Video Inpainting using 3D Convolutional Autoencoder

Garrett Sterling Decker, *ASU ID: 1206082081,* gsdecker@asu.edu
Jajati Keshari Routray, *ASU ID: 1211086640,* jroutray@asu.edu

Abstract—A convolutional neural network is trained to generate frames of videos without any unwanted moving objects deteriorating parts of the video. The video dataset used for training the neural network is obtained by creating a synthetic dataset with stationary background and constant field of view. In this project we trained the neural network to recognize moving objects from the array of frames, segment the moving objects from the video and generate video frames after filling the pixels covered by the unwanted moving objects.

Index Terms—Video inpainting, Background estimation, Neural Networks, Deep Learning, 3D Convolution

1 Introduction

A Convolutional Neural Network is designed and trained to generate video background without any unwanted moving objects declining the quality of video. All the pixels of the stationary background are never completely visible all the time in the video. Generating the background frames from a video with constant field of view saves huge amount of data during video compression and transmission. It is also applicable in restoring old video footage shot from tape-based video camera where few frames of the video contain distortions, glitches and noise. Video inpainting also finds applications in video surveillance, stellar imaging and computational photography. The video dataset used for training the neural network is synthetically generated with a constant field of view. A stacked denoising autoencoder using 3D Convolutional/De-convolutional Neural Network layers is trained to learn the moving unwanted object's features without any supervision.
Our main contributions to the project can be summarized as follows.

- Generating the dataset synthetically using Open Source python programming language and Open Source Python library Pygame.
- Utilizing a stacked autoencoder architecture with 3D Convolution, Deconvolution, Maxpooling layers to learn the features from the video and remove unwanted moving objects deteriorating the video.

The remaining content is organized as follows. We provide a brief review of related work in Section 2. We present the procedure used to generate the computer generated dataset in Section 3. The architecture of the proposed network, as well as training, testing details are presented in Section 4. In Section 5, we discuss the Experimental results and analysis.

- *Garrett S. Decker and Jajati K. Routray are with the Ira A. Fulton Schools of Engineering (CIDSE, ECEE), Arizona State University, Tempe, AZ, 85281.*
E-mail: {gsdecker, jroutray}@asu.edu

Project Final Report received May 1, 2017

2 Related Work

Algorithmically, the background from a video sequence with constant field of view is generated by obtaining the median of the pixel value at every pixel from the stream of video frames. Unwanted moving objects does not change the median of pixel value at a particular location significantly. LaBGen method algorithm developed by Droogenbroeck et al. [1] generates background with very low error and uses pixel-wise temporal median filter on patches of background with moving object.

Background initialization models created by Giordano et al. [2] and Chacon-Murgia et al. [3] and other papers presented in conferences use artificial neural networks to generate the background but does not implement Convolutional Neural Networks. Giordano et al. [2] developed background generation method by using weightless neural networks which does not update weights of the neural networks but uses temporary RAM neurons which stores the RGB values of images random-pixel data for the sequence of video frames. During training, the RAM contents of the neural network are incremented (reward) by a positive number up to a maximum value and others are decremented when no change in the pixel value is observed and is passed through a function to classify the pixel and identify patterns. During background modeling, RAM neurons return RGB values at each pixel with most frequent sub pattern and generate the image.

The BE-AAPSA model developed by Chacon-Murgia et al. [3] attempts to determine a background model given a series of frames from a video by computing the average value of pixels classified as foreground in the initial 40 frames of the video. This average is used to classify the video into one of four types, each handled by a separate module. Each module consists of separate unique mechanism to work with the type of video input and uses algorithms such as Fuzzy C-Means and Speeded-Up Robust Features (SURF) feature detector and descriptor. Once a module has been selected, pixels in each frame are divided into three classes,

represented in a matrix $A(x,t)$, where x is location and t is time or frame. This matrix $A(x,t)$ has an effect on the learning rate of the background estimation. The background is estimated by iteratively calculating a weight for each pixel for each frame, $WBM(x,t)$. Future weights per pixel depend on the previous iteration, as well as the learning rate, whose formula depends on which module was chosen, and which class the pixel currently belongs to. Generally, the first few frames have a large effect on the weights, as the learning rate follows an exponential decay formula.

