A Novel Text to Image Conversion Method Using Word2Vec and Generative Adversarial Networks

LIU XINRUI, Inwhee Joe*
Dept. of Computer Software, Hanyang University
*Dept. of Computer Software, Hanyang University
e-mail : yefengcuirenlei@naver.com, *iwjoe@hanyang.ac.kr

Abstract

In this paper, we propose a generative adversarial networks (GAN) based text-to-image generating method. In many natural language processing tasks, which word expressions are determined by their term frequency -inverse document frequency scores. Word2Vec is a type of neural network model that, in the case of an unlabeled corpus, produces a vector that expresses semantics for words in the corpus and an image is generated by GAN training according to the obtained vector. Thanks to the understanding of the word we can generate higher and more realistic images. Our GAN structure is based on deep convolution neural networks and pixel recurrent neural networks. Comparing the generated image with the real image, we get about 88% similarity on the Oxford-102 flowers dataset.

1. Introduction

In this article, we mainly improve the performance of the two parts. First, the method of text extraction is simplified and the vector of semantics is obtained. In the second part, I redesigned the original architecture of generative adversarial networks (GAN). And image generation by upsampling fusion features. For the convolutional neural network, the fully connected layer is removed and the global layer is added to improve the efficiency of the convolutional neural network. Under the condition of ensuring the image generation speed of GAN structure, a new generator is added and the resolution of the image is improved.

2. Skip-Gram Architecture for Word2Vec

For the text embedding extraction method I am using skip-gram [1]. The input to the skip-gram model is a word w_i whose output is the w_{01} to w_{0c} of w_i, and the window size of the context is C. For example, here is a sentence "this flower is pink and white in color, with petals that are multi colored." If we use "flower" as the training input data, the word group {"this", "is", "pink", "and", "white", "in", "color", "with", "petals", "that", "are", "multi", "colored"} is the output. All these words, we will do one-hot coding (see Figure 1).

The ith row of the weight matrix W between the input layer and the hidden layer represents the weight of the ith word in the vocabulary. This weight matrix W is the target we need to learn, because this weight matrix contains the weights of all words in the vocabulary Information. Each output word vector also has an N x V dimensional output w'. The final model also has a hidden layer of N nodes. We can find that the input of the hidden layer node h_i is the weighted sum of the inputs of the input layer. Therefore, since the input vector x is one-hot encoded, only non-zero elements in the vector can produce input to the hidden layer. Thus for the input vector x where $x_k = 1$ and $x_{k'} = 0$, $k \neq k'$. So the output of the hidden layer is only related to the k line of the weight matrix, which is mathematically proved as follows:

$$h = x^T w = w_{k,.} := V_{wI} \qquad (1)$$

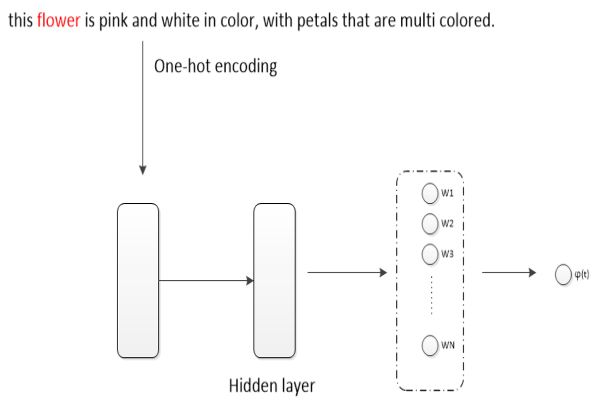
Since the input is one-hot encoded, there is no need to use the activation function here. Similarly, the input of the model output node C×V is also calculated by the weighted summation of the corresponding input nodes:

$$u_{Cj} = V'^{T}_{\omega_j} h \qquad (2)$$

Every word in the output layer is shared weight, so we have $u_{c,j} = u_j$. Finally we generate the polynomial distribution of the number C words by the softmax function:

$$p(w_{C,j} = w_{O,C} | w_I) = y_{C,j} = \frac{\exp(u_{C,j})}{\sum_{j'=1}^{V} \exp(u_{j'})} \qquad (3)$$

