Hong Kong University of Science and Technology
COMP 4211: Machine Learning
Spring 2020

Programming Assignment 2
Due: 21 April 2020, Tuesday, 11:59pm

1 Objective

The objective of this assignment is to practise the use of the `PyTorch` machine learning framework through building two deep learning models to solve two interesting and realistic tasks around the common theme of movies. Throughout the assignment, students will be guided to develop the models step by step and study the effects of different model settings.

2 Major Tasks

The assignment consists of two parts and a written report:

Part 1: Build an end-to-end multi-label convolutional neural network (CNN) model to predict the genres of movies from their posters.

Part 2: Build an end-to-end recurrent neural network (RNN) model to predict the sentiments of movie reviews.

Written Report: Report the results and answer some questions.

The two parts will be elaborated in Sections 4 and 5, respectively. Note that [Qn] refers to a specific question (the nth question) that you need to answer in the written report.

3 Setup

- Make sure that the libraries `torch`, `torchvision`, `torchsummary`, `torchtext`, `matplotlib` and `sklearn` have been installed on your machine.
- For this assignment, it suffices to use only `PyTorch` with `Python 3.7` or above. Other machine learning frameworks including `TensorFlow` and `Keras` are not accepted.
- You may read the CSSystem GPU user guide and use the GPU servers provided by the Department of Computer Science and Engineering or the Colaboratory, or called Colab, provided by Google (`https://colab.research.google.com/`).
- The datasets for this assignment are available as a ZIP file (`./pa2_data.zip`). After unzipping the file, you will find the datasets for two parts in `./pa2_data/part1_data` and `./pa2_data/part2_data`, respectively.

4 Part 1: Movie Genre Prediction

For movie lovers, a movie poster often gives the first impression about the movie content and its genre. In this task, your goal is to predict the genres of movies from their posters using a *multi-label* CNN model.

4.1 Multi-label Classification

From the lectures and tutorials, you have learned different multi-class classification models which predict exactly *one* label per sample. However, in movie genre prediction, one movie can belong to *multiple* genres. Problems of this type are referred to as multi-label classification problems.

You can modify a single-label model to handle multiple labels by changing the target representation and the prediction criterion:

- Represent the multi-label target as multiple binary labels
- During the testing phase, output all the classes that have probability higher than a predefined threshold θ.

Table 1: Representation of multi-label targets

id	Class A	Class B	Class C
x	1	0	0
y	0	1	1
z	1	1	1

Table 2: Prediction (**bold**) with $\theta = 0.5$

id	Class A	Class B	Class C
x	**0.71**	0.11	0.21
y	0.24	**0.62**	**0.93**
z	**0.82**	**0.77**	**0.61**

4.2 Dataset

The movie poster dataset can be found in the `/part1_data` directory. All csv files contain an **imdbId** column and seven genre columns: **Action**, **Adventure**, **Animation**, **Comedy**, **Drama**, **Horror**, and **Romance**. All the movie posters are located in the `/images` directory named by the movies' **imdbId**.

The files `train.csv` and `test.csv` contain labeled data, where a '1' in a genre column indicates the presence of that genre. On the other hand, the file `submission.csv` contains unlabelled data. You have to predict the movie genres in `submission.csv` and include the predictions as part of your assignment submission.

4.3 Tasks

This section will guide you to build the CNN model.

Table 3: Description of the data files for Part 1

File	# of instances	Has label?	Purpose
`train.csv`	10,365	yes	training
`test.csv`	2,000	yes	testing
`submission.csv`	591	no	prediction

4.3.1 Dataset and Dataloader

The first step is to build a custom `Dataset` class. A `Dataset` object loads the poster images in the `/images` directory using the **imdbId** from the csv files. You need to *apply at least one image transformation operation* to process the input images.

In this part, you are required to use *holdout validation* for hyperparameter tuning. You are suggested to split the data from `train.csv` into a training set and a validation set in a 80% : 20% ratio and create the `Dataloader` objects.

4.3.2 CNN Model Design

A CNN model takes an image as input and produces class probabilities (or scores) as output. Here, you need to build a CNN model from scratch with the following requirements:[1]

- The model should use *at least one but less than 10* `Conv2d` layers.
- The model should use *at least one* batchnorm and/or maxpool and/or dropout layer(s).
- The model should have *less than 1,000,000* trainable parameters.
- You should *not* use any pre-trained or built-in model.[2]

[Q1] "Print" the model architecture using `torchsummary.summary` and include it in the written report.

