ComposeTransformers: an end-to-end woman clothing recommender system via image-text content retrieval

Viet-Hoang Tran[1], Ngoc-Ba-Thi Hoang[2]
Faculty of Computer science
University of Information Technology, VNU-HCM

Email: {18520785[1], 19522255[2]}@gm.uit.edu.com

December 27, 2021

Abstract

This article introduces ComposeTransformers, the end-to-end recommender system for woman clothing via image-text content retrieval.

Keywords: Vision Transformer, BERT, multi-modal search.

1 Introduction

Over the last few years, E-commerce has become an indispensable part of the global retail framework. Thanks to E-commerce exchanges, people are able to buy anything from anywhere with just some clicks and touches on the screen. Developers have tried their best to make the buying and selling procedures become more convenient. However, finding desired products is not always easy, especially with fashion products. Thus, we introduce ComposeTransformers - an en-to-end method with AI - to solve the problem of retrieving suitable clothes that match the query text and the sample image given by the user. For the first step towards shirt products, our work implement a subset of "Fashion200K" dataset consisting of 23K women clothing images of categories: shirt and t-shirt, along with their associated product descriptions from online shopping websites. Empirical results show that our method reaches a good result on the small-scale database. Specifically, when performing 1200 queries and taking top 50 search results, we retrieve 55.42% of relevant images on the total of 2,646 test images on the database.

1.1 Objective

Our work research on a clothing recommender system which user can give a query text combine with a query image and then receive images of clothes which come from the database and match the image-text query from user.

1.2 Problem Formulation

Given the set of query images $X = \{x_1, x_2, ..., x_n\}$ and corresponding query texts $T = \{t_1, t_2, ..., t_n\}$ denotes the set of corresponding query texts. The database $Y = \{y_1, y_2, ..., y_m\}$ denotes the set of target images. Our goal is finding the mapping function from the pair of query image and text (x, t) to the set of semantically relevant target images $Y^\star$ in database Y.

We discuss the inference phase first, let $\alpha(\cdot)$ and $\beta(\cdot)$ respectively denotes as image and text feature extractor, both of these models return feature vectors in d and h dimensional space. Let $g(x, t)$ denotes a ComposeTransformers, the end-to-end framework including the learnable composed representation with the extracted image and text. Along with the $cos(\cdot, \cdot)$ as the cosine similarity kernel with two arguments: the composed query feature and target feature vector, the multi-modal search for any pair of query image and text (x, t) in the inference phase is formulated as:

$$\arg\max_{Y^\star \subset Y} \sum\nolimits_{y \in Y^\star} cos(g(x, t), \alpha(y)) \tag{1.1}$$

For the training phase, the objective is to learn a composed representation of the image-text query (x, t) to the corresponding target image y by the supervised learning method, the composed representation module is denoted by $g(x, t, \theta)$, the training objective is maximizing:

$$\arg\max_{\theta} cos(g(x, t, \theta), \alpha(y)) \tag{1.2}$$

where θ denotes all the network parameters.

1.3 Complex Projection For Feature Composition

Related Works

Feature composition between image and text has been extensively studied in the field of vision and language, especially in the Visual Question Answering (VQA) task. The objective function maximizes the similarity between the output of the composition function of the image-text query and the target image features, various methods for the learnable feature composition have been used by Deep Metric Learning. For the general deep learning concept, researchers aim to formulate the learning problem in the restricted solution space, which is resolvable but still guarantees semantics. This helps in learning better and robust representations. The most related work, TIRG [1] proposes the composition modeling query image and text to the same space as the target image. In short, TIRG emphasizes query image features by putting it directly in the composed representation module along with the query text features for modification in image space rather than entirely different feature space. As proved in the work [2], the drawback of TIRG does not really clarify the important role of text features in defining the relationship between the query image and the target image.

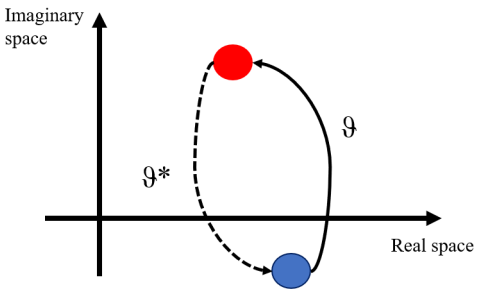

Figure 1: Conceptual diagram of rotation of the images in complex space. We suppose that the composition query features (blue circle) can encounter target image features (red circle) in the complex space by the rotation operation. δ represents the transition in the complex space, learned from query text features. $\delta^{\star}$ represents the reversed transition, learned to satisfy the *rotational symmetry constraint*.

