Assignment – 2

Part – 2 (Text to Math Program) :

Data PreProcessing :

For the input vocabulary I have taken the word frequency greater than 3 and then mapped them from word to int and int to word.

For the lf vocab, all the words are considered.

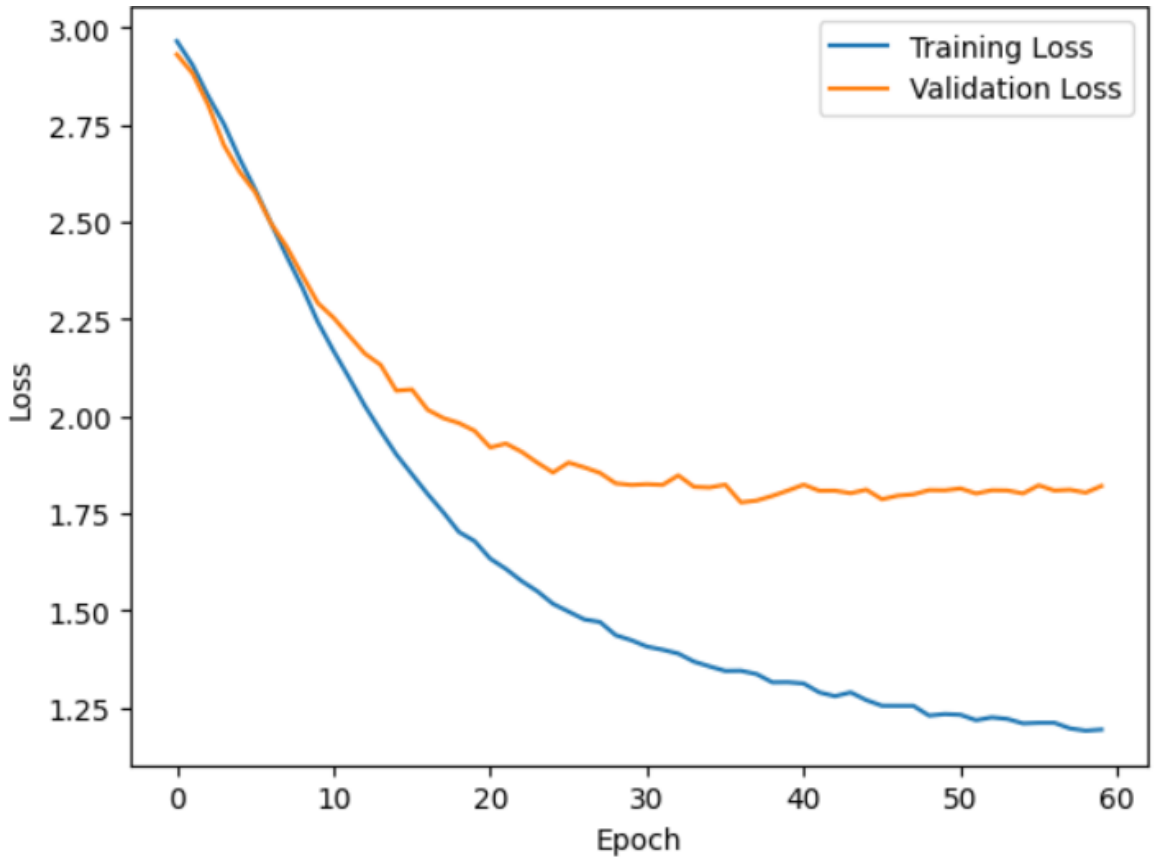
1. Seq2Seq Model with Glove Embeddings using an Bi-LSTM encoder and an LSTM decoder

