Visual Dynamics: Stochastic Future Generation via Layered Cross Convolutional Networks

Tianfan Xue*, Jiajun Wu*, Katherine L. Bouman, and William T. Freeman

Abstract—We study the problem of synthesizing a number of likely future frames from a single input image. In contrast to traditional methods that have tackled this problem in a deterministic or non-parametric way, we propose to model future frames in a probabilistic manner. Our probabilistic model makes it possible for us to sample and synthesize many possible future frames from a single input image. To synthesize realistic movement of objects, we propose a novel network structure, namely a *Cross Convolutional Network*; this network encodes image and motion information as feature maps and convolutional kernels, respectively. In experiments, our model performs well on synthetic data, such as 2D shapes and animated game sprites, and on real-world video frames. We present analyses of the learned network representations, showing it is implicitly learning a compact encoding of object appearance and motion. We also demonstrate a few of its applications, including visual analogy-making and video extrapolation.

Index Terms—future prediction, frame synthesis, probabilistic modeling, convolutional networks, cross convolution

✦

1 INTRODUCTION

FROM just a single snapshot, humans are often able to imagine multiple possible ways that the scene can change over time. For instance, due to the pose of the girl in Fig. 1, most would predict that her arms are stationary but her leg is moving. However, the exact motion is often unpredictable due to an intrinsic ambiguity. Is the girl's leg moving up or down? How large is the movement?

In this work, we aim to depict the conditional distribution of future frames given a single observed image. We name the problem *visual dynamics*, as it involves understanding how visual content relates to dynamic motions. We propose to tackle this problem using a probabilistic, content-aware motion prediction model that learns this distribution without using annotations. Sampling from this model allows us to visualize the many possible ways that an input image is likely to change over time.

The visual dynamics problem is in contrast to two traditional ways to model motion. The first is to assume object motion is deterministic, and to learn the direct mapping from an image's visual appearance to its motion [1], [2]. These methods are more likely to produce accurate results when there is little ambiguity in object motion (*e.g.*, when long-range videos are available). For single image future prediction, however, the results are unlikely to align with reality. The second way to model motion is to derive its prior distribution, which is invariant to image content [3], [4]. Understanding motion priors, like understanding image

priors [5], is a problem of fundamental scientific interest. However, as the motion we observe in real life is strongly correlated with visual content, methods ignoring visual signals do not work well on future frame prediction.

Modeling content-aware motion distributions is highly challenging mostly for two reasons. First, it involves modeling the correlation between motion and image content; to predict plausible future motion, the model must be able to relate visual content to semantic knowledge (*e.g.*, object parts) in order to generate reasonable motions. Second, natural images lie on a high dimensional manifold that is difficult to describe precisely. Despite recent progresses on applying deep learning methods for image synthesis [6], building a generative model for real images is far from being solved.

We tackle the first problem using a novel convolutional neural network. During training, the network observes a set of consecutive image pairs from videos, and automatically infers the relationship between them without any supervision. During testing, the network then predicts the conditional distribution, $P(J|I)$, of future RGB images J (Fig. 1b) given an RGB input image I that was not in the training set (Fig. 1a). Using this distribution, the network is able to synthesize multiple different image samples corresponding to possible future frames for the input image (Fig. 1c).

We use a conditional variational autoencoder to model the complex conditional distribution of future frames [7], [8]. This allows us to approximate a sample, J, from the distribution of future images by using a trainable function $J = f(I, z)$. The argument z is a sample from a simple (*e.g.*, Gaussian) distribution, which introduces randomness into the sampling of J. This formulation makes the problem of learning the distribution more tractable than explicitly modeling the distribution.

To synthesize complex movement of objects, we proposed a novel layer-based synthesis network. The network splits an image into multiple segments and then uses a layered model to predict how each segment moves. This is a much easier task than modeling the motion of an entire image. Note that

• T. Xue is with Google Inc., Mountain View, CA, 94043 USA.
E-mail: tianfan.xue@gmail.com.
• J. Wu is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 02139 USA. E-mail: jiajunwu@mit.edu.
• K. L. Bouman is with Harvard University, Cambridge, MA, 02138 USA. E-mail: klbouman@gmail.com.
• W. T. Freeman is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, and Google Inc., Cambridge, MA, 02139 USA. E-mail: billf@mit.edu.

A Conditional Distribution of Future Frames

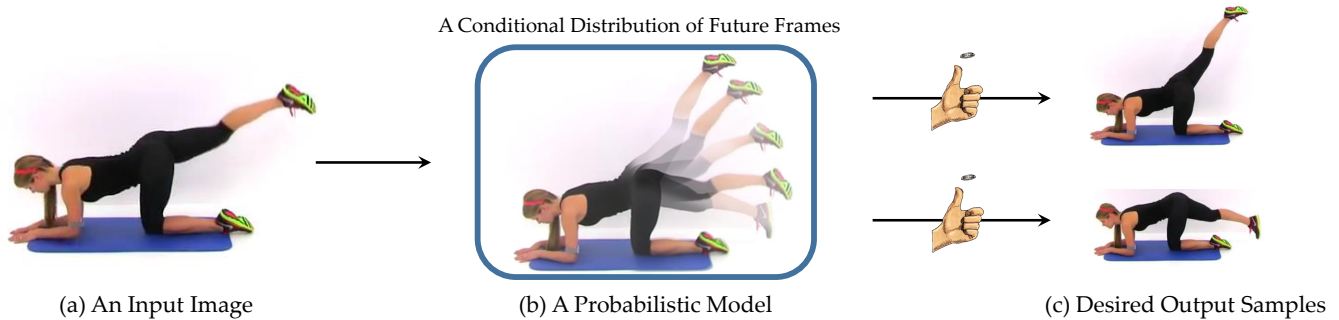(a) An Input Image (b) A Probabilistic Model (c) Desired Output Samples

Fig. 1. Predicting the movement of an object from a single snapshot is often ambiguous. For instance, is the girl's leg in (a) moving up or down? We propose a probabilistic, content-aware motion prediction model (b) that learns the conditional distribution of future frames, and produces a probable set of future frames (c). This schematic illustrates the idea behind our method, but does not show actual results produced by our model.

here we call each layer of an image as a segment to avoid confusion with convolutional layers. This layered prediction model synthesizes motion using a novel cross-convolutional layer. Unlike in standard convolutional layers, the values of the kernels are image-dependent, as different images may have different motions. Our model has several advantages. First, avoiding blurry outputs, the model does not directly synthesize the output image, but instead transforms pixels in the input frame based on sampled motion parameters. Second, the model only samples the movement of each layer, instead of all the pixels or a dense flow field. Because the motion of each layer lies on a lower-dimension manifold, its distribution is easier to model and the network can sample more diverse and realistic motions.

We test the proposed model on four datasets. Given an RGB input image, the algorithm can correctly model the distribution of possible future frames, and generate different samples that cover a variety of realistic motions. Our system significantly outperforms baselines in quantitative evaluation, and our results are in general preferred by humans in user studies.

We present analyses to reveal the knowledge captured by our model: the cross convolutional network is able to discover semantically meaningful parts that have coherent motion patterns in an unsupervised fashion; and the latent representation z in the variational autoencoder is in essence a compact, interpretable encoding of object motion, as visualized in Section 6. Our model has wide applications: we demonstrate that it can be applied to visual analogy-making and video extrapolation straightforwardly, with good qualitative and quantitative performance.

2 RELATED WORK

Motion priors. Research studying the human visual system and motion priors provides evidence for low-level statistics of object motion. Pioneering work by Weiss and Adelson [4] found that the human visual system prefers slow and smooth motion fields. More recent work by Lu and Yuille [9] found that humans make similar motion predictions as a Bayesian ideal observer. Roth and Black [5] analyzed the response of spatial filters applied to optical flow fields. Fleet *et al.* [3] also found that a local motion field can be represented by a linear combination of a small number of bases. All these works focus on the distribution

of a motion field itself without considering any image information. On the contrary, our context-aware model captures the relationship between an observed image and its motion field.

These prior works focused on modeling the distribution of an image's motion field using low-level statistics without any additional information. In real life, however, the distribution of motion fields is not independent of image content. For example, given an image of a car in front of a building, many would predict that the car is moving and the building is fixed. Thus, in our work, rather than modeling a motion prior as a context-free distribution, we propose to model the *conditional* motion distribution of future frames given an input image by incorporating a series of low- and high-level visual cues.

Motion or future prediction. Given an observed image or a short video sequence, models have been proposed to predict a future motion field [10], [11], [12], [13], [14], future trajectories of objects [2], [15], [16], [17], [18], or a future visual representation [19]. However, unlike in our proposed model, most of these works use a deterministic prediction model [11], [19], which cannot model the uncertainty of the future.

Concurrently, Walker *et al.* [14] identified the intrinsic ambiguity in deterministic prediction, and has proposed a probabilistic prediction framework. Our model is also probabilistic, but it directly predicts the pixel values rather than motion fields or image features.

