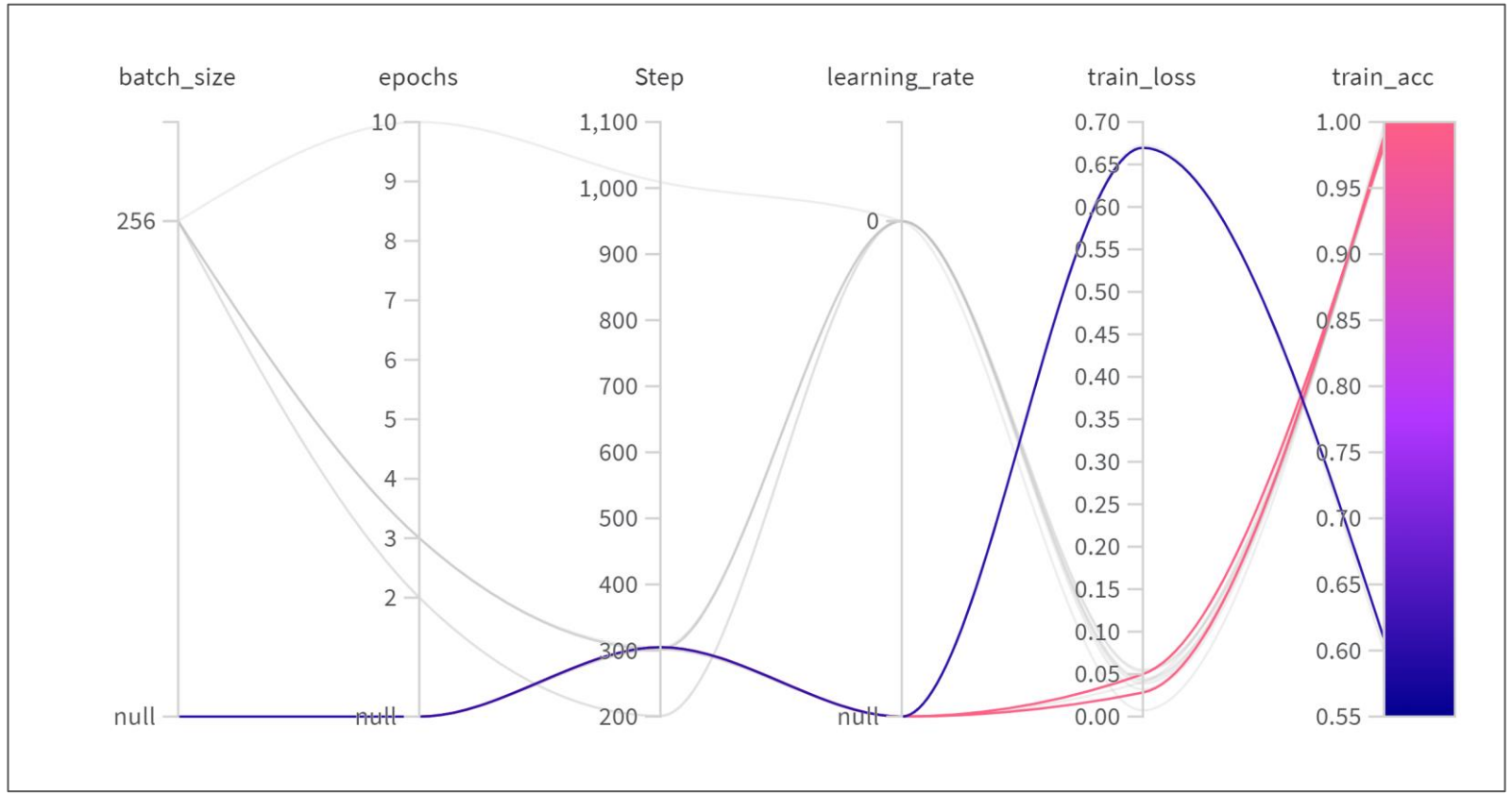
5e-5가 성능이 가장 좋음

5e-4는 값이 발산

https://wandb.ai/goorm_team_2/project1/reports/Learning-Rate--VmlldzoxNzcyNTE0

04 프로젝트 결과
04 fine-tuning

- **Train + Valid + Learning Rate 변화**

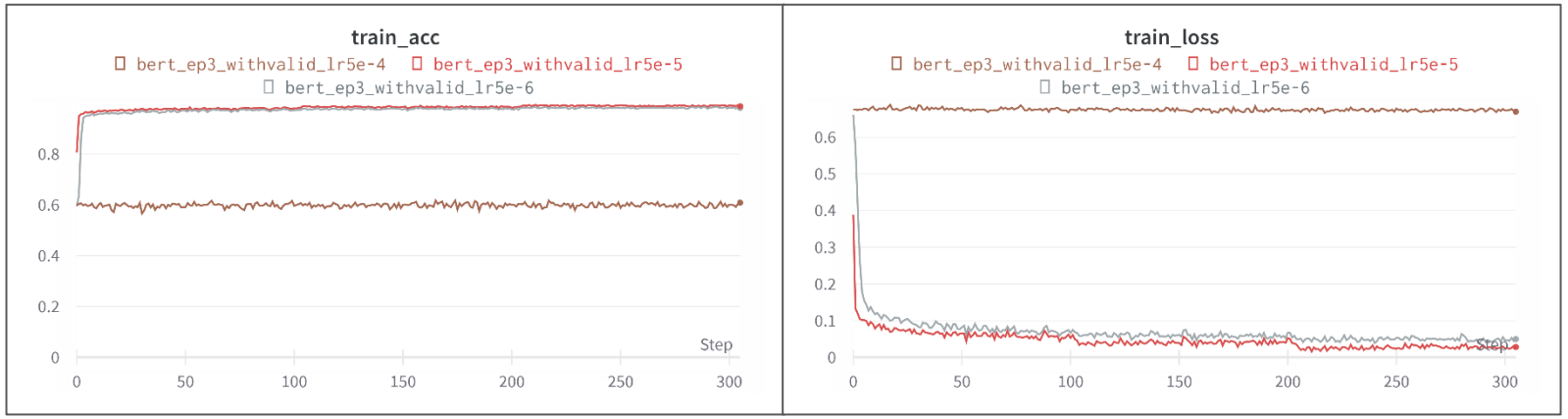


기존의 train data와 valid data를 합쳐서 실행

https://wandb.ai/goorm_team_2/project1/reports/Only-train--VmlldzoxNzcyNTQz

04 프로젝트 결과
04 fine-tuning

- Train + Valid + Learning Rate 변화

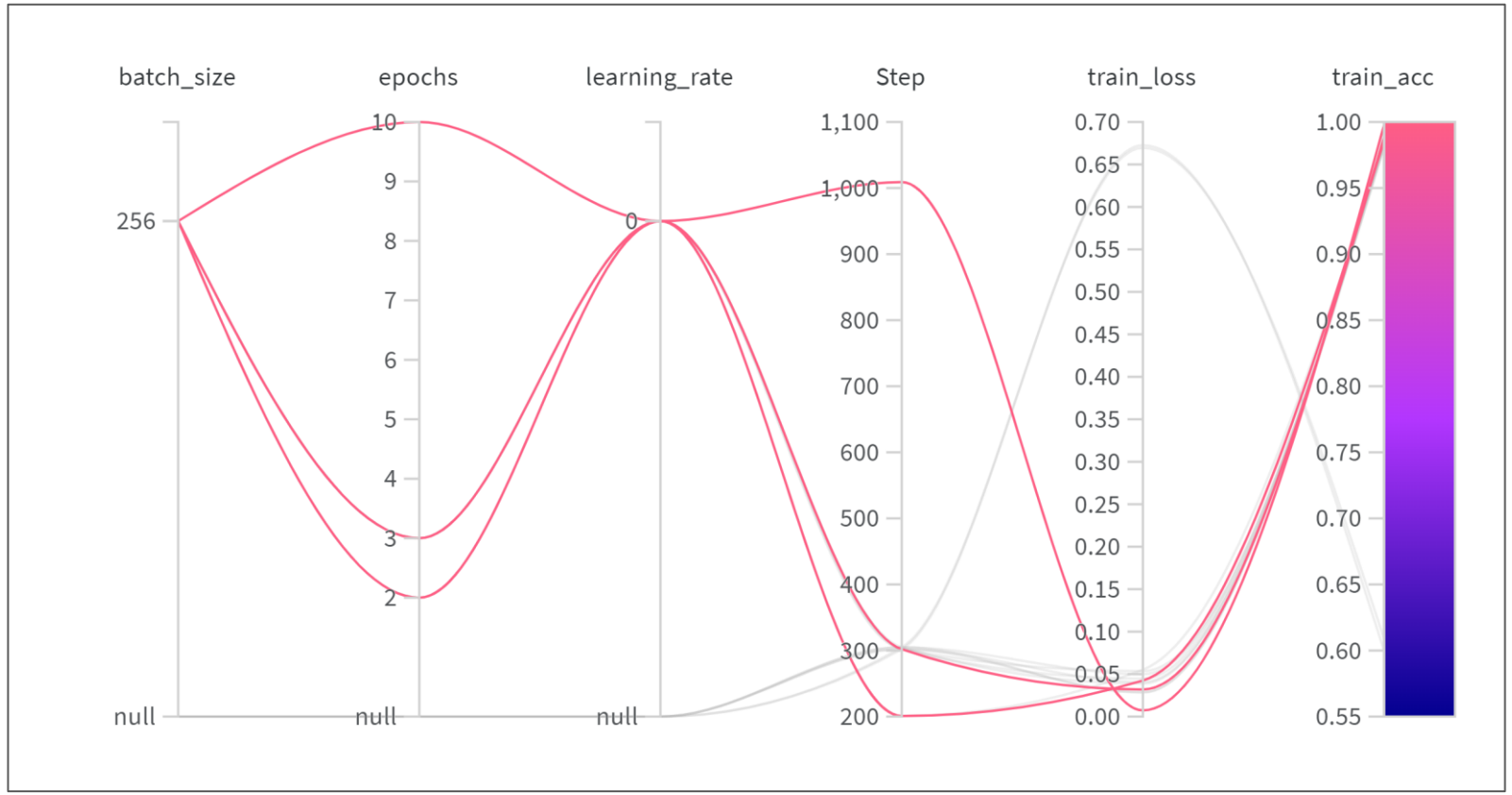