Complex Projection Method

As a further effort to improve TIRG, ComposeAE [2] proposes complex space as the solution space for mapping the composition query features into the target image features. Based on the assumption that the source image and target image are rotations of each other in some complex space. In this context, the role of query text features is transition angle from query image to target image in this space. And the transition operation must be symmetric, i.e. the reversed transition can encode the target image features to query image features also along with the query text features. This reversed constraint is defined in the training phase, which is mentioned as the *rotational symmetry constraint* in the original paper. The effectiveness of this constraint is emphasize the semantic relationship of query image and target image through the query text. Moreover, it allows model to perform retrieval without requiring the query caption. That makes the model to be user-friendly since it does not ask the user to describe the query image.

2 Model Components

Transformers [3] is the modern method of deep learning, are originally invented to address natural language processing problems and having a large dominance in this field. It relies by a simple yet powerful mechanism called Self-Attention. In general, this mechanism contextually updates weights by the relevance of certain information. It helps sequence data like words to be transmitted and processed parallelly. That creates a great advantage since it is possible to take advantage GPU's power to capture the multi-contextual and long-range dependency of words and be able to process data types such as image. In fact, the work [4] proves that with the help of the relative positional encoding [5], the Multi-Head Self-Attention (MHSA) layer can express any convolutional layer. In addition to the role of a language model, this proof brings Transformers-based model as a novel approach for image feature extraction task. In this work, we uniformly employ the

Transformers-based methods for text and image feature extraction.

2.1 ViT $\alpha(\cdot)$

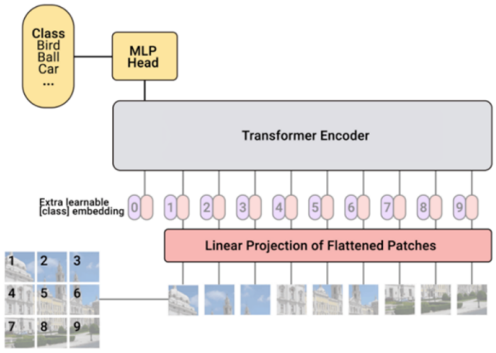

Figure 2: Architecture of ViT [6], the first Transformer-based neural network with convolution free for image classification. As you can see, the input image is splitted into fixed-size patches, so the image size must be divisible by the patch size.

Vision Transformer (ViT) [6] is the first Transformer-based model that approaches state-of-the-art on multiple image recognition benchmarks in the early 2021. In general, ViT treats the image as an input sequence and our expected output is its feature vector. Initially, (1) image is splitted and flatten into a sequence of 2D patches by the linear projection. So each patch of image is now considered to be a sequence of words (as mentioned in its article's title). (2) Analogously as BERT [7], they add an extra embedding [CLS] token for classification task (right after flattening the image) which locates at the top of linear projection's embedding features and all of them are added with 1D positional embedding. The positional embedding preserves location information from input data. (3) The embedded outputs are subsequently fed into Encoder, where the attention mechanism is implemented. The Encoder architecture is similar to the original paper [3]. (4) From the output feature map of the Encoder, ViT gets a transformed [CLS] token located on top of the feature map. In order to use ViT as the image feature extractor, a linear projection layer is employed to transform this token into the compatible dimension.

For the input query image x, we extract the image feature vector living in a d-dimensional space, using the modified ViT with convolution stem in the first Transformer block as the image model $\alpha(x)$, denotes as:

$$\alpha(x) = z \in \mathbb{R}^d \tag{2.1}$$

2.2 BERT $\beta(\cdot)$

BERT (Bidirectional Encoder from Transformer) is an attention mechanism that uses Transformer's Encoder to learn the relations between words in a document. Inside a BERT there are several stacked encoder cells, similar to encoders in transformers. There are various BERT architectures, spanning many different corpus sets in many languages and topics.

