Comp 7607 Final Project

Created	@December 14, 2023 11:47 PM
Tags	

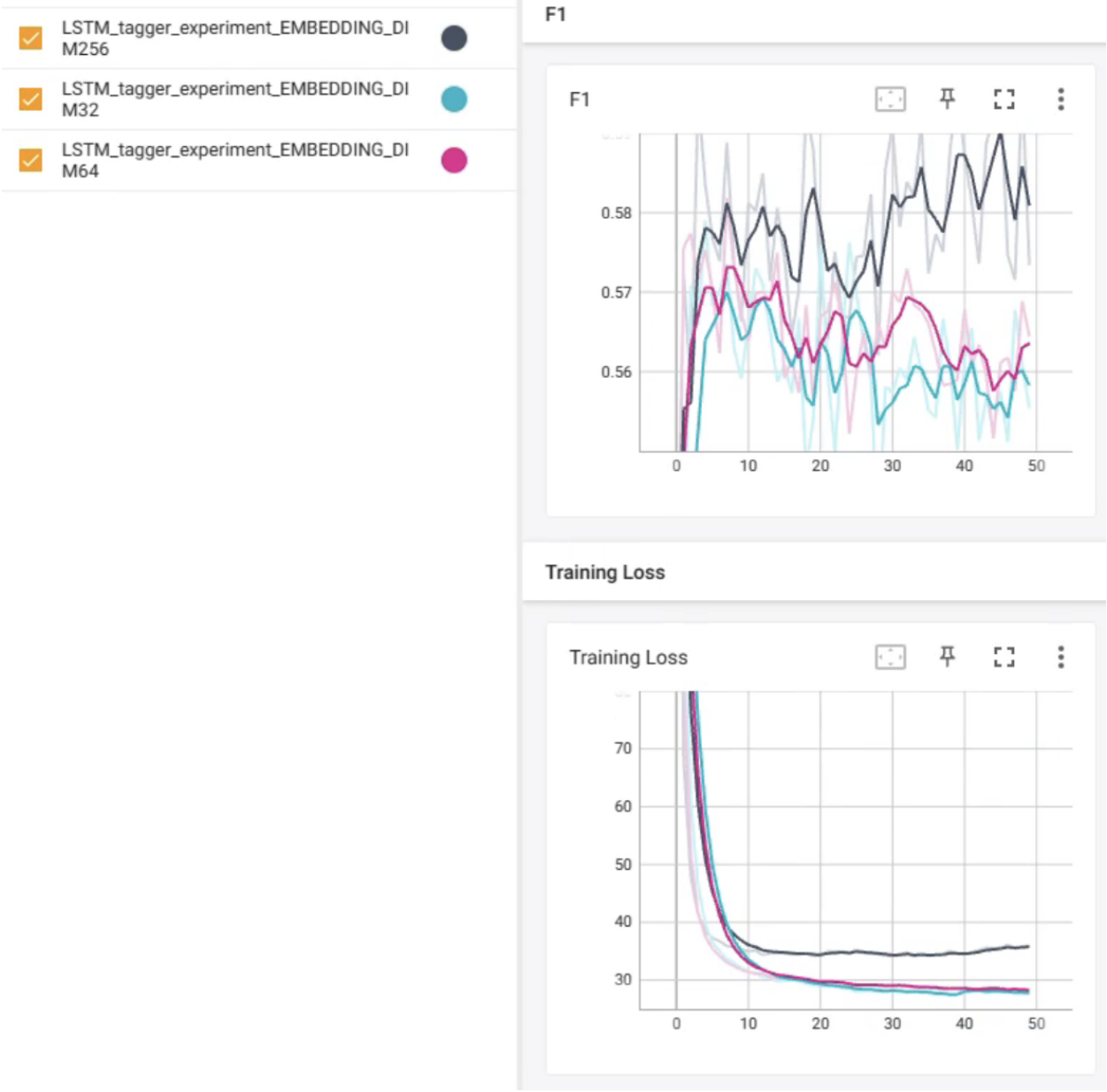
Architecture of Tagger

- Both taggers used the same set of dataloaders, where we read inputs, extract dictionary relations of words-index and tag-index, and batch them for further use.
- Both taggers used cross-entropy loss and Adam optimizer with lr=1e-3
- Both taggers used seqeval's classification_report and tensorboard to recorded loss curve for training loop, and F1 score for validation dataset.
    - detailed guideline on how to check tensorboard data is also in README.md, which is pretty simple:

```
tensorboard --logdir runs_LSTM
# or
tensorboard --logdir runs_Transformer
```

LSTM

- This tagger implements the basic LSTM model from the source provided, where Bidirectional is implemented and layer size doubled to fit the setting
- The ratio of EMBEDDING_DIM and HIDDEN_DIM were 1:4 for 32 and 256 EMBEDDING_DIM, and 1:2 for 64 EMBEDDING_DIM (due to time limit, no further experiment were conducted)



