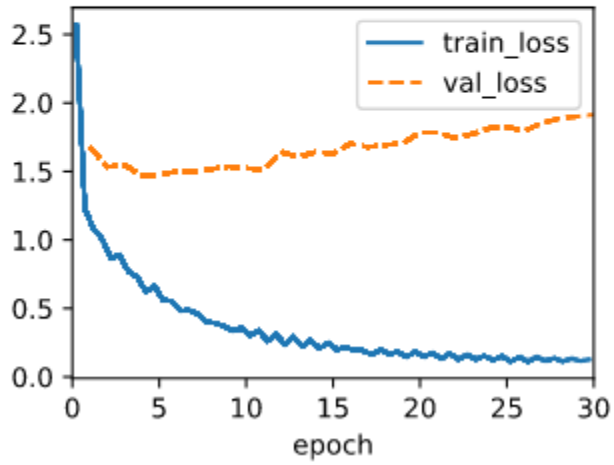
https://github.com/Christian-Martens-UNCC/ECGR-4106/tree/main/Homework_5-Sequence_to_Sequence

Problem 1:

A) Using the training graph on page 27 of lecture 12 as the model to beat, I constructed a list of parameters that could be altered. I created 12 different models each with a unique combination of number of layers, embedding size, and number of hidden states. The values of these parameters were chosen based on some experimentation I did beforehand. As a result, every model that was constructed had a better training and validation loss than the given model. Below are a few of the more interesting models that were generated.

Model 1.1 – 1 Layer, 128 Embedding, 128 Hidden

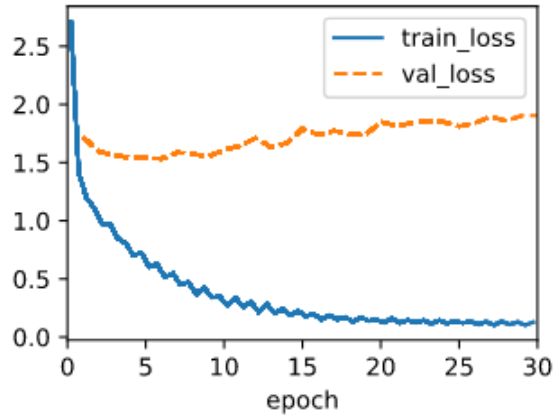
```
Total Training Time : 27.05219 s
Estimated Average Training Time per Epoch : 0.90174 s
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['venez', '<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```



The first model trained. It is much simpler than the model shown in the book. Regardless, using the same parameters other than the ones listed in the figure's title, this model outperformed the book.

Model 1.2 - 1 Layer, 128 Embedding, 256 Hidden

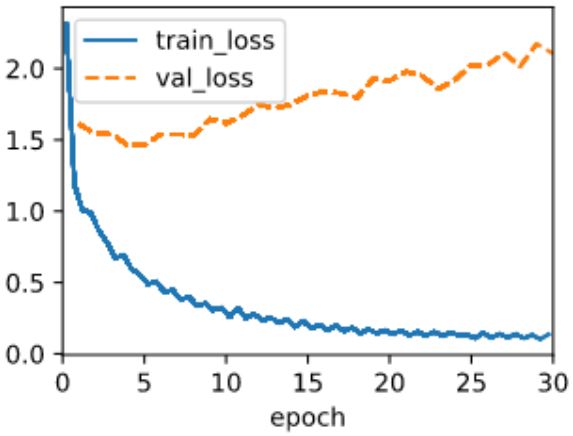
```
Total Training Time : 36.54909 s
Estimated Average Training Time per Epoch : 1.2183 s
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['je', 'suis', 'calme', '.'], bleu,0.537
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```



This was the first model which was able to, although not entirely accurately, have a non-zero Bleu rating for “he’s calm”.

