MODELLING AND SYNTHESIZING OF 3D SHAPE WITH STACKED GENERATIVE ADVERSARIAL NETWORK

PRIYADHARSHINI.M[1], SATHYA. J [1], SIVA GANESH. R, SANGEETHAPRIYA. J [2]

[1]Information Technology, Saranathan College of Engineering, Trichy, India
[2]Assistant Professor, Information Technology Saranathan College of Engineering, Trichy, India

---

Abstract - *3D Generative Adversarial Network by using the recent advances from a probabilistic space in convolution networks and generative adversarial nets it generates 3D objects. 3D generated models can provide a good estimate of the eventual outcome of a particular process, and they can show us what is going to get build. We implemented that our method generates high-quality 3D objects, and our unsupervised learned features achieve impressive performance on 3D object recognition, comparable with those of supervised learning methods. The benefits of our model are three-fold instead of traditional heuristic method we use the adversarial criteria which enable the generator to capture object structure implicitly and to synthesize high-quality 3D objects, we can sample objects without a reference image if the generator maps from a low-dimensional probabilistic space to the space of 3D objects, the adversarial discriminator provides a powerful 3D shape descriptor which can learn without supervision. We demonstrate 3D-StackGAN for 3D object generation. We demonstrate that our models are able to generate precise 3D objects.*

Key Words : GAN, StackGAN, Voxel, Inception, Batch Normalization.

1. INTRODUCTION

A generative adversarial network (GAN) is a machine learning model in which two neural networks compete with each other to become more accurate in their predictions. GANs typically run unsupervised and use a cooperative zero-sum game framework to learn[8].The Generative Adversarial Networks (GANs) have shown remarkable success in various tasks, but they face challenges in generating 3D shapes. Stacked Generative Adversarial Networks (StackGANs) focused at generating high-resolution photo-realistic images. Two-stage generative adversarial network architecture is used. StackGAN, for text-to-image synthesis. The Stage-I GAN starts and generates the primitive shape and colors of a scene based on a given text, yielding low-resolution objects. The Stage-II GAN takes Stage-I results and the text as inputs, and generates high-resolution objects with accurate details [5].

Here comes the 3D generative adversarial network,we believe a good generative model should be able to synthesize 3D objects that are both highly varied and realistic [5]. The most important thing is that a 3D objects to have the variations a generative model must be able to go beyond the technique of memorizing and recombining the parts or pieces from the repository that is pre-defined in order to produce a novel shapes. And to generate a realistic object then there must be fine details in the generated example [11].

We demonstrate that modeling volumetric objects in a general-adversarial manner could be a promising solution to generate objects that are both novel and realistic. Our approach combines the merits of both general-adversarial modeling and volumetric convolutional networks [8]. It shows the difference from traditional heuristic criteria, generative-adversarial modeling introduces an adversarial discriminator to classify whether an object is synthesized or real [3].

2. EXISTING SYSTEM

The existing system for the text to image conversion is implemented using the convolution neural network and deep recurrent neural networks [8]. Text classification tasks such as sentiment analysis have been successful with Deep Recurrent Neural Networks that are able to learn discriminative vector representations from text description. Here, what StackGAN do is, generates a 2D high resolution photo realistic images with an input of text description. It includes Stage-I GAN and Stage -II GAN, where Stage-I generates low resolution image and Stage-II yields the high resolution 2D image with the help of text description [5].

Even 3D GAN is also available; here it generates 3D real world objects by giving text input.

Disadvantages : RNNs cannot be stacked into very deep models. RNNs are not able to keep track of long-term dependencies and similarly there is issue of increasing gradients at each step called as exploding gradients [8].

3. PROPOSED SYSTEM

We study the problem of 3D object generation. We propose a framework, namely 3D Stack Generative Adversarial Network, which generates 3D objects from a probabilistic space by leveraging recent advances in volumetric convolution networks and generative adversarial nets.

As usual the generator network is an up- sampling network. It generates a 3D image with a shape that is similar to the input image in terms of its length, breadth, height, and channels by up-sampling a noise vector (a vector from probabilistic latent space). The discriminator network is a down-sampling network. Using a series of 3D convolution operations and a dense layer, it identifies whether the input data provided to it is real or fake.