Hyperparameters :

```
embedding_dim = 300
enc_hidden_size = 512
dec_hidden_size = 512
n_layers = 1
```

```
# Loss and optimizer
criterion = nn.CrossEntropyLoss(ignore_index=0)
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

The training loss and validation curves are as follows :



The execution accuracy is around 30%. The beam search with k=10 is implemented and teacher forcing ratio 0.6

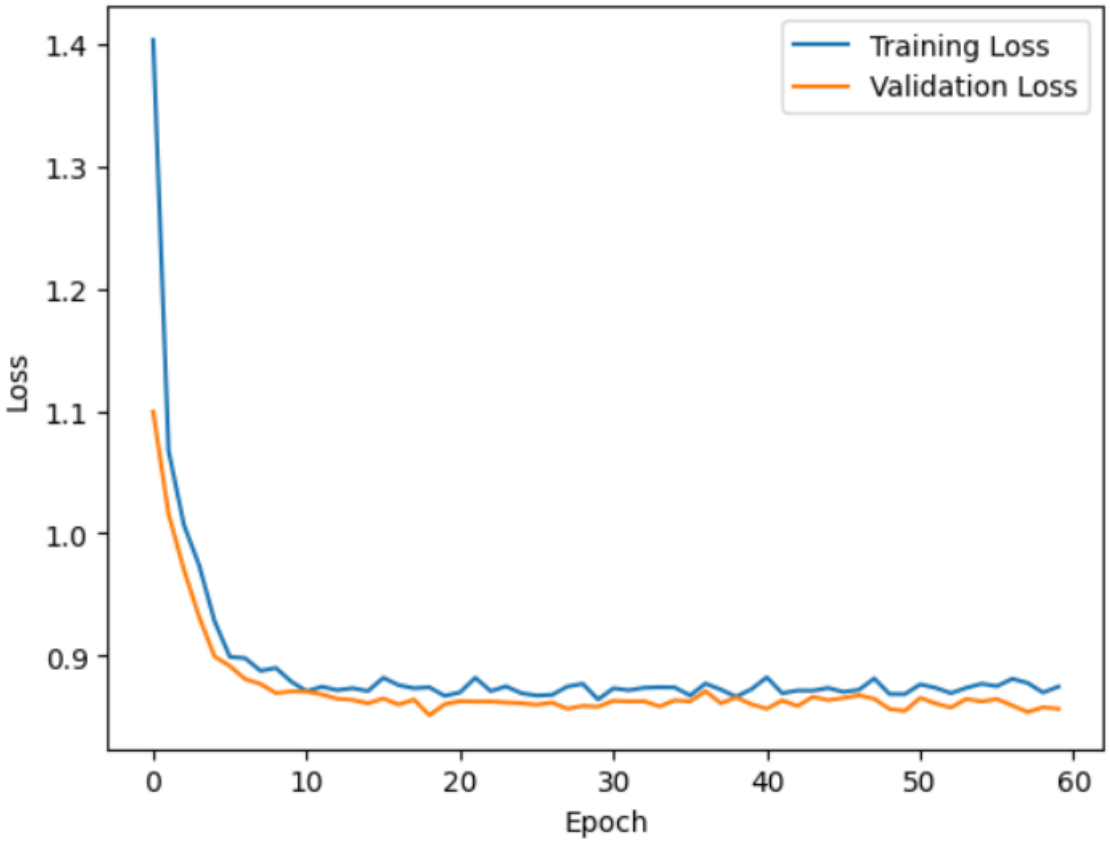
2. A Seq2Seq+Attention model with GloVe embeddings, using an Bi- LSTM encoder and an LSTM decoder

Hyperparameters :

