Flexible Spatio-Temporal Networks for Video Prediction

Chaochao Lu [1,2] Michael Hirsch [2] Bernhard Schölkopf [2]
[1] University of Cambridge [2] Max Planck Institute for Intelligent Systems
{clu, mhirsch, bs}@tue.mpg.de

Abstract

We describe a modular framework for video frame prediction. We refer to it as a Flexible Spatio-Temporal Network (FSTN) as it allows the extrapolation of a video sequence as well as the estimation of synthetic frames lying in between observed frames and thus the generation of slow-motion videos. By devising a customized objective function comprising decoding, encoding, and adversarial losses, we are able to mitigate the common problem of blurry predictions, managing to retain high frequency information even for relatively distant future predictions. We propose and analyse different training strategies to optimize our model. Extensive experiments on several challenging public datasets demonstrate both the versatility and validity of our model.

1. Introduction

Videos contain rich spatial and temporal structure and capture non-trivial dependencies between objects and contextual information along with scene characteristics like depth, occlusion, and illumination. Accurate modelling of videos ultimately requires high-level understanding of 3D spatio-temporal information, which resembles humans' ability to understand their surrounding physical world. One fundamental problem in video modelling is to predict future frames involving the construction of an internal representation that models, to some degree, both video content and dynamics. Even for short-term future frames, video prediction has remained a challenging problem until now, owing to the complexity and ambiguity inherent in video data. Video prediction is thus still in its infancy, in particular the long-term prediction of video sequences.

There are a number of recent works that predict unseen future video frames [20, 19, 25, 16]. Predicting future images from a video sequence requires the learning of an internal representation that captures spatio-temporal correlations and models the image evolution accurately. This might include information about how objects move, deform or behave, about occlusion and object boundaries, scene depth and so on. Video prediction is a promising research direction that will spawn a wealth of stimulating applications [1, 14, 17, 13, 30, 29, 27, 5, 7, 28, 10, 23, 18].

Key to the success of previous methods is the use of Long Short-Term Memory (LSTM) autoencoder networks which are able to capture spatio-temporal long-range dependencies. Although these works have shown encouraging results for short-time prediction, a common shortcoming is the increasing amount of blur for predictions more than a single frame ahead.

In the present work we propose a new computational model for video prediction that does not suffer from this nuisance. It also allows for the prediction of an entire image sequence rather than just a single frame ahead. Furthermore, it is not only able to extrapolate in time but is equally suited for temporal interpolation between subsequent frames. Our network can be trained end-to-end on full frame images. Once trained, it is able to make predictions in almost real-time.

In particular, we describe two new network modules, for extrapolation and interpolation, each of which predicts a single video frame. These modules can be concatenated to enable the prediction of an entire video sequence with arbitrary many frames. Since each module is fully differentiable they allow for end-to-end training even when combined. Inspired by recent work of Pătrăucean et al. [19], each of the modules comprises a spatio-temporal video autoencoder consisting of a convolutional image encoder-decoder with a nested memory module composed of convolutional LSTM (ConvLSTM) cells. The ConvLSTM features a modified Spatial Transformer Network (STN) layer [9] to capture temporal changes and motion across time by optical flow estimation and prediction.

Our approach also integrates a recently proposed type of loss functions named deep perceptual similarity metric (DeePSiM) [4] that has been shown to better reflect the perceptual similarity of images. It measures distances between image features extracted by deep neural networks and employs the adversarial network of Goodfellow et al. [6]. In our context, the extrapolation and interpolation modules take the role of the generative network, while a discrimina-

tive network is trained jointly to guide unsupervised training.

While our approach leads to significantly improved results, the training procedure and optimization of model parameters is non-trivial. We devise and discuss different training strategies of our combined system and demonstrate its merits in a comprehensive comparison with the state-of-the-art. Our contributions are summarized as follows:

- We present a versatile and flexible framework for video extrapolation and interpolation.
- We devise a novel objective function that comprises decoding, encoding, and adversarial losses, and analyse their effects and contributions.
- We propose different optimization strategies and discuss them in detail.
- We perform a comprehensive comparison of recent state-of-the-art video prediction methods, showing that our approach has an advantage in terms of long-term video prediction.

2. Related Work

A number of recent works use neural networks for video prediction. Ranzato et al. [20] propose a recurrent convolutional network architecture inspired from language modelling. It performs prediction of future video frames and interpolation of intermediate frames based on visual words obtained by clustering image patches. Srivastava et al. [25] adapt a LSTM model [8] and use an autoencoder that jointly reconstructs the input sequence and predicts the unseen future frames. While [25] used a fully connected LSTM layer, Shi et al. [22] use a convolutional LSTM and apply their model for precipitation nowcasting. To counter the problem of inherently blurry predictions common to previous methods, Mathieu et al. [16] propose a multi-scale architecture in combination with a revised loss function that also accounts for the difference of the respective gradient images. In addition they employ a generative adversarial training method [6, 3] for next frame prediction. Concurrently to [16], Pătrăucean et al. [19] describe a spatio-temporal video autoencoder with a nested differentiable short-term memory module that features a modified Spatial Transformer Network Layer [9] for improved motion estimation and prediction. Kalchbrenner et al. [11] propose a generative video model estimating the discrete joint distribution of the pixel values in the video. Different from the tasks above, Bhattacharyya et al. [1] describe an approach for predicting future boundary frames of segmented videos.