- As dimension goes higher, the predictability of model increases, with a trade-off in training loss fluctuation, which is still acceptable
- Final Prediction F1 score: 0.467

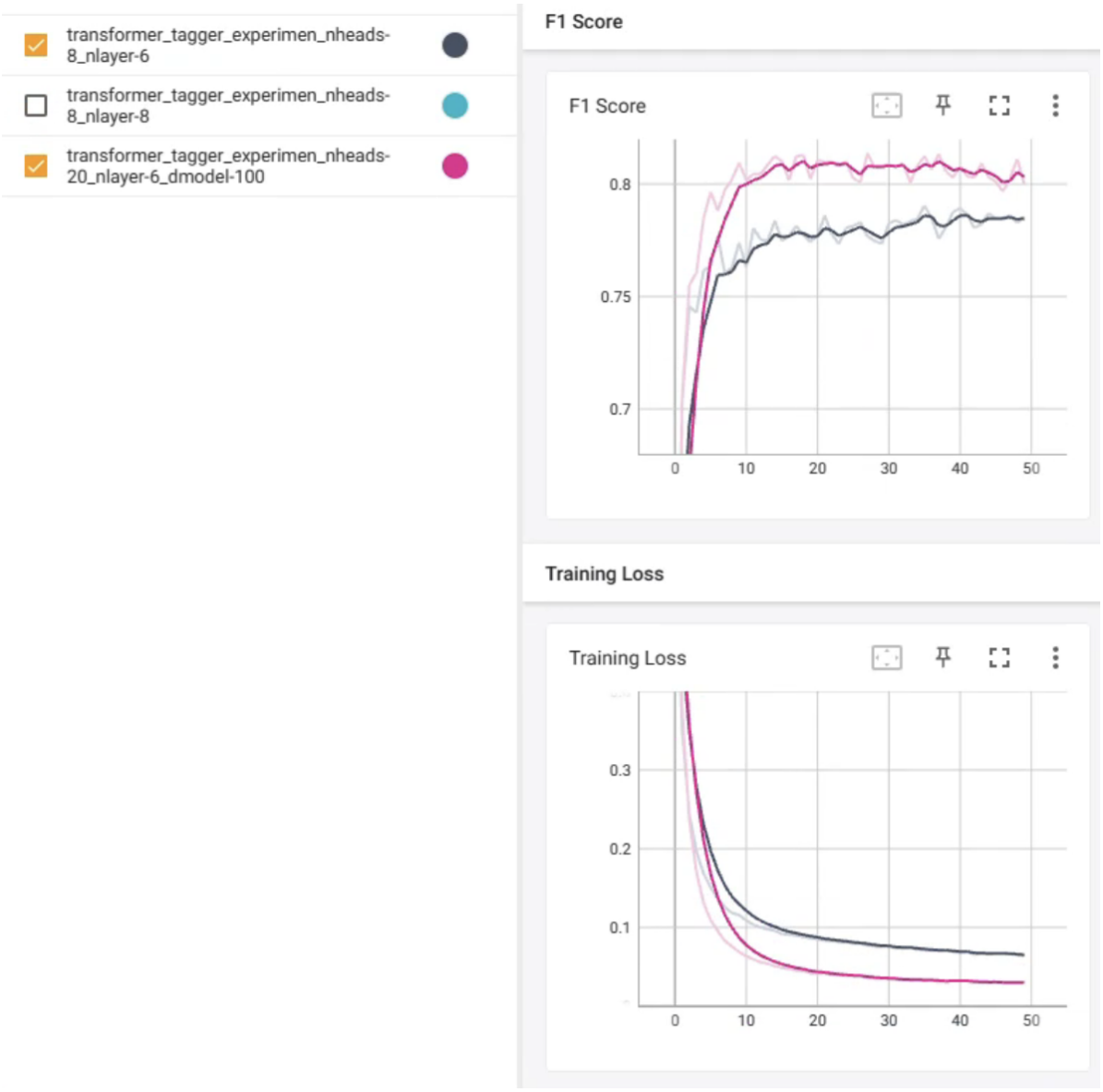
Transformer

- This tagger implements the basic transformer provided in the link, specifically
  - the last linear layer now has len(tags) outputs, instead of n_words

- In particular, to feed our data to the original transformer with [sequence*batch*tagsize] input shape, we transpose the elements in our data.
- Model is initialized with:

```
model = TransformerModel(ntoken=ntokens, d_model=d_model, nhe
                d_hid=d_hid, nlayers=nlayer, tags=len(tag_dic
                dropout=dropout).to(device)
```

- Due to time limit, we only modified n-heads and n-layers, and conducted one set of d_model modifications



    - We can see the overall performance with more heads and less dimension is significantly better
- Final Prediction F1 score: 0.698
- Moreover, there were a series of failed tests given very large d_model and d_hidden, my guess is that model is too complicated for this simple task, and very hard to drive the gradient to converge