2.2.1 Generate text feature vector using BERT

Input

Special Token: [CLS] is used as a very first token added at the beginning of the input sentence. [SEP] is used as a separator between different sentences when multiple input sentences are passed. For example: Pass two sentences "Hello world" and "Welcome to NLP" to BERT. The tokenizer will first tokenize these sentences as: ['[CLS]', 'Hello', 'world', '[SEP]', 'Welcome', 'to', 'NLP', '[SEP]'] and then convert into numerical tokens. Along with [CLS] token, BERT takes 3 types of input:

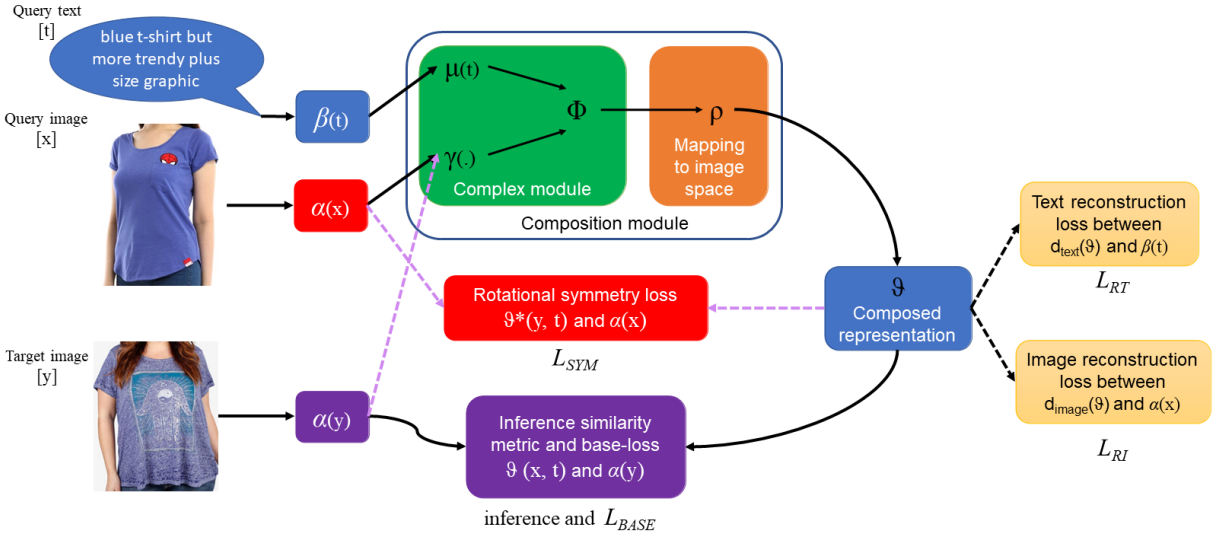

Figure 3: ComposeTransformers architecture for the multi-modal (image-text) retrieval task. Black solid lines indicates the connections for inference and calculate L_{BASE} in the training phase. Black dashed lines indicates 2 reconstruction losses for image and text features L_{RT} and L_{RI}. Violet dashed lines indicates the connection for calculating *rotational symmetry loss* L_{SYM}.

1. Token Embeddings: Token Embeddings are used to first breakdown larger or complex words into simple words and then convert them into tokens.
2. Segment Embeddings: The segment embeddings are used to help BERT distinguish between the different sentences in a single input. The elements of this embedding vector are all the same for the words from the same sentence and the value changes if the sentence is different. For example: Tokenizer will first tokenize the sentence as: ['[CLS]', 'Hello', 'world', '[SEP]', 'Welcome', 'to', 'NLP', '[SEP]']. And the segment embeddings for these will look like: [0,0,0,0,1,1,1,1].
3. Mask tokens: The mask tokens that help BERT to understand what all input words are relevant and what all are just there for padding. Since BERT takes a 512-dimensional input, and suppose we have an input of 10 words only. To make the tokenized words compatible with the input size, we will add padding of size 512 - 10 = 502 at the end. Along with the padding, we will generate a mask token of size 512 in which the index corresponding to the relevant words will have 1s and the index corresponding to padding will have 0s.

Position Embeddings: Finally, Position Embeddings are generated internally in BERT and that provide the input data a sense of order. It is the same as what we discussed in Transformers.

Output

BERT considers only the output corresponding to the first token [CLS] and produces the output feature vectors corresponding to all the other tokens.

