This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



# Generating Realistic Videos from Keyframes with Concatenated GANs

Shiping Wen, Weiwei Liu, Yin Yang, Tingwen Huang, and Zhigang Zeng

**Abstract**—Given two video frames $X_0$ and $X_{n+1}$, we aim to generate a series of intermediate frames $Y_1, Y_2, \cdots, Y_n$, such that the resulting video consisting of frames $X_0$, $Y_1$-$Y_n$, $X_{n+1}$ appears realistic to a human watcher. Such video generation has numerous important applications, including video compression, movie production, slow-motion filming, video surveillance, and forensic analysis. Yet, video generation is highly challenging due to the vast search space of possible frames. Previous methods, mostly based on video prediction and/or video interpolation, tend to generate poor-quality videos with severe motion blur. This paper proposes a novel, end-to-end approach to video generation using generative adversarial networks (GANs). In particular, our design involves two concatenated GANs, one capturing motions and the other generating frame details. The loss function is also carefully engineered to include adversarial loss, gradient difference (for motion learning) and normalized product correlation loss (for frame details). Experiments using three video datasets, namely Google Robotic Push, KTH human actions, and UCF101, demonstrate that the proposed solution generates high-quality, realistic and sharp videos, whereas all previous solutions output noisy and blurry results.

**Index Terms**—video generation, GANs, concatenated GANs

## 1 Introduction

We study the problem of automatic video frame generation from two given key frames $X_0$ and $X_{n+1}$, which are the first and last frames of the video clip to be generated, respectively. The goal is to generate $n$ frames in between $X_0$ and $X_{n+1}$, so that the resulting video clip correctly captures subject/object motions, presents sharp details of the scene, and appears realistic to the human eye. Such a technique has numerous real-world applications. For instance, it can be applied to increase the frame rate of a given video, which can then be converted into a slow-motion one. Meanwhile, video frame generation can be utilized in movie/animation production, which could supplement existing model-based computer-generated imagery (CGI) techniques. Further, video frame generation could also be applied to alter an existing video, which might have far-reaching effects on security and forensics. For instance, currently surveillance videos and dashboard camera videos are commonly used as evidence in legal investigations. The credibility of such evidence might be compromised, however, if these videos can be falsified through video frame generation, e.g., an intruder might hack into a surveillance system, and modify parts of the video footage that record certain actions.

Deep convolutional neural networks (ConvNets) [38] have been successfully applied in various video analytics tasks, such as video classification [39], and activity detection [40]. Recently, there have been several attempts to apply ConvNets to video generation, among which a common approach is to employ *Generative Adversarial Networks* (*GANs*) [10]. In a nutshell, a GAN contains both a generator network $G$ and a discriminator network $D$, which are trained simultaneously. $D$ aims to distinguish real inputs (i.e., video frames in our contexts) from generated ones produced by $G$; meanwhile $G$ aims to generate realistic video frames that can fool $D$. GANs have been shown to be effective in generating static images, such as scenes [27] and human faces [25], as well as increasing the resolution of given images [22]. For video frame generation, however, existing approaches are insufficient for our problem, yielding rather poor results according to our experiments, due to inherent challenges in the problem as well as deficiencies of previous methods, explained below.

First, there is a vast search space for possible video frames, with no explicit constraints; yet, the actual result space is rather restrictive, i.e., a sequence of frames that smoothly and realistically connect the given two key frames. In the video analytics literature, in order to reduce the search space, video interpolation is often narrowly defined as generating a single frame, given two input frames immediately before and after it in the video (e.g., [41], [42]). Theoretically, one could apply this approach iteratively to generate intermediate frames, e.g., to generate 9 frames $Y_1$-$Y_9$ in between two given input key frames $X_0$ and $X_{10}$, we could first interpolate $X_0$ and $X_{10}$ to produce $Y_5$ (the frame in the middle of the sequence), and then continues to interpolate $X_0$ and $Y_5$ to generate $Y_2$, and so on. The problem with this approach is that it often fails to correctly capture the *motions* in the video, which can be nonlinear (e.g., a person waving hands in the KTH dataset [43]) and complex (e.g., several objects interacting in the Google Robotic Push dataset [5]).

Second, most existing video generation techniques are simply not designed for our purpose, i.e., generating video between two given key frames $X_0$ and $X_{n+1}$. Specifically, several methods (e.g., [4], [7], [8]) aim to predict future frames given past ones. In other words, they *extrapolate* a given video rather than interpolate key frames. Not surprisingly, the extrapolated frames from the

- *Shiping Wen, Weiwei Liu, and Zhigang Zeng are with School of Automation, Huazhong University of Science and Technology, and also with the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan 430074, China, E-mail: {wenshiping226;liuvv;zgzeng}@hust.edu.cn. Yin Yang is with College of Science and Engineering, Hamad Bin Khalifa University, E-mail: yyang@hbku.edu.qa. Tingwen Huang is with Science Program, Texas A & M University at Qatar, E-mail: tingwen.huang@qatar.tamu.edu*
- *This work was supported by the Natural Science Foundation of China under Grants 61673187 and 61673188. This publication was made possible by NPRP grants: NPRP 9-466-1-103 from Qatar National Research Fund. The statements made herein are solely the responsibility of the author.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



input first frame $X_0$ often do not match the given last frame $X_{n+1}$. Other solutions generate video clips from random noise, e.g., [1], [6]. The main motivation to have a random vector as input is to generate a variety of results, e.g., in GAN-based static image generation, different outputs (e.g., face images) can be produced with different input vectors. Result variety, however, is not a major consideration in our setting, since the intermediate frames connecting two key frames are usually predictable from a human perspective, especially when the given key frames are close in time, e.g., within a second. In fact, the entropy in the random inputs often leads to noisy outputs as we found in our experiments, which hurts performance.

Motivated by this, we propose a novel solution for video frame generation from input key frames, using a pair of concatenated GANs. Figure 1 illustrates the architecture of the proposed ConvNet, which contains two generator networks, $G_1$ and $G_2$, as well as two discriminator networks, $D_1$ and $D_2$. The outputs of $G_1$ feed to $G_2$ as inputs. In particular, $G_1$ is mainly responsible for learning motions from training videos, whereas $G_2$ enhances the outputs of $G_1$ by adding finer details. In our design, $G_2$ follows the U-Net architecture [18], which is originally used in image segmentation. U-net architecture, which can keep the structure information of inputs, consists of a contracting path which can transfer the detailed context information of inputs to outputs. Thus U-net can help the second generator to keep the motion information learned by the first generator. $D_1$ and $D_2$ provide adversarial training for $G_1$ and $G_2$, respectively, and following standard ConvNet pipelines. The resulting video frames are extracted from the outputs of $G_2$.

Besides a novel network architecture, another major contribution of this work is the loss function when training the network. The main idea is to view the ConvNet as both a GAN and an *autoencoder* [3]. In particular, the proposed loss function incorporates both adversarial loss (i.e, the effectiveness of $D_1$ and $D_2$ at distinguishing real and generated frames) and autoencoding loss, which includes pixel-level and gradient-level differences between the generator outputs and training inputs. Extensive experiments, using two real datasets KTH [43], Google Robotic Push [5] and UCF101 [52], demonstrate that the proposed solution generates smooth and clear video clips, whereas previous approaches lead to rather blurry ones. Meanwhile, both the concatenated GAN structure and all components of the loss function are vital in obtaining sharp, realisitic output videos.

The rest of the paper is organized as follows. Section 2 reviews related work on GANs and video generation. Section 3 describes the proposed solution. Section 4 contains an extensive set of experiments that evaluate the proposed method both qualitatively and quantitatively. Section 5 concludes the paper with directions for future work.

## 2 Related Work

**GAN.** Generative adversarial network (GAN) [10] is a popular approach to generating new data that follow similar distribution as those in the training sets. For many data generation tasks (e.g., face image generation [25] and visual manipulation [45]), GANs have obtained significantly improved results compared to previous approaches, such as variational autoencoders [19] and deep belief networks [20]. Specifically, a GAN contains two networks, a generator $G$ and a discriminator $D$, which are trained together (e.g., alternatively [10]). The two networks are adversaries to each other: $D$ aims to tell real inputs from generated ones from $G$, whereas the goal of $G$ is to maximize the error of $D$. The goal of training a GAN is to converge to a Nash Equilibrium [44], in which neither $G$ nor $D$ can gain by unilaterally changing its internal parameters. Intuitively, at convergence $G$ should be able to generate realistic datasets, at least from $D$'s point of view. Straightforward GAN models often suffer from problems such as model collapse [26]. Hence, various improvements, e.g., DCGAN [24], WGAN [26], WGAN-GP [27], BEGAN [25], have been proposed to improve the convergence and performance of GANs. Conditional-GAN [28] and info-GAN [29] add extra condition information to the network, in order to further customize the generative results. The proposed solution is based on the GAN approach.

