CS415 – Final Project Report – Fall 2020

Name: Dheeraj Kumar Kondaparthi(dkonda2@uic.edu)

Project Title: Synthesizing Motion using Deep Learning

1. Personal Views about Computer Vision and its Applications

Computer Vision is an interdisciplinary scientific field of research that deals with enabling computers to gain high level understanding from images and videos. At high level, it involves tasks and methods to acquire, process, analyze, understand digital images and extract high dimensional data or features in order to provide numerical or symbolic information for better decision making.

Typical functions that are found in many computer vision systems are 1. Image acquisition 2. Pre-processing 3. Feature Extraction 4. Detection/Segmentation 5. High level processing and 6. Decision Making.

Personally, I think the field has evolved very much in recent years due to advent of deep learning and availability of computational power. With state of the art computer vision systems available in the market now, applications of computer vision are everywhere in human day to day life ranging from simple optical character recognition in images to complex systems such as Autonomous driving cars. All these applications have positively impacted human life and the possibilities for this field are endless.

But, broadly applications of Computer Vision are, Automatic inspection in manufacturing plants, Assisting humans in identification tasks such as species identification, Controlling process such as Industrial robots, Detecting events such as people counting or visual surveillance, Modelling objects and environments such as medical image analysis, Navigation such as Autonomous vehicles.

1.1 What I have learned in this course

Through Computer Vision CS415 course in Fall 2020 semester, I have started with learning basics of rule based computer vision via programming assignments on images starting from image filtering using convolution and correlations and then to Canny edge detection on images which has many smart ideas from digital image processing. Then, I have learnt detections tasks such as implementation of Hough Transform algorithm for Line detection and histogram, gaussian based skin detection which gave me understanding of color based segmentation. Then, I have learnt how to do image classification by building local features, image level features, visual dictionary and finally classification and evaluation using K-Nearest Neighbor(KNN) algorithm. Finally, I have learnt to built deep learning based model to classify Fashion MNIST dataset images using Convolution Neural Networks, fully connected neural networks using popular deep learning framework called PyTorch.

Finally, in my course project, I have learnt to implement deep learning based encoder and decoder architecture to synthesize motion from still images. Overall, in this course using course material and assignments, I have started with rule based computer vision tasks implementation to deep learning based computer vision tasks.

1.2 What I would like to study more

I would like to learn more to implement and design machine learning and deep learning based computer vision models especially on Recurrent Neural Networks for video recognition using RNNs, LSTMs for action recognition and text recognition. I would also like learn and implement more models on Generative adversarial networks especially for image enhancements, image restoration and image translation. Finally, I would like to know, how attention is applied to the models for image and video recognition, vision and language and synthesis. These topics are most likely covered in Deep Learning for Computer Vision course.

2. Project Description

2.1 Goal

In this project, we explore effectiveness of deep recurrent networks in the task of understanding motion. Our goal is to implement a modular architecture for inferring realistic motion from still 'seed images' based on recent success of general adversarial networks. Our goal is also to evaluate this mode architecture on various synthetic dataset to show that network is able to learn motion and generate new videos(gifs) from the still images.

2.2 Problem Statement

Producing realistic sequence of images from initial seed image requires understanding 'rules of motion', which is critical. Each frame should follow the frame that precedes it. Objects depicted in frame should move, transform and interact in ways consistent with physical rules that govern objects in motion. Structuring the video generation task to account for the importance of consistency with regards to rules of motion is a challenging task. In this project, we apply deep learning to the problem of producing realistic sequence of images from initial seed image. Specifically, we use Long Short Term Memory(LSTM) network to predict next frame in current sequence. Output of combined network is enhanced by adversarial discriminator trained along side the frame generation network.

3. Introduction

Humans are adept at intuiting the rules of motion from imperfect visual information. We understand that object, which is motion, and should continue to move in same direction unless acted upon by an external force. We also understand that, when two objects moving towards one another on the same plane collide, they should reverse their direction and move apart. Unfortunately, as is frequently in the case of computer vision, what comes naturally to humans is a sophisticated task for computers to understand these rules of motion.