Model 1.7 – 2 Layers, 128 Embedding, 128 Hidden

```
Total Training Time : 32.12674 s
Estimated Average Training Time per Epoch : 1.07089 s
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'mouillé', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

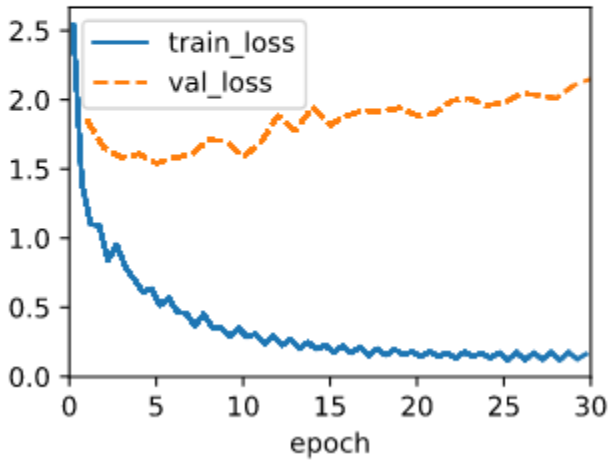


This was the best model that was trained based on training time, error, and qualitative accuracy.

B) Since the Encoder has more layers than the Decoder, the method I used to reduce the dimensionality of the initial state was to only push the states of the last two layers of the Encoder. The resulting model has a better loss graph than the model in the book but a worse loss graph compared to the model that I trained with the same parameters. It did, however, perform better on the qualitative portion than the same-parameter model I trained.

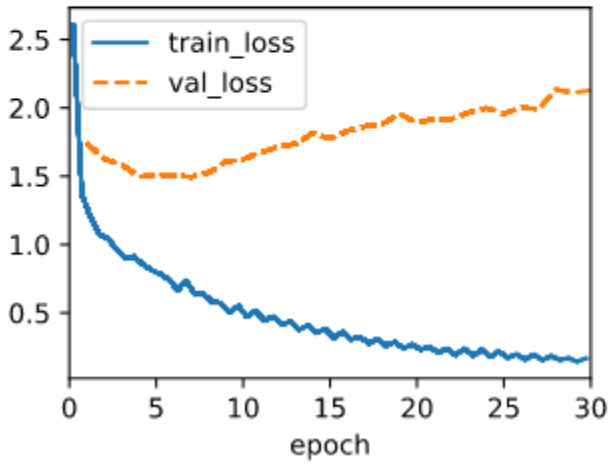
```
class Seq2SeqEncoder_2(d2l.Encoder):
    """The RNN encoder for sequence to sequence learning.

    Defined in :numref:`sec_seq2seq`"""
    def __init__(self, vocab_size, embed_size, num_hiddens, num_layers,
                 dropout=0):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.rnn = d2l.GRU(embed_size, num_hiddens, num_layers, dropout)
        self.apply(init_seq2seq)

    def forward(self, X, *args):
        # X shape: (batch_size, num_steps)
        embs = self.embedding(d2l.astype(d2l.transpose(X), d2l.int64))
        # embs shape: (num_steps, batch_size, embed_size)
        outputs, state = self.rnn(embs)
        # outputs shape: (num_steps, batch_size, num_hiddens)
        # state shape: (num_layers, batch_size, num_hiddens)
        return outputs, state[-2:][:][:]
```

```
Total Training Time : 64.27806 s
Estimated Average Training Time per Epoch : 2.1426 s
go . => ['<unk>', '!'], bleu,0.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'venu', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```



C) Much like the previous homework, the GRU model seemed to outperform the LSTM model. It was more accurate, less validation loss, and had a faster training time.

```
Total Training Time : 77.27733 s
Estimated Average Training Time per Epoch : 2.57591 s
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['<unk>', 'ceci', '.'], bleu,0.000
i'm home . => ['je', 'suis', '<unk>', '.'], bleu,0.512
```

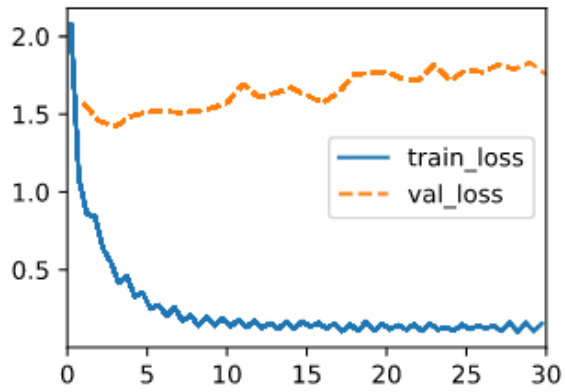


Problem 2:

A) Using the Encoder-Decoder with Bahdanau Attention, I was able to fully construct all four of the requested models with each having a different number of layers. Generally speaking, I found that increasing the number of layers did not provide any substantive increase in accuracy for these models. Additionally, the added training time to produce the models of higher layer counts leads me to recommend that only a single layer be used for this dataset.