Several of the above mentioned works [20, 25, 16] have stressed the importance and difficulty of designing an appropriate loss function in order to render predicted frames less blurry and more realistic. In a recent work, Dosovitskiy et al. [4] propose a new class of loss functions named deep perceptual similarity metrics (DeePSiM) which measures similarity on learned features. Combined with a generative adversarial training the authors show significantly improved results in a number of applications that involve the automated generation of static images.

Our work combines a number of recent insights in a unified framework: it builds upon the convolutional LSTM autoencoder framework with improved motion capture capabilities of [19], and proposes a novel loss function that utilizes the findings of [6, 16, 4]. In contrast to [19] that can only predict one future frame, our model can predict longer-term video sequences as well as interpolate between two frames.

3. Model Description

3.1. Architecture

The building blocks of our model are the basic units recently proposed in [19]: encoder (**E**), decoder (**D**), ConvLSTM unit, optical flow module, grid generator, and sampler (**S**). **E** and **D** constitute a spatial autoencoder, where **E** contains a convolutional layer, a non-linearity layer and a spatial max-pooling layer, while **D** consists of a nearest-neighbour spatial upsampling layer and a convolutional layer. The ConvLSTM unit is a special LSTM unit with the replacement of biased linear (fully-connected) transformations by spatial local convolutions. The optical flow module generates a dense transformation map with the same size as the memory output of ConvLSTM, one 2D flow vector per pixel, representing the displacement in horizontal and vertical directions due to motion between consecutive frames. It integrates a smoothness penalty to ensure that nearby pixels follow a locally smooth motion and hence is capable of capturing pixel-wise motion across consecutive frames. The grid generator and **S** are a modified version of the Spatial Transformer Network (STN) [9], where they accept one transformation per pixel, instead of a single transformation for the entire image as originally proposed in [9]. Since we focus on learning features for motion prediction, they provide immediate feedback on the flow map predicted by the optical flow prediction module.

Different from [19], we use these basic units to assemble two new modules: an extrapolation module and an interpolation module, as shown in Fig. 1 (a) and 2.[1] Each of these can be viewed as a spatio-temporal video autoencoder consisting of a convolutional image encoder-decoder with a nested memory module composed of convolutional LSTM (ConvLSTM) cells acting as a temporal encoder. Also, we incorporate an adversarial network (**A**), that takes all generated frames at the same time and hence guides the training from a global perspective.

[1] For simplicity, the optical flow module and grid generator are included in a ConvLSTM unit in the figures.

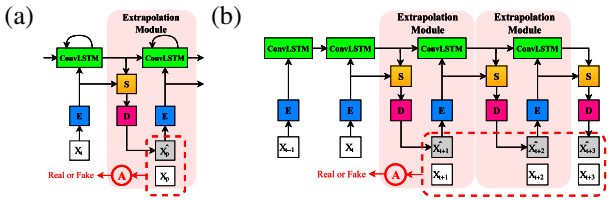

Figure 1. Proposed Extrapolation Model. E: Encoder, D: Decoder, S: Sampler, A: Adversarial Network. (a) Extrapolation model; (b) Unfolding of the extrapolation model with two recurrent steps.

Each of the proposed modules takes two inputs at a time t: the predicted optical flow of ConvLSTM and the features of the encoder unit at time $t-1$. Similarly, their two outputs correspond to the outputs of the ConvLSTM and encoder units at the current time t respectively. The only difference between the extrapolation and interpolation modules is whether they contain ground-truths for the predicted frames or not.

Extrapolation Model. Our model for video extrapolation consists of two ConvLSTMs - the input ConvLSTM and the extrapolation module as shown in Fig. 1 (a). In Fig. 1 (b) we provide a simple example by unfolding the two recurrent sequences over several time steps, where $\{\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t+2}, \boldsymbol{X}_{t+3}\}$ is a sequence of ground truth frames to be predicted from the input frames $\{\boldsymbol{X}_{t-1}, \boldsymbol{X}_t\}$ in a video sequence, and $\{\hat{\boldsymbol{X}}_{t+1}, \hat{\boldsymbol{X}}_{t+2}, \hat{\boldsymbol{X}}_{t+3}\}$ is a sequence of frames predicted by our model. The ground truth frames constitute the supervised information for each layer to guide the training of our model.

Interpolation Model. Our FSTN model can also be used for video interpolation. As shown in Figure 2, given two input frames $\{\boldsymbol{X}_{t-1}, \boldsymbol{X}_t\}$ and one output ground truth frame $\{\boldsymbol{X}_{t+1}\}$, we can interpolate q frames between any two consecutive frames (e.g., $\boldsymbol{X}_{t-1}$ and $\boldsymbol{X}_t$, $\boldsymbol{X}_t$ and $\boldsymbol{X}_{t+1}$) by inserting q interpolation modules between them. Unlike the extrapolation module, the interpolation module does not need the ground truth frames. The FSTN interpolation model displayed on the left in Fig. 2 is the simplest one, but one can extend it to more complex cases as shown in the right half of Fig. 2. In this case, in addition to predicting p future frames one also wants to generate q frames between any two consecutive predicted future frames. This can be readily achieved by stacking p interpolation & extrapolation modules (boxed in black in Fig. 2), where each interpolation module has q time steps.

