COMP5329 DEEP LEARNING ASSIGNMENT 2

Lu Liu
520087608
University of Sydney
lliu9089@uni.sydney.edu.au

Tongxin Shi
520353972
University of Sydney
tshi9092@uni.sydney.edu.au

Yunlin Peng
480014858
University of Sydney
ypen3323@uni.sydney.edu.au

May 20, 2022

ABSTRACT

Multi-label learning, as a traditional machine learning task, has also attracted much attention in recent years. However, there will be the following difficulties: 1. The number of class labels is uncertain. Some samples may have only one class label, and some samples may have dozens or even hundreds of class labels. 2. The class labels are interdependent. For example, the samples containing the blue sky class label have a high probability of containing white clouds. How to solve the dependency problem between class labels is also a big difficulty. 3. The multi-label training set is more difficult to obtain. Also we added on image caption which requires not only complete image semantic understanding, but also complex natural language expression. In our assignment, we provide image embedding (ReNeXt50) and text embedding (Bi-GRU Attention), then combined by concatenate embedding to achieve both multi-label classification and image caption.

***K*eywords** Multi-label Classification, Image Caption

1 Introduction

1.1 Aim and importance

Our assignment aims to implement a multi-label image classifier that predicts the labels of data sample given the image and a short summarised caption.

There is still a huge gap between human and machines' understanding about image. For humans, we can perceive space, time, and emotional information, but for computers, they can only have a deep understanding based on low-abstract, high-abstract information and features based on complex neural network.

In machine learning, the single label classification is an usual task which already have have high achievements. Multi-label learning is a traditional machine learning task which was primarily motivated by the emerging need for automatic text-categorization and medical diagnosis. However, multi-label tasks gain more and more attention in real world problem. Compared with traditional learning problems, the labelling of multi-label data is difficult to achieve, and the larger label space brings higher labelling costs. Also, for image captioning, it is to generate a textual description of an image relative to the content of the image. An AI system not only needs to recognize the picture,

but also needs to understand and interpret the content of the picture it sees, and be able to describe the relationship between the objects in the picture like a human being.

1.2 Dataset description

The datasets provided by the COMP5329 Deep Learning team having are 40000 images, a training CSV which contains the images ID, the labels, and the caption per image. We attempt to train a model which can identify one or more possible labels based on the image and caption of these datasets.

ImageID	Labels	Caption
30000 unique values	1 47% 17 3% Other (14950) 50%	29236 unique values
0.jpg	1	Woman in swim suit holding parasol on sunny day.

Figure 1: Example of Data (training csv)

Figure 2: Example of Data (images)

2 Related Works

2.1 Multi-Label Learning

In [1], the current general classification of multi-classification tasks is introduced, which are Extreme Multi-Label Classification(for label space and feature space may be very huge), Multi-Label with Limited Supervision(can be divided into MLC with Missing Labels, Semi-Supervised MLC, and Partial Multi-Label Learning), Deep Multi-Label Classification, Online Multi-Label Classification(solves the problem that the current Off-line MLC model is difficult to directly use the full amount of data), and Statistical Multi-Label Learning(explore statistical properties of multi-label learning).

According to our assignment, we applied Deep Learning for MLC, which mainly applies Deep Embedding Methods. The powerful fitting ability of deep neural networks allows us to efficiently handle more and more difficult tasks, such as multi-classification problems. In addition, there are Advanced Deep Learning for MLC, such as, CNN-RNN [2], and ML-GCN [3] which have achieved good results, especially ML-GCN introduces MLC with graph neural network GNN.

2.2 Resnet and Resnext

The ResNet model was first introduced in the paper Deep residual learning for image recognition [4]. It is modified on the basis of the VGG19 network, and a residual unit (Figure 3) is added through a short-circuit mechanism. The change is mainly reflected in the fact that ResNet directly uses the convolution of stride=2 for downsampling, and replaces the fully connected layer with the global average pool layer. An important design principle of ResNet is that when the feature map size is reduced by half, the number of feature maps is doubled, which maintains the complexity of the network layers. Compared with ordinary networks, ResNet adds a short-circuit mechanism between every two layers, which forms residual learning. Learning residual can let the network become sensitive, and a small disturbance can make the network produce a larger response, that is, the

optimization algorithm makes the weights in the network approach 0 to achieve the purpose of identity mapping.

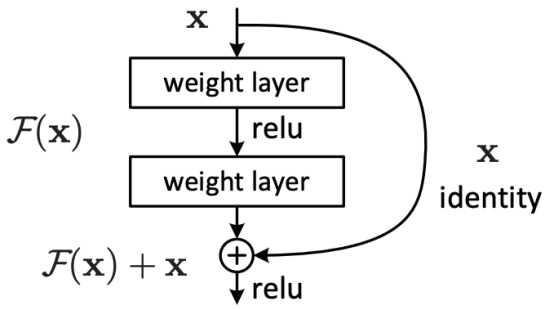

Figure 3: Residual Learning (From *Deep Residual Learning for Image Recognition*)

ResNeXt is an improved version of ResNet introduced in Aggregated Residual Transformations for Deep Neural Networks [5]. The basic structure combines the shortcuts from the block to block, stacking layers and adapts split-transform-merge strategy like Inception.

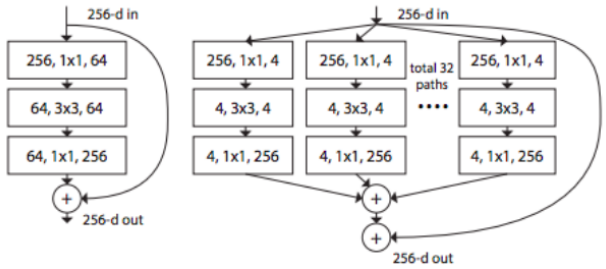

Figure 4: ResNext basic block (Xie et al., 2016)

2.3 Image Caption, GRU and Attention

For image captioning, Encoder-Decoder was first introduced in the field of machine translation. Because it can solve the requirement that the input sequence and the output sequence are strictly equal in length, the source sentence and the target sentence in the machine translation task are often unequal in length [6]. Then [7] proposed the multimodal Recurrent Neural Network (m-RNN), which creatively combines CNN and RNN to solve problems such as image annotation and image sentence retrieval. It uses deep learning methods to solve the Image Caption problem. Then Google introduced the NIC model [8]. Compared with m-RNN, NIC replaces RNN with LSTM. Also the introduction of attention is coming. The attention mechanism can better consider the relationship between objects and assign different weights towards input so that model can focus on more important information. Paper from Xu et.al [9] introduces the attention mechanism in the human visual system into deep learning, and introduces soft attention and hard attention respectively.

