This ECCV 2018 workshop paper, provided here by the Computer Vision Foundation, is the author-created version. The content of this paper is identical to the content of the officially published ECCV 2018 LNCS version of the paper as available on SpringerLink: https://link.springer.com/conference/eccv

Context Graph based Video Frame Prediction using Locally Guided Objective

Prateep Bhattacharjee[1]{orcid.org/0000-0003-1319-3430},
Sukhendu Das[2]{orcid.org/0000-0002-2823-9211}

Department of Computer Science & Engineering,
Indian Institute of Technology Madras, Chennai, Tamil Nadu, India
[1]prateepb@cse.iitm.ac.in, [2]sdas@iitm.ac.in,

Abstract. This paper proposes a feature reconstruction based approach using pixel-graph and Generative Adversarial Networks (GAN) for solving the problem of synthesizing future frames from video scenes. Recent methods of frame synthesis often generate blurry outcomes in case of long-range prediction and scenes involving multiple objects moving at different velocities due to their holistic approach. Our proposed method introduces a novel pixel-graph based context aggregation layer (PixGraph) which efficiently captures long range dependencies. PixGraph incorporates a weighting scheme through which the internal features of each pixel (or a group of neighboring pixels) can be modeled independently of the others, thus handling the issue of separate objects moving in different directions and with very dissimilar speed. We also introduce a novel objective function, the Locally Guided Gram Loss (LGGL), which aides the GAN based model to maximize the similarity between the intermediate features of the ground-truth and the network output by constructing Gram matrices from locally extracted patches over several levels of the generator. Our proposed model is end-to-end trainable and exhibits superior performance compared to the state-of-the-art on four real-world benchmark video datasets.

1 Introduction

Although video understanding has been one of the key areas of computer vision, the problem of predicting frames from natural video scenes has not been explored till recently. Compared to image reconstruction tasks, generation of multiple video frames requires understanding of non-trivial spatio-temporal feature representations. Past approaches in this area involve the use of Long Short Term Memory (LSTM) networks [1], [2], recurrent neural networks [3] and action conditional deep networks [4]. Majority of the recent approaches [5], [6], [7] focus on predicting the semantics which is useful in decision making problems. Mathieu *et.al.* [8] proposed a frame prediction model based on Generative Adversarial Networks (GAN). Contrary to the semantic based approaches, this multi-scale GAN uses the future frames (during training) as target from large amount of natural video scenes to produce crisp and clear output. This method also overcomes the issue of producing blurry output when the L2 penalty is used as the objective function, by introducing a gradient based loss (GDL). Recently, Villegas *et.al.* [9] incorporates different encoding streams for motion and content by using LSTM

based encoders. Another approach in synthesizing frames is to use pixel-autoregressive models [10], [11]. Also, latent variable models *viz.* Variational Auto-encoders (VAE) [12] have been used for both single [13] and multi-frame predictions [14]. Although these approaches (also [15], [16], [17], [18]) offer improvement in the quality of the produced frames over the semantic based models, they often fail to perform satisfactorily in environments differing greatly from the training set and in case of faraway predictions. Very recently, [19] used a two-stage GAN based framework and introduces two novel objective functions based on cross-correlation and a distance based divergence measure. This work captures the spatial as well as temporal information through the use of a cross-correlation based loss. Although this improves the scores over the state-of-the-art, it often fails on situations where multiple objects having very different velocity are present in the scene simultaneously. We overcome this issue by modeling the internal features corresponding to the pixels (or a group of pixels) using graph based structures which capture contextual dependencies in spatio-temporal regions.

Our proposed work incorporates two GAN based models: (a) feature generating GAN (F-GAN) and (b) reconstruction GAN (R-GAN). These two networks act in an encoder-decoder arrangement and are trained together end-to-end. F-GAN learns to generate intermediate feature representations, while R-GAN reconstructs the future frames from the generated feature space. The PixGraph layer sits on top of the convolutional part of F-GAN. The learning process of F-GAN is aided by a Locally Guided Gram Loss (LGGL) function. This minimizes the distance between the intermediate feature maps of the ground truth frames, produced by another auxiliary network and the feature maps from the F-GAN itself, by forming soft "local guidance" regions. The auxiliary network is trained simultaneously along with the two generators. The raison d´être of the feature generating network is to break the process of frame prediction from a direct coarse strategy into a finer multi-step method. The reconstruction network minimizes the L1 loss along with a gradient based objective function, Gradient Divergence Loss (GDL) [8]. The salient parts of our work are: (a) graph based context aggregation layer (PixGraph) with a novel weighting scheme for context aggregation, (b) novel encoder-decoder type GAN based architecture in the domain of video frame prediction and (c) a novel feature based objective function (LGGL) for bridging the gap between the predicted and target outputs efficiently. We quantitatively and qualitatively evaluate the proposed method using four popular real world video datasets: (a) KTH [20], (b) Weizmann [21], (c) UCF-101 [22] and (d) KITTI [23].

2 Proposed architecture

The overall proposed system for predicting future frames mimics an encoder-decoder like arrangement via an amalgamation of two GANs [24]. Overall, these are composed of two sub-networks: (a) the Generator (G) and (b) the Discriminator (D). The generator (G) is trained to produce realistic outputs by learning the underlying true data distribution p_{data} and consequently making the job of differentiating between true and synthetic outputs by the discriminator harder. In contrast, the discriminator D learns to distinguish between the real data and the synthetic outputs produced by the generator.