기존 valid 데이터를 train 데이터에 추가 하였고 5e-5가 가장 성능이 좋음

https://wandb.ai/goorm_team_2/project1/reports/Only-Train--VmlldzoxNzcyNTU5

04 프로젝트 결과
04 fine-tuning

- **Epoch 변화**

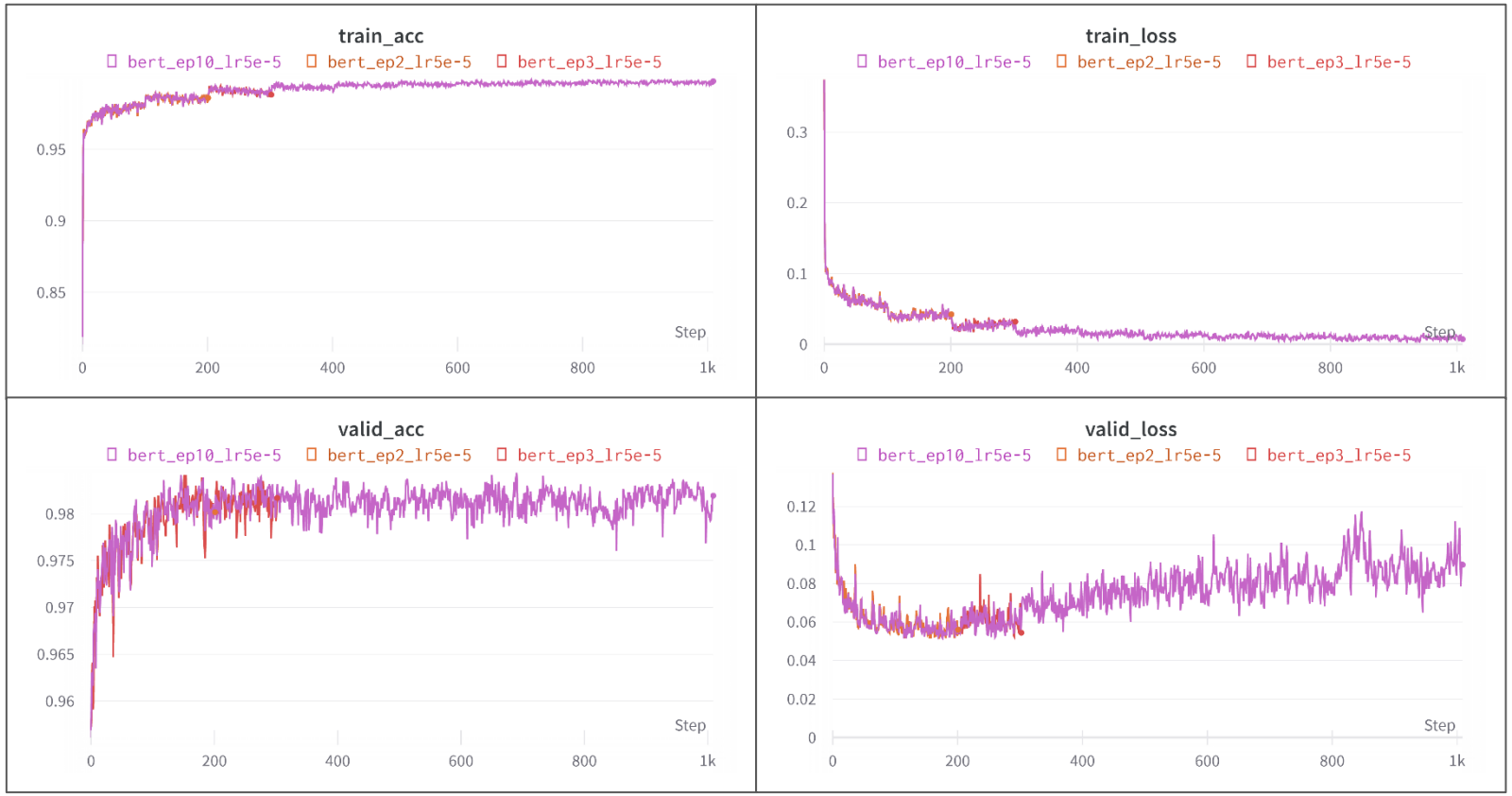


Epoch를 2, 3, 10 각각 실행

https://wandb.ai/goorm_team_2/project1/reports/Epoch--VmlldzoxNzcyNTcw

04 프로젝트 결과
04 fine-tuning

- **Epoch 변화**

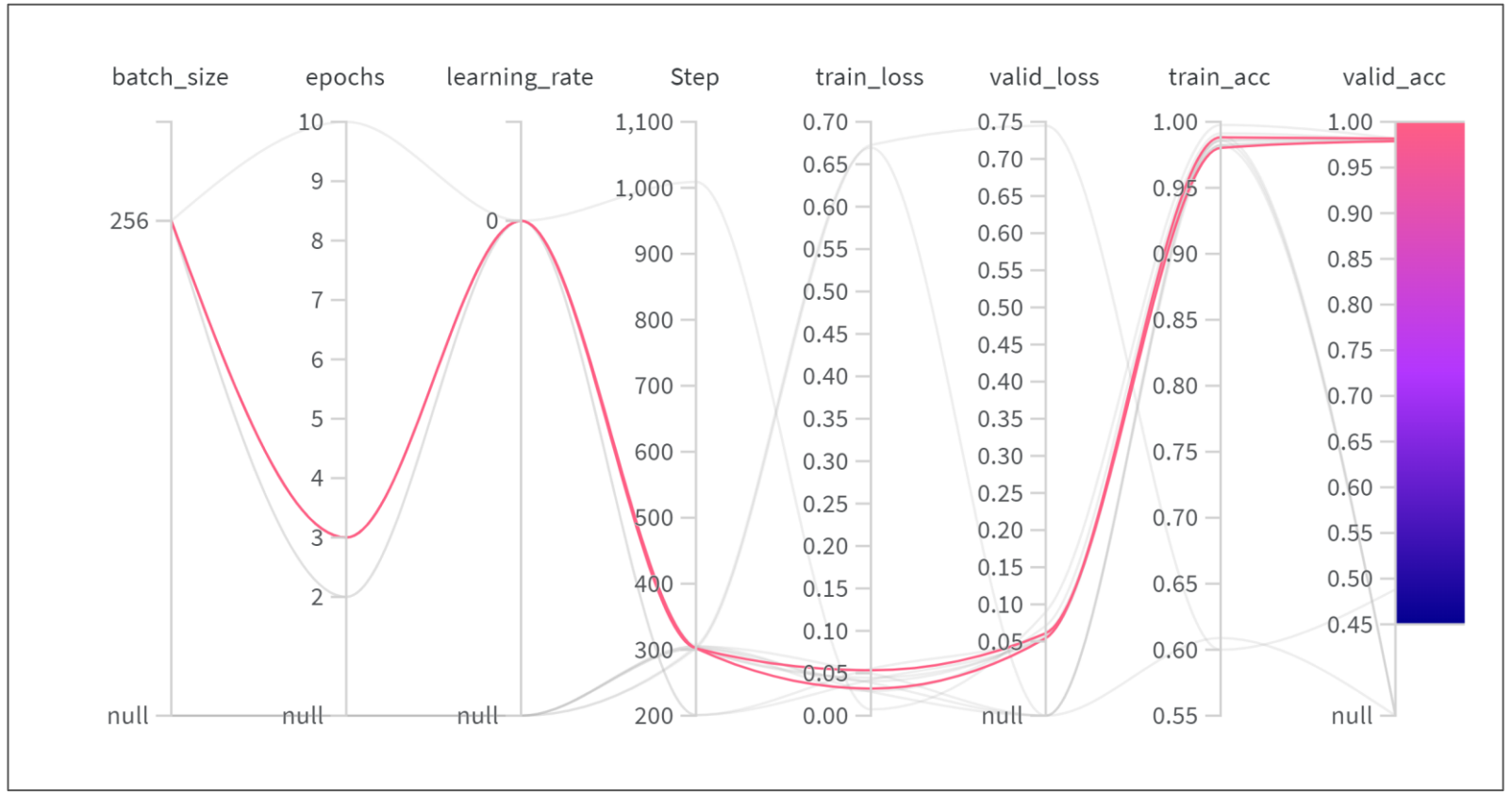


Epoch가 증가 할수록 train data 에 대해 성능이 좋아지지만 overfitting이 발생

https://wandb.ai/goorm_team_2/project1/reports/Epoch--VmlldzoxNzcyNTY3