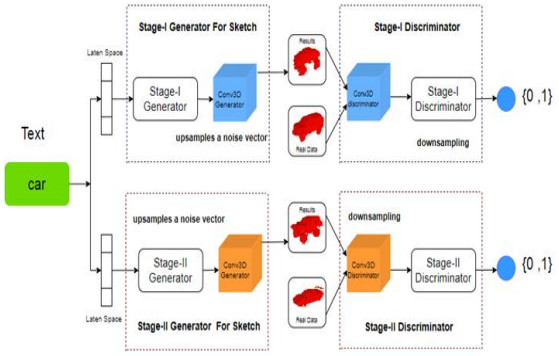


Fig -1: Architecture

Fig-1 Architecture states the complete process of 3D StackGAN. For the generation of 3D object, text is given as input and it is forwarded to stage-1 generator(Inception Generator) .It starts and sketches the basic shape and also it generates the shape of the object. Then again text is given to discriminator, it compares with the real objects says whether it is fake or not .And Stage-2 generates(Batch Normalization Generator) the primitive shape of the object and also it generates the accurate shape of the object by changing the activation function, optimize and learning rate .And Discriminator says whether the generated object is real or fake by comparing with the real objects.

3D Convolution: The input data is applied with 3D filter by the 3D convolution operations along with three directions they are x,y and z. This leads to the creation of a stacked list of 3D feature maps. The output generated shape will be similar to the shape of the cube or the cuboid. The 3D convolution operation is illustrated by the following image. The part which is highlighted in the part of left cube is nothing but an input data. The kernel is present in the middle with a shape of (3, 3, 3). The block which is in right-hand is the output of the convolution operation[3].

Voxel : A voxel is a point in three-dimensional space. A position is defined by voxel with three coordinates in x, y, and z directions. For representing 3D images a voxel is a fundamental unit. The preceding image is a stacked representation of voxels. The gray-colored cuboid represents one voxel. Now you understand what a voxel is, let's load and visualize 3D images in the next section[3].

3.1 INCEPTION GENERATOR

The Inception generator network contains five volumetric, fully convolutional layers with the following configuration:

```
def Inception_generator():

    Inception_gen_filters = [512, 256, 128, 64, 1]
    Inception_gen_kernel_sizes = [4, 4, 4, 4, 4]
    Inception_gen_strides = [1, 2, 2, 2, 2]
    Inception_gen_activations = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
    Inception_gen_convolutional_blocks = 5
    Inception_gen_lr = 0.0025
    Inception_gen_optimizer = Adam(lr=Inception_gen_lr, beta=0.5)
```

Fig -2: Inception Generator Config

In above Fig-2 Inception_gen_filters represents the dimensionality of the output space ,which the dimensionality (512…..64) states that the size of the image is getting precise from the scratch(512) to the end(64) .Above mentioned optimizer and activation functions are the functions which also works to get the possible images.

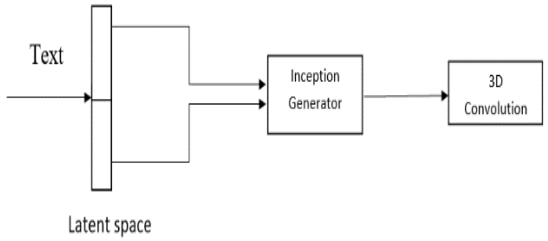


Fig -3: Inception Generator

In Fig-3, In this inception generator, text is forwarded to latent vector which generates the possible shapes in the probabilistic space.

3.2. INCEPTION DISCRIMINATOR

The Inception discriminator network contains five volumetric convolutional layers with the following configuration:

```
def Inception_discriminator():

    Inception_dis_filters = [64, 128, 256, 512, 1]
    Inception_dis_kernel_sizes = [4, 4, 4, 4, 4]
    Inception_dis_strides = [2, 2, 2, 2, 1]
    Inception_dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
    Inception_dis_activations = ['leaky_relu', 'leaky_relu', 'leaky_relu',
                    'leaky_relu', 'sigmoid']
    Inception_dis_convolutional_blocks = 5
    Inception_dis_lr = 10e-5
    Inception_dis_optimizer = Adam(lr=Inception_dis_lr, beta=0.5)
```

Fig -4: Inception Discriminator Config

From Fig-4 ,the discriminator network mostly mirrors the generator network. An important difference is that it uses LeakyReLU instead of ReLU as the activation function by using the above mentioned discriminator channel size. Also, the sigmoid layer at the end of the network is for binary classification and predicts whether the provided image is real or fake. The last layer has no normalization layer, but the other layers use batch normalization to regularize the input.

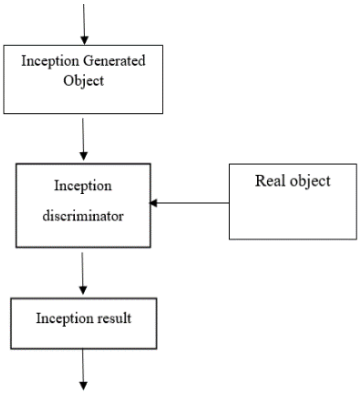


Fig -5: Inception Discriminator

From Fig.5 inception discriminator, generally it compares the generated object with real object and says it is real or fake. These actions are taking place by changing the value of channels.

