2022 Deep Learning HW2

310706034 資管碩一 吳啓玄

1.1 Data Preprocessing

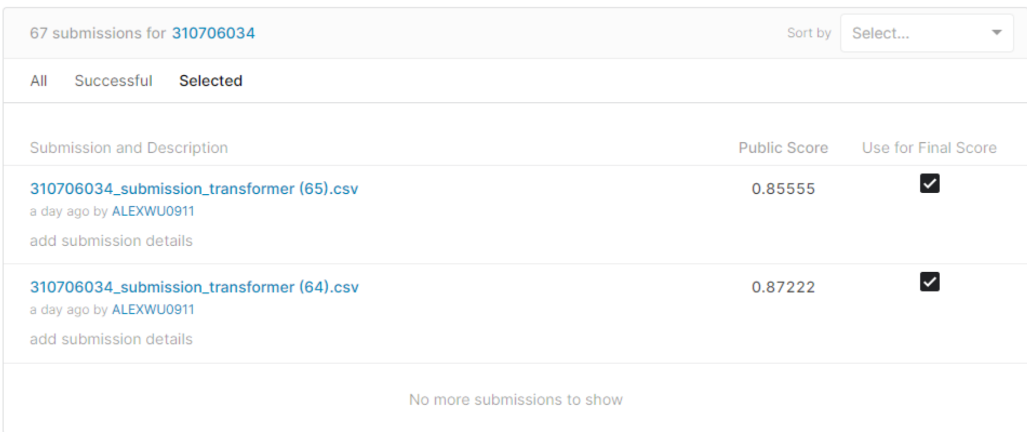
1. Tokenizer 使用 NLTK (Natural Language ToolKit) 套件，如果用 white space 去 tokenize the text，那麼 ” SpaceX’ s Falcon 1 ” 這個句子會被拆成 [SpaceX’ s, Falcon, 1]，但使用 NLTK 則會拆成 [SpaceX, ’ , s, Falcon, 1]，這樣的斷詞好處是英文縮寫以及帶有句點的地方可以斷開，例如 didn't、 good.，這樣更能增加文字本身的詞義。

2. 當使用的模型只接受固定長度句子時，需要 <pad> 充當字詞，將不足長度的句子補齊，<unk> 是沒有對應到字典的單字、或是詞頻太低的單字，會用 <unk> 取代。

3. Tokenizer 使用 NLTK，並將 Title 和 Description 用句點加在中間進行合併，另外還有做以下文字處理:
   - 刪除 English stop words
   - 刪除標點符號 (punctuation)
   - 刪除 ”lt”、”gt” ，因為這兩個字代表 <>，實際上是沒意義的
   - 刪除 ”reuters” 路透社為新聞社名字，與新聞類別沒有關係
   - 使用 min_count = 1，因為每個字詞都帶有資訊量
   - Word embedding 使用 GloVe.6B.300d

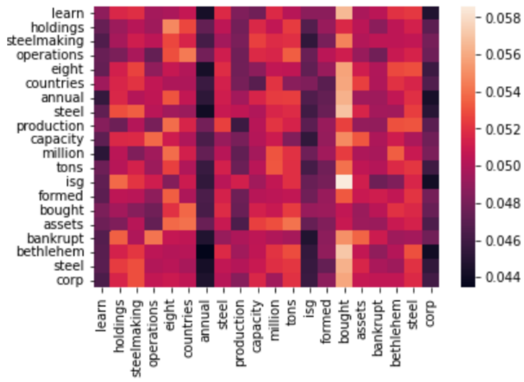
1.2 Transformer

1.



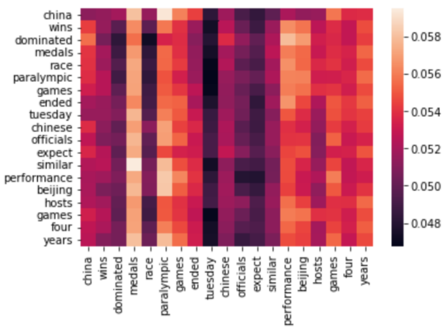
2. attention map

```
test sentence_id =  2
['learn', 'holdings', 'steelmaking', 'operations', 'eight', 'countries', 'annual', 'steel', 'production', 'capacity', 'million', 'tons', 'isg', 'formed', 'bought', 'assets', 'bankrupt', 'bethlehem', 'steel', 'corp']
predict_label = Business
sentence length =  20
```