04 프로젝트 결과
04 fine-tuning

- 정규화 추가 개선

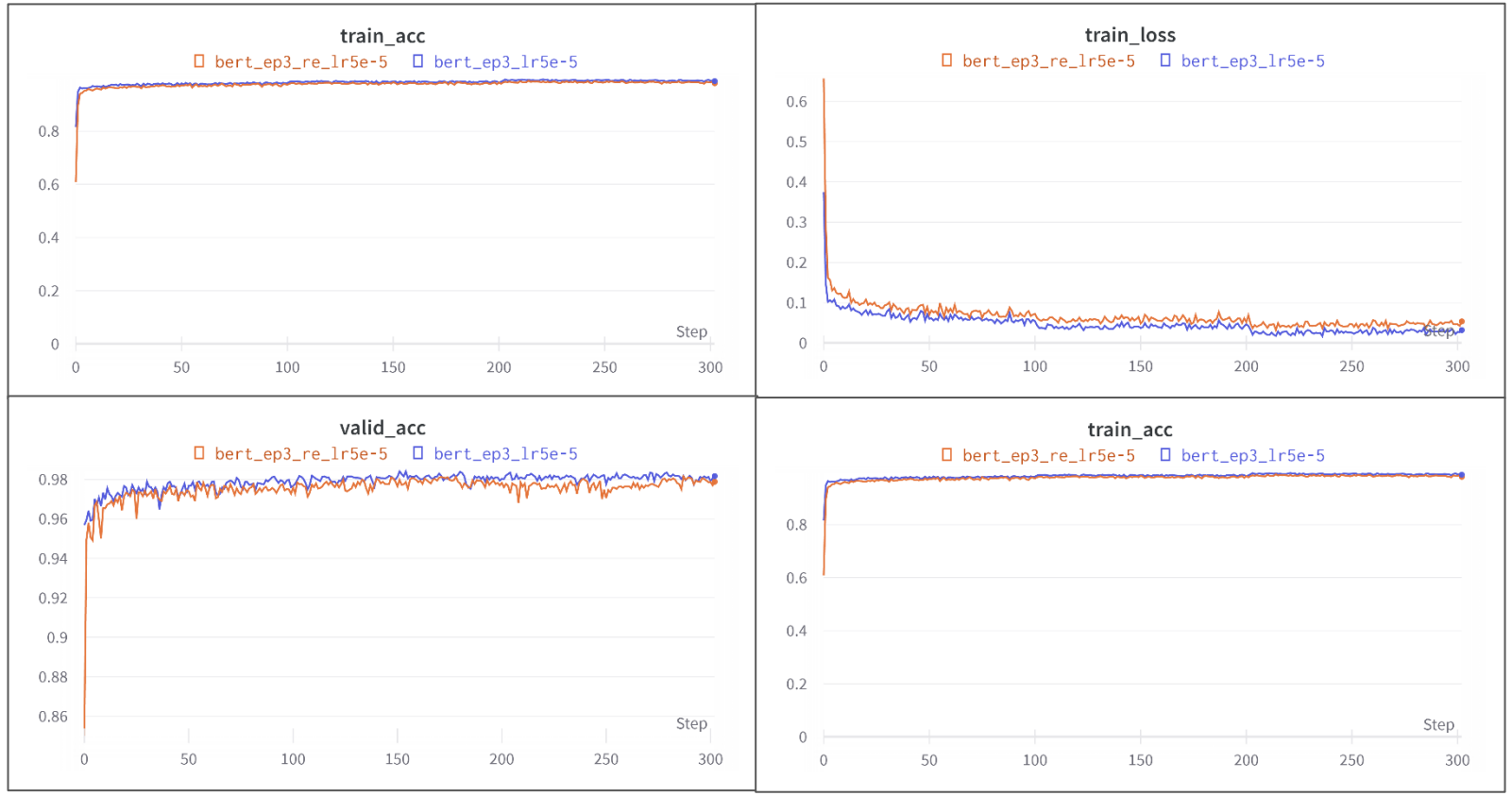


정규식 표현을 추가하여 데이터의 불순물을 제거

https://wandb.ai/goorm_team_2/project1/reports/Regularization--VmlldzoxNzcyNjE5

04 프로젝트 결과
04 fine-tuning

- 정규화 추가 개선

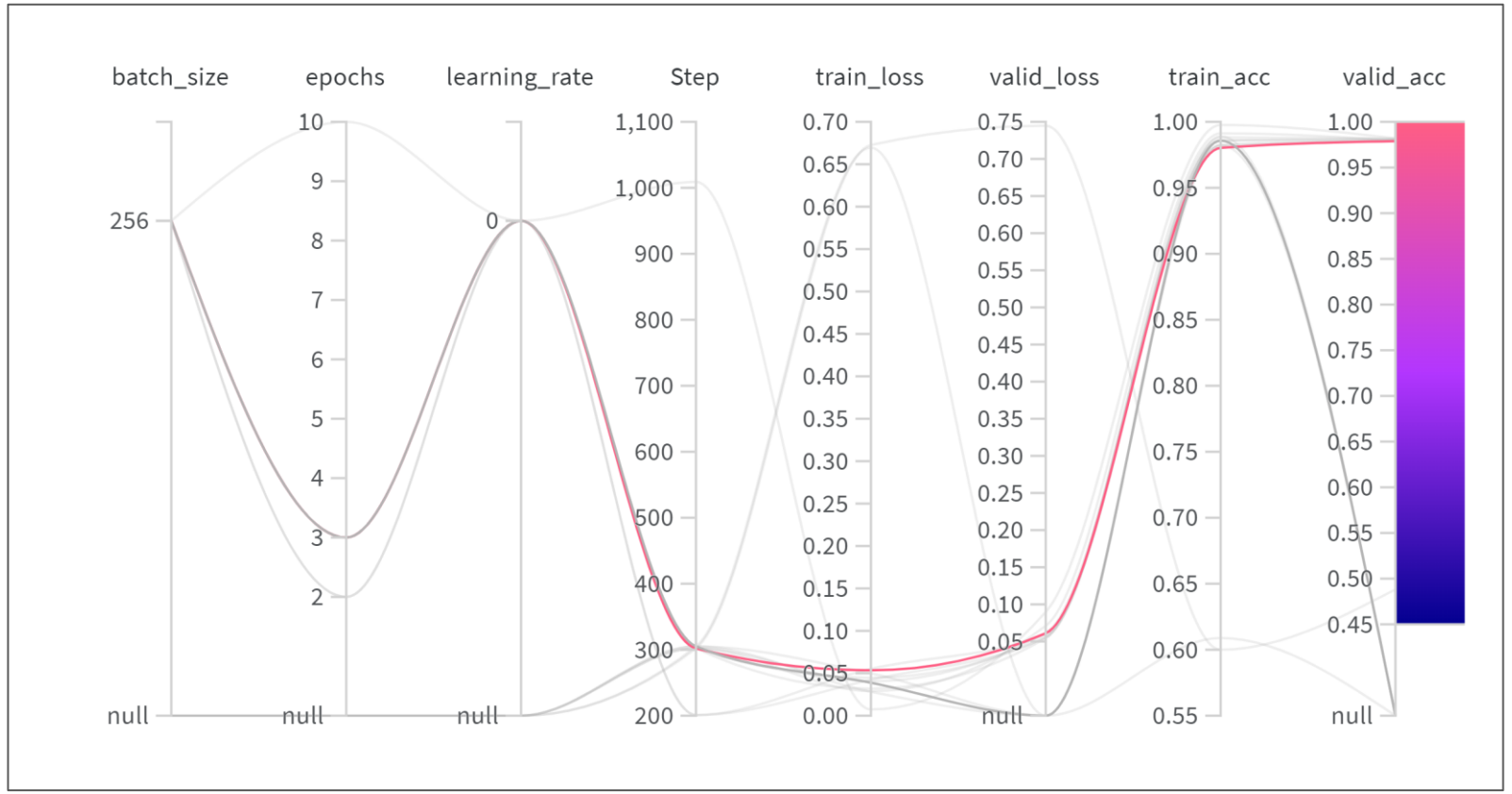


정규화를 하였지만 기존보다 loss acc 두 부분 모두 성능의 향상이 없음

https://wandb.ai/goorm_team_2/project1/reports/Regularization--VmlldzoxNzcyNjEy

04 프로젝트 결과
04 fine-tuning

- 정규화 추가 개선 (Train, Train + Valid)