3.3 BATCH NORMALIZATION GENERATOR

Generator training requires tighter integration between the generator and the discriminator than discriminator training requires the following:

```
def Batch_generator():

    Batch_gen_filters = [512, 256, 128, 64, 1]
    Batch_gen_kernel_sizes = [4, 4, 4, 4, 4]
    Batch_gen_strides = [1, 2, 2, 2, 2]
    Batch_gen_activations = ['Selu', 'Selu', 'Selu', 'Selu', 'hard_sigmoid']
    Batch_gen_convolutional_blocks = 5
    Batch_gen_lr = 0.0001
    Batch_gen_optimizer = Adam(lr=Batch_gen_lr, beta=0.5)
```

Fig -6: Batch_Normalization Generator Config

Unlike in inception generator here it is using adelta optimizer and changing of all activation functions for yielding the possible images.Fig-6 clearly shows that activation functions are changing compared to Inception generator.

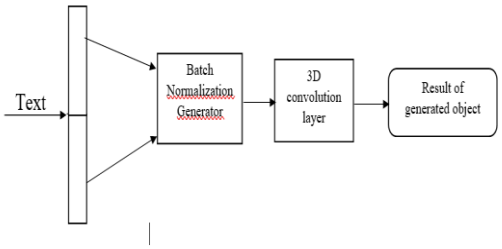


Fig-7: Batch Normalization Generator

From the Fig-7, In batch normalization generator the same text is given and it is forwarded to latent space then the possible images is generated in the probabilistic space.

3.4 BATCH NORMALIZATION DISCRIMINATOR

In Batch Normalization Discriminator, it gets the result from batch normalization generator and compares with the real time object. It only displays whether the object is fake or not by applying five convolution layers.

```
def Batch_discriminator():

    Batch_dis_filters = [64, 128, 256, 512, 1]
    Batch_dis_kernel_sizes = [4, 4, 4, 4, 4]
    Batch_dis_strides = [2, 2, 2, 2, 1]
    Batch_dis_alphas = [0.2, 0.2, 0.2, 0.2, 0.2]
    Batch_dis_activations = ['elu', 'elu', 'elu','elu', 'tanh']
    Batch_dis_convolutional_blocks = 5
    Batch_dis_lr = 10e-5
    Batch_dis_optimizer = Adam(lr=Batch_dis_lr, beta=0.5)
```

Fig -8: Batch_Normalization Discriminator Config

The process is as same as inception generator but the internal functions differs. It also shown in the Fig-8 such as activations,convolution blocks etc..

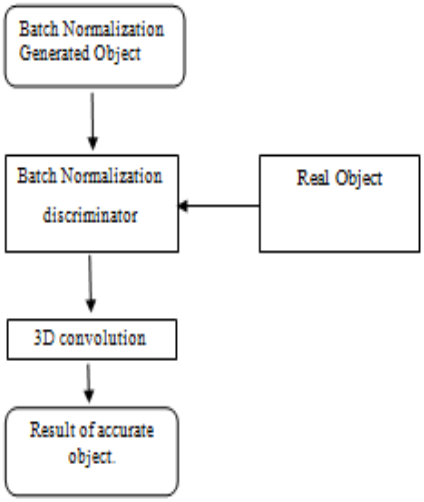


Fig-9: Batch Normalization Discriminator

Fig.9 shows that Batch Normalization discriminator, generally it compares the generated object with real object and says it is real or fake. These actions are taking place by changing the value of channels.

Advantages- Highly plausible 3D objects are generated and More efficient than a normal AI technology. The use of an adversarial criterion, instead of traditional heuristic criteria, enables the generator to capture object structure implicitly and to synthesize high-quality 3D objects.

4. LOSS FUNCTION

Binary Cross-Entropy: The default loss function cross-entropy is used for binary classification problems. It is intended to use with binary classification where the target values are in the set {0, 1}.

Binary Cross-entropy will calculate a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1. The score is minimized and a perfect cross-entropy value is 0[13].

Objective Function: The objective function is the main method for training a 3D-StackGAN. It provides loss values, which are used to calculate gradients and then to update the weight values[4]. The adversarial loss function for a 3D-GAN is as follows:

$$LOSS = \log_2 D(x) + \log_2 (1-D(G(z)))$$

Where,

$Log_2D(x)$- binary cross entropy loss.

$Log_2(1-D(G(z)))$-adversarial loss.

z -latent vector from probabilistic space p(z).

D(x) - output from the discriminator network.

G(z)- output from the generator network.

5. IMPLEMENTATION

For this process, datasets is need to be loaded in the system. In the inception phase, the primitive object is obtained with respective to the text given as input. Changing the 3D Convolution factors such as data format, Activation function, optimizer and changing of learning rate to both generator and discriminator to get more accuracy in images in batch normalization phase. Binary cross-entrophy loss is added in both inception and batch normalization process.

