IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 18, 2019, accepted November 28, 2019, date of publication December 12, 2019,
date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2959019

Video Frame Synthesis via Plug-and-Play Deep Locally Temporal Embedding

ANH-DUC NGUYEN[1], WOOJAE KIM[1], JONGYOO KIM[2], (Member, IEEE), WEISI LIN[3], (Fellow, IEEE), AND SANGHOON LEE[1], (Senior Member, IEEE)

[1]Department of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea
[2]Microsoft Research Asia, Beijing 100080, China
[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798

Corresponding author: Sanghoon Lee (slee@yonsei.ac.kr)

This work was supported by Institute for Information and Communications Technology Promotion through the Korea Government through the Ministry of Science, ICT and Future Planning (MSIP) (Development of mobile GPU hardware for photo-realistic real-time virtual reality) under Grant 2016-0-00204.

ABSTRACT We propose a generative framework that tackles video frame interpolation. Conventionally, optical flow methods can solve the problem, but the perceptual quality depends on the accuracy of flow estimation. Nevertheless, a merit of traditional methods is that they have a remarkable generalization ability. Recently, deep convolutional neural networks (CNNs) have achieved good performance at the price of computation. However, to deploy a CNN, it is necessary to train it with a large-scale dataset beforehand, not to mention the process of fine tuning and adaptation afterwards. Also, despite the sharp motion results, their perceptual quality does not correlate well with their pixel-to-pixel difference metric performance due to various artifacts created by erroneous warping. In this paper, we take the advantages of both conventional and deep-learning models, and tackle the problem from a different perspective. The framework, which we call deep locally temporal embedding (DeepLTE), is powered by a deep CNN and can be used instantly like conventional models. DeepLTE fits an auto-encoding CNN to several consecutive frames and embeds some constraints on the latent representations so that new frames can be generated by interpolating new latent codes. Unlike the current deep learning paradigm which requires training on large datasets, DeepLTE works in a plug-and-play and unsupervised manner, and is able to generate an arbitrary number of frames from multiple given consecutive frames. We demonstrate that, without bells and whistles, DeepLTE outperforms existing state-of-the-art models in terms of the perceptual quality.

INDEX TERMS Frame synthesis, video processing, manifold learning, convolutional neural network, unsupervised learning.

I. INTRODUCTION

With the advances in video-capturing devices, there has been an increasing demand for high-quality videos. Because of this demand eruption, several video applications have been of particular interest, such as image morphing and slow-motion video capturing. However, recording a high-frame-rate video usually involves expensive devices and doing so with hand-held devices is impractical, as the process requires expensive and bulky cameras, large memory storage, and intense power supplies from the devices. Moreover, along with the development of high-refresh-rate displays, end users desire to watch high-quality videos created by upgrading low frame-rate videos captured by older cameras. For these reasons and others, it is natural to ask whether we can enhance the frame rate of existing videos, *i.e.*, interpolate video frames in the middle of existing ones.

Video Frame Interpolation: is among the most long-standing and challenging problems in video processing. This problem is inherently ill-posed because given two consecutive frames, a number of solutions can be valid. Conventionally, optical-flow-based methods are extensively studied to deal with this problem [19], [24], [48], and are often regarded as *Lagrangian* methods because they require solving optimization problems to compute optical flow. On the other hand, another approach called *Eulerian* eliminates the need for flow computation as it characterizes motion over time at each fixed location. Notable studies in this category

The associate editor coordinating the review of this manuscript and approving it for publication was Tony Thomas.

include phase-based methods [27], [45]. Despite such huge efforts, performance is still limited due to various obstacles. Although optical flow estimation is another classic and difficult task itself, phase-based methods are shown to be suitable for small-motion videos only. Nonetheless, a merit of these methods is that they are off-the-shelf models and can be used instantly, and hence always work at their full capacities. We will frequently refer to these bottom-up and non-learning based methods as ''conventional'' models from now on.

Recent years have seen the huge success of convolutional neural networks (CNNs), a variant of neural networks (NNs), in various computer-vision areas. In brief, an NN is a parametric function of composites and superpositions of simple nonlinearities, and it has been proven to be able to approximate any function. A CNN is a modification of an NN in which the dot product is replaced by the convolution operator, which enables the system to work with multi-dimensional arrays directly without flattening them into vectors. The performance of deep-CNN-based methods is superior to that of the conventional algorithms because of their phenomenal approximation ability, but unfortunately, training on a large-scale dataset is usually required beforehand, which is generally time-consuming and infeasible in many situations. Furthermore, to make such networks operate at their fullest potential, adaptation or fine tuning might be required because in training, one may impose several regularizations on the models, which produces biased estimators.

An important concept for many ideas in machine learning is that of a *manifold*. Loosely speaking, a manifold is a connected set of points that can be represented well by only a few dimensions in a high-dimensional space [8]. In machine learning, the manifold hypothesis is that data occur only along a thin manifold, and interesting variations in the data occur only along directions that lie along the manifold. The rationale of this hypothesis is that the probability distribution over images is highly concentrated because a randomly generated image is just as good as noise. In addition, many transformations of an image, such as translation and rotation, result in other images which can be imagined to be in the neighborhood of the original image. Such descriptions fit video frames quite well; that is, most information in a frame is likely to be present in the adjacent frames, and these frames can establish a manifold in a much lower-dimensional space because the variations among frames should be small. Despite such potential, few methods based on manifold learning have been introduced, and these methods only deal with problems which can be regarded as ''toy problems'' according to modern standards [3]. A possible reason is that most manifold learning models require abundant data so that most variations in the data are included, which is not met by videos in daily life. Nevertheless, a merit of the interpolation based on manifold learning is that the synthesis is more direct and intrinsic and is not based on any extrinsic tools that are difficult to acquire such as phase and optical flow.

As a key factor, perceptual image quality is an essential aspect in the image-synthesis field. From the perceptual quality perspective, existing optical-flow-based methods motivate the development of consistent quality over the global spatial region. Fig. 1 depicts examples of the interpolated images in (a)–(c) and its corresponding perceptual error maps in (d)–(f) obtained by our previous image quality assessment (IQA) framework [16] with respect to the reference in (g). In Fig. 1, each of the interpolated images is obtained by an existing state-of-the-art optical-flow-based method [21] in (a), simple averaging of two input frames in (b), and the proposed framework in (c), respectively. In Fig. 1(a), the interpolated frame is sharply synthesized over the global region except for specific local region (see the human hand region). Correspondingly, in the perceptual error map of Fig. 1(d), the dark regions indicate erroneously predicted regions. Because the optical flow model can easily generate prediction errors in local regions with fast motions, it can easily lead to inaccurate image warping, which causes strong local artifacts in Fig. 1(a). On the other hand, if the two input images are simply averaged, as shown in Fig. 1(b), although heavily blurred signals are generated over the fast-motion region, perceptually, the image is still as pleasant as in Fig. 1(a). This point is illustrated in Figs. 1(d) and (e), where errors around the hand region are at a similar level in terms of the gray intensity. Furthermore, distortions causing frequent discontinuities in images are more noticeable [14], [15], thus the perceived quality might be low. Therefore, if we can reduce the blurriness created from averaging motion regions, the perceived quality can be much improved over that of the optical-flow-based methods. So far, none of the plug-and-play alternatives to the optical-flow-based approach have become competitive in practical video benchmarks. However, we argue that the time might be right to revisit and further innovate traditional alternatives to optical-flow-based methods to see whether they can provide

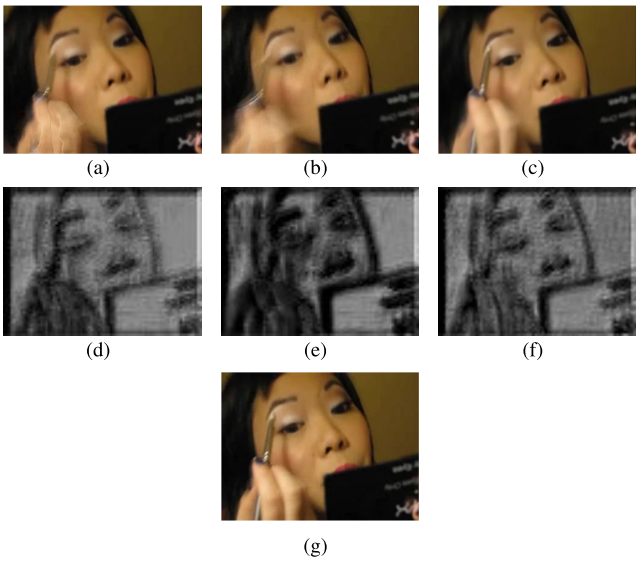

FIGURE 1. (a)–(c) Interpolated frames by the scheme in [21], averaging, and our framework. (d)–(f) Perceptual error maps obtained by the framework in [16] of the synthesized images with respect to reference (g).

a better perceptual synthesis quality, which is lacking in current optical-flow-based approaches.

Combining all the advantages of the discussed approaches, we attempt to solve the problem of frame interpolation intrinsically by introducing a novel framework, dubbed as deep locally temporal embedding (DeepLTE). DeepLTE is built upon the manifold hypothesis that in some latent space, consecutive frames lie very close to each other on a manifold. Based on this hypothesis, we can explicitly impose some constraints so that the true manifold can be approximated by a simple formulation. Thus, new latent codes can be sampled from the estimated manifold and new frames can be produced by decoding the new latent codes. To this end, we resort to the computing power of an auto-encoding CNN. Because the constraints are defined to preserve the temporal relationship of the inputs, the encoder of the CNN can be seen as a temporal embedding, which provides the name of the proposed method. The described framework can be seen as a special case of averaging pixels. Concretely, because convolution works in a sliding fashion, each dimension in the latent space corresponds to a certain patch of the original image. The latent constraint, which is a weighted-average strategy, then averages the latent codes in the latent space, which can be seen as averaging patches of the same spatial locations. Fig. 1(c) demonstrates an image obtained by our method. As can be seen, our result does not show any irritating artifacts like those found in the optical flow method, and is much sharper and more pleasant than simply averaging two input images.

