Composing Text and Image for Image Retrieval

Alinstein Jose
University of Victoria
alinsteinjose@uvic.ca

Abstract

The project implements a method to retrieve a target image from an input query of an image and a text by learning a single embedding space for the 'text plus image' query and the target images. Input text contained the desired modifcations to the input image. The input image and input text is combined by learning a function $\bar{t} = f_{comb}(i, x)$ to extract the features ($\bar{t}$), where i is the input image and x is the input text. Also, a similarity function $f_{similarty}(t, \bar{t})$ is developed to choose the most similar image from the database to composed image ($\bar{t}$). The project also includes a method to generate the target image from the extracted features from the input using StackGAN. The model is trained on synthetic dataset, CLEVR, which contained 2D and 3D images. The final model achieved a retrieval recall of 73.

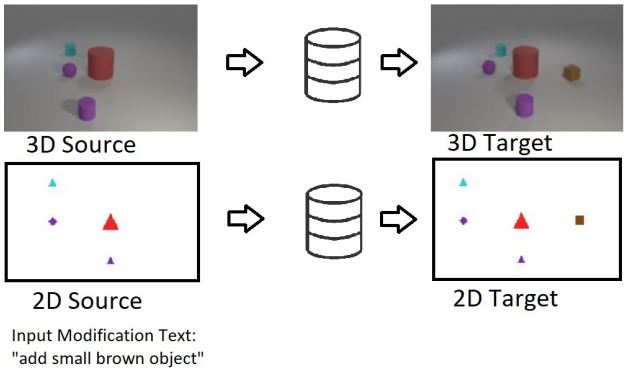

Figure 1. This is a sample of CLEVR dataset, input contains input image and text query which contain the desired modifications and the target image. The dataset is available in both 2D and 3D images.

1. Introduction

The main aim of the project is to retrieve an image a user wants, by giving an image and modification the user wants in the image in the form of text. In this project, the query considered as combination of input image plus input text string describes some desired modification to the image. This represents a typical scenario in session search: users can use an already found image as a reference, and then express the difference in text, with the aim of retrieving the image.[1] This is closely related to attribute-based product retrieval, here we use multi word text as input. The similarity between the search query and the target image is computed using triplet loss. The main contribution of this paper is about learning a function that combining an image and a text, which are of two different input modalities. This implemented by modifying the features of query image by text features while the modified features still leave in the same space as the target image.

The orginal paper is trained on three datasets, CLEVR (CSS), MIT-States and Fashion200k. In this project we have only used the CSS dataset the original paper provides [1], which is based on CLEVR framework[5], since the entire model same for all the datasets except the input pipline.

A sample representation of dataset is shown in the Figure-1. The CSS dataset contains two sets of images, 2D and 3D representation of same image. The 2D image is synthetically generated while training, which represents the 3D image. Code for this project is available at following link :Github link.

My contribution to this project are:

- Minor modification in program and TIRG model and trained and evaluvated the model.
- Implemented StackGAN to generate target image from the features extracted using TIRG model.
- TensorFlow implementation of the TIRG model.
- Visualization of extracted features (embeddings projections) on TensorBoard.
- Researched on methods for transfer domain from 3D to 2D.

