# Deep Learning (Homework 2)

Due date : 2021/05/07 23:55:00

- High-level API are accepted in this homework, such as Keras, slim, TFLearn, Hugginface, Allennlp etc. But you can earn **40% bonus** if you do not use any high-level API in this homework.
- If you have any problem about implementation, DO NOT directly upload your code on the E3 discussion

## 1 RNN and Transformer for News Classification

You are given a csv file (train.csv) that contains the corresponding label for BBC news with title and content. You are required to implement a recurrent neural network (RNN, LSTM or GRU) and transformer to correctly classify the news in (test.csv) and submit the result to the Kaggle competition. To start this problem, some preliminary steps need to be conducted first:

1. **Join the in-class competition for RNN on Kaggle (Here)**
2. **Join the in-class competition for transformer on Kaagle (Here)**
3. **Change the team name by using your student id number**
4. **Download the data and check for the description**

### 1.1 Text Preprocessing (10%)

1. How do you choose the tokenizer for this task? Could we use the white space to tokenize the text? What about we use the complicated tokenizer instead? Make some discussion. (5%) (You might want to explain it by showing the performance comparison with different tokenizer. If you are not familiar with tokenizer, check Here.
2. Why we need to use special token like ⟨pad⟩ and ⟨unk⟩? (2%)
3. Briefly explain how your procedure handles the text data. (3%) (tokenize, stop word, min_count setting, etc)

### 1.2 RNN (30%)

Build the recurrent neural network to solve this task and answer the following question

1. Outperform the RNN baseline on the Kaggle in-class competition. (20%)
2. How do you choose the initial embedding for this task? Why we often use the pretrained embedding instead of random initialization? (5%)
3. Discuss the model structure (e.g. RNN, GRU or LSTM) you design. (5%)

### 1.3 Transformer (30%)

Build the transformer to solve this task and answer the following question

1. Outperform the transformer baseline on the Kaggle in-class competition. (20%)
2. Please show the attention map in the last layer of transformer of some examples to find out which token contributes higher attention in the classification result. Does your finding make sense? Please make some discussion. (5%)
3. Discuss the model setting (e.g. the hyperparameters in transformer or the input formation) you design. (5%)

### 1.4 Hints

You might want to follow these steps to start your work:

1. Tokenize the given text data with some off-the-shelf software like NLTK or Spacy.
2. Build the Vocab (like the dictionary object in python) to map the token with some unique ID.
3. Setup the collate function of data loader with the hands of tokenizer and Vocab. (If you are a Pytorch user.)
4. Select the pretrained embedding (Glove, Fasttext, W2V) as the initialization of your embedding layer. (Not necessary, but recommended)
5. Construct your text classification model. You might first start with some sequence to sequence module then sequence to vector module and finally end up with some simple feedforward module.
6. Choose the suitable optimizer (Adam might be not suitable) and activation function (ReLU might be not suitable. Try Tanh or Swish?)
7. Try some tricks like learning rate scheduler and packed_padding_sequence.
8. Check some tutorial such as Here.

## 2 Variational Autoencoder for Image Generation

It is important to study the paper on variational autoencoder (VAE) before the implementation. You need to implement VAE with two kinds of decoder where different variational posteriors are calculated. The decoder should be selected according to data type. There are two datasets. The first one is the TibetanMNIST (gray images, TibetanMNIST.npz) and the other one is the animation faces (RGB images, anime-faces.zip), some examples are shown below.

Figure 1: left: TibetanMNIST digits, right: animation faces

This task is to build VAE to reconstruct images. The following steps should be implemented individually for these two datasets. Some examples are shown in each step.