The proposed method is significantly different from most, if not all, existing methods in a number of ways. First, it is based on manifold learning, which is completely different from existing methods based upon optical flow, image phase, or contemporary deep learning. Moreover, even though there are several methods utilizing manifold learning, ours is the first to be applied to natural and daily-life videos consisting of complex motions. Second, unlike previous deep-learning-based studies in frame synthesis, we propose an alternative to the contemporary training–testing paradigm of deep learning. In the contemporary deep learning literature, CNN is generally trained over a large dataset. By contrast, here we use only the video sequence to be interpolated as input without the need of a separate training phase or any external data. Given a set of consecutive video frames, we define several frames as anchors and interpolate the in-between frames corresponding to the given middle frames. We then optimize the CNN's weights by minimizing a reconstruction loss between the reconstructed and given anchors, and another reconstruction loss between the interpolated and given middle frames. After the weights are optimized, we can freely interpolate any middle frames by arbitrarily interpolating the latent codes. Hence, it works in a plug-and-play like conventional methods, which makes it always ready to test without training. Also, the optimization is carried out blindly on the test set without any external supervision signal, hence the method is unsupervised. Accordingly, our method is essentially a bridge that connects conventional and deep-learning-based methods.

Compared to existing methods, DeepLTE enjoys a number of advantages. First, existing deep learning methods have a separate phase for training. Thus, in the testing phase, they must use the network architectures defined in the training step. In contrast, being plug-and-play, the proposed method has the freedom to choose and change its structure based only on the information of an input sequence. To best suit an input sequence, there is a degree of freedom to choose different network architectures for the encoder and decoder, downsampling strategies, cost functions, and so on. Second, powered by a deep auto-encoding CNN, DeepLTE possesses the performance level of deep learning, which has been shown to be superior to conventional methods in image generation. Last but not least, DeepLTE can synthesize an arbitrary number of frames in a run. To the best of our knowledge, no existing method can include all these properties at the same time. DeepLTE is also optimized with a perceptual cost function so the synthesized images have a much better perceptual quality than those produced by existing methods. To summarize all the differences between the proposed method and existing works, we tabulate all the notable features in Table 1 as depicted.

Our work mainly focuses on the following contributions.

1) We propose a new frame-synthesis framework that bridges the gap between conventional and deep-learning-based models. The method is plug-and-play, but achieves a deep-learning-standard performance.
2) Unlike existing methods that mostly employ extrinsic tools, DeepLTE is entirely based on intrinsically manipulating the underlying latent structures of videos, and this methodology has never been applied to real-world videos consisting of complex motions.
3) We discover that deep networks can capture a great deal of video statistics, which may not only be beneficial to the frame-synthesis field but may also fuel advances in other deep unsupervised-learning areas.

The rest of the paper is organized as follows. In Section II, we review some of recent studies. In Section III, we introduce DeepLTE and explore a number of its variants. We present the benchmarks of the proposed algorithm, including comparisons with state-of-the-art methods, ablation studies, and analyses in Section IV. Finally, we conclude the paper in Section V.

II. RELATED WORK

Video frame interpolation is a classic problem in video processing. Traditionally, this is done by estimating motions in consecutive frames [1], [48]. Mahajan *et al.* defined "path," which is similar optical flow, and then used 3D Poisson reconstruction for in-between frame generation [24]. As an alternative to optical-flow-based methods, Meyer *et al.* utilized phase information to interpolate frames, but this method may fail to retain high-frequency details in regions containing large motions [27].

TABLE 1. Comparison of features according to video frame-interpolation method. "v" indicates that a feature is included in the method and "–" specifies the opposition. "End-to-end" implies that the method is designed to produce interpolated frames directly, whereas the opposite case does not include the interpolation step in its design.

Method	Approach	Architecture				Usage	
		Plug-and-play	Deep	End-to-end	Perceptually optimal	Multiple inputs (>2)	Multiple outputs
[31,33]	adaptive convolution	-	v	v	v	-	-
[26,44]	pixel hallucination	-	v	v	-	-	-
[22,23]	optical flow	-	v	v	-	-	-
[13]	optical flow	-	v	v	v	-	v
[7,12,37,46,49]	optical flow	-	v	-	-	-	v
[4,25]	optical flow	v	-	-	-	-	v
[27]	image phase	-	v	v	-	-	-
[28,47]	image phase	v	-	v	-	-	-
[3]	manifold	-	-	v	-	v	-
DeepLTE	manifold	v	v	v	v	v	v

Since the huge success of deep CNNs in image recognition/classification [9], [20], [40], many researchers have proposed estimating optical flow using CNNs [7], [44], [47]. These methods are not optimized to directly synthesize new images; hence, the generated images may contain many artifacts due to inaccurate flows and warping. The recent state-of-the-art models estimate an optical-flow field in an unsupervised manner [12], [21], which enhances the performance significantly but artifacts are still inevitable. In a different direction, the method in [25] is based on a technique called pixel hallucination which directly generates new pixels from scratch. However, the results are not visually pleasing [21]. Finally, a recent technique based on adaptive convolution was proposed in [30], [32]. These methods learn to produce a set of pixel-dependent filters and convolve these filters with the input frames to interpolate each pixel. The results are notable, but the method requires the network to learn a set of filters for each pixel, which is computationally unfriendly. Above all, all these methods require a separate training step on big data and may need fine tuning or adaptation to take full advantage of the networks.

The interpolation method closest to DeepLTE was proposed by Bregler and Omohundro also relies on manifold learning [3]. However, they estimated a nonlinear manifold for the entire series of frames and the video sequence used simply contained talking lips, which is quite trivial according to the current standard. In a remote study [46], an observation that latent variables can encode motions supports the proposed idea, but their latent codes are randomly sampled from a Gaussian distribution, and their work concentrates on predicting the trajectory of motions given a still image; hence the generated images are nowhere near realistic.

The technical inspiration for our work is locally linear embedding (LLE) [37]. This method seeks a linear relationship between neighboring samples in their original space and then finds another space, preferably having lower dimension, where this relationship is preserved. We briefly describe the work in Appendix -A. Applying LLE directly to frame interpolation, however, is infeasible for two reasons. First, like other manifold learning methods, LLE requires abundant data so that the manifold is well-sampled. Unlike facial expression images in [37] or talking lips in [3], real-world videos do not contain any common objects or structures. Given a target frame, only a few neighboring frames are useful for estimating the manifold. Second, the weights of the linear combinations in LLE are obtained from optimization, which does not provide any intuition so that we can sample new latent codes in the neighborhood of a point. Nevertheless, LLE provides us with several key ideas that motivate the proposed framework. First, we do not need to work with the whole video but only several successive frames, and assume that their underlying manifold in some latent space can be easily approximated. Second, if we can encode video frames into this latent space and then map their codes back into image space, then new frames can simply be synthesized in the latent space. In the next section, we depict how DeepLTE is built based on this idea.

III. DEEP LOCALLY TEMPORAL EMBEDDING

A. PRELIMINARIES, NOTATIONS, AND OVERALL DESCRIPTION

We use calligraphy letters (*e.g.*, $\mathcal{X}$) to denote sets, bold uppercase letters (*e.g.*, $\mathbf{X}$) for matrices and tensors, italic uppercase letters (*e.g.*, X) for subtensors, and italic lowercase letters (*e.g.*, x) for column vectors and scalars.

Fig. 2 illustrates the difference between the proposed scheme and existing models. Current trends usually require a separate training stage with a huge amount of data for CNNs. After the networks are trained properly, they can be deployed in testing phase where they predict outputs for unseen data. In contrast, DeepLTE performs both optimization and inference on the fly using testing sequence itself without any training set. Suppose we have a sample $\mathbf{X}$ in some domain $\mathcal{X}$, we jointly optimize an encoder, which maps $\mathbf{X}$ into $\mathbf{Z}$ in a lower-dimensional subspace of $\mathcal{X}$, and a decoder, which reconstructs the corresponding interpolated samples from $\mathbf{Z}$.

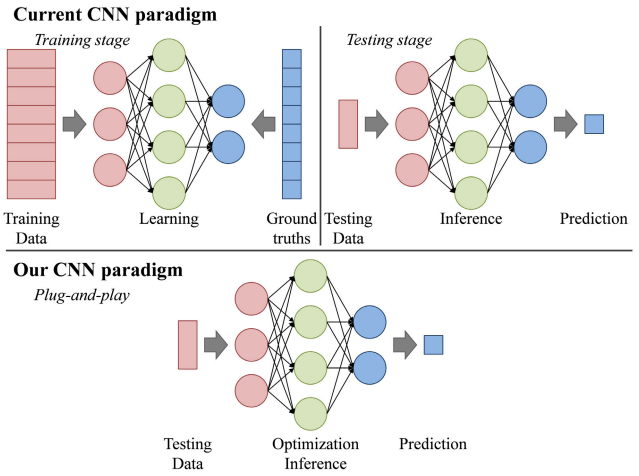

FIGURE 2. Difference between a typical CNN paradigm and DeepLTE. Most studies involving deep networks have a separate training step that requires the networks to learn on a dataset of big data. After the networks are properly trained, they can be deployed in a testing phase in which they perform inference on unseen data. On the other hand, the proposed paradigm does not have a separate training with a big external dataset. Instead, it performs "blind" optimization on testing data, which is called "plug-and-play" in this paper. Even though the plug-and-play paradigm seems to be the same as the testing phase, the essence of it is completely different.