Parametric image synthesis. Early work in parametric image synthesis mostly focused on texture synthesis using hand-crafted features [20]. More recently, works in image synthesis have begun to produce impressive results by training variants of neural network structures to produce novel images [21], [22], [23], [24]. Generative adversarial networks [6], [25], [26] and variational autoencoders [7], [8] have been used to model and sample from natural image distributions. Our proposed algorithm is also based on the variational autoencoder, but unlike previous works, we also model the temporal consistency between frames.

Video synthesis. Techniques that exploit the periodic structure of motion in videos have also been successful at generating novel frames from an input sequence. Schödl *et al.* proposed to shuffle frames from an existing video to generate a temporally consistent, looping image sequence [27]. This

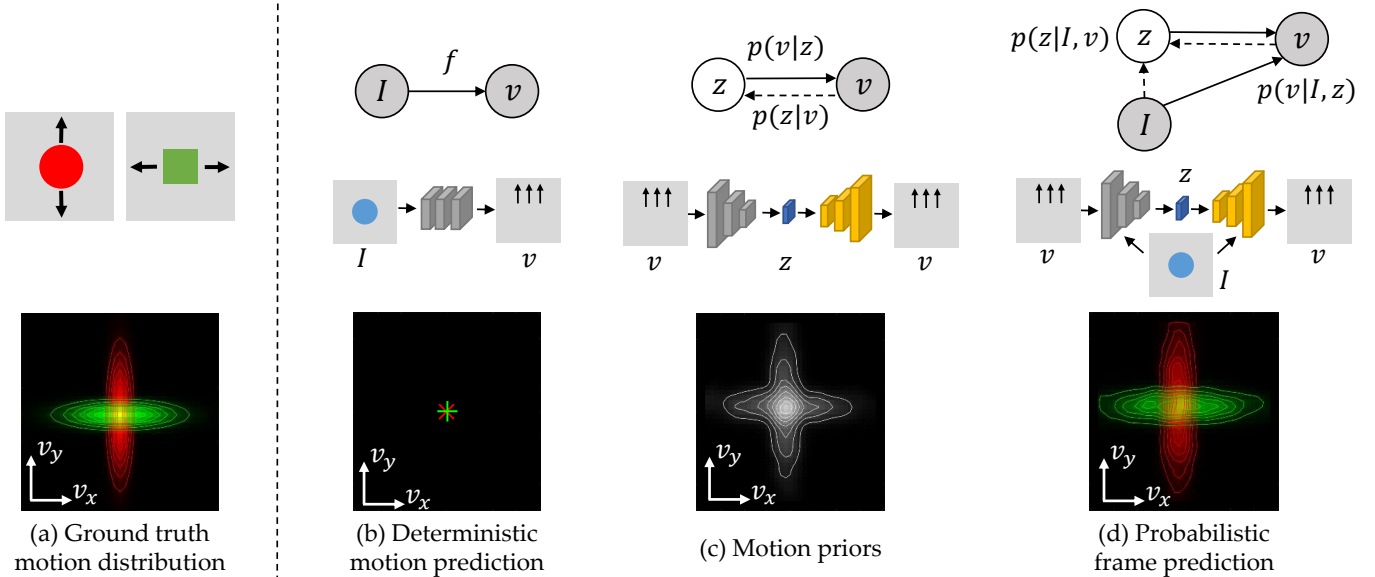| (a) Ground truth motion distribution | (b) Deterministic motion prediction | (c) Motion priors | (d) Probabilistic frame prediction |

Fig. 2. A toy example. Imagine a world composed of circles that move mostly vertically and squares that move mostly horizontally (a). We consider three different models (b-d) to learn the mapping from an image to a motion field. The first row shows graphical models, the second row shows corresponding network structures, and the third row shows estimated motion distributions. The deterministic motion prediction structure shown in (b) attempts to learn a one-to-one mapping from appearance to motion, but is unable to model multiple possible motions of an object, and tends to predict a mean motion for each object (the third row of (b)). The content-agnostic motion prior structure shown in (c) is able to capture a low-dimensional representation of motion, but is unable to leverage cues from image appearance for motion prediction. Therefore, it can only recovers the joint distribution of all objects (third row of (c)). The content-aware probabilistic motion predictor (d) brings together the advantages of models of (b) and (c) and uses appearance cues along with motion modeling to predict a motion field from a single input image. Therefore, the estimated motion distribution is very close to the ground truth (compare the last row and (a) and (d)).

idea was later used in video inpainting [28], and was extended to generate cinemagraphs [29] and seamlessly looping videos containing a variety of objects with different motion patterns [30], [31]. While these techniques are able to generate high-resolution and realistic-looking videos, they are often applicable only to videos with periodic motions, and they require a reference video as input. In contrast, we build an image generation model that does not require a reference video during testing.

Recently, several neural network architectures have been proposed to synthesize a new frame from observed frames. They infer the future motion either from multiple previous frames [1], [32], user-supplied action labels [33], [34], or directly model the joint distribution of all frames without conditioning on the input [35]. In contrast to these approaches, our network takes a single frame as input and learns the conditional distribution of future frames without any supervision.

3 FORMULATION

In this section, we first present a rigorous definition for the visual dynamics problem. Using a toy example, we then discuss three approaches to this problem, and show how the approach we take in our proposed model is more suitable for the task than the other two. We further present how our approach could be realized with a conditional variational autoencoder.

3.1 Problem Definition

In this section, we describe how to sample future frames from a current observation image. Here we focus on next

frame synthesis; given an RGB image I observed at time t, our goal is to model the conditional distribution of possible frames observed at time $t + 1$.

Formally, let $\{(I^{(1)}, J^{(1)}), \ldots, (I^{(n)}, J^{(n)})\}$ be the set of image pairs in the training set, where $I^{(i)}$ and $J^{(i)}$ are images observed at two consecutive time steps. Using this data, our task is to model the distribution $p_\theta(J|I)$ of all possible next frames J for a new, previously unseen test image I, and then to sample new images from this distribution (θ is the set of model parameters). In practice, we choose not to directly predict the next frame, but instead to predict the difference image $v = J - I$ between the observed frame I and the future frame J (also known as the Eulerian motion). The task is then to learn the conditional distribution $p_\theta(v|I)$ from a set of training pairs $\{(I^{(1)}, v^{(1)}), \ldots, (I^{(n)}, v^{(n)})\}$.

3.2 A Toy Example

To understand how to design a model to best characterize the conditional distribution of object motion, consider a simple toy world that only consists of circles and squares. Circles mostly move vertically, while squares mostly move horizontally. As shown in Fig. 2a, the ground truth distribution of a circle is $(v_x, v_y) \sim N((0, 0), (0.2, 1))$ and the distribution of a square is $N((0, 0), (1, 0.2))$, where $N(\vec{\mu}, \vec{\sigma})$ is a Gaussian distribution with mean equals to $\vec{\mu}$ and diagonal variation equal to $\vec{\sigma}$. Using this toy model, we discuss how each of the three models shown in Fig. 2b-d is able to infer the underlying motion.

Approach I: Deterministic motion prediction. In this structure, the model tries to find a deterministic relationship between the input image I and object motion v (Fig. 2b). In

our toy world, $I \in \{\text{circle}, \text{square}\}$ is simply the binary label of each possible object and v is a 2D motion vector[*].

In order to evaluate this model, we generate a toy training set which consists of 160,000 samples as follows. For each sample, we first randomly generate the object label I with equal probabilities of being circles or squares, and then sample the 2D motion vector of the object based on its label. The model is trained by minimizing the reconstruction error $\sum_i ||v^{(i)} - f(I^{(i)})||$ on this toy training set. The two other models we will soon introduce are also trained on this toy dataset.[†]

One drawback of this deterministic model is that it cannot capture the multiple possible motions that a shape can have. Essentially, the model can only learn the average motion of each object, I. The third row of Fig. 2b shows the estimated motion of both circles (the red cross) and squares (the green cross). Since the both circles and squares have zero-mean, symmetric motion distributions, this method predicts a nearly static motion field for each input image.

Approach II: Motion priors. A simple way to model the multiple possible motions of future frames is to use a variational autoencoder [7], as shown in Fig. 2c. This model contains a latent representation, z that encodes the intrinsic dimensionality of the motion fields. The network that learns this intrinsic representation z consists of two parts: an encoder network f that maps the motion field v to an intrinsic representation z (the gray network in Fig. 2c, which corresponds to $p(z|v)$), and a decoder network g that maps the intrinsic representation z to the motion field v (the yellow network, which corresponds to $p(v|z)$). During training, the network learns the latent representation z by minimizing the reconstruction error on the training set $\sum_i ||v^{(i)} - g(f(v^{(i)}))||$.

A shortcoming of this model is that the network does not see the input image when predicting the motion field. Therefore, it can only learn a joint distribution of both objects, as illustrated the third row of Fig. 2c. Thus, during test time, the model is not be able to disambiguate between the specific motion distribution of circles and squares.