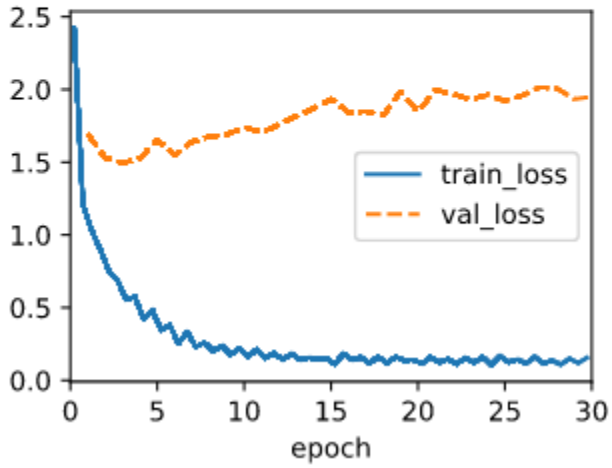
```
model2_1 = train_enc_attdec(num_layers=1)
```

```
Total Training Time : 68.56555 s
Estimated Average Training Time per Epoch : 2.28552 s
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['soyez', 'calme', '.'], bleu,0.492
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```



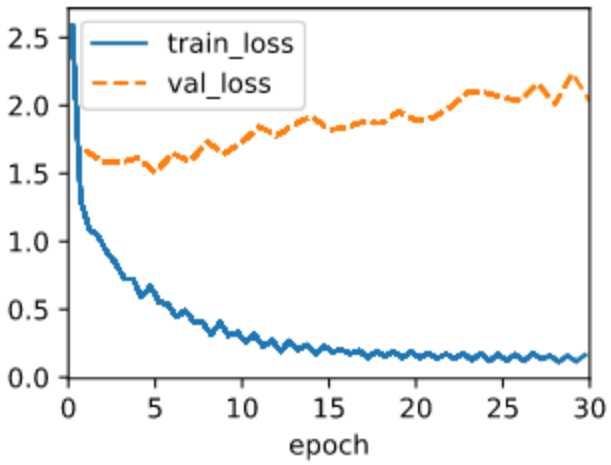

```
model2_2 = train_enc_attdec(num_layers=2)
```

```
Total Training Time : 85.56423 s
Estimated Average Training Time per Epoch : 2.85214 s
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['soyez', 'calmes', '!'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```



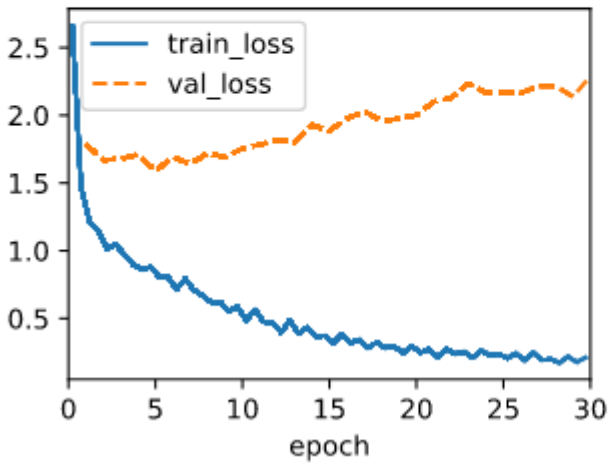
In [83]:
```
model2_3 = train_enc_attdec(num_layers=3)
```

```
Total Training Time : 112.20182 s
Estimated Average Training Time per Epoch : 3.74006 s
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['il', 'est', 'mouillé', '.'], bleu,0.658
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```




```
model2_4 = train_enc_attdec(num_layers=4)
```

```
Total Training Time : 128.83769 s
Estimated Average Training Time per Epoch : 4.29459 s
go . => ['pars', '!'], bleu,0.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'porte', 'bien', '.'], bleu,0.548
```



B) Unfortunately, despite many hours of attempted debugging, I could not get the LSTM to run on the Encoder-Decoder with Attention setup. However, considering my previous experience with GRU vs LSTM models, I would suspect that the LSTM model does not provide a significant benefit to the model and only adds to the training time.