

# From image to emotion : Multi-label image classification based on Triplet Loss

Youngeun Choi  
2020-29885  
yechoi7@snu.ac.kr

Jeunghyun Lee  
2020-25946  
jhleeangel@snu.ac.kr

Junghyun Ryu  
2021-26380  
jhryu30@snu.ac.kr

## Abstract

*This paper provides a novel approach to multi-label image classification using triplet loss embeddings. The data for our work are advertisement images and their corresponding sentiment labels. We extracted features from the images with a pre-trained model (ResNet 50) and used them as initial states in the next step. The extracted features were fed to embedding layers consisting of two fully connected layers. We used triplet loss for training and got the final embedding size of 256 (batch size)  $\times$  256 (embedding dimension). For classification, we added a softmax classifier at the end of our embedding layer. Since triplet loss is not used for training classifiers, we adopted binary cross entropy loss function for training and separately trained while freezing all the parameters from the embedding layers. The experimental results are still behind compared to the baseline model, yet our proposal has its significance as an attempt to integrate various concepts from the lecture to solve the issue.*

## 1. Introduction

Expressing oneself with images is prevalent these days, perhaps even more than texts. Millions of new images are posted on social network services everyday. Enterprises are eager to collect information from such posts and try to follow the preference trend. With the help of a fine model which reads exact emotions from all those images, one may find a way to figure out the needs of ordinary people. Here, as a team, we proposed some ideas and conducted experiments on classifying images according to their representing emotions to address the mentioned issue.

We chose the Pittsburgh advertisement image dataset as a target of analysis for two reasons. First, advertisement images are expected to grab people's attention. To make that goal, these images tend to express some dramatic moments. Inferring emotions would be much easier with these images than with the images capturing everyday life. Second, the

Pittsburgh dataset provides labels given by multiple annotators. This leads to a multi-class and multi-label classification problem, which lets us deal with the data imbalance issues, and explore various loss functions and metrics to best fit our data. Our approach is distinguished from previous works in that we used triplet loss to train the embedding layer for a multi-label classification task.

This paper is organized as follows: Section II gives an overview of the dataset. Sections III and VI covers the architecture of our model and the experimental settings respectively. Section V provides the metric and results of the experiment while Section VI concludes the paper with discussion.

## 2. Dataset

The dataset from Hussain et al [1] contains 64882 advertisement images verified by human annotators on Amazon Mechanical Turk and 30340 labeled images. Images are split into 11 subfolders and five of them are labeled. For each image, different annotators of three to five made a list of annotations. Each annotation is categorized into 30 sentiments, such as 'active', 'afraid', or 'alarmed'. Labels were given as a list for each image for an annotator and the project is proceeded with arranged labels in a list.

However as previously introduced in Philli et al [2], sentiment label is heavily skewed towards certain labels, especially to 'creative' which is labeled to '12'. The claim made in reference is that images labeled 'creative' are highly labeled due to the characteristic of advertisement rather than sentiment shown in the image. Philli et al [2] suggested that the preprocessing data by a method of undersampling can lead to higher performance in the task. We found this claim is reasonable and followed the suggestion. We tried to approach this problem with three different methods; one is to use plain data without any undersampling and two different undersampling methods. The first undersampling strategy we took is to undersample data which are above two times the mean of distribution. In order to go one step further, our plan for the next approach is to apply Tomek links

undersampling. This method will be progressed after the mid-report submitted since our prioritized goal is to form a general model for the task.

### 2.1. Default

As a basic experiment, we used the whole plain dataset without any undersampling strategy.

### 2.2. Undersampling

First we attained the mean counted value of each label. Then, the labels above two times above the mean were undersampled. The labels were chosen randomly to go through undersampling.

Each image of given data has the label in the form of a list and we found it is difficult to handle since we have a multi-label, multi-class task. So we re-expressed label one hot encoded form for easier handling the dataset. Then we counted how many images we have for each label and take the mean of counts. For labels with counts above two times above the mean, we randomly chose (count-2\*mean) images for each label and removed them.