The task of video generation is decomposed into two sub-tasks, first, finding an effective feature encoding of video frames that can easily and effectively reproduce original frames and second generating future frames from the feature encoding of the current frame.

So, to solve this problem, we use recurrent adversarial architecture for frame generation which is trained on two synthetic datasets to produce natural motion.

In section 4, we discuss previous approaches used in literature to solve this problem, that has influenced this work. In section 5, we outline the datasets used for this task including pre-processing steps. In section6, we describe in detail about the architecture and training procedure. In section7, we discuss about the results and finally in section8, we detail the lesson learnt through this project and suggest future directions to improve model results.

4. Related Work

In this section we will explore the literature surrounding the sub-tasks related to video generation,

4.1 Image generation

Architectures like PixelCNN[1], Variational Autoencoders (VAE)[2] and Generative Adversarial Networks (GANs)[3] have proved to be effective in learning latent distribution from new images than can be sampled. In particular, experiments have shown that reconstruction from VAEs using pixel-wise difference loss generally results in blurry outputs [4]. In this regard, generative adversarial networks (GANs) have drastically outperformed other alternatives in producing sharp images. GANs work by having the generative model compete with a binary classifier known as the discriminator; the former tries to fool the latter in thinking its outputs are real. This idea is used in this project by having a discriminator network to produce clear outputs.

4.2 Future Prediction

Recurrent neural networks(RNNs) have been widely used in sequence prediction tasks such as video game sequence prediction[5] and human trajectory estimation[6]. In this project, recurrent network is used to generate future frames from the current frame. In particular Long Short Term Memory recurrent network(LSTM) is used because of its training stability and excellent results.

4.3 Future video prediction

Srivasta et. al[4] have used encoder decoder LSTM to learn representations of video sequences to predict future frames. Even though their results exhibit clear motion, their outputs are blurry. Zhou and Berg[8] have used a combination of convolution neural network, autoencoder and LSTM to generate time lapse videos from still images, but they only use encoding of initial frame to generate all future frames. In this project, we use previous generated frame as input to the next timestep and we also use variational autoencoder for smoother latent space. Further to this model is trained on unsupervised data so that network can learn motion just from the frames.

5. Data

5.1 Datasets

In this project, two datasets are used for the various experiments, first one is Bouncing Balls dataset[9]. This dataset consists of animations of two balls bouncing around a square box. Collisions between the two balls are simulated consistently and momentum is conserved realistically during collisions both between the balls and between balls and the bounding box of the frame.

The second dataset used in this project is synthetic Moving MNIST dataset[4]. This dataset has animations of two randomly chosen MNIST digits moving within and bouncing against the edges of the frame. Here, moving digits do not collide, pass through each other instead. In this dataset, movement is more realistic

than in Bouncing Balls dataset. Both datasets have training size of 80,000 and validation and test size of 10,000 each. These datasets are generated using open source scripts[10][11].

5.2 Pre-processing of Data

To standardize the data, all gifs are down sampled to the size of 64 x 64 size and 20 frames long.

6. Methods

To generate motion from still images, it is important to capture spatial and temporal motion. CNNs are proved to be effective to capture spatial information from the images and recurrent models such as LSTM can help us in dealing with sequential tasks. So, the network architecture used both these ideas to synthesize motion from still images. Models are implemented using PyTorch and trained used Google Colab using tensor processing unit(TPU).

6.1 Model Architecture

Model architecture as shown in Figure1 consists of four modules: an encoder network, a decoder network, an LSTM network and a discriminator network. At a high level, input is encoded into latent vector representation using encoder network. At each time step *t,* LSTM takes encoded representation of last generated frame as well as previous hidden/cell state as input and attempts to generate the latent representation of next frame in the sequence. The decoder network then converts this latent representation back to an image.

The discriminator network is first pitted against encoder and decoder network to improve output of autoencoder. Then the discriminator is also fed the output at each timestep of the LSTM in order to push LSTM to produce output closer to initial data.