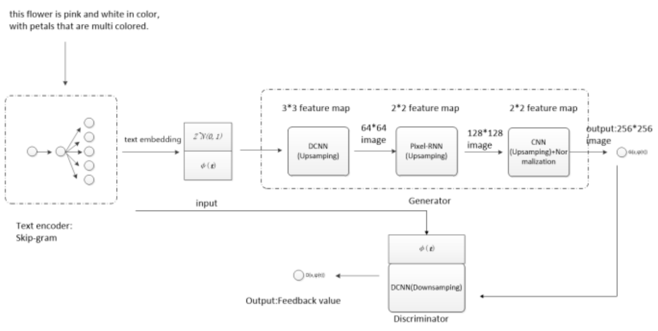
This value is the probability of the jth node of the cth output word.


this flower is pink and white in color, with petals that are multi colored.
One-hot encoding
Hidden layer
W1
W2
W3
WN
φ(t)

(Fig.1) Skip-Gram Architecture

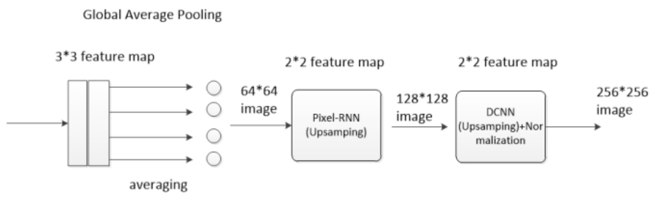
3. GAN architecture

The architecture of the GAN is shown as follows in Table 1 [2]. We use the following notation. The generator network is denoted $G: R^Z \times R^T \rightarrow R^D$, the discriminator as $D: R^D \times R^T \rightarrow \{0.1\}$, where T is the dimension of the text description embedding, D is the dimension of the image, and Z is the dimension of the noise input to G. We illustrate our network architecture in Figure 2.



(Fig.2) GAN Architecture

In the generator G, first we sample from the noise prior $Z \in R^z \sim N\{0.1\}$ and we encode the text query T using text encoder $\varphi(t)$. The description embedding $\varphi(t)$ is first compressed using a fully-connected layer to a small dimension (in practice we used 128) followed by leaky-ReLU and then concatenated to the noise vector Z. Following this, inference proceeds as in a normal convolutional network: we feed-forward it through the generator G; a synthetic image $\hat{x}$ is generated via $\hat{x} \leftarrow G(Z, \varphi(t))$. The training passes through 3 generators. The first G uses a traditional deep convolutional neural network, but we removed the fully connected layer and the global layer is added to improve the efficiency of the CNN, replacing it with global averaging pooling layer [4]. That preliminary image is generated after receiving Z and then enters the second generator. The second main generation structure [5] is pixel recurrent neural networks. The pixel is generated based on the pixel points to increase the resolution of the generated picture. After the image pixel is raised, the third image is entered to generate the final image. As show in Figure 3.



(Fig.3) Generator Architecture

In the discriminator D, we perform several layers of stride-2 convolution with spatial batch normalization [3] followed by leaky ReLU. We again reduce the dimensionality of the description embedding $\varphi(t)$ in a (separate) fully-connected layer followed by rectification. When the spatial dimension of the discriminator is 4×4, we replicate the description embedding spatially and perform a depth concatenation. We then perform a 1×1 convolution followed by rectification and a 4×4 convolution to compute the final score from D. Batch normalization is performed on all convolutional layers.

<Table 1> GAN Architecture

Concatenate	Layer	Output shape
Noise:		z: 512vector
text		$\varphi(t)$: 128vector
Training Image:		64*64*3
G:Up-samping	Conv 3*3	64*64
	Conv 2*2	128*128
	Conv 2*2	256*256
D:Down-samping	Conv 4*4	{0~1}
output		256*256*3

4. Experiments on Text Conversion

In this section we present results on the Oxford-102 dataset of flower images. The Oxford-102 contains 8,189 images of flowers from 102 different categories. There are 7,034 training sets and 1,155 test sets. Table 2 is the architecture design of dataset.