After this process, we got the same amount of images with labeled alleviated skewness. We got 1, 4, 12, 14, 18 labels downsized as a result. In particular, we get 'creative' labels shrink about 43 percent.



	image (image_num)	labels	labels meaning in sentiment
(a)	 '0/80990.jpg'	[4, 6, 11, 12, 21, 24, 25] after undersampling : [4, 6, 11, 12, 21, 24, 25]	4. "Alert" : attentive, curious 6. "Amused" : humored, laughing 11. "Conscious" : aware, thoughtful, prepared 12. "Creative" : inventive, productive 21. "Inspired" : motivated, ambitious, empowered, hopeful, determined 24. "Manly" 25. "Persuaded" : impressed, enchanted, immersed
(b)	 '2/97682.jpg'	[1, 4, 9, 12, 30] after undersampling: [1, 4, 9, 30]	1. "Active" : energetic, adventurous, vibrant, enthusiastic, playful 4. "Alert" : attentive, curious 9. "Cheerful" : delighted, happy, joyful, carefree, optimistic 12. "Creative" : inventive, productive 30. "Youthful?" : childlike

Figure 1. example of dataset. The image '80990.jpg' is contained in subfolder-0. The image is labeled to '4', '6', '11', '12', '21', '24', '25'. Each label means 'alert', 'amused', 'conscious', 'creative', 'inspired', 'manly', 'persuaded' correspondingly. Also, the category 'alert' embraces sentiments such as 'attentive' and 'curious'. Labels are not changed after the undersampling process example of an image, image named '97682.jpg'. same as (a). After the undersampling process, labels of image '97682.jpg' are changed. Label '12'(creative) is removed.

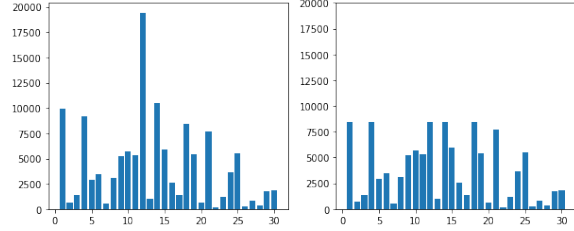


Figure 2. distribution of labels. The x-axis represents label categories and y-axis represents counts for each label. The bar chart shows the distribution of labels before (left) undersampling and after (right) undersampling.

### 3. Related Works

For the baseline of our research, we referenced the work from Pilli et al(2020) [2]. The same Pittsburgh advertisement image dataset was used in the referenced work. The architecture proposed in this paper is as follows.

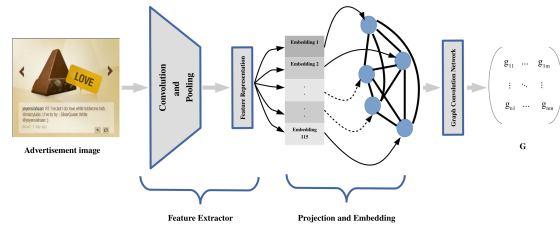


Figure 3. Reference architecture [2]

The authors used ResNet 152 for feature extraction and projected the feature vector into sentiment label word embeddings to fully exploit the context of each sentiment label. For embedding initialization, Google news pre-trained word embeddings were used. Graph Convolutional Network is then employed to exploit the semantic relations between the sentiments of the advertisements. Not only this paper highlights the importance of semantic relations between labels, it also shows the promising results for the multi-modal multi-label image classification task.

### 4. Model architecture

To clarify and to fully understand the research subject, we started by running a simplest classification model. Since the dataset consists of advertisement images, we followed the general advice of transfer learning by using one of the pre-trained models. After several trials for data exploration, We formulated the model architecture with two different training stages.

The first part includes training embedding layers with triplet loss. In most cases the image feature is extracted from the pre-trained model and this feature vector is passed