3.2. Loss Function

In this section, we describe three distinct types of loss functions used in our model. Since our proposed interpolation and extrapolation models use the same loss function for training, for simplicity, here we only focus on the extrapolation model. Let q be the number of concatenated extrapolation modules, $\boldsymbol{X}_{<t}$ be several input frames before time

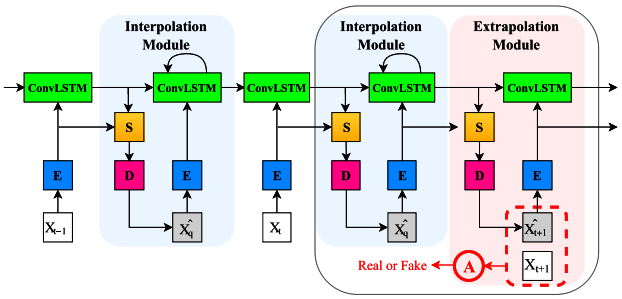

Figure 2. Proposed Interpolation Model. E: Encoder, D: Decoder, S: Sampler, A: Adversarial Network.

t, $\boldsymbol{X}_{t+i}$ and $\hat{\boldsymbol{X}}_{t+i}$ be the ground truth frame and the predicted frame at time step $t+i$ ($i = 1, \ldots, q$), respectively. We further assume that $\boldsymbol{E}(\boldsymbol{X}_{t+i})$ denotes the output of the encoder unit at time step $t+i$, where $\boldsymbol{X}_{t+i}$ is its input.

Decoder Loss. We call the loss function over the output of the decoder unit $\hat{\boldsymbol{X}}_{t+i}$ at time step step $t+i$ *decoder loss*, denoted by $\mathcal{L}^D_{t+i}$. To enforce edge preservation and ensure local smoothness, we adopt the loss function from [19]. Additionally, since our basic goal is to make $\hat{\boldsymbol{X}}_{t+i}$ close to $\boldsymbol{X}_{t+i}$, the l_2 loss between them is also included. Therefore, at time step $t+i$, we have the decoder loss $\mathcal{L}^D_{t+i} = \omega_D \|\hat{\boldsymbol{X}}_{t+i} - \boldsymbol{X}_{t+i}\|_2^2 + \omega_G \|\bigtriangledown\hat{\boldsymbol{X}}_{t+i} - \bigtriangledown\boldsymbol{X}_{t+i}\|^2 + \omega_H \mathcal{H}(\bigtriangledown\mathcal{T}_{t+i})$, where $\mathcal{H}(\cdot)$ is the Huber loss, $\bigtriangledown\mathcal{T}$ is the local gradient of the flow map, and $\omega_D, \omega_G, \omega_H$ are the parameters weighting the data term, decoder loss, and smoothness constraint.

Encoder Loss. Following [4] we include a loss that measures similarity between images by computing the difference of their corresponding features extracted by the encoder network. We thus define the *encoder loss* $\mathcal{L}^E_{t+i}$ at time step $t+i$ as $\mathcal{L}^E_{t+i} = \|\boldsymbol{E}(\hat{\boldsymbol{X}}_{t+i}) - \boldsymbol{E}(\boldsymbol{X}_{t+i})\|^2$.

Adversarial Loss. Adversarial networks consist of a discriminative network D and a generative network G, where D learns to determine whether a frame is from the dataset or produced by G. These two networks are simultaneously trained, thus improving until D can no longer discriminate the frames generated by G. In theory, when the training is done, the generative network G can perfectly approximate the generation process of real data.

In our work, the extrapolation model can be regarded as the generative network G, generating samples $\hat{\boldsymbol{X}}_{t+i}$. Using the corresponding ground truth frame $\boldsymbol{X}_{t+i}$, we define the adversarial loss on D as

$$\begin{aligned}\mathcal{L}^A_D &= \sum_i \log\left(D\left(\boldsymbol{X}_{t+i}\right)\right) + \log\left(1 - D\left(G\left(\boldsymbol{X}_{<t}\right)\right)\right) \\ &= \sum_i \log\left(D\left(\boldsymbol{X}_{t+i}\right)\right) + \log\left(1 - D\left(\hat{\boldsymbol{X}}_{t+i}\right)\right), \\ \mathcal{L}^A_G &= \sum_i \log\left(D\left(G\left(\boldsymbol{X}_{<t}\right)\right)\right) = \sum_i \log\left(D\left(\hat{\boldsymbol{X}}_{t+i}\right)\right).\end{aligned}$$

More specifically, considering the extrapolation module as a generator, we view its outputs and ground truth frames, denoted by $\{\hat{\boldsymbol{X}}_{t+1}, \hat{\boldsymbol{X}}_{t+2}, \hat{\boldsymbol{X}}_{t+3}\}$ and $\{\boldsymbol{X}_{t+1}, \boldsymbol{X}_{t+2}, \boldsymbol{X}_{t+3}\}$ respectively, as the fake and real inputs of a discriminator with labels $\{0, 1\}$ for the adversarial training.

Loss Analysis. In our model we use all the above-mentioned loss functions, as they describe complementary types of information: the decoder loss captures the information loss in image space, encouraging the predicted frame to be as close to the ground truth as possible at each time step. The encoder loss, on the other hand, focuses on the feature space, and compares images in the lower-dimensional representation space. The adversarial loss, finally, optimizes the model from a global perspective, because it takes as input all the frames at all time steps. The adversarial loss thus takes into consideration the distribution difference between predicted frames and ground truth frames, and hence facilitates the global optimization of the whole model.

4. Training

In the training phase, we tested three different strategies to optimize our model. Since the adversarial loss considers all predicted frames simultaneously, it is used in the same way in all three strategies. We thus elaborate only on the decoder and encoder loss.