**Image generation using GANs.** GANs have been successfully applied to generating samples of photo-realistic images [15]. Such images can be created from a short text description [21], or from a simple sketch [22]. GANs have also been utilized to generate high-resolution images from lower-resolution ones [14], [22]. Additional applications of GANs include image translation [13], [16] and automatic image editing [23]. According to [46], Facebook has been using GANs in production to generate realistic-looking images. In these applications, it is common to build a GAN following an autoencoder architecture (e.g., in [3], [13]); meanwhile, in contrast to the original GAN [10], which takes random noise as input, in these applications the inputs are often real images in their respective problem domain (e.g., image translation in [13] and super-resolution in [22]). The proposed solution is inspired by these designs.

Besides 2D images, GANs have also been used to generate 3D models, e.g., from 2D inputs [33]. However, these solutions in general are not suitable for generating video frames, which are a sequence of images with temporal semantics, e.g., subject/object motions. Hence, GANs for video generation (including this work) often follow very different architectures from those for generating static images, explained below.

**Video generation.** Since a video has both spatial and temporal aspects, it is difficult to handle videos using a simple ConvNet [30], which is known to be good at capturing spatial features but often not temporal ones. A popular approach to video analytics is to combine ConvNet with recurrent neural network (RNN) [32] and its variants such as LSTM [31], which are known to capture temporal features well. In [17], the authors combine the RNN with restricted Boltzmann machine (RBM) [47] to predict bouncing ball motions. Ref. [5] combines ConvNet and LSTM to predict object movements from pixels in previous frames, and generate future video frames accordingly. As explained in Section 1, predicting future frames (i.e., video extrapolation) is a different problem than ours (interpolation); thus, the method in [5] does not directly apply to our setting.

Regarding GAN-based video generation models, Ref. [7] uses an auto-encoding GAN to predict future frames for time-lapse videos. In [4], future frames are predicted using gradient loss to enhance the quality of results. Domain knowledge on the human skeleton is exploited to help generate skeleton motion frames in [2]. The proposed method is inspired in part by these approaches; nevertheless, predicting future frames is an orthogonal problem as described above. Ref. [1] generates video frames from random noise, using two stream networks and a GAN. These results demonstrate that it is possible to generate realistic video frames with a convolutional GAN, without using RNN. However,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



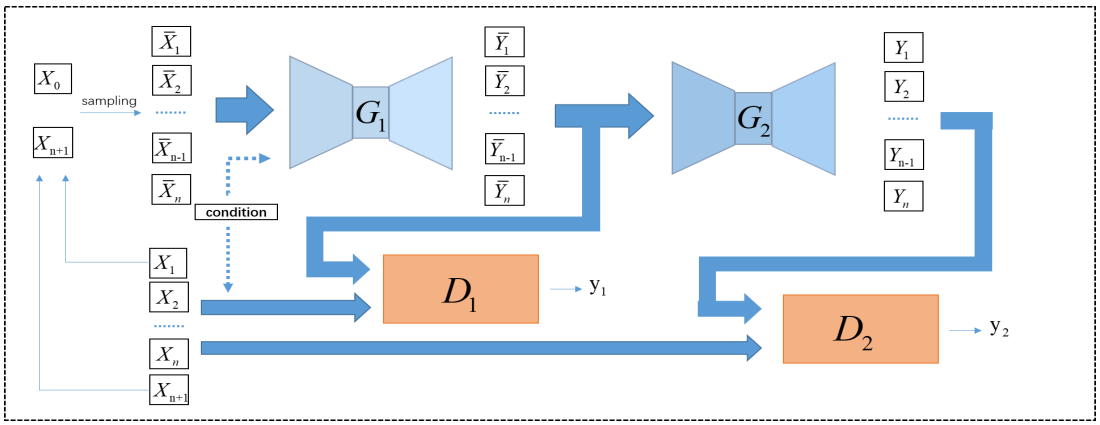


Fig. 1. Illustration of the proposed concatenated-GAN architecture: $X_0$ to $X_{n+1}$ denote the real video frames in the training set. $\bar{X}_1$ to $\bar{X}_n$ represent interpolated frames from $X_0$ and $X_{n+1}$ obtained using a simple deterministic method. $D_1, G_1$ (resp. $D_2, G_2$) are the generator and discriminator of the first (resp. second) GAN, respectively. $\bar{Y}$ and $Y$ denote the outputs of $G_1$ and $G_2$, whereas $y_1$ and $y_2$ are the outputs of $D_1$ and $D_2$, respectively.

as explained in Section 1, randomness in the output frames is not a major objective in our problem setting; rather, randomized inputs hurt performance by introducing noise in the output frames. Inspired by reinforcement learning, some researchers predict the future frames in Atair games based on conditioned action [3]. The objects of the game can be controlled by the actions directly. They predict futures frames using strong supervised learning way which is different with [4], [50] and our proposed solution. To effectively predict the future frames using unsupervised learning way, authors in [50] proposed to decompose videos motion and content to generating dynamics. They also use encoder-to-decoder architecture and convolutional LSTM as the basic model for frame pixel prediction. And their results on UCF101 dataset [52] and KTH dataset [43] are better than others. However the method can not generate longer sequence frames and accurate motion. Future frames are predicted by using hierarchical methods which model the motion structure from inputs to predict the motions evolved in the future [51]. They also built their model using convolution LSTM and encoder to decoder network. However, neither method can predict a long sequence of video frames.

## 3 Proposed Solution

Section 3.1 overviews the network architecture of the proposed method for automatic video generation. Section 3.2 focuses on adversarial training aspects of our solution. Section 3.3 presents the loss function. Section 3.4 provides important details in implementation and training.

### 3.1 Network Architecture

Recall from Section 1 that in our problem setting, the input consists of two frames $X_0$ and $X_{n+1}$, and the outputs are frames $Y_1, \cdots, Y_n$ such that the video with $n+2$ frames $X_0, Y_1, \cdots, Y_n, X_{n+1}$ appears natural to the human eyes. To do this, we train a ConvNet-GAN using a dataset containing similar videos as the ones to be generated during test time. The neural network consists of 4 components, two generators $G_1$, $G_2$ and two discriminators $D_1$, $D_2$, as illustrated in Figure 1. The discriminators follow a standard ConvNet pipeline, with convolution, pooling, and fully-connected layers. Each discriminator outputs a single Boolean value signifying whether or not it believes the input is from the real dataset. Details of our specific implementation of the discriminators are provided later in Section 3.4.

Next we focus on the generators. Figure 2 exhibits the concatenated generators architecture. Following common practice in GAN design, each generator is a fully convolutional network, containing exclusively convolution layers. There is no pooling layer in the network; instead, we control feature map sizes using strides in the convolution layers. Further, we use only 2D convolutions: there is no expensive 3D convolutions or recurrent layers in our network. Meanwhile, each generator can be viewed as an encoder (containing down-sampling convolution layers) followed by a decoder (up-sampling deconvolutions). The outputs of each generator are of the same shape as the inputs, i.e., $n+2$ video frames with the same resolution as the input frames. Note that our network design assumes the number of frames $n$ to be generated is known in advance. This assumption holds in many applications, for example, in video compression and slow-motion video generation. Generating videos with a varying number of frames is left as future work.

The inputs of generator $G_1$ include (i) $X_0$ and $X_{n+1}$, which are the given start and end frames of the video to be generated, respectively, and (ii) $\bar{X}_1, \bar{X}_2, \cdots, \bar{X}_n$, which are derived from $X_0$ and $X_{n+1}$, and do not depend on the real frames $X_1, X_2, \cdots, X_n$. Frames $\bar{X}_1$-$\bar{X}_n$ are created using a simple, deterministic interpolation method. However, the linear interpolation is a simple way to expand the number of inputs. In our implementation, we obtain them with pixel-wise linear interpolation of $X_0$ and $X_{n+1}$. Since we use only 2D convolutions, each frame is treated simply as three color channels (red, green and blue). Hence, in total the input to a generator has $3 \times (n+2)$ channels. Note that in our design, there is no randomness in the inputs; hence, the outputs are deterministic given the input key frames $X_0$ and $X_{n+1}$ and network parameters.

Generator $G_1$ outputs frames $\bar{Y}_1, \cdots, \bar{Y}_n$, each with the same resolution as the input key frames $X_0$ and $X_{n+1}$. During training, these frames are compared to the real frames from the input video $X_1, \cdots, X_n$ for autoencoder training (detailed in Section 3.3. Meanwhile, $\bar{Y}_1$-$\bar{Y}_n$ are also fed to discriminator $D_1$ as inputs for adversarial training (explained in Section 3.2).