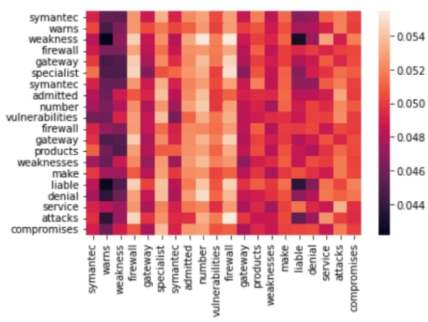
isg 和 bought 的 attention 是上表 attention map 的最大值，因此具有句子的代表性，並分類為 business。

```
test sentence_id =  18
['china', 'wins', 'dominated', 'medals', 'race', 'paralympic', 'games', 'ended', 'tuesday', 'chinese', 'officials', 'expect', 'similar', 'performance', 'beijing', 'hosts', 'games', 'four', 'years']
predict_label = Sports
sentence length =  19
```



Medals 和 其他字詞的 attention 都相對其他字詞大，因此也具有體育類句子的代表性，並分類為 sports。

```
test sentence_id =  27
['symantec', 'warns', 'weakness', 'firewall', 'gateway', 'specialist', 'symantec', 'admitted', 'number', 'vulnerabilities', 'firewall', 'gateway', 'products', 'weaknesses', 'make', 'liable', 'denial', 'service', 'attacks', 'compromises']
predict_label = Sci/Tech
sentence length =  20
```



Firewall 和 其他字詞的 attention 都相對其他字詞大，因此具有科技類句子的代表性，並分類為 Sci/Tech。

3. 模型架構如下:

```
class Transformer(nn.Module):
    def __init__(self, vocab_size, embedding_dim, max_len, d_model, num_layers, head, num_class, embedding_weight, dropout=0.2):
        super(Transformer, self).__init__()

        self.embedding_dim = embedding_dim
        self.max_len = max_len
        self.d_model = d_model
        self.head = head
        # self.dim_feedforward = num_layers
        self.dropout = dropout

        self.embedding_layer = nn.Embedding(vocab_size, embedding_dim)
        self.embedding_layer.weight.data.copy_(embedding_weight)
        self.embedding_layer.weight.requires_grad = False

        self.position_encoder = PositionalEncoding(max_len, embedding_dim)

        self.encoder_layer = nn.TransformerEncoderLayer(embedding_dim, head, num_layers, dropout=0.2, activation='gelu')
        self.encoder = nn.TransformerEncoder(self.encoder_layer, num_layers, norm=None)

        self.fc = nn.Linear(embedding_dim, d_model)
        self.tanh = nn.Tanh()
        self.fc2 = nn.Linear(d_model, num_class)


    def generate_mask(self, batch_length):
        self.mask = torch.zeros(len(batch_length), self.max_len).to(device)
        for i, length in enumerate(batch_length):
            self.mask[i][:length] = 1
        masked = self.mask.float().masked_fill(self.mask == 0, float('-inf')).masked_fill(self.mask == 1, float(0.0))
        return masked


    def forward(self, x, masked, return_attention=True):
        embedding = self.embedding_layer(x) * math.sqrt(self.embedding_dim)
        out = self.position_encoder(embedding)
        out = out.permute(1, 0, 2)
        out= self.encoder(out, src_key_padding_mask=masked)
        out = out.permute(1, 0, 2)
        self.mask = self.mask.unsqueeze(2)
        out = out.sum(1) / (self.mask.sum(1) + 1e-5)
        out = self.tanh(self.fc(out))
        out = self.fc2(out)

        return out
```

模型參數如下:

使用 RAdam+ASAM 作為 Optimizer，因為 ASAM 有泛化性佳的優點

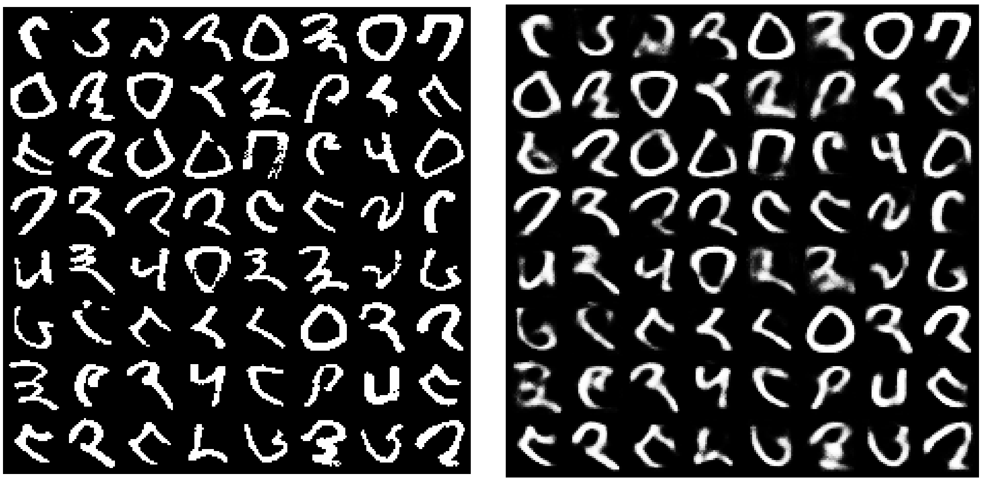
```
fix_length = 200
LR = 0.0008
epoch = 20
embedding_size = 300     ### must be divisible by num_heads
batch_size = 200

num_layers = 5
head = 15
d_model = 512
num_class = 4
dropout = 0.025
```

