Model analysis:

我所使用的 model 定義如下

```python
class CharRNN(torch.nn.Module):
    def __init__(self, vocab_size, embed_dim, hidden_dim):
        super(CharRNN, self).__init__()

        # Embedding層
        self.embedding = torch.nn.Embedding(num_embeddings=vocab_size,
                                            embedding_dim=embed_dim,
                                            padding_idx=tokenizer.stoi['.'])

        # RNN層
        self.rnn_layer1 = torch.nn.RNN(input_size=embed_dim,
                                       hidden_size=hidden_dim,
                                       batch_first=True)
        self.rnn_layer2 = torch.nn.RNN(input_size=hidden_dim,
                                       hidden_size=hidden_dim,
                                       batch_first=True)
        # output層
        self.linear = torch.nn.Sequential(torch.nn.Linear(in_features=hidden_dim,
                                                          out_features=hidden_dim),
                                          torch.nn.ReLU(),
                                          torch.nn.Linear(in_features=hidden_dim,
                                                          out_features=vocab_size))
    def forward(self, batch_x):
        batch_x = self.encoder(batch_x)
        batch_x = self.linear(batch_x)
        return batch_x

    def encoder(self, batch_x):
        batch_x = self.embedding(batch_x)
        batch_x , _ = self.rnn_layer1(batch_x)
        batch_x , _ = self.rnn_layer2(batch_x)
        return batch_x
```

Loss function:

```python
criterion = torch.nn.CrossEntropyLoss( reduction='mean')
```

備註：雖然助教在 smaple code 裡面 padding 並沒有計算 loss，但是我自己實際測試的結果還是 padding 還是需要計算 loss，否則模型就算 loss 有在下降，訓練結果還是很糟

Tokenizer:

```python
class Tokenizer():
    def __init__(self,tokens , pad , eos , sos):
        self.tokens = [pad , eos , sos] + list(tokens)
        self.stoi = {ch:i for i,ch in enumerate(self.tokens)}
        self.itos = {i:ch for i,ch in enumerate(self.tokens)}
    def encoder(self , string):
        return [self.stoi[s] for s in string]
    def decoder(self, idx):
        return ''.join([self.itos[i] for i in idx])
    def token_len(self):
        return len(self.tokens)
```

```python
tokenizer = Tokenizer('0123456789-+=()' , pad='.' , eos='_' , sos=':')
tokenizer.decoder(tokenizer.encoder(':0123456'))
```

```
':0123456'
```

在這裡我定義了一個 sos:start of sequence 等等會用到。

我的 tokenizer 很簡單，就是將字元跟數字在換。

Training , validation result:

```
Training epoch 1: 100%|██████████| 3157/3157 [00:43<00:00, 71.96it/s, loss=0.121]
Testing epoch 1: 100%|██████████| 790/790 [00:10<00:00, 73.23it/s, loss=0.138]
Training epoch 2: 100%|██████████| 3157/3157 [00:43<00:00, 72.09it/s, loss=0.116]
Testing epoch 2: 100%|██████████| 790/790 [00:11<00:00, 71.43it/s, loss=0.128]
Training epoch 3: 100%|██████████| 3157/3157 [00:43<00:00, 72.15it/s, loss=0.0973]
Testing epoch 3: 100%|██████████| 790/790 [00:11<00:00, 71.70it/s, loss=0.103]
Training epoch 4: 100%|██████████| 3157/3157 [00:43<00:00, 72.50it/s, loss=0.0925]
Testing epoch 4: 100%|██████████| 790/790 [00:10<00:00, 72.46it/s, loss=0.107]
Training epoch 5: 100%|██████████| 3157/3157 [00:43<00:00, 72.72it/s, loss=0.0786]
Testing epoch 5: 100%|██████████| 790/790 [00:10<00:00, 72.95it/s, loss=0.0685]
```

Output 結果:

Expression:　　predict:　　　　　　　　answer:　　　　　　correct:

```
45+46-16= .........79_........ .........75_........ False
35-0+20= ........54_......... ........55_......... False
44+9-15= ........37_......... ........38_......... False
8+6+1= ......15_........... ......15_........... True
13+35+1= ........49_......... ........49_......... True
47+31-12= .........68_........ .........66_........ False
26+12-31= .........7_......... .........7_......... True
36-20+46= .........62_........ .........62_........ True
0+5-47= .......-42_......... .......-42_......... True
12-29-22= .........-40_....... .........-39_....... False
42-17+5= ........30_......... ........30_......... True
13-26-5= ........-18_........ ........-18_........ True
16+5-24= ........-3_......... ........-3_......... True
42+19+19= .........81_........ .........80_........ False
37+20+12= .........70_........ .........69_........ False
1-6-7= ......-13_.......... ......-12_.......... False
5-4-3= ......-3_........... ......-2_........... False
15-41-8= ........-33_........ ........-34_........ False
35+20+12= .........69_........ .........67_........ False
20-3-47= ........-30_........ ........-30_........ True
0.45
```