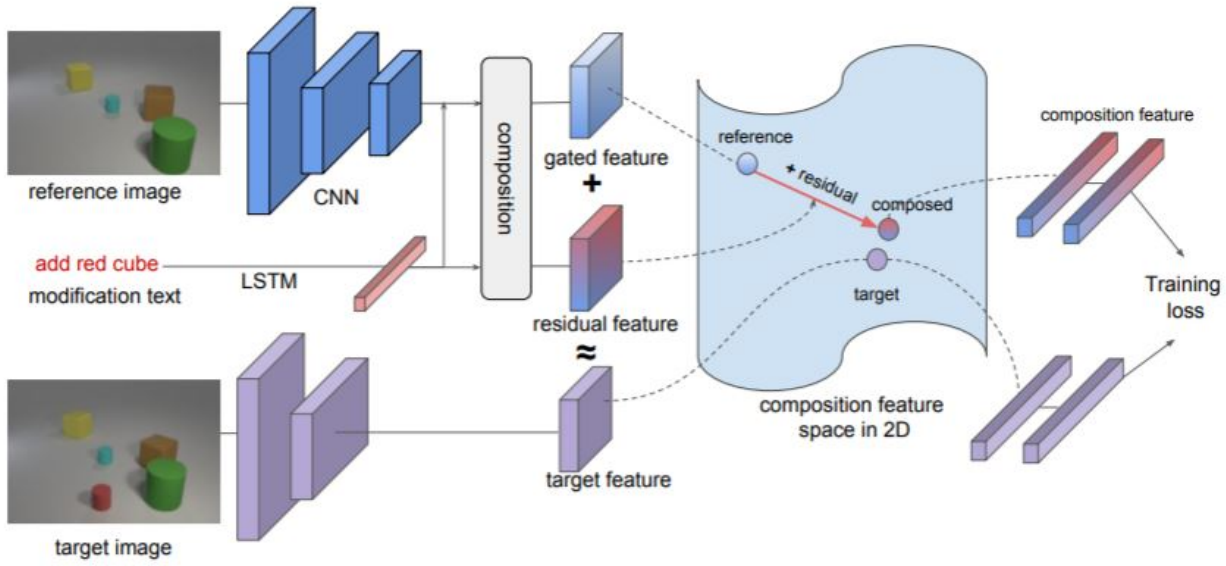

Figure 2. This diagram represents entire pipeline of TIRG model.

2. TIRG Model

The systematic pipeline for the model is shown in the figure 2. Initially, the reference image is passed to ResNet-18 CNN followed by the Global Average Pooling2D layer to extract the image features $f_{img}(x) = \phi_x \epsilon R^{\mathrm{C}}$, where C is the feature channels and C = 512, whereas the orginal paper is implemented to extract 2D feature map without the Global Average Pooling2D layer over the ResNet. Because in the next stage of my project 1D features are needed as input conditions for StackGAN. The text features are extracted using GRU, while orginal paper is implemented on LSTM. I have replaced LSTM with GRU is because most of the input text sequences are significantly small. The GRU use less training parameters and therefore use less memory, execute faster and train faster than LSTM's whereas LSTM is more accurate on the dataset using longer sequence. In short, GRU layer could better choice in this dataset. The text feature extractor function $f_{text}(t) = \phi_t \epsilon R^{\mathrm{d}}$, where d (embedded dimension) is 512. Finally, the extracted features from the text and image are combined using the Text Image Residual Gating function (they refer the function by TIRG in the paper), $f_{combine}(\phi_x, \phi_t) = \phi_{xt} \epsilon R^{\mathrm{d}}$, where d is 512.

The mathematical representation of TIRG model is as follows:

$$\phi_{xt} = f_{combine}(\phi_x, \phi_t) = w_{\mathrm{g}} f_{\mathrm{gate}}(\phi_x, \phi_t) + w_{\mathrm{r}} f_{\mathrm{res}}(\phi_x, \phi_t) \quad (1)$$

where f_{gate} and f_{res} generate an output of dimension R^{512}. The gating and residual part of the model is shown in figure 2. The ratio between gate and residual output is regulated using learnable weights w_g and w_r. The gate connections are computed as follows:

$$f_{\mathrm{gate}}(\phi_x, \phi_t) = \sigma(W_{\mathrm{g2}} * RELU(W_{\mathrm{g1}} * [\phi_x, \phi_t])) \times \phi_x \quad (2)$$

Initially extracted features are concatenated using concat module in Pytorch. The W_{g2} and W_{g1} are linear models to map the output dimensions to R^{512}, σ represents sigmoid function and $\times$ represents elementwise multiplications.

The Residual connections are computed as follows:

$$f_{\mathrm{res}}(\phi_x, \phi_t) = W_{\mathrm{r2}} * RELU(W_{\mathrm{r1}} * [\phi_x, \phi_t]) \quad (3)$$

Initialy extracted features are concated using concat module. The W_{r2} and W_{r1} are linear models to map the output dimension to R^{512}.