4. Private score

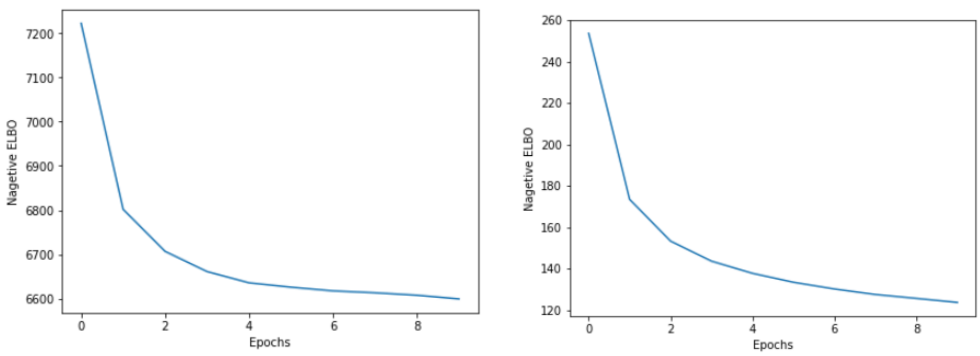
2. VAE

1. show the learning curve and some reconstructed samples

left: Real samples in dataset, right: Reconstructed samples using VAE



left: Real samples in dataset, right: Reconstructed samples using VAE

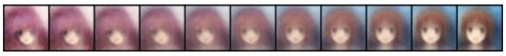


Left : learning curve of animation faces, Right : learning curve of TibetanMNIST

2. Sample the prior p(z) and use the latent codes z to synthesize some examples.

Left : sampled images of animation faces, Right : sampled images of TibetanMNIST

3. Show the synthesized images based on the interpolation of two latent codes z between two real samples.

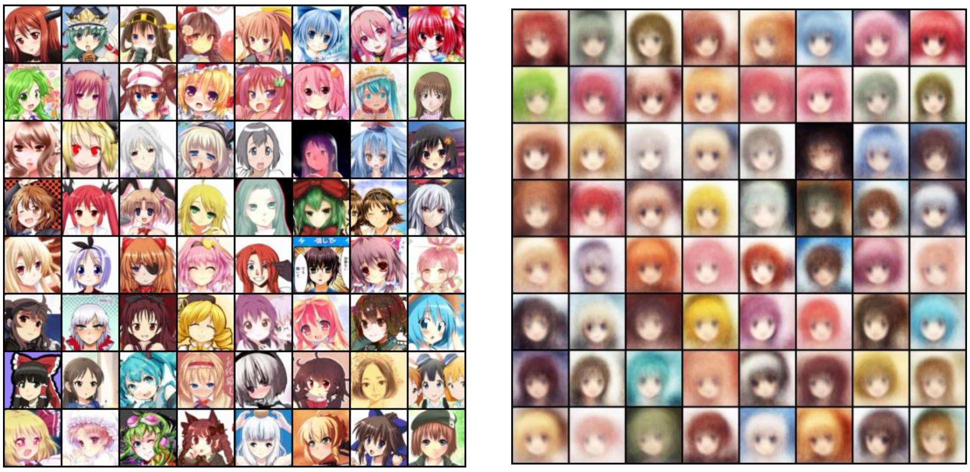


left: synthesized images of animation faces, right: synthesized images of TibetanMNIST

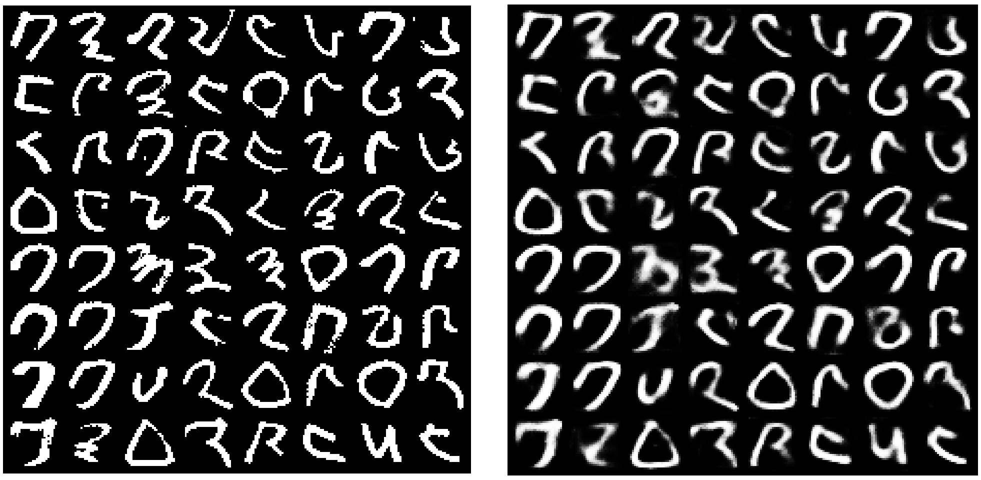
4.