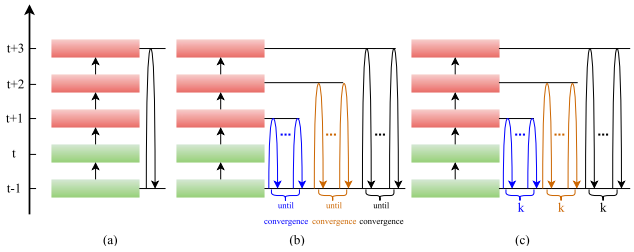

Figure 3. Different training strategies. (a) Strategy I: Globally-First Training; (b) Strategy II: Convergence-First Training; (c) Strategy III: Locally-First Training.

Strategy I: Globally-First Training. A straightforward strategy is the globally-first training as shown in Fig. 3 (a). In each iteration it sums up all the decoder and encoder losses at different time steps and regards the sum as the final objective function, i.e. $\mathcal{L}^{\text{I}}_{\text{FSTN}} = \sum_i \mathcal{L}^D_{t+i} + \mathcal{L}^E_{t+i}$. Given the current loss we make a forward pass and backward pass through the entire model. This strategy gathers all the losses and distributes them to the whole model, which might prevent it from getting stuck in local minima, but at the same time makes the model difficult to optimize.

Strategy II: Convergence-First Training. In contrast, this strategy promotes the other extreme, as shown in Fig. 3 (b). In this case, the model is optimized step by step. Initially, the first sub-model is optimized until it has (almost) converged. Its parameters are used for initializing the second sub-model, and these two sub-models are then optimized jointly, and so on. Formally, at time step $t+1$, we optimize the following objective function until it converges, $\mathcal{L}^{\text{II}}_{\text{FSTN}} = \mathcal{L}^D_{t+1} + \mathcal{L}^E_{t+1}$. After convergence, we continue jointly optimizing the first two sub-models until convergence, $\mathcal{L}^{\text{II}}_{\text{FSTN}} = \sum_{i=1}^{2} \mathcal{L}^D_{t+i} + \mathcal{L}^E_{t+i}$. In this manner, we can optimize the entire model. Compared to Strategy I, this makes the model easier to train as each time it finds a relatively good initial point for the training of the next sub-model. However, since the whole model is not taken into consideration until the very final iterations, this method is prone to local minima.

Strategy III: Locally-First Training. Considering the weaknesses of both strategies above, there is a trade-off between Strategy I and Strategy II. In the third strategy as shown in Fig. 3 (c), each sub-model is only trained k times in each iteration. Therefore, this strategy considers not only sub-models but also the whole model during the training phase. The parameter k needs to be tuned depending on the complexity of data and model.

4.1. Analysis on Training Strategies

To verify the effects of different training strategies, we trained the same model with the above three training strategies on the PV dataset (see Section 5.1), as shown in Fig. 4. Figure 4 (c) shows the effect of the globally-first training strategy, from which we can see that the training loss increases quickly and dramatically rises to a large value. This may be caused by the saturation of parameters resulting from large gradients. On this dataset, it is thus hard for our model to find a good initialization using **Strategy I**. The training curve for **Strategy II** is shown in Fig. 4 (f). The curve has large fluctuation, indicating that it is almost impossible for the model to converge on this complex dataset. A reasonable trade-off is **Strategy III**, training the model step by step, as depicted in Fig. 4 (a), (b), (d), and (e), where the model is trained using $k = 1, 200, 500, 1000$. This increases the chance for the model to be optimized in a good direction (note that Fig. 4 (d) shows an overall downward trend). Note that the periodic structure is caused by the periodic training in Strategy III, where each sub-model is only trained k times in each iteration.

5. Experiments

5.1. Datasets

We performed experiments on three real-world datasets: UCF101 [24], Sports1m [12], and a dataset of video sequences extracted from the PROST [21] and ViSOR [26] sets. Additionally, we show results for the toy dataset of Srivastava et al. [25] consisting of $10,000$ video sequences, each of which is 20 frames long and contains two moving

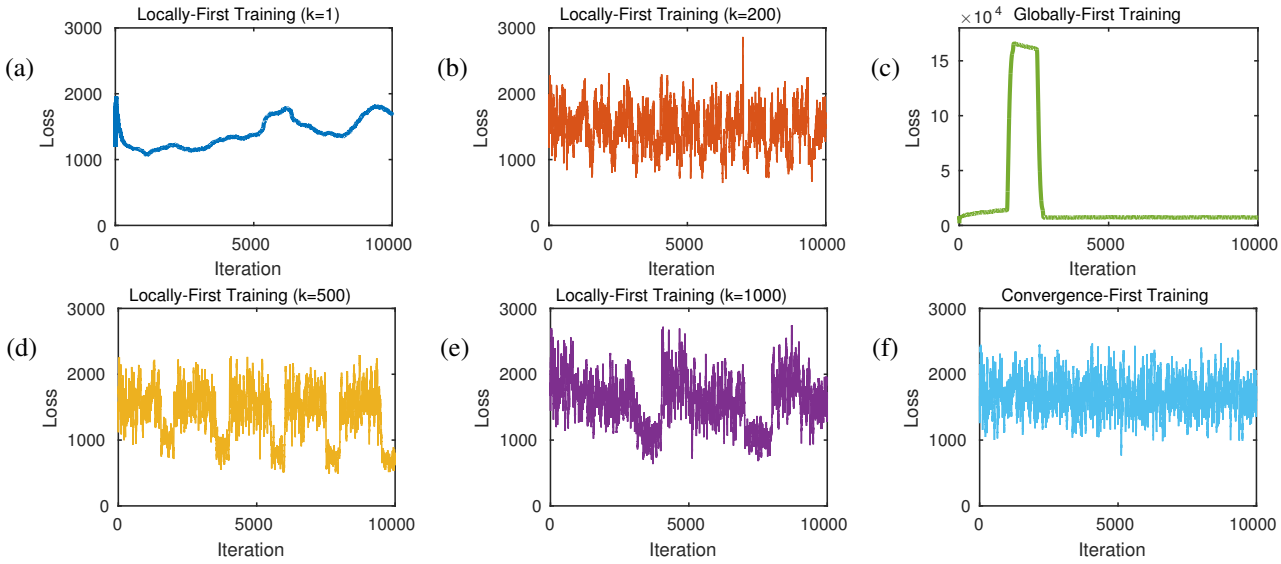

Figure 4. Training loss with different training strategies on the PV dataset.