Dataset analysis:

我將助教給的 dataset 裡面包含*()的算式全部刪掉

```
data = data[data['src'].apply(lambda x:'*' not in x)]
data = data[data['src'].apply(lambda x:'/' not in x)]
data = data[data['src'].apply(lambda x:'(' not in x)]
```

然後我將 tgt 的左側塞入了跟 src 長度一樣的 padding

在這裡我的 padding 、 eos 定義成 “.” 、 “_”

```python
def rshift(row):
    return ''.join(['.' for _ in range(len(row['src']))]) + row['tgt']
data['tgt'] = data.apply(rshift, axis=1)
data['src'] = data['src'].apply(lambda x : x + '_')
data
```

✓ 3.2s

	src	tgt
0	0+0=_	....0_
1	0-0=_	....0_
15	0+0+0=_	......0_
16	0-0-0=_	......0_
18	0+0-0=_	......0_
...	...	...
2632476	49-49+48=_	.........48_
2632491	49+49+49=_	.........147_
2632492	49-49-49=_	.........-49_
2632494	49+49-49=_	.........49_
2632497	49-49+49=_	.........49_

然後將 src 、 tgt 的長度不足 20 的部分補上 padding，讓 scr、tgt 長度皆為 20

	src	tgt
0	0+0=_...............	....0_..............
1	0-0=_...............	....0_..............
15	0+0+0=_.............	......0_............
16	0-0-0=_.............	......0_............
18	0+0-0=_.............	......0_............
...	...	...
2632476	49-49+48=_..........	.........48_........
2632491	49+49+49=_..........	.........147_.......
2632492	49-49-49=_..........	.........-49_.......
2632494	49+49-49=_..........	.........49_........
2632497	49-49+49=_..........	.........49_........

結果是模型的 output 跟 tgt 類似

```
45+46-16= .........79_........ .........75_........ False
35-0+20= ........54_......... ........55_......... False
44+9-15= ........37_......... ........38_......... False
8+6+1= ......15_........... ......15_........... True
13+35+1= ........49_......... ........49_......... True
47+31-12= .........68_........ .........66_........ False
26+12-31= .........7_......... .........7_......... True
36-20+46= .........62_........ .........62_........ True
0+5-47= .......-42_......... .......-42_......... True
12-29-22= .........-40_....... .........-39_....... False
42-17+5= ........30_......... ........30_......... True
13-26-5= ........-18_........ ........-18_........ True
16+5-24= ........-3_......... ........-3_......... True
42+19+19= .........81_........ .........80_........ False
37+20+12= .........70_........ .........69_........ False
1-6-7= ......-13_.......... ......-12_.......... False
5-4-3= ......-3_........... ......-2_........... False
15-41-8= ........-33_........ ........-34_........ False
35+20+12= .........69_........ .........67_........ False
20-3-47= ........-30_........ ........-30_........ True
0.45
```

準確度約為 0.45

Another dataset:

如果將有()的算式保留結果如下並且讓模型 fine-tune 這些算式

```
38+39-7= ........68_......... ........70_......... False
19-41+49= .........28_........ .........27_........ False
44+30-26= .........48_........ .........48_........ True
48-30-15= .........4_......... .........3_......... False
0-(5+12)= .........-17_....... .........-17_....... True
22+(22-22)= ...........22_...... ...........22_...... True
(44-16)+42= ...........68_...... ...........70_...... False
(46+48)-9= ..........85_....... ..........85_....... True
1+29+13= ........42_......... ........43_......... False
39-(20+16)= ...........3_....... ...........3_....... True
(49-8)+20= ..........60_....... ..........61_....... False
(3-42)+10= ..........-28_...... ..........-29_...... False
27+11+40= .........78_........ .........78_........ True
26-(6+49)= ..........-28_...... ..........-29_...... False
37-29+26= .........35_........ .........34_........ False
42-(8+31)= ..........4_........ ..........3_........ False
(21-9)+35= ..........47_....... ..........47_....... True
(8+9)-11= .........7_......... .........6_......... False
6-41-2= .......-35_......... .......-37_......... False
31-30+16= .........17_........ .........17_........ True
0.4
```

那麼原先約 0.45 的準確度會下降至約 0.4

這個結果其實合理，因為如果要考慮()那難度的確會上升，所以準確度也會下降

Discussion:

Learning rate comparision:

當 learing rage 為 0.001 且 batch_size 為 128 時訓練的 loss 下降情況

```
Training epoch 1: 100%|██████████| 6282/6282 [01:26<00:00, 72.98it/s, loss=0.127]
Testing epoch 1: 100%|██████████| 1571/1571 [00:21<00:00, 72.25it/s, loss=0.139]
Training epoch 2: 100%|██████████| 6282/6282 [01:25<00:00, 73.21it/s, loss=0.104]
Testing epoch 2: 100%|██████████| 1571/1571 [00:21<00:00, 72.22it/s, loss=0.103]
Training epoch 3: 100%|██████████| 6282/6282 [01:25<00:00, 73.51it/s, loss=0.0939]
Testing epoch 3: 100%|██████████| 1571/1571 [00:21<00:00, 72.05it/s, loss=0.0936]
Training epoch 4: 100%|██████████| 6282/6282 [01:27<00:00, 71.99it/s, loss=0.0838]
Testing epoch 4: 100%|██████████| 1571/1571 [00:21<00:00, 72.49it/s, loss=0.0743]
Training epoch 5: 100%|██████████| 6282/6282 [01:25<00:00, 73.49it/s, loss=0.0765]
Testing epoch 5: 100%|██████████| 1571/1571 [00:22<00:00, 70.94it/s, loss=0.0798]
```

Output:

```
38+39-7= ........68_......... ........70_......... False
19-41+49= .........28_........ .........27_........ False
44+30-26= .........48_........ .........48_........ True
48-30-15= .........4_......... .........3_......... False
0-(5+12)= .........-17_....... .........-17_....... True
22+(22-22)= ...........22_...... ...........22_...... True
(44-16)+42= ...........68_...... ...........70_...... False
(46+48)-9= ..........85_....... ..........85_....... True
1+29+13= ........42_......... ........43_......... False
39-(20+16)= ...........3_....... ...........3_....... True
(49-8)+20= ..........60_....... ..........61_....... False
(3-42)+10= ..........-28_...... ..........-29_...... False
27+11+40= .........78_........ .........78_........ True
26-(6+49)= ..........-28_...... ..........-29_...... False
37-29+26= .........35_........ .........34_........ False
42-(8+31)= ..........4_........ ..........3_........ False
(21-9)+35= ..........47_....... ..........47_....... True
(8+9)-11= .........7_......... .........6_......... False
6-41-2= .......-35_......... .......-37_......... False
31-30+16= .........17_........ .........17_........ True
0.4
```

當 learning rate 為 0.1 , batch_size 為 128 時: (learning rate 為原本的 100 倍)

```
Training epoch 1: 100%|██████████| 6282/6282 [01:24<00:00, 74.11it/s, loss=0.385]
Testing epoch 1: 100%|██████████| 1571/1571 [00:21<00:00, 71.41it/s, loss=0.393]
Training epoch 2: 100%|██████████| 6282/6282 [01:24<00:00, 74.10it/s, loss=0.669]
Testing epoch 2: 100%|██████████| 1571/1571 [00:21<00:00, 71.94it/s, loss=0.677]
Training epoch 3: 100%|██████████| 6282/6282 [01:25<00:00, 73.69it/s, loss=0.666]
Testing epoch 3: 100%|██████████| 1571/1571 [00:21<00:00, 72.33it/s, loss=0.682]
Training epoch 4: 100%|██████████| 6282/6282 [01:27<00:00, 71.41it/s, loss=0.683]
Testing epoch 4: 100%|██████████| 1571/1571 [00:23<00:00, 66.80it/s, loss=0.671]
Training epoch 5: 100%|██████████| 6282/6282 [01:30<00:00, 69.41it/s, loss=0.684]
Testing epoch 5: 100%|██████████| 1571/1571 [00:23<00:00, 66.91it/s, loss=0.682]
```

Output:

```
38+39-7= .................... ........70_......... False
19-41+49= .................... .........27_........ False
44+30-26= .................... .........48_........ False
48-30-15= .................... .........3_......... False
0-(5+12)= .................... .........-17_....... False
22+(22-22)= .................... ...........22_...... False
(44-16)+42= .................... ...........70_...... False
(46+48)-9= .................... ..........85_....... False
1+29+13= .................... ........43_......... False
39-(20+16)= .................... ...........3_....... False
(49-8)+20= .................... ..........61_....... False
(3-42)+10= .................... ..........-29_...... False
27+11+40= .................... .........78_........ False
26-(6+49)= .................... ..........-29_...... False
37-29+26= .................... .........34_........ False
42-(8+31)= .................... ..........3_........ False
(21-9)+35= .................... ..........47_....... False
(8+9)-11= .................... .........6_......... False
6-41-2= .................... .......-37_......... False
31-30+16= .................... .........17_........ False
0.0
```

模型的訓練效果很差，原因是因為 learning rate 太大，以至於模型無法下降到 minimum point(optimizer 盡力了)，不過模型至少還是知道了 padding 是最常出現的。

Batch comparision: (learning rate 與一開始一樣為 0.001，把 batch size 改成 10 倍)