2.2.2 Sentence-BERT

Sentence-Bert (SBERT) is a modification of the pre-trained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using Cosine-similarity. SBERT adds a pooling operation to the output of BERT to derive a fixed sized sentence embedding. The pooling strategy is computing the mean of all output vectors (MEAN-strategy). The SBERT model that we used in this article was trained on the combination of the SNLI [8] and the Multi-Genre NLI [9] dataset. The SNLI is a collection of 570,000 sentence pairs annotated with the labels contradiction, eintailment, and neutral. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text. SBERT performed well by reducing the effort for finding the most similar pair from 65 hours with BERT to about 5 seconds, while maintaining the accuracy from BERT.

In this work, given the input text query t, we extract the text feature vector living in a h-dimensional space, using SBERT as the language model $\beta(x)$, denotes as:

$$\beta(t) = q \in \mathbb{R}^h \tag{2.2}$$

2.3 Composition Module

Taken query image features $z \in \mathbb{R}^d$ from section 2.1 and text features $q \in \mathbb{R}^h$ from section 2.2, the objective of this module is to learn a appropriate composition function ϑ that mapping these query features into the complex space $\mathbb{C}^k$.

As discussed in the section 1.3, the role of encoded text features is representing how much rotation is needed to get from source to target image is encoded via the query text features. To achieve this, both query text features and image features learn their own mapping projection into k-dimension before going to the complex space.

First, we formulate $\gamma(\cdot)$ and $\mu(\cdot)$ respectively are projection functions of z and q. In that, $\gamma(\cdot) : \mathbb{R}^d \to \mathbb{R}^k$ and $\mu(\cdot) : \mathbb{R}^h \to \mathbb{R}^k$. They are all implemented in practice as a multi-layer perceptron (MLP). The formulation of this stage can be written as:

$$\hat{z} = \gamma(z) \tag{2.3}$$

$$\hat{q} = \mu(q) \tag{2.4}$$

Secondly, the complex projection operation is performed, which encoded text features influences angle of complex rotation, the complex conjugate $\phi \in \mathbb{C}^k$ is formulated as:

$$\phi = \text{Concat}(\hat{z}cos(\hat{q}), i \cdot \hat{z}sin(\hat{q})) \tag{2.5}$$

where Concat$(\cdot)$ is experimentally defined as the concatenation operation between two vectors. i is the square root of -1, but in practice, we set $i = 1$ when computing the similarity of between the composed features and the target image features extracted from the image model.

Finally, in order to perform the similarity computation task, we learn the linear mapping function $\rho(\cdot) = \mathbb{C}^k \to \mathbb{R}^d$ to project the composed features into the feature space of target image, $\rho(\cdot)$ is also implemented as MLP. The complete composition module is formulated as:

$$\vartheta = \rho(\phi) \tag{2.6}$$

Actually, original work [2] also employs the additional convolutional block $\rho_{conv}(\cdot)$ to capture dominant components of the cross-modal features of data. This module can further emphasize the role of text features

for the cases requiring visual queries. Practically, it may satisfy the user text queries demanding the localized spatial modification on the query images. E.g. in the case of a user requesting Lacoste branding logo on the blank t-shirt. In fact, their work experiment indicates that $\rho_{conv}(\cdot)$ does not bring much effect for their empirical results. In their experimental setup, the output features of $\rho_{conv}(\cdot)$ account for less 10% effect of the final results. In this work, we modify the learning model appropriately to meet the dataset by ejecting this module.

2.4 Reconstruction Modules

In the training phase, we learn to reconstruct extracted image features z and text features q from the composed features ϑ respectively by the image and text reconstruction modules: $d_{image}(.)$ and $d_{text}(.)$, experimentally they are also learnable MLP. These reconstruction modules serves as a regularization method of the learning model, since they encourage the composition module to preserve the relevant information of text and image features in its final representation, i.e. decreasing variation, the reason of over-fitting. We define the reconstructed image and text features by formulations below:

$$\hat{z} = d_{image}(\vartheta_i) \tag{2.7}$$

$$\hat{q} = d_{text}(\vartheta_i) \tag{2.8}$$

where $d_{image}(.) : \mathbb{R}^d \to \mathbb{R}^d$ and $d_{text}(.) : \mathbb{R}^d \to \mathbb{R}^h$ in order to get the same dimension with their corresponding extracted features. The reconstruction modules are only available in the training phase and visualized as yellow blocks in figure 3.