Because the optimization is not based on any development set as done in a typical NN training scheme, the optimization can be run until convergence or terminated at a fixed point. We hypothesize (and show evidence) that the decoder of the optimized network can reconstruct a neighboring point of $\mathbf{Z}$ into a neighboring point of $\mathbf{X}$; *i.e.*, the decoder learns how to decode the neighbors of $\mathbf{Z}$ into the neighbors of $\mathbf{X}$. Therefore, to repurpose the network for generating new data, once the network is optimized, we can sample latent variables from the manifold learned by the network's encoder. In the following sections, we outline how to apply the described scheme to video frame interpolation.

B. DEEP LOCALLY TEMPORAL EMBEDDING FRAMEWORK

Let $\mathbf{I} \in \mathcal{X}^{m+1}$ contain $m+1$ ($m > 1$) consecutive frames $I_0, I_1, \ldots, I_m \in \mathcal{X}$ which are temporally uniform as shown in Fig. 3. We refer to the frames above the time axis as *references*. Furthermore, let $\mathcal{N}$ be a set of indices of n specially selected frames from the $m+1$ frames, which we call *nodes*. In Fig. 3, it is specified that frames I_0 and I_m are selected as nodes to play the role of anchors for frame interpolation. Other reference frames can also be chosen depending on the choice of constraint for the latent space, which is detailed in the next section, but the nodes always include at least the first and last frames of the sequence as indicated in Fig. 3; *i.e.*, indices 0 and m are always in $\mathcal{N}$ ($0, m \in \mathcal{N}$). Unlike previous studies, we do not assume that motions are symmetric over the middle frame or that all frames are stabilized with respect to the starting frame. With the references and nodes as described above, DeepLTE performs an optimization process to construct a manifold

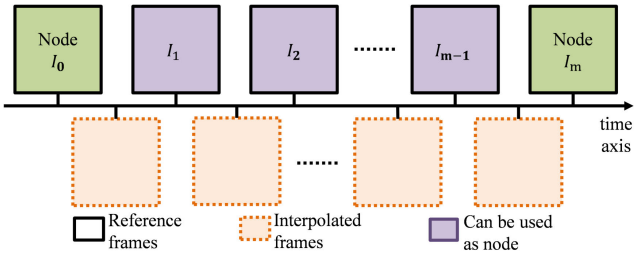

FIGURE 3. Notations of video frames. All frames above the temporal axis are *references*. All frames below the axis are interpolated. Green denotes frames that are *nodes*. Purple frames can be used as nodes depending on the type of constraint used in the latent space.

from the given nodes. After that, the constructed manifold can be used to interpolate new frames between two nodes.

We first describe the optimization process. The overall framework of DeepLTE is shown in Fig. 4. Let $\mathbf{I}_{\mathbf{nodes}} \in \mathcal{X}^n$ accommodate n nodes $I_j, j \in \mathcal{N}$ from the sequence. As can be seen from Fig. 4, at first glance, DeepLTE simply fits a CNN to a set of input images. The encoder $\mathrm{g} : \mathcal{X}^n \to \mathcal{Z}^n$ of DeepLTE first maps $\mathbf{I}_{\mathbf{nodes}}$ into $\mathbf{Z}_{\mathbf{nodes}}$, which contains n latent variables $z_j \in \mathcal{Z}, j \in \mathcal{N}$.

$$\mathbf{Z}_{\mathbf{nodes}} = \mathrm{g}(\mathbf{I}_{\mathbf{nodes}}; \Theta_{\mathbf{g}}), \tag{1}$$

where $\Theta_{\mathbf{g}}$ are trainable parameters of the encoder. Next, we perform interpolation on the latent codes using an LTE module in the latent space. The LTE module embeds a constraint on the latent manifold so that new codes can be easily interpolated and drawn from the manifold. Let us denote this interpolation as $\mathrm{lte} : \mathcal{Z}^n \to \mathcal{Z}^{m+1}$. The interpolation can be expressed as

$$\hat{\mathbf{Z}} = \mathrm{lte}(\mathbf{Z}_{\mathbf{nodes}}), \tag{2}$$

where $\hat{\mathbf{Z}}$ contains $m+1$ interpolated codes $\hat{z}_k$ of the reference frames in its rows. Note that the LTE module is generally defined to be an identity function of the nodes, and hence, $\hat{z}_j = z_j$ for $j \in \mathcal{N}$ if not otherwise specified. Finally, the interpolated codes are decoded back into the image space by the decoder $\mathrm{h} : \mathcal{Z}^{m+1} \to \mathcal{X}^{m+1}$ and we obtain the reconstructed reference images as

$$\hat{\mathbf{I}} = \mathrm{h}(\hat{\mathbf{Z}}; \Theta_{\mathbf{h}}), \tag{3}$$

where $\hat{\mathbf{I}} \in \mathcal{X}^{m+1}$ represents the reconstructed versions of the reference frames, and $\boldsymbol{\Theta}_{\boldsymbol{h}}$ parametrizes the decoder h of DeepLTE. As in a common auto-encoder framework, we define an expected loss $\mathrm{L}(\mathbf{I}, \hat{\mathbf{I}})$ between the reference images and their reconstructions and optimize it for optimal $\boldsymbol{\Theta}_{\boldsymbol{g}}$ and $\boldsymbol{\Theta}_{\boldsymbol{h}}$.

Once the fitting is done, to generate new images between frames I_j and I_{j+1}, we can simply define an interpolation rule to synthesize new codes and use the decoder to transform the interpolated codes into interpolated frames. In the next section, we shed light on how to design an LTE module so that sampling in latent space is simple and intuitive.

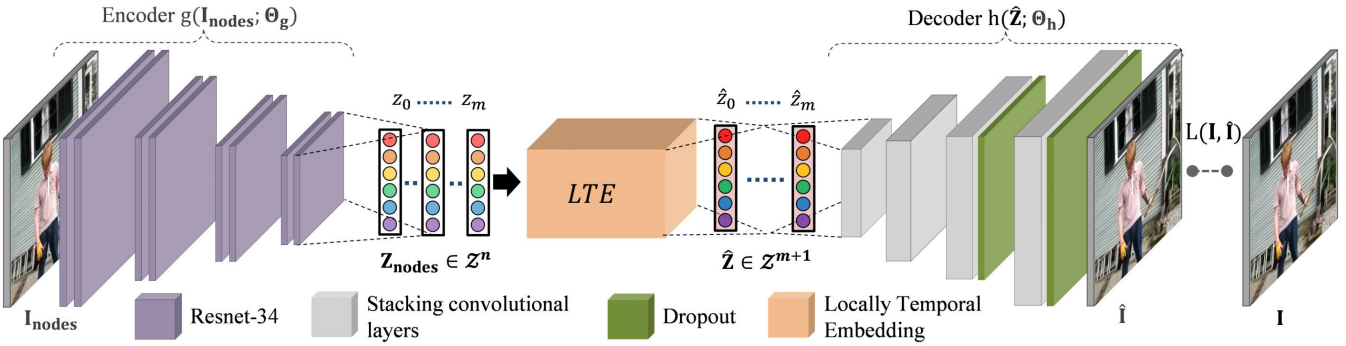

FIGURE 4. The overall framework of DeepLTE. First, the encoder g maps image nodes into a latent space, where they are fed to an LTE module to interpolate the latent representations of non-node images. Then, all the latent codes are decoded by the decoder h to reconstruct the reference images.

C. LATENT CONSTRAINTS

Inspired by the work of Roweis and Saul [37], we develop a simple constraint that establishes a relationship in a locality of points in the latent space. In LLE, the weights are determined through an optimization process. Such complicated weights are not suitable in our method because LLE is designed to reduce data dimension, not to interpolate new latent variables or produce new data. Hence, we do not have any intuition about how to sample new latent codes using the learned weights. Instead, because nearby frames are close to each other in the image space, it is reasonable to assume that there exists a latent space such that the latent representations of these frames are also close to each other. Using this assumption, we can approximate the locality of these points by a polynomial manifold in the latent space. Specifically, out of many possible choices, we choose Lagrange's polynomial approximation for its simplicity. Then, the interpolation proceeds as follows. Suppose the s^{th}-order polynomial is chosen ($s < m$). Denote $\mathcal{N} = \{j | z_j \in \mathbf{Z}_{nodes}\}$ as the set containing 0, m, and the indices of $s-1$ latent nodes z_j for some $j \in (0, m)$ ($n = s+1$). Lagrange's formula for the interpolation polynomial can be defined as

$$\hat{z}_t = \sum_{i \in \mathcal{N}} z_i l_i(t), \tag{4}$$

where $\hat{z}_t$ is an interpolated point depending on the position $t \in \mathcal{R}, 0 < t < m$ by using the s^{th}-order Lagrange polynomial, and $l_i(t)$ is defined as

$$l_i(t) = \prod_{j \in \mathcal{N}, j \neq i} \frac{t-j}{i-j}. \tag{5}$$

Fig. 5(a) shows a plot of interpolated points between 0 and 2 using the first-order and second-order Lagrange approximations of a sine function in half a period. We can see that the nonlinear curve represents the sine function much better but requires more anchors. Because in latent space, it is unlikely that the manifold is linear, it is expected that using more frames in the interpolation results in better synthesis quality. We highlight that other polynomial interpolation methods can also be applied, but all polynomial methods have the same interpolation error, so it is unlikely that other interpolation methods could result in any performance gain.

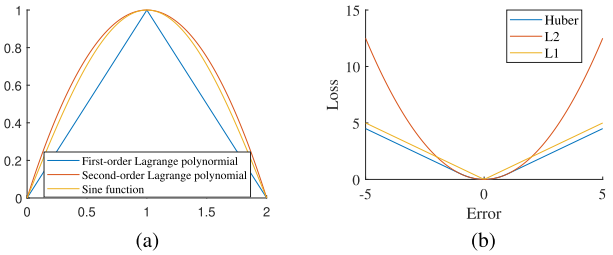

FIGURE 5. (a) Plot of the first-order and second-order Lagrange polynomials. (b) Plot of Huber loss, L2 loss, and L1 loss. The threshold of Huber loss is 1.