digits (random direction and velocity) inside a square of size 64×64. The UCF101 dataset contains $13,320$ videos with an average length of 6.2 seconds from 101 different action categories. We provide a quantitative evaluation of our model on this dataset. The Sport1m dataset consists of 1 million YouTube clips, and we used a subset of 600 randomly selected videos for the training of our model. As for the last dataset (PROST and ViSOR), that we will refer to as PV for short, we used it to train our model in order to compare with Pătrăucean et al. [19]. We followed the same setting as described in [19] and selected a subset consisting of 11 video sequences extracted from PROST and ViSOR. For simplicity, all the used images on UCF101, Sports1m, and PV are converted to grayscale and rescaled to the same size of 128×128 pixels.

5.2. Network Architectures and Training

Most of the existing approaches differ in the exact tasks they are trained on. This makes it hard to compare all of them at the same time, and we instead opted for providing individual comparisons. For a fair comparison, we train our model with a similar I/O layout as the respective approach we compare to in Sec. 5.4. Specifically, we use the FSTN extrapolation model with four input frames and four extrapolation modules in our comparison with Pătrăucean et al. [19]; for Srivastava et al. [25], a larger FSTN extrapolation model with 10 input frames and ten extrapolation modules is applied; in the comparisons with Mathieu et al. [16] and Ranzato et al. [20], we use the same model featuring four frames and two extrapolation modules. Note that none of the above models include an interpolation module. Finally, in Section 5.5, where we compare our interpolation module with the one of Ranzato et al., we use our FSTN interpolation model with two input frames and three interpolation modules.

For the adversarial training, the architecture of the discriminative network A is chosen as $4 \times 4 \times 64(2) \longrightarrow 4 \times 4 \times 128(2) \longrightarrow 4 \times 4 \times 128(2) \longrightarrow 4 \times 4 \times 256(2) \longrightarrow 4 \times 4 \times 512(2) \longrightarrow 4 \times 4 \times 1$, where the numbers in each group mean (width, height, number, stride) of the filters, and each convolutional layer is followed by a batch normalization layer. A Leaky ReLU nonlinearity with negative slope 0.2 is used after all layers except for the final layer, where we use a sigmoid function instead.

In the training phase, **Strategy III** is used to train all the above models. Unless stated otherwise, we adopt the setup and parameter setting of [19]. In particular, we use *rmsprop*, with parameters $\epsilon = 10^{-5}$, $\rho = 0.9$, and a decaying learning rate following the rule $\eta = 10^{-4}(\sqrt[100]{1/2})^{\text{epoch}}$. Our implementation uses the Torch library [2] and is based in parts on the implementation of [19].

5.3. Comparisons of Loss Functions

To assess and appreciate the contribution of the different loss functions, we provide a quantitative evaluation similar to [16]. We conducted five comparative experiments, whose results are summarized in Table 1. We report the Peak Signal to Noise Ratio (PSNR) and the sharpness criterion [16] of the first and fourth predicted frame respectively, where in both cases higher scores indicate better results. In the experiments, we trained the FSTN extrapolation model with the same architecture but different combinations of loss functions on the PV dataset. The scores indicate that all the frames predicted by our model have similar PSNR and sharpness values, showing that our model takes all future frames into consideration and is able to preserve image quality and sharpness over time.

Furthermore, Table 1 reveals the complementary nature

Table 1. Comparison of the accuracy of the predictions using different loss functions. The best performance is obtained using the combination of decoder, encoder and adversarial loss. Note, moreover, that image quality (measured by PSNR and Sharpness [16]) does not decay much from the 1^{st} to the 4^{th} frame.

Loss Function	1^{st} frame prediction scores		4^{th} frame prediction scores	
	PSNR	Sharpness	PSNR	Sharpness
$\mathcal{L}^D$	26.4	0.69	25.9	0.62
$\mathcal{L}^E$	18.6	0.47	18.2	0.41
$\mathcal{L}^A$	24.3	0.56	23.9	0.50
$\mathcal{L}^D + \mathcal{L}^E$	27.8	0.89	27.3	0.87
$\mathcal{L}^D + \mathcal{L}^E + \mathcal{L}^A$	30.6	0.99	30.1	0.98

of the different loss functions. Although the encoder loss alone shows poor performance, it is able to improve overall performance when combined with the decoder loss. The adversarial loss is able to boost performance further, and leads to the best PSNR and sharpness values when used in combination with decoder and encoder loss.