3 Training

3.1 Soft Margin Triplet Loss L_{BASE}

Similar to previous works that addressing Cross-Modal Retrieval problems, Deep Metric Learning (DML) is considered a popular approach for the works classified as supervised learning. When employing DML, our purpose is measuring the similarity between data samples and our goal is helping the model indicates that the distances between samples of the same class are smaller than the distances between the samples of different classes.

Follow the discussed objective in section 1.2, the similarity metric $cos(\vartheta, \alpha(y))$ is defined as cosine similarity kernel between the composed image-text features ϑ and extracted target image features $\alpha(y)$. Follow the original work [2], we implement the DML approach with the soft margin triplet loss [10] as base loss. Generally, for every sample i from the training batch of size N, the principle of this loss function is maximizing the similarity kernel between the composed image-text features ϑ_i and the corresponding extracted target image features $\alpha(y_i)$. It is also encourage the similarity minimization between the ϑ_i and their random-picked negative image $\alpha(\tilde{y}_{i,m})$ (m from their corresponding negative image set of size M) simultaneously from the mini-batch. The formulation is presented below:

$$L_{BASE} = \frac{1}{MN} \sum_{i=1}^{N} \sum_{m=1}^{M} \log \left\{ 1 + \exp \left\{ \cos \left(\vartheta_i, \alpha \left(\tilde{y}_{i,m} \right) \right) \ - \cos \left(\vartheta_i, \alpha \left(y_i \right) \right) \right\} \right\} \tag{3.1}$$

where M denotes the number of triplets for each training sample i.

3.2 Image-Text Reconstruction Losses L_{RI}, L_{RT}

Incorporated with the base loss, the reconstruction losses are implemented by the mean squared error between the extracted features of image and text queries (z_i, q_i) and the output of reconstruction modules ($\hat{z}_i$, $\hat{q}_i$) (as discussed in section 2.4) for every data sample i in the training batch of size N. By formulation, image and text reconstruction losses are respectively given by:

$$L_{RI} = \frac{1}{N} \sum_{i=1}^{N} \|z_i - \hat{z}_i\|_2^2 \tag{3.2}$$

$$L_{RI} = \frac{1}{N} \sum_{i=1}^{N} \|q_i - \hat{q}_i\|_2^2 \tag{3.3}$$

3.3 Rotational Symmetry Loss L_{SYM}

The learning composition module learns to map the features of query and target image features in the complex space with the encoded text features is a transition agent. For further regularizing the module in the practical cases, the *rotational symmetry constraint* requires the composition features of target images and text queries must be similar to the corresponding image queries by the reversed transition, which also is performed in this complex space. In order to achieve this, we follow the progress of section 2.3 and slightly modify the conjugate by the reversed complex projection. First, we obtain the reversed complex conjugate by the below formulation:

$$\phi^\star = \text{Concat}(\hat{z}cos(\hat{q}^\star), i \cdot \hat{z}sin(\hat{q}^\star)) \tag{3.4}$$

where $\hat{z}$, $\hat{q}^\star$ now represent the query text features and target image features respectively and $i = -1$.

Then the linear projection ρ is also applied to move the composed feature back to image feature dimension.

$$\vartheta^\star = \rho(\phi^\star) \tag{3.5}$$

Analogously as L_{BASE}, L_{SYM} maximizes the cosine similarity kernel between the composed target image query text and corresponding target image features: $cos(\vartheta^\star, z)$ by the soft margin triplet loss:

$$L_{SYM} = \frac{1}{MN} \sum_{i=1}^{N} \sum_{m=1}^{M} \log \{1 + \exp \{\cos(\vartheta_i^\star, \alpha(z_i)) \ - \cos(\vartheta_i^\star, \alpha(z_j))\}\} \tag{3.6}$$

3.4 Total loss L_{TOTAL}

The total loss is weighted sum of above mentioned losses, where the weighted parameters is experimentally set from the original paper.

$$L_{TOTAL} = L_{BASE} + \frac{1}{10} L_{RI} + \frac{1}{10} L_{RT} + \frac{1}{10} L_{SYM} \tag{3.7}$$

4 Experiment

4.1 Dataset

Figure 4: Two images and related captions have the same parent caption. Given the right image as the target image, we use the left image and the right text respectively as the query image and query text.