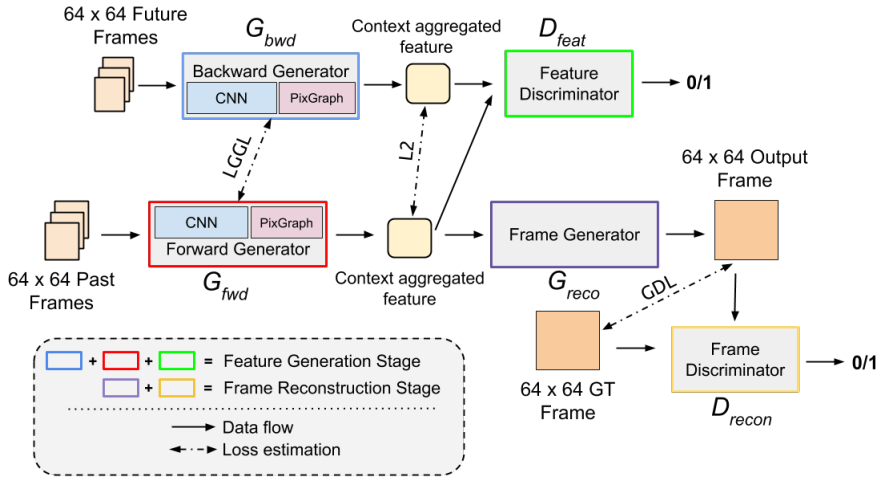


Fig. 1: The proposed GAN based network architecture for video frame prediction

In short, GANs use an alternating two player min-max game-like learning strategy [24] to achieve the mixed equilibrium of generating better synthetic outputs. The objective function minimized by the GANs [24] is given by

$$\min_G \max_D v(D,G) = \mathbb{E}_{x\sim p_{data}}[log(D(x))] + \mathbb{E}_{z\sim p_z}[log(1-D(G(z)))] \quad (1)$$

where, x is a sample from the true distribution p_{data} and vector z is sampled from a uniform (or Gaussian) noise distribution p_z. As our work deals with videos instead of arbitrary data distributions, the input to F-GAN is a sequence of video frames (discussed in section 2.2), while R-GAN receives a collection of 2D intermediate context aggregated feature map(s) (refer figure 1). The following sub-sections describe the working principles of the proposed PixGraph module along with the feature generation and reconstruction stages.

2.1 PixGraph module

The feature generation stage comprises of two generators, G_{fwd} and G_{bwd} and a discriminator D_{feat}. The main purpose of this feature generating stage is to incorporate the idea of context aggregation and local region representation. We propose PixGraph, a graph based Recurrent Neural Network (RNN) [25] layer to perform context aggregation in this paper. In the setting of video frame prediction, it is natural to assume that nearby pixels are dependent to each other and graph based RNN models this contextual dependency in a elegant way.

Forward pass through PixGraph Mathematically, the working principle of RNNs [25] is expressed as:

$$\begin{aligned} h^{(t)} &= f(W_{IH}x^{(t)} + W_H h^{(t-1)} + b) \\ y^{(t)} &= g(W_{HO}h^{(t)} + c) \end{aligned} \tag{2}$$

where, the hidden layer $h^{(t)}$ at step t is represented as a non-linear function f over the current input $x^{(t)}$ and the hidden layer at the previous time step $h^{(t-1)}$; W_{IH} is the weight matrix between the input and hidden layers, W_H among the hidden layer themselves and W_{HO} is the output matrix connecting the hidden and output layers; b, c are the bias vectors; $y^{(t)}$ is the output layer and $g(.)$, a non-linear activation function.

Although the standard RNN [25] captures temporal dependencies in sequential data such as sentences, where a chain-structure is present, they are not suitable for images (or a sequence of frames). The reason behind this is the fact that, to make the image data usable for the standard RNN cells, the feature maps $x \in \mathbb{R}^{h\times w\times d}$ need to be reshaped into vectors $\hat{x} \in \mathbb{R}^{(h.w)\times d}$, thereby losing the spatial dependency of the pixel elements. Therefore, we employ a spatio-temporal graphical structure which respects the spatial as well as temporal arrangement of the pixels as shown in figure 2.

We represent the configuration of the feature map of the target frame as a graph $G = \{V, E\}$, where $V = \{v_i\}_{i=1:N}$ is the vertex set and $E = \{e_{ij}\}$ is the edge set with e_{ij} denoting a connection from vertex v_i to v_j. As the model generates the forward propagation sequence by traversing the graph G, it should be noted that a node is processed only after all its predecessors are processed. For this step, we assume that contextual dependency falls in four categories based on the direction of motion: north-east ($0^\circ - 89^\circ$), north-west ($90^\circ - 179^\circ$), south-west ($180^\circ - 269^\circ$) and south-east ($270^\circ - 359^\circ$). Using this four-way decomposition, a pixel is processed only if all the pixels in its corresponding predecessor direction (*i.e.* top-right pixels for the north-east decomposition) has been processed by the RNN. Figure 2b shows this predecessor relationship for the north-west decomposition graph.