3 Techniques

3.1 Pre-processing data

Data pre-processing aims to transfer these raw data to the consolidated format. In this assignment, the data we have contains captions, images and labels. Thus it is necessary to do pre-processing for each section for data separately.

3.1.1 Image Pre-processing

For image preprocessing, we applied image resizing, random horizontal and vertical flip images for data augmentation, and normalize the pixel values.

Data augmentation is a technique of artificially expanding a training dataset by allowing limited data to generate more equivalent data. It is an effective method to overcome the lack of training data, and is currently widely used in various fields of deep learning. In order to do data augmentation based on basic image manipulations, we do both horizontal and vertical flip.

Standardization is to transform the data into a distribution with a mean of 0 and a standard deviation of 1. Generally speaking, the most common normalization for 3-channel RGB images is mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]. This is calculated from the ImageNet dataset and can be used as a generic mean and variance. Standardization can make the distribution of data more uniform, reduce the possibility of the model learning the data distribution, and improve the generalization ability of the model.

3.1.2 Label Pre-processing

For multi-label classification problems, a sample may belong to multiple classes at the same time. Although lists of sets or tuples are a very intuitive format for multi-label data, they are cumbersome to process. Thus we applied MultiLabelBinarizer, which is a converter to convert between this intuitive format and the supported multi-label format.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
4	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
29996	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29997	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29998	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29999	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5: Multi Label Binarizer

3.1.3 Caption Pre-processing

Firstly, we performed punctuation and stop-word removing. Most stop world and punctuation is meaningless for image classification task, so removing those information help improve performance.

Then, Since the training CSV contains text which needs to transfer into the lower case and lemmatize for further processing. Both Lemmatization and transfer to lower case aim to keep words consistent. Lemmaztization is an important part of text preprocessing, which is to remove the affixes of words

and extract the main part of words. Usually, the extracted words will be words in the real-world dictionary. For example, the word "cars" after morphological restoration is "car", the lemmatized word "ate" is "eat".

Then, to achieve the text embedding, we encode the word based on total training caption data and pad to a fixed length. This process does not involve converting words into word vectors, but simply tokenizing plain text, adding [PAD], [UNKNOWN] tokens, and then converting these words Convert to dictionary index.

3.1.4 ResNet50

The proof network of the ResNet network is able to go deeper (with more hidden layers). The ResNet50 network first performs a convolution operation on the input, then contains 4 residuals, and finally performs a full connection operation to facilitate the classification task. The network composition diagram is shown below (Figure 4), and ResNet50 contains 50 conv2d operations. Since the ResNet50 network has more layers, the training time is longer.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 6: ResNet50 (From *Deep Residual Learning for Image Recognition*)

3.1.5 ResNeXt-50 (32x4d)

The following is the configuration list of ResNeXt-50 (32x4d). 32 refers to the number of groups cardinality of the first ResNeXt basic structure entering the network. For 32, 4d means depth, that is, the number of channels in each group is 4 (so the number of input channels for the first basic structure is 128. It can be seen that ResNet-50 and ResNeXt-50 (32x4d) have the same parameters, but the accuracy is higher high.

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

Figure 7: ResNext50 Structure

3.2 EDA

EDA stands for Exploratory Data Analysis, which refers to the exploration of existing data (especially the original data obtained from surveys or observations) with as few a prior assumptions as possible. A data analysis method that explores the structure of data by means of calculating and visualizing feature quantities.

3.2.1 Number of target per image

Firstly, we checked the number of targets per image. The result displays that most images have only 1 target. The maximum target one image can have is 7.

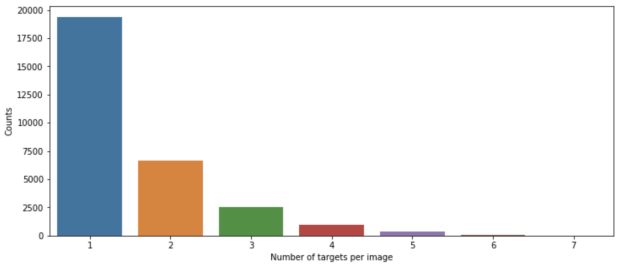

Figure 8: number of target per image

Number of targets	1	2	3	4	5	6	7
counts	19377	6651	2563	978	339	81	11

Figure 9: number of target per image (df)

3.2.2 Number of classes

Since it is a multi class classification, the process to check the number of classes is significant for EDA. The plot obviously showed the unbalance label distribution. Label 1 occupies most of the data, which may affects the model only predict label 1 and ignore other classes. The results also show another problem with our data which is label 12 missing. There is no training data contain label 12, it may cause the model failing to detect label 12 in the testing data.

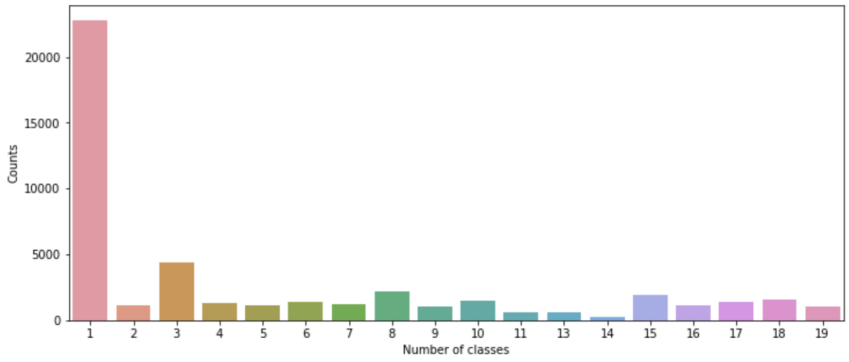

Figure 10: number of classes (Plot)

number of classes	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19
counts	22797	1164	4365	1272	1132	1395	1221	2210	1042	1471	604	605	252	1934	1099	1430	1525	1020

Figure 11: number of classes (df)

3.2.3 Caption length

After preprocessing the data, most caption contains around 3 to 7 words. The maximum caption length is 28 and minimum caption length is 0. Most of caption length of our data are between 2 to 7.