<Table 2> Dataset Architecture

Dataset/Number of images	Train	Test	Total
Flowers	7,034	1,155	8,192

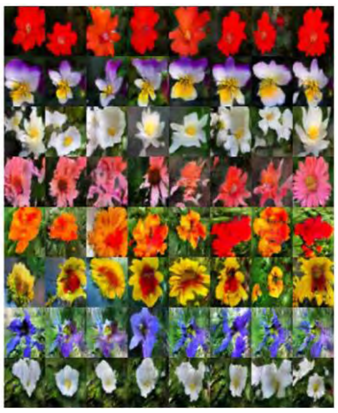
We split these into class-disjoint training and test sets [6] [7]. Oxford-102 has 82 train+val and 20 test classes. We used 5 captions per image. During mini-batch selection for training we randomly pick an image view of the image and one of the captions. As shown Fig.4.

For text features, we first pre-train a skip-gram text encoder on structured joint embedding of text captions with 1,024-dimensional GoogleNet image embedding as described in subsection 2. For Oxford-102 we used a Word2Vec Net with a recurrent neural network.

Category	#ims	Category	#ims	Category	#ims
alpine sea holly	43	buttercup	71	fire lily	40
anthurium	105	californian poppy	102	foxglove	162
artichoke	78	camellia	91	frangipani	166
azalea	96	canna lily	82	fritillary	91
ball moss	46	canterbury bells	40	garden phlox	45
balloon flower	49	cape flower	108	gaura	67

(Fig.4) Oxford-102 Flowers Dataset

Results on Oxford-102 Flowers dataset can be seen in Figure 5 and similarity results between generate images and real images can be seen in Table 3. In this case, the methods can generate plausible flower images that match the description. The basic GAN tends to have the most variety in flower morphology (i.e. one can see very different petal types if this part is left unspecified by the caption), while the methods tend to generate more class-consistent images.



the flower shown has yellow anther red pistil and bright red petals.

this flower has petals that are yellow, white and purple and has dark lines.

the petals on this flower are white with a yellow center.

this flower has a lot of small round pink petals.

this flower is orange in color, and has petals that are ruffled and rounded.

the flower has yellow petals and the center of it is brown.

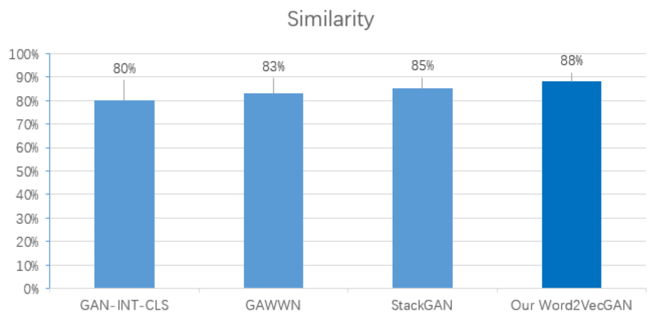
this flower has petals that are blue and white.

these white flowers have petals that start off white in color and end in a white towards the tips.

(Fig.5) Generate Result

Finally, we also carried out an image similarity test. According to the generated image and the real image in the dataset, the similarity is 88%, which is about 8% higher than the similarity generated by other models.

<Table 3> Comparison of Similarity Result



5. Conclusion

In this work we developed a simple and effective model for generating images based on detailed visual descriptions. The Skip-gram+GAN model proposed in this paper has a significant improvement in the text2image generation aspect with the original GAN model. First, a semantic vector is obtained according to the skip-gram model, and then a realistic image conforming to the description is generated by the GAN model. In future work, we aim to further scale up the model to higher resolution images and add more types of text.

References

[1] Tomas Mikolov, Hya Sutskever, Kai Chen, Greg Corrado, Jefferey Dean, Distributed Representations of Words and Phrases and their Compositionality. In NIPS, 2015.

[2] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In ICCV, 2017.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed,Dragomir Anguelov, Dumitru Erhan1, Vincent Vanhoucke1, Andrew Rabinovich. Going Deeper with Convolutions In CVPR, 2015.

[4] Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu. Pixel Recurrent Neural Networks. In CVPR, 2016.

[5] Reed, S., Akata, Z., Lee, H., and Schiele, B. Learning deep representations for fine-grained visual descriptions. In *CVPR*, 2016.

[6] Akata, Z., Reed, S., Walter, D., Lee, H., and Schiele, B. Evaluation of Output Embeddings for Fine-Grained Image Classification. In CVPR, 2015.

[7] Henning Petzka, Asja Fischer, and Denis Lukovnikov. On the regularization of wasser stein GANs. In International Conference on Learning Representations, 2018.