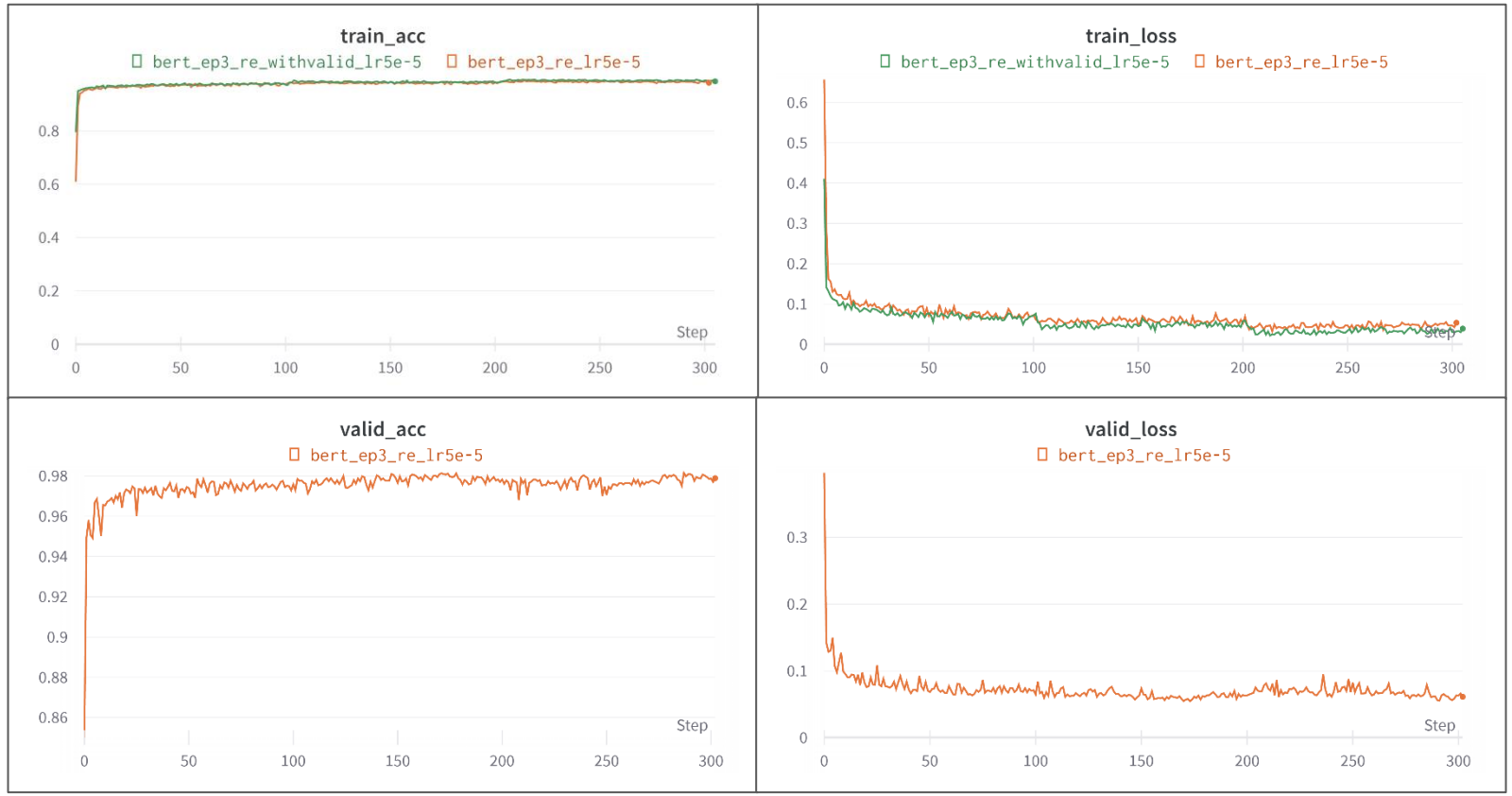


정규화를 통해 불순물을 제거 하고 valid 데이터를 train 데이터에 포함하여 실행

https://wandb.ai/goorm_team_2/project1/reports/Regularization--VmlldzoxNzcyNTk3

04 프로젝트 결과
04 fine-tuning

- 정규화 추가 개선 (Train, Train + Valid)

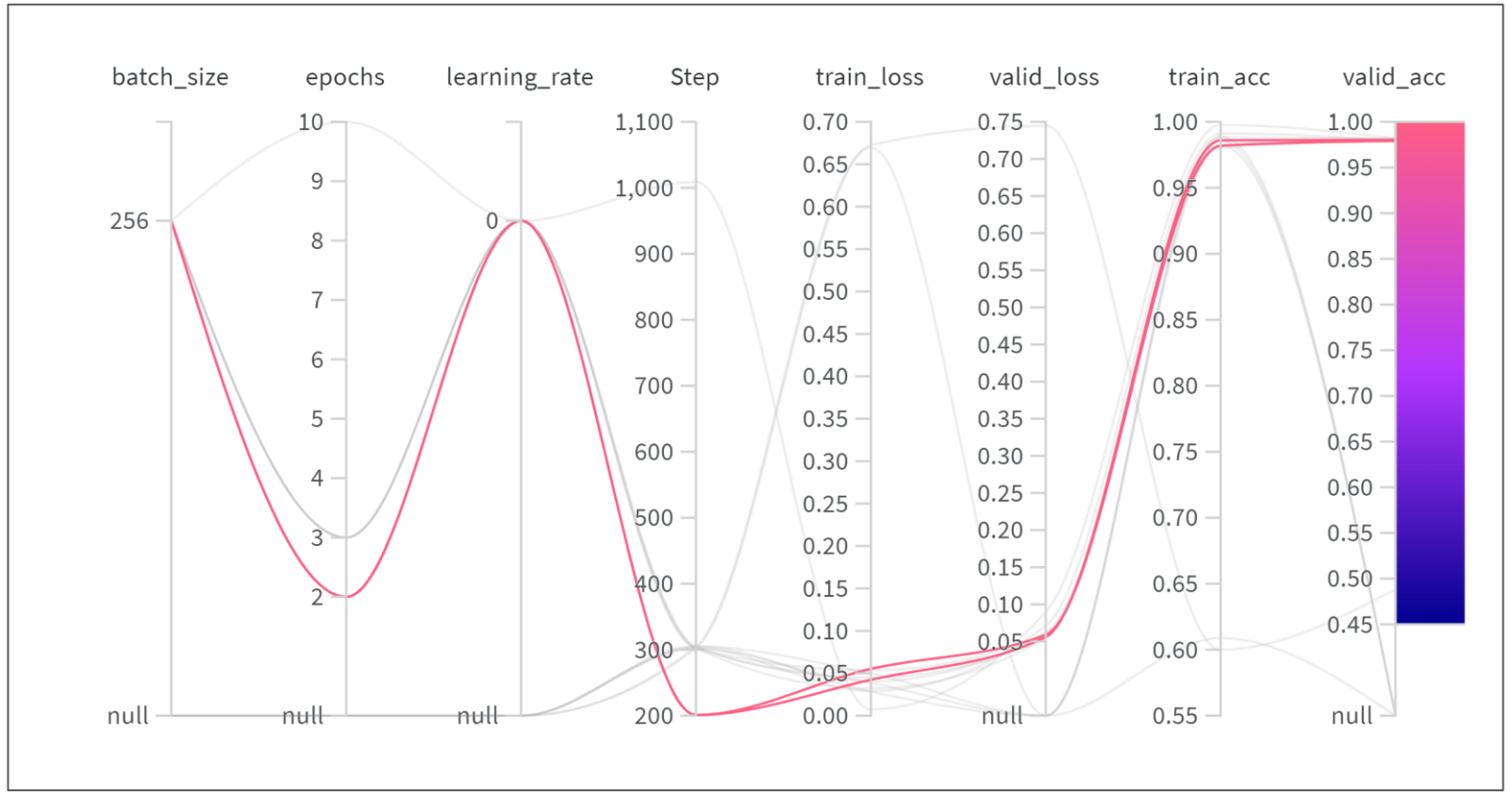


training data에 대해 성능이 향상 됐지만 score는 오르지 않음

https://wandb.ai/goorm_team_2/project1/reports/Regularization--VmlldzoxNzcyNTkw

04 프로젝트 결과
04 fine-tuning

- **XLnet Bert와 비교 (1)**



XLNet 같은 경우 BERT와 데이터 처리 에서의 차이점이 있음(cased)