當 learing rage 為 0.001 且 batch_size 為 1280 時訓練的 loss 下降情況

```
Training epoch 1: 100%|██████████| 629/629 [01:08<00:00,  9.17it/s, loss=0.153]
Testing epoch 1: 100%|██████████| 158/158 [00:19<00:00,  8.25it/s, loss=0.187]
Training epoch 2: 100%|██████████| 629/629 [01:08<00:00,  9.17it/s, loss=0.12]
Testing epoch 2: 100%|██████████| 158/158 [00:19<00:00,  8.29it/s, loss=0.126]
Training epoch 3: 100%|██████████| 629/629 [01:10<00:00,  8.90it/s, loss=0.133]
Testing epoch 3: 100%|██████████| 158/158 [00:19<00:00,  8.27it/s, loss=0.112]
Training epoch 4: 100%|██████████| 629/629 [01:08<00:00,  9.15it/s, loss=0.105]
Testing epoch 4: 100%|██████████| 158/158 [00:19<00:00,  8.22it/s, loss=0.0995]
Training epoch 5: 100%|██████████| 629/629 [01:10<00:00,  8.88it/s, loss=0.113]
Testing epoch 5: 100%|██████████| 158/158 [00:19<00:00,  8.26it/s, loss=0.0924]
```

Output:

```
38+39-7= ........76_......... ........70_......... False
19-41+49= .........26_........ .........27_........ False
44+30-26= .........46_........ .........48_........ False
48-30-15= .........3_......... .........3_......... True
0-(5+12)= .........-17_....... .........-17_....... True
22+(22-22)= ...........21_...... ...........22_...... False
(44-16)+42= ...........66_...... ...........70_...... False
(46+48)-9= ..........84_....... ..........85_....... False
1+29+13= ........44_......... ........43_......... False
39-(20+16)= ...........3_....... ...........3_....... True
(49-8)+20= ..........66_....... ..........61_....... False
(3-42)+10= ..........-39_...... ..........-29_...... False
27+11+40= .........76_........ .........78_........ False
26-(6+49)= ..........-29_...... ..........-29_...... True
37-29+26= .........34_........ .........34_........ True
42-(8+31)= ..........3_........ ..........3_........ True
(21-9)+35= ..........46_....... ..........47_....... False
(8+9)-11= .........6_......... .........6_......... True
6-41-2= .......-35_......... .......-37_......... False
31-30+16= .........16_........ .........17_........ False
0.35
```

可以看到準確度由約 0.4 下降至約 0.35

不過訓練的時間也由 9 分鐘下降至 7 分 23 秒

我個人是覺得因為 Batch size 變大導致 backward 的次數太少，參數更新的次數不夠才導致 accuarcy 下降

注: 我是使用電腦上的 VScode 並且有安裝 CUDA

硬體條件:

```
AMD Ryzen 7 5800H with Radeon Graphics
NVIDIA GeForce RTX 3060 Laptop GPU GDDR6 @ 6GB (192 bits)
```

Model disscussion:

我所使用的模型主體為 RNN，我認為 RNN 適合做這個任務，因為 hidden state 的存在讓 RNN 適合處理 serial 的 data，當在處理 input 的算式時應該要等看到”eos”時才開始進行答案的生成，並且模型本身應該要有能力紀錄目前存下來的訊息(看過的 context)，而 hidden state 會在 model 吃 input 時不斷的更新，概念上來說就是記下看到了甚麼。

如果拔掉 hidden state，將我的模型改成 torch.nn.Linear 的 stacks

```python
class CharRNN(torch.nn.Module):
    def __init__(self, vocab_size, embed_dim, hidden_dim):
        super(CharRNN, self).__init__()

        # Embedding層
        self.embedding = torch.nn.Embedding(num_embeddings=vocab_size,
                                            embedding_dim=embed_dim,
                                            padding_idx=tokenizer.stoi['.'])

        # RNN層
        self.rnn_layer1 = torch.nn.Linear(in_features = hidden_dim , out_features = hidden_dim)

        self.rnn_layer2 = torch.nn.Linear(in_features = hidden_dim , out_features = hidden_dim)
        # output層
        self.linear = torch.nn.Sequential(torch.nn.Linear(in_features=hidden_dim,
                                                          out_features=hidden_dim),
                                          torch.nn.ReLU(),
                                          torch.nn.Linear(in_features=hidden_dim,
                                                          out_features=vocab_size))
    def forward(self, batch_x):
        batch_x = self.encoder(batch_x)
        batch_x = self.linear(batch_x)
        return batch_x

    def encoder(self, batch_x):
        batch_x = self.embedding(batch_x)
        batch_x = self.rnn_layer1(batch_x)
        batch_x = self.rnn_layer2(batch_x)
        return batch_x
```