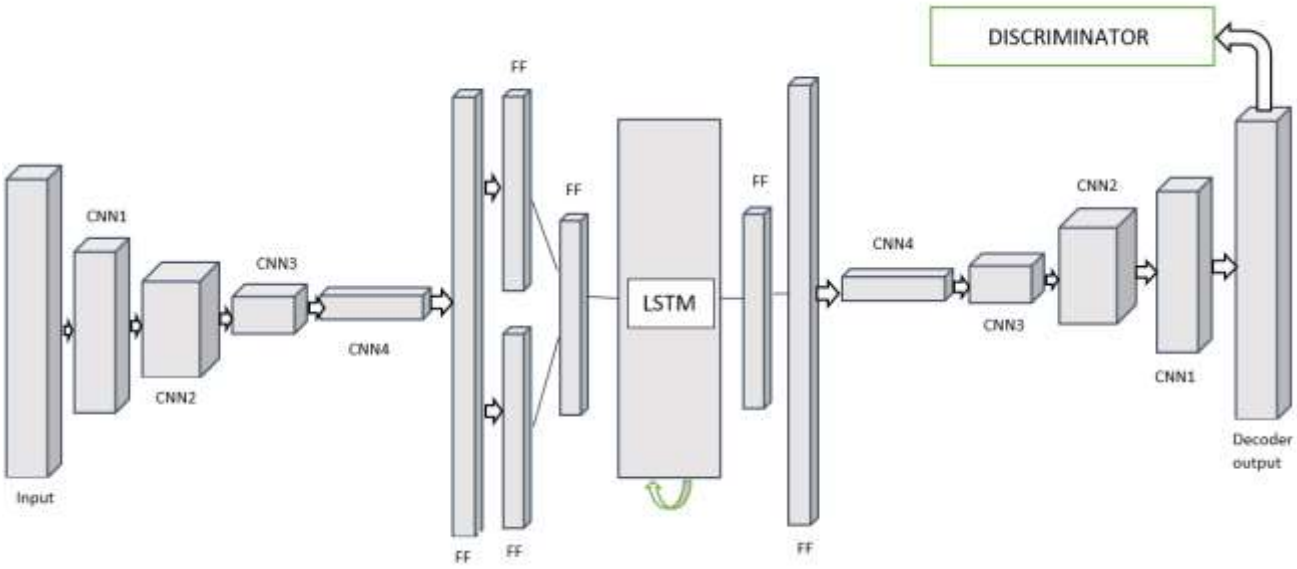


Figure 1: Model Architecture

6.1.1 Encoder – Decoder Network

The encoder takes inputs of size 64 x 64 and applies four convolution layers of the form:

a) 32 filters of size 4 x 4 with stride 2 and padding 1 – CNN1

b) 64 filters of size 4 x 4 with stride 2 and padding 1 – CNN2

c) 128 filters of size 4 x 4 with stride 2 and padding 1 – CNN3

d) 256 filters of size 4 x 4 with stride 2 and padding 1 – CNN4

Each of these layers is followed by bath normalization and a leaky ReLU activation function. The output is flatted and connected via two fully connected layers that each output a vector of size 95 and mean μ and log variance ε respectively to sample from the vector *z*. The *z* sampled normally with mean and standard deviation is given by output of the encoder as per below equation,

$$z \sim N(\mu, \varepsilon)$$

The decoder architecture is exactly opposite of the encoder architecture, with up-sampling of the nearest neighbor followed by convolution, batch normalization and leaky ReLU operations. The final output is given by tanh non-linearity.

6.1.2 LSTM Network

A single layer LSTM is used with hidden cell state of 200. It takes input of size 95(which is same as output of encoder) and gives an output of size 200. This output is passed through fully connected layer to produce a new encoding of size 95 and then to decoder to produce next frame.

6.1.3 Discriminator Network

Discriminator network has architecture similar to encoder network. However flattened output of this is network is of this architecture is passed into a fully connected layer that outputs a vector of size 95, which then undergoes Batch Normalization and a leaky ReLU activation function. Following this, the vector is passed into another fully connected layer that outputs a scalar logit. In section 6.2, we will discuss how this discriminator is trained in detail but in summary, it uses Binary-Cross Entropy loss.

