**Generative Adversarial Networks: Video Generation with DCGAN**

*Harshvardhan Singh*

*Department of Computer Science and Engineering,*

*SRM Institute of Science and Technology, Kattankulathur, India, 603203*

*e-mail: hs7805@srmist.edu.in*

*S. S. Saranya*

*Department of Computer Science and Engineering,*

*SRM Institute of Science and Technology, Kattankulathur, India, 603203*

*e-mail: saranyas6@srmist.edu.in*

**Abstract**

Generative Adversarial Networks (GANs) are unsupervised generative models that use deep learning to generate an output that is closely identical to the input dataset. In our research, we have developed a Deep Convolutional Generative Adversarial Network (DCGAN) which generates the frames of a video, and these frames are then converted into video using helper functions in Python such as imageio and glob.

Deep Convolutional Generative Adversarial Networks are different from simple GANs in certain ways, such as in DCGANs there are no pooling layers in neither the strided convolutions, (discriminator), nor the fractional strided convolutions (generator).

A DCGAN has a generator that generates the frames of a video, and a discriminator which is a convolutional neural network (CNN) based binary classifier which determines whether the input data to the discriminator is generated by the discriminator (fake), or from the dataset (real).

In our DCGAN we use batch normalization in both the generator and the discriminator.

We use a ReLU activation in the generator for all the layers except the output which uses the TanH activation function.

And in the discriminator, we use LeakyReLU activation in all layers except for the output where we use a sigmoint so

that the discriminator is able to do binary classification.

**Keywords**

DCGAN, Data Processing, Generator, Discriminator, DCGAN Training

**Introduction**

The Generative Adversarial Network framework is a promising deep neural network framework that has yet to realise its full potential. GAN Networks can generate a range of data types, including images, audio, and three-dimensional objects, and come with a variety of applications.

There are various types of GANs such as Conditional GAN, Least Square GAN(LSGAN), Auxilary Classifier GAN(ACGAN), Dual Video Discriminator GAN, SRGAN, Cycle GAN, etc. Out of these, DCGAN has been the most successful and is widely used for photo generation. In our project, we have used DCGAN for the generation of the frames of a video, which are nothing but a sequence of photos.

In order to generate frames for a video, the DCGAN model is first trained to generate real looking frames using the training dataset. The generator is fed with a random nose, which it converts into an output which closely matches the input from the dataset. The generated frame is then fed to the discriminator, which deciphers if the frame is generated or real. If the discriminator is able to distinguish the generated image, the discriminator output is closer to 0, and the generated image is fed back to the generator in inter for it to generate a frame that matches the input frame more closely, or else if the output from the discriminator is closer to 1, the generated frame matches closely to the input from the training dataset, and the discriminator is deceived to believe that it is a real image.

**Data Preprocessing**

Our dataset consists of training set and test set. The training dataset consists of 60,000 unique frames, and our testing set consists of 10,000 unique frames. For each input from our dataset, the pixel values are ranged from 0 to 255, hence we normalize the pixel value by dividing them by 255. It is done for both the test inputs and train inputs. This is done because our neural network has to process larger weight values if the pixel value is too high, which can lead to slower convergence and poor performance. After normalizing our dataset, we perform the batching and shuffling of the data so that it can be fed to the model in many batches of tensors. We create the training dataset using x\_train from using the tf.data.dataset format, combine the consecutive elements in the dataset into batches and then prefetch the elements from the dataset.

**Generator**

The generator network of our DCGAN model uses Rectified Linear Unit activation. We input noise in the form of a fixed length random vector to the generator for the first iteration.

**Discriminator**

A discriminator is basically a CNN base binary classifier. We pass in the generated frames to our discriminator and the discriminator arrives to a decision, which is a binary decision. The discriminator’s output is a set of real values between 0 and 1, where if the value is close to 0, it is indicative of a fake input but of the value is close to 1 then it is a real input from the training distribution.

**Architecture Diagram**

In our DCGAN model, the generator network first takes an input which is a vector of random noise. Correspondingly, we fetch an input from the training dataset. Every input of random noise we want to corresponding into a sample from the training distribution. Our DCGAN model consists of two neural networks, the generator network and discriminator network. Our generator network tried to fool the discriminator by generating real looking frames, and then the discriminator network tries to distinguish between real and fake frames. Hence, our two neural networks are competing with each other and in the process, the entire DCGAN model is getting trained for generating a more realistic looking frame. The output generated from the generator are called “fakes”, and our discriminator is trained to distinguish between fakes and real frames from the dataset. Once we have a certain number of frames generated, we then use functions in Python such as imageio and glob in order to convert the frames into a video.

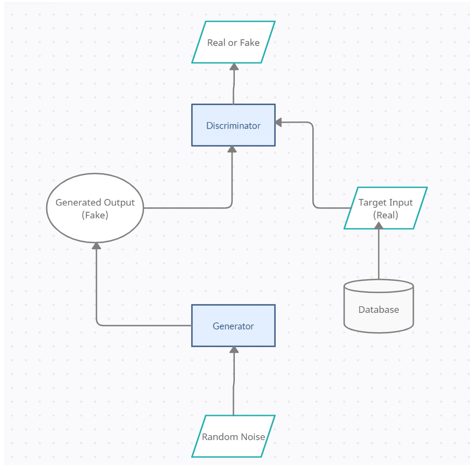


Fig 1. Architecture Diagram

**DCGAN Training**

In order to train our DCGAN, we first train our discriminator and then we moved on to train our generator. In order to train the discriminator, we sample a mini batch of noise samples and then also a sample of real data from our training dataset. We then pass the noise to our generator and get the fake outputs. By doing so, we then have a mini batch of both fake and real data. We do a certain number of iterations to train the discriminator. For the generator to work, we want it to successfully fool the discriminator with high success rate in order to optimize the generated output from the generator. We are also saving the particular network outputs so that we can track how the DCGAN improves after each epoch of training and view the improvement through qualitatively better generated frames.