Step 1.

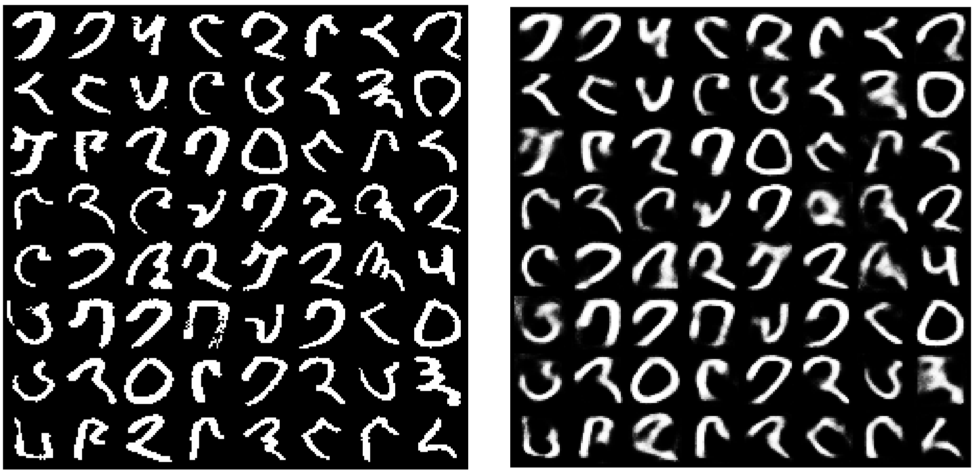
left: Real samples in dataset, right: Reconstructed samples using VAE (scale=25)



left: Real samples in dataset, right: Reconstructed samples using VAE (scale=75)



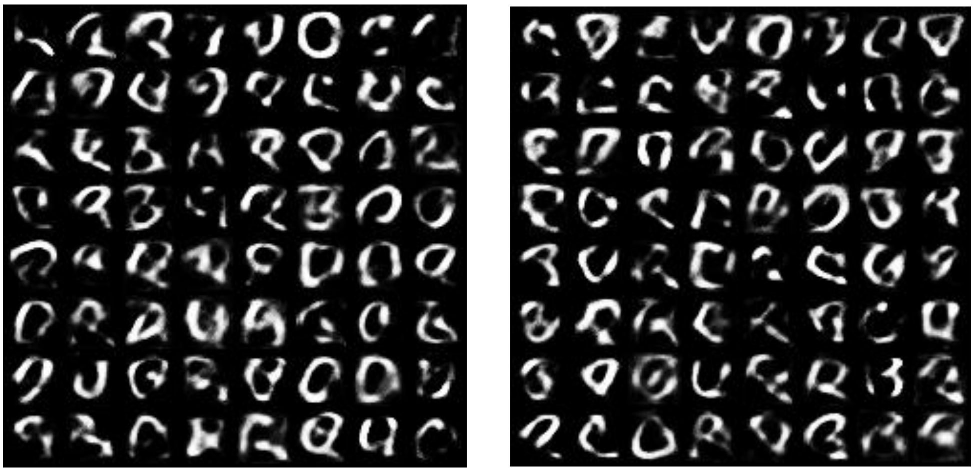
left: Real samples in dataset, right: Reconstructed samples using VAE (scale=25)



left: Real samples in dataset, right: Reconstructed samples using VAE (scale=75)

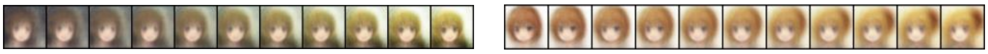
Step 2.

Left : sampled images of animation faces (scale=25), Right : sampled images of animation faces (scale=75)



Left : sampled images of TibetanMNIST (scale=25), Right : sampled images of TibetanMNIST (scale=75)

Step 3.



left: synthesized images of animation faces (scale=25), right: synthesized images of animation faces (scale=75)

left: synthesized images of TibetanMNIST (scale=25), right: synthesized images of TibetanMNIST (scale=75)

當 scale 的數值越小時，圖片的重建會越清晰；反之，若 scale 數值越大，圖片的重建會越模糊，因 KLD 代表的是散度，當越接近 100% 時會越分散。