6.2 Objective Function

Progress of network is evaluated by combination of different loss functions to backpropagate the network. First section the loss is reconstruction loss and it is measured by mean squared error (MSE) between the generated frames and ground truth frames of a sequence as per below equation,

$$L_{recon} = |G(x) - x|^2$$

Here $G(x)$ refers to output of the network.

After reading the literature, it is clear that, just by using reconstruction loss results in blurry output[4]. So, from the recent success GANs in producing high quality images, binary cross entropy loss is used to train the discriminator as per below equation,

$$L_{gan} = \log\Big(D\big(G(x)\big)\Big) - \log\left(1 - D(G(x)\right)$$

Further to this, encoder is also penalized if the network is moving away from standard distribution of input images. This requirement is necessary so that smooth transformation of outputs is obtained as we smoothly vary latent variable. This loss is computed using KL divergence between encoder distribution and standard distribution as per below equation,

$$L_{KL} = D_{KL}(q_\theta(z|x)||\mathrm{N}(0,1))$$

Overall loss is given by combination of above loss functions weighted by some constants as per below equation,

$$L = L_{recon} + \lambda_{gan} L_{gan} + \lambda_{KL} L_{KL}$$

In this model, $\lambda_{gan} = 1e-4$ and $\lambda_{KL} = 1e-8$ are used as per literature.

7. Experiments

In this section, experiments performed on two datasets are discussed in detail. Visualizations of gifs, frame by frame outputs are produced in this report by animations of actual and generated gifs can be access by below link because PDF document will not allow us to display gifs

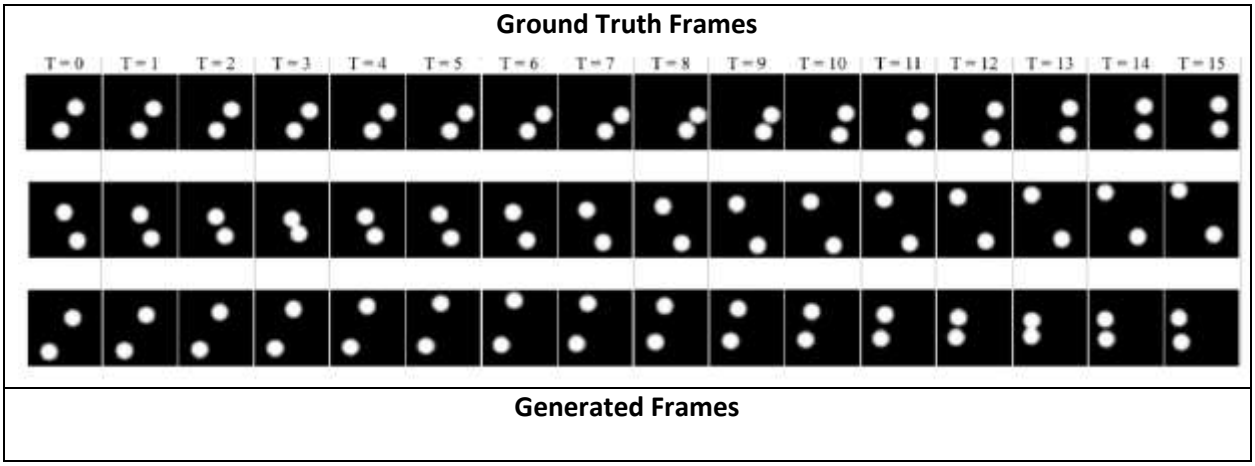
Results link :

https://docs.google.com/presentation/d/1lu4GGeK_4jWEw1hEO1eQBOZMNCqi90kg_DSVvOwfZKk/edit?usp=sharing