For a concrete example, let us consider the first-order linear interpolation. In that case, we choose a set of $|\mathcal{N}| = 2$ nodes, *i.e.*, only the first and last frames are in the set of nodes. The interpolated point can be written as

$$\begin{aligned} \hat{z}_t &= -\frac{t-m}{m} z_0 + \frac{t}{m} z_m \\ &= z_0 + \frac{t}{m}(z_m - z_0). \end{aligned} \tag{6}$$

An illustration of this strategy is shown in Fig. 6. Obviously, this is an oversimplification of the underlying manifold. The real manifold in any non-trivial case should be highly nonlinear and there is no guarantee that the latent codes of the references are co-planar, much less on the same line. However, this is a reasonable choice because it is difficult to walk on the manifold if the manifold is complicated, and this oversimplification is very intuitive for choosing a frame position to interpolate.

The importance of the LTE module is somewhat analogous to that of the Naive Bayes assumption. When too few statistics of a high-dimensional space are obtained, Naive Bayes imposes a conditional independence assumption on the features so that the joint distribution over the features can be easily factored. This assumption is very "naive", but it often works very well when data is limited [29]. In this study, we encounter the same situation in which the dimension of the latent space is high but the number of points is extremely

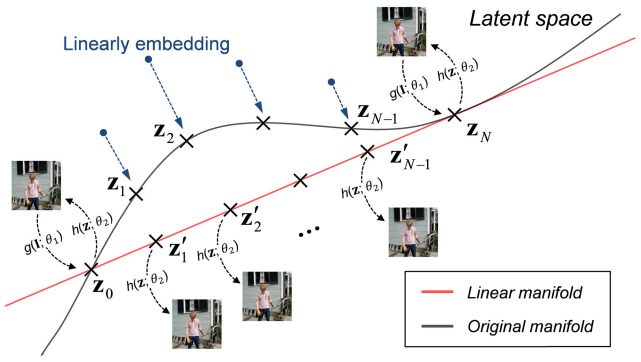

FIGURE 6. Linear interpolation in latent space. In most cases, the real latent representations $z_0, \ldots, z_m$ can lie arbitrarily on a nonlinear manifold illustrated by the black line. However, we approximate these points by linearly interpolating between z_0 and z_m so that all the latent codes $\hat{z}_j, j \in [1, m-1]$ are evenly distributed on the line segment between the codes of the two nodes (the red line).

small, which makes it impossible to construct a sophisticated manifold. However, these points are very close to each other, so we can assume that all the points are close enough that we can construct a polynomial estimation to the true manifold, and it is much easier to sample from the manifold made up by these points. This assumption is "naive" but works reasonably well, and more crucially, it makes sampling from the manifold possible.

1) ANOTHER PERSPECTIVE

We would like to have a closer look at the proposed framework from the adaptive average point of view. As briefly mentioned before, because of the way convolution works, each dimension in the latent space is a nonlinear combination of a certain patch of the original image. By performing interpolation between the latent codes, we actually introduce some sort of resampling mechanism between patches of the same spatial positions in the node images. This resampling mechanism is better than the naive averaging method because it is adaptive, and when the motions are too fast and complicated, it synthesizes a blurry image at worst. In fact, under some assumptions, we can show that there exists at least one solution that can relate our method to the adaptive convolution approach in [30], [32]. We present a proof of this in Appendix -B.

D. AUTO-ENCODER STRUCTURE

The details of the network are described in Fig. 4. DeepLTE employs an auto-encoding CNN to go back and forth between the image and latent spaces. To the best of our knowledge, there exists no commonly accepted way to reconstruct an image with fine detail from a low-dimensional representation, as the best techniques to compress a high-resolution image without loss require a significant number of bits to represent the image [38]. In order to produce sufficiently high-fidelity images, the CNN must have a high capacity, which can be trivially increased within the limits of the hardware. In our experiments, we have consistently observed that expanding the network capacity enhances image quality. Therefore, we utilize an architecture that can exhaustively use all available GPU memory.

Concretely, we use the 34-layer deep residual network (ResNet-34) [9] as the encoder of DeepLTE after removing the first mean pooling layer, global average pooling layer, and softmax layer. Another modification is that we replace the rectified linear unit (ReLU) [28] activation with leaky ReLU (LReLU) [23] with a leaky parameter of 0.1. The reason is that we do not want the network to be robust against small changes in inputs, which is caused by functions having saturating regions [5]. Another reason is that it is more linear than all of its siblings (see Appendix -B). We perform extensive experiments to corroborate this choice.

For the decoder part, we found that simply stacking up convolutional layers interleaved with bilinear upsampling works consistently well in all our simulations. The numbers of convolutional layers in each stacking block are 3, 5, 7, and 9. All the kernels in the decoder have a receptive field size of 5×5. LReLU is used in all layers except for the output layer, which is activated by the hyperbolic tangent function (tanh). The reconstructed images are simply scaled by $\hat{\mathbf{I}} = 255(\mathbf{Y}/2 + 0.5)$, where $\mathbf{Y}$ is the tanh-activated output from the decoder. The input to the network is also preprocessed in the similar manner. Interestingly, we discovered that dropout [41] reduces glaring artifacts of the generated images very efficiently. Therefore, we add two dropout layers with a dropout probability of 0.5 after the third and fourth stacking modules. The interesting effect of dropout is investigated more in the Results section. We note that a more sophisticated choice of the network structure may improve performance, but it is not the main objective of our work.

E. LOSS FUNCTION

In this paper, to optimize DeepLTE, we use the reconstruction loss, gradient loss and perceptual loss between the output images and ground truths.

1) RECONSTRUCTION LOSS

For the reconstruction loss, we use Huber loss. As can be seen from Fig. 5(b), Huber loss is linear in the error for large errors but quadratic for small errors. Therefore, unlike L1 loss, Huber is less sensitive to small errors, and unlike L2, it does not magnify large errors but penalizes them according to their magnitudes. Thus, Huber loss is robust to outliers by estimating the median as L1 loss and also to small Gaussian perturbations by returning the mean as L2 loss. For the sake of completeness, we present the formula of the loss as follows

$$\mathrm{H}(x, y) = \begin{cases} \frac{1}{2}\tau^2 + \tau(|x-y| - \tau) & \text{if } |x-y| \geq \tau \\ \frac{1}{2}(x-y)^2 & \text{otherwise,} \end{cases} \tag{7}$$

where x and y are two scalar inputs and τ is a predefined threshold, which we set to 1 in our implementation. The cost between a reference and its reconstructed version

can be calculated as

$$\mathrm{R}(\mathbf{I}, \hat{\mathbf{I}}) = \frac{1}{p} \sum_{(x_i, y_i) \in (\mathbf{I}, \hat{\mathbf{I}})} \mathrm{H}(x_i, y_i), \tag{8}$$

where p is the number of pixels in each image.

2) PERCEPTUAL LOSS

Following [13], we use the *feature reconstruction loss* to measure the high-level semantic differences between an image and its reconstruction. Instead of measuring the Euclidean distance between the original inputs, the loss first maps both inputs to another space and then reports the distance between the transformed inputs. We use VGG16 [40] pretrained on the ImageNet recognition dataset after removing all the fully connected layers as the mapping. It is well-known that deep NNs extract low-level features such as edges and blobs in early layers and perceptual and semantic features in deeper layers, which justifies the name ''perceptual''. Furthermore, VGG16 is trained on natural images, so it ''overfits'' to the manifold of natural images. A badly synthesized image input to VGG16 may not be recognized by the network and will easily fall off the manifold, which eventually results in a high cost. The perceptual loss is defined as

$$\mathrm{P}(\mathbf{I}, \hat{\mathbf{I}}) = \frac{1}{l} \left\| \phi(\mathbf{I}) - \phi(\hat{\mathbf{I}}) \right\|_2^2, \tag{9}$$

where ϕ is the VGG16 mapping and l is the number of elements in each transformed input. Note that we keep the weights of the pretrained VGG16 fixed during optimization.

3) GRADIENT LOSS

To further enhance the sharpness of the generated images, we utilize difference of Gaussians (DoG) loss. Concretely, the loss is calculated as

$$\mathrm{G}(\mathbf{I}, \hat{\mathbf{I}}) = \frac{1}{p} \left\| DoG(\mathbf{I}) - DoG(\hat{\mathbf{I}}) \right\|_2^2, \tag{10}$$

where $DoG(\mathbf{U}) = g_1(\sigma_1) * \mathbf{U} - g_2(\sigma_2) * \mathbf{U}$ denotes the difference of two low-passed images obtained by convolving the input image $\mathbf{U}$ with two Gaussian filters g_1 and g_2 at two scales σ_1 and σ_2 ($\sigma_1 > \sigma_2$), respectively, and p is the number of pixels in each image. The rationale for our choice over other common image gradient operators is shown in Fig. 7. As can be seen from the figure, the DoG image has many more edge details and less noise than the gradient image, so the DoG loss can better increase the sharpness of the generated images. In our implementation, we set $\sigma_1 = 1.6$ and $\sigma_2 = 1$, which efficiently simulates the well-known edge detector Laplacian of Gaussian.