We implement the task by the real-world Fashion200k dataset [11] which consists of about 200k images of 5 different fashion categories, namely: pants, skirts, dresses, tops and jackets. Each image has a simply human-annotated caption, not a real conversation of clients e.g. "blue knee-length skirt". Since the scope of the project is limited to recommending the t-shirts and shirts for women, so we only employ the t-shirt and shirt image folders and their relevant documents (their annotations and test queries). We collect totally 24,657 images and 7,945 unique captions (i.e. unique clothes) to these images to perform our task.

Through the statistic in the development set, we collect 13,335 images and 4,227 unique captions. In order to let the model know the slight-semantic modification during the training phase, all images whose captions are annotated in such a way using same their particular parent annotations (for instance, "white logo print t-shirt" and "white bear logo t-shirt" in the figure 4 have "white print t-shirt" as the parent annotations). By this format of development set, we stochastically allocate the query text having the same parent annotation with the query image and the target image is query text's ground truth annotation.

For the evaluation task, we obtain 1,200 image-text query pairs to perform the retrieval to the collected 2,646 images, which are annotated test annotation files as ground truth.

4.2 Experimental Setup

Model Pre-training Status and Parameters

For the feature extraction module, we use the ImageNet [12] pretrained data-efficient version of ViT: DeiT [13], integrated with a simple MLP to produce a image model resulting 384-dimensional image feature vector. And we use the pre-trained sentence-transformers: all-MiniLM-L6-v1 [14] as the text model to map a input text sequence to 384-dimensional feature vector for a text query without fine-tuning. More specifically, our ComposeTransformers has 45.7M total parameters, of which the image model has 21.8M parameters; text model has 22.7M parameters; composition module has 0.89M parameters and image-text reconstruction has 0.3M parameters. In the training phase, we freeze 11/12 Transformers Encoder blocks of image model and freeze all parameters of text model. The learnable parameters (including the last block of image model, composition module and image-text reconstruction module) are reduced into 6.7M.

Image pre-processing

Since the ViT model requires fixed size of input images, we uniformly scale the input images to 224×224 without preserving the aspect ratio.

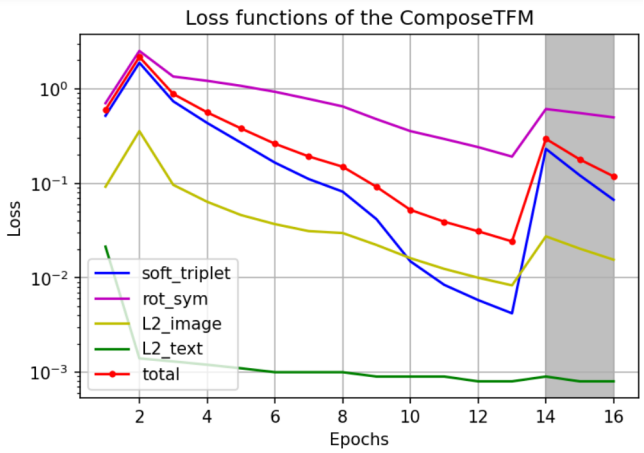

Figure 5: The graph describes the reduction of 4 training loss functions. Where 'soft_triplet' represents L_{BASE}; 'L2_image', 'L2_text' respectively represent L_{RI}, L_{RT}; 'rot_sym' represents L_{SYM} and 'total' represents L_{TOTAL}. Where shaded area is the training epochs with minimum learning rate and reduced batch-size.

Text pre-processing

We simply replace unexpected signs such as "?", ".", "*" and "&" into "questionmark", "dotmark", "starmark" and "andmark" respectively.

Image Data Augmentation

Image model significantly has a important role for the multi-modal retrieval task with the target is image. Therefore training the model with data augmentation is crucial, we implement some slight transformations, which consists of the geometric transformations such as $\pm 5^o$ rotation, $\pm 15^o$ translation, $(0.85, 1.15)$ scaling, ± 0.1 translation in both x and y-axis, horizontal flipping and blurring with 5×5 Gaussian kernel. All transformations have a 0.5 chance of implementation.

Optimization

We implement the training by the Adam optimizer [15], set the learning rate of the image model to half of the rest. In the first epoch, the base learning rate of the model is $5e-4$ and the training batch-size is 32. Because the training progress is difficult and consumes too much hardware resource, so we just can train one epoch for each time and hand-craftily reduce the base learning rate to half for every 3 epochs. For further improving evaluation results on the evaluation set, we set learning rate and batch-size at the end epochs respectively to 5e-6 and 16 from 14^{th} epoch (shaded area in figure 5). Now the losses increase significantly, but in return, the model gains higher performances on evaluations. Detail of evaluation set is depicted in and the results are displayed in the table 1.