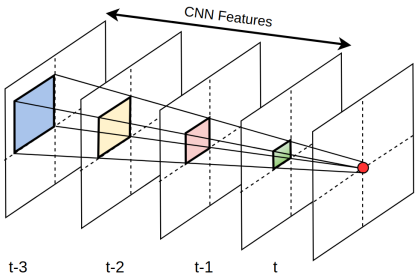
For representing the aforementioned decomposition more formally, let the set of four direction based graphs be $\hat{G} = \{G_1, G_2, G_3, G_4\}$. The graph based RNN is applied simultaneously and independently on each of these decompositions to generate four corresponding hidden layer feature maps h_i, $i = 1:4$. These are finally weight-aggregated to form the final hidden layer feature map h. Mathematically, these operations are expressed as:

$$h_i^{(v_j)} = f(W_{IH}^i x^{(v_j)} + \sum_{v_k \in \mathcal{P}_{G_i}(v_j)} W_{HH}^i h_i^{(v_k)} + b_i) \tag{3}$$

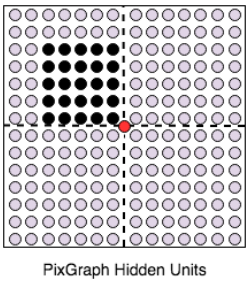
$$o^{(v_j)} = g(\sum_{G_i \in \hat{G}} \lambda_i W_{HO}^i h_i^{(v_j)} + c) \tag{4}$$

where, $\mathcal{P}_{G_i}(v_j)$ is the set of parent or predecessor pixel units of v_j in the component graph G_i and λ_i are the corresponding weights for each of the components.

Equations 3-4 consider the spatially contextual relations only for the current time-step. This is extended to multiple time-steps in the past for capturing the temporal context by considering a set of component graph sets $\hat{\mathbf{G}} = \{\hat{G}_{t-3}, \hat{G}_{t-2}, \hat{G}_{t-1}, \hat{G}_t\}$. Also, we



(a) The hidden state of the current processing unit (in red) is connected to all the parent input units in the shaded regions of previous time-steps (denoted by t). The dimension of this valid predecessor region grows as we go further into the past



(b) The hidden state of the current processing unit (in red) is also connected to the hidden states of the already processed units. To limit influence from far-away sections, the predecessor set of vertices is restricted to a small neighborhood (black units)

Fig. 2: The PixGraph module. Both (a) and (b) illustrates the north-west component of the context graph. Similar logic applies for the other directions

assume that the motion is smooth such that each pixel (or a set of pixels) is only dependent on a small neighborhood around it in the previous step. Following this, we restrict the valid predecessor units to a small 3×3 neighborhood region in the corresponding direction of the component. This local region is gradually grown to a 9×9 patch for $\hat{G}_{t-3}$ to capture the motion efficiently over a short time-period. Considering the fact that each of the graphs $\hat{G}_{t-i}$ is broken into 4 directional components, a 9×9 local window practically captures information from a 17×17 neighborhood centered around the currently processing pixel unit. Under this framework, equation 3 is re-written as:

$$h_i^{(v_j)} = f(W_{IH}^i x^{(v_j)} + \sum_{\substack{v_l \in \mathcal{P}_{G_{t-a}^i}(v_j) \\ a \in \{0,1,2,3\}}} W_{PH}^{t-a,i} x^{(v_l)} + \sum_{v_k \in \mathcal{P}_{G_i}(v_j)} W_{HH}^i h_i^{(v_k)} + b_i) \quad (5)$$

where, the first and last summation terms capture the contextual information from the previously processed pixel features whereas, the second summation involving $W_{PH}^{t-a,b}$ captures motion information from the local regions of the feature maps of the previous frames with $t - a$ denoting the time-step and i representing the corresponding directional component as discussed before. Equation 4 remains unchanged as the weighted-aggregation logic stays the same under this framework.

Computing the gradients To facilitate the learning process using backpropagation algorithm the derivatives are calculated for the PixGraph module in the backward pass. In this case, each of the vertices are processed in the reverse order of the previous forward propagation sequence. Hence, instead of considering the predecessor set $\mathcal{P}(v_j)$ as discussed before, a successor set $\mathcal{S}(v_j)$ is calculated. Using this, the error accumulated to the hidden unit for vertex v_j is broken into two components: (a) errors from v_j:

$\frac{\partial o^{(v_j)}}{\partial h^{(v_j)}}$ and (b) sum of errors from the successor set: $\sum_{v_k} \frac{\partial o^{(v_k)}}{\partial h^{(v_j)}} = \sum_{v_k} \frac{\partial o^{(v_k)}}{\partial h^{(v_k)}} \frac{\partial h^{(v_k)}}{\partial h^{(v_j)}}$. This way, the derivatives are calculated as:

$$\begin{aligned} dh_i^{(v_j)} &= \frac{\partial o^{(v_j)}}{\partial h_i^{(v_j)}} + \sum_{v_k} \frac{\partial o^{(v_k)}}{\partial h_i^{(v_k)}} \frac{\partial h_i^{(v_k)}}{\partial h^{(v_j)}} \\ &= \lambda_i (W_{HO}^i)^T g'(o^{(v_j)}) + \sum_{v_k \in \mathcal{S}_{G_i}(v_j)} (W_{HH}^i)^T dh_i^{(v_k)} \circ f'(h_i^{(v_k)}) \end{aligned} \tag{6}$$

where, $\mathcal{S}$ is the successor set of vertex v_j and $\circ$ represents the Hadamard product. Similarly, the gradients of the other learned parameters are computed as:

$$\nabla W_{HO}^i = g'(\lambda_i W_{HO}^i h^{(v_j)} + c)(\lambda_i h_i^{(v_j)})^T \tag{7}$$

$$\nabla W_{HH}^i = \sum_{v_k \in \mathcal{S}_{G_i(v_j)}} dh_i^{(v_k)} \circ f'(h_i^{(v_k)}) h_i^{(v_j)} \tag{8}$$

$$\nabla W_{IH}^i = dh_i^{(v_j)} \circ f'(h_i^{(v_j)})(x^{(v_j)})^T \tag{9}$$

$$\nabla W_{PH}^{t-a,i} = \sum_{v_l \in \mathcal{S}_{G_{t-a}^i}(v_j)} dh_i^{(v_j)} \circ f'(h_i^{(v_l)})(x^{(v_j)})^T \tag{10}$$

where, the symbols bear the usual notation as mentioned before.
Apart from the derivatives of the weight matrices, the λ parameter can be fixed at some predefined value (*viz.* $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}$ to give equal importance to the features from all the direction based contexts), or can be learned as well. To give our model the maximum flexibility, we chose to learn the λ_i values and compute the derivatives as follows:

$$\nabla \lambda_i(v_j) = g'(\lambda_i W_{HO}^i h^{(v_j)} + c)(W_{HO}^i h_i^{(v_j)})^T \tag{11}$$