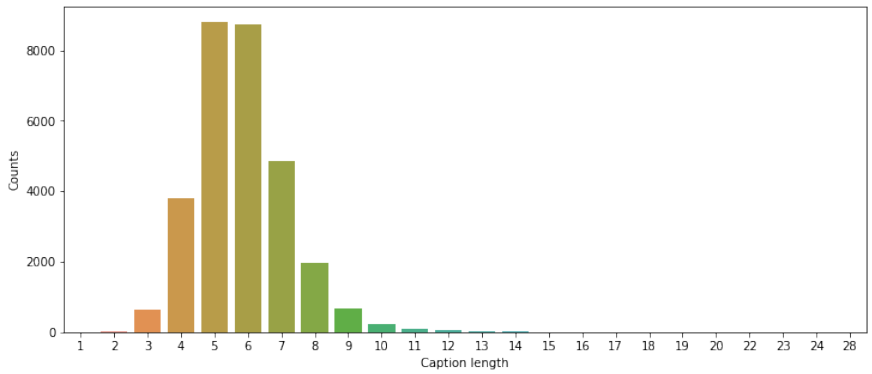

Figure 12: Caption length (Plot)

3.3 Dataloader

DataLoader is a tool in pytorch for processing model input data, combining datasets and samplers, and providing single-threaded or multi-threaded iterable objects on the dataset. We manually decided the training validation split ratio and created three data loaders: train loader, validation loader, and

test loader. The validation is a set of samples set aside separately during the model training process, which can be used to adjust the hyperparameters of the model, to make a preliminary evaluation of the ability of the model. The reason that we need a validation set is that it is used to adjust the model parameters to select the optimal model and to avoid model overfitting the training data and has a bad performance on the testing.

3.4 Loss

Since it is a multi-label classification problem, we use Binary Cross-Entropy loss(BCELoss). BCELoss is used for binary classification and it will generate output as a form of probability.

3.4.1 Cross-entropy Loss versus Binary Cross-Entropy loss

Cross-entropy applied softmax as the method to generate output probability, all probabilities for classed sum to 1. If the probability of one class increases, then the probability of at least one class should be decreased. Thus it is more suitable for multi-class classification, not multi-label classification. The equation of Cross entropy loss is:

$$L = \frac{1}{N}\sum L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic} log(p_{ic}) \tag{1}$$

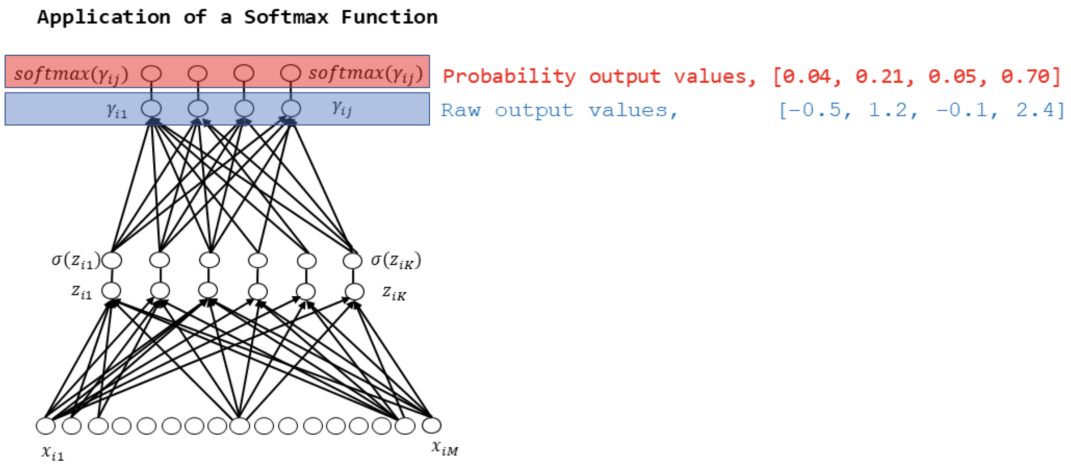

Figure 13: Softmax (From *Multi-label vs. Multi-class Classification: Sigmoid vs. Softmax*)

BCE loss generates output as a form of probability and uses sigmoid to process the single output. The probabilities that are produced are all independent since the sigmoid looks at each raw output value separately. We can just set a threshold to decide whether there is this label in the image, thereby we can achieve multi-label classification. BCEWithLogitsLoss is another subtition loss functaion, it combines sigmoid operations with BCELoss while calculating. The equation of BCELoss is:

$$l(x, y) = L = l_1, l_2, ...l_n \; where \; l_n = -w_n[y_n log\hat{y}_n + (1 - y_n) log(1 - \hat{y}_n)] \tag{2}$$

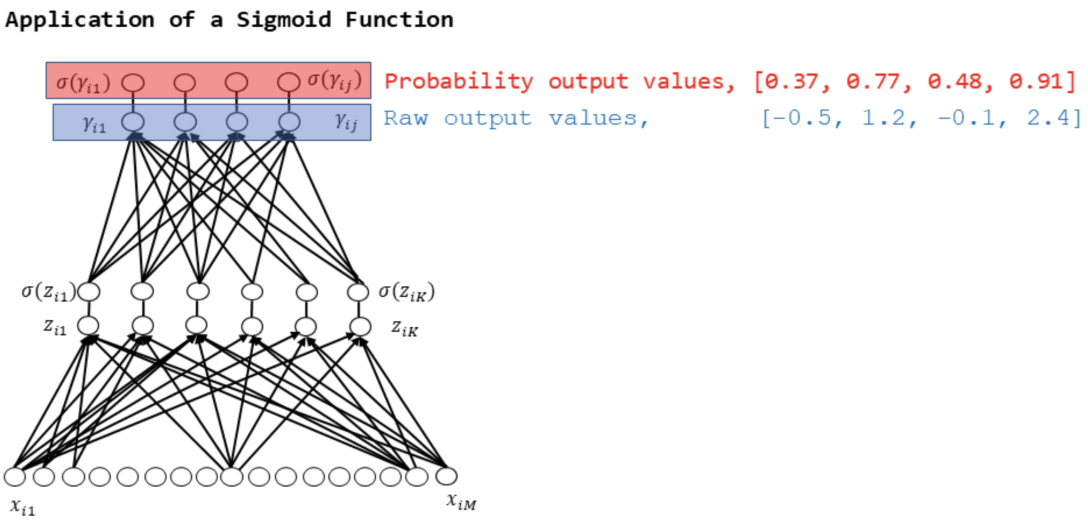

Figure 14: Sigmoid (From *Multi-label vs. Multi-class Classification: Sigmoid vs. Softmax*)

3.5 Embeddings

Embedding is a way of transforming discrete variables into continuous vectors, has greatly expanded the application of neural networks in various aspects. In neural networks, embedding is very useful because it can not only reduce the spatial dimension of a discrete variable, but also represent the variable in a meaningful way.