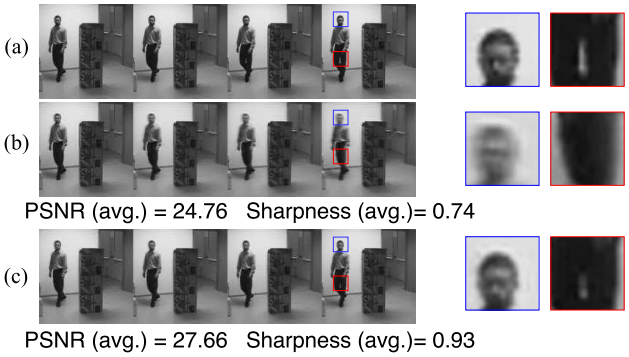

Figure 5. Comparison of results on the PV dataset: (a) Ground truth; (b) Spatio-Temporal Video Autoencoder [19]; (c) our FSTN Extrapolation Model. The magnified details on the r.h.s. show that our face prediction is sharper; also, we capture the opening gap between the legs missed by [19].

5.4. Video Extrapolation

In this section, we compare both our models, the FSTN extrapolation model and the FSTN intrapolation model, with state-of-the-art video prediction approaches.

Comparison to the Spatio-Temporal Video Autoencoder of Pătrăucean et al. [19]

Figure 5 presents a comparison with the spatio-temporal video autoencoder recently proposed by [19]. We use four input frames to predict four future frames. Since the method of [19] can predict only a single frame ahead, we applied their model recursively by using the newly generated frame as an input. As shown in the second row, their predicted frames become increasingly blurry over time, while all four frames predicted by our model (third row) look similarly sharp. In addition, considering the zoom of the image patches in the right side, we can see that more details about the motion are captured by our model. A more comprehensive experiment, which also shows the limitations of our approach, is shown in Fig. 11, where 36 future frames are predicted.

Comparison to the LSTM Approach of Srivastava et al. [25]

Figure 6 compares the results of the LSTM approach of [25] to our model. To this end, we trained our model on the moving MNIST digits.[2] The results in the third row in Fig. 6 are generated by their two layer composite model with a conditional future predictor. Although these are the best results in [25], they do not reach the quality of our predictions (last row), especially for the first three and the last two images.

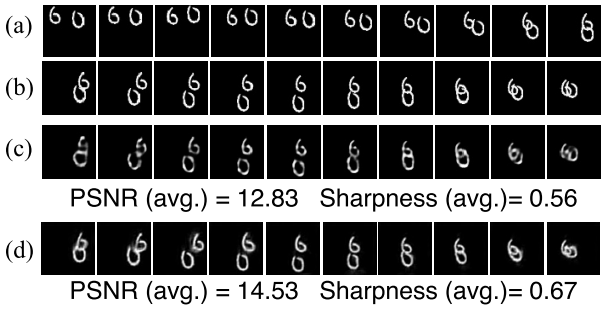

Figure 6. Comparison of results on a moving digits sequence taken from [25]: (a) input frames; (b) ground truth; (c) LSTM approach [25]; (d) our FSTN extrapolation model.

Comparison to the Deep Multi-Scale Video Prediction Approach of Mathieu et al. [16]

To compare our model with [16], we trained our model on the Sports1m dataset and tested it on a video taken from [16]. The results are shown in Fig. 7. The close-ups show that our predicted frames are significantly sharper. More importantly, both of our predicted frames are similarly sharp, while in their results the second frame appears more blurry than the first one.

Comparison to the rNN Approach of Ranzato et al. [20]

In this experiment, we employ the same architecture as above, using four frames to predict two future frames. Figure 8 shows a comparison with the results of [20]. Since the method of [20] operates on image patches, each predicted frame is generated by averaging 64 predicted

[2] http://www.cs.toronto.edu/~emansim/datasets/bouncing_mnist_test.npy

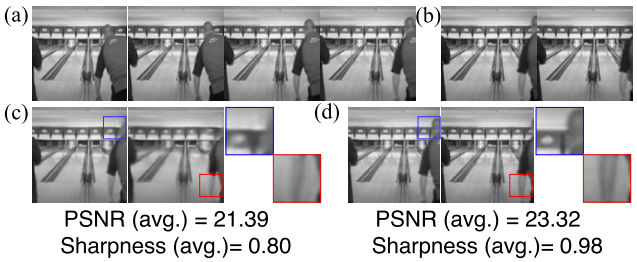

Figure 7. Comparison with Mathieu et al. [16] of results on a video taken from their paper: (a) input frames; (b) ground truth continuation; (c) two frames generated by the deep multi-scale video prediction model of [16]; (d) the same frames generated using our FSTN extrapolation model.

patches. Again our model is able to generate sharper images than the rNN approach of [20].

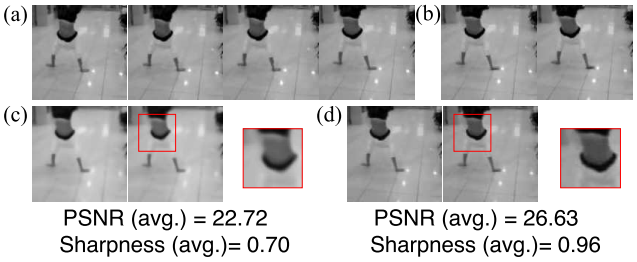

Figure 8. Comparison with Ranzato et al. [20] of results on the Hand Stand Walking UCF101 video clip taken from [20]: (a) input frames; (b) ground truth; (c) rNN approach; [20]; (d) our FSTN extrapolation model.

5.5. Video Interpolation

In this section, we evaluate our model on the task of frame interpolation in videos. Given two frames at two different time steps t and $t+4$, we want to predict three consecutive frames at time steps $t+1$, $t+2$, and $t+3$ between them. As shown in Fig. 9, among all the three frames interpolated by other methods, the middle one is obviously more blurry than its two neighbors, while our model performs significantly better in this aspect. From the closeups we can see that our model can handle and preserve the details well even in the middle frame.