```
embedding_dim = 300
enc_hidden_size = 512
dec_hidden_size = 512
n_layers = 1

encoder = bilstm_encoder(len(prb_word2int), embedding_dim, enc_hidden_size, n_layers).to(device)
decoder = lstm_attn_decoder(len(lf_word2int), embedding_dim, dec_hidden_size, n_layers).to(device)
model = seq2seqAttn(encoder, decoder).to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.1)
```



3. A Seq2Seq+Attention model using a pre-trained frozen BERT-base-cased encoder and an LSTM decoder

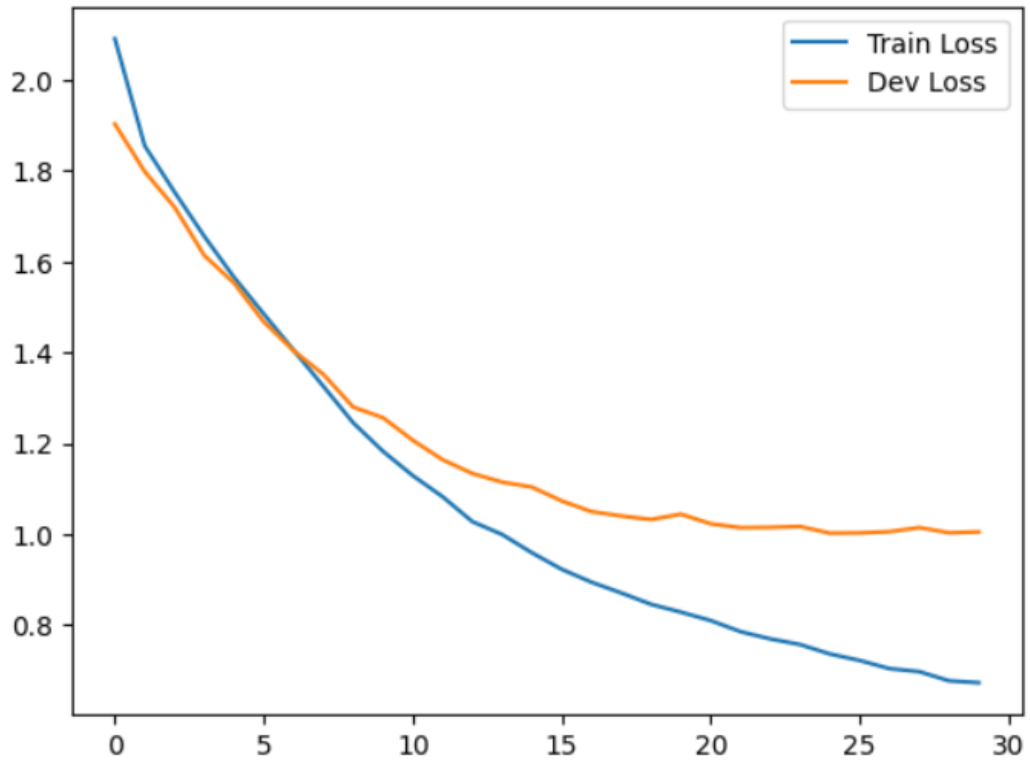
Hyperparameters :

```
emb_dim = 768
hidden_dim = 768
num_layers = 1

# Model
bert_enc = bert_encoder().to(device)
bert_attn = bert_attn_ntwrk(emb_dim, hidden_dim, num_layers).to(device)
lstm_dec = lstm_bert_decoder(len(lf_word2int), emb_dim, hidden_dim, num_layers).to(device)
model = seq2seq_attnbert(bert_enc, bert_attn, lstm_dec).to(device)

# Loss and optimizer
criterion = nn.CrossEntropyLoss(ignore_index=0)
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

The training and validation loss are as follows :



The execution accuracy is around 60% for this.

4. *A Seq2Seq+Attention model using a tuned BERT-base-cased encoder and an LSTM decoder*

```
emb_dim = 768
hidden_dim = 768
num_layers = 1

# Model
bert_enc = bert_encoder().to(device)
bert_attn = bert_attn_ntwrk(emb_dim, hidden_dim, num_layers).to(device)
lstm_dec = lstm_bert_decoder(len(lf_word2int), emb_dim, hidden_dim, num_layers).to(device)
model = seq2seq_attnbert(bert_enc, bert_attn, lstm_dec).to(device)

# Loss and optimizer
criterion = nn.CrossEntropyLoss(ignore_index=0)
optimizer = optim.Adam(model.parameters(), lr=0.001)
```