3.5.1 Image Embedding

An image embedding is a low-dimensional dense vector representation of the image which can be used for many tasks such as classification.

ResneXt50 ResNeXt is a combination of ResNet and Inception. ResNext does not need to manually design complex Inception structural details, but each branch adopts the same topology. The essence of ResNeXt is Group Convolution, which controls the number of groups through the variable Cardinality.

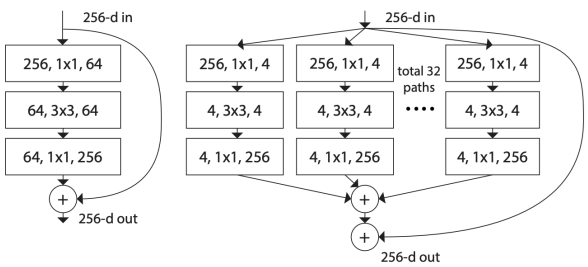

Figure 15: ResNet50 (From *Aggregated Residual Transformations for Deep Neural Networks*)

3.5.2 Text Embedding

Word embedding is a class of methods for text representation. It can express text as a low-dimensional vector, not as long as one-hot. Words with similar meanings are also relatively similar in vector space. And it is very versatile and can be used in different tasks.

Word2vec Word2vec is one of the methods of Word Embedding. CBOW (Continuous Bag-of-Words Model) and Skip-gram (Continuous Skip-gram Model) are two training modes of Word2vec. CBOW predicts the current value through context. It is equivalent to deducting a word from a sentence and letting you guess what the word is. Skip-gram uses the current word to predict the context. It is equivalent to giving you a word and asking you to guess what words may appear before and after. CBOW predicts behavior is almost equal to the number of words in the entire text, the complexity is about O(V), and the speed is faster. However, although skip gram method has more comparisons and performs slowly, but the output word embedding contains more information. So in our model, we compared the influence of CBOW and Skip gram method towards the final result prediction.

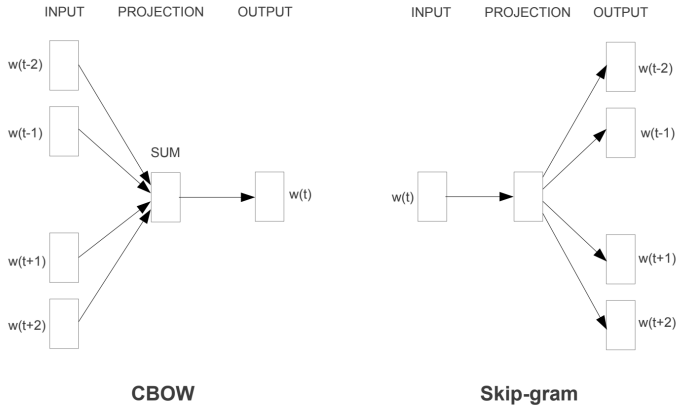

Figure 16: CBOW vs. Skip-gram (From *Efficient Estimation of Word Representations in Vector Space*)

FastText fastText is a word vector calculation and text classification tool open-sourced by Facebook in 2016. Its advantage is that in text classification tasks, fastText (shallow network) can often achieve accuracy comparable to that of deep networks, but it is faster than training time. Deep networks are many orders of magnitude faster. The CBOW model architecture of word2vec is very similar to the fastText model. Word2vec treats each word in the corpus as an atom, and it generates a vector for each word. This ignores morphological features inside words. To overcome this problem, fastText uses character-level n-grams to represent a word. The advantage of this is that the word vector generated by low-frequency words will be better. Because their n-grams can be shared with other words. And for words outside the training lexicon, word vectors for them can still be constructed. We can stack their character-level n-gram vectors.

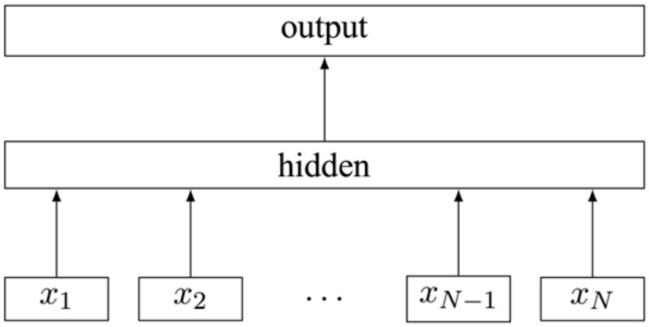

Figure 17: fastText (From *FastText: stepping through the code*)

BiLSTM BiLSTM is the abbreviation of Bi-directional Long Short-Term Memory, which is composed of forward LSTM and backward LSTM. It is often used to model contextual information in natural language processing tasks. Longer-distance dependencies can be better captured using an LSTM model. LSTM can learn which information to remember and which information to forget through the training process, but there is still a problem with using LSTM to model sentences: it cannot encode information from back to front. BiLSTM can better capture bidirectional semantic dependencies.

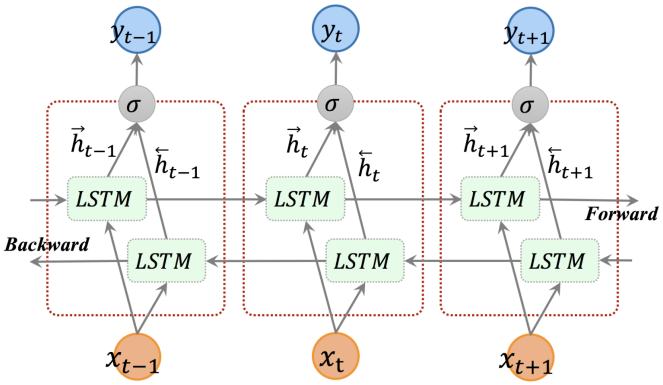

Figure 18: BiLSTM with three consecutive steps (From *Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction*)

BiGRU-Attention GRU (Gated Recurrent Unit) is also called gated recurrent unit structure. It is also a variant of traditional RNN. Like LSTM, it can effectively capture the semantic correlation between long sequences and alleviate the phenomenon of gradient disappearance or explosion. At the same time, its structure and calculation require Simpler than LSTM. BiGRU is a neural network model composed of unidirectional and opposite GRUs whose output is jointly determined by the states of the two GRUs. At each moment, the input provides two GRUs in opposite directions at the same time, and the output is jointly determined by these two unidirectional GRUs. Bi-GRU and Bi-LSTM have the same logic, they do not change their internal structure but apply the model twice in different directions, and then splice the LSTM results obtained twice as the final output. The BiGRU-Attention model has a total of It is divided into three parts: text vectorization input layer, hidden layer, and output layer. Among them, the hidden layer consists of three layers: BiGRU layer, attention layer, and Dense layer (full connection layer).