Approach III: Probabilistic frame prediction. In this work, we combine the deterministic motion prediction structure (approach I) with a motion prior (approach II), to model the uncertainty in a motion field and the correlation between motion and image content. We extend the decoder in (2) to take two inputs, the intrinsic motion representation z and an image I (see the yellow network in Fig. 2d, which corresponds to $p(v|I, z)$). Therefore, instead of solely being able to model a joint distribution of motion v, it is now able to learn a conditional distribution of motion given the input image I.

In this toy example, since squares and circles move primarily in one (although different) direction, the intrinsic motion representation z only records the magnitude of motion along their major and minor directions. Combining the intrinsic motion representation with the direction of motion inferred from the image content, the model can

correctly model the distribution of motion. Fig. 2d shows that the inferred motion distribution of each object is quite similar to the ground truth distribution.

3.3 Conditional Variational Autoencoder

In this section, we will formally derive the training objective of our model, following the similar derivations [7], [8], [36]. Consider the following generative process that samples a future frame conditioned on an observed image, I. First, the algorithm samples the hidden variable z from a prior distribution $p_z(z)$; in this work, we assume $p_z(z)$ is a multivariate Gaussian distribution where each dimension is i.i.d. with zero-mean and unit-variance. Then, given a value of z, the algorithm samples the intensity difference image v from the conditional distribution $p_\theta(v|I, z)$. The final image, $J = I + v$, is then returned as output.

A variational upper-bound. In the training stage, the algorithm attempts to maximize the log-likelihood of the conditional marginal distribution $\sum_i \log p(v^{(i)}|I^{(i)})$. Assuming I and z are independent, the marginal distribution is expanded as $\sum_i \log \int_z p(v^{(i)}|I^{(i)}, z)p_z(z)dz$. Directly maximizing this marginal distribution is hard, thus we instead maximize its variational upper-bound [7]. Each term in the marginal distribution is upper-bounded by

$$\mathcal{L}(\theta, \phi, v^{(i)}|I^{(i)}) \approx - D_{\text{KL}}(q_\phi(z|v^{(i)}, I^{(i)})||p_z(z)) \\ + \frac{1}{L} \sum_{l=1}^{L} \left[\log p_\theta(v^{(i)}|z^{(i,l)}, I^{(i)}) \right], \quad (1)$$

where D_{KL} is the KL-divergence, $q_\phi(z|v^{(i)}, I^{(i)})$ is the variational distribution that approximates the posterior $p(z|v^{(i)}, I^{(i)})$, and $z^{(i,l)}$ are samples from the variational distribution. Recall that z and $I^{(i)}$ are independent, so that $p_z(z|I^{(i)}) = p_z(z)$. Please see Section 9 for detailed derivation of Eq. (1). For simplicity, we refer to the conditional data distribution, $p_\theta(\cdot)$, as the *generative model*, and the variational distribution, $q_\phi(\cdot)$, as the *recognition model*.

In practice, we always choose $L = 1$. Therefore, the upper bound of the KL-divergence can be simplified as

$$- D_{\text{KL}}(q_\phi(z|v^{(i)}, I^{(i)})||p_z(z)) + \log p_\theta(v^{(i)}|z^i, I^{(i)}). \quad (2)$$

If the assumption that I and z are independent does not hold, we can convert a latent variable that depends on I to one that does not, without affecting the expressiveness of our generative model, as suggested by Sohn *et al.* [36], [37].

Distribution reparametrization. We assume Gaussian distributions for both the generative model and recognition model[‡], where the mean and variance of the distributions are functions specified by neural networks, that is[§]

$$p_\theta(v^{(i)}|z^{(i,l)}, I^{(i)}) = \mathcal{N}(v^{(i)}; f_{\text{mean}}(z^{(i,l)}, I^{(i)}), \sigma^2 \mathbf{I}), \quad (3)$$

$$q_\phi(z^{(i,l)}|v^{(i)}, I^{(i)}) = \mathcal{N}(z^{(i,l)}; g_{\text{mean}}(v^{(i)}, I^{(i)}), g_{\text{var}}(v^{(i)}, I^{(i)})), \quad (4)$$

---

[*]. Although in practice we choose v to be the RGB intensity difference between consecutive frames ($v = I - J$), for this toy example we define v as the 2D motion vector.

[†]. The last row of Fig. 2 shows actual predictions by our model trained on this dataset.

[‡]. A complex distribution can be approximated as a function of a simple distribution, such as a Gaussian. This is referred to as the reparameterization trick [7].

[§]. Here the bold $\mathbf{I}$ denotes an identity matrix, whereas the normal-font I denotes the observed image.
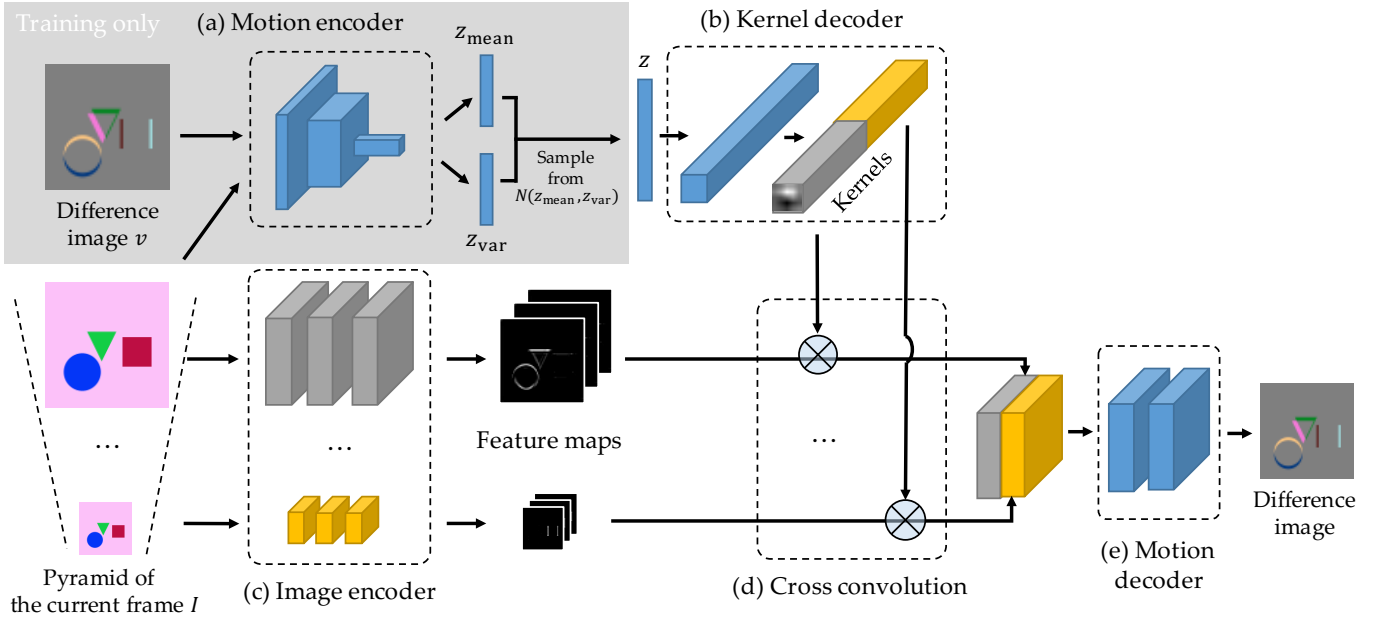
Fig. 3. Our network consists of five components: (a) a motion encoder, (b) a kernel decoder, (c) an image encoder, (d) a cross convolution layer, and (e) a motion decoder. Our image encoder takes images at four scales as input. For simplicity, we only show two scales in this figure. See Section 4 for details of our model. The motion encoder (the grayed region) is only used in training. At testing time, the motion vector z is sampled from its empirical distribution (see text for more details).

where $\mathcal{N}(\,\cdot\,;a,b)$ is a Gaussian distribution with mean a and variance b. f_{mean} is a function that predicts the mean of the generative model, defined by the generative network (the yellow network in Fig. 2d). g_{mean} and g_{var} are functions that predict the mean and variance of the recognition model, respectively, defined by the recognition network (the gray network in Fig. 2d). Here we assume that all dimensions of the generative model have the same variance σ^2, where σ is a hand-tuned hyperparameter.

The objective function. Plugging Eq. (3) and Eq. (4) in to Eq. (1), for $L = 1$ (only use one sample for each training iteration) we obtain the objective function that we minimize for each sample:

$$D_{\text{KL}}(q_\phi(z|v^{(i)}, I^{(i)})\|p_z(z)) + \lambda\|v^{(i)} - f_{\text{mean}}(z^{(i)}, I^{(i)})\|, \quad (5)$$

where $z^{(i)}$ is sampled from the distribution defined by Eq. (4), and λ is a constant. During training, we use stochastic gradient descent to minimize the variational lower bound defined in Eq. (5).

We will describe in Section 4 the neural networks that define the generative function f_{mean} and recognition function g_{mean} and g_{var}.