Learning the λ_i values instead of giving equal weight to all the components helps to model situations where multiple close-by objects move in different directions *e.g.* a busy road-crossing where pedestrians are moving in all sorts of directions.

2.2 Stage-1: Feature generation using PixGraph

As shown in figure 1, two similar generators, G_{fwd} and G_{bwd} are used to generate an intermediate representation of the target frame(s). For ease of discussion, we will assume that the network takes as input a sequence of M frames and generates a single frame at the next time-step. All the input and output frames have the same dimension $W \times H$ ($W = H$ for simplicity). G_{fwd} is used to output the internal representation of the target frame at time-step $t + 1$ with a sequence of M frames at time-steps $T = t - M + 1, t - M + 2, ..., t$ as input. On the other hand, G_{bwd} also creates another feature representation of the same target frame at time-step $t + 1$ from a sequence of N consecutive frames at time-steps $T = t + 2, t + 3, ..., t + N + 1$. Essentially, G_{fwd} creates a representation in the forward direction while G_{bwd} does the same in the backward direction. From figure 1, the initial part of both the networks are standard convolutional

networks with the last layer being the PixGraph module as discussed in the previous subsection. Note that, G_{bwd} is only used in the training phase for future frame generation. Although, the same model can be used for interpolation of frames as well, in which case we retain the G_{bwd} in the evaluation stage also. Thus, using this forward-backward encoding of intermediate features, the model serves a dual purpose of interpolation as well as extrapolation of video frames. The outputs from both these networks are fed to the feature discriminator (D_{feat}). As the motivation of using the feature generator is to bring the intermediate feature maps of the forward and backward generators closer in the feature space, D_{feat} is trained to distinguish between the synthesized feature maps produced by G_{fwd} and G_{bwd}. The target labels chosen for features fed to D_{feat} are: '0' for the feature maps generated by G_{fwd} and '1' for that from G_{bwd}. Apart from minimizing the adversarial objective of GAN, the model also minimizes the L2 distance of the intermediate feature maps produced by G_{fwd} and G_{bwd}.
Although GANs are notorious for their training instability, our proposed model of simultaneous learning of the discriminator along with the forward and backward generators is elegant and successful due to: *(a)* Although minimizing the sum squared distance (L2) is a traditional method used in optimization literature, it results in production of blurry frames. This shortcoming is evaded in our work by minimizing the L2 distance between a large number of high-dimensional feature maps instead of the pixels in the generated frames and *(b)* Projecting the past and future frames to a shared space of context aggregated features using PixGraph and a novel non-linear objective function (described in section 3) increases separability for better discrimination.

2.3 Stage-2: Reconstruction

The reconstruction phase of the frame prediction framework is essentially another Generative Adversarial Network fine-tuned to produce sharp, high-quality sequence of frames. This phase consists of two networks: (a) frame generator (G_{recon}) and (b) frame discriminator (D_{recon}). The context-aggregated internal feature map produced by the PixGraph module on top of G_{fwd} of dimension $W_{int} \times H_{int}$ is fed as input to G_{recon}, which in turn produces the final output frame at time-step $t+1$. For simplicity, we keep the number of frames generated to 1. The discriminator, (D_{recon}), at this stage is trained to distinguish between the synthetic and true data in the RGB image space.
Exploration of different architectures for the feature generation and reconstruction stages led to an observation that using residual blocks in the reconstruction stage produces comparatively better outputs at less training time. Also, the same helps in diminishing the effect of vanishing gradients through by-passing higher level features using identity pathways.

3 Locally Guided Gram Loss (LGGL)

The joint problem of projecting the input video frame sequence in an intermediate feature space shared by the features from the future frames and optimizing the discriminator to differentiate between them simultaneously is a non-trivial problem. For these situations, we use feature maps from several layers of G_{fwd} and G_{bwd}. By breaking the

feature maps into local regions, the proposed objective function guides the neurons of G_{fwd} to transform the original input data to the targeted feature space.
Let $FM_k^{fwd}(X)$ and $FM_k^{bwd}(Y)$ denote the feature representations of the past and future frame sequences X and Y used as input to G_{fwd} and G_{bwd} respectively at layer k of G_{fwd} and G_{bwd} respectively. Also, each of the columns of FM_k^{fwd} and FM_k^{bwd} are vectorized *i.e.* $FM_k \in \mathbb{R}^{W_k \times H_k \times N_k}$ where, W_k, H_k and N_k are the width, height and number of feature maps at layer k respectively. Our aim is to transform each small local (spatially) region of the feature maps from G_{fwd}, at several layers by using those from G_{bwd}. We divide the feature maps into R non-overlapping square regions and use the concept of Gaussian Guidance Maps (GGM). GGMs are essentially normalized ($[0, 1]$) image maps and specifies how much weightage is given to a specific neuron in the guiding policy. We noticed that the neurons near the border of all the R regions contribute almost equally to capture an intermediate feature; whereas, those near the center are often mutually exclusive in their role. Armed with this intuition, we assign a higher weightage for the center neurons in the GGM and lower it near the borders. This is modeled with the help of a Gaussian distribution. Note that the guidance maps have zero values for areas outside the boundaries of a particular region, thereby suppressing unwanted effects from those (*e.g. sky*) far away from the current one (*e.g. road*).
The feature maps of each of the K layers are multiplied with R GGM to compute the Locally Guided Gram matrix (LGG) $\mathcal{G}$. This is defined as:

$$[FM_k^{fwd}]_r = [GGM]_r \circ FM_k^{fwd} \tag{12}$$

$$[\mathcal{G}_k^{fwd}]_r = [FM_k^{fwd}]_r^T [FM_k^{fwd}]_r \tag{13}$$

where, $r \in R$, $\circ$ represents element-wise multiplication, $[GGM]_r$ is the normalized GGM for region r and $[\mathcal{G}_k^{fwd}]_r$ is the LGG of the same. We compute the values for the feature maps of the backward network simultaneously, using analysis similar to equations 12 and 13.
Each of the LGG matrices now becomes the target for optimization for the corresponding regions in the feature space. This way, the objective function for LGGL is expressed in Sum of Squared Distance (SSD) form as:

$$\mathcal{L}_{LGGL} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{R^2 (N_k)^2} \sum_{r=1}^{R} \left[[\mathcal{G}_k^{fwd}]_r - [\mathcal{G}_k^{bwd}]_r \right]^2 \tag{14}$$

As the granularity of the extracted features vary proportionally with the depth of CNNs, $\mathcal{L}_{LGGL}$ minimizes the feature distance in a hierarchical fashion and greatly enhances the stability of the feature generating stage.

4 Overall objective function

We integrate the objective functions described in section 3 with the traditional adversarial loss function [24] as well as the well-known $L1$ and $L2$ metrics for the proposed

frame prediction network. The generator (G_{fwd}) in the feature generation stage thus optimizes the following weighted function:

$$\mathcal{L}_{comb}^{fwd} = \lambda_{adv}^{fwd}\mathcal{L}_{adv}^{G_{fwd}}(X) + \lambda_{LGGL}\mathcal{L}_{LGGL}(X,Y) + \lambda_{L2}^{fwd}\mathcal{L}_{L2}(X_{int},Y_{int}) \quad (15)$$

where, the symbols bear the standard notations as mentioned in the previous sections. The optimal values for the co-efficients λ_{adv}^{fwd}, λ_{L2}^{fwd} and λ_{LGGL} are empirically found to be $0.10, 0.45$ and 0.45. Similarly, the objective function for the backward generator G_{bwd} is:

$$\mathcal{L}_{comb}^{bwd} = \lambda_{adv}^{bwd}\mathcal{L}_{adv}^{G_{bwd}}(X) + \lambda_{LGGL}\mathcal{L}_{LGGL}(X,Y) + \lambda_{L2}^{bwd}\mathcal{L}_{L2}(X_{int},Y_{int}) \quad (16)$$

where, the co-efficients are kept equal to their forward generator counterparts. On the other hand, the combined loss function for the reconstruction phase generator (G_{recon}) includes the Gradient Divergence Loss (GDL) [8] for enhancing the sharpness of the generated frame(S) and is represented as:

$$\mathcal{L}_{comb}^{recon} = \lambda_{adv}^{recon}\mathcal{L}_{adv}^{G_{recon}}(X_{int}) + \lambda_{L1}^{recon}\mathcal{L}_{L1}(Y,Y') + \lambda_{GDL}\mathcal{L}_{GDL}(Y,Y') \quad (17)$$

In all our experiments, we keep the weights λ_{adv}^{recon}, λ_{L1}^{recon} and λ_{GDL} as $0.10, 0.25$ and 0.65 respectively (determined empirically for best performance).
These combined objectives in equations 15 - 17 when minimized simultaneously, reduce the gap in the intermediate and RGB features spaces of the target (GT) and generated frame(s), thereby generating better results.

5 Experiments

We evaluate our frame prediction network on four popular benchmark datasets: (a) KTH [20], (b) Weizmann [21], (c) UCF-101 [22] and (d) KITTI [23]. Among these, KTH and Weizmann contain various simple human actions in a predominantly static background. The UCF-101 dataset comprises of scenes with 101 different complex human actions. Although it contains much complex scenes than the KTH and Weizmann, it still suffers from the problem of static background which can lead the network to just learn to copy pixel values. To overcome this issue, we use Sports-1M [26], a large database of natural sports videos, for training. Lastly, KITTI contains street view scenes taken from a car mounted camera and generally does not suffer from the static background problem.
Quantitative studies with recent state-of-the-art and other methods have been carried out using two image quality measurement metrics: (a) Peak Signal to Noise Ratio (PSNR) [27] and (b) Structural Similarity Index Measure (SSIM) [28].

