GAN-based Photo Video Synthesis

A Project Report

Presented to

Dr. Chris Pollett

Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

CS 297

By

Lei Zhang

Dec, 2019

ABSTRACT

Generative Adversarial Networks (GANs) have shown impressive results of generating synthetic images. However, video generation is still hard to do even for these neural networks. At the time of writing, the best video that GAN can make are a few seconds long, distorted, and low-resolution. As part of CS297 project, I studied how to use various GAN models to generate videos. In particular, I developed a deep convolutional GAN (DCGAN) to generate digits. This model can be extended to generate any images. Convolutional layers were used in all GAN models in this project. Then, I explored how to use GAN with 3D convolutions and pix2pix models to generate short video clips with random noise vectors as input. After that, I tried using an extra Long Short-Term Memory (LSTM) layer to improve the stability of video generation. This model deflated the 3D GAN model to a 2D GAN model, but it can learn all the features that the 3D convolutions GAN models can learn.

Keywords - Generative Adversarial Network (GAN), video generation, temporal layer, 3D convolutions

TABLE OF CONTENTS

I. INTRODUCTION

Video synthesis is an artificial intelligence technology that uses machine learning models to generate videos. In general, it includes future video prediction and unsupervised video generation. Future video prediction requires an input with a sequence of previous frames to predict the next frame. Video generation aims at constructing a plausible new video with learned features which does not depend on a prediction. In this project, I focus mostly on video generation with unsupervised learning. In particular, the model can generate videos with random Gaussian noise vectors as input.

Two-dimensional convolutional neural networks (2D CNNs) have been successfully used in image generation. However, 2D CNNs cannot be used on video generation because of their inability to learn temporal features. Therefore, 3D CNNs were invented to capture temporal features to make convolutions able to learn video features [3]. One of the easiest approaches to unsupervised video generation is Generative Adversarial Networks (GANs) together with 3D CNNs [1][2][3][4][5][6]. This project explored how to use GAN with 3D CNNs to generate videos.

However, GANs with 3D CNNs do not generate good videos because it learns three dimensions at one time. The standard 3D CNNs have a few drawbacks, which include more memory consumption and easy to cause overfitting [12]. People have started to find alternative models to generate videos instead of direct use of 3D CNNs with GAN. Recently, 2D + 1D models have been used to do video generation in paper [2][5]. As the name, 2D + 1D model includes two layers: a 2D neural network layer can learn spatial information in images and a 1D neural network layer learns temporal information among frames. Therefore, 2D + 1D model can learn the same features as 3D CNNs can learn, but with less memory consumption and less chance of causing overfitting. LSTM was used as part of GAN layers in this project to learn

temporal features in videos. However, LSTM is not the only technique to learn temporal features in videos, and more technologies are mentioned in the conclusion of this report.

In the experiments, I used three data sets: UCF101, Weizmann Action database and MPII Cooking Activities. UCF101 is an action recognition data set that includes 101 categories. It consists 13320 320p videos [11], and all the videos have a fixed frame rate of 25 FPS. To the best of my knowledge, this is the largest dataset of human actions. MPII cooking activities dataset [13] includes 224 1624p videos with a static background. Weizmann Action database contains 9 actions and 81 videos [14].

In Deliverable 1, I tried to use GAN to generate Chinese character numbers. A deep convolutional GAN was used to generate synthesis numbers. Deliverable 2 explored how to use 3D convolutions GAN to generate videos. In Deliverable 3, I studied how to use pix2pix to generate videos. Pix2pix can learn a mapping from a source image to a target image, and I used that feature to generate video frames. Furthermore, Deliverable 4 used an LSTM layer to improve the stability of video generation in Deliverable 2.

II. DELIVERABLE 1: DCGAN FOR DIGITS GENERATION

The goal of this deliverable was to get familiar with convolutional layers and GAN. Furthermore, understand how to use the autoencoder method to generate synthetic data. An autoencoder is a type of neural network that can learn to generate an output as close as its input, which is the key to generate images. DCGAN is a popular technology to generate images.

My dataset had ten Chinese character digits and it was expanded using the Keras preprocessing package to 10,000 images. I used rotation, rescale, zoom, and shearing transformations methods to expand the original dataset. I started to train the data with a GAN without convolutional layers, and then I tried to use DCGAN for the same dataset. Apparently, DCGAN generated better images.

The DCGAN in this experiment includes a generator and a discriminator. The original training images have a resolution 82 X 73 which has been reshaped to 28 X 28 to pass into the discriminator. Both batch normalization and dropout were used in the discriminator. Moreover, “leaky ReLU” function was used instead of a normal “ReLU” function in the discriminator as suggested in [9] with leakiness value of 0.2. The final output has 4,096 parameters.

The generator takes a noise vector with 100 dimensions, and then it reshapes them to 7 X 7 X 128. There were three convolutional layers in the generator. Both batch normalization and leaky ReLU were used in the convolutional layers. I used “UpSampling2D” function in Keras to upscale the image to a final output with shape 28 X 28 X 1.

Fig 1. DCGAN training result

Fig 1. shows the result after training the DCGAN with 20,000 epochs. It generated Chinese character images from 0 to 9. Though some strokes may not be very clear, it was easy to identify which number they belong to.