Generator $G_2$ takes as input the outputs of $G_1$, i.e., frames $\bar{Y}_1$-$\bar{Y}_n$. It outputs frames $Y_1, Y_2, \cdots, Y_n$, with the same resolution as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



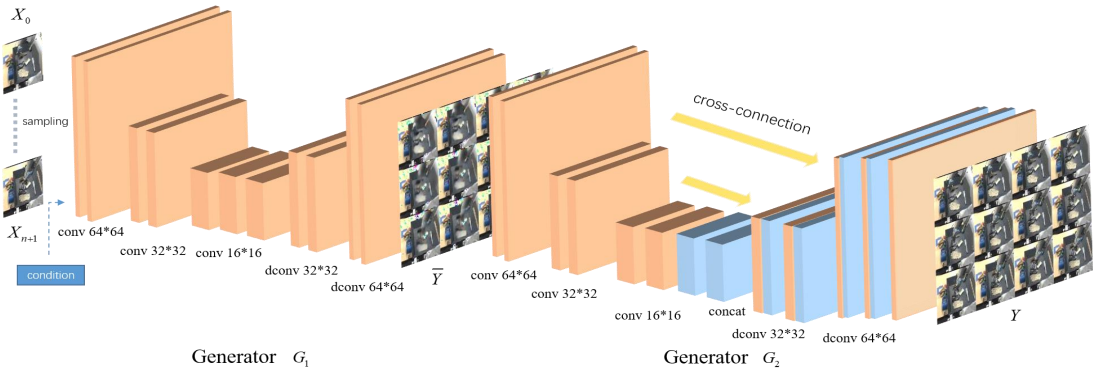


Fig. 2. Illustration of concatenated generators architecture. Generator $G_1$ is shown on the left, whose output feeds into generator $G_2$ on the right. Yellow rectangles represent feature maps with different sizes. $conv$ and $dconv$ indicate the convolution and transpose convolution layers, respectively, and the numbers (e.g., 64*64) are the size of features. There is no pooling layer in the network; instead, we control the feature map sizes through strides in the convolution layers. The blue rectangle on the left represents motion information, and we set it as a action-constrain to guide the network to learn the motion information. Generator $G_2$ is based on the U-net architecture, where the orange arrows are the cross connections. We concatenate different feature maps as new features.

the input frames $X_0$ and $X_{n+1}$. Similar to $G_1$, to train $G_2$, $Y_1$-$Y_n$ are compared to true frames $X_1$-$X_n$, and fed to discriminator $D_2$ for adversarial training. The main difference between $G_1$ and $G_2$ is that the latter follows the U-Net architecture [18], with cross-connections between layers of the same frame resolution, e.g., the first layer is fed to the last layer as input, as shown in Figure 2. The U-Net is known to be able to add fine details to coarse-grained results, e.g., in image segmentation tasks. The U-net architecture is known to be used in segmentation tasks and pix-to-pix tasks because it consists of a contracting path which can transfer the detailed context information of inputs to outputs. In our experiment, U-net can help the second generator to keep the detailed information such as the motion information learned by the first generator. By the way, the motion distribution does not exist in inputs frames of first generator, so we design the first generator using the auto-encoder architecture to learning the motion distribution without U-net. We use the U-Net for exactly this purpose, i.e., refining the output frames with details.

**Discussion.** In our network architecture we made several design choices that are important for performance. First and most importantly, we use two GANs instead of one. Intuitively, the first GAN learns motions using a normal autoencoder pipeline, whereas the second learns frame details with U-Net. If we were to use one GAN only, we would have to choose between frame details (for which a U-Net helps significantly) and motions (for which U-Net hurts performance). This is confirmed in our experiments, as results obtained with two GANs are noticeably better than those from a single GAN. Second, we use only 2D convolutions, without recurrent layers or 3D convolutions, which are computationally expensive. We empirically found that for generating short video sequences (e.g., less than a second), 2D convolutions are sufficient; for very long sequences, the results are rather poor either way, due to the difficulty to learn and generate long and complex motions. Lastly, the generators in the GANs are also trained as autoencoders, which is detailed in the following subsections.

### 3.2 Adversarial Training

A GAN is essentially a zero-sum game between a pair of generator/discriminator neural networks, where the discriminator aims to minimize classification error of real and generated inputs, whereas the generator aims to maximize the discriminator's error. Therefore, we include in the generator's loss function the following term for adversarial loss:

$$L_{\text{adv}} = \mathrm{E}_{X\sim P_{data}}(\log(D(X)) + \mathrm{E}_{\bar{X}\sim P_{data}}(\log(1-D(G(\bar{X}))). \quad (1)$$

In our neural network architecture, there are two pairs of generators/discriminators, namely $G_1$/$D_1$ and $G_2$/$D_2$. Each pair of networks are trained with the above loss independently. Observe that neither generators $G_1$ or $G_2$ has access to the real frames $X_1, \cdots, X_n$ in the training set from their respective inputs. Instead, information on these real frames are back propagated through the adversarial losses given by their respective discriminators. The generators also obtain information on the real frames through auto-encoder training, elaborated in the next subsection. Adversarial training complements auto-encoder training, because the latter focuses on pixel- and gradient- level matches between the real and generated frames, where as adversarial training additionally corrects the generated frames in higher level features extracted by the discriminators' deep neural networks.

It is worth mentioning that a naive implementation of GAN training process is not guaranteed to converge to a Nash Equilibrium, and the generator may collapse by always outputting the same frames regardless of the inputs. There are various training techniques to mitigate these issues, e.g., [48]. This is an orthogonal topic and we omit further details for brevity.

$$L = \lambda_1 * L_{\text{adv}} + \lambda_2 * L_{mse} + \lambda_3 * L_{gdl} + \lambda_4 * L_{npcl}. \quad (2)$$

where $\lambda_1$-$\lambda_4$ are hyperparameters, whose values are given in the next subsection.

### 3.3 Joint Loss Function

As explained earlier, the generators $G_1$ and $G_2$ in the proposed solution also undergo autoencoder training, in which we compare their outputs with real frames in the training set. In our design, we compare real and generated frames in three different aspects.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