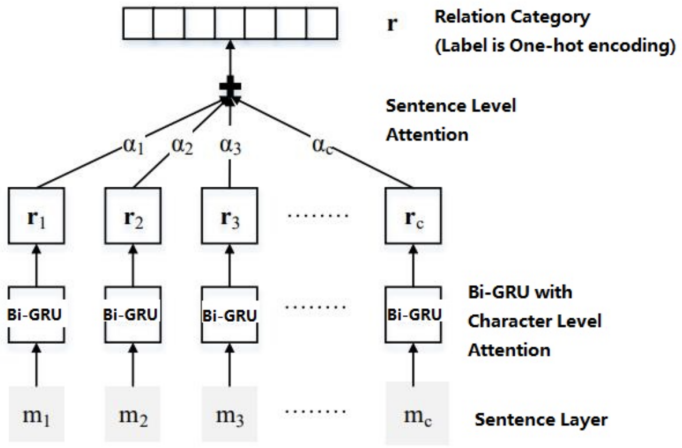

Figure 19: BiGRU-Attention (Modified by *Neural Relation Extraction with Selective Attention over Instances*)

3.5.3 Final Embedding Concatenation

The merge modes for network layers are "sum", "mul", "concat", "ave", "cos", "dot". Among them, concatenation is to splicing the output of the layer to be merged along the last dimension, so the output of the layer to be merged is required to be different only in the last dimension. The concatenation operation is a very important operation in network structure design. It is often used to combine features, fuse features extracted by multiple convolutional feature extraction frameworks, or fuse information from the output layer. Concatenation is the merging of the number of channels, that is to say, the features describing the image itself have increased, but the information under each feature has not increased.

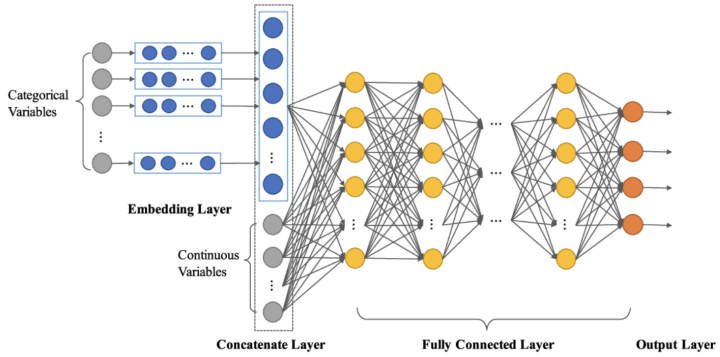

Figure 20: Deep neural networks with entity embeddings. The architecture includes Embedding Layer, Concatenate Layer, Fully Connected Layer and Output Layer. (From *Travel Mode Choice Prediction Using Deep Neural Networks With Entity Embeddings*)

3.5.4 Early Stopping

The process of training the model is the process of learning and updating the parameters of the model. This parameter learning process often uses some iterative methods, such as the gradient descent learning algorithm. Early stopping is a method of truncating the number of iterations to prevent overfitting, that is, stopping iterations before the model converges on the training dataset iteratively to prevent overfitting. The specific approach of the Early stopping method is to calculate the accuracy of the validation data at the end of each epoch, and stop training when the accuracy is

no longer improved. This approach is very intuitive, because the accuracy is no longer improved, and it is useless to continue training, it will only increase the training time. The way to make sure that the validation is no longer improving is to record the best validation accuracy so far during the training process. When 10 consecutive Epochs (or more) fail to achieve the best accuracy, it can be considered that the accuracy is no longer improving. At this point, the iteration can be stopped.

3.5.5 Novelty of model

In our model, we not only used image embedding, but we also incorporated textual embedding so that the information from the whole dataset can be combined to make the final prediction. For text embedding extraction, we combined traditional sequence extraction model (Bi-direction GRU) and attention layer to focus more on relevant part of text towards final result prediction.

4 Experiments

4.1 Evaluation metrics

The evaluation metrics in this multi-label classification task is the Mean F1-Score. The F1 score measures the accuracy by combining other two evaluation metrics, the precision denoted as p and the recall denoted as r, with equal weights. As the result, model with reasonably favourable performance on both precision and recall will outperform models with extremely favourable performance on one of these two evaluation metrics or extremely unfavourable performance on the other. The formula of F1 score is defined as follows:

$$F1 = 2 * \frac{p * r}{p + r} \tag{3}$$

where $p = \frac{tp}{tp+fp}, r = \frac{tp}{tp+fn}$, and *tp*, *fp*, and *fn* stand for true positive, false positive, and false negative.

- **True Positive (TP)**: the predicted class and true class are all positive class
- **True Negative (TN)**: the predicted class and true class are all negative class
- **False Positive (FP)**: the true class is negative while the predicted class is positive
- **False Negative (FN)**: the true class is positive while the predicted class is negative

For this multi-classification task, the Mean F1-Score will be the mean values of the F1 score in all classes.

4.2 Ablation Study

In this section, the effect of the caption, different word embedding models, attention layer in the text model, and the batch normalization on this task will be examined. We will start with a base model with the learning rate of 3e-4, the hidden dimension of 128, the word embedding obtained from the FastText model with the skip-gram with the dimension of 150, the image and text embedding dimension of 256, the batch size of 64, and with an attention layer in the text model. When testing the effect of each component, only the corresponding component will be changed, and others will remain the same in order to conduct a control variable experiment. In order to examine the model performance, the training dataset will be divided into the training set and the validation set with the ratio of 80% and 20%. The training set will be used to train the model, and the validation set will be used to test the model performance. The training loss, validation loss, and mean validation f1 score will be used as the comparison method for determining the optimal model in each component

comparison. Finally, the model with the optimal component in each comparison will be chosen to obtain the best model in this section.