III. DELIVERABLE 2: 3D CNNS GAN FOR VIDEO GENERATION

The objective of this deliverable was to use 3D CNNs to generate videos. I had read [1][3][6], and I found 3D CNNs is a good technology to generate videos. 3D CNNs can learn both spatial and temporal features without introducing complex layers.

I tried to implement the ideas in [6] to generate videos with foreground and background separately. This method did not go well since I cannot generate a decent video at all. Most of the clips that I created were noise. Also, Li et al. [1] suggested it is not necessary to separate foreground and background, so, I wrote a GAN with 3D CNNs and without distinguishing foreground and background. This was my first successful attempt to generate videos.

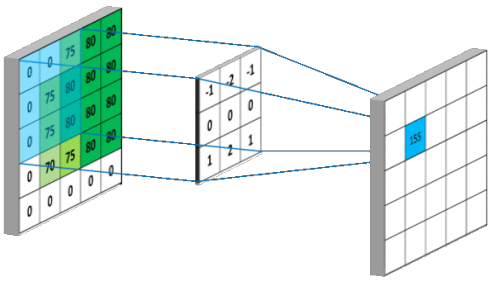

Fig 2. 2D convolutions

2D convolutions use a square kernel filter to learn features in images as Fig 2. indicates. However, 2D convolutions cannot learn time information in videos. Ji et al. [3] proposed a novel 3D CNN model for action recognition. As Fig 3. shows, 3D convolutions use a cube kernel filter to learn both spatial and temporal features in videos.

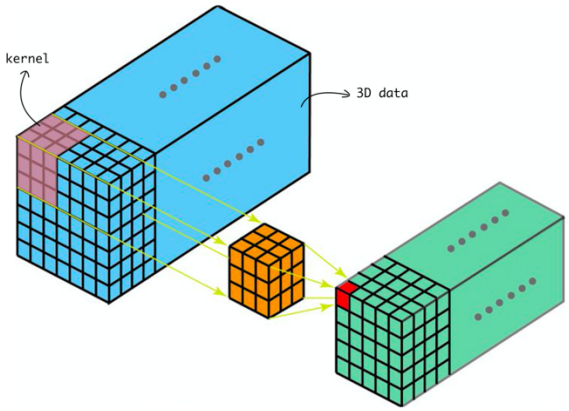

Fig 3. 3D convolutions

The GAN with 3D CNNs model I implemented can learn motions after training only 20 epochs. This model consists of a discriminator and a generator which is similar to DCGAN in Deliverable 1. The discriminator and generator are both 3D CNNs.

In the discriminator, it expects the input shape of video as 8 X 128 X 128 X 3, which 8 is a default setting for number of frames in video clips. I used four 3D convolutional layers with kernel size 4 and stride size 2. Finally, the binary cross-entropy loss function was used.

In the generator, instead of using "UpSampling2D" in DCGAN, I chose to use "Conv3DTranspose" to upscale the video with stride side equals to two. The input was a 100 dimensions noise vector, and I reshaped it to 1 X 8 X 8 X 128 as a start shape for videos. There were four "Conv3DTranspose" layers in the generator along with leaky ReLU functions to upscale both spatial and temporal features in videos. Moreover, there was a full connected 3D convolutional layer to output videos with 8 frames and image resolution 128 X 128 X 3, which 3 was the channel size for color images.

Fig 4. shows the predicted video frames by this model. The model can generate both static and dynamic contents.

Fig 4. 3D CNNs generated frames

IV. DELIVERABLE 3: USE PIX2PIX TO GENERATE VIDEOS

The aim of this deliverable was to learn different ways to generate videos. Pix2pix can generate plausible images due to its excellent ability to learn mappings from one image to another. The hypothesis is to use pix2pix to learn transition between two continuous frames, and then have it predicted the next frame by giving a random start frame from the training dataset. It is a similar idea to have with Hidden Markov Model (HMM), we might use the Independence Assumption. "Each observation O_t only depends on the immediate hidden state S_t." I assume the future frame only depends on its previous past frame. It is easy to expand this theory to use more than one frame to predict a future frame.

Pix2pix is part of image-to-image translation technologies [8]. It maps an image from one domain to another. Pix2pix uses a special type of Conditional GAN to approach that. Indeed, pix2pix is a promising way for learning mapping between two images. In this deliverable, I explored how good pix2pix can predict video frames.

There are four steps to pix2pix video generation. I used a single mp4 video in UCF101 dataset for the training. First, I extracted frames from video A and created paired pictures on which the left and right are two continuous frames. The resolution of paired frames was 256 X 512. Then, I compressed the paired frames to a single npz file which is a file format by Numpy that provides storage of array data using gzip compression. I was using npz compress type to train which can speed up the reading process. After that, I used the pix2pix model to train the npz file and saved the trained model as an h5 file. Lastly, I loaded the trained model to predict the future frames.

I used two ways to predict frames with the pix2pix model. Starting with one original frame A to predict a future frame B, then uses B to predict C. This was repeated. Then I can generate a video clips with plausible predictions. Fig 5. shows the predicted frames from one single original frame.