4.3.3 Training and Validation

Loss minimization: You can treat a multi-label model as multiple binary classifiers, each of which predicts the presence of one class. You can use the `Binary Cross Entropy Loss` or `Binary Cross Entropy Loss with Logits` provided by `PyTorch` as the loss function depending on your model output.[3] You are free to use any optimizer.[4]

Training: During training, you are required to plot the training and validation losses over time using `Tensorboard` or `matplotlib`.[5] You are advised to adjust your model architecture if the training loss does not drop after 10 epochs. By now, you should have a working model with a decreasing loss.

[Q2] Paste the screenshots of the plots and report the final training and validation losses obtained.

[1] You should follow these requirements throughout Part 1.
[2] You can look at some popular CNN architectures such as VGGNet and ResNet as reference.
[3] Optional to try other multi-label losses.
[4] Optional to try some optimizing techniques such as learning rate decay, special initialization, etc.
[5] Optional to monitor other metrics, such as F1 score, for a better understanding of the learning performance.

Hyperparameter Tuning: A lot of things can be modified to improve the performance of the CNN model, such as the model architecture, optimizer, batch size, learning rate, loss, or even data augmentation. Based on your observation in Q2, make at least one change to improve your model.[6] For example, you can change the learning rate, change the batch size, add more convolutional layers, or augment the data, etc.

[Q3] Discuss the change(s) and improvement and report the final hyperparameter setting. You may discuss the results in terms of the running time, training loss, validation loss, convergence rate, etc. You are expected to use graphs, tables or other visual means to assist your analysis.

4.3.4 Testing

You are required to compute the classification report from `sklearn.metrics` by training the CNN model with the whole training set and testing it against `test.csv` using the optimal hyperparameters found in the previous holdout validation. You may want to train the model with more epochs for a better result. Please be reminded that a multi-label model selects the classes with probability larger than a predefined threshold θ as the prediction. Repeat the evaluation by choosing different values of θ and observe how it affects the precision and recall.

[Q4] Paste the screenshots of the classification reports. Describe how varying θ affects the precision and recall. Explain the reason behind.

[Q5] Given that the cost of misclassification is higher than that of missing the true label(s), how should you set the threshold and why?

Note: Multi-label classification is more difficult than single-label classification because the input images and output labels are more complex. So, do not worry if you cannot achieve high precision, high recall, or high F1 score. It is more important that you demonstrate your understanding in building and tuning a CNN model from end to end. Beyond the assignment, you may borrow ideas from some popular CNN models like ResNet or VGGNet and build a deeper CNN model for better results.

4.3.5 Prediction

Finally, you have built your own movie genre classification model! The last step is to predict the movie genres in `submission.csv` and save the predictions as a csv file. Your predictions should be aimed to produce an optimal F1 score.

[P1] Submit your predictions as a csv file. The format should be the same as that of `train.csv`: one **imdbId** column and seven genre columns in the same order, where '1' in the data rows indicates the presence of a genre. The order of the **imdbId** column should be the same as that in `submission.csv`.

[6] You may apply grid search or random search. However, try to limit the search space to avoid having a long running time.

5 Part 2: Movie Review Sentiment Prediction

After watching a movie, some people write up their reviews on the web. These reviews can be roughly categorized into positive reviews and negative reviews. Since text is sequential in nature, it is common to use RNNs for textual data. In this part, you are required to build a binary RNN classifier to predict the sentiments of movie reviews.

5.1 Binary Text Sentiment Classification

From a tutorial, you have learned to use an RNN model to predict stock price. For text classification, the workflow is also similar. The RNN model takes a sentence (as a sequence of words) as input and models the sequential relationships between the words to predict a sentiment score.[7]

5.2 Dataset

The movie review dataset can be found in the directory `/part2_data`. For all csv files, each row represents one movie review. For `train.csv`, `validation.csv` and `test.csv`, there are one **label** column and one **text** column. The label column indicates the sentiments of the reviews **(0: Negative, 1: Positive)**; the text column contains the textual movie reviews.

For `submission.csv`, all the labels are set to **3**, meaning **Unknown**. You are required to predict and submit the predicted labels of the reviews in `submission.csv`.

Table 4: Description of the data files for Part 2

File	# of instances	Has label?	Purpose
`train.csv`	40,000	yes	training
`validation.csv`	5,000	yes	validation
`test.csv`	3,000	yes	testing
`submission.csv`	2,000	no	prediction

5.3 Tasks

This section will guide you to build the RNN model. Please create a new `Jupyter` notebook for this part.