4.2.1 Caption

This part is to evaluation whether caption information matters for the final class prediction. In this part, 4 models in total will be compared, the model without caption, the model with caption obtained from FastText with skip-gram, the model with caption obtained from word2vec with skip-gram, and the model with caption obtained from word2vec with cbow. This is because different word embedding methods can generate the word embedding with information from different dimensions, and it is important to examine whether the caption really enhances the model performance, and which types of the word embedding method result in the best performance.

Table 1: The effect of caption

Feature	Training loss	Validation loss	Validation f1 score
without caption	0.029	0.124	0.792
FastText (skip-gram)	0.068	0.079	0.871
Word2vec (skip-gram)	0.591	0.589	0.688
Word2vec (cbow)	0.575	0.577	0.648

Table 1 reveals that the model with captions with the word embedding generated by FastText with skip-gram outperforms other models with the mean f1 score of 0.871. Moreover, the mean f1 score for the model without analyzing captions is 0.79 and for model with word embedding generated by word2vec is about 0.67. This implies that the model with a suitable word embedding types can enhance the image-only model performance significantly.

4.2.2 Attention

Attention layer for text embedding extraction is to focus more on important part or more relevant part of whole text. Even though the irrelevant stopwords are removed during preprocessing step, not all information in the text are relevant to the final prediction. From the table, the f1 score of Bi-GRU with attention is 0.09 higher than that without attention, which also validated our assumption that attention layer to focus on important information is significant for our final model.

Table 2: Attention

Feature	Training loss	Validation loss	Validation f1 score
Without Attention	0.137472	0.1302	0.7402
With Attention	0.063095	0.0777	0.8564

4.2.3 Batch normalization

Batch normalization is to normalize all the features value among batch data to the same distribution. It is used to solve the internal covariate shift during the network training process. Deep neural networks involve the superposition of many hidden layers, and the parameter update of each layer will lead to changes in the input data distribution of the upper layer. Through different superposition of layers, the input distribution of the upper layer will change a lot. The upper layer need to constantly re-adapt to the parameter update of the bottom layer, which reduces the efficiency of training. With batch normalization, the distributions are shifted to the same one and can accelerate the convergence of loss. From the table, without batch normalization can increase both training and validation loss, so batch normalization step is significant for the model.

Table 3: Batch Normalization

Feature	Training loss	Validation loss	Validation f1 score
Without Batch Normalization	0.093752	0.1169	0.7573
With Batch Normalization	0.063095	0.0777	0.8564

4.3 Hyperparameter Analysis

In this section, hyperparameters such as the learning rate in the concatenation model, the hidden dimension in the text model, word embedding dimension, image and text embedding dimension in the concatenation model, and the batch size of the concatenation model will be tuned in order to obtain the best model, and the tuning range for each hyperparameters are displayed in Table 4 below.

Table 4: The tuning range for each hyperparameters.

Hyperparameter	Tuning range
learning rate	3e-4, 3e-3, 3e-2
hidden dimension	32, 64, 128
word embedding dimension	50, 100, 150
image and text embedding dimension	64, 128, 256
batch size	32, 64, 128

We will start with a the same base model as the one in the ablation study which is also the best model we obtained from the ablation study. It contains the learning rate of 3e-4, the hidden dimension of 128, the word embedding obtained from the FastText model with the skip-gram with the dimension of 150, the image and text embedding dimension of 256, the batch size of 64, and with an attention layer in the text model. When tuning each hyperparameter, only the value of that hyperparameter in the base model will be changed, and all other hyperparameters remain constant. The final best model will be obtained by consisting of the best hyperparameter values in each hyperparameter based on the tuning results. Besides, the Mean F1-Score on the validation set of each model will be used as the comparison method on the accuracy for each hyperparameter tuning, and the model size of each model will be examined to see if it is larger than 100m, which is used as the comparison method on the efficiency.

4.3.1 Learning Rate

The learning rate in the concatenation model controls the updating rate of parameters in the model. In the learning rate is too small, then more training epochs will be required to obtain the optimal parameter values, and it requires more time to get out of the local minimum if stunk in it. If the learning rate is too, then it might overshoot the minimum which fails to converge. Based on the selected learning rates, large learning rate like 3e-2 did not suit the given feature space and led to a poor performance. Relative smaller learning rate like 3e-4 gave best f1 score (0.871) for validation dataset. The downward trend in Figure 22 as the learning rate increases also reveals this conclusion. Moreover, all models have the size less than 100m.

	learning rate	f1 score	model size
0	0.0003	0.871	96.36
1	0.0030	0.801	96.36
2	0.0300	0.649	96.36

Figure 21: Performance of different learning rates(table)

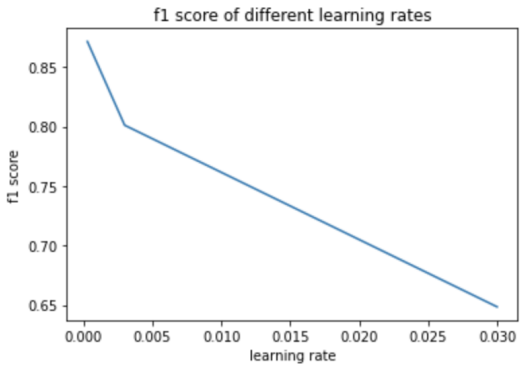

Figure 22: Performance of different learning rates(plot)

4.3.2 Hidden Dimension for text embedding model

This part compares the performance of the mean f1 scores given different hidden dimensions of the text embedding model. Higher hidden dimension can provide more information but the increased model complexity is likely to result in overfitting, while small hidden dimension with limited information is likely to obtain an less desirable performance. The table in Figure 21 implies that high dimension text embedding like 128 contains more information of the captions, thus having the highest f1 score of 0.871. The overall upward trend in Figure 24 as the hidden dimension increases also support this conclusion. Moreover, as the the hidden dimension increases, the model size also increase, but their overall are less than 100m.

	hidden dimension	f1 score	model size
0	32	0.655	95.49
1	64	0.646	95.73
2	128	0.871	96.36

Figure 23: Performance of hidden dimensions (table)

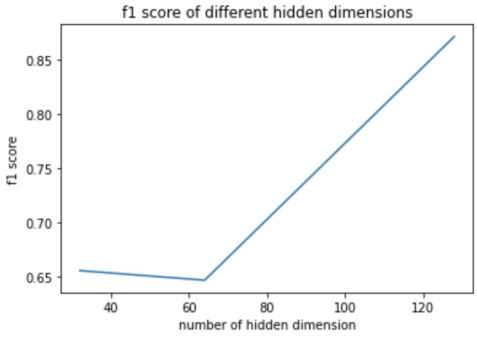

Figure 24: Performance of different hidden dimensions (plot)