7.1 Experiments and Results on Bouncing Balls dataset

First experiments are performed on the Bouncing Balls dataset. This dataset is a good starting point as the Bouncing Ball GIFS are relatively simple, since they have only one color channel and the balls are clearly defined and therefore easy for a network to recognize. Thus, the dataset is best in facilitating the

network in learning rules of motion and collision, without the additional burden of reproducing more complex shapes and colors. While training, Adam update is used with a learning rate of 1e-3 for the VAE-LSTM as well as the discriminator and a batch size of 32. Figure 2 shows the model's frame-by-frame output on the test set. The results are fairly encouraging for several reasons. Firstly, we see that between each pair of consecutive frames, the balls undergo motion. Moreover, local acceleration and deceleration appear realistic, further indicating that the network understands rules of how balls move between frames. In addition, each ball generally retains its shape under translation rather than deforming, meaning that the network recognizes that its primary task is that of translation. Finally, the balls – when they collide with each other (as in Figure 2 second and fourth row) or with the wall (as in Figure 2 third and sixth row), move away from the object the collided with, meaning that the model has learned the general mechanic of collisions. That said, the model's output is not quite perfect. For example, while the local acceleration and deceleration of balls is realistic, it is not globally consistent and so the balls occasionally jitter back and forth in an inconsistent manner (as in Figure 2 third and sixth row). In addition, it is noticed that the network was prone to occasionally producing new balls, which we hypothesized was because of the VAE's learned latent space varying smoothly and may be it is susceptible to small variations.



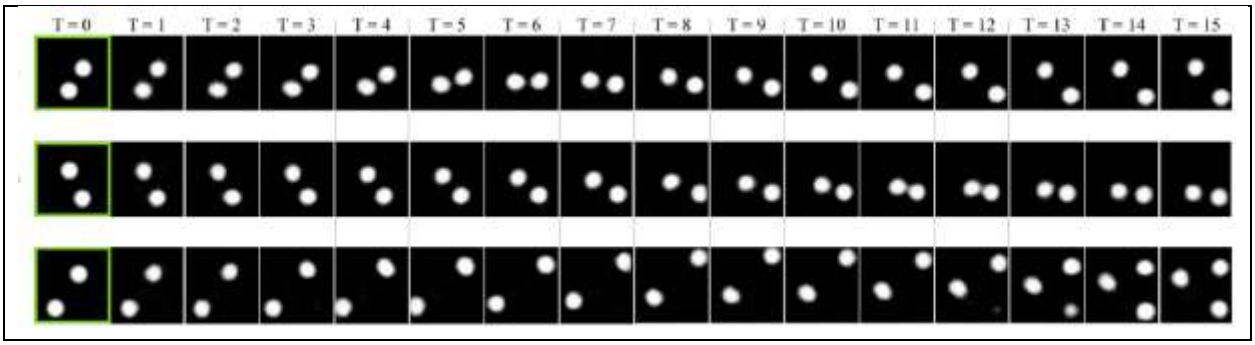


Figure 2: Ground truth and generated frames for Bouncing Balls dataset

7.2 Experiments and Results on Moving MNIST dataset

The Moving MNIST dataset provides several interesting challenges for a video generation model, in addition to those presented by the Bouncing Balls dataset. Firstly, the shapes of these digits are significantly more complex than the balls. In addition, the digits pass through each other instead of moving apart upon collision, and so the model must be able to disentangle the digits and move them separately regardless of their location. As a result, the model must not only learn the physics of natural motion and collision (with walls), but also be able to keep track of each moving entity individually, even if they overlap. Adam update is used to optimize the network. Learning rate of 1e-3 is used for the VAE-LSTM as well as the discriminator and a batch size of 32.

7.2.1 Analysis of smoothness of motion

The model's frame-by-frame output on the test set can be seen in Figure 3. Firstly, we note that the generated outputs demonstrate clear and consistent motion between frames which serves as a promising reinforcement that our network can understand and synthesize notions of movement. Moreover, upon seeing the animated motions between the generated GIFs for the Bouncing Ball dataset and the Moving MNIST dataset (as shown on this page), we noticed the motion in the latter was actually much smoother and more consistent. We believe this might have to do with the digits being asymmetric and thus allowing the network to use the asymmetries as points of reference.