Performance results on evaluation set								
Training stt.	Precision				Recall			
	P@1	P@10	P@50	P@100	R@1	R@10	R@50	R@100
Epoch 13	0.0017	0.048	0.26	0.42	0.043	0.215	0.502	0.644
Epoch 16	0.0025	0.055	0.316	0.58	0.049	0.226	0.554	0.757

Table 1: Evaluation results of model in two states: epoch 13^{th} and epoch 16^{th}.

4.3 Results

4.3.1 Evaluation metrics

Precision and Recall are two evaluation metrics that measure how well an information retrieval system retrieves the relevant documents requested by users. The measurements are formulated as below:

$$\text{Precision} = \frac{\text{number of relevant documents retrieved}}{\text{number of retrieved documents of system}} \tag{4.1}$$

$$\text{Recall} = \frac{\text{number of relevant documents retrieved}}{\text{number of actually relevant documents in the database}} \tag{4.2}$$

In this context, for a image search on database of images, Precision is the number of truly relevant images divided by the number of all returned images from the retrieval model. Precision takes all retrieved images into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the model. This measure is called Precision at n or $P@n$. For example, $P@10$ indicates the Precision for top 10 ranking returned results.

Besides, Recall is the number of correct images divided by the number of images that should have been returned. Recall also can evaluate at a given cut-off rank. For instance, $R@10$ indicates the Recall for top 10 ranking returned results.

4.3.2 Evaluation results

The table 1 presented our evaluation results, the model gains better results when continuing to train with reduced batch-size. Compared to the original paper, their model significantly achieves state-of-the-art results when evaluated on the complete validation set of Fashion200k. In detail, their retrieval performance at R@10 is 55.3% and for R@50 they gain 73.4%.

4.3.3 Inference

For the purpose of practical inference, we index all 24,657 images in the dataset into 384-dimensional feature vectors. The inference time (including compute query composition feature vector, compute similarity to all images in the database) costs around 10 seconds on a single computer with Intel Core I5-8300H processor, running at 2.30 MHz using 16 GB of RAM, running Window 10 OS and graphic card GTX 1050. You can implement the inference phase with the source code and pre-trained model found here: https://github.com/hoangtv2000/Clothing-MMRetrieval

5 Conclusion

In this work, we study and indicate the potential of the Transformers'based approach for image-text feature extraction and the method of multi-modal retrieval by the composed representation of these features. Moreover, we are impressed with the performance of an approach that the deep learning model can combine with the idea of complex space projection to merge the feature representation of multi-media information.

References

[1] C. K. M. L.-J. L. L. F.-F. J. H. Nam Vo, Lu Jiang, "Composing text and image for image retrieval - an empirical odyssey." [Online]. Available: https://arxiv.org/abs/1812.07119

[2] M. U. Anwaar, E. Labintcev, and M. Kleinsteuber, "Compositional learning of image-text query for image retrieval," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2021, pp. 1140–1149.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv e-prints*, p. arXiv:1706.03762, Jun. 2017.

[4] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "On the Relationship between Self-Attention and Convolutional Layers," *arXiv e-prints*, p. arXiv:1911.03584, Nov. 2019.

[5] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context," *arXiv e-prints*, p. arXiv:1901.02860, Jan. 2019.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv e-prints*, p. arXiv:2010.11929, Oct. 2020.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv e-prints*, p. arXiv:1706.03762, Jun. 2017.

[8] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 632–642. [Online]. Available: https://aclanthology.org/D15-1075

[9] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. [Online]. Available: https://aclanthology.org/N18-1101

[10] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin, "SoftTriple Loss: Deep Metric Learning Without Triplet Sampling," *arXiv e-prints*, p. arXiv:1909.05235, Sep. 2019.

[11] X. Han, Z. Wu, P. X. Huang, X. Zhang, M. Zhu, Y. Li, Y. Zhao, and L. S. Davis, "Automatic spatially-aware fashion concept discovery," in *ICCV*, 2017.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[13] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers amp; distillation through attention," in *International Conference on Machine Learning*, vol. 139, July 2021, pp. 10 347–10 357.

[14] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[15] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.