訓練的 loss 下降情況:

```
Training epoch 1: 100%|██████████| 6282/6282 [01:13<00:00, 85.12it/s, loss=0.567]
Testing epoch 1: 100%|██████████| 1571/1571 [00:19<00:00, 80.38it/s, loss=0.576]
Training epoch 2: 100%|██████████| 6282/6282 [01:13<00:00, 86.02it/s, loss=0.557]
Testing epoch 2: 100%|██████████| 1571/1571 [00:20<00:00, 78.49it/s, loss=0.569]
Training epoch 3: 100%|██████████| 6282/6282 [01:13<00:00, 85.82it/s, loss=0.554]
Testing epoch 3: 100%|██████████| 1571/1571 [00:20<00:00, 77.38it/s, loss=0.573]
Training epoch 4: 100%|██████████| 6282/6282 [01:12<00:00, 86.07it/s, loss=0.563]
Testing epoch 4: 100%|██████████| 1571/1571 [00:20<00:00, 77.15it/s, loss=0.563]
Training epoch 5: 100%|██████████| 6282/6282 [01:12<00:00, 86.06it/s, loss=0.567]
Testing epoch 5: 100%|██████████| 1571/1571 [00:21<00:00, 74.24it/s, loss=0.573]
```

Output:

```
38+39-7= ........-.......... ........70_........ False
19-41+49= ........-......... .........27_........ False
44+30-26= ........-......... .........48_........ False
48-30-15= ........-......... .........3_......... False
0-(5+12)= ........-......... .........-17_....... False
22+(22-22)= .........-........ ...........22_...... False
(44-16)+42= .........-........ ...........70_...... False
(46+48)-9= .........-........ ..........85_....... False
1+29+13= ........-.......... ........43_........ False
39-(20+16)= .........-........ ...........3_....... False
(49-8)+20= .........-........ ..........61_....... False
(3-42)+10= .........-........ ..........-29_...... False
27+11+40= ........-......... .........78_........ False
26-(6+49)= .........-........ ..........-29_...... False
37-29+26= ........-......... .........34_........ False
42-(8+31)= .........-........ ..........3_........ False
(21-9)+35= .........-........ ..........47_....... False
(8+9)-11= ........-......... .........6_......... False
6-41-2= .......-........... .......-37_......... False
31-30+16= ........-......... .........17_........ False
0.0
```

因為沒有 hidden state，結構較為簡單，所以並且最常看到 padding，所以模型又放棄了，開始全部印 padding，還有一點，在應該要開始 generate 答案的位置都有一個”-”，這可能是因為有一定比例的 tgt 是負數，而這一點被這個 linear layer 的 stack 給捕捉到了

Tgt 為負數的比例:

```
data = pd.read_csv('arithmetic.csv')
```
✓ 1.1s

```
data['tgt'] =  data['tgt'].apply(str)
neg = data[data['tgt'].apply(lambda x : '-' in x)]
len(neg) / len(data)
```
✓ 0.6s

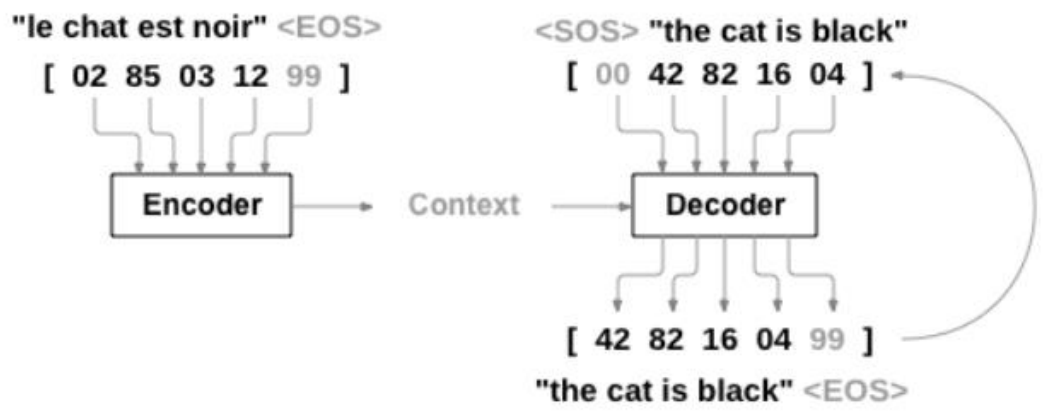
```
0.2585375118708452
```

Bonus:

接下來我嘗試將模型改成 encoder ， decoder 架構:

原版保有在 Main.ipynb 中

Encoder-decoder architecture 保有在 Encoder_decoder_architecture.ipynb



不過 tokenizer 還是會用跟原本的架構，概念上來說不像上圖是 sentence

To sentence ， 2a 其實更加接近 volcabury to volcabury

這邊的 decoder 我計畫使用一個 GRU，傳遞 context 的部分由 encoder 最後一層的 RNN 的 hidden state 提供。

.

會希望使用這個架構的原因是因為，我個人覺得將 arithmetic expression 轉成

對應的答案其實很像"翻譯"，也就是上面提到的 volcabury to volcabury

.