1. Implement VAE and show the learning curve and some reconstructed samples like the given examples. (10%) Hint:
   - Convert the gray images into binary images first
   - Variational posteriors are Bernoulli for binary data and Gaussian for real value data

Figure 2: left: Real samples in dataset, right: Reconstructed samples using VAE

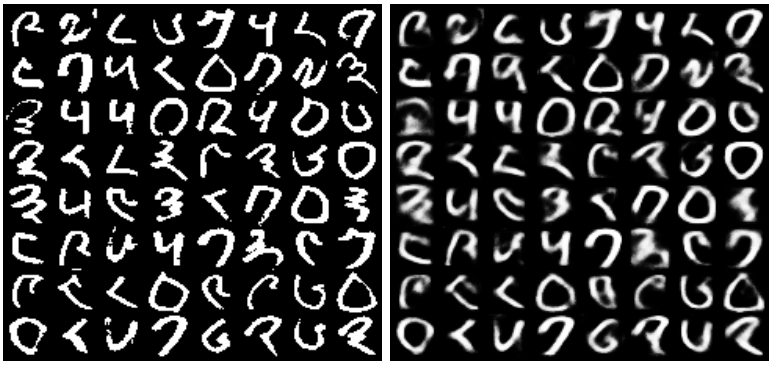

Figure 3: left: Real samples in dataset, right: Reconstructed samples using VAE

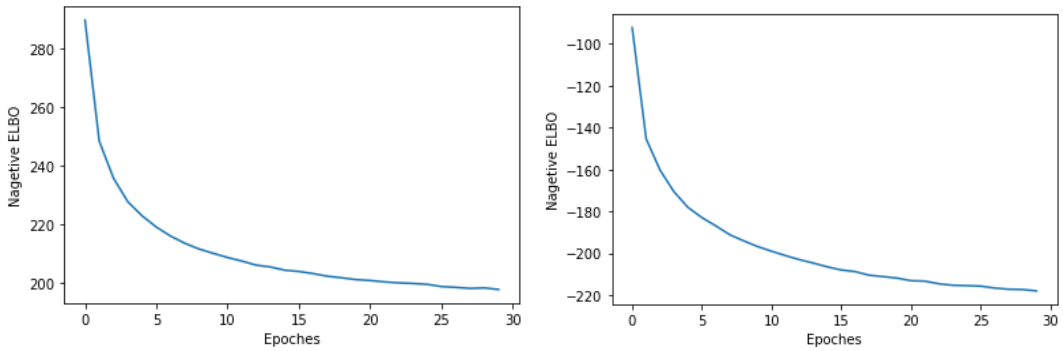


Figure 4: left: Learning curve of TibetanMNIST digits, right: Learning curve of animation faces

2. Sample the prior $p(\mathbf{z})$ and use the latent codes $\mathbf{z}$ to synthesize some examples when your model is well-trained. (5%)

Figure 5: left: synthesized images of TibetanMNIST, right: synthesized images of animation faces

3. Show the synthesized images based on the interpolation of two latent codes $\mathbf{z}$ between two real samples. (5%)

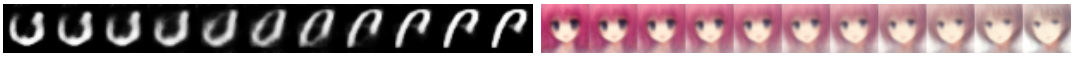

Figure 6: left: synthesized images of TibetanMNIST digits, right: synthesized images of animation faces

4. Multiply the Kullback-Leiblier (KL) term with a scale $\lambda$ and tune $\lambda$ (e.g. $\lambda = 0$ and $\lambda = 100$) then show the results based on steps 1, 2, 3 and some analyses. (10%)

## 3 Rules

- Please name the assignment as hw2_StudentID.zip (e.g. hw1_0123456.zip).
- You need to upload the zip file which has the same file structure with hw2_0850000.zip (Example file) in E3 announcement.
- Make sure the provided codes (RNN and transformer) can generate the submission file you previously uploaded to Kaggle.
- You will only earn the bonus 40% if you only use the package listing in bonus_package.txt to finish your work. If you think the package you use can be included in bonus_package.txt, please let TAs know.
- Only Python implementation is acceptable.
- Pytorch library is recommended for the implementation.
- DO NOT PLAGIARISM. (We will check program similarity score.)