5.6. Long-term Video Prediction

To demonstrate the ability of our model to predict long-term video sequences in both extrapolation and interpolation tasks, we visualize some samples generated by our models (see more video demos in the supplementary material). In Fig. 10, the interpolation model takes as input two frames and interpolates seven consecutive frames between them. Figure 11 shows that our extrapolation model with four input frames is capable of predicting 36 sharp future frames (e.g., the row generated by FSTN30). Compared

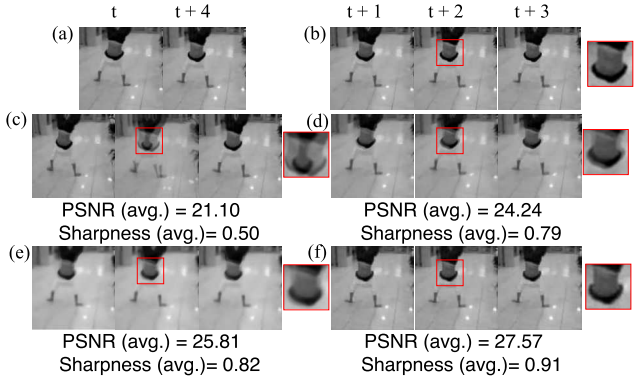

Figure 9. Comparison of results on the Hand Stand Walking UCF101 clip appearing in [20]: (a) input frames; (b) ground truth; (c) Optical Flow [20]; (d) Linear Interpolation; (e) rNN approach [20]; (f) our FSTN interpolation model, producing the visually best result especially for the middle image.

with traditional methods that apply the model recursively by using the newly generated frame as input to extrapolate further in the future, our model clearly has the edge when it comes to long-term prediction.

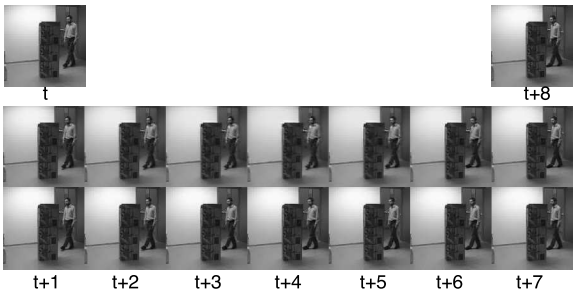

Figure 10. Example of long-term video interpolation. First row: two input frames in the interpolation model; Second row: interpolated frames using FSTN; Third row: ground truth frames. **Best viewed in the form of video in the supplementary material.**

On the left-hand side of Fig. 11, the performance of our models, and in particular of FSTN30, is impressive. Note that this is partly due to the fact that the training setup [19] is such that it does not require generalization across different videos: the training set contains some videos that are similar to those in the test set. From those results, we would thus not claim that the method genuinely learns how to continue novel motion in video. The results do show, however, that the model is able to store complex motions, and reconstruct them from the first four frames, across a dataset of diverse videos. In particular, note that the network does not simply reproduce these videos by recalling explicit templates: it does not store training videos explicitly, but it has to regenerate them based on the first four frames.

To investigate the generalization power of our model to a video that is completely new (i.e., no part of it has been included in the training set), we also test on a separate video