5.3.1 Dataset and Dataloader

The first step is to build a custom `TabularDataset` class. A `TabularDataset` object loads the data from `train.csv`, `validation.csv` and `test.csv` with properly defined fields for label data and text data.

For this part, you do not need to split the training set further. You should train the model on the training set and evaluate it on the validation set provided. (Hint: the vocabulary of training, testing and predicting stages should only be built on the training data provided.)

[7] You may refer to the tutorial notes on some related concepts and techniques for dealing with textual data, such as tokenization, vocabulary, word embedding, etc.

5.3.2 RNN Baseline Model

An RNN model takes a sequence of words as input and produces class probabilities (or scores) as output. Here, you need to build a baseline RNN model with the following settings:

Table 5: Configuration of the baseline model

Parameter	Value	Remarks
# embedding layers	1	embedding dimension = 64
# RNN layers	1	RNN cell = `nn.RNN` hidden dimension = 128
# dropout layers	≥ 1	$p = 0.5$
# fully connected layers	1	
Batch size	32	
Optimizer	Adam	

[Q6] "Print" the model architecture and the number of trainable parameters for the model and include it in the written report. (Hint: since `torchsummary.summary` does not support multi-input RNN model directly, you may use other functions introduced in the tutorial notes.)

Loss Minimization: Since this is a binary classification task, you may use `Binary Cross Entropy Loss` or `Cross Entropy Loss`. (Hint: there are slight differences for the input of the two losses.) provided by `PyTorch`. For optimization, you are advised to start with the default setting of the Adam optimizer.

Training: During training, you are required to plot the training and validation losses in one plot using `matplotlib`.[8] The model should converge within 10 epochs. It is useful for you to save the training and validation loss profiles as well as the test accuracy for later parts.

Testing: You are required to compute the accuracy of the baseline model against the test set `test.csv`. You may combine the training and validation sets to train the model. We will name this model the *baseline model* for the rest of the assignment.

[Q7] Paste the screenshot of the plot and report the test accuracy.

5.3.3 Empirical Study

Different RNN Settings

In this section, you need to build four models in addition to the baseline model. These four models are variants of the baseline model and are used to compare the performance of different RNN settings. You should only change the settings indicated in Table 6 and keep the other parameters the same as the baseline model.

Model Description: Models 1 and 2 replace the baseline vanilla RNN cell with GRU and LSTM cells. Model 3 uses bidirectional LSTM. 'Bidirectional' here means that the sequential relationship of the sequences is modeled in both forward and backward directions. This can be set in the parameters of `nn.LSTM`. Model 4 examines the effect of an embedding layer. You need to remove the `nn.Embedding` layer from the baseline model and connect the input words directly to the LSTM cells.

[8] Optional to monitor other metrics, such as F1 score, for a better understanding of the learning performance.

Table 6: Configurations of models 1-4

	RNN cell	Bidirectional?	Embedding layer?
Model 1	`nn.GRU`	no	yes
Model 2	`nn.LSTM`	no	yes
Model 3	`nn.LSTM`	yes	yes
Model 4	`nn.LSTM`	no	no

Training and Testing: Same as the baseline model, you are required to keep track of the training and validation losses of all four models. You also need to report the test accuracies of the four models as what you did for the baseline model. It will be very useful for you to *save the trained weights of the models, especially model 2*, for the later task.

[Q8] Plot the training and validation losses over time of the *baseline model*, *model 1* and *model 2* in one plot. Briefly explain your observation and suggest some possible reason(s).

[Q9] Plot the training and validation losses over time of *model 2* and *model 3* in one plot. Briefly explain the effect of adding the backward direction in bidirectional LSTM compared to the standard LSTM.

[Q10] Plot the training and validation losses over time of *model 2* and *model 4* in one plot. What is the effect of removing the embedding layer?

[Q11] Rank the performance of the *baseline model*, *model 1 to model 4* according to their test accuracy.

(Hint: you can discuss the training and validation losses, convergence rate, and the gap between losses in your analysis.)

Word Embedding:

From the above analysis, you may notice that the embedding layer plays an important role in text processing. When you train the RNN model, you also train the embedding layer. Such embedding layer is called a *word embedding* in the context of natural language processing. It embeds discrete words into numerical vectors, which carry the meanings of the words. In this subsection, you will investigate the embedding trained from *model 2*.

Load the trained model 2 and compute the L2 distance between the word embeddings of the words:

- 'good' and 'happy'
- 'boring' and 'bad'
- 'good' and 'boring'

[Q12] Report the L2 distances of the three word pairs. What do you observe?