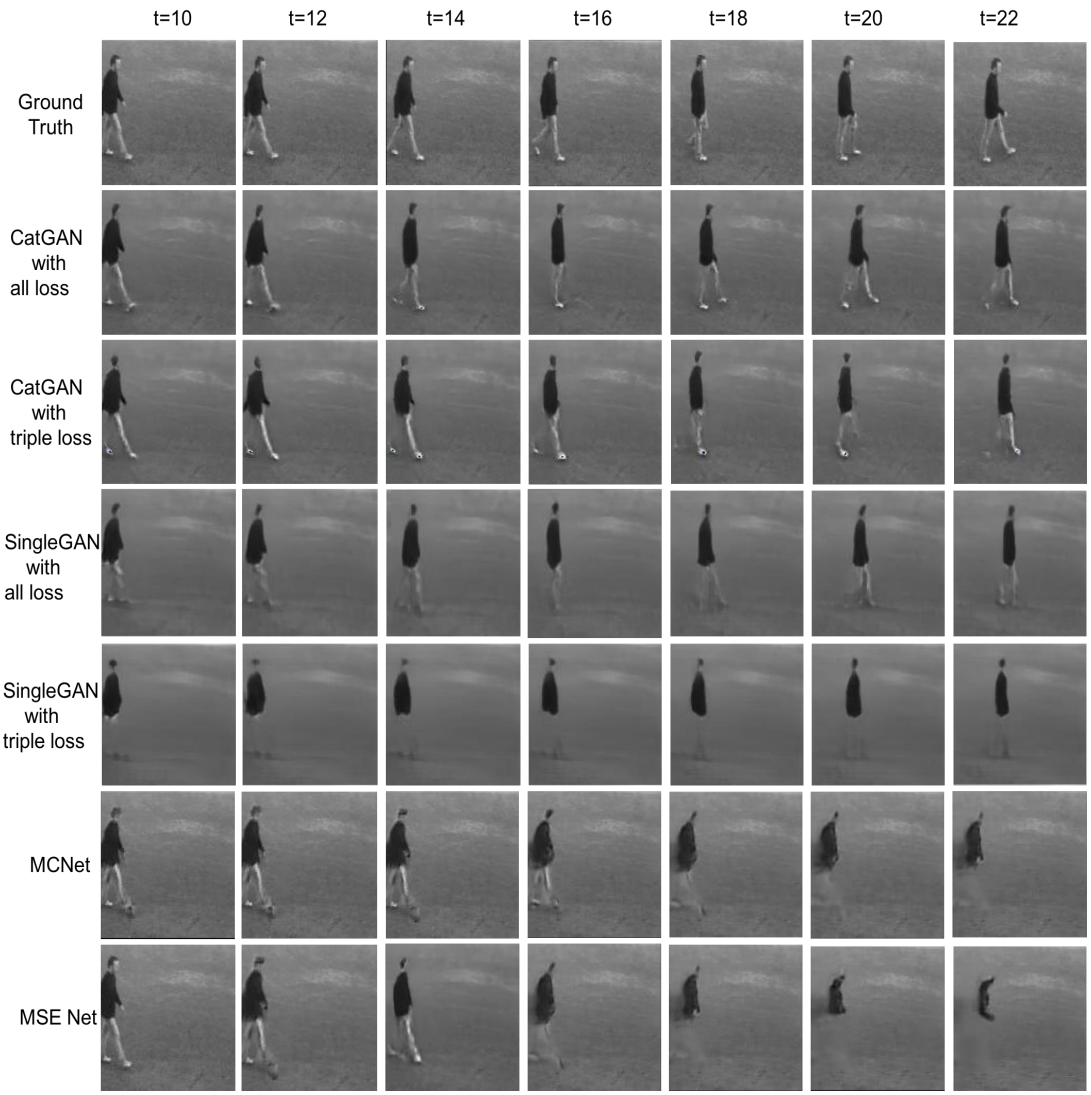


Fig. 3. Qualitative comparison in KTH dataset between our proposed solution, MCNet model [50] and MSE model [4]. The inputs for our model are the 10th frame and the 24th frame to predict the middle frames, the first ten frames for MCNet model and MSE model to predict the next 14 frames. The first row is the ground truth frames from the 10th frame to 22th frame. catGAN denote our proposed deep net, and catGAN + triple loss row denotes the results come from catGAN with adversarial loss, mse loss and gdl loss. All losses include triple loss and our proposed NPCL loss. We also made compared experiments to validate the effective of NPCL loss. Given 4 input frames from the $6^{th}$ frame to $9^{th}$ frame, the MCNet and the MSE model predict next 24 frames. Given the $4^{th}$ frame and the $29^{th}$ frames, our methods predict the missing 24 frames between them.

Without loss of generality, here we focus on generator $G_1$, which outputs frames $\bar{Y_1}, \cdots, \bar{Y_n}$. First, we measure the mean squared error (MSE) to estimate the difference between real frames $X_1$-$X_n$ and generated ones $\bar{Y_1}$-$\bar{Y_n}$. For each pair of frames $X$ and $\bar{Y}$, the corresponding MSE loss is defined as follows:

$$L_{mse} = ||\bar{Y} - X||_{\mathrm{p}}^{p} \tag{3}$$

where $p$ is set to 1 in our implementation [1]. Intuitively, MSE directly penalizes the mean differences of pixels in the real frames $X$ and generator output $Y$.

The MSE loss considers each frame independently, and does not capture frame transitions. Hence, we introduce another term in the loss function: gradient difference loss (GDL) [4]. GDL directly penalizes the differences of image gradient predictions.

1. We have also tested other values of $p$ empirically, and found $p = 1$ yields sharper frames.

$$\begin{aligned} L_{gdl} = &\sum_{i,j} \left| |\bar{Y}_{i,j} - \bar{Y}_{i-1,j}| - |\mathrm{X}_{i,j} - \mathrm{X}_{i-1,j}| \right|^{\alpha} \\ &+ \left| |\bar{Y}_{i,j-1} - \bar{Y}_{i,j}| - |\mathrm{X}_{i,j-1} - \mathrm{X}_{i,j}| \right|^{\alpha} \end{aligned} \tag{4}$$

In the above definition, $\alpha$ is a hyper-parameter, which is set to 1 in our implementation. Compared to Mean Square Error loss, GDL also ensures smooth frame transitions, and reduces motion blur; $i \in H, j \in W$, where $H$ and $W$ are the frame's height and width.

Lastly, we also include in the loss function the normalized product correlation loss function (NPCL) [36], which is a metric commonly used in image matching. NPCL loss is defined as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



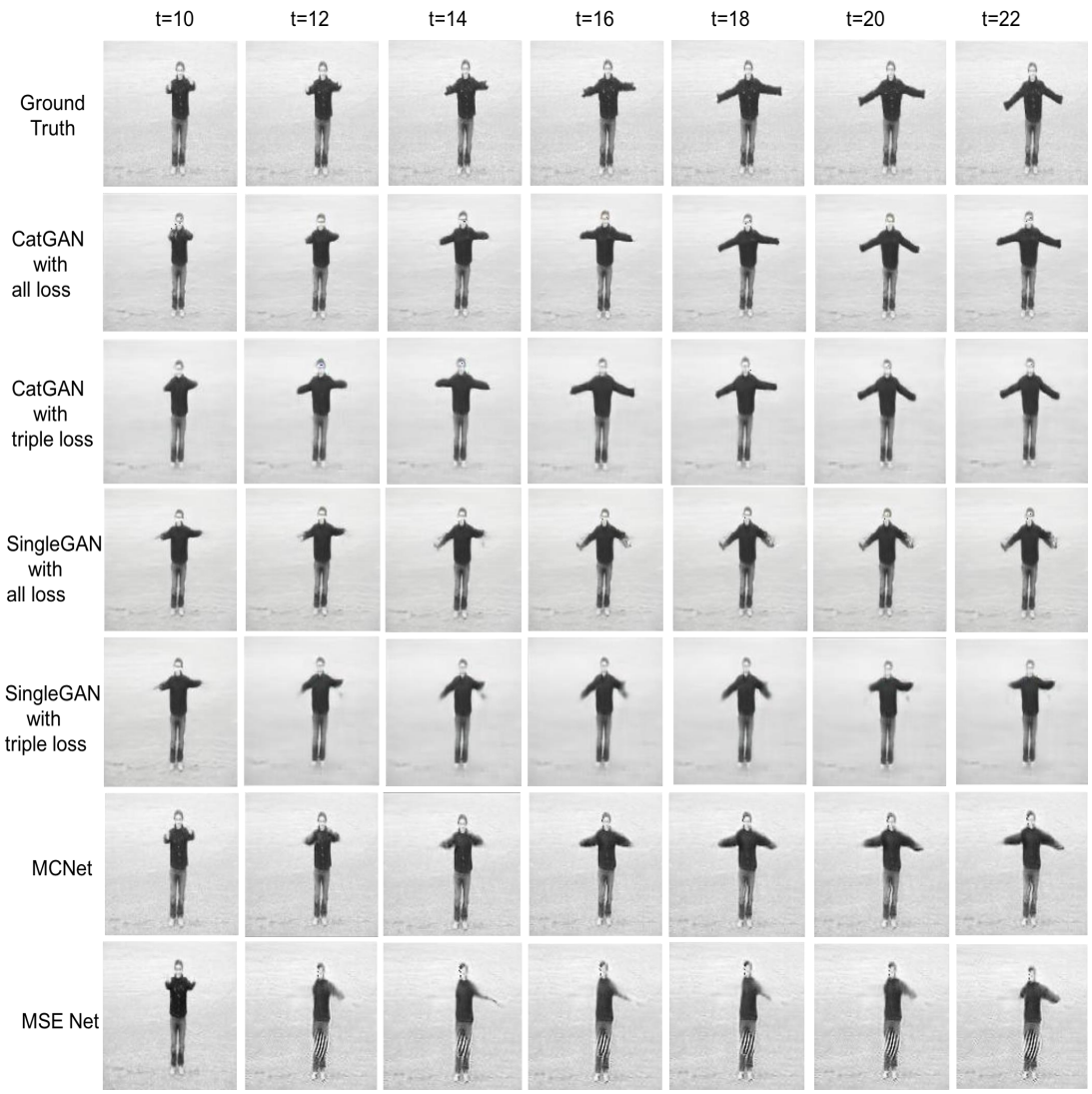


Fig. 4. Qualitative comparison in KTH dataset. We display the predictions starting from $4^{th}$ frames in every 2 timesteps. Given 10 input frames, the MCNet and the MSE model predict next 24 frames. Given the $10^{th}$ frame and the $35^{th}$ frames, our methods predict the missing 24 frames between them.

$$R(\bar{Y}, X) = \frac{\sum_{i,j} [\bar{Y}_{i,j} * X_{i,j}]}{\sqrt{\sum_{i,j} \bar{Y}_{i,j}^2 * \sum_{i,j} {X_{i,j}}^2}} \tag{5}$$

$$L_{npcl} = -\log(R(\bar{Y}, X)). \tag{6}$$

The logarithmic operation converts its value range from [0,1] to $(\infty, 0)$. When $R$ is equal to 1, $L$ is 0, and we can easily estimate that $L$ is convex function. Compared to MSE and GDP, NPCL is scale-free, and resilient to white noise. Hence, it measures the difference of real and generated frames from another perspective.