4 METHOD

We present an end-to-end trainable neural network structure, defining the generative function f_{mean} and recognition functions g_{mean}, and g_{var}. Once trained, these functions can be used in conjunction with an input image to sample future frames. We first describe our newly proposed cross convolutional layer, which naturally characterizes a layered motion representation [38]. We then explain our network structure and demonstrate how we integrate the cross convolutional layer into the network for future frame synthesis.

4.1 Layered Motion Representations and Cross Convolutional Networks

Motion can often be decomposed in a layer-wise manner [38]. Intuitively, different semantic segments in an image should have different distributions over all possible motions; for example, a building is often static, but a car moves.

To model layered motion, we propose a novel cross convolutional network (Fig. 3). The network first decomposes an input image pyramid into multiple feature maps (segments) through an image encoder (Fig. 3c). It then convolves these maps with different kernels (Fig. 3d), and uses the outputs to synthesize a difference image (Fig. 3e). This network structure naturally fits a layered motion representation, as each feature map characterizes an image *segment* and the corresponding kernel characterizes the motion of that segment. In other words, we model motions as convolutional kernels, which are applied to feature maps of images at multiple scales.

Unlike a traditional convolutional network, these kernels should not be identical for all inputs, as different images should be associated with different motions (kernels). We therefore propose a cross convolutional layer to tackle this problem. The cross convolutional layer does not learn the weights of the kernels itself. Instead, it takes both kernel weights and image segments as input and performs convolution during a forward pass; for back propagation, it computes the gradients with respect to both convolutional kernels and image segments.

The characteristics of a cross convolutional layer naturally fit the layered motion representation, as we can think of each feature map as an image segment, and the corresponding kernel characterizes the layer's motion. In other words, we model motions as convolutional kernels, which are applied to image segments (layers) at multiple scales. Concurrent

papers [34], [39] have also explored similar ideas. While they applied the learned kernels on input images, we jointly learn image segments and kernels without direct supervision.

4.2 Network Structure

As shown in Fig. 3, our network consists of five components: (a) a motion encoder, which is a variational autoencoder that learns the compact representation, z, of possible motions; (b) a kernel decoder, which learns the motion kernels from the compact motion representation z; (c) an image encoder, which consists of convolutional layers extracting segments from the input image I; (d) a cross convolutional layer, which takes the output of the image encoder and the kernel decoder, and convolves the image segments with motion kernels; and (e) a motion decoder, which regresses the difference image from the combined feature maps. We now introduce each part in detail.

During training, our motion encoder (Fig. 3a) takes the current frame and a difference image as input, both at resolution 128×128. The network then applies six 5×5 convolutional and batch normalization layers (number of channels are {96, 96, 128, 128, 256, 256}) to the concatenated images, with some pooling layers in between. The output has a size of 256×5×5. The kernel encoder then reshapes the output to a vector, and splits it into a 3,200-dimension mean vectors z_{mean} and a 3,200-dimension variance vector z_{var}, from which the network samples the latent motion representation $z \sim N(z_{\mathrm{mean}}, z_{\mathrm{var}})$. The motion encoder takes the current frame as input, in addition to the motion image, so that it can learn to model the *conditional* variational distribution ($q_\theta(\cdot)$ in Eq. (5)).

Next, the kernel decoder (Fig. 3b) sends the 3,200 = 128×5×5 tensor into two additional convolutional layers, each with 128 channels and a kernel size of 5. They are then split into four sets, each with 32 kernels of size 5×5.

Our image encoder (Fig. 3c) operates on four different scaled versions of the input image I (256×256, 128×128, 64×64, and 32×32)¶. At each scale, there are four sets of 5×5 convolutional and batch normalization layers (number of channels are {64, 64, 64, 32}), two of which are followed by a 2×2 max pooling layer. Therefore, the output size of the four channels are 32×64×64, 32×32×32, 32×16×16, and 32×8×8, respectively. This multi-scale convolutional network allows us to model both global and local structures in the image, which may have different motions.

The core of our network is a cross convolutional layer (Fig. 3d), which, as discussed in Section 4.1, applies the kernels learned by the kernel decoder to the feature maps (layers) learned by the image encoder. The cross convolutional layer has the same output size as the image encoder.

Our motion decoder (Fig. 3e) starts with an up-sampling layer at each scale, making the output of all scales of the cross convolutional layer have a resolution of 64×64. This is then followed by one 9×9 and two 1×1 convolutional and batch normalization layers, with {128, 128, 3} channels. These final feature maps (layers) are then used to regress the output difference image.

¶. For the input image of size 128×128, we used five different scales instead. In that case, the size of motion vector is 4,000 ($= 5 \times 5 \times 32 \times 5$).

Training and testing details. During training, the image encoder takes a single frame $I^{(i)}$ as input, and the motion encoder takes both input frame $I^{(i)}$ and the difference image $v^{(i)} = J^{(i)} - I^{(i)}$ as input, where $J^{(i)}$ is the next frame. The network aims to regress the difference image that minimizes the objective function Eq. (5).

During testing, the image encoder still sees a single image I; however, instead of using a motion encoder, we directly sample motion vectors $z^{(j)}$ from the prior distribution $p_z(z)$ (therefore, the gray part in Fig. 3 is not used in testing). In practice, we use an empirical distribution of z over all training samples as an approximation to the prior, a.k.a. the variational distribution $q_\phi(z)$ in the literature, as Doersch *et al.* [40] show that this sampling strategy works better than sampling from the prior distribution $p_z(z)$. Our sampling of z is independent of the input image I, satisfying the independence assumption discussed in Section 3.3. The network then synthesizes possible difference images $v^{(j)}$ by taking the sampled latent representation $z^{(j)}$ and an RGB image I as input. We then generate a set of future frames $\{J^{(j)}\}$ from these difference images: $J^{(j)} = I + v^{(j)}$.

5 EVALUATIONS

We now present a series of experiments to evaluate our method. We start with a dataset of 2D shapes, which serves to benchmark our model on objects with simple, yet nontrivial, motion distributions. Following Reed *et al.* [41], we then test our method on a dataset of video game sprites with diverse motions. In addition to these synthetic datasets, we further evaluate our framework on real-world video datasets. Again, note that our model uses consecutive frames for training, requiring no supervision. Visualizations of our experimental results are also available on our project page‖.

5.1 Movement of 2D Shapes

We first evaluate our method using a dataset of synthetic 2D shapes. This dataset serves to benchmark our model on objects with simple, yet nontrivial, motion distributions. It contains three types of objects: circles, squares, and triangles. Circles always move vertically, squares horizontally, and triangles diagonally. The motion of circles and squares are independent, while the motion of circles and triangles are correlated (when the triangle moves up, the circle moves down). The shapes can be heavily occluded, and their sizes, positions, and colors are chosen randomly. There are 20,000 pairs for training, and 500 for testing.

Fig. 4 shows the results. Fig. 4a and Fig. 4b show a sample of consecutive frames in the dataset, and Fig. 4c shows the reconstruction of the second frame after encoding and decoding with the ground truth image pairs. Fig. 4d and Fig. 4e show samples of the second frame; in these results the network only takes the first image as input, and the compact motion representation, z, is randomly sampled. Note that the network is able to capture the distinctive motion pattern for each shape, including the strong correlation of triangle and circle motion.

To quantitatively evaluate our algorithm, we compare the displacement distributions of circles, squares, and triangles

‖. Our project page: http://visualdynamics.csail.mit.edu

(a) Input (b) Next frame (c) Reconstruction (d) 3 sampled future frames (e) Zoomed-in views
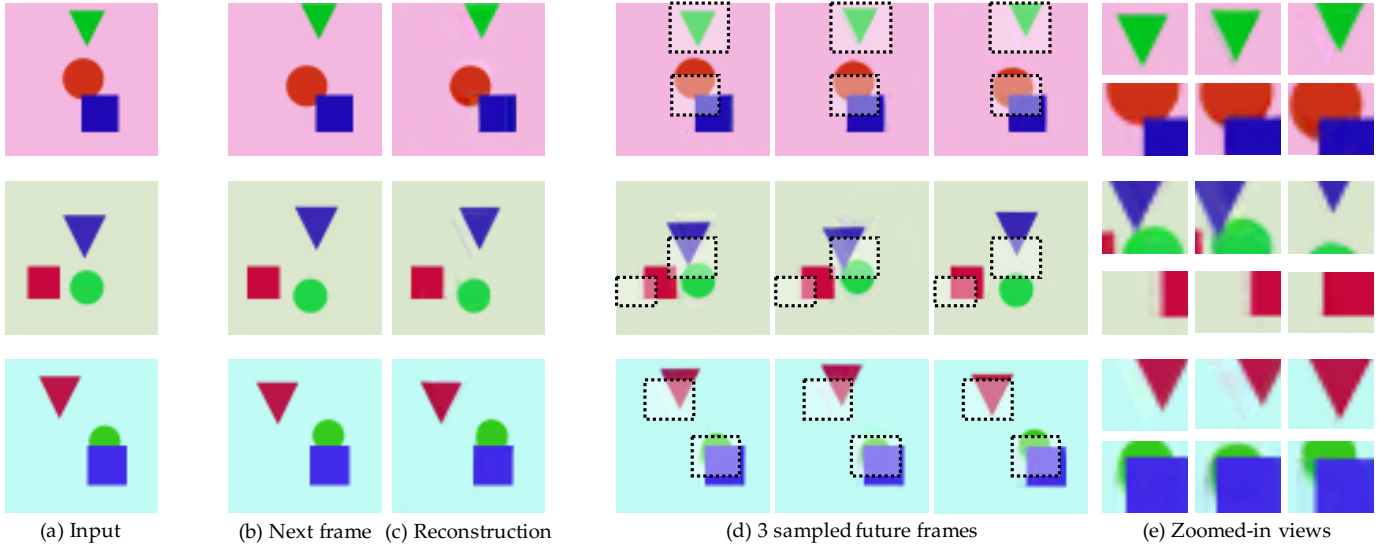
Fig. 4. Results on the *Shapes* dataset containing circles, squares, and triangles. For a given frame (a) our goal is to predict probable motion. In (b) we show the ground truth future frame. Notice how squares move horizontally, circles vertically, triangles diagonally, and the triangle's motion is correlated with the circle's. Our model is able to reconstruct the motion (c) after encoding and decoding with the ground truth image pairs. By sampling from the latent representation, we can also synthesize additional novel future frames with probable motion (d). In (e), we show zoomed-in regions for these samples. Note the significant variation among the sampled frames.