4.3.3 Word Embedding Dimension

The word embedding dimension in the text model determine the amount of information of the word that the embedding contains. Small word embedding dimension might result in limited information, but large word embedding might result in large model size. Figure 25 shows that the word embedding dimension of 150 outperforms other two dimensions significantly, which might because of the larger text information it provides highly enhance the concatenation model performance. The plot displayed in Figure 26 shows that the dimension 100 results in the worst performance. Moreover, all models have the size less than 100m.

word embedding dimension	f1 score	model size
50	0.652	93.77
100	0.530	95.06
150	0.871	96.36

Figure 25: Performance of word embedding dimensions

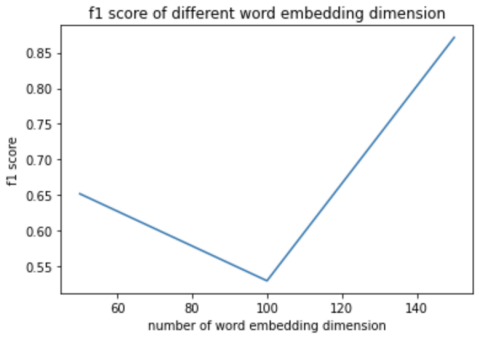

Figure 26: Performance of word embedding dimensions

4.3.4 Image and Text Embedding Dimension

Before image and text embedding actually concatenate together in the concatenation model, their embedding dimensions will be unified. The embedding dimension chosen for unified strongly influences the accuracy and the model size of the final model. Table 27 reveals that the embedding dimension of 256 results in the best performance. Though the embedding dimension of 512 has similar mean f1 score as 256, its model size is 101.60m which is larger than 100m.

Image and text embedding dimension	f1 score	model size
128	0.525	95.61
256	0.871	96.36
512	0.862	101.60

Figure 27: Performance of batch size (table)

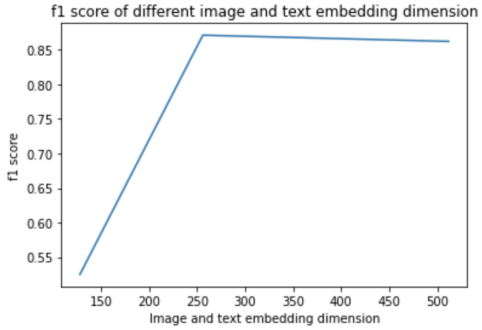

Figure 28: Performance of batch size (plot)

4.3.5 Batch Size

Batch size determines the number of samples for one training iteration. For large dataset, it is not feasible to load all data once towards model. Small batch size can have a deep learning towards each sample while slow the whole training process. Large batch size can speed up training process and improve model's generalization ability, while it may lead to memory explosion. The plot indicates that based on our trials, batch size 64 had the best f1 score 0.871. In general, larger batch size can have a higher f1 score in validation dataset.

batch size	f1 score	model size
32	0.656	96.36
64	0.871	96.36
128	0.867	96.36

Figure 29: Performance of batch size (table)

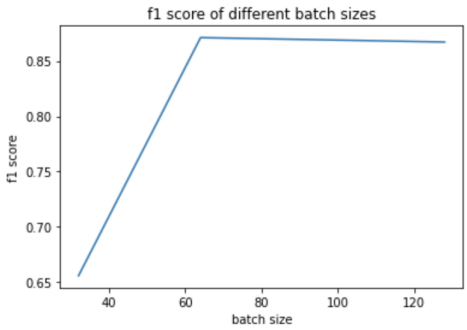

Figure 30: Performance of batch size (plot)

5 The Best Model Performance

The structure of our best model established with multi embedding. We applied the image embedding ResNeXt50 with fully connected layer, and text embedding Bi-GRU with attention layer. Then by adopting concatenated embedding with fully connected layer and activation function sigmoid. It displays in below figure.

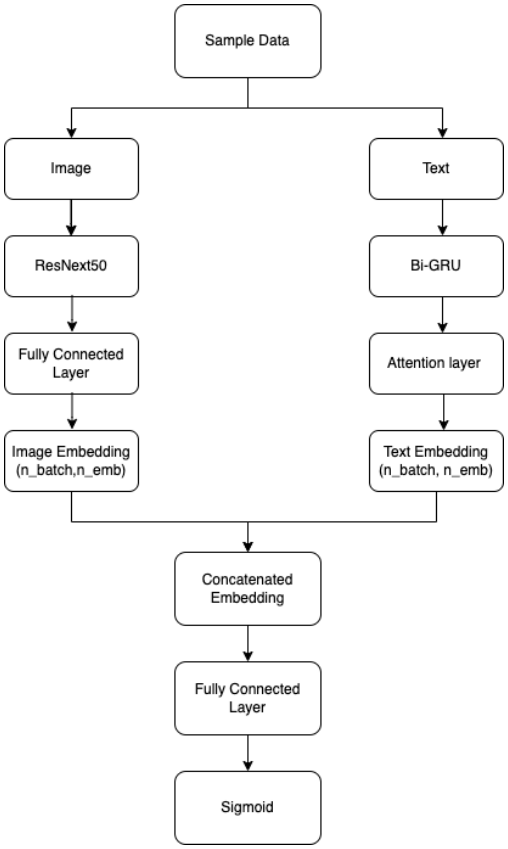

Figure 31: The design of the best model.

After conducting this experiment, the final best model will be learning rate with 3e-4, 128 hidden nodes, 150 word embedding dimension, batch size 64, fastText skip gram as text embedding type, and with attention on the text embedding.