The joint loss loss function for each generator in our solution is then the weighted sum of the above loss components, as follows.

### 3.4 Implementation and Training

We implement the proposed solution using Tensorflow [35], and choose three datasets including UCF101, Google Push and KTH dataset to valid the effectiveness of our proposed solution. The first two dataset have more than 10000 action video clip. We choose to generate 128*128 frames for KTH and Google Push dataset and 240*320 resolution frames for UCF101 dataset, as in papers [4] and [50]. The detailed dataset information will be shown in Section 4. The generators in our solution involve 2D convolutions/deconvolutions with RELU activation, and Instance Norm between layers [49]. We choose Adam optimizer to optimize our Deep Net with beta1 as 0 and beta2 as 0.99. We use the following learning rate schedule: the initial learning rate is 0.0001, and it decays every 4000 iterations with decay factor 0.95. One epoch in training has more than 20000 iterations when trained on UCF101 dataset and Google Push dataset. Learning rate decay stops once its value is smaller than 1E-7. During training, the discriminators were first optimized and generators next. For detail, we first optimize discriminator1 and discriminator2 five times, and then optimize generator1 and generator2 one time. Our deep nets were trained in end to end way. The details of our deep net are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



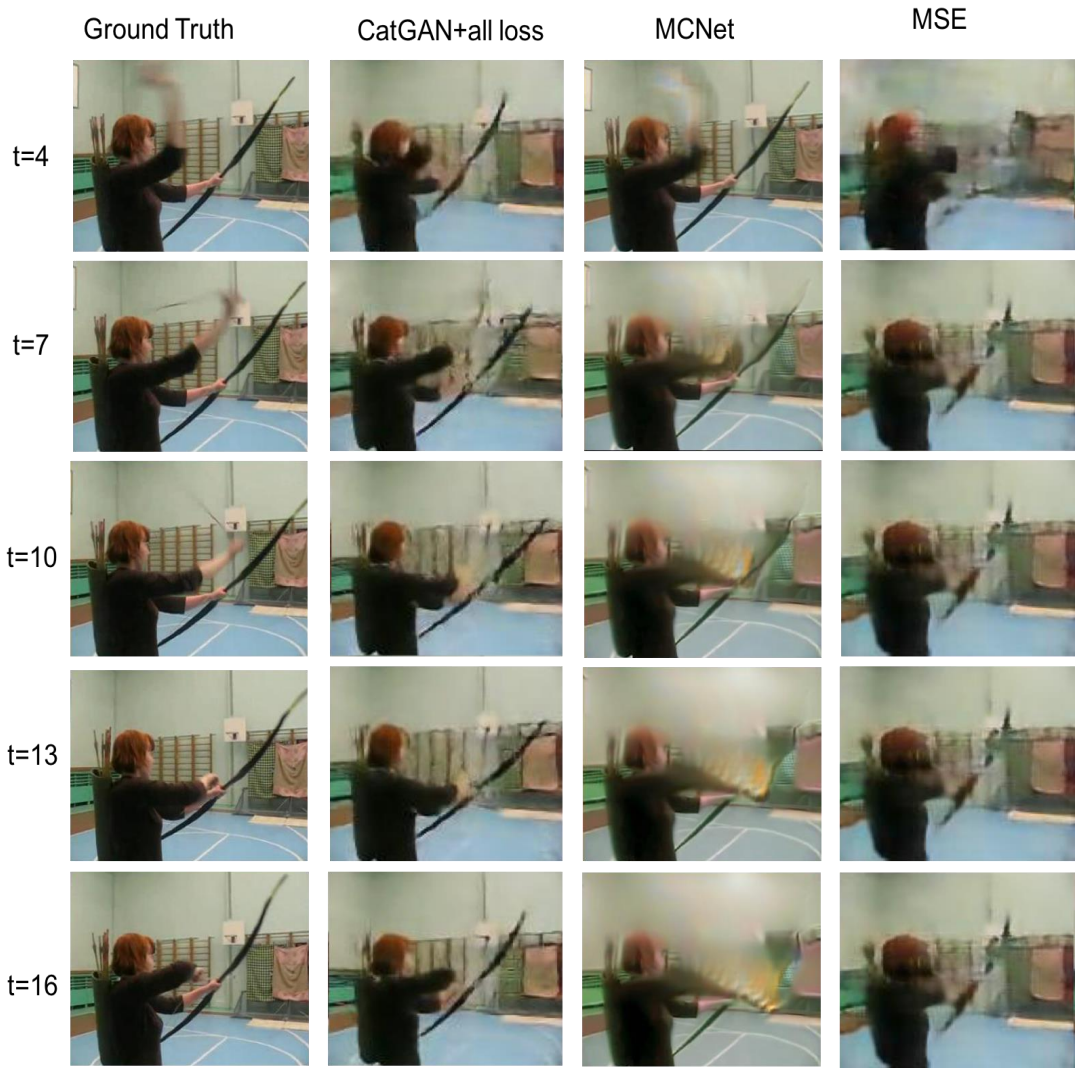


Fig. 5. Qualitative comparison in UCF101 dataset. Following the MCNet [50], we choose the first 4 frames as the inputs of MCNet [50] and MSE model [4], the 4th frame and the 19 th frame for our proposed solution. The triple losses include adversarial loss, mse loss and gdl loss. All loss include triple losses and NPCL loss. We display the predictions starting from $4^{th}$ frames in every 3 time steps.

| layers | encoder framework |
|---|---|
| 1 | Conv2d(5, 64, stride=1) ReLU |
| 2 | Conv2d(3, 128, stride=2) ReLU |
| 3 | Conv2d(3, 128, stride=1) ReLU |
| 4 | Conv2d(3, 256, stride=2) ReLU |
| 5-6 | Conv2d(3, 256, stride=1)x2 ReLU |
| 7-8 | Conv2d(3, 256, stride=1)x2 ReLU |
| layers | decoder framework |
| 1-2 | Conv2d(3, 256, stride=1)x2 ReLU |
| 3 | Deconv2d(4, 128, stride=2) ReLU |
| 4 | Conv2d(3, 128, stride=1) ReLU |
| 5 | Deconv2d(4, 64, stride=2) ReLU |
| 6 | Conv2d(3, 64,stride=1) ReLU |
| 7 | Conv2d(3, 3, stride=1) tanh |

TABLE 1
ConvNet structure for generator $G_1$.

| layers | details |
|---|---|
| 1-2 | Conv2d(3, 128, stride=1)x2 ReLU |
| 3 | Conv2d(3, 256, stride=2) ReLU |
| 4 | Conv2d(3, 256, stride=1) ReLU |
| 5 | Conv2d(1, 256, stride=2) ReLU |
| 6 | Conv2d(1, 256, stride=1) ReLU |
| 7 | Deconv2d(3, 512,stride=2) ReLU |
| 8-9 | Conv2d(3, 512, stride=1) x2 ReLU |
| 10 | Deconv2d(3, 512,stride=2) ReLU |
| 11-12 | Conv2d(3, 512, stride=1) x2 ReLU |
| 13 | Conv2d(3, 3, stride=1) tanh |

TABLE 2
ConvNet structure for generator $G_2$.

provided in Tables 1 and 2.