Method	Shapes			
	Circles	Squares	Triangles	Circles-Triangles
Flow	6.77	7.07	6.07	8.42
AE	8.76	12.37	10.36	10.58
Ours	**1.70**	**2.48**	**1.14**	**2.46**

KL divergence, $D_{KL}(p_{gt} \| p_{pred})$, between predicted and ground truth distributions
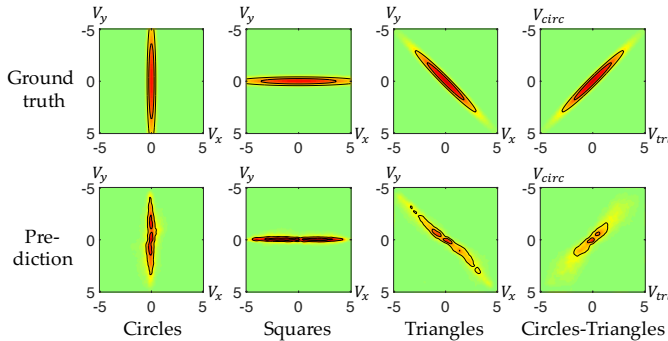
Fig. 5. The motion of the sampled data is consistent with the ground truth motion distributions. Left: for each object, comparison between its ground-truth motion distribution and the distribution predicted by our method. It shows the network learns to move circles vertically, squares, horizontally, and the motion of circles and triangles is correlated. Right: KL divergence between ground-truth distributions and distributions predicted by three different algorithms. Our network scores much better than a simple nearest-neighbor motion transfer algorithm.

in the sampled images with their ground truth distributions. We sample 50,000 images and use the optical flow package by Liu *et al.* [42] to calculate the mean movement of each object. We plot them in Fig. 5 as well as the isolines using the *contour* function in MATLAB.

We also compute their KL-divergence. Here, we divide the region $[-5, 5] \times [-5, 5]$ into 41×41 bins and approximate the predicted distribution with the 2D histogram. We compare our algorithm with a simple baseline that copies the optical flow field of the closest image pairs from the training set ('Flow' in Fig. 5); for each test image, we find its 10-nearest neighbors in the training set (the retrieval is based on ℓ^2-distance between query image and images in the training dataset), and randomly transfer one of the corresponding optical flow fields. To illustrate the advantage of using a variational autoencoder (VAE) over a standard autoencoder, we also modify our network by removing the KL-divergence loss and sampling layer ('AE' in Fig. 5). Fig. 5 shows our predicted distribution is very close to the ground-

truth distribution. It also shows that a VAE helps to capture the true distribution of future frames.

5.2 Movement of Video Game Sprites

We evaluate our framework on a video game sprites dataset**, also used by Reed *et al.* [41]. The dataset consists of 672 unique characters; for each character, there are 5 animations (spellcast, thrust, walk, slash, shoot) from 4 different viewpoints. The length of each animation ranges from 6 to 13 frames. We collect 102,364 pairs of neighboring frames for training, and 3,140 pairs for testing. The same character does not appear in both the training and test sets. Sampled future frames are shown in Fig. 6. From a single frame, our method captures various possible motions that are consistent with those in the training set.

As a quantitative evaluation on the success rate of our image synthesis algorithm, we conduct behavioral experiments

**. Liberated pixel cup: http://lpc.opengameart.org

Method	Resolution	
	32×32	64×64
Flow	29.7	21.0
Ours	**41.2**	**35.7**

Percentages (%) of results labeled as real by human subjects

(a) Input   (b) Next frame   (c) Reconstruction   (d) 2 sampled future frames   (e) Zoomed-in views
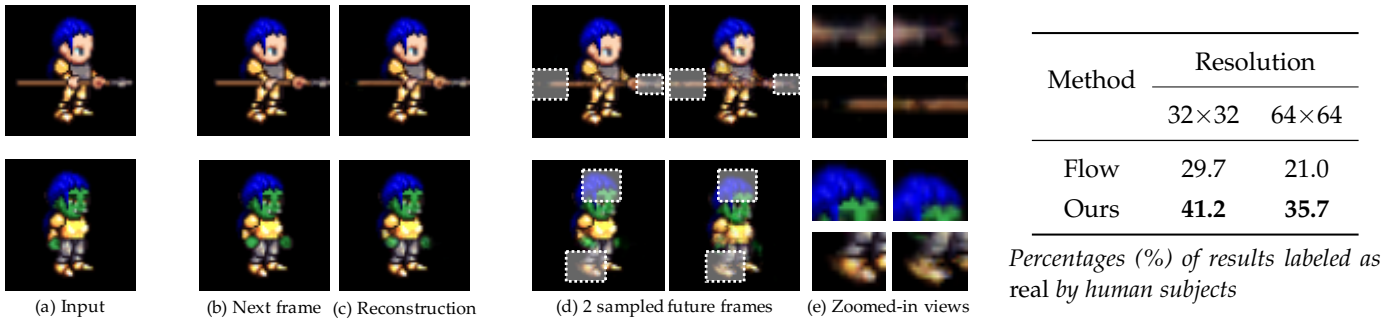
Fig. 6. Left: Results on the *Sprites* dataset, where we show input images (a), ground truth next frames (b), our reconstruction (c), two sampled future frames (d), and corresponding zoomed-in views (e). Right: Percentages (%) of synthesized results that were labeled as real by human subjects in two-way forced choices on Amazon Mechanical Turk, at resolution 32×32 and 64×64. A perfect algorithm would achieve a percentage around 50%.

Method	Resolution	
	32×32	64×64
Flow	31.3	25.5
Ours	**36.7**	**31.3**

Percentages (%) of results labeled as real by human subjects

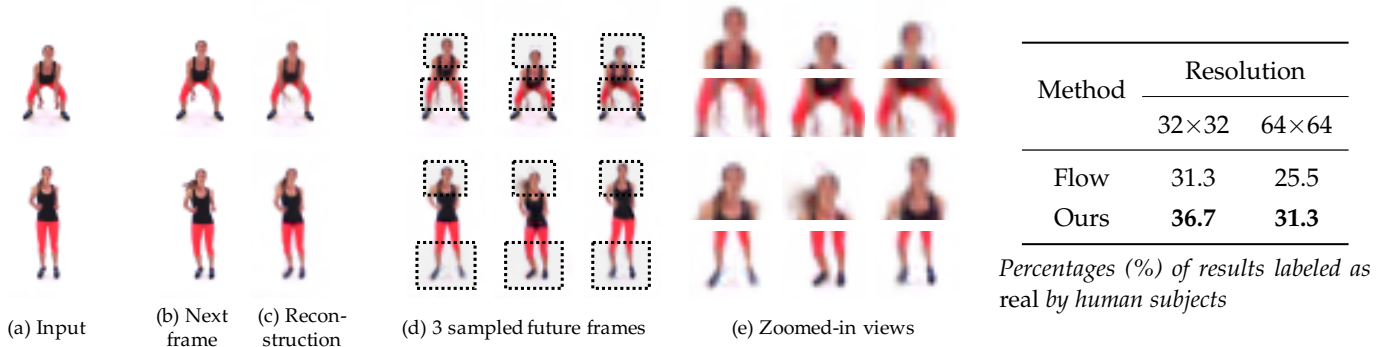(a) Input   (b) Next frame   (c) Reconstruction   (d) 3 sampled future frames   (e) Zoomed-in views

Fig. 7. Left: Results on *Exercise* dataset, where we show input images (a), ground truth next frames (b), our reconstruction (c), three sampled future frames (d), and corresponding zoomed-in views (e). Right: Percentages (%) of synthesized results that were labeled as real by human subjects in two-way forced choices on Amazon Mechanical Turk, at resolution 32×32 and 64×64. A perfect algorithm would achieve a percentage around 50%.

on Amazon Mechanical Turk. We randomly select 200 images, sample a possible next frame using our algorithm, and show them to multiple human subjects as an animation side by side with the ground truth animation. We then ask the subject to choose which animation is real (not synthesized). An ideal algorithm should achieve a success rate of 50%. In our experiments, we present the animation in both the original resolution (64×64) and a lower resolution (32×32). We only evaluate on subjects that have a past approval rate of >95% and have also passed our qualification tests. Fig. 6 shows that our algorithm significantly outperforms a baseline that warps the input by transferring one of closest flow fields from the training set. Our results are labeled as *real* by humans 41.2% of the time at 32×32, and 35.7% of the time at 64×64. Subjects are less easily fooled by 64×64 images, as it is harder to hallucinate realistic details in high-resolution images.