Table 5: Hyperparameter for Best model

Hyperparameter	Value
Learning Rate	3e-4
Number of hidden nodes	128
Word Embedding dimension	150
Image and text embedding dimension	256
Batch size	64
Attention on the text embedding	True
Text embedding type	FastText skip gram

Table 6: Evaluation of Best model

Evaluation	Precision	Recall	f1-score
micro average	0.91	0.78	0.84
macro average	0.86	0.63	0.72
weighted average	0.90	0.78	0.83
samples average	0.93	0.85	0.87

Table 7: Evaluation of Best model for each class

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Precision	0.94	0.85	0.76	0.93	1.00	0.97	0.97	0.71	0.91	0.83	0.97	0.00	0.90	0.91	0.89	0.89	0.97	0.95	0.97
Recall	0.94	0.43	0.58	0.72	0.92	0.72	0.87	0.44	0.64	0.49	0.63	0.00	0.54	0.62	0.33	0.63	0.89	0.78	0.87
F1-score	0.94	0.57	0.66	0.81	0.96	0.83	0.92	0.55	0.75	0.62	0.76	0.00	0.68	0.74	0.48	0.74	0.93	0.86	0.92

The table 6 above presents the performance of the best model on different evaluation metrics. For all micro average, macro average, weighted average and sample average, the precision and recall all received high score, but precision is higher than recall. It reflects the false positive is less than false negative in our prediction. The overall macro average is less than micro average, which presents that the score of contributions in all categories is higher than the score of the evaluation metrics are calculated for each class. The weighted average provide a score which set different weight on each class. Since we already noticed that there is the classes imbalance problem in the dataset which also show in table 7. The precision, recall and f1-score of class 12 is equal to 0 because there is missing label 12 in training dataset. We can see the overall weighted average is just lower than micro average and sample average, which provide the high precision, recall and f1-score. Moreover, the f1-score is highest for class 5, indicating that this model work best at predicting class 5. Thus weighted average provide more valuable evaluation. Furthermore, it takes about 2339 seconds which is about 39 minutes to run the model, and the model size is 96.39m which is less than 100m. In addition, the model results in the mean f1 score of 0.86299 in the 20% of the test set as displayed in the Kaggle.

6 Conclusion

In this project, we implemented multi-label classification based on a multi-modal dataset which contained image and caption information. The best model combines image embeddings extracted from a pretrained ResNext50 model and text embeddings extracted from Bi-GRU + Attention model. After concatenating them together and pass the embeddings through fully connected layers and sigmoid layer, we set threshold to justify whether the label belonged to the sample data. The final result has validation f1 score 0.87 and test set score of 0.86299 in the Kaggle as well as the model size of 96.36m and the running time of 39min.

7 Improvement

For improvement of our model, the point we can focus on is missing label(label 12) within training dataset which cause the prediction of the particular label cannot be identified. For simple solution, we can enlarge the dataset and include more training data. Current study also [16] explored a method to solve multi-label learning with missing data. They proposed a unified model of label dependencies by constructing a mixed graph to formulate this learning problem transductively as a convex quadratic matrix optimization problem. Moreover, we can use other methods to solve the data imbalance problem, like data upsampling for multi-modal data and implemented class importance weight to focus more on class with less recall. Finally, we can not only implement attention layer in text embedding extraction, but also use attention layer in the concatenated features to selected the most relevant image or text features.

8 Appendix

8.1 How to run the code

Please open the notebook using the google colab, and run directly. If you want to run the best model, then run all code block before the section called experiment, and it will automatically generate the submission csv file that contains all predicted class for the test set, and the mean f1 score for the best model is named as best_f1_score. If the data cannot be download to the colab, please and upload the train.csv, test.csv, and data.zip which contains all images into the google drive, and obtain the share id of each of them, and replace the share id in the first code block for each file with yours.

8.2 Write report

We used Latex to write the report.

8.3 Hardware and software specification of the computer

Table 8: Hardware and software specification

Components	**Settings**
CPU	2 GHz Dual-Core Intel Core i5
Disk space	256GB
Memory	8GB
Operating System	MacOS
Application	Python 3.7.6
Package	Numpy, Pandas, Matplotlib, zipfile, math, time, Pytorch

References

[1] Liu, W.; Shen, X.; Wang, H. Tsang, I. W. (2020), 'The Emerging Trends of Multi-Label Learning.', CoRR abs/2011.11197 .

[2] Wang, J.; Yang, Y.; Mao, J.; Huang, Z.; Huang, C. Xu, W. (2016), 'CNN-RNN: A Unified Framework for Multi-label Image Classification' , cite arxiv:1604.04573Comment: CVPR 2016 .

[3] Chen, Z.-M.; Wei, X.-S.; Wang, P. Guo, Y. (2019), Multi-Label Image Recognition With Graph Convolutional Networks., in 'CVPR' , Computer Vision Foundation / IEEE, , pp. 5177-5186 .

[4] He, K.; Zhang, X.; Ren, S. Sun, J. (2015), 'Deep Residual Learning for Image Recognition' , cite arxiv:1512.03385Comment: Tech report .

[5] Xie, S.; Girshick, R. B.; Dollár, P.; Tu, Z. He, K. (2017), Aggregated Residual Transformations for Deep Neural Networks., in 'CVPR' , IEEE Computer Society, , pp. 5987-5995 .

[6] Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H. Bengio, Y. (2014), 'Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation' , cite arxiv:1406.1078Comment: EMNLP 2014 .

[7] Mao, J.; Xu, W.; Yang, Y.; Wang, J. Yuille, A. L. (2014), 'Explain Images with Multimodal Recurrent Neural Networks.', CoRR abs/1410.1090 .

[8] Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2022. Show and Tell: A Neural Image Caption Generator. [online] arXiv.org. Available at: <https://arxiv.org/abs/1411.4555> [Accessed 19 May 2022].

[9] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. and Bengio, Y., 2022. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. [online] arXiv.org. Available at: <https://arxiv.org/abs/1502.03044> [Accessed 19 May 2022].

[10] scikit-learn. 2022. sklearn.preprocessing.MultiLabelBinarizer. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html> [Accessed 19 May 2022].

[11] Draelos, V., 2022. Multi-label vs. Multi-class Classification: Sigmoid vs. Softmax. [online] Glass Box. Available at: <https://glassboxmedicine.com/2019/05/26/classification-sigmoid-vs-softmax/> [Accessed 19 May 2022].

[12] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2022. Efficient Estimation of Word Representations in Vector Space. [online] arXiv.org. Available at: <https://arxiv.org/abs/1301.3781> [Accessed 19 May 2022].

[13] Cui, Z., Ke, R., Pu, Z. and Wang, Y., 2022. Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values.

[14] Lin, Y., Shen, S., Liu, Z., Luan, H. and Sun, M., 2022. Neural Relation Extraction with Selective Attention over Instances.

[15] Ma, Y. Zhang, Z.-J. (2020), 'Travel Mode Choice Prediction Using Deep Neural Networks With Entity

[16] Embeddings.', IEEE Access 8 , 64959-64970. Ieeexplore.ieee.org. 2022. ML-MG: Multi-label Learning with Missing Labels Using a Mixed Graph. [online] Available at: <https://ieeexplore.ieee.org/document/7410830/> [Accessed 20 May 2022].