The first generator $G_1$ also utilizes explicit motion information, if such information is available in the dataset, e.g., motion vectors in Google Robotic Push. Such information is fed to $G_1$ together with the inputs. Nevertheless, the proposed solution does not rely on such explicit motion information, and it obtains good performance on datasets (such as KTH) without motion

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



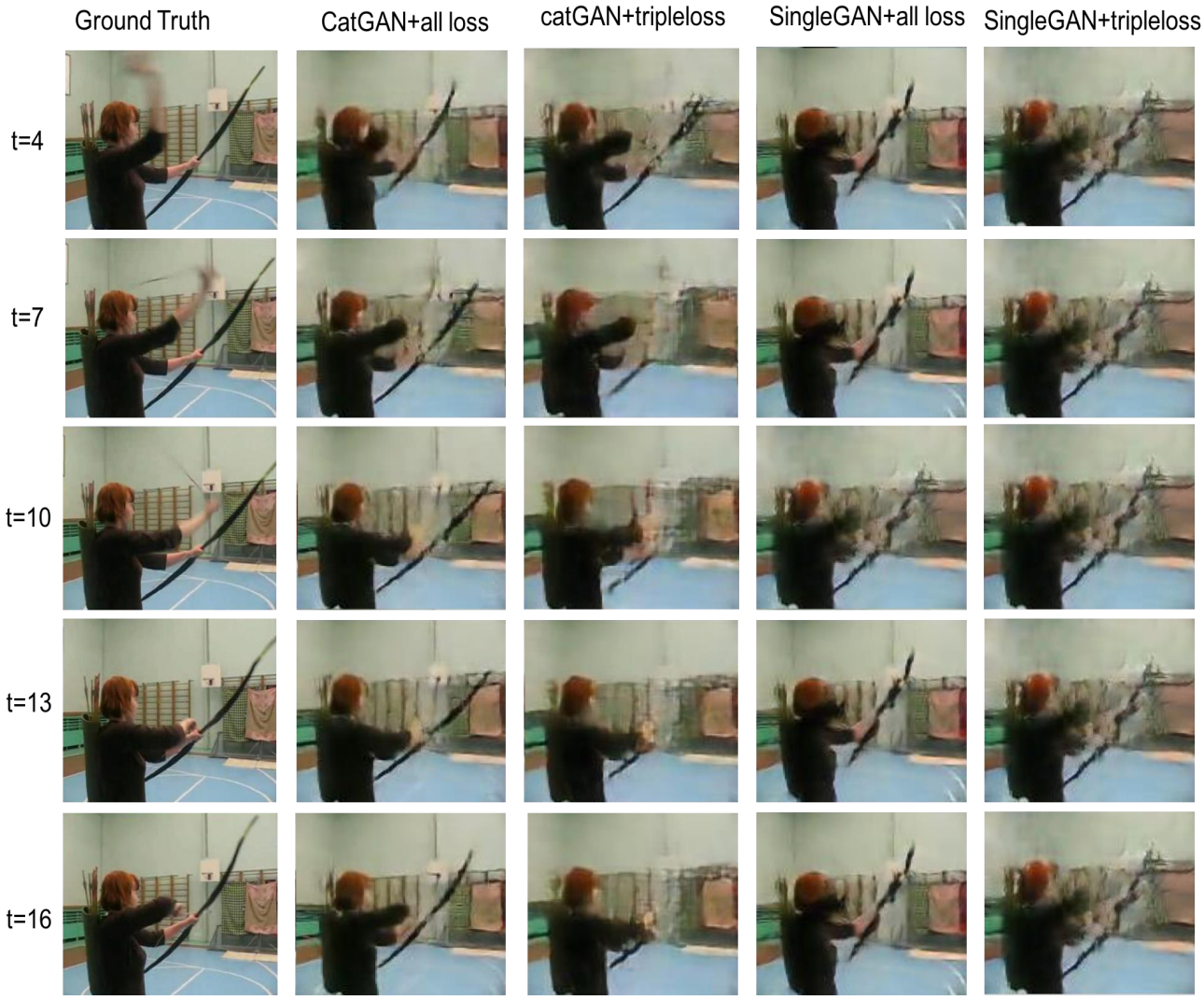


Fig. 6. The results of our methods. The first column is the ground truth. The last columns are the comparison of catGAN and single GAN model. We also validate the function of NPCL loss. We display the predictions starting from $4^{th}$ frames in every 3 time steps.

vectors, as shown in our experiments. For KTH and UCF dataset without motion vector, we train first generator to learn the image difference. It takes the key frames as inputs and output the image difference map. By adding to the starting frame, we can get the outputs $\bar{Y}$ of $G_1$ which used as the inputs of the second generator and first discriminator. The second generator $G_2$ is a U-Net, which contains cross-connections. Specifically, each convolution layer is connected to the corresponding deconvolution layer with the same image resolution. Both generators $G_1$ and $G_2$ are trained using the improved WGAN technique.

The two discriminators $D_1$ and $D_2$ share the same structure, which is a standard ConvNet pipeline, summarized in Table 3.

Hyperparameters $\lambda_1$, $\lambda_2$, $\lambda_3$, $\lambda_4$ are set to 0.001, 0.99, 1.0, and 1.0, respectively. The number of frames to be generated is set to $n = 10$ in our experiments. For the weight of NPCL loss, we experimentally set it as 1.0 because we observed NPCL loss is numerically similar with (slightly smaller than) the MSE loss and GDL loss when its weight is 1.0. Meanwhile we set the other hyperparamters inspired by [4] and [13].

| layers | details |
|---|---|
| 1 | Conv2d(5, 64, stride=2) LeakyReLU |
| 2 | Conv2d(3, 64, stride=1) LeakyReLU |
| 3 | Conv2d(3, 128, stride=2) LeakyReLU |
| 4 | Conv2d(3, 256, stride=2) LeakyReLU |
| 5 | Conv2d(3, 256, stride=2) LeakyReLU |
| 6 | Conv2d(3, 512, stride=2) LeakyReLU |
| 7 | Conv2d(3, 512, stride=1) LeakyReLU |
| 8 | fullyconnect(1) |

TABLE 3
Discriminator ConvNet structure.

## 4 Experiments

We evaluate the proposed solution using two datasets, described in Section 4.1. Section 4.2 shows the experimental results in terms of generated video frames[2]. Section 4.3 defines two quantitative

2. Note that since the results are video clips, their quality is not immediately clear when shown as 2D images. Hence, we encourage the reader to view the generated video samples in the supplementary materials.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



quality metrics PSNR and SSIM, and shows the evaluations results on these metrics.

### 4.1 Datasets

The KTH dataset includes six types of actions completed by 25 people in four different scenarios. There is only one subject in each video, and the background is fixed. We split the dataset for train and valid. The train dataset includes the first 16 person actions videos and the videos of the last actions for valid following [50]. Sample images in the dataset are shown in Figure 7.

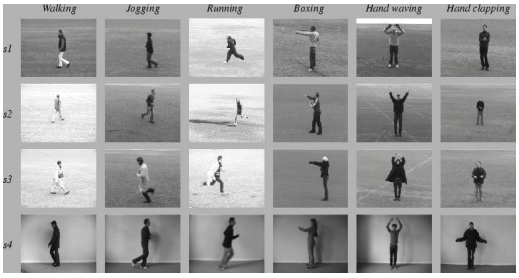


Fig. 7. Samples of KTH dataset.

Google Push dataset is collected from 10 robotic arms and their interaction sequences. It contains roughly 59,000 examples of robot pushing motions, including one training set (train) and two test sets of previously seen (testseen) and unseen (testnovel) objects. The dataset includes not only the video frames but also the corresponding gripper poses, which are stored as a vector. The dataset has already been divided into a training set and a test set. Sample images in the dataset are shown in Figure 8.

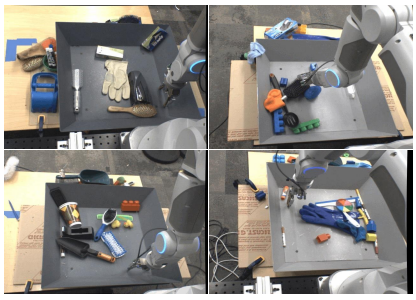

Fig. 8. Samples of Google Robotic Push dataset.

UCF101 is built for action recognition, which includes 13320 realistic action videos from 101 action categories, collected from Webnet. The videos in UCF101 have largest diversity in terms of actions and the large variations in camera motion. Therefore, it is a better dataset to valid the effective of video prediction and video compression.

### 4.2 Results

In order to expand train dataset, we split the video in train dataset into video clips in every 48 frames and discard videos shorter than 48 frames. In KTH dataset, given 4 input frames from the $6^{th}$ frame to the $9^{th}$ frame, the MCNet and the MSE model predict next 24 frames. Given the $4^{th}$ frame and the $29^{th}$ frames, our methods predict the missing 24 frames between them. For UCF

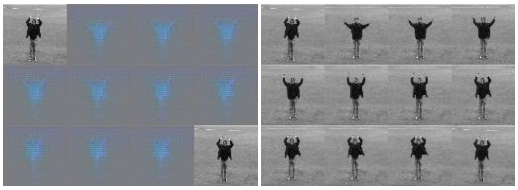

Fig. 9. Motion learning. Left: motion features extracted from generator $G_1$. Right: ground truth.

dataset, we use the even frames as the training data. We can get 24 even frames from the 48 frames video. And the first 4 frames are used for MCNet and MSE model to predict the next 10 frames. And the $4^{th}$ frames and the $15^{th}$ frames are used in our method to generate the missing 10 frames. For google robotic dataset, we extract the dataset from the ckpt files which proposed by google and use the first frame and the $13^{th}$ frame to predict the missing 12 frames.