Similar to BE-AAPSA model we aim to utilize the previous frame weights per pixel in the future to remove unwanted objects from the video frames. Hence, we utilize the 3-dimensional convolution layers to gain temporal information about the moving objects from the video frames.

3 Synthetic Dataset Generation:

Videos with extended scenes and static backgrounds with moving objects are ideal candidates for data. Initial attempt to utilize Public domain & free to use stock footage, time-lapse videos and video dataset from ChangeDetection.net, SMBC (Scene Background Modelling Contest), SBI (Scene Background Initialization) proved unsuccessful due to shortage of videos. We then proceeded to create our own synthetic dataset to train the neural network. We obtained images to be used as our dataset background from the Flickr30k images dataset [4]. The images with different sizes from the dataset were resized and cropped to a standard size of 152×152 pixels.

(a) 500×339 (b) 152×152

Fig. 1. Flickr30k dataset sample image with pixel size

Open Source Python programming language and Open Source Python library 'Pygame' was used to generate our dataset. The resized image was set as background and rectangles were drawn on the image. The initial rectangles were created of size randomly between $12-15$ pixels and of random color. The rectangles were moved in the x or y direction randomly at a rate of $3-4$ pixels per frame. To mimic the change in shape of the object, the rectangle width was changed randomly by squeezing or expanding it by 2 pixels. The above procedure was repeated to obtain 8 frames of the image. Due to high memory requirements of a 3D convolution neural network, the video frames were converted from 3 channel RGB as displayed in Fig. 2 to grayscale image. The 8 frames generated to form a sample video are displayed in the Fig.3.

Due to huge memory requirements of the 3D Convolutional Neural Network, the video frames were divided to form minibatches. The grayscale video frames were normalized and mini-batches were formed with each mini-batch containing 1 video. The mini-batches were stored in '.h5' (Open Source HDF5 format) format using Open Source 'h5py' Python library.

4 3D Convolutional Neural Network

We redesigned the 2D Denoising autoencoder neural network to 3D Denoising autoencoder neural network in Open Source Python library Keras. The aforementioned neural network is used to generate the video frames without any moving objects deteriorating the video and we provide the explanation for not adopting Generative Adversial Networks (GANs) to generate the background.

4.1 Why Not GANs?

We initially planned to attempt to generate the background video frames without deteriorating moving objects using Generative Adversial Networks. We decided against using Generative Adversarial Networks to learn and restore the video frames from input videos because the background was a replication from the video.

Consider a timelapse video with objects moving around. The background pixels are not always visible in all the frames of the video but are visible when they are not blocked by moving objects. Our aim is to replicate the background by combining information about the background present at different times in the video. If a GAN tries to predict the missing background starting with some random noise, then it will not be taking advantage of the groundtruth pixel information available in frames when a pixel is not blocked by a moving object. GANs may be useful when the background object is completely blocked throughout the video and we have no way of knowing the background. Using GANs to predict the background objects can be worked on in a future project, but will not be required to replicate the background of the datasets that we used.

4.2 3D Autoencoder Network Architecture:

We studied and explored Denoising Autoencoder Networks [5] to encode and decode noisy images. Autoencoders are composed of a series of convolutional layers followed by a series of deconvolutional layers, with the convolutional and its corresponding deconvolutional layers sharing the same weights. The last convolution layer output between the convolution and deconvolution contains a compressed version of the input. If the input contains noise, the encoder learns which part of the input tends to be the signal, and which part tends to be noise. Then, the encoder only saves the signal part of the input. The decoder can then decode the compressed signal into a denoised version of the input.