Our final objective function is

$$\mathrm{L}(\mathbf{I}, \hat{\mathbf{I}}) = \mathrm{R}(\mathbf{I}, \hat{\mathbf{I}}) + \lambda_1 \mathrm{G}(\mathbf{I}, \hat{\mathbf{I}}) + \lambda_2 \mathrm{P}(\mathbf{I}, \hat{\mathbf{I}}). \tag{11}$$

We empirically set $\lambda_1 = 10^{-3}$ and $\lambda_2 = 10^{-4}$ in all experiments. The network is optimized end-to-end so that the decoder can guide the encoder to find a latent space where both the reconstructions from the nodes' latent codes and the interpolated codes are possible. We initialize the weights of the auto-encoder using He initialization [9]. We use the ADAM optimization scheme [17] with a learning rate of 10^{-4}, and the other parameters are set to the authors' suggestions. Because the fitting is blindly operated, we can either run until convergence or terminate the optimization at some fixed iteration. In our experiments, we chose the latter and set the number of iterations to 5000. As a side note, better performance might be achieved by running the optimization longer. We implemented our model using Theano.[1] [2]

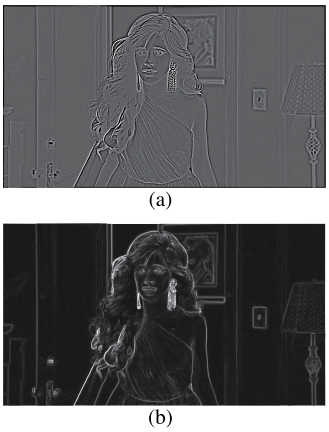

FIGURE 7. An example of (a) DoG and (b) the first derivative of an image. The DoG image contains much more edge information than the first derivative image. Best viewed in digital zoom.

IV. EXPERIMENTAL RESULTS

We considered three variants of DeepLTE: DeepLTE-1, DeepLTE-2, and DeepLTE-3, which correspond to the first-, second-, and third-order latent interpolation in DeepLTE, respectively.

We assessed our framework using UCF-101 [41], DAVIS [35] and real-life videos. For UCF-101, we tested our method on the test set provided in [25]. The testing is carried out as follows. For DeepLTE-3, we used frames 1, 2, 3, 5, 6, and 7 for optimization. In optimization, frames 1, 3, 5 and 7 are used as input to the CNN. The CNN reconstructs these frames and interpolates frames 2 and 6 as output. We optimized the weights of the CNN based on these reconstructions (frames 1, 3, 5, and 7 are used as ground truths) and interpolations (frames 2 and 6 are ground truths). There is no additional data necessary for optimization. After optimizing the weights, the CNN will interpolate frame 4 and we measured the performance between the original and interpolated frame 4. For DeepLTE-2, we used frames 1, 2, 3, and 5 for optimization in which 1, 3 and 5 are input and 2 is interpolated as above. Similarly, for DeepLTE-1, we used frames 1, 3, and 5 for optimization. Frames 1 and 5 are used to predict frame 3 in optimization as above. As we consider the visual quality with repsect to the HVS, we used two metrics, namely visual information fidelity (VIF) [39]

[1] Available at https://github.com/justanhduc/DeepLTE.

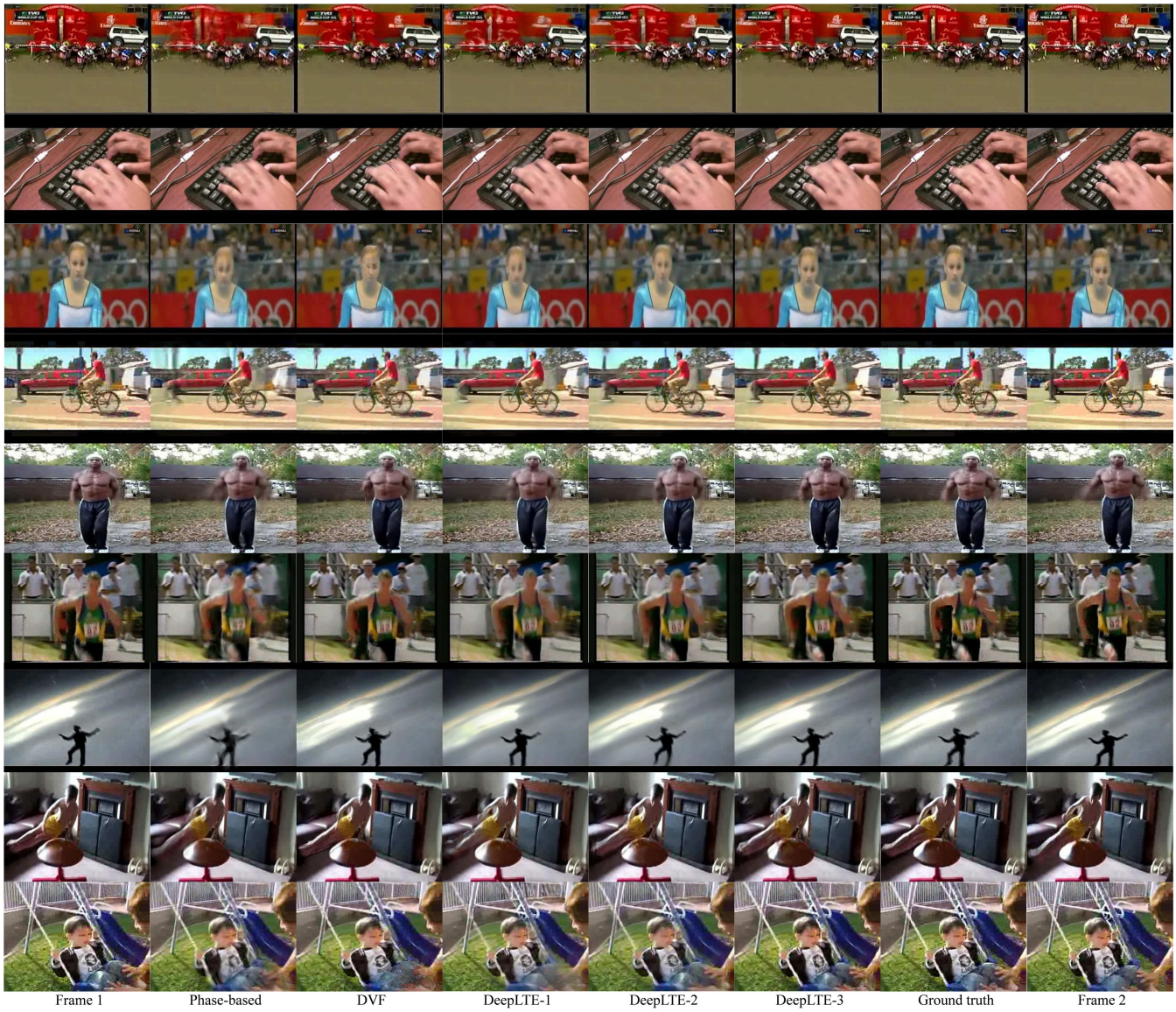

FIGURE 8. Qualitative results on UCF101. From left to right: frame1, phase-based [27], DVF [21], DeepLTE-1, DeepLTE-2, DeepLTE-3, ground truth middle image, and frame 2. Best viewed in color and zoom.

and structural similarity index (SSIM), which are widely used in image quality-assessment literature, to evaluate the synthesized images. Following [21], [25], we filter only the motion regions and evaluated on the filtered image. All input images were processed at their original resolution (240×320) and normalized into the range of $[-1, 1]$.

For DAVIS, we selected several videos and only measured the performance visually, as there is no previous work performing any benchmark on this database. To demonstrate the ability to generate any number of frames between two nodes, we interpolated three new frames between each pair of reference frames. We applied the same strategy to several real-life video segments including discoveries, music videos, and movies. Since real-life videos usually contain frames of different scenes, we optionally used shot detection [33], [34] to determine which frames are eligible to be used as the anchors. Because of the limited GPU memory, all sequences are processed at 224×384.

To benchmark our model, we chose several state-of-the-art methods in the field: FlowNet2-based [11], a state-of-the-art optical flow method; DVF [21], an implicit optical-flow-based method; Beyond MSE [25], a state-of-the-art pixel hallucination method; SepConv [30], a state-of-the-art model based on adaptive convolution; Super SloMo [12], a model similar to DVF; and phase-based frame interpolation [27], a well-known plug-and-play and non-learning alternative to optical-flow-based methods.

A. QUALITATIVE RESULTS

Fig. 8 demonstrates the interpolated frames of DVF [21], the phase-based method [27], and three variants of our proposed models that use the first, second and third-order latent

interpolation, along with the corresponding ground truths. All these sequences contain very fast and large motions between the two input frames.

In the first row, while both the phase-based method and DVF present many noticeable artifacts in the interpolated images, all variants of DeepLTE perform properly. As can be seen from the figure, the image synthesized by DeepLTE-3 is the most impressive because it is interpolated smoothly in the far background areas, which have even larger and faster motions than the foreground objects. In the second row, even though the motions are simple, the first three methods fail to deliver visually pleasant outcomes. Only DeepLTE-2 and especially DeepLTE-3 are able to produce satisfactory interpolated frames that are sharp and pleasing. We further note that when there exists motion blur, as in the first frame, the phase-based method and DVF usually fail to produce high-quality interpolated frames, which suggests the vulnerability of the existing methods to motion blur. In the third row, the phase-based method, DVF and DeepLTE-1 struggle to generate satisfying results. Although DeepLTE-2 creates blur in the areas of large motions, DeepLTE-3 is still consistent in its performance when synthesizing a very sharp in-between frame. In the last row, when motions are too large, all methods fail in different ways. The phase-based method and DeepLTE-1 produce double vision. DVF confusingly warps the image and the result is terribly distorted. DeepLTE-3, however, still manages to generate an image with heavy blur in the motion regions, yet all the objects and textures are still intact and recognizable. The rest of Fig. 8 strongly reinforces our analyses above. One may reason that the improvement in the performance of DeepLTE-2 and DeepLTE-3 should simply be credited to the use of more frames. However, in any non-trivial video, frames are abundant and also, to our knowledge, no existing method is able to harness this abundance. Our framework provides an elegant solution to effectively utilize this rich information from a video.