5.3 Movement in Real Videos Captured in the Wild

To demonstrate that our algorithm can also handle real videos, we collect 20 workout videos from YouTube, each about 30 to 60 minutes long. We first apply motion stabilization to the training data as a pre-processing step to remove camera motion. We then extract 56,838 pairs of frames for training and 6,243 pairs for testing. The training and testing pairs come from different video sequences.

Fig. 7 shows that our framework works well in predicting the movement of the legs and torso. Specifically, the algorithm is able to synthesize reasonable motions of the human in various poses. The Mechanical Turk behavioral

experiments also show that the synthesized frames are visually realistic. In particular, our synthesized images are labeled as real by humans 36.7% of the time at a resolution of 32×32, and 31.3% of the time at 64×64.

Given an input frame, how often can our algorithm sample realistic motion? We have run an additional human study to evaluate this: for each of 100 randomly selected test images, we sample 100 future frames at 128×128, and ask AMT subjects whether the two consecutive frames, one real and one synthesized, characterize realistic human motion. The success rate is 44.3% (averaged over the 100 test images), with a standard deviation of 4.3%.

Synthesizing images with realistic human motion is challenging. Traditional methods often require a high quality 3D model of the human body for real-time synthesis and rendering [44]. Although our algorithm does not require a 3D model, it can still simulate how a human moves between frames. Note that synthesized large motions in our results often happen around locations corresponding to skeletal joints, implying that our model has an implicit understanding of the structure and correlation between body parts.

At last, we test our algorithm on realistic videos in the wild. We use the PennAction dataset [43], which contains 981 sequences of diverse human motion with complex backgrounds. We extract 7,705 pairs of frames for training and 987 pairs for evaluation. To remove trivial panning motions, we detect the human in each frame [45] and crop each frame to center the subject.

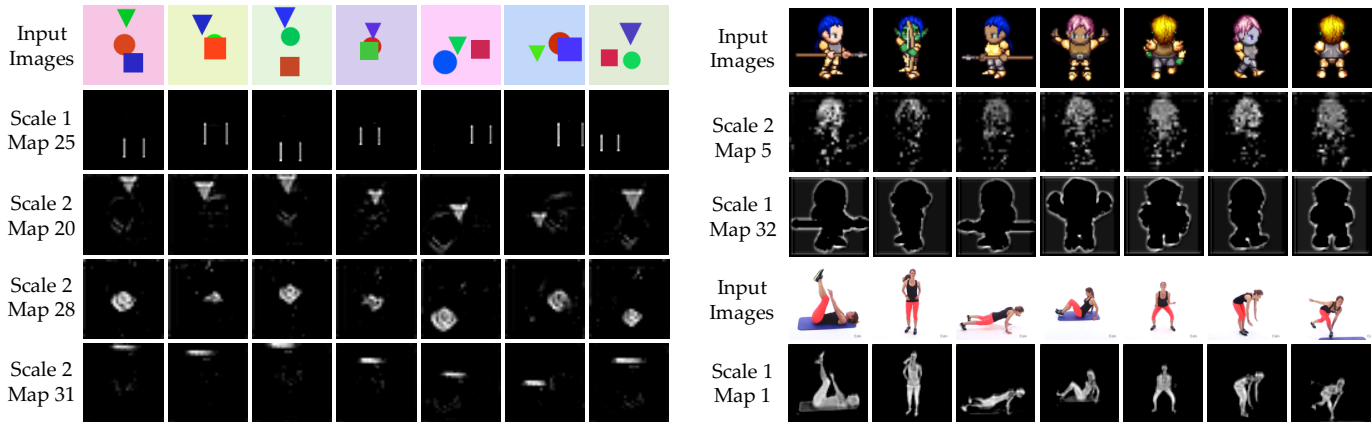The results are shown in Fig. 9. While in-the-wild videos

Fig. 8. Learned layers on the Shapes dataset (left), the Sprites dataset (top right), and the Exercise dataset (bottom right). Our system is able to implicitly discover semantic structure from this self-supervised task. On the Shapes dataset, it learns to detect circles and triangles; it also detects vertical boundaries of squares, as squares always move horizontally. On the Exercise dataset, it learns to respond to only humans, not the carpet below them, as the carpet never moves.
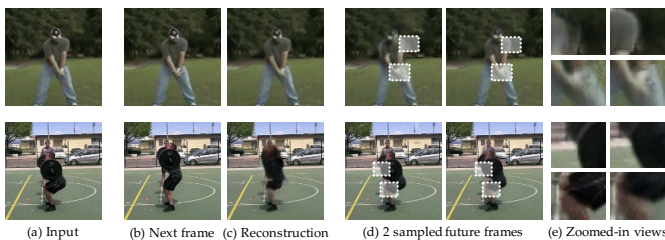

Fig. 9. Sampling results on the PennAction dataset [43], where we show input images (a), ground truth next frames (b), our reconstruction (c), three sampled future frames (d), and corresponding zoomed-in views (e).

TABLE 1
Statistics of the 3,200-dimensional motion vector z

Dataset	Shapes	Sprites	Exercise
Non-zero element in z_{mean}	299	54	978
Dominating PCA components	5	5	47

The network learns a sparse latent representation, encoding high-level knowledge using minimal bits.

are more challenging, our model synthesizes multiple realistic future frames from a single image. Especially, when synthesizing future frames, our model keeps the original background intact, suggesting the learned motion representation separates the region of interest from the background. In the future, we aim to develop prediction models that are able to better handle more complex visual scenes.

6 ANALYSES

In this section, we present an analysis of the learned network to demonstrate what it captures. In particular, we visualize the learned feature maps (Section 6.1), and show that the network implicitly learns object and edge detectors. Additionally, we compute statistics of the latent representation (Section 6.2) to verify that the network is learning a compact, informative representation of the motion manifold. We then visualize certain dimensions of the latent representation to reveal their meanings (Section 6.4).

6.1 Visualizing Learned Layers

Our network synthesizes the movement of objects by first identifying layers in the input and transferring each layer. Therefore, we expect that the these layers carry both low-level information such as object contours, and high-level information such as object parts.

To verify this, we visualize the learned feature maps (Fig. 3b) in Fig. 8. Even without supervision, our network learns to detect objects or contours in the image. For example, we see that the network automatically learns object detectors and edge detectors on the shapes dataset. It also learns a hair detector and a body detector on the sprites and exercise datasets, respectively. The part detectors have a sharper boundary on the shapes dataset than on the other two. This is because the shapes dataset has well-defined concepts of independently movable parts; in contrast, body parts in real videos have indistinguishable motion distributions.

6.2 The Sparsity of the Latent Representation z

Although our latent motion representation, z, has 3,200 dimensions, its intrinsic dimensionality is much smaller. We show statistics in Table 1. There are two main messages. First, z_{mean} is very sparse. There are 299 non-zero elements of z_{mean} in the shapes dataset, 54 in sprites, and 978 in exercise. The sprites dataset requires fewer non-zero elements than the shapes dataset, because while the visual appearance of the characters are more complex, their motion falls into a few predefined categories (*e.g.*, thrust) and is thus simpler than the continuous motion of the shapes. Second, the independent components of z are even fewer. We run principle component analysis (PCA) on the z_{mean}s obtained from a set of training images, and find that for each dataset, only a small fraction of components cover 95% of the variance in z_{mean} (5 in shapes, 5 in sprites, and 47 in exercise).