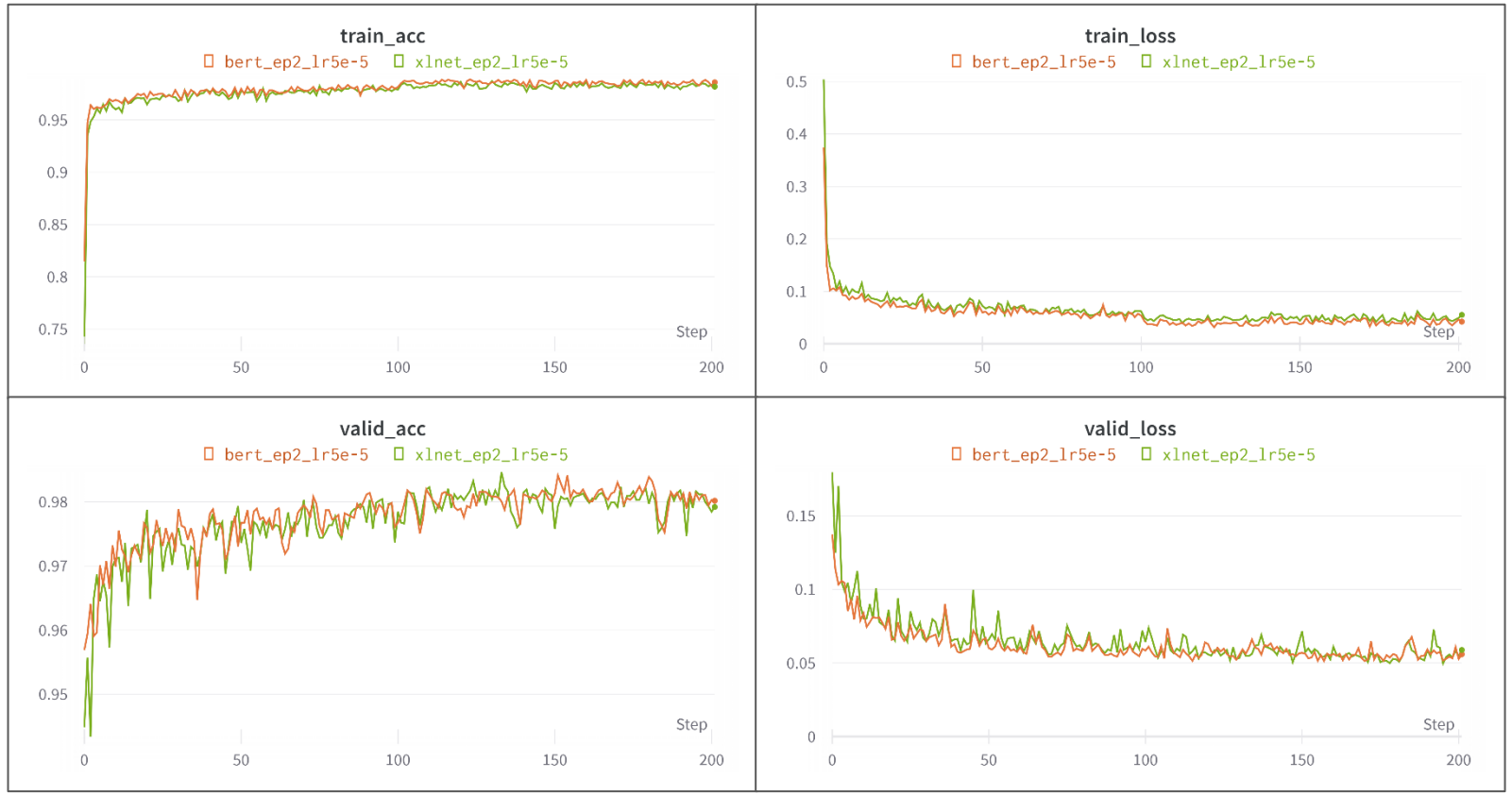
BERT, XLNet 모델을 사용

Epoch 2

https://wandb.ai/goorm_team_2/project1/reports/xlnet-Epoch2--VmlldzoxNzcyNjMx

04 프로젝트 결과
04 fine-tuning

- **Xlnet Bert와 비교 (1)**

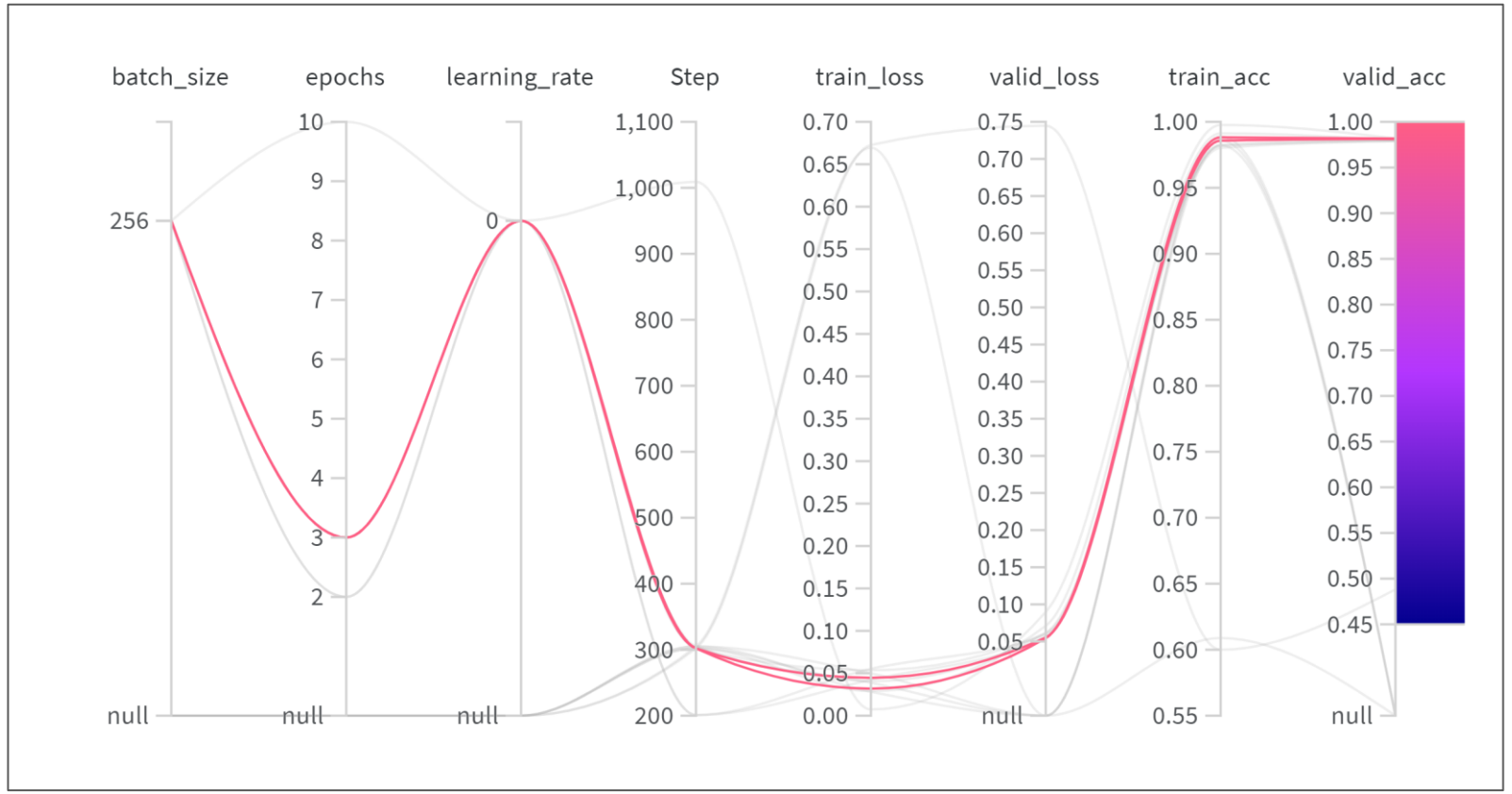


두 모델 다 좋은 성능을 보여 줍니다.
Loss 같은 경우 BERT모델이 약간 우세한 것으로 보입니다.