5.1 Results on KTH and Weizmann

The KTH dataset comprises of 6 different human actions from 25 subjects. Videos in this dataset have rather simplistic motions which are periodic and predominantly exhibit a static background. Following [9], we use persons $1-16$ for training and the rest for testing purpose. Apart from these, we also selected walking, running, one and

Fig. 3: Effect of layer selection in the feature generation stage. Zoomed-in patches of outputs from models optimized with LGGL taking into account: (a) early, (b) terminal and (c) optimum choice (mixed) of layers from G_{fwd} and G_{bwd} networks. The same patch from the ground-truth frame is shown in (d). Observe the trade-off between blurriness and sharpness in each of the cases (Best viewed in color)

two hands waving classes from the Weizmann database. All the experiments are done using 64×64 dimension input and target frames. The network is trained using a set of 10 consecutive frames to predict up-to 10 frames in the future. For a fair comparison with [9], clips from the running and jogging classes are sampled every 3rd frame while for the rest of the classes the sample period is one in every 10 frames. As our network predicts 10 frames in the future, the sampling rate is chosen to be 10 instead of 20 (as in [9]). Additional results for deeper prediction are shown the supplementary document. The layers conv_2, conv_4, conv_6 ,conv_9 and conv_10 of G_{fwd} and G_{bwd} were found to be the optimum choice for the Locally Guided Gram Loss (LGGL) in the feature generation stage (refer sections 1 & 2 in supplementary document for architecture). Choosing only the early layers for LGGL produced features which forced G_{recon} to predict output frames having sharp edges with overall color and texture informations intact, but lacking in finer details (e.g. cloth patterns). A reverse effect was observed (sharper details but blurry edges) when features were chosen from later parts of the feature generation network (see figure 3).

From the quantitative results presented in table 1, it is evident that the proposed framework is superior to its closest competitor MCNet both in terms of PSNR and SSIM quality measures. Also note that the drop in reconstruction quality is substantially less than other methods, as we go deeper into the future. This can be attributed to the weighted context aggregation of the PixGraph module which captures the temporal information in locally restricted areas. MCNet uses LSTMs as well as two separate networks for capturing both the temporal and spatial information. While the complex framework of MCNet learns periodic motions, it often fails to perform in case of complex scenes with highly non-periodic motion. Also, as our method learns an intermediate feature representation that is explicitly trained to fuel the prediction quality, the output frames look much more photo-realistic (refer to supplementary figures for additional illustrations).

5.2 Results on UCF-101

The UCF-101 dataset contains realistic scenes of 101 types of human actions collected from YouTube. Compared to KTH and Weizmann, this database is complex and larger

Table 1: Comparison of performance for KTH and Weizmann datasets using PSNR/SSIM scores. GDL stands for Gradient Divergence Loss [8]. Best results in bold.

	Frame-1		**Frame-4**	
Methods	**Weizmann**	**KTH**	**Weizmann**	**KTH**
ConvLSTM + RES [9]	36.2/0.97	33.8/0.94	29.8/0.92	27.9/0.86
MCNet + RES [9]	36.8/0.97	33.9/0.94	31.9/0.94	28.3/0.88
Adv + L1 (w/o PixGraph)	27.3/0.83	25.2/0.82	22.4/0.78	21.3/0.72
Adv + L1 + LGGL (w/o PixGraph)	30.1/0.85	29.9/0.85	24.8/0.79	23.5/0.73
Adv + L1 (PixGraph)	38.4/0.95	35.1/0.93	33.7/0.94	30.8/0.89
Adv + L1 + LGGL (PixGraph)	41.8/0.97	39.8/0.96	38.2/0.95	36.1/0.91
Adv + L1 + LGGL + GDL (PixGraph)	**42.5/0.98**	**40.8/0.96**	**38.8/0.96**	**36.7/0.91**

in size. Following [8], we train our models using 4 consecutive frames as input and the 5th as target at the training phase. As the scenes from this dataset contain many frames having static background, we train using random 64×64 (patches) exhibiting motion, estimated using the $L2$ distance between frames in consecutive time-steps. Apart from the UCF-101 frames, Sports-1M [26] is also used for reasons similar to the experimental settings described in [8]. We sampled one in every 10 videos of the test split of UCF-101 for the evaluation phase and identified regions containing significant motion by calculating optical flow features (using [30]). The architecture is kept the same as that used for predicting frames from KTH and Weizmann dataset. Also, we again use the combination of features from convolutional layers $2, 4, 5$ and 7 of the feature generation stage for calculating LGGL. Notice the substantial increase in the PSNR/SSIM values of the PixGraph based proposed model over the basic one in table 2. Similar to the results obtained in case of KTH and Weizmann datasets, the trend of a slower rate of degradation (measured using PSNR and SSIM) of the output (produced) frames is also evident in the UCF-101 ablation studies due to the efficient context aggregation power of PixGraph. Qualitative results are shown in figures 3 and 4 of the supplementary document and also in figure 5. Inclusion of residual blocks in the reconstruction stage generator plays a more involved role in UCF-101 predictions than in KTH and Weizmann datasets.

5.3 Results on KITTI

KITTI is a traffic scene database containing videos captured from a camera mounted on cars. In contrast to KTH, Weizmann and UCF-101, the videos from this dataset do not generally suffer from the issue of static background. Hence, we did not use any auxiliary dataset such as Sports-1M for training. For all our experimental studies, videos from (a) road and (b) city have been used. The models were trained using four consecutive frames as input and the subsequent four as target. For quantitative evaluation, we selected one among every five videos from the above mentioned classes to predict 8 frames in the future. Table 2 provides the quantitative comparison of several of our models for KITTI. Interesting to note that, the bare model with only $L1$ objective and the adversarial losses fail to reconstruct realistic versions of KITTI scenes. This is due