In general, it is observed that the images generated by our method are of equal quality or even visually more pleasant compared with state-of-the-art learning-based methods. In addition, taking a closer look, we notice that our method actually reduces artifacts and distortion from the original images. The same phenomenon has been observed and discussed in [6], a study concurrent to ours. We conclude that our method is preferable to existing methods in many situations because it does not require pre-training on big data and works similarly to conventional methods, but at the same time possesses the computing power of deep learning, which is the key to the immense improvements of the synthesized quality over conventional models and places this method on par with state-of-the-art deep-learning-based methods.

Fig. 9 shows some of our interpolated frames from several DAVIS and real-life videos using DeepLTE-1. We processed several segments of the videos and ran them at 12 *frames per second* for a slow-motion effect. We highly encourage readers to view our webpage[1] for more results, as it is impossible to evaluate temporal coherence on still paper.

FIGURE 9. Examples of our interpolated frames in DAVIS and real-life videos using DeepLTE-1. Best viewed in color and zoom.

TABLE 2. Quantitative performance on UCF-101. Plug-and-play methods are italic. Higher scores are better. Red, blue, and green indicate the first-, second-, and third-best methods, respectively.

Method	SSIM	VIF
Beyond MSE [26]	0.924	-
FlowNet2-based [12]	0.930	-
DVF [22]	0.930	0.622
SepConv [31]	0.935	-
Super SloMo [13]	0.938	-
Phase-based [28]	0.929	0.575
DeepLTE-1	0.928	0.603
DeepLTE-2	0.932	0.610
DeepLTE-3	0.943	0.637

B. QUANTITATIVE RESULTS

Table 2 shows the benchmark on UCF-101 between the proposed methods versus the chosen models. As expected, our method outperforms all other benchmarking methods in terms of perceptual quality. Concretely, the synthesized images from DeepLTE-3 have higher SSIM and VIF scores than current state-of-the-art performance. SSIM and VIF are well-known for their correlations with the human-perceived quality. High performance on these metrics is a strong indicator that our generated images are likely to be favored by viewers. We further emphasize the noteworthy performance gap between DeepLTE and the phase-based method, which is also a plug-and-play method. We deduce that performance can be substantially improved by blending a deep computing architecture into the traditional plug-and-play fashion.

C. ABLATION STUDY

We verify some of our choices for the proposed framework. To perform the ablation study, we randomly selected 20 videos from the UCF-101 dataset. We used DeepLTE-1 in all the ablation studies, if not otherwise specified. All settings except for the ablated components were set to our defaults.

TABLE 3. Quantitative ablation study on different components of our framework. Red, blue, and green indicate the first-, second-, and third-best methods, respectively.

Ablation	DeepLTE-1 (w/o LTE)	DeepLTE-1 (w/o dropout)	DeepLTE-1 (w/o perceptual loss)	DeepLTE-1 (ReLU)	DeepLTE-1 (ELU)	DeepLTE-1 (SELU)	DeepLTE-1 (LReLU)
SSIM	0.925	0.943	0.951	0.947	0.951	0.941	0.959
VIF	0.569	0.629	0.623	0.618	0.619	0.604	0.632

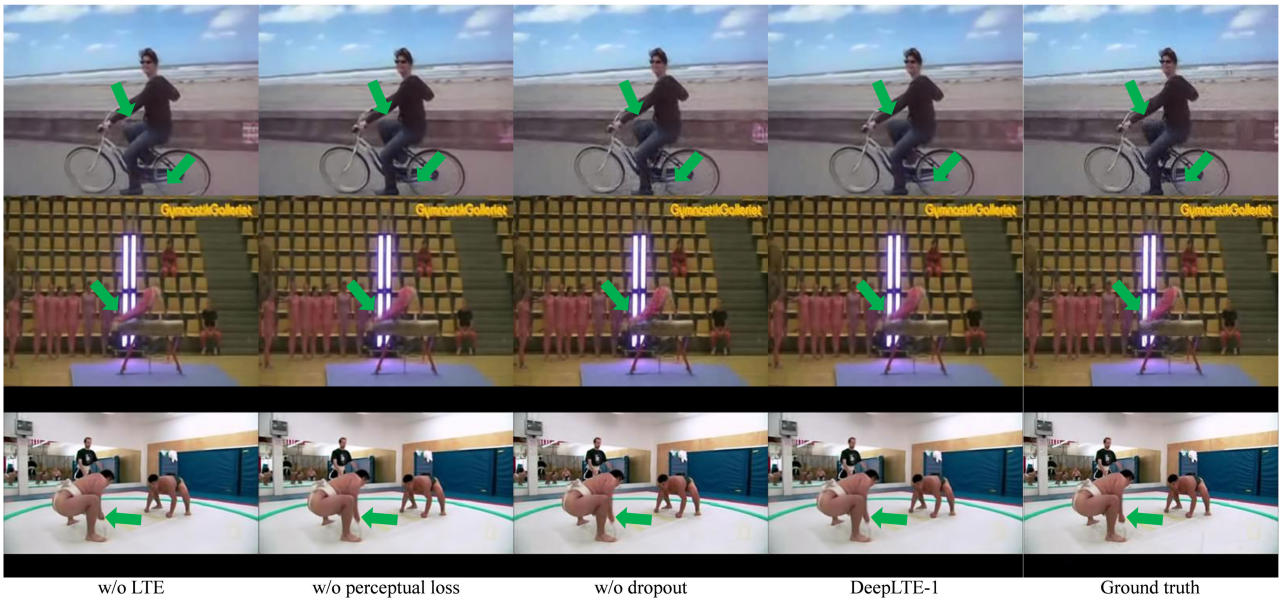

FIGURE 10. Qualitative ablation study on different components of DeepLTE-1. Best viewed in color and zoom.

1) LOCALLY TEMPORAL EMBEDDING MODULE

To study the effect of the LTE module, we trained an auto-encoding CNN by a common training scheme. Concretely, we removed the LTE module and used all the reference frames as input. The network architecture is the same as in Section III-D. Table 3 and Fig. 10 show the quantitative and qualitative performance when the LTE module is removed in training. In general, the quality of the generated images is blurrier than our full framework. In all sequences, the synthesized images contain limited motion or even none at all. They look almost the same as either of the two nodes. The numerical results consensually suggest that without LTE, the quality is much worse. We can see that this case is equivalent to using only two reference frames for the linear constraint ($m = 1$). Therefore, our method can work with only two input frames by discarding the LTE module. However, this is highly discouraged. Without the module, the network is not taught how to decode the points on the line segment passing through the two nodes. By explicitly embedding the interpolation rule on the latent manifold, we guided the network to encode the nodes so that the synthesized latent codes can be decoded properly.

Nevertheless, an important observation from this benchmark is that deep networks capture a great deal of the underlying manifolds of video sequences becaues they can decode an interpolated latent variable even though they are not optimized to do so. In [6], the authors discovered that deep networks can capture the structure of an image, and using it as a prior, they can blindly restore or super-resolve the image. Here, we have found that in addition to image structures, they can also capture video structures. This analysis can be greatly beneficial to the advancement of deep unsupervised learning. We demonstrate this potential in Section IV-D.

2) SCALABILITY

To perform this experiment, we randomly chose 10 sequences from DAVIS becaue UCF-101 does not have high-resolution video. Due to the limitation of our hardware memory (only 1 Titan X 12GB GPU), we could not scale our method to HD resolution as in other methods, which are trained on low-resolution images and tested on HD images. Nonetheless, we performed a benchmark on different resolutions within the permissible range and showed the results in Table 4. From the results, we can see that our model can work with high-resolution images by mapping images to a latent space of appropriate dimension. The reason is that when the input frames are large, the network filters are relatively small and cannot capture information in a large neighbor. By performing a series of filtering and downsampling, the network can gradually capture more and more global information.

TABLE 4. Quantitative results of DeepLTE-1 on different resolutions.

Resolution	SSIM
448 × 768	0.934
224 × 384	0.953
112 × 192	0.968
448 × 768 (5 poolings)	0.946

TABLE 5. Quantitative performance in multi-frame synthesis scenario.

Method	SSIM
DeepLTE-1	0.937
Phase-based [28]	0.933

We found that when the dimension of the latent space is too small, the generated image motions are better but the generated image is blurrier. Hence, there is a trade-off between the generated image quality and motion quality in DeepLTE.

3) MULTI-FRAME SYNTHESIS

In this experiment, we used 7 frames in each test sample. Frames 1 and 7 were used as nodes. Frame 4 was used as the ground truth for optimization. Using those 3 frames, we generated 2 frames in each interval and evaluated them using the corresponding ground truths. The benchmark is tabulated in Table 5. It is noted that the nodes in this case are 5 frames apart, which causes any frame-synthesis method to struggle.

4) EFFECT OF DROPOUT

Fig. 10 shows the visual results of our method with and without dropout. As can be seen from the figure, the images generated without dropout have noticeable artifacts in the interpolated regions, whereas those with dropout do not show any sort of artifacts. As expected, as can be seen from Table 3, the perceptual numerical results of using dropout are also somewhat higher. The effect of dropout on the generated images can be useful in situations where we want to both restore and upsample low-frame-rate videos.

5) CHOICE OF ACTIVATION

As can be seen from Table 3, our method using LReLU surpasses all other configurations of activation functions. An explanation is that ReLU, SELU, and ELU saturate when the argument is below some threshold. This saturation makes the network robust against small perturbations in input [5], which is quite unwanted in our method because the two nodes' latent variables and the interpolated codes are very close to each other. This experiment implies that although deep NNs are a powerful computing tool, one may need to design their architectures based on thorough analyses of the problems to which they are applied.