Our experiment results are shown in Figures 9, 10, 3, 3.2, and 5. In Figure 9, we compare the motion features extracted from generator $G_1$ and the ground truth. Figures 3, 3.2, and 5 show the results generated on the Google Robotic Pushing dataset and the KTH dataset, respectively. In the experiment of hand-waving video generation, we did not add the extra action condition to the generator $G_1$ in our solution, though information such as human skeleton structure can be utilized to further enhance results.

We emphasize that since the results of our solution are video clips, it is best to view them in the supplemetary materials, instead of the static frames shown in the paper. Additionally, the supplementary materials also include videos generated by previous approaches, whose quality is noticeably worse than our solution. More results can be found online: https://github.com/LDOUBLEV/Results

### 4.3 Quantitative Evaluation

We calculate the Peak Signal to Noise Ratio (PSNR) and SSIM to evaluate the quality of the generative frames resulting from the tested experiments. PSNR is defined as follows:

$$PSNR = -10\log_{10}(\frac{(2^n-1)^2}{MSE}) \tag{7}$$

where MSE is the mean square error between the ground truth $X$ and prediction results $Y$:

$$MSE = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}|X_{i,j} - Y_{i,j}| \tag{8}$$

We also use SSIM [34] to measure the image similarity from image brightness, contrast to structure. The value of SSIM is range from 0 to 1, and the larger value of SSIM means the results have higher quality.

$$SSIM(X,Y) = \frac{(2\mu_{XY}\mu_Y + C_1)(2\sigma_{XY} + C_2)}{(\mu_X^2 + \mu_Y^2 + C_1)(\sigma_X^2 + \sigma_Y^2 + C_2)} \tag{9}$$

where $\mu_X$ and $\mu_Y$ denote the mean values of images $X$ and $Y$, $\sigma_X$ and $\sigma_Y$ denote the variances of $X$ and $Y$ , $\sigma_{XY}$ is covariance between $X$ and $Y$, and $C$ is a constant. In order to avoid the case of denominator is zero, it is usually set $C$ as the followed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2018.2867934, IEEE Transactions on Circuits and Systems for Video Technology



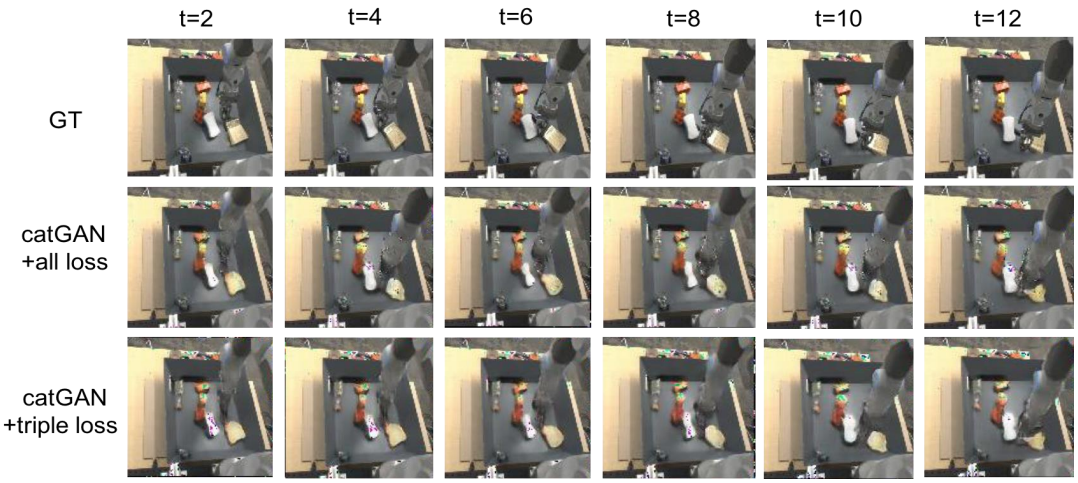


Fig. 10. The results of our methods on google push dataset. We display the predictions starting from the $2^{th}$ frames in every 2 time steps.

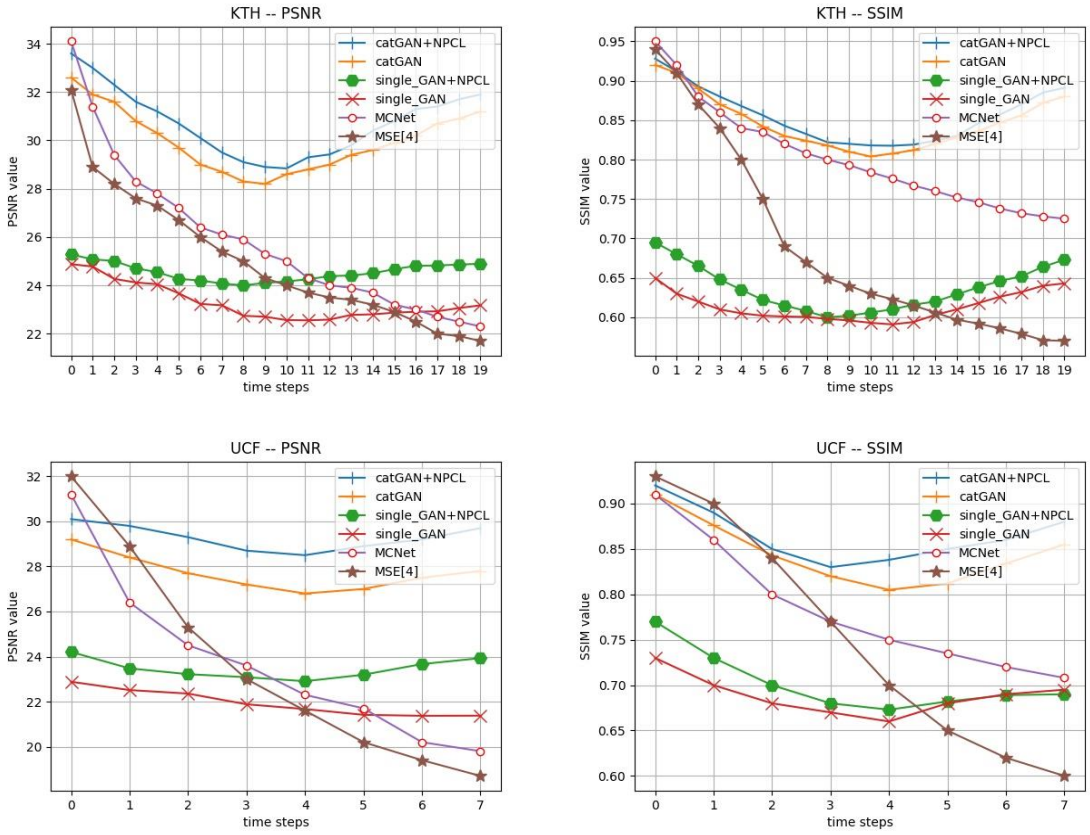


Fig. 11. Quantitative evaluation results. We use PSNR and SSIM as the metrics of models. The large value of PSNR and SSIM means the according model have better performance. We compare our algorithm against the MSE [4] and MCNet [50]. The detailed evaluation experiments setting can be found in Section 4.2

form, where $K_1$, $K_2$, and $L$ are equal to 0.01, 0.02, and 255 respectively:

$$C_1 = (K_1 * L)^2 \tag{10}$$

$$C_2 = (K_2 * L)^2 \tag{11}$$

We calculate the values of SSIM and PSNR, and show them in Figure 11, from which We can easily know that N-PCL loss can improve the quality of images, and we can get the most satisfying results using the concatenated nets with the joint loss. The code that we use to calculate SSIM is available on online: https://github.com/tensorflow/models/blob/master/

research/compression/image_encoder/msssim.py.

Summarizing the experimental results, the proposed solution successfully generates realistic videos from the input key frames, with correct motions and frame details. The concatenated GAN architecture is vital for the performance of our solution, and all components of the loss function contribute to the quality of the generated video frames. Quantitive measurements also confirm that the generated frames closely match those in the original videos. Overall, the proposed solution is highly effective in all tested settings.

## 5 Conclusion

We have designed and implemented an end-to-end concatenated GAN to generate videos using non-adjacent images as inputs. The proposed network architecture features two concatenated generators, for learning motions and frame details (using a U-Net), respectively. Extensive experiments demonstrate that the proposed solution generates clear and smooth frames that correctly captures both motions and frame details. Regarding future work, we plan to investigate the case where the number of frames $n$ is not given in advance, but as an input parameter. Another important direction is to generate longer video sequences involving complex motions and interactions.

## References


[1] Vondrick, Carl, Hamed Pirsiavash, and Antonio Torralba. "Generating videos with scene dynamics." Advances In Neural Information Processing Systems. 2016.