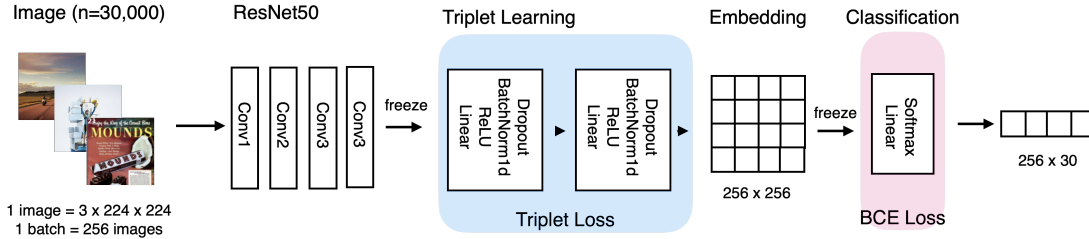


Figure 4. Architecture of the proposed model

directly to the classifier. However, we intended to provide the notion of similarity and dissimilarity to the extracted feature vector and replaced the top head classification of ResNet50 with two fully connected layers. These layers were trained with the triplet loss. As for the triplet sampling, the detailed description will be given in section 3.1.1. For the latter stage, a simple softmax classification layer was employed and we trained the layer with Binary-cross entropy. Considering our dataset has multi-labels, we binarized all the labels for each image beforehand. Below is the architecture we propose in this paper.

Following is a detailed description of each training process. The first training step aimed to shade relationships between each label onto the embeddings extracted from the input images. By the first step, we were able to enrich the contextual link of the embedding. After freezing the pre-trained model, we trained the embedding layer with triplet loss. The second training took place on the embedding layer using the binary cross entropy loss function to go through the softmax layer as a classifier.

### 4.1. Triplet Learning

We applied a technique called triplet sampling to achieve embedding more effectively[3]. Each triplet is composed with three elements, (anchor, positive, negative). Triplet sampling is progressed based on triplet loss function.

### 4.2. Triplet Loss

Triplet loss( $L$ ) is given as

$$L = \max(\|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|_2^2 + \alpha, 0) \quad (1)$$

where  $A, P, N, \alpha$  is anchor, positive, negative, triplet margin correspondingly and  $f(\cdot)$  is the image embedding function that maps an image to a point in a Euclidean distance in embedding space. Through minimizing the loss, we intend to learn the embedding function  $f(\cdot)$  that makes the embedding distance between anchor points and positive points to be closer than the distance between anchor points and negative points [3].

### 4.3. Triplet Sampling

Since our task deals with a multi-labeled multi-class dataset, we had to carefully sample triplets to ensure more precise embeddings. We referred to an existing code (<https://github.com/abarthakur/multilabel-deep-metric>). In general multi-class tasks, positives and negatives are selected by whether they belong to the same class as an anchor. However in the multi-label setting, we needed an advanced method. Hence we took similarity between labels into account to select positives and negatives. First, all data points were explored within a mini-batch and set as anchors. Next, we selected points to be positive if the point has a similarity score higher than the value set in advance. Here we computed similarity scores by inner product of among the labels.

For the hard negative mining, we picked the samples with closer distance with  $x$  embedding with similarity bigger than 1 but less similar than the positive. In other words, negative samples selected in this way are actually closer to the anchor in the embedding space, but their label similarities are lower compared to the positive example. This means that these samples are not placed properly in the embedding space and need to be moved to the proper place according to label similarities. We additionally considered one easy negative with small similarities. As embedding is not well performed for easy negative and too hard negative, we increased the proportion of comparably semi-hard negatives. Through this process of generating negatives per positive, we design to create a triplet for the next anchor when it exceeds the preset value.

## 5. Experiments

We ran our model on the provided server and used 70 percent of the dataset as a training set, 20 percent of the dataset as a validation set and 10 percent of the dataset as a test set. Our experiment was done with the Pytorch version 1.8.1+cu102 setting. Train time was about 250min per each experiment.

We conducted 4 different experiments with altering hy-

	Number of epochs	Triplet Margin
Experiment1	10	1
Experiment2	10	2
Experiment3	20	1
Experiment4	20	2

Table 1. Configurations of the conducted experiments.

perparameters. The experimental configuration that we conducted is as follows.

Triplet margin is a hyperparameter of how much more we see the difference between the embedding distance between anchor-positive and anchor-negative in the triplet loss.

The learning rate of all experiments was initially set to 0.01. For experiment (2) and (3), which were conducted with 20 epochs, we reset the learning rate to 0.001 after the 15th epoch because the loss decreases smaller. Hyperparameters other than number of epochs and triplet margin were fixed. Also, while generating triplet samples, we set the maximum number of triplets per anchor as 1, as well as the maximum number for every positive. This generated a single triplet for each anchor and one negative for each positive. All the other parts of the structure of the model were kept identical.