The loss of the model is computed using soft triplet loss. To acheive this the triplets are generated each time from the incoming batch. Each triplet conatains a reference feature, a postive feature and a negative feature. The loss pushes the features of composed and target image closer, while pulling apart the features of non-similar images. This loss is implemented as follows:

$$L = \frac{1}{(B)(B-1)} \sum_{i=1}^{B} \sum_{m=1}^{B-1} log(1 + exp(\kappa(\psi_{\mathrm{i}}, \phi_{\mathrm{i,m}^-}) - \kappa(\psi_{\mathrm{i}}, \phi_{\mathrm{i}}^+))) \quad (4)$$

where κ is the similarity Kernal and is implemented as the dot product or l2 distance, B is the batch size,ψ is the target image feature, $\phi^+$is the modified image and $\phi^-$is the negative case.

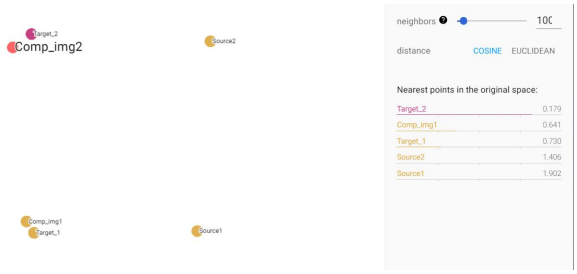

Figure 3. 2D PCA projection of features of images are displayed using tensorboard. Figure also includes the table containing the most similar point composed image 2, using cosine similarity.

2.1. Tensorboard Embedding projector

The features extracted from the images using the TIRG model are visualized using the Tensorboard Projections, displayed in figure 3. The figure contains 2 triplets. 'Source1', 'Comp-img1' and 'Target1' representing source image, composed image, and target image and second set is 'Source2', 'Comp-img2' and 'Target2' . Considering the first triplet, we can see that the 'Comp-img2' is pushed towards 'Target2' from the 'Source2'. This is also verified using the near-point table for 'Comp-img2' shown in the right of figure 3, which shows that the most similar point to 'Comp-img2' is 'Target2'. This part is verify that the model is working as expected. The Code of this implementation can be seen in 'Feature visualize.py'.

2.2. Results

The metric for retrieval is recall at rank K (R@K), computed as the percentage of test queries where the target or correct labeled image is within the top K retrieved images. The most similar images are retrieved using similarity kernel and it is implemented as the dot product between the features of images (query images A ϵ $R^{18000x512}$ and target images B ϵ $R^{19012x512}$ – $A * B^T$). For this experiment, I have calculated for 5 cases of K values. Evaluated testing results for 3D to 3D images and 2D to 2D are shown in table 1 and table 2. The model can achieve a recall (K=1) of 73.28 for 3D to 3D images and 74.24 for 2D to 2D images. This shows that model can achieve approximately same performance for 3D to 3D images and more performance for 2D to 2D case when using GRU layer instead of LSTM, which also increase the training speed and testing speed of the model. The plot of recall (R@1) and loss for 3D images against the epoch is shown in figure 5 and figure 4 respectively. To achieve this performance model was trained for 420 epoches. Moreover, the model trained on the 3D to 3D image can be used for the 2D to 2D image by transfer learning was able to achieve a recall of 60 without any training. However, the model cannot be used for 2D to 3D learning and recall (k=10), was near to 0.1 percent, because the feature space of 3D images and 2D images are different. A few of the drawbacks I found for this model are long testing time since huge matrix mulitplication ($R^{19012x18000}$), also this paper haven't taken consideration when the target image is not in the database. One of the ways to tackle the second problem could be generating target images instead of retrieving the target image from the database.

3D to 3D	Performance
Recall of Top 1	0.7328
Recall of Top 5	0.9361
Recall of Top 10	0.9597
Recall of Top 50	0.9887
Recall of Top 100	0.9945

Table 1. Recall performance for 3D to 3D

2D to 2D	Performance
Recall of Top 1	0.7424
Recall of Top 5	0.9343
Recall of Top 10	0.9604
Recall of Top 50	0.9883
Recall of Top 100	0.9935

Table 2. Recall performance for 2D to 2D

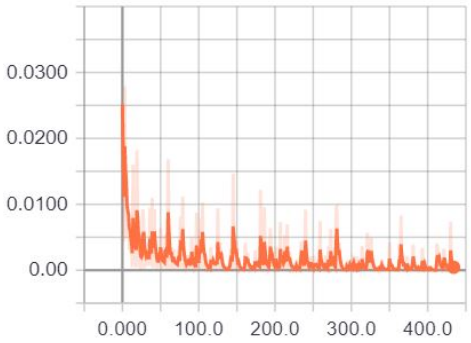

Figure 4. Plot of Loss for TIRG.