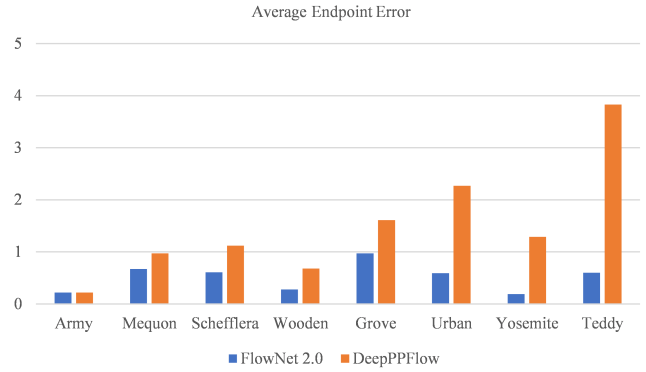

FIGURE 11. Quantitative results on the Middlebury dataset.

D. AN INTERESTING CAPABILITY OF DEEP CNN

Traditional manifold learning methods require a large random sample so that all the variations on the underlying manifold can be covered. In this study, we have hypothesized and shown evidence that even with a few neighboring data points, we can still construct an interesting manifold because deep CNN can capture interesting statistics of videos. We demonstrated how the captured statistics can be applied to the frame-synthesis problem. In this section, we show how the statistics obtained in this manner can be applied to another area and thereby highlight the potential of our discovery to the unsupervised learning area.

Because the latent representations of a set of adjacent frames can be used to interpolate in-between images, we conjecture that these representations must enclose video motions. To retrieve these motions, we adopt the flow-estimation subnetwork in [31]. The only modification is that instead of training the network with a set of big data, we optimize it in the proposed plug-and-play fashion. We evaluate our method, deep plug-and-play flow (DeepPPFlow), on the Middlebury dataset [1] and obtain the results from the Middlebury server.[2] We show the results of DeepPPFlow in Fig. 11. The competing model is FlowNet2 [11], a state-of-the-art deep supervised method in optical flow. We note that the demonstration is solely to verify the usefulness of the captured statistics by the CNN because the model is basic and highly unpolished. We conclude that CNNs working in a plug-and-play fashion are profoundly promising to boost the deep unsupervised-learning area.

V. CONCLUSION

In this paper, we have presented a new method for in-between frame generation. Our method constructs a polynomial approximation of the true manifold of the latent representations of consecutive frames obtained from an auto-encoding CNN. Our model is an off-the-shelf method that can be applied instantly to a video, like conventional methods, while possessing deep-learning-level performance. Being plug-and-play, our method has a high expressive power in

[2]http://vision.middlebury.edu/flow/eval/results-anh-duc-nguyen/

that a wide selection of latent constraints and network structures can be incorporated into our algorithm. DeepLTE can synthesize an arbitrary number of images among any number of frames simultaneously. Different benchmarks reveal that our synthesized images are comparable with state-of-the-art performance in the field. Moreover, our work exposes that deep CNNs capture a great deal of video statistics, which may be greatly beneficial to the study of deep unsupervised learning.

A. LOCALLY LINEAR EMBEDDING

To make the paper self-contained, we briefly summarize the idea of locally linear embedding (LLE) [37]. Suppose we have a dataset $X \in \mathcal{R}^{m\times n}$ having m samples of n dimensions. We want to find a compact representation of X, *i.e.*, a lower-dimensional space that preserves local relationships of each point. We assume that there are sufficient data so that all the twists and variations in the manifold are included. Based on this condition, we hypothesize that each point is a linear combination of those in its neighborhood. To find such a linear relationship W, for each X_i, we find a set $\mathcal{N}_i$ containing all the indices of the neighbors of X_i according to some distance threshold and then solve the following optimization to find the weights W_{ij}:

$$\min_W \sum_i \left\| X_i - \sum\nolimits_{j\in\mathcal{N}_i} W_{ij}X_j \right\|^2, \quad s.t \quad \sum_j W_{ij} = 1, \qquad (12)$$

where $W_{ij} = 0$ if $X_j \notin \mathcal{N}_i$, *i.e.*, X_j is not in the neighborhood of X_i. The goal of LLE is to find a new low-dimensional space in which the linear relationship W is preserved. Suppose we have a function that maps each X_i in the original space to each $Y \in \mathcal{R}^{m\times k}$ in some latent space where $k \ll n$, then we can find Y by solving the following problem:

$$\min_Y \sum_i \left\| Y_i - \sum\nolimits_{j\in N_i} W_{ij}Y_j \right\|^2,$$

which is similar to (12) but the minimization is over Y instead of W.

B. RELATIONSHIP TO ADAPTIVE CONVOLUTION METHODS

We attempt to look at the proposed method as an adaptive average approach. We show that under some mild conditions, our linear approach has a close relationship to the adaptive convolution method in [30], which means that with a special design of g, synthesizing new latent codes has the same effect as adaptive convolution.

Suppose we have two consecutive frames I_1 and I_2. From [30], we have

$$\hat{I}(x, y) = F_1(x, y) * P_1(x, y) + F_2(x, y) * P_2(x, y), \quad (13)$$

where x and y are the spatial indices of the images, $P_1(x, y)$ and $P_2(x, y)$ are patches centered at pixel (x, y) in I_1 and I_2, respectively, $F_1(x, y)$ and $F_2(x, y)$ are two pixel-dependent kernels carrying motion and resampling information corresponding to I_1 and I_2, respectively, $\hat{I}(x, y)$ is the result of the synthesis process, and "$*$" denotes the 2D convolution operator. Intuitively, each pixel in $\hat{I}$ can be calculated by convolving its own pair of kernels with the patches centered at the same spatial position in I_1 and I_2. The two kernels $F_1(x, y)$ and $F_2(x, y)$ are originally obtained through training a CNN with a big dataset consisting of image triplets. Learning with big data requires heavy computational overhead, not to mention that the method learns a set of filters for each pixel in the synthesized image. To establish a relationship between this method and our approach, we will prove the following theorem.

Theorem 1: Let $g(\cdot)$ be a function that maps $\hat{I}$, I_1, and I_2 to $\hat{z}$, z_1, and z_2, respectively. If g is chosen to be a homomorphism of the image space onto some latent space that transforms addition and convolution into themselves, and the images of the adaptive filters corresponding to each input image belong to some same coset of the kernel of g, then in the latent space, (13) can be expressed as

$$\hat{z} = \alpha z_1 + (1 - \alpha)z_2,$$

where α is a real scalar.

Proof: First, applying g to (13) we obtain

$$\begin{aligned}\hat{z}(i, j) &= g\left(F_1(x, y) * P_1(x, y) + F_2(x, y) * P_2(x, y)\right)\\ &= g\left(F_1(x, y) * P_1(x, y)\right) + g\left(F_2(x, y) * P_2(x, y)\right)\\ &= g\left(F_1(x, y)\right) * g\left(P_1(x, y)\right)\\ &\quad + g\left(F_2(x, y)\right) * g\left(P_2(x, y)\right), \end{aligned} \qquad (14)$$

where the point (i, j) is the image of the point (x, y) in the latent space. The second and third equalities use the fact that g is a homomorphism of the image space onto some latent space that transforms addition and convolution into themselves. Next, suppose that g is a surjection from the image space to the latent space. Because the set of adaptive filters corresponding to an input frame belongs to some same coset of the kernel of g, we can find a function g that maps these filters to a scalar in the latent space. Concretely, let the support of the image of the filters be $[-\frac{k}{2}; \frac{k}{2}]^2$ for some odd k. Then, we choose the images of the filter pairs in the latent space to be

$$g\left(F_1(x, y)\right) = \alpha \quad \forall x, y, \qquad (15)$$

and

$$g\left(F_2(x, y)\right) = 1 - \alpha \quad \forall x, y. \qquad (16)$$

where α is a real scalar. Such a function g exists because the image space and latent space are homomorphic. Substituting (15) and (16 into (14), we have

$$\hat{z} = \alpha z_1 + (1 - \alpha)z_2,$$

that is, the new code $\hat{z}$ is on the line segment passing through z_1 and z_2. □

In summary, with some certain choice of g, the latent code of the synthesized image produced by the adaptive convolution method can be made to lie on the line segment between the codes of the two input frames, and instead of finding a set of filters for each pixel, which is not resource-friendly, we can find a suitable encoder and decoder. Finding these maps over all possible functions, however, is a formidable task. The encoder g should find a latent space so that the image space and the latent space are homomorphic with respect to addition and convolution. To make the matter worse, g should map the set of adaptive filters of an image to a single scalar, which requires that a coset of the kernel of g contain all the filters. There can be many such functions, and yet there are no clear objectives in order to find such a function. Fortunately, we have a perfect tool for function approximation, which is neural networks (NNs). Because an NN is a universal approximator [10], it is able to approximate the true latent space without needing satisfy all the requirements. Therefore, we choose a CNN, a special type of NN, as the encoder, which maps the image space to latent space, and another CNN as the decoder, which performs the inverse mapping, and we train both networks end-to-end because we have targets only for the decoder, and none for the encoder. The requirement that g should be a homomorphism can guide us to choose appropriate activation functions for our network. For example, we can choose LReLU as the network's activation function, which is a homomorphism up to some scaling constant. Thus, in all of our experiments, we use LReLU and heuristically show in the Results section that other functions, such as exponential linear unit (ELU) [5] and scaled exponential linear unit (SELU) [18], which transform addition in the image space to multiplication in the latent space, decrease the performance of our model.

Above, we have pointed out at least one solution that explains why our method works. However, which solution the CNN determines in this case is still unclear. Because of the vast selection of possible functions, the CNN can approximate the true function differently, and furthermore, even the true function can take numerous forms other than the ones we have discussed. We must admit that there is still a room for theoretical work to be done in order to explain how interpolation in latent space can induce interpolation in image space.