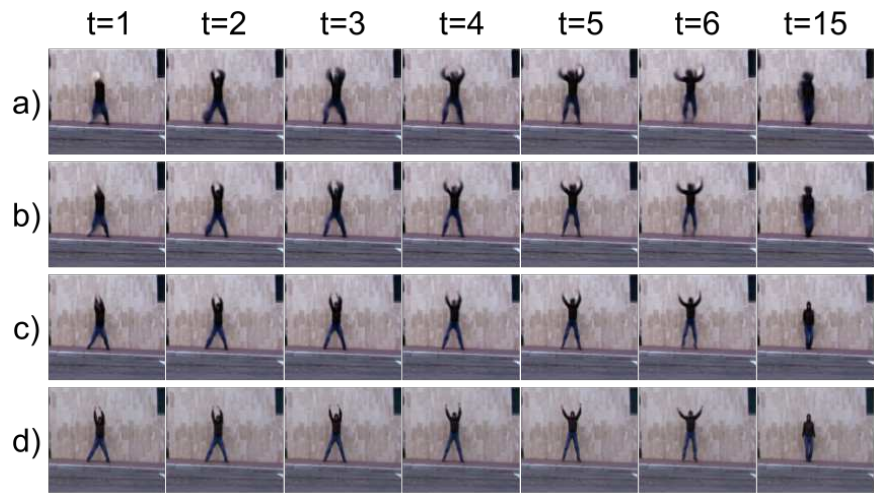


Fig. 4: Qualitative result for Weizmann dataset. The combination of models used are a) L1 + LGGL , b) L1 + LGGL (PixGraph) and c) GDL [8] + L1 + LGGL and d) GDL [8] + L1 + LGGL (PixGraph). The last row represents ground-truth frames

to the fact that the model seems to reconstruct by averaging between every two consecutive frames producing ghost image type artifacts. This issue is almost entirely subdued by the use of PixGraph and LGGL (see table 2), thereby confirming once again its role in guiding the feature generation network into building a rich context aggregated intermediate feature space.

5.4 Cross-dataset evaluation

Apart from the ablation studies discussed above, we also tested our model for generalization using cross-dataset evaluations. For this, we chose three different combinations of datasets (listed as training $\rightarrow$ testing): (a) UCF-101 $\rightarrow$ KTH, (b) UCF-101 $\rightarrow$ Weizmann and (c) UCF-101 $\rightarrow$ KITTI.

UCF-101 $\rightarrow$ KTH and Weizmann Although KTH and Weizmann are simpler in nature than UCF-101 and KITTI, the scenes in these datasets are quite different in style, color and motion pattern. In spite of these inherent differences, our proposed models when trained with UCF-101 for predicting frames, performed remarkably well when tested with KTH or Weizmann videos. The model without any of our proposed objectives failed miserably to reconstruct the frames. Inclusion of LGGL greatly helps to generate legible scenes whereas, combination of all the proposed objectives successfully produce near photo-realistic images. The quantitative results for UCF-101 $\rightarrow$ KTH and UCF-101 $\rightarrow$ Weizmann are shown in table 3. As both KTH and Weizmann have a large static background, the color patterns were easy to predict (simple pixel copy). From the values in table 3 (last two rows), it can be observed that the residual version

Table 2: Quantitative comparison of performance of different methods for the UCF-101 and KITTI datasets using PSNR/SSIM scores. (*) indicates that models are fine tuned on patches of size 64×64 [8]. (†) represents model trained on 2 frames as input. (-) denotes unavailability of performance results. GDL stands for Gradient Divergence Loss [8]. Last 5 rows report the scores obtained using the proposed method. Best results are given in bold.

	Frame-1		**Frame-2**	
Methods	**UCF-101**	**KITTI**	**UCF-101**	**KITTI**
GDL L1* [8]	29.9/0.90	-	26.4/0.87	-
Adv + GDL fine-tuned* [8]	32.0/0.92	-	28.9/0.89	-
Optical flow [8]	31.6/0.93	-	28.2/0.90	-
ConvLSTM [9]	27.3/0.87	-	23.3/0.78	-
ConvLSTM + RES [9]	29.8/0.90	-	25.1/0.82	-
MCNet [9]	27.9/0.87	-	23.8/0.80	-
MCNet + RES UCF101 [9]	30.5/0.91	-	27.7/0.86	-
Deep Voxel Flow† [29]	35.8/0.96	-	-	-
SNCCL + PCDL + L1 [19]	38.2/0.95	40.2/0.94	36.8/0.93	37.7/0.91
Adv + L1 (w/o PixGraph)	24.2/0.82	28.3/0.81	23.5/0.81	26.8/0.80
Adv + L1 + LGGL (w/o PixGraph)	27.5/0.86	30.1/0.86	26.7/0.84	28.1/0.83
Adv + L1 (PixGraph)	34.2/0.93	36.3/0.92	33.5/0.92	35.9/0.91
Adv + L1 + LGGL (PixGraph)	38.8/0.95	41.2/0.95	37.6/0.94	39.3/0.93
Adv + L1 + LGGL + GDL (PixGraph)	**40.1/0.96**	**42.3/0.96**	**39.2/0.95**	**40.1/0.94**

performs significantly better than the other models. This is evident in the qualitative results also (see figure 1 in the supplementary) as it produces sharp edges and maintains the overall texture and minor details (*e.g.* details of the dresses) quite successfully.

UCF-101 → KITTI We also tested our models trained for UCF-101 on the KITTI dataset by feeding 6 input frames to produce 8 future frames. The PSNR/SSIM values in table 4 indicate the generalizability of our proposed models in this cross-dataset arrangement. Note that, in this particular case, the PSNR values decrease (with increasing frame number) a bit faster compared to that for other databases, while the trend of slow rate in decrease of the SSIM value remains unaltered. As PSNR is a pixel-wise difference measure, small changes in a large number of pixels result in a far worse value de-

Table 3: Comparison of UCF-101 → KTH and Weizmann cross-dataset performance using PSNR/SSIM measures for different adversarial models. Best results in bold.