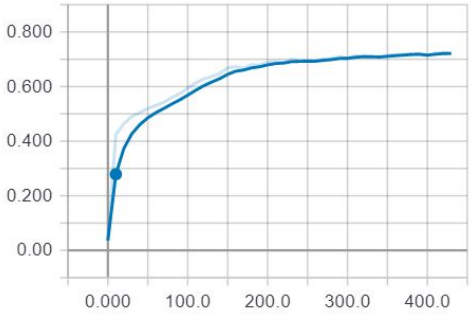

Figure 5. Plot of Recall of Top 1 for TIRG model.

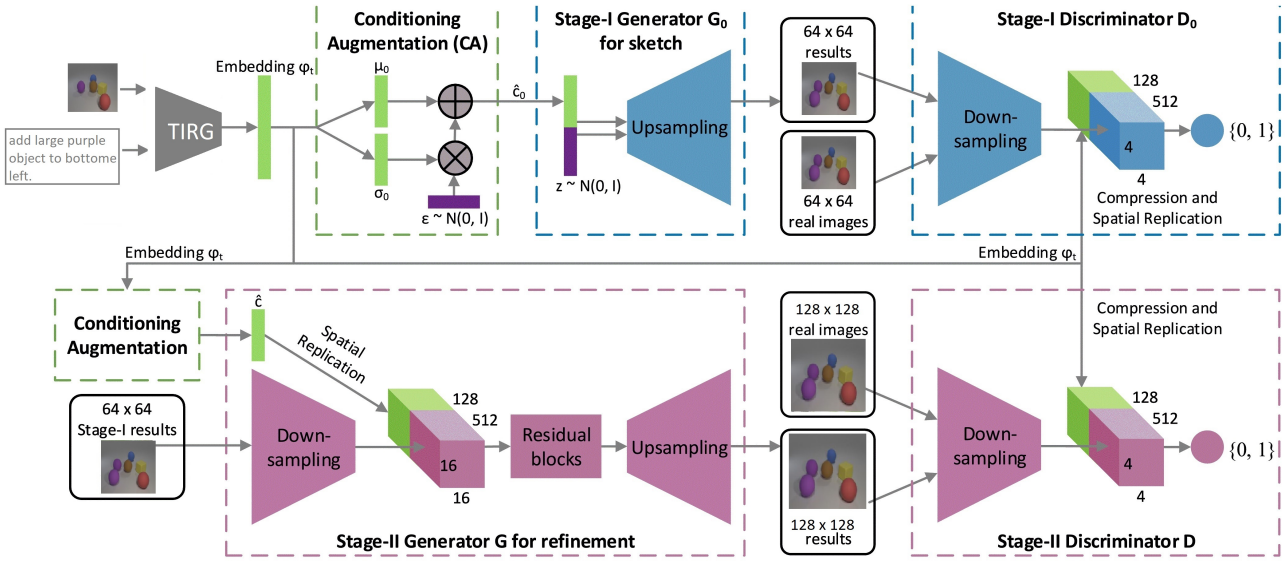

Figure 6. Example of a short caption, which should be centered.

3. StackGAN Model

To overcome the issue of missing target images in the database, I have come up with an idea to generate images using Generative Adversarial Networks. This part of the project is experimental implementation to generate the images using StackGAN [2], from the compositional feature space produced by TIRG model, originally used to generated images from text. To implement this part, a pretrained TIRG model was used to extract the features of the images. These features are passed on to the StackGAN to generate the image. The visual block diagram of the entire pipeline is shown in figure 6. StackGAN is using a similar concept of conditional GAN to generate a high-resolution image by passing the conditions as feature vector as input.