[2] Yan, Yichao, et al. "Skeleton-aided articulated motion generation." Proceedings of the 2017 ACM on Multimedia Conference. ACM, 2017.

[3] Oh, Junhyuk, et al. "Action-conditional video prediction using deep networks in atari games." Advances in neural information processing systems. 2015.

[4] Mathieu, Michael, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error." arXiv preprint arXiv:1511.05440 (2015).

[5] Finn, Chelsea, Ian Goodfellow, and Sergey Levine. "Unsupervised learning for physical interaction through video prediction." Advances in neural information processing systems. 2016.

[6] Saito, Masaki, Eiichi Matsumoto, and Shunta Saito. "Temporal generative adversarial nets with singular value clipping." IEEE International Conference on Computer Vision (ICCV). Vol. 2. No. 3. 2017.

[7] Zhou, Yipin, and Tamara L. Berg. "Learning temporal transformations from time-lapse videos." European Conference on Computer Vision. Springer, Cham, 2016.

[8] Walker, Jacob, et al. "An uncertain future: Forecasting from static images using variational autoencoders." European Conference on Computer Vision. Springer, Cham, 2016.

[9] Xue, Tianfan, et al. "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks." Advances in Neural Information Processing Systems. 2016.

[10] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[11] Lotter, William, Gabriel Kreiman, and David Cox. "Deep predictive coding networks for video prediction and unsupervised learning." arXiv preprint arXiv:1605.08104 (2016).

[12] Vondrick, Carl, Hamed Pirsiavash, and Antonio Torralba. "Anticipating visual representations from unlabeled video." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[13] Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint (2017).

[14] Wu, Huikai, et al. "Gp-gan: Towards realistic high-resolution image blending." arXiv preprint arXiv:1703.07195 (2017).

[15] Taigman, Yaniv, Adam Polyak, and Lior Wolf. "Unsupervised cross-domain image generation." arXiv preprint arXiv:1611.02200 (2016).

[16] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." arXiv preprint (2017).

[17] Sutskever, Ilya, Geoffrey E. Hinton, and Graham W. Taylor. "The recurrent temporal restricted boltzmann machine." Advances in neural information processing systems. 2009.

[18] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[19] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[20] Hinton, Geoffrey E. "Deep belief networks." Scholarpedia 4.5 (2009): 5947.

[21] Reed, Scott, et al. "Generative adversarial text to image synthesis." arXiv preprint arXiv:1605.05396 (2016).

[22] Ledig, Christian, et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network." CVPR. Vol. 2. No. 3. 2017.

[23] Brock, Andrew, et al. "Neural photo editing with introspective adversarial networks." arXiv preprint arXiv:1609.07093 (2016).

[24] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[25] Berthelot, David, Thomas Schumm, and Luke Metz. "BEGAN: boundary equilibrium generative adversarial networks." arXiv preprint arXiv:1703.10717 (2017).

[26] Arjovsky, Martin, Soumith Chintala, and Lon Bottou. "Wasserstein gan." arXiv preprint arXiv:1701.07875 (2017).

[27] Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." Advances in Neural Information Processing Systems. 2017.

[28] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).

[29] Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." Advances in neural information processing systems. 2016.

[30] LeCun, Yann, and Yoshua Bengio. "Convolutional networks for images, speech, and time series." The handbook of brain theory and neural networks 3361.10 (1995): 1995.

[31] Hochreiter, Sepp, and Jrgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[32] Kawakami, Kazuya. Supervised Sequence Labelling with Recurrent Neural Networks. Diss. PhD thesis. Ph. D. thesis, Technical University of Munich, 2008.

[33] Wu, Jiajun, et al. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." Advances in Neural Information Processing Systems. 2016.

[34] Wang, Zhou, et al. "Image quality assessment: from error visibility to structural similarity." IEEE transactions on image processing 13.4 (2004): 600-612.

[35] Abadi, Martn, et al. "Tensorflow: a system for large-scale machine learning." OSDI. Vol. 16. 2016.

[36] Li J S, Shen X B. Image Blocking Parallel Processing Approaches to a Normalized Product Correlation Image Matching Algorithm[J]. Mini-micro Systems, 2004, 25(11):1986-1989.

[37] Aslandogan, Y. Alp, and Clement T. Yu. "Techniques and systems for image and video retrieval." IEEE transactions on Knowledge and Data Engineering 11.1 (1999): 56-63.

[38] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[39] Karpathy, Andrej, et al. "Large-scale video classification with convolutional neural networks." Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2014.

[40] Niu, Wei, et al. "Human activity detection and recognition for video surveillance." ICME. Vol. 1. 2004.

[41] Niklaus, Simon, Long Mai, and Feng Liu. "Video frame interpolation via adaptive convolution." IEEE Conference on Computer Vision and Pattern Recognition. Vol. 1. No. 2. 2017.

[42] Niklaus, Simon, Long Mai, and Feng Liu. "Video frame interpolation via adaptive separable convolution." arXiv preprint arXiv:1708.01692 (2017).

[43] Schuldt, Christian, Ivan Laptev, and Barbara Caputo. "Recognizing human actions: a local SVM approach." Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. Vol. 3. IEEE, 2004.

[44] Myerson, Roger B. "Refinements of the Nash equilibrium concept." International journal of game theory 7.2 (1978): 73-80.

[45] Zhu, Jun-Yan, et al. "Generative visual manipulation on the natural image manifold." European Conference on Computer Vision. Springer, Cham, 2016.

[46] Denton, Emily L., Soumith Chintala, and Rob Fergus. "Deep generative image models using a laplacian pyramid of adversarial networks." Advances in neural information processing systems. 2015.
[47] Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." Proceedings of the 24th international conference on Machine learning. ACM, 2007.
[48] Salimans, Tim, et al. "Improved techniques for training gans." Advances in Neural Information Processing Systems. 2016.
[49] Ulyanov, Dmitry, A. Vedaldi, and V. Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization." (2017).
[50] Villegas, Ruben, et al. "Decomposing motion and content for natural video sequence prediction." arXiv preprint arXiv:1706.08033 (2017).
[51] Villegas, Ruben, et al. "Learning to generate long-term future via hierarchical prediction." arXiv preprint arXiv:1704.05831 (2017).
[52] Soomro, Khurram, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild." arXiv preprint arXiv:1212.0402 (2012).

**Shiping Wen** received the M. Eng. degree in Control Science and Engineering, from School of Automation , Wuhan University of Technology, Wuhan, China, in 2010, and received the Ph.D degree in Control Science and Engineering, from School of Automation, Huazhong University of Science and Technology, Wuhan, China, in 2013. He is currently an Associate Professor at School of Automation, Huazhong University of Science and Technology, and also in the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan, China. His current research interests include memristor-based circuits and systems, neural networks, and deep learning.

**Weiwei Liu** received his B. Eng. degree from Information Engineering school, Henan University of Science and Technology, Wuhan, China in 2017. He is currently working towards the M. Eng. degree from Huazhong University of Science and Technology, Wuhan, China. His research interests include computer version, multi-label classification, generative adversarial network and deep learning.

**Yin Yang** is currently an Assistant Professor in the College of Science and Engineering, Hamad Bin Khalifa University. His main research interests include cloud computing, database security and privacy, and query optimization. He has published extensively in top venues on differentially private data publication and analysis, and on query authentication in outsourced databases. He is now working actively on cloud-based big-data analytics, with a focus on fast streaming data.

**Tingwen Huang** is a professor at Texas A & M University-Qatar. He received his B.S. degree from Southwest Normal University (now Southwest University), China, 1990, his M.S. degree from Sichuan University, China, 1993, and his Ph.D. degree from Texas A & M University, College Station, Texas, 2002. After graduated from Texas A & M University, he worked as a Visiting Assistant Professor there. Then he joined Texas A & M University at Qatar (TAMUQ) as an Assistant Professor in August 2003, then he was promoted to Professor in 2013. His research interests include neural networks based computational intelligence, distributed control and optimization, nonlinear dynamics and applications in smart grids. He has published more than three hundred peer-review reputable journal papers, including more than one hundred papers in IEEE Transactions. Currently, he serves as an associate editor for four journals including IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Cybernetics, and Cognitive Computation.

**Zhigang Zeng** received his B.S. degree from Hubei Normal University, Huangshi, China, and his M.S. degree from Hubei University, Wuhan, China, in 1993 and 1996, respectively, and his Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 2003. He is a professor in School of Automation, Huazhong University of Science and Technology, Wuhan, China, and also in the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan, China. His current research interests include neural networks, switched systems, computational intelligence, stability analysis of dynamic systems, pattern recognition and associative memories.