https://wandb.ai/goorm_team_2/project1/reports/xlnet-Epoch2--VmlldzoxNzcyNjI3

04 프로젝트 결과
04 fine-tuning

- **Xlnet Bert와 비교 (2)**

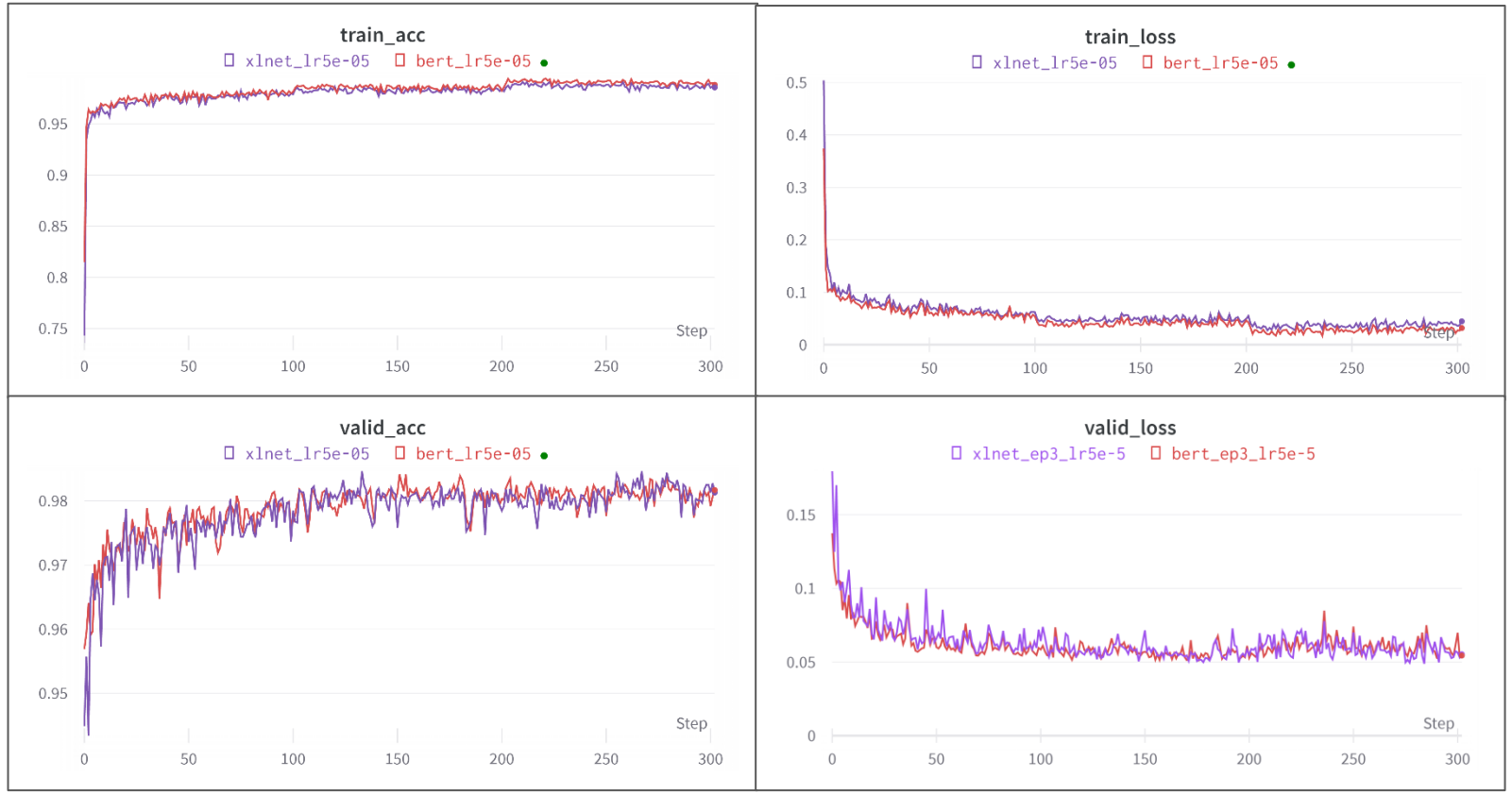


Epoch 3

https://wandb.ai/goorm_team_2/project1/reports/xlnet-Epoch3--VmlldzoxNzcyNjQy

04 프로젝트 결과
04 fine-tuning

- **Xlnet Bert와 비교 (2)**

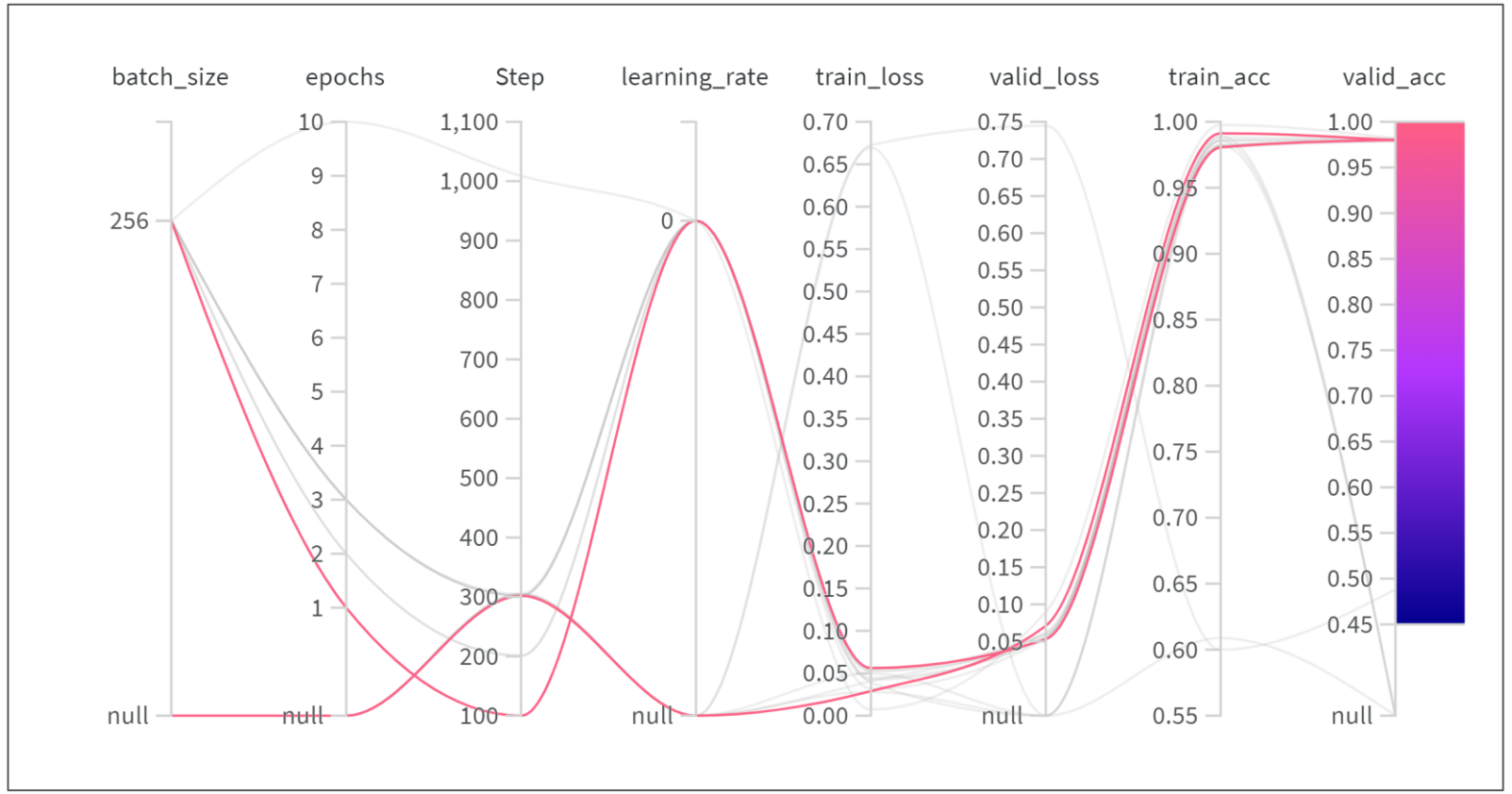


앞의 epoch 2와 비슷한 성능을 보입니다.