Training of StackGan is a two-stage process, in the first stage the model generates the images of size 64x64, and in the second stage generates an image of 128x128. We have reduced the final output resolution of images to 128 x 128 to reduce the training time and resource requirements. The first stage of StackGan generates an approximate representation and correct colors block in the image and the second stage is for generating more sharp image. To increase the training speed, before training the image features are extracted using the TIRG model, later these features are used for training StackGAN. Initially, We trained the first stage of the model for 125 epochs and used this model to train the second stage while training the stage II, the weights of stage I are kept constant and are not updated.

3.1. Results

Sample images generated by Stage I and Stage II StackGAN are shown in figure 7 and figure 8 respectively. From generated images we can depict the model is able to reconstruct some of the major colors in the image, however, it could not generate the expected (target) image correctly. The samples generated are not satisfying, this could be due to the random image generation behavior of the GAN, less training data, less training data (18K images) and vanishing gradient as the TIRG model is long. These issues could be somewhat reduced by incorporating some of the functionalities like Skip-Connections from the U-Net model.

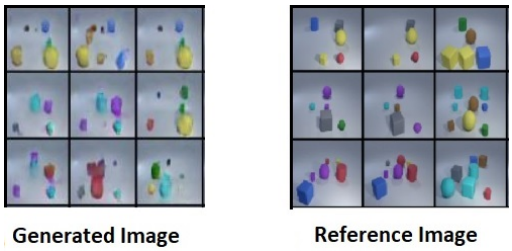

Figure 7. Generated images of Stage I.

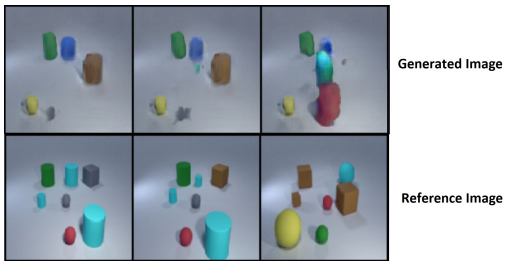

Figure 8. Generated images of Stage II.

4. Other Contributions and Future Scopes

Although, the TIRG model work good with 2D -2D and 3D - 3D image retrieval, the model could not achieve the same performance on 2D - 3D image retrieval task (recall: 0.1), this because the model maps the 2D and 3D images to different feature space. To tackle this problem we have tried a few methods, initially we have used different image encoders for 3D and 2D images which gave use recall of 8.4 percent. Further a few more method were also researched to reduce the distance between 2D and 3D feature space. The first paper that came across was to learn a shared embedding, which introduce a new loss that reduce distance between 2D and 3D features in space and forcing them to be close to each other. $L_{embed}(t) = f_{CE}(e_i, e_j)$, where e_i, e_j are the features from 2D and 3D images and f_{CE} is softmax cross-entropy.

The original pager is implemented in Pytorch, I have ported the code from Pytorch to TensorFlow, which is available in 'TensorFlow_implementation' :Github link: Tensorflow. I have DenseNet121 instead of ResNet18 as the image encoder with an embedded dimension of 1024. The greatest challenge that encountered while implementing was choosing the optimizer and learning rate that minimized the Triplet loss. A few types of optimizer and learning rates were tried and finally chose the RMS optimizer.

5. Conclusion

In this work, we explored the method to compose image and text in the context of image retrieval. An experimental method to generate the target image was also implemented using StackGAN.

6. References

References

[1] Nam Vo, Lu Jiang, Chen Sun, Kevin Murphy, Li-Jia Li, Li Fei-Fei, James Hays, *Composing Text and Image for Image Retrieval - An Empirical Odyssey,2018*. arXiv:1812.07119 , Computer Vision and Pattern Recognition.

[2] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas, *StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, 2017*. Computer Vision and Pattern Recognition, arXiv:1612.03242.

[3] Nam Vo, Lu Jiang, James Hays, *Let's Transfer Transformations of Shared Semantic Representations,2019*. Computer Vision and Pattern Recognition, arXiv:1903.00793.

[4] Leslie Lamport, *Unsupervised Domain Adaptation by Backpropagation, 2015* Machine Learning, arXiv:1409.7495

[5] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, Ross Girshick *CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning, 2016* Computer Vision and Pattern Recognition, arXiv:1612.06890