We propose using a 3D autoencoder to remove moving objects deteriorating the quality of the video from the videos. The input to our autoencoder would be a series of frames from a video. This would be a 3-dimensional input consisting of two spatial dimensions and one temporal dimension. We theorize that the moving foreground objects will be treated as noise because most of the input frames will not contain the same foreground objects in the

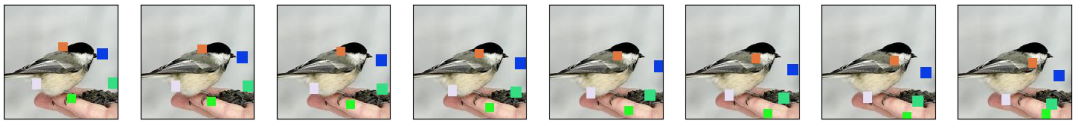

Fig. 2. Generated 8 frames of color video

Fig. 3. Generated 8 frames to form the dataset sample video

Fig. 4. Ground truth video frames

TABLE 1
3D Autoencoder Network Architecture

Type of Layer	Output Shape
Input Layer	(8, 152, 152, 1)
3D Convolution Layer-1 [a]	(8, 152, 152, 5)
3D MaxPooling Layer-1	(4, 76, 76, 5)
3D Convolution Layer-2 [a]	(4, 76, 76, 5)
3D MaxPooling Layer-2	(2, 38, 38, 5)
3D Deconvolution Layer-1 [a]	(2, 38, 38, 5)
3D UpSampling Layer-1	(4, 76, 76, 5)
3D Deconvolution Layer-2 [a]	(4, 76, 76, 5)
3D UpSampling Layer-2	(8, 152, 152, 5)
3D Convolution Layer-3 [b]	(8, 152, 152, 1)

[a] Filter size: 5, Kernel Size = (3, 3, 3)
[b] Filter size: 1, Kernel Size = (3, 3, 3)
Maxpool Size = (2, 2, 2)
Upsampling factor = (2, 2, 2)

same locations over time. The 3-dimensional autoencoder will learn that the intended signal is the static background and the noise is the dynamic foreground rectangles in the temporal domain. To allow the 3D autoencoder to learn proper weights, we provided datasets containing video inputs and the corresponding ground truth background frames as shown in Fig. 4.

The 3D autoencoder network used in our project consists of 3D convolution, 3D deconvolution, 3D maxpooling and 3D upsampling layers. Each convolution or deconvolution (upsampling convolution) layer except the last layer consists of 5 filters with kernel size of (3, 3, 3) and is followed by a Rectified Linear Unit (ReLU) activation. The last 3D convolutional layer consists of only one filter followed by sigmoid activation.

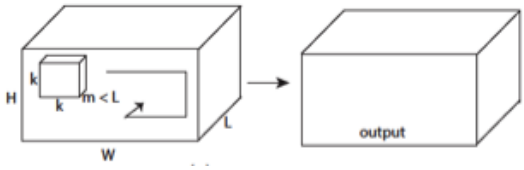

Fig. 5. 3D Convolution [6]

With X as the input, the convolutional and deconvolutional layer outputs are expressed as a function

$$F(X) = max(0, W * X + B) \quad (1)$$

where W, B, $*$ are the filter weights, biases and convolution. It is then passed through the Rectified Linear Unit (ReLU) activation as $G(X) = max(0, X)$. Given a pair of input and its groundtruth videos, the mapping between them is learnt by the neural network in our project by trying to learn the weights and minimizing the Mean Squared Error (MSE) loss function between the provided videos using 'adam' optimizer [7] for 8 epochs.

5 Results

After setting up the architecture and creating synthetic data, we trained the autoencoder for 8 epochs with 490 videos per epoch. See the bottom of the report for following figures. We have plotted the MSE loss for each epoch in Fig. 6, ending with a loss of 0.0094. Figures Fig. 7 and 8 show some sample video frames that we fed into our network during training. On the other hand, figures 11, 13, and 15 show some testing input video frames that were not used during training. Figures 9 through 16 show the corresponding denoised video frames for both the training and testing sample videos.