.

.

.

.

.

架構:

```
class CharRNN(torch.nn.Module):
    def __init__(self, vocab_size, embed_dim, hidden_dim):
        super(CharRNN, self).__init__()

        # Embedding層
        self.embedding = torch.nn.Embedding(num_embeddings=vocab_size,
                                            embedding_dim=embed_dim,
                                            padding_idx=tokenizer.stoi['.'])

        # RNN層
        self.rnn_layer1 = torch.nn.RNN(input_size=embed_dim,
                                       hidden_size=hidden_dim,
                                       batch_first=True)
        self.rnn_layer2 = torch.nn.RNN(input_size=hidden_dim,
                                       hidden_size=hidden_dim,
                                       batch_first=True)
        # output層
        self.linear = torch.nn.Sequential(torch.nn.Linear(in_features=hidden_dim,
                                                          out_features=hidden_dim),
                                          torch.nn.ReLU(),
                                          torch.nn.Linear(in_features=hidden_dim,
                                                          out_features=vocab_size))
        self.embedding2 = torch.nn.Embedding(num_embeddings=vocab_size,embedding_dim=embed_dim , padding_idx=tokenizer.stoi['.'])
        self.gru = torch.nn.GRU(input_size=embed_dim,hidden_size=embed_dim,batch_first=True)

    def forward(self, batch_x , target):
        hidden = self.encoder(batch_x)
        output , _ = self.decoder(prev_hidden=hidden , target=target)
        output = self.linear(output)
        return output

    def encoder(self, batch_x):
        batch_x = self.embedding(batch_x)
        batch_x , ht = self.rnn_layer1(batch_x)
        batch_x , ht = self.rnn_layer2(batch_x)
        return ht
    def decoder(self , prev_hidden , target):
        decoder_hidden = prev_hidden
        decoder_outputs ,_ = self.forward_step(target , decoder_hidden)
        #decoder_outputs = F.log_softmax(decoder_outputs, dim=-1)
        return decoder_outputs, decoder_hidden

    def forward_step(self,input , hidden):
        output = self.embedding2(input)
        output , hidden = self.gru(output , hidden)
        return output , hidden
```

實作上將 encoder 吃完 input 後產生的 hidden state 給 decoder 用，之後給 decoder 加上 sos 的 input src (跟 encoder 的 input 差在開頭多了 sos) 再進行 forward

在這裡我將”:” 定義成 sos 。 en_src 就是給 decoder 的 input，dataset 僅包含+-()

	src	tgt	en_src
0	0+0=_..............	....0_.............	:0+0=..............
1	0-0=_..............	....0_.............	:0-0=..............
15	0+0+0=_............	......0_...........	:0+0+0=............
16	0-0-0=_............	......0_...........	:0-0-0=............
18	0+0-0=_............	......0_...........	:0+0-0=............
...	...	...	...
2632495	(49+49)-49=_.......	...........49_.....	:(49+49)-49=.......
2632496	49+(49-49)=_.......	...........49_.....	:49+(49-49)=.......
2632497	49-49+49=_.........	.........49_.......	:49-49+49=.........
2632498	(49-49)+49=_.......	...........49_.....	:(49-49)+49=.......
2632499	49-(49+49)=_.......	...........-49_....	:49-(49+49)=.......

加上 decoder 訓練過程的 loss 變化:

```
Training epoch 1: 100%|██████████| 6282/6282 [02:10<00:00, 48.06it/s, loss=0.107]
Testing epoch 1: 100%|██████████| 1571/1571 [00:31<00:00, 49.24it/s, loss=0.113]
Training epoch 2: 100%|██████████| 6282/6282 [02:10<00:00, 48.04it/s, loss=0.0815]
Testing epoch 2: 100%|██████████| 1571/1571 [00:32<00:00, 48.69it/s, loss=0.0747]
Training epoch 3: 100%|██████████| 6282/6282 [02:10<00:00, 48.07it/s, loss=0.0604]
Testing epoch 3: 100%|██████████| 1571/1571 [00:32<00:00, 48.77it/s, loss=0.0495]
Training epoch 4: 100%|██████████| 6282/6282 [02:10<00:00, 48.06it/s, loss=0.0632]
Testing epoch 4: 100%|██████████| 1571/1571 [00:31<00:00, 49.33it/s, loss=0.0449]
Training epoch 5: 100%|██████████| 6282/6282 [02:10<00:00, 48.26it/s, loss=0.035]
Testing epoch 5: 100%|██████████| 1571/1571 [00:32<00:00, 48.85it/s, loss=0.0326]
```