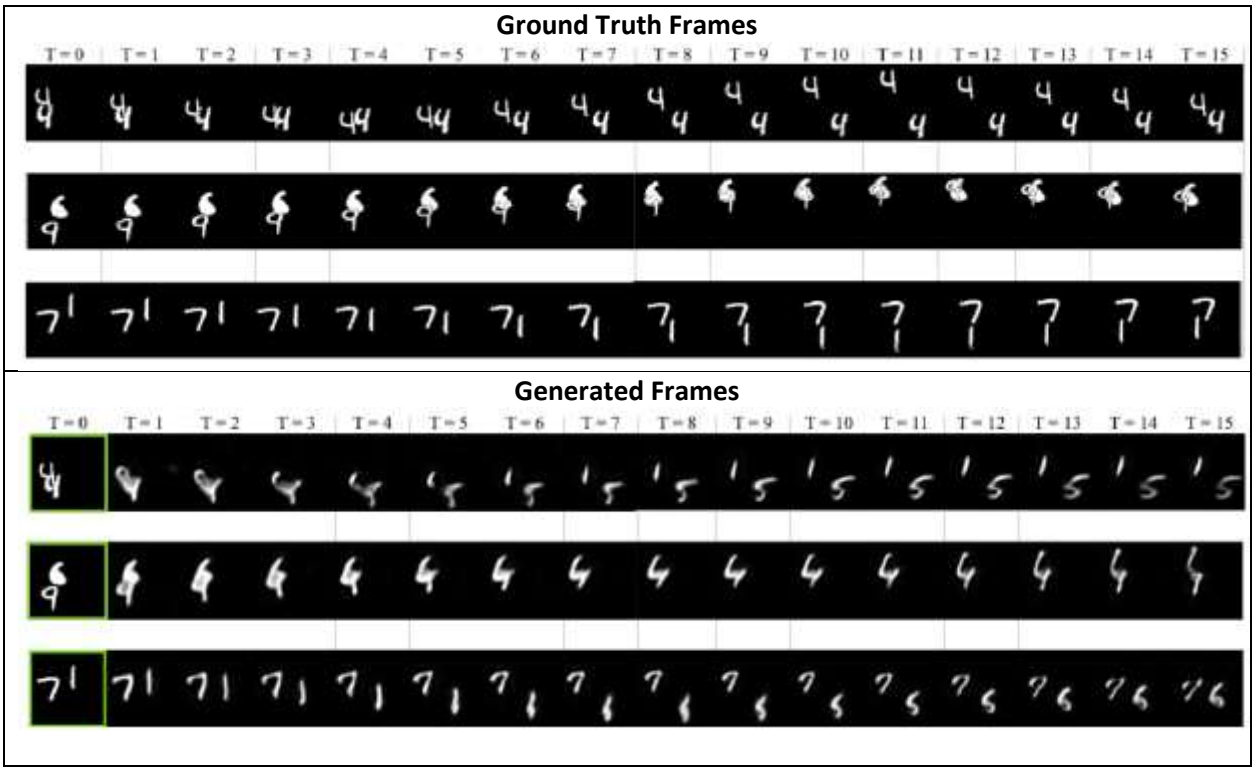


Figure 3: Ground truth and generated frames for Moving MNIST dataset

7.2.2 Analysis on morphing and disentangling digits

We also noticed that the same problem from the Bouncing Ball dataset seems to appear in the generations of this dataset. In particular, the digits morph from their original shape into other digits as they move. Again, we think this is due to the way the latent space is structured. Since the variational encoder ensures the generations transform smoothly, we see the digits morph smoothly from one to another as they move. Another thing the network noticeably struggles with is disentangling the digits when they go through each other.