C. RELATIONSHIP TO CONDITIONAL VARIATIONAL AUTOENCODER

Our study has a very close relationship with [46]. Concretely, in [46], the authors employed a variational autoencoder (VAE) and condition on an input image to generate a set of possible subsequent frames. To generate trajectories, the VAE takes an image and a set of latent variables sampled from a standard Gaussian distribution and performs a series of convolutions and downsamplings/upsamplings. The authors observed that the latent variables encode motion directions. Specifically, interpolating between motion prediction can be done via interpolating between latent variables. However, this type of interpolation is simply a blind walk in the latent space, which makes most of the trajectories nowhere near real motions. DeepLTE can be seen as a restriction on the trajectories in latent space. DeepLTE allows only movements on a plane containing the latent representations the input frames, which guarantees more realistic motions in the interpolated images. In that sense, DeepLTE generates deterministic trajectories to an input instead of probabilistic trajectories as in [46].

D. RELATIONSHIP TO WASSERSTEIN AUTOENCODER

The autoencoder we employ in this study has a loose relation with [43]. Both our study and [43] directly embed some constraints on the latent space so that sampling is possible. However, there are two fundamental differences. Firstly, [43] aims to generate diverse images with any latent drawn from a prior distribution while our goal is to synthesize neighboring samples given some input images. Secondly, the constraint we adopt in our study is deterministic while the one utilized in [43] is probabilistic. Specifically, Tolstikhin *et al*. minimize a divergence between $E_{X\sim P_X}[Q(Z|X)]$ and a priorly specified distribution P_Z, where Z is a latent variable drawn from a simple distribution P_Z, X is an input data point drawn from the data distribution P_X, and $Q(Z|X)$ is the modeled conditional distribution of Z given X. Both constraints in the two studies require some simplifications and assumptions on the true models and by imposing those, both can result in reasonable performances.

REFERENCES

[1] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, 2011.

[2] J. Bergstra, O. Breuleux, F. F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A CPU and GPU math compiler in Python," in *Proc. Python Sci. Comput. Conf.*, 2010, pp. 3–10.

[3] C. Bregler and S. M. Omohundro, "Nonlinear image interpolation using manifold learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1995, pp. 973–980.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2004, pp. 25–36.

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*. [Online]. Available: https://arxiv.org/abs/1511.07289

[6] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Nov. 2018, pp. 9446–9454. [Online]. Available: http://arxiv.org/abs/1711.10925, doi: 10.1109/CVPR.2018.00984.

[7] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2758–2766.

[8] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[10] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[11] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2, Jul. 2017, pp. 2462–2470.

[12] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super SloMo: High quality estimation of multiple intermediate frames for video interpolation," 2017, *arXiv:1712.00080*. [Online]. Available: https://arxiv.org/abs/1712.00080
[13] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 694–711.
[14] H. Kim and S. Lee, "Transition of visual attention assessment in stereoscopic images with evaluation of subjective visual quality and discomfort," *IEEE Trans. Multimedia*, vol. 17, no. 12, pp. 2198–2209, Dec. 2015.
[15] H. Kim, S. Lee, and A. C. Bovik, "Saliency prediction on stereoscopic videos," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1476–1490, Apr. 2014.
[16] J. Kim and S. Lee, "Deep learning of human visual sensitivity in image quality assessment framework," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1676–1684.
[17] D. P. Kingma and J. Ba, "Adam: A method for Stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2014, pp. 1–13.
[18] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 972–981.
[19] R. Krishnamurthy, J. W. Woods, and P. Moulin, "Frame interpolation and bidirectional prediction of video using compactly encoded optical-flow fields and label fields," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 5, pp. 713–726, Aug. 1999.
[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 84–90.
[21] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep Voxel flow," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, Oct. 2017, pp. 4463–4471.
[22] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu, "Learning image matching by simply watching video," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 434–450.
[23] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML Work. Deep Learn. Audio, Speech Lang. Process.*, 2013, pp. 1–6.
[24] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur, "Moving gradients: A path-based method for plausible image interpolation," *ACM Trans. Graph.*, vol. 28, no. 3, p. 42, Aug. 2009.
[25] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," 2015, *arXiv:1511.05440*. [Online]. Available: https://arxiv.org/abs/1511.05440
[26] S. Meyer, A. Djelouah, B. McWilliams, A. Sorkine-Hornung, M. Gross, and C. Schroers, "Phasenet for video frame interpolation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 498–507.
[27] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung, "Phase-based frame interpolation for video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1410–1418.
[28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[29] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 841–848.
[30] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive separable convolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 261–270.
[31] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1701–1710.
[32] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 670–679.
[33] M. Paul, W. Lin, C.-T. Lau, and B.-S. Lee, "Explore and model better I-frames for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 9, pp. 1242–1254, Sep. 2011.
[34] M. Paul, W. Lin, C.-T. Lau, and B. S. Lee, "A long-term reference frame for hierarchical B-picture-based video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 10, pp. 1729–1742, Oct. 2014.
[35] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 724–732.
[36] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1164–1172.
[37] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.
[38] K. Sayood, *Introduction to Data Compression*. New York, NY, USA: Elsevier, 2012.
[39] H. R. Sheikh and A. C. Bovik, "A visual information fidelity approach to video quality assessment," in *Proc. 1st Int. Work. Video Process. Qual. Metrics Consum. Electron.*, 2005, pp. 23–25.
[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[41] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*. [Online]. Available: https://arxiv.org/abs/1212.0402
[42] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 843–852.
[43] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," 2017, *arXiv:1711.01558*. [Online]. Available: https://arxiv.org/abs/1711.01558
[44] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Deep end2end Voxel2Voxel prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 402–409.
[45] N. Wadhwa, M. Rubinstein, F. Durand, and W. Freeman, "Phase-based video motion processing," *ACM Trans. Graph.*, vol. 32, no. 4, p. 80, Jul. 2013.
[46] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 835–851.
[47] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 1385–1392.
[48] M. Werlberger, T. Pock, M. Unger, and H. Bischof, "Optical flow guided TV-L1 video interpolation and restoration," in *Proc. Int. Work. Energy Minimization Methods Comput. Vis. Pattern Recognit.* Berlin, Germany: Springer, 2011, pp. 273–286.

ANH-DUC NGUYEN received the B.Eng. degree in automatic control from the Hanoi University of Science and Technology, Vietnam, in 2015. He is currently pursuing the joint M.Sc. and Ph.D. degrees with Yonsei University, South Korea. His research interests are image/video analysis, geometric computer vision, and deep learning.

WOOJAE KIM received the B.S. degree in electronic engineering from Soongsil University, Seoul, South Korea, in 2015. He is currently pursuing the M.S. and Ph.D. degrees with the Multidimensional Insight Laboratory, Yonsei University. He was a Research Assistant with the Laboratory for School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore, in 2018, under the supervision of Prof. W. Lin. His research interests include image and video processing based on the human visual systems, image/video quality assessment, computer vision, and machine learning.

JONGYOO KIM received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2011, 2013, and 2018, respectively. He joined Microsoft Research Asia, in March 2018. His research interests include quality assessment of 2D/3D image and video, 3D computer vision, and face geometry and texture. He was a recipient of the Global Ph.D. Fellowship by the National Research Foundation of Korea, from 2011 to 2016.

WEISI LIN (M'92–SM'98–F'16) received the Ph.D. degree from King's College, London University, U.K. He is currently a Professor with the School of Computer Science and Engineering, Nanyang Technological University. He has published more than 200 journal articles and 230 conference papers. He holds seven patents and authored two books. His areas of expertise include image processing, perceptual signal modeling, video compression, and multimedia communication. He is a Fellow of IET and an Honorary Fellow of the Singapore Institute of Engineering Technologists. He has been a Technical Program Chair for the IEEE ICME 2013, PCM 2012, QoMEX2014, and the IEEE VCIP 2017. He has been an invited/panelist/keynote/tutorial speaker for more than 20 international conferences, and a Distinguished Lecturer of the IEEE Circuits and Systems Society, from 2016 to 2017, and the Asia–Pacific Signal and Information Processing Association (APSIPA), from 2012 to 2013. He has been an AE of the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHONOLOGY, the IEEE TRANSACTIONS ON MULTIMEDIA, and the IEEE SIGNAL PROCESSING LETTERS.

SANGHOON LEE (M'05–SM'12) received the B.S. degree in electrical engineering from Yonsei University, in 1989, the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), in 1991, and the Ph.D. degree in electrical engineering from the University of Texas at Austin, in 2000. From 1991 to 1996, he worked for Korea Telecom. From 1999 to 2002, he worked for Lucent Technologies on 3G wireless and multimedia networks. In March 2003, he joined the Faculty of the Department of Electrical and Electronics Engineering, Yonsei University, Seoul, South Korea, where he is currently a Full Professor. His research interests include image/video quality assessment, computer vision, graphics, cloud computing and multimedia communications, and wireless networks. He received the 2015 Yonsei Academic Award from Yonsei University, the 2012 Special Service Award from the IEEE Broadcast Technology Society, and the 2013 Special Service Award from the IEEE Signal Processing Society. He was the Chair of the IEEE P3333.1 Quality Assessment Working Group, in 2011. He was the Technical Program Co-Chair of the International Conference on Information Networking (ICOIN) 2014 and the Global 3D Forum 2012 and 2013, and the General Chair of the 2013 IEEE IVMSP Workshop. He was an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, from 2010 to 2014, the IEEE SIGNAL PROCESSING LETTERS, in 2014, and the *Journal of Electronic Imaging*, in 2015. He is currently serving on the Technical Committee for the IEEE Multimedia Signal Processing, in 2016 and the IEEE IVMSP Technical Committee, in 2014. He also served as a special issue Guest Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, in 2013, and an Editor of the *Journal of Communications and Networks* (JCN), from 2009 to 2015.