Fig. 10 further shows detailed statistics of z_{mean} and z_{logvar} (a.k.a., $\log(z_{\mathrm{var}})$). For each dataset, we randomly select 1,000 train samples and calculate the corresponding z_{mean} and
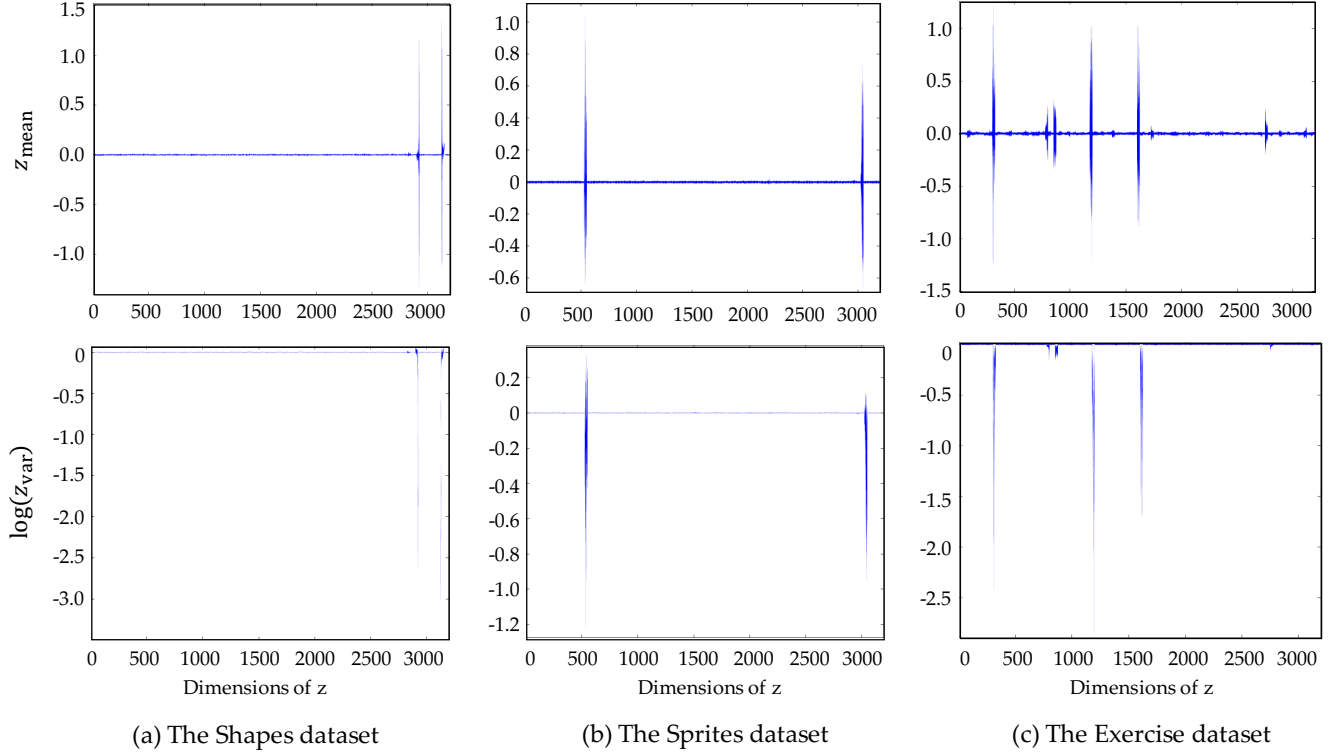
(a) The Shapes dataset   (b) The Sprites dataset   (c) The Exercise dataset

Fig. 10. Statistics of latent vectors z_{mean} and z_{logvar} extracted from 1,000 image pairs from the Shapes, Sprites, and Exercise datasets, respectively. Each vertical line is for a single dimension in z. Although the z vector has 3,200 dimensions, only a small number of those have values deviating from the prior (mean 0, log-variance 0). This shows the system is learning to encode motion information in a compact way. Also, as the motion in the Exercise dataset is more complex, the system needs more bits for encoding (*c.f.*, Table 1).

z_{logvar} through the motion encoder (Fig. 3a). The first row shows the distribution of z_{mean} and the second row shows the distribution of z_{logvar}. The x-axis in each figure corresponds to the 3,200 dimensions in z_{mean} and z_{logvar}. For each dimension k, the blue region reflects the interval

$$[\mathrm{mean}(z_k) - \mathrm{std}(z_k), \mathrm{mean}(z_k) + \mathrm{std}(z_k)], \qquad (6)$$

where $\mathrm{mean}(z_k)$ is the mean of the 1,000 values in the k dimension for the 1,000 samples, and $\mathrm{std}(z_k)$ is the standard deviation.

One interesting observation from Fig. 10 is that, for most dimensions, z_{mean} is very close to 0 and z_{var} is very close to 1 (z_{logvar} equals to 0). This is because the training object Eq. (1) minimizes the KL divergence between $N(\vec{0}, \boldsymbol{I})$ and $N(z_{\mathrm{mean}}, \mathrm{diag}(z_{\mathrm{var}}))^{\dagger\dagger}$, and the minimizer of the KL divergence is $z_{\mathrm{mean}} = \vec{0}$ and $z_{\mathrm{var}} = \vec{1}$.

However, not all dimensions of z_{mean} and z_{logvar} are 0, and those non-zero dimensions actually encode the semantic information of how objects move between frames. Recall that to calculate the compact motion representation z, the motion encoder first estimates z_{mean} and z_{var} from two input frames, and then samples z from $N(z_{\mathrm{mean}}, \mathrm{diag}(z_{\mathrm{var}}))$. Let us rewrite the sampling as $z_k = z_{\mathrm{mean},k} + \epsilon_k z_{\mathrm{var},k}$, where k is a dimension index and ϵ_k is a random Gaussian noise with zero mean and unit variance. If $z_{\mathrm{var},k}$ is large, the corresponding dimension in the motion vector z is mostly corrupted by the random Gaussian noise, and only those dimensions with

††. $\mathrm{diag}(z_{\mathrm{var}})$ denotes a diagonal matrix whose diagonal elements are z_{var}.

small $z_{\mathrm{var},k}$ are able to transmit motion information for next frame synthesis.

In other words, z_{mean} carries the actual motion information, while z_{var} is an indicator of whether a dimension is being actively used. This corresponds to our observation in Fig. 10, where only uncorrupted dimensions (*i.e.*, those with small z_{var}) have non-zero z_{mean}. Similar discussions were also presented by Hinton and van Camp [46] and concurrently by Higgins *et al.* [47].

To further demonstrate how KL-divergence criterion ensures the sparsity of motion vector z, we also vary the weight of KL-divergence criterion λ during training. As shown in Fig. 11, when λ is small, most of dimensions of $\log(z_{\mathrm{var}})$ are smaller than 0, and z_{mean} is not sparse (left of Fig. 11). When we increase λ, the network is forced to encode the information in a more compact way, and the learned representation becomes sparser (right of Fig. 11).

All these results suggest that our network has learned a compact representation of motion in an unsupervised fashion, and encodes high-level knowledge using a small number of bits, rather than simply memorizing training samples. In the next subsection, we will also illustrate what motion information is actually learned.

6.3 Varying the Size of the Latent Representation

We have explored the sparsity of the latent representation z, suggesting the model learns to encode essential information using only a few bits, despite z itself has 3,200 dimensions. Here, we conduct an additional ablation study to understand
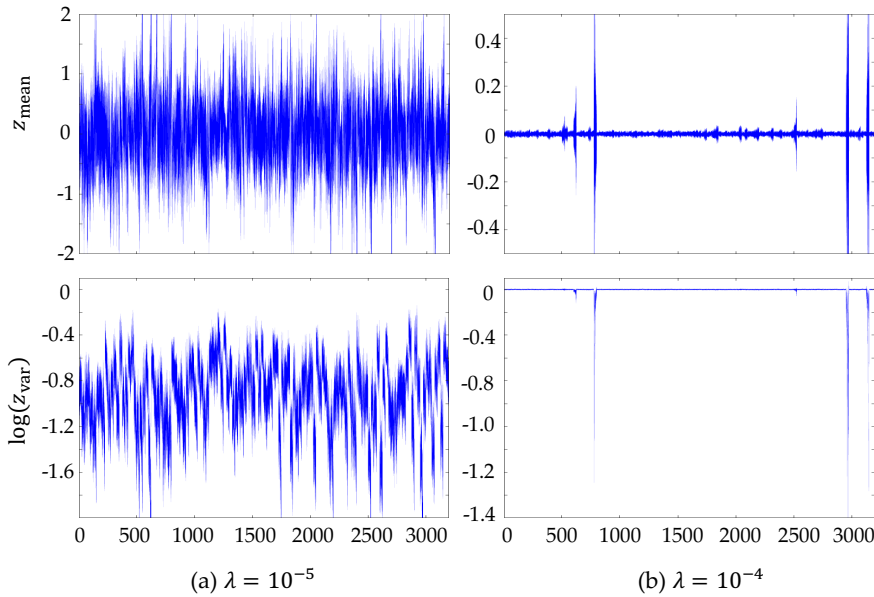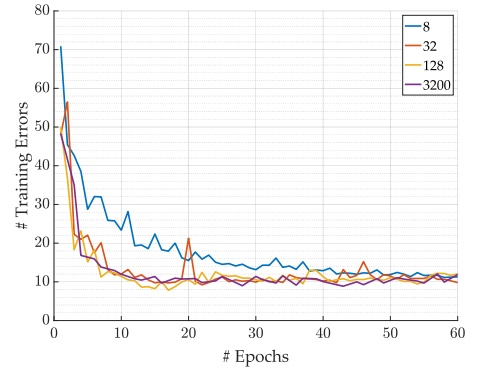
Fig. 11. Statistics of latent vectors z extracted from 1,000 image pairs from the Shapes dataset with networks learned using different values of λ. The latent vector becomes sparser when λ is larger, *i.e.*, the network is encoding motion in a more compact way.