**Observation**

The generator is observed to generate a much better quality of frames with higher number of iterations or “epochs”, however it took significantly more time to train the model. After 700 epochs, the discriminator was able to deceive the generator with an output of 0.91456093 by the discriminator.

We noted the average discriminative values for 5 frames, and have tabulated them below:

|  |  |  |
| --- | --- | --- |
| S. No. | Epochs | Discriminative value |
| 1. | 50 | 0.43657865 |
| 2. | 100 | 0.55847384 |
| 3. | 150 | 0.64384733 |
| 4. | 200 | 0.71495843 |
| 5. | 250 | 0.79384954 |
| 6. | 300 | 0.83594852 |
| 7. | 350 | 0.89435839 |
| 8. | 400 | 0.91384738 |
| 9. | 450 | 0.93483495 |
| 10. | 500 | 0.93854939 |
| 11. | 550 | 0.94385394 |
| 12. | 600 | 0.94748385 |
| 13. | 650 | 0.94943584 |
| 14. | 700 | 0.95248584 |

Table 1: Discriminative values

Graph 1: Discriminative values for different number of epoch during GAN training

**Results**

We were successful in generating frames that look closely identical to the real frames from the dataset. We noticed that the generator crossed 91% accuracy after completing 400 epochs, and improved by just 4% over next 300 epochs. We were also successful in converting the generated frames (photos) to video using Python libraries imageio and glob.

**References**

[1]P. Korshunov and S. Marcel, “Deepfakes: a New Threat to Face Recognition? Assessment and Detection,” arXiv preprint arXiv:1812.08685, 2018.

[2] “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, Alec Radford, Luke Metz, Soumith Chintala

[3] R. Cellan-Jones, “Deepfake Videos Double in Nine Months,” 2019. [Online]. Available: <https://www.bbc.com/news/technology-49961089>

[4] BBC Bitesize, “Deepfakes: What Are They and Why Would I Make One?” 2019. [Online]. Available: https://www.bbc.co.uk/ bitesize/articles/zfkwcqt

[5] A. Swaminathan, M. Wu and K.J.R. Liu, “Digital Image Forensics via Intrinsic Fingerprints,” IEEE Transactions on Information Forensics and Security, vol. 3, no. 1, pp. 101–117, 2008.

[6] H. Farid, “Image Forgery Detection,” IEEE Signal Processing Magazine, vol. 26, no. 2, pp. 16–25, 2009.

[7] M. Stamm and K. Liu, “Forensic Detection of Image Manipulation Using Statistical Intrinsic Fingerprints,” IEEE Transactions on Information Forensics and Security, vol. 5, no. 3, pp. 492–506, 2010.

[8] A. Rocha, W. Scheirer, T. Boult, and S. Goldenstein, “Vision of the Unseen: Current Trends and Challenges in Digital Image and Video Forensics,” ACM Computing Surveys, vol. 43, no. 4, pp. 1–42, 2011.

[9] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro, “An Overview on Video Forensics,” APSIPA Transactions on Signal and Information Processing, vol. 1, pp. 1–18, 2012.

[10] A. Piva, “An Overview on Image Forensics,” ISRN Signal Processing, vol. 2013, pp. 1–22, 2013.

[11] P. Korus, “Digital Image Integrity - a Survey of Protection and Verification Techniques,” Digital Signal Processing, vol. 71, pp. 1–26, 2017.

[12] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and ¨ M. Nießner, “FaceForensics++: Learning to Detect Manipulated Facial Images,” in Proc. IEEE/CVF International Conference on Computer Vision, 2019.

[13] J. Galbally, S. Marcel, and J. Fierrez, “Biometric Anti-Spoofing Methods: A Survey in Face Recognition,” IEEE Access, vol. 2, pp. 1530– 1552, 2014.

[14] A. Hadid, N. Evans, S. Marcel, and J. Fierrez, “Biometrics Systems Under Spoofing Attack: An Evaluation Methodology and Lessons Learned,” IEEE Signal Processing Magazine, vol. 32, no. 5, pp. 20–30, 2015.

[15] S. Marcel, M. Nixon, J. Fierrez, and N. Evans, Handbook of Biometric Anti-Spoofing (2nd Edition), 2019.

[16] J. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, H. Proenc¸a, and J. Fierrez, “GANprintR: Improved Fakes and Evaluation of the Stateof-the-Art in Face Manipulation Detection,” IEEE Journal of Selected Topics in Signal Processing, 2020.

[17] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. Jain, “On the Detection of Digital Face Manipulation,” in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. [18] C. Canton, L. Davis, E. Delp, P. Flynn, S. McCloskey, L. Leal-Taixe, P. Natsev, and C. Bregler, “Applications of Computer Vision and Pattern Recognition to Media Forensics,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. [Online]. Available: <https://sites.google.com/view/mediaforensics2019>

[19] B. Biggio, P. Korshunov, T. Mensink, G. Patrini, D. Rao, and A. Sadhu, “Synthetic Realities: Deep Learning for Detecting AudioVisual Fakes,” in International Conference on Machine Learning, 2019. [Online]. Available: <https://sites.google.com/view/audiovisualfakes-icml2019/>

[20] L. Verdoliva and P. Bestagini, “Multimedia Forensics,” in ACM Multimedia, 2019. [Online]. Available: https://acmmm.org/tutorials/ #tut3