原版:

```
Training epoch 1: 100%|██████████| 6282/6282 [01:26<00:00, 72.98it/s, loss=0.127]
Testing epoch 1: 100%|██████████| 1571/1571 [00:21<00:00, 72.25it/s, loss=0.139]
Training epoch 2: 100%|██████████| 6282/6282 [01:25<00:00, 73.21it/s, loss=0.104]
Testing epoch 2: 100%|██████████| 1571/1571 [00:21<00:00, 72.22it/s, loss=0.103]
Training epoch 3: 100%|██████████| 6282/6282 [01:25<00:00, 73.51it/s, loss=0.0939]
Testing epoch 3: 100%|██████████| 1571/1571 [00:21<00:00, 72.05it/s, loss=0.0936]
Training epoch 4: 100%|██████████| 6282/6282 [01:27<00:00, 71.99it/s, loss=0.0838]
Testing epoch 4: 100%|██████████| 1571/1571 [00:21<00:00, 72.49it/s, loss=0.0743]
Training epoch 5: 100%|██████████| 6282/6282 [01:25<00:00, 73.49it/s, loss=0.0765]
Testing epoch 5: 100%|██████████| 1571/1571 [00:22<00:00, 70.94it/s, loss=0.0798]
```

可以看得出來，加上 decoder 後模型的 loss 下降的更多

超參數、tokenizer、dataset 兩個版本皆一致

```
batch_size = 128
epochs = 5
embed_dim = 256
hidden_dim = 256
lr = 0.0001
grad_clip = 1
input_dim = tokenizer.token_len()
```

原版精準的確度為:0.408

```
30-18+19= .........31_........ .........31_........ True
41+(38-48)= ...........29_...... ...........31_...... False
21+18+49= .........88_........ .........88_........ True
12+31= ......42_........... ......43_........... False
3-3-23= .......-23_......... .......-23_......... True
25-(3+25)= ..........-3_....... ..........-3_....... True
0.408
```

加上 decoder 後精準度為 0.677

```
30-18+19= .........31_........ .........31_........ True
41+(38-48)= ...........30_...... ...........31_...... False
21+18+49= .........89_........ .........88_........ False
12+31= ......44_........... ......43_........... False
3-3-23= .......-23_......... .......-23_......... True
25-(3+25)= ..........-3_....... ..........-3_....... True
0.677
```

會有這樣的差異，除了加上 decoder 後模型擁有更多的參數外，再來就是 encoder 的 hidden state 記錄下的資訊對於 decoder 來說的確有用。最後，如同 padding eos 在 encoder 中有發揮作用，我覺得 decoder 的 sos 也有發揮作用，加入 eos、sos 某種程度上會對 hidden state 產生特殊影響，讓 hidden 類似 state machine，幫助 decoder 找到解讀算式/生成結果的時機。

這個 encoder-decoder 架構主要是來源於這篇文章:

https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

在文章後半給 decoder 加上了 attention 機制並且取得了更好的表現，也就是說，理論上這個架構還能更強

2024/4/21 更新

根據助教的要求

“另外一個普遍的問題是很多同學直接把 LSTM 的 output 和 label 算 loss，這樣是不對的。請注意，一定要把等號到 eos 的部分先截取出來，因爲那部分才是你希望模型學會的東西。”

我將程式碼做了一些修改，簡單來說我在 dataset 的 dataframe 中新增了 tgt 的 eos 的 index

```python
def rshift(row):
    return ''.join(['.' for _ in range(len(row['src']))]) + row['tgt']
data['tgt'] = data.apply(rshift, axis=1)
data['tgt_eos'] = data['tgt'].apply(len)
data['tgt'] = data['tgt'].apply(lambda x : x + '_')
data['src'] = data['src'].apply(lambda x : x + '_')
data
```

同時 dataset 的 get_item、collate_fn 也有修改

```python
    def __getitem__(self, index):
        # Get input and output from the dataframe
        input_data = self.data.iloc[index,0]
        output_data = self.data.iloc[index,1]
        eos_idx = self.data.iloc[index , 2]
        # Convert input and output to PyTorch tensors
        input_tensor = torch.tensor(input_data)
        output_tensor = torch.tensor(output_data)

        return input_tensor, output_tensor , eos_idx
    def __len__(self):
        return len(self.data)

def collate_fn(batch):
    batch_x = [torch.tensor(data[0]) for data in batch] # list[torch.tensor]
    batch_y = [torch.tensor(data[1]) for data in batch] # list[torch.tensor]
    batch_eos = [data[2] for data in batch]
    batch_x_lens = torch.LongTensor([len(x) for x in batch_x])
    batch_y_lens = torch.LongTensor([len(y) for y in batch_y])

    # torch.tensor
    # [[1968, 1891, 3580, ... , 0, 0, 0],
    #  [1014, 2242, 2247, ... , 0, 0, 0],
    #  [3032,  522, 1485, ... , 0, 0, 0]]
    #                   padding↑
    pad_batch_x = torch.nn.utils.rnn.pad_sequence(batch_x,
                                                  batch_first=True, # shape=(batch_size, seq_len)
                                                  padding_value=tokenizer.stoi['.'])
    pad_batch_y = torch.nn.utils.rnn.pad_sequence(batch_y,
                                                  batch_first=True, # shape=(batch_size, seq_len)
                                                  padding_value=tokenizer.stoi['.'])

    return pad_batch_x, pad_batch_y,batch_x_lens, batch_y_lens , batch_eos
```