https://wandb.ai/goorm_team_2/project1/reports/xlnet-Epoch3--VmlldzoxNzcyNjQx

04 프로젝트 결과
04 fine-tuning

- **Optimizer 변경**

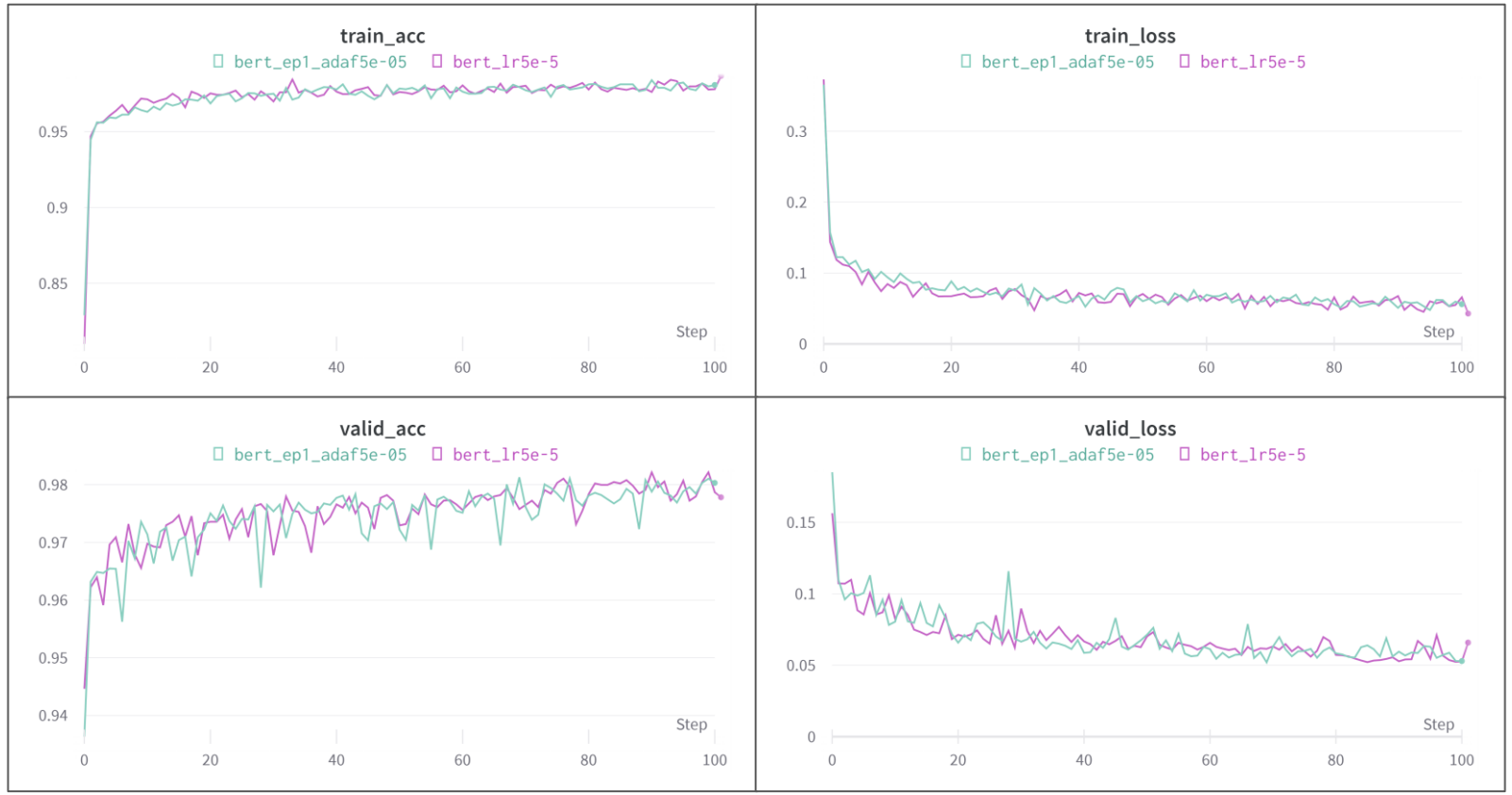


Optimizer를 AdamW 에서 Adafactor로 변경

https://wandb.ai/goorm_team_2/project1/reports/Adafactor--VmlldzoxNzczMDA4

04 프로젝트 결과
04 fine-tuning

- **Optimizer 변경**

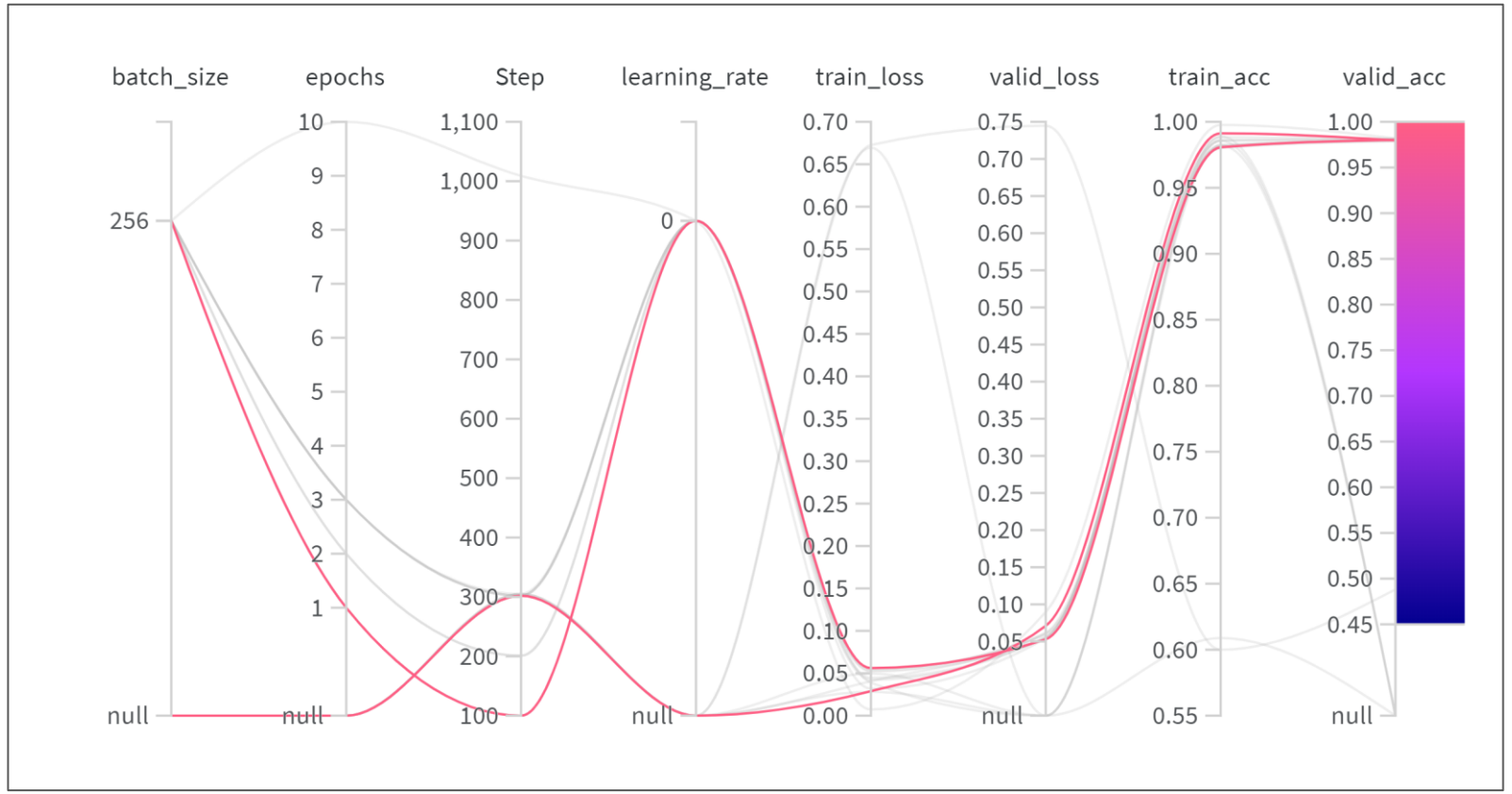


비슷한 성능을 보이지만
Adafactor의 값들이 AdamW보다
불안정한 것으로 보입니다.

https://wandb.ai/goorm_team_2/project1/reports/Adafactor--VmlldzoxNzczMDEy

04 프로젝트 결과
04 fine-tuning

- LR Scheduler(get_cosine_schedule_with_warmup)