8. Future Work

While promising, results are still fairly preliminary, and opportunities for future research in this space are enormous. Work in the immediate future should be concentrated on making models which are more robust against the compounding of small errors over longer time frames. This is especially true of the Moving MNIST dataset, for which predicted sequences frequently portray numbers melding together or

fading altogether, blurry and dissipating like smoke. We think that this issue happens because during train-time the model is only evaluated on its ability to generate single-frame predictions from seed frames, because the seed is effectively 'reset' to ground truth after each predicted frame is generated, the model is not penalized for introducing small irregularities that, when compounded over several iterations, lead to significant deviations from ground truth. In the bouncing ball dataset, this issue is visible in the lack of conservation of the momentum of individual balls over long periods. Locally, over small time sequences, ball movement is surprisingly even, displaying smooth acceleration and deceleration. This is critical to maximizing the model's training objective; over longer sequences, however, balls will frequently switch direction, meandering organically and seemingly at random (though of course the direction of movement is in fact determined by small biases in the randomly generated dataset). Modifying the training objective to penalize deviations over longer time sequences would likely improve the continuity of the balls' momentum. Alongside modifying the training objective, several other techniques appear promising in minimizing long-term deviations. Training and evaluating on longer seed frames would likely improve the consistency of the model, especially in the first few generated frames. Similarly, novel architectures that allow conditioning a model's output on a chosen direction or set of directions could improve consistency. Finally, Temporal LSTM models have also been proposed, and seem highly applicable to this problem. Of course, synthetic datasets do not present particularly realistic or sophisticated examples of objects in motion more realistic datasets present a exciting area for future exploration.

9. Conclusion

In this project, we propose an architecture for generating animated GIFs from still image frames. The model combines the most effective techniques from generative networks, future sequence prediction, and adversarial training in a modular way to effectively understand and synthesize motion in images. We

show that our model performs successfully on this task, generating animations of objects that move smoothly and bounce off each other and walls when colliding.

Nevertheless, there are lot of avenues for improvement in our model. Although it is able to synthesize new motion, the smoothness of the motion could be better and moreover, the continuity of this motion over time could be markedly more consistent. We believe this problem mainly occurs from the fact that the network only has a single seed frame from which it arbitrarily picks motion to generate. Training network with longer seed frames or conditioning on a direction would possibly yield more robust results.

Furthermore, the network could perform better in retaining the forms of the objects as the animation progresses. The error that arises from this issue compounds as the generation progresses since the last generated frame is used to generate the next one. To this end, we hope to improve the network by possibly conditioning on the objects present in the initial frame or bolstering our discriminator with a categorical classifier that penalizes the generator for producing digits that are not in the original frame. In general, quantitatively evaluating generative models is a difficult problem and work has been done in using better metrics [12]. We hope to apply some of these ideas to our model in the future.

10. Acknowledgements and topics I hope to learn more

This project is implemented as part of Computer Vision course(CS415), Fall 2020. So, I would like to thank Professor Wei Tang for his excellent lectures and course materials. I would also like to thank all the referenced authors who are generous enough to upload their research to internet without which we would not able to carry out this project.

I hope to learn more about, advanced deep learning architectures in computer vision especially around human pose detection, video generation, topography generation.

11. References

[1] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu.Pixel recurrent neural networks. CoRR, abs/1601.06759, 2016.

[2] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. ArXiv e-prints, Dec. 2013.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. ArXiv e-prints, June 2014.

[4] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. CoRR, abs/1502.04681, 2015.

[5] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 802–810. Curran Associates, Inc., 2015.

[6] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2863–2871. Curran Associates, Inc., 2015.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735–1780, Nov. 1997.

[8] Y. Zhou, Y. Song, and T. L. Berg. Image2gif: Generating cinemagraphs using recurrent deep q-networks. In WACV, 2018.

[9] W. Lotter, G. Kreiman, and D. D. Cox. Unsupervised learning of visual structure using predictive generative networks. CoRR, abs/1511.06380, 2015.

[10]https://github.com/zhegan27/TSBN_code_NIPS2015/blob/master/bouncing_balls/data/data_handler_bouncing_balls.py.

[11] T. Lee. https://gist.github.com/tencia/afb129122a64bde3bd0c.

[12] X. Hou, L. Shen, K. Sun, and G. Qiu. Deep feature consistent variational autoencoder. CoRR, abs/1610.00291, 2016.

[13] Ali Malik, Michael Troute, Brahm Capoor, Deep-GIFs