In these frames, the moving rectangles have been removed by our network, and a slight blurring effect can be discerned. Overall, these frames appear to be fairly

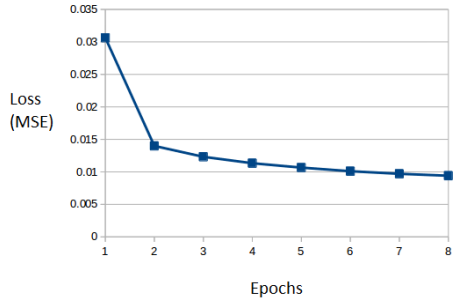

Fig. 6. MSE loss of video frames during training epoch

good depictions of the background from each video sample, meaning that the network has been generalized to work on videos that it has not encountered before.

6 Conclusion and Future work

In this project we explored Generative Adversarial Networks and 3-Dimensional Convolutional Autoencoder Networks. We estimated the feasibility of using them to generate background images from videos. Ultimately, we decided that a 3-Dimensional Convolutional Autoencoder Network was the more promising candidate, and proceeded to construct a network architecture. We also created our own synthetic dataset to train the network on. Additionally, we gained practical experience with the training and tuning of networks, and ended with fairly good results. Determining whether this network can be generalized to work on real world videos is a potential future avenue of research.

References

[1] B. Laugraud, S. Piérard, and M. Van Droogenbroeck, "Labgen: A method based on motion detection for generating the background of a scene," *Pattern Recognition Letters*, 2016.

[2] M. De Gregorio and M. Giordano, "Background modeling by weightless neural networks," in *International Conference on Image Analysis and Processing*. Springer, 2015, pp. 493–501.

[3] G. Ramirez-Alonso, J. A. Ramirez-Quintana, and M. I. Chacon-Murguia, "Temporal weighted learning model for background estimation with an automatic re-initialization stage and adaptive parameters update," *Pattern Recognition Letters*, 2017.

[4] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," *CoRR*, vol. abs/1505.04870, 2015. [Online]. Available: http://arxiv.org/abs/1505.04870

[5] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[6] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3D: generic features for video analysis," *CoRR*, vol. abs/1412.0767, 2014. [Online]. Available: http://arxiv.org/abs/1412.0767

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[8] T. Bouwmans, L. Maddalena, and A. Petrosino, "Scene background initialization: a taxonomy," *Pattern Recognition Letters*, 2017. [Online]. Available: http://sbmi2015.na.icar.cnr.it/SBIdataset.html

[9] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1168–1177, 2008.

[10] M. Braham and M. Van Droogenbroeck, "Deep background subtraction with scene-specific convolutional neural networks," in *Systems, Signals and Image Processing (IWSSIP), 2016 International Conference on*. IEEE, 2016, pp. 1–4. [Online]. Available: https://orbi.ulg.ac.be/bitstream/2268/195180/1/Braham2016Deep.pdf

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[12] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generating images with recurrent adversarial networks," *CoRR*, vol. abs/1602.05110, 2016. [Online]. Available: http://arxiv.org/abs/1602.05110

Fig. 7. Sample training input video frames 1

Fig. 8. Sample training input video frames 2

Fig. 9. Trained video frames output by the autoencoder for sample video frames 1

Fig. 10. Trained video frames output by the autoencoder for sample video frames 2

Fig. 11. Sample testing input video frames 1

Fig. 12. Denoised video frames output by the autoencoder for sample testing video frames 1

Fig. 13. Sample testing input video frames 2

Fig. 14. Denoised image output by the autoencoder for sample testing video frames 2

Fig. 15. Sample testing input video frames 3

Fig. 16. Denoised video frames output by the autoencoder for sample testing image 2