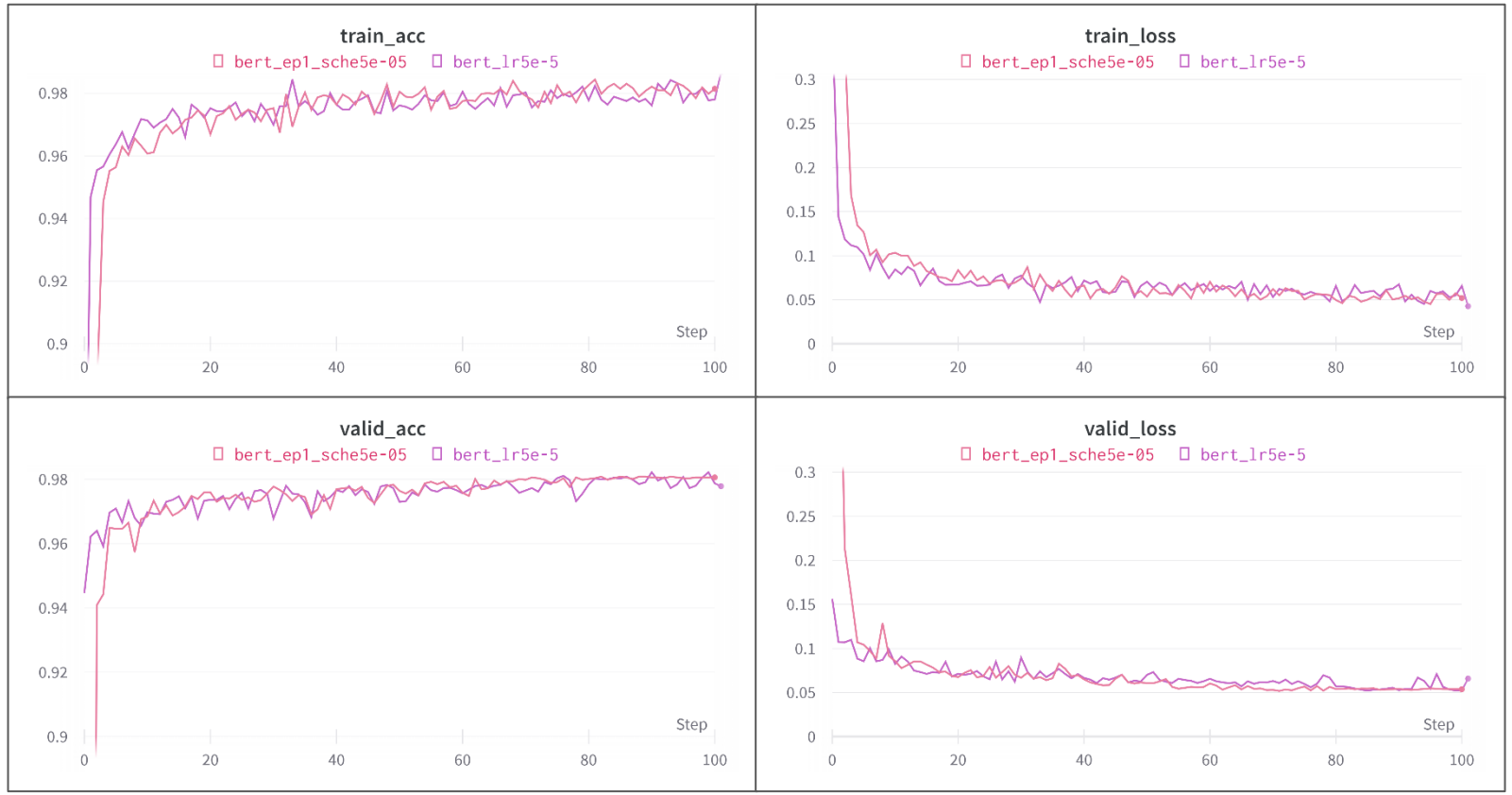


LR Scheduler 추가

https://wandb.ai/goorm_team_2/project1/reports/Scheduler_get_cosine_schedule_with_warmup--VmlldzoxNzczMDMy

04 프로젝트 결과
04 fine-tuning

- **LR Scheduler(get_cosine_schedule_with_warmup)**



Learning Rate가 조절되면서
global optimum에 더 잘 접근

https://wandb.ai/goorm_team_2/project1/reports/Scheduler_get_cosine_schedule_with_warmup--VmlldzoxNzczMDMw

04 프로젝트 결과

04 fine-tuning

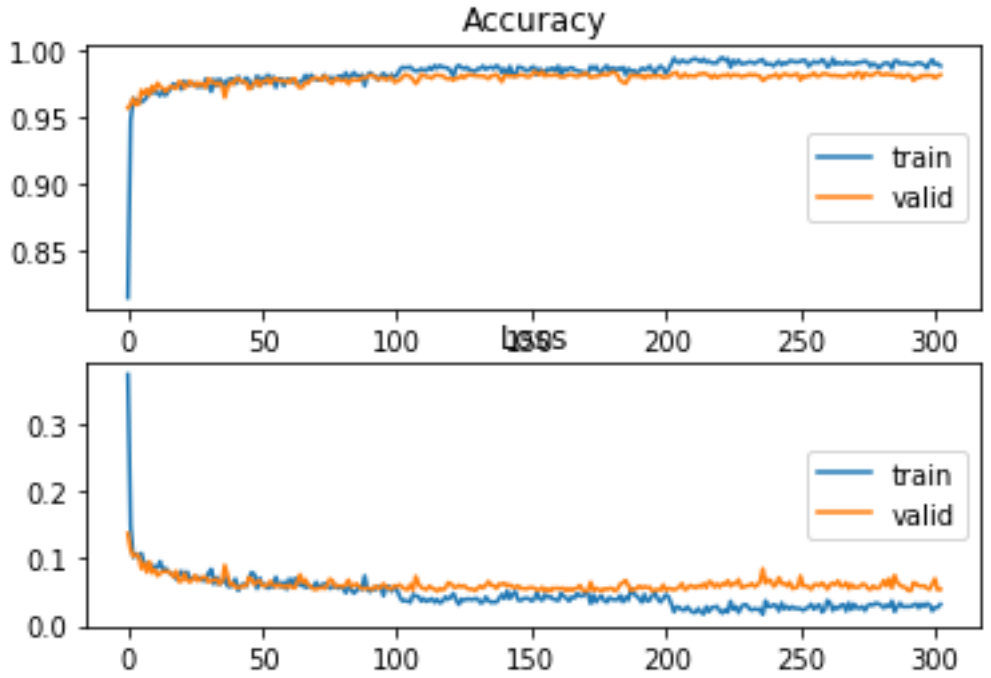
- 최종 모델 Visualization

BERT-base-uncased

Epoch: 3

LR: 5e-5

Accuracy: 98.7



05 자체 평가 및 피드백

05 평가 및 보완점

1. 한정된 GPU 자원으로 더욱 다양한 hyper parameter를 실험해보지 못했으며 Large Model들을 fine-tuning시키지 못했다.
2. 성능 기록을 처음부터 잘 했으면 하는 아쉬움이 있다.
3. 데이터 Augmentation 등 더욱 다양한 기법을 사용하지 못했다.
4. 일정 수준 이상의 정확도를 내지 못했다.
5. Tool 사용법을 익히는데 시간이 예정보다 오래 걸렸다.
6. SVM 알고리즘 등 머신러닝 기법들과 혼합하여 사용하지 못했다.

05 자체 평가 및 피드백

05 조원 간의 피드백

조현수: 프로젝트를 진행하면서 여러가지 프레임워크를 배우고 적용해 볼 수 있었으며 성능 향상을 위해 여러가지 기법을 사용해보는 좋은 경험이었습니다. 프로젝트를 처음부터 진행해보며 배우는게 많았기에 다음 프로젝트에서는 더 수월하게 진행할 것 같습니다.

손동협: 프로젝트를 진행하면서 여러가지 성능향상의 가능성을 보았고 데이터를 내는것에 대한 좋은 툴을 많이 알게 되어 다음 프로젝트에는 더 좋은 결과를 보여 줄것 으로 예상 됩니다.

이준엽: 프로젝트를 진행하면서 어떤 파라미터를 고쳐야 성능이 오를지 많은 고민을 해봤지만 생각보다 좋은 결과를 얻지 못해 아쉽습니다. 하지만 스케쥴러, 스윕등 좋은 툴들을 시도해보고 적용하는데 성공했습니다. 다음 프로젝터는 조금 더 좋은 환경에서 할 수 있기 때문에 더 좋은 성적을 내도록 노력하겠습니다.

김남준: 프로젝트를 진행하는 과정에서 시간과의 싸움이 어떤 것인지 알 수 있었고 기본적인 시각화 방법 외에 다양한 방법을 알 수 있어 새로운 시각에 눈을 뜰 수 있었습니다. 또한 저의 부족한 부분이 무엇인지 알 수 있었고 앞으로 있을 프로젝트에서는 단점을 보완할 수 있을 것입니다.

정명관: 프로젝트를 통해 많은 성능 변화를 보았으며 수업으로만 듣던 내용을 실제로 시각화 해보아서 수업의 이해도를 확인할 수 있는 시간이였습니다

김영수: 프로젝트를 통해 실제 내부에 들어가는 값들을 확인하고, 어떤 원리로 돌아가는지에 대한 파라미터 체킹 및 각 함수별 동작 원리를 보는것이 공부에 도움이 되었으며 시각화 도구를 사용하며 외부 api 활용의 중요성에 대해 다시금 실감할 수 있는 시간이였습니다. 마지막으로 스스로의 부족한 부분을 파악하고, 공부의 방향성을 알게해준 중요한 시간이였습니다.

조승연: 프로젝트를 통해 자연어 처리 프로젝트를 해볼 수 있었다. 이론적으로만 이해하는 것과 실제로 개발하는 과정에서 배우는 농도가 달랐던 만큼 부트캠프가 끝난 이후에도 꾸준한 학습이 필요할 것 같다.

Q & A

Thank You