Fig 5. Pix2pix predicted frames

On the other hand, I used all the original frames to predict the next frames and ignored all the temporal changes. With this method, I generated a video that almost identical to the original training video.

In this deliverable, pix2pix showed excellent results to predict video frames. However, pix2pix cannot predict a longer video with a single start frame as the input. In other words, pix2pix can predict well only if an input image does look like the original training dataset.

V. DELIVERABLE 4: ADD LSTM TO 3D CNNS

This deliverable aims at an improvement of 3D CNNs GAN for video generation. Xie et al. [12] suggested that 3D CNNs are expensive to compute and are prone to overfit. This paper proposed a S3D model which using stacking 2D spatial CNNs and 1D temporal CNNs to speed up the training and to improve the accuracy of video classification. My method is similar, but I used LSTM layers for the generator. The discriminator is still 3D CNNs, but it has five layers instead of four layers compared to the Deliverable 2.

Traditional neural networks cannot learn a sequence of data, which means it cannot remember the previous data though they are maybe relative. Recurrent neural networks (RNNs) try to address this with loops. RNNs make decisions not only on the current data but also on the previous data in time. RNNs have been used successfully in many areas like image captioning and language modeling. However, it cannot learn long-term dependencies because of the vanishing gradient problems [15]. LSTM is a special kind of RNN to address this problem. I used LSTM in this GAN model to upscale video temporal features.

This model includes two inputs: a noise vector and an image. Two inputs were combined in the “build_generator” method and passed to the generator. In the generator, there were 5 convolutional LSTM layers to extend the temporal features of input shape to 16 frames. Then, I used 3D transposed convolutions to upscale the spatial features in the video. Finally, it generated a video with 16 frames and image resolution of 128 X 128.

Fig 6. LSTM 3D CNNs predicted frames

VI. CONCLUSION

GAN with 3D Convolutions can generate decent videos, and it provides a natural temporal dimension to learn transitions among a sequence of frames. However, GAN with 3D Convolutions increases the complexity by bringing in many variations in the additional time dimension [5]. Therefore, it is unstable to train and sometimes it is hard to converge. GAN with 3D CNNs are capable of learning high-level temporal features, but it cannot fill in more details in the experiments.

Many researchers have explored separating temporal and spatial training, some have reached promising results in video generation [2][5]. 2D + 1D network seems a more promising method than GAN with 3D Convolutions, since its simplicity and saves time to train. There are a few techniques that can be used to learn the temporal features of videos too. For example, Gated Recurrent Unit (GRU) and ResNet (Residual Neural Network) are all capable of learning temporal features.

The studies and models were delivered in CS297 would benefit greatly to me in CS298 as fundamental methods for video generation. DCGAN in Deliverable 1 is a fundamental GAN to generate images, and convolutional layers are going to be used in most of GAN models. My CS298 project will build on top of GAN with 3D convolutions that I wrote in CS297, then I will mix it with other models to generate high-resolution and longer videos. Moreover, Pix2pix in Deliverable 3 may be used to upscale frames to high resolutions. Finally, I will use my experiences in CS297 to create a mixed GAN model to generate better videos in CS298. I will compare the produced videos with [5], which is a state-of-the-art video generation method, as declared in the paper at that time.

REFERENCES

[1] Y. Li, D. Roblek, and M. Tagliasacchi, "From here to there: Video inbetweening using direct 3d convolutions," *arXiv preprint arXiv:1905.10240*, 2019.

[2] M. Saito, E. Matsumoto, and S. Saito, "Temporal generative adversarial nets with singular value clipping, " In *ICCV*, 2017.

[3] S. Ji, W. Xu, M. Yang, and K. Yu. "3D convolutional neural networks for human action recognition," *TPAMI*, *35(1):221– 231*, 2013.

[4] S. Aigner and M. Körner. "Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing autoencoder gans," *arXiv preprint arXiv:1810.01325*, 2018.

[5] S. Tulyakov, et al., "Mocogan: Decomposing motion and content for video generation," In *CVPR*, 2018.

[6] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Proc. Adv. Neural Inf. Process. Syst.*, 2016.

[7] A. Clark, J. Donahue, and K. Simonyan, "Efficient video generation on complex datasets," *arXiv preprint arXiv*:*1907.06571*, 2019.

[8] P.Isola, et al., "Image-to-image translation with conditional adversarial networks," In *CVPR*, 2017.

[9] T. Karras, et al., "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[10] C. Finn, I. J. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," In *NIPS*, 2016.

[11] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions calsses from videos in the wild,"Technical Report CRCV-TR-12-01, *UCF Center for Research in Computer Vision*, 2012.

[12] S. Xie, et al., "Rethinking spatiotemporal feature learning for video understanding," *arXiv preprint arXiv:1712.04851*, 2017.

[13] M. Rohrbach, et al., "A Database for Fine Grained Activity Detection of Cooking Activities," In *CVPR*, 2012.

[14] L. Gorelick, et al., "Actions as space-time shapes," In *TPAMI*, 29(12):2247-2253, 2007.

[15] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.