最後是將 output 出來的結果根據 eos 的 index 做裁切

```
for idx in batch_eos:
    #print(idx)
    batch_y = batch_y[: , :idx]
    batch_pred_y = batch_pred_y[: , :idx , :]
batch_y = batch_y.to(device)
```

Loss function 跟之前一致 padding 的 loss 會被計算

```
criterion = torch.nn.CrossEntropyLoss( reduction='mean')
```

Loss:

```
Training epoch 1: 100%|██████████| 6282/6282 [03:25<00:00, 30.52it/s, loss=0.00505]
Testing epoch 1: 100%|██████████| 1571/1571 [00:33<00:00, 46.43it/s, loss=0.00617]
Training epoch 2: 100%|██████████| 6282/6282 [04:21<00:00, 24.02it/s, loss=0.00409]
Testing epoch 2: 100%|██████████| 1571/1571 [00:30<00:00, 50.92it/s, loss=0.0205]
Training epoch 3: 100%|██████████| 6282/6282 [04:27<00:00, 23.48it/s, loss=0.00402]
Testing epoch 3: 100%|██████████| 1571/1571 [00:30<00:00, 51.09it/s, loss=0.00159]
Training epoch 4: 100%|██████████| 6282/6282 [04:18<00:00, 24.27it/s, loss=0.0201]
Testing epoch 4: 100%|██████████| 1571/1571 [00:35<00:00, 44.64it/s, loss=0.00263]
Training epoch 5: 100%|██████████| 6282/6282 [04:24<00:00, 23.78it/s, loss=0.0255]
Testing epoch 5: 100%|██████████| 1571/1571 [00:31<00:00, 49.75it/s, loss=0.00261]
```

我不確定這是不是助教想傳達的”正確做法”，但是實驗的結果很糟

```
37+35-3= ........78 ........69 False
4-31-46= ........-30 ........-73 False
26+46+16= .........58 .........88 False
12-(32+8)= ..........-40 ..........-28 False
36+(8-33)= ..........78 ..........11 False
11-26-21= .........-30 .........-36 False
49-26+34= .........56 .........57 False
34-(40+30)= ...........189 ...........-36 False
46-16-40= .........-10 .........-10 True
30-18+19= .........36 .........31 False
41+(38-48)= ...........28 ...........31 False
21+18+49= .........96 .........88 False
12+31= ......48 ......43 False
3-3-23= .......-20 .......-23 False
25-(3+25)= ..........-1 ..........-3 False
0.046
```

正確度不到 5%

另外這是 padding 不計算 loss 的結果、也就是 loss function 跟 sample code 一致

```
criterion = torch.nn.CrossEntropyLoss(ignore_index=tokenizer.stoi['.'], reduction='mean')
```

Loss:

```
Training epoch 1: 100%|██████████| 6282/6282 [03:07<00:00, 33.44it/s, loss=0.62]
Testing epoch 1: 100%|██████████| 1571/1571 [00:26<00:00, 58.49it/s, loss=1.22]
Training epoch 2: 100%|██████████| 6282/6282 [03:08<00:00, 33.36it/s, loss=0.612]
Testing epoch 2: 100%|██████████| 1571/1571 [00:26<00:00, 59.54it/s, loss=0.753]
Training epoch 3: 100%|██████████| 6282/6282 [03:08<00:00, 33.37it/s, loss=0.755]
Testing epoch 3: 100%|██████████| 1571/1571 [00:26<00:00, 59.17it/s, loss=1.37]
Training epoch 4: 100%|██████████| 6282/6282 [03:07<00:00, 33.55it/s, loss=1.15]
Testing epoch 4: 100%|██████████| 1571/1571 [00:26<00:00, 58.84it/s, loss=0.709]
Training epoch 5: 100%|██████████| 6282/6282 [03:20<00:00, 31.28it/s, loss=1.15]
Testing epoch 5: 100%|██████████| 1571/1571 [00:31<00:00, 50.50it/s, loss=1.54]
```

結果

```
30-18+19= --327213125 .........31 False
41+(38-48)= -334858511-10 ...........31 False
21+18+49= 5-115138485 .........88 False
12+31= --115145 ......43 False
3-3-23= --3-3---21 .......-23 False
25-(3+25)= 531151-1---1 ..........-3 False
0.0
```

正確度是 0

剩下的 learning rate 、 batch size 的實驗就不重做了，我想結果也不會差太多

根據助教要求修改的模型保有在 model_4/21_update.ipynb 當中。