## 6. Results

### 6.1. Experiment Results

Figure 4 above shows that triplet loss decreased as training progressed across all the experiment settings. We could ensure that the training went well, and the loss barely decreased after 10 epochs.

The consistent trend of decreasing loss binary cross entropy classification results also ensure that the classification went well. The tendency increased with the increased epoch.

### 6.2. Evaluation and Metrics

Evaluation protocol is constructed as follows. We use mAP and overall F1 score while evaluating the experiment results and comparing them with the benchmark. Generally in an emotion classifier where multiple labels are assigned to a single output after classification, different metrics are applied to evaluate the model performance. During model prediction, the model takes an image as an input and predicts a vector of probabilities for each label. Here, the probability vector functions as a threshold to obtain a binary vector similar to ground-truth binary vectors.

#### 6.2.1 Accuracy

One of the simplest ways to compute the performance metric of each model is to measure accuracy on exact binary

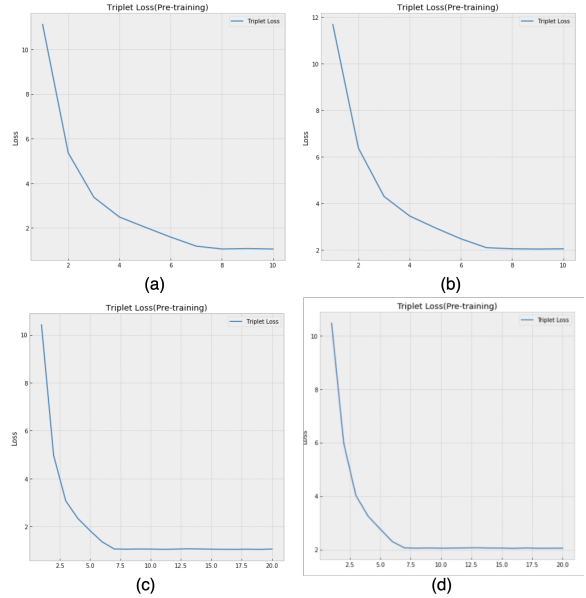


Figure 5. Embedding layer training results, (a) 10epochs, margin 1, (b) 10 epochs, margin 2, (c) 20 epochs, margin 1, (d) 20 epochs, margin 2

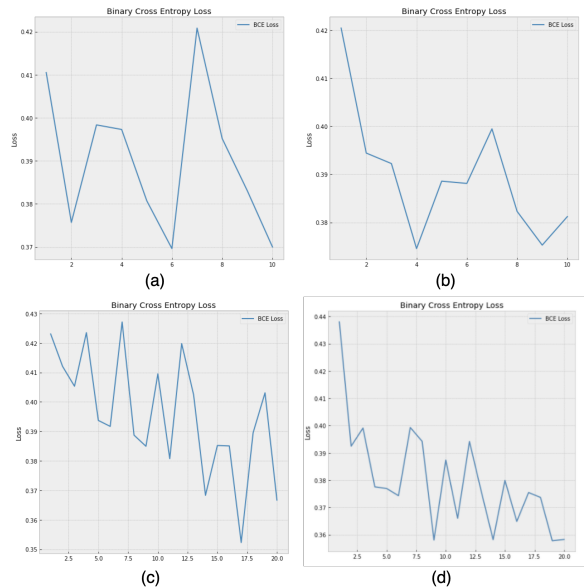


Figure 6. Classification training with BCE Loss, (a) 10epochs, margin 1, (b) 10 epochs, margin 2, (c) 20 epochs, margin 1, (d) 20 epochs, margin 2

vector matching.

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN} \quad (2)$$

If we look back at the table where we had FP, FN, TP,

and TN counts for each of our classes. We can sum up the values across classes to obtain global FP, FN, TP, and TN counts for the classifier as a whole. This would allow us to compute a global accuracy score using the formula for accuracy. Yet, accuracy is not the best performance metric for our dataset. As it simply computes the number of correct predictions over the total amount of samples, partial errors are not taken into account. For instance, even though the model predicts two out of three labels in the image correctly, it doesn't get any credit for that since accuracy counts the prediction as incorrect. In other words, this method of performance evaluation is therefore too penalizing for not tolerating partial errors of the model.