5.3.4 Prediction

In the prediction stage, choose the model you ranked highest in [Q11] and predict the sentiment labels of the movie reviews in `submission.csv`. Save the predictions as a csv file.

[P2] Submit your prediction as a csv file. The format should be the same as `train.csv`: one label column and one text column in the same order. '0' and '1' in the label rows indicate the negative and positive sentiments of a review, respectively.

6 Written Report

Answer [Q1] to [Q12] in one single written report.

7 Some Programming Tips

As is always the case, good programming practices should be applied when coding your program. Below are some common ones but they are by no means complete:

- Using a small subset of data to test the code
- Using checkpoints to save partially trained models
- Using functions to structure program clearly
- Using meaningful variable and function names to improve readability
- Using consistent styles
- Including concise but informative comments

8 Assignment Submission

Assignment submission should only be done electronically using the Course Assignment Submission System (CASS):

`https://cssystem.cse.ust.hk/UGuides/cass/student.html`

There should be five files in your submission with the following naming convention required:

1. **Report** for part 1 and part 2 (with filename `<StudentID>_report.pdf`): in PDF format.
2. **Prediction** for part 1 (with filename `<StudentID>_p1.csv`): in csv format.
3. **Prediction** for part 2 (with filename `<StudentID>_p2.csv`): in csv format.
4. **Part 1 source code and a README file** (with filename `<StudentID>_code1`): compressed into a single ZIP or RAR file.
5. **Part 2 source code and a README file** (with filename `<StudentID>_code2`): compressed into a single ZIP or RAR file.

Note: The source code files should contain all necessary code for running your program as well as a brief user guide for the TA to run the programs easily to verify your results, all compressed into a single ZIP or RAR file. The data should not be submitted to keep the file size small. When multiple versions with the same filename are submitted, only the latest version according to the timestamp will be used for grading. Files not adhering to the naming convention above will be ignored.

9 Grading Scheme

This programming assignment will be counted towards 20% of your final course grade. The maximum scores for different tasks are as follows:

Table 7: [C]: code, [Q]: written report, [P]: prediction

Grading Scheme	Code (60)	Report (32)	Prediction (8)
PART 1: Movie Genre Prediction	(30)	(16)	(4)
Dataset and Dataloader			
- [C1] Dataset Class	4		
- [C2] Image Transformation	2		
- [C3] Training-Validation Split and Dataloader	3		
CNN Model Design			
- [C4] CNN Model + [Q1]	4	+2	
Training and Validation			
- [C5] Loss Function	2		
- [C6] Optimizer	1		
- [C7] Training Loop	4		
- [C8] Validation Loop	4		
- [Q2] Loss Curve		4	
- [Q3] Hyperparameter Tunning		4	
Testing			
- [C9] Test Evaluation + [Q4]	4	+4	
- [Q5] Misclassification Cost and Label-missing Cost		2	
Prediction			
- [C10] Prediction on Submission Images + [P1]	2		+4
PART 2: Movie Review Sentiment Prediction	(30)	(16)	(4)
Dataset and Dataloader			
- [C11] Dataset Class	4		
- [C12] Vocabulary	2		
RNN Model			
- [C13] Baseline Model + [Q6]	4	+2	
- [C14] Model 1	2		
- [C15] Model 2	2		
- [C16] Model 3	2		
- [C17] Model 4	2		
Training and Testing			
- [C18] Loss Function	2		
- [C19] Training Loop	2		
- [C20] Validation Loop	2		
- [Q7] Baseline Loss Curves and Test Accuracy	2	+3	
Empirical Study			
- [Q8] Comparison of Different Recurrent Units		3	
- [Q9] Comparison of Bidirectional and Standard LSTM		2	
- [Q10] Comparison of Use of Embedding Layer		2	
- [Q11] Ranking of Models		2	
- [Q12] Word Embedding	2	+2	
Prediction			
- [C21] Prediction on Submission Movie Reviews + [P2]	2		+4

Late submission will be accepted but with penalty. The late penalty is deduction of one point (out of a maximum of 100 points) for every minute late after 11:59pm. Being late for a fraction of

a minute is considered a full minute. For example, two points will be deducted if the submission time is 00:00:34.

10 Academic Integrity

Please read carefully the relevant web pages linked from the course website.

While you may discuss with your classmates on general ideas about the assignment, your submission should be based on your own independent effort. In case you seek help from any person or reference source, you should state it clearly in your submission. Failure to do so is considered plagiarism which will lead to appropriate disciplinary actions.