6. EXPERIMENTAL RESULTS

After the continuous training of epochs, the output is obtained it is attached. For every epoch loss is calculated during the training of generator and discriminator as per below figure .

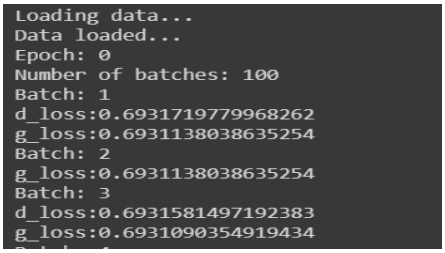
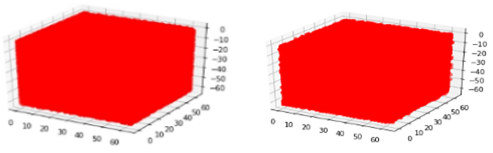


Fig -10 : Training 3D- StackGAN

We used Google collaborative for training the model . In Google collaborative the runtime usage is limited to 12 hours . Therefore we are able to train 510 epochs with respective 100 batch size, after 510 epochs we got only these images which is mentioned below.



Epoch 0 Epoch 510

Fig-11 : Result

Below figure states that ,when we train more epochs we can get the primitive shape of the object .

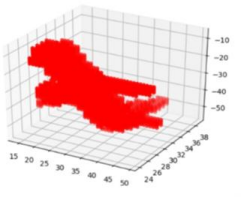


Fig-12 : Generated 3D Object

The finalized output for Inception phase(low resolution) and Batch normalization phase(high resolution) is given below.

1.CAR

LOW RESOLUTION HIGH RESOLUTION

2.TABLE

LOWRESOLUTION HIGH RESOLUTION

Fig-13 : Desired Output

7. CONCLUSION

We proposed 3D-StackGAN for 3D object generation, as well as for learning an image to 3D model mapping. We demonstrated that our models are able to generate novel objects and to reconstruct 3D objects from text. We showed that the discriminator in GAN, learned without supervision and it can be used as an informative feature representation for 3D objects, achieving impressive performance on shape classification.

REFERENCES

[1] Aayush Bansal, Bryan Russell, and Abhinav Gupta, Marr revisited.'2d-3d alignment via surface normal prediction', Conference on Computer Vision and Pattern Recognition, April 2016.

[2] A.Brock, T. Lim, J. M. Ritchie, and N. Weston.'Neural photo editing with introspective adversarial networks', International Conference on Learning Representations, February 2017.

[3] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai, Deeppano.'Deep panoramic representation for 3-d shape recognition'.IEEE Signal Processing Letters,Vol.22,pp.2339–2343, December 2015.

[4] Jiajun Wu,Chengkai Zhang,Tianfan Xue William T. Freeman Joshua, B. Tenenbaum, Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling,Jan 2017.

[5] Han Zhang, Tao Xu, Hongsheng Li, 'StackGAN -text to image photo realistic synthasis using stacked Generative Adversarial Networks,IEEE International Confrence on Computer Vision,Oct 2017.

[6] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. 'Deep generative image models using a laplacian pyramid of adversarial networks', Conference on Neural Information Processing Systems,pp.1486-1494, December 2015.

[7] Ian Goodfellow ,TimSalimans, , Vicki Cheung . 'IMPROVED TECHNIQUES FOR TRAINING GANs', Neural Information Processing system.June 2016.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, SherjilOzair, Aaron Courville, and YoshuaBengio. 'Generative adversarial nets', Conference on Neural Information Processing Systems, Vol.2,pp.2672-2680, December 2014.

[9] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. 'Generative visual manipulation on the natural image manifold', European Conference on Computer Vision, December 2016.

[10] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. 'Generative adversarial text-to-image synthesis', International Conference on Machine Learning,Vol.48,pp.1060-1069, June 2016.

[11] A.Radford, L. Metz, and S. Chintala. 'Unsupervised representation learning with deep convolutional generative adversarial networks'. International Conference on Learning Representations, January 2016.

[12] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and VladlenKoltun. 'Probabilistic reasoning for assembly-based 3d modeling', ACM Transactions on Graphics, Vol.30,pp.35, July 2011.

[13] J. Zhao, M. Mathieu, and Y. LeCun. 'Energy-based generative adversarial network', International Conference on Learning Representations, March 2017.

[14] https://medium.com/vitalify-asia/create-3d-model-from-a-single-2d-image-in-pytorch-917aca00bb07

[15] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv3D
[16] https://www.machinecurve.com/index.php/2019/10/18/a-simple-conv3d-example-with-keras/