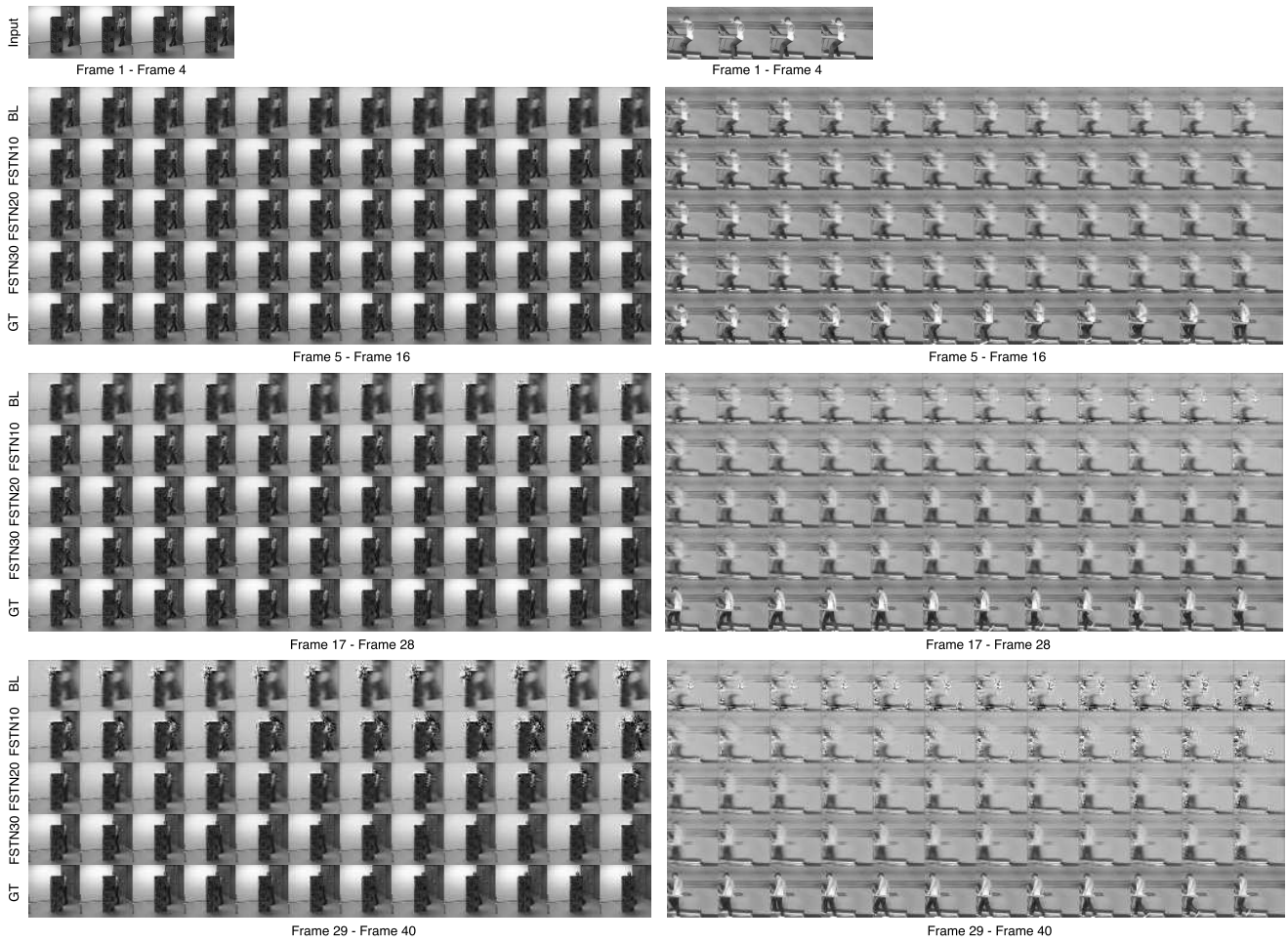

Figure 11. Example of long-term video extrapolation. Left: test result of a video that is similar to training videos; Right: test video is significantly different from training videos (downloaded from YouTube: https://www.youtube.com/watch?v=UiOweJioSD0). BL: baseline ([19]); GT: ground truth; FSTN10, FSTN20, and FSTN30 are all our models, with different numbers of extrapolation modules (e.g., FSTN10 refers to the FSTN model with 10 extrapolation modules.) **Best viewed in the form of video in the supplementary material.**

taken from YouTube, as shown on the right-hand side of Fig. 11. Compared with [19], it is evident that our model can predict aspects of the long-term motion in the video, although the predicted frames are more blurry than they are for the other videos.

6. Conclusion and Future Work

We presented a modular data-driven framework for video extrapolation and interpolation based on an end-to-end differentiable network architecture. We devised a novel objective function and proposed different optimization strategies for model training. Extensive experiments on public datasets illustrated both the validity and versatility of our approach.

Since our model demonstrated its capability to capture complex spatio-temporal dependences and motion patterns, we believe that models pre-trained with our pipeline may also be useful when used in a supervised setup targeting high-level vision task such as object tracking, semantic segmentation or action recognition based on videos. Our model might also find interesting applications in the context of video compression.

In addition to the above applications, our work was motivated by the observation that video prediction may form a basic component of an animal's model of the world and, as pointed out by Yann LeCun [15] [3], it is an elegant way of converting a problem of modelling unlabelled data into a supervised learning task.

[3] http://cilvr.nyu.edu/lib/exe/fetch.php?media=deeplearning:2016:lecun-20160308-unssupervised-learning-nyu.pdf

References

[1] A. Bhattacharyya, M. Malinowski, and M. Fritz. Spatio-temporal image boundary extrapolation. *arXiv preprint arXiv:1605.07363*, 2016. 1, 2

[2] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 5

[3] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a Laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494, 2015. 2

[4] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*, 2016. 1, 2, 3

[5] A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016. 1

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 1, 2

[7] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis. Learning temporal regularity in video sequences. *arXiv preprint arXiv:1604.04574*, 2016. 1

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[9] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2008–2016, 2015. 1, 2

[10] D. Jayaraman and K. Grauman. Slow and steady feature analysis: higher order temporal coherence in video. *arXiv preprint arXiv:1506.04714*, 2015. 1

[11] N. Kalchbrenner, A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016. 2

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 4

[13] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012. 1

[14] B. Klein, L. Wolf, and Y. Afek. A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4840–4848, 2015. 1

[15] Y. LeCun. Unsupervised learning, 2016. 8

[16] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *International Conference on Learning Representations*, 2016. 1, 2, 5, 6, 7

[17] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015. 1

[18] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. *arXiv preprint arXiv:1511.03476*, 2015. 1

[19] V. Pătrăucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309*, 2015. 1, 2, 3, 5, 6, 7, 8

[20] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014. 1, 2, 5, 6, 7

[21] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 723–730. IEEE, 2010. 4

[22] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*, 2015. 2

[23] J. Shin, N. Tajbakhsh, R. Todd Hurst, C. B. Kendall, and J. Liang. Automating carotid intima-media thickness video interpretation with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2526–2535, 2016. 1

[24] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 4

[25] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. *arXiv preprint arXiv:1502.04681*, 2015. 1, 2, 4, 5, 6

[26] R. Vezzani and R. Cucchiara. Video surveillance online repository (ViSOR): an integrated framework. *Multimedia Tools and Applications*, 50(2):359–380, 2010. 4

[27] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. 1

[28] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating the future by watching unlabeled video. *arXiv preprint arXiv:1504.08023*, 2015. 1

[29] J. Walker, A. Gupta, and M. Hebert. Patch to the future: Unsupervised visual prediction. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3302–3309. IEEE, 2014. 1

[30] J. Yuen and A. Torralba. A data-driven approach for event prediction. In *European Conference on Computer Vision*, pages 707–720. Springer, 2010. 1