### 6.2.2 F1 score

F1 score for a certain class in image classification refers to the harmonic mean of its precision and recall. Widely used as an overall measure of the quality of a classifier's prediction, we also computed macro-average F1 score for the model. The metric averages the precision and recall scores over each class. We decided to use macro-average f1 score rather than micro-average as the data we have is imbalanced. Macro-averaging weighs each of the classes equally which tackles the issue with imbalanced data that occurs when the classifier performs well on certain classes while not on the others.

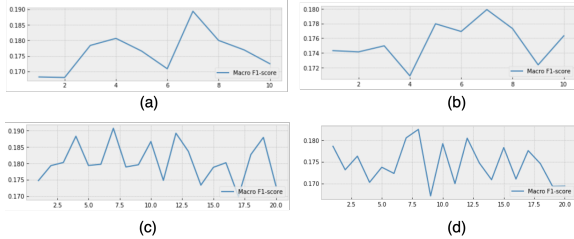


Figure 7. F1 score of Training for classifier, (a) 10epochs, margin 1, (b) 10 epochs, margin 2, (c) 20 epochs, margin 1, (d) 20 epochs, margin 2

### 6.3. mAP score

Also, in this part, an average precision (AP) will be computed for each model during experiments. Average precision is one of the most widely used metrics these days to evaluate and compare the performance of algorithms. Precision represents the amount of successful predictions over all. Recall, also known as sensitivity, is the amount of successful predictions over the ground truths.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{allpredictions} \quad (3)$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{allGroundTruths} \quad (4)$$

PR curve is a method of evaluating the performance of a classifier due to changes in the threshold value for the confidence level. The confidence level tells how confident the algorithm is about what it does. The area under the PR curve is the AP value. Since Precision and Recall are often complementary, the area under this curve is an indicator of the overall performance of the model. The mean average precision can be calculated by taking the mean AP values over all classes and overall thresholds (0.5). Note that the reported results below took validation sets into account.

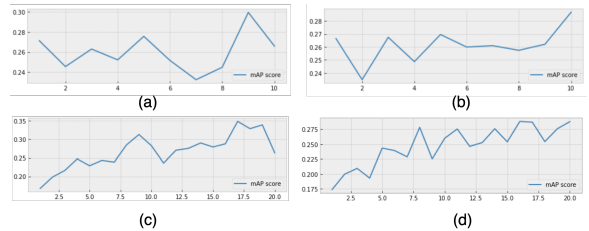


Figure 8. mean Average Precision score of Training for classifier, (a) 10epochs, margin 1, (b) 10 epochs, margin 2, (c) 20 epochs, margin 1, (d) 20 epochs, margin 2

	mAP	f1 score
Pilli (2020)	0.233	0.399 (overall)
GCN (2019)	0.332	0.470 (overall)
Experiment 1	0.271	0.179 (macro)
Experiment 2	0.273	0.176 (macro)
Experiment 3	0.252	0.175 (macro)
Experiment 4	0.257	0.177 (macro)

Table 2. Model evaluation with benchmark and experimented results

## 7. Conclusion

To best our knowledge, this paper is the first attempt to integrate transfer learning and triplet training. Initially we extracted positional and semantic information from the image with the pre-trained ResNet50. In addition to that, we employed the triplet loss to build embedding layers to give the notion of similarity and dissimilarity between the points.

If we have a chance to further develop the idea that we propose, we could proceed with more experiments. Applying contrastive loss and see if there is a significant difference between the two settings. Also we can consider extracting the text from the advertisement image and exploit this information and move the focus from the multi-label classification to multi-modal classification problem.

## References

- [1] Z. Hussain, M. Zhang, X. Zhang, K. Ye, C. Thomas, Z. Agha, N. Ong, and A. Kovashka. Automatic understanding of image and video advertisements. *CVPR*, 2017.
- [2] S. Pilli, M. Patwardhan, N. Pedanekar, and S. Karander. Predicting sentiments in image advertisements using semantic relations among sentiment labels. *CVPR*, 2020.
- [3] J. Wang, Y. song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. *IEEE conference on computer vision and pattern recognition*, 2014.