	Frame-1		**Frame-4**		**Frame-10**	
Methods	**Weizmann**	**KTH**	**Weizmann**	**KTH**	**Weizmann**	**KTH**
L1 + LGGL (w/o Pix-Graph)	28.7/0.84	28.2/0.85	23.4/0.78	22.3/0.72	19.8/0.73	19.6/0.70
L1 + LGGL (PixGraph)	**40.2/0.96**	**38.2/0.95**	**36.4/0.94**	**33.9/0.90**	**31.2/0.90**	**28.9/0.84**

Fig. 5: Qualitative results for UCF-101 on a scene involving high amount of camera motion. The combination of models used are a) L1, b) NCCL + PCDL + L1 [19], c) Deep Voxel Flow [29] and d) LGGL + L1 (PixGraph). The last row represents GT frames

Table 4: Comparison of UCF-101 $\rightarrow$ KITTI cross-dataset performance using PSNR/SSIM measures for different adversarial models. Best results in bold.

Methods	**Frame-1**	**Frame-2**	**Frame-4**	**Frame-8**
L1 + LGGL (w/o PixGraph)	29.2/0.85	26.9/0.82	24.7/0.80	20.8/0.78
L1 + LGGL (PixGraph)	**38.6/0.94**	**37.5/0.91**	**34.2/0.86**	**31.1/0.83**

spite being perceptually acceptable. As SSIM uses features for measuring the similarity, it does not get affected by this undesired phenomenon. Figure 2 in the supplementary shows qualitative results of this cross-dataset experimentation.

6 Conclusion

This paper proposes a 2-stage encoder-decoder type GAN for predicting photo-realistic future frames with a novel graph based context aggregation layer, PixGraph. Further, for diminishing the issues of instability and building a meaningful intricate intermediate feature space, we employed a novel region based guidance objective: the Locally Guided Gram Loss (LGGL). Extensive evaluation on popular benchmark datasets and KITTI, a database previously not quite explored in the genre of frame prediction, reveal the superiority of our proposed model, especially the PixGraph module, over the recent state-of-the-art methods.

References

1. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using LSTMs. In: International Conference on Machine Learning. (2015) 843–852
2. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8) (1997) 1735–1780
3. Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604 (2014)
4. Oh, J., Guo, X., Lee, H., Lewis, R.L., Singh, S.: Action-conditional video prediction using deep networks in atari games. In: Advances in Neural Information Processing Systems. (2015) 2863–2871
5. Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. In: Advances In Neural Information Processing Systems. (2016) 613–621
6. Lan, T., Chen, T.C., Savarese, S.: A hierarchical representation for future action prediction. In: European Conference on Computer Vision, Springer (2014) 689–704
7. Walker, J., Gupta, A., Hebert, M.: Patch to the future: Unsupervised visual prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 3302–3309
8. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. International Conference on Learning Representations (ICLR) (2016)
9. Villegas, R., Yang, J., Hong, S., Lin, X., Lee, H.: Decomposing motion and content for natural video sequence prediction. ICLR **1**(2) (2017)
10. Oord, A.v.d., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759 (2016)
11. Kalchbrenner, N., Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., Kavukcuoglu, K.: Video pixel networks. In: International Conference on Machine Learning. (2017) 1771–1779
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
13. Xue, T., Wu, J., Bouman, K., Freeman, B.: Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In: Advances in Neural Information Processing Systems. (2016) 91–99
14. Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R.H., Levine, S.: Stochastic variational video prediction. arXiv preprint arXiv:1710.11252 (2017)
15. Vondrick, C., Torralba, A.: Generating the future with adversarial transformers. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
16. Lu, C., Hirsch, M., Schölkopf, B.: Flexible spatio-temporal networks for video prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 6523–6531
17. Zhou, Y., Berg, T.L.: Learning temporal transformations from time-lapse videos. In: European Conference on Computer Vision, Springer (2016) 262–277
18. Liang, X., Lee, L., Dai, W., Xing, E.P.: Dual motion gan for future-flow embedded video prediction. arXiv preprint (2017)
19. Bhattacharjee, P., Das, S.: Temporal coherency based criteria for predicting video frames using deep multi-stage generative adversarial networks. In: Advances in Neural Information Processing Systems. (2017) 4268–4277
20. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: Proceedings of the 17th IEEE International Conference on Pattern Recognition, 2004. Volume 3. (2004) 32–36

21. Blank, M., Gorelick, L., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. In: Tenth IEEE International Conference on Computer Vision, 2005. Volume 2. (2005) 1395–1402
22. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
23. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11) (2013) 1231–1237
24. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. (2014) 2672–2680
25. Elman, J.L.: Finding structure in time. Cognitive science **14**(2) (1990) 179–211
26. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: IEEE International Conference on Computer Vision and Pattern Recognition. (2014)
27. Bovik, A.C.: The Essential Guide to Video Processing. Academic Press, 2nd edition (2009)
28. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing (TIP) **13**(4) (2004) 600–612
29. Liu, Z., Yeh, R., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: International Conference on Computer Vision (ICCV). Volume 2. (2017)
30. Brox, T., Bregler, C., Malik, J.: Large displacement optical flow. In: IEEE Conference on Computer Vision and Pattern Recognition. (2009) 41–48