Size of z	8	32	128	3,200
Dominating PCA components	3	3	3	5

Fig. 12. Ablation study on the size of z. Our model automatically discovers the underlying dimension of the motion in the *Shapes* dataset. When we shrink the size of z, the convergence gets a little slower (especially when z has only 8 dimensions), but the results are essentially the same.
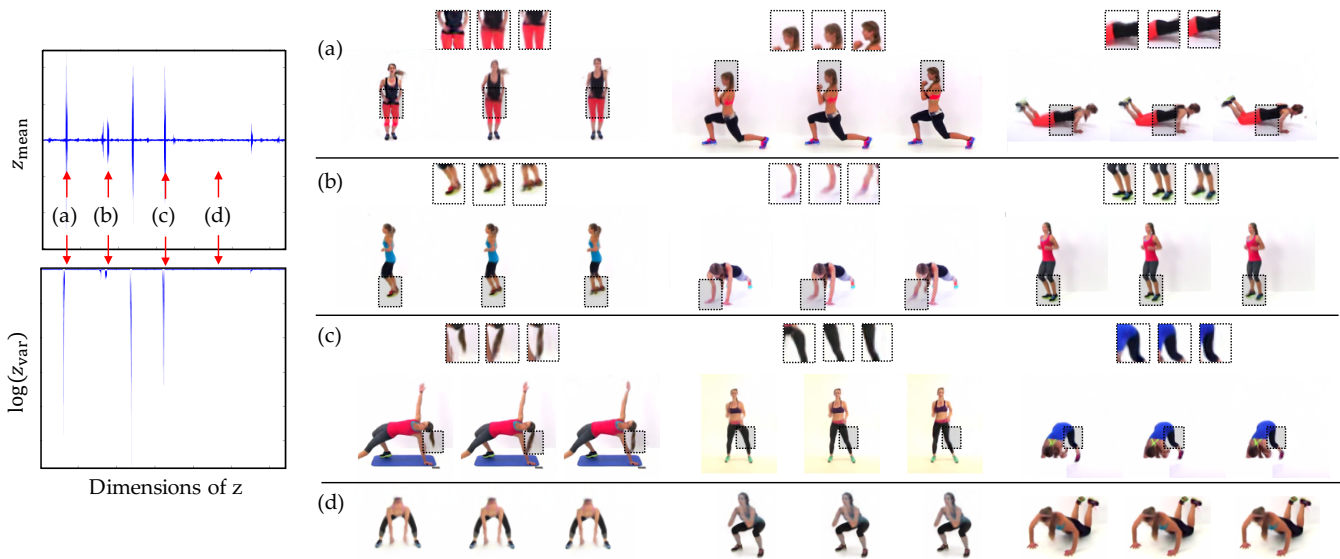

Fig. 13. We visualize the effect of varying individual dimensions in the latent representation z, revealing the system is learning a disentangled, interpretable motion representation. For the dimensions whose log-variance is smaller than 0 (a–c), they record a certain type of motion. For example, dimension (a) corresponds to humans move upwards, dimensions (b) and (c) correspond to moving arms, hair, or legs to the left. For the dimensions whose log-variance is very close to 0, they record no motion information: changing the value of dimension (d) results in no motion in predicted frames.

the minimal number of dimensions that are necessary for the network to initialize the learning process.

We experiment on the *Shapes* dataset. Table 1 tells us that our model can learn to encode motion information using only 5 dominating PCA components. We therefore explore 4 different sizes of z: 8, 32, 128, and 3,200, as shown in Fig. 12. All networks converge regardless of the dimensions of z, with a slight difference in convergence time (the network with smaller z takes longer to converge). Also, the number of dominating PCA components is always small (3–5), suggesting that as long as the dimension of the latent representation z is larger than its intrinsic dimension,

the model consistently learns to encode the information compactly, regardless of its dimension.

6.4 Visualizing the Latent Representation

We visualize the encoded motion by manipulating individual dimensions of the representation z, while keeping other dimensions constant. Through this, we have found that each dimension corresponds to a certain type of motion. We show results in Fig. 13. On the exercise dataset, varying one dimension of z causes the girl to stand-up, and varying another causes her to move her leg. The effect is consistent across input images, showing individual dimensions in the

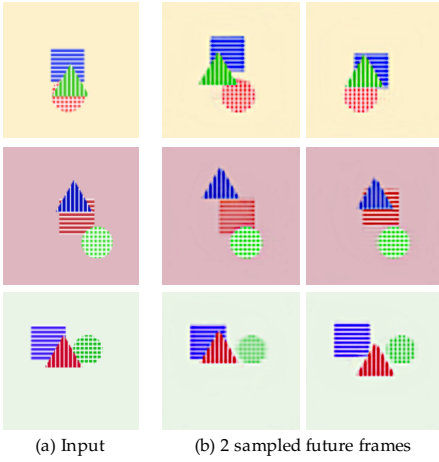(a) Input (b) 2 sampled future frames

Fig. 14. Our model handles disocclusions well. On the *Shapes+Texture* dataset, our model is able to complete shapes with their corresponding texture after hallucinating their motion.
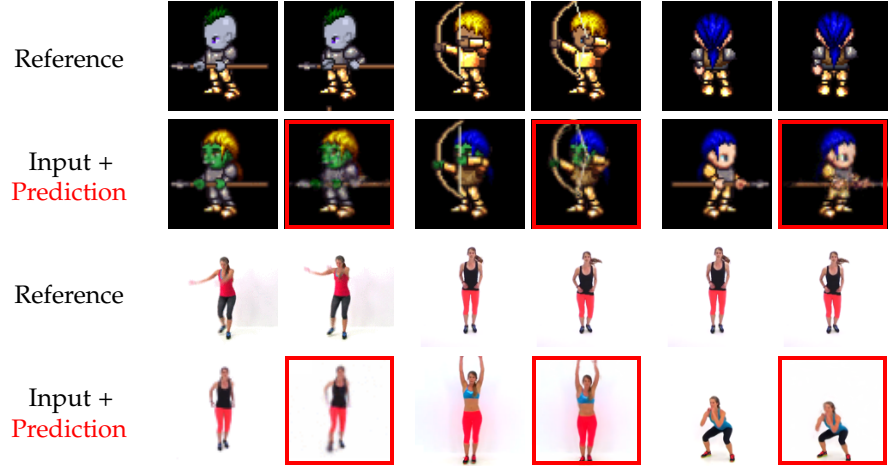

Fig. 15. Results on visual analogy-making, where we want to transfer the motion in a reference pair to a target image. We conduct experiments on two datasets, Sprites and Exercise, and mark the predicted frames in red. Our algorithm is able to transfer high-level motion (*e.g.* downwards motion of a human body, as opposed to pixel-wise motion) in a semantically plausible way.

TABLE 2
Mean squared pixel error on test analogies, by animation

Model	spellcast	thrust	walk	slash	shoot	average
Add	41.0	53.8	55.7	52.1	77.6	56.0
Dis	40.8	55.8	52.6	53.5	79.8	56.5
Dis+Cls	13.3	24.6	17.2	**18.9**	40.8	23.0
Ours	**9.5**	**11.5**	**11.1**	28.2	**19.0**	**15.9**

The first three models (Add, Dis, and Dis+Cls) are from Reed et al. [41]. Compared with them, our model achieves lower errors.

latent vector z carries abstract, higher-level knowledge. Also notice that only dimensions with smaller variance z_{var} contains semantic motion information (Fig. 13a–c). Manipulating dimensions with variances close to 1 results in no significant motion (Fig. 13d).

6.5 Handling Disocclusions

Synthesizing future frames often involves handling disocclusions. Here we systematically evaluate it on a new dataset, *Shapes+Texture*, where the primitives in the *Shapes* dataset now have horizontal (squares), vertical (triangles), or checkerboard patterns (circles). Fig. 14 shows our model handles disocclusions well, correctly synthesizing the object texture even if it's not visible in the input. This is further supported by Fig. 8, showing the network learns feature maps that corresponding to *amodal* segments of shapes.

7 APPLICATIONS

Our framework models a general problem and therefore has a wide range of applications. Here we present two possible applications of our framework: visual analogy-making and extrapolation for generating video sequences.

7.1 Zero-Shot Visual Analogy-Making

Recently, Reed *et al.* [41] studied the problem of inferring the relationship between a pair of reference images and synthesizing an image analogy by applying the inferred relationship to a test image. For example, the character shown in the top row of Fig. 15a leans toward to the right; the task is to transfer its leaning motion to the target, bottom-left image.

The method by Reed *et al.* requires a set of quadruples during supervision (two source images and two target images). In contrast, our network is able to preform this task without first training using the quadruple sets. Specifically, we extract the motion vector, z, from two reference frames using our motion encoder (Fig. 3a). We then use the extracted motion vector z to synthesize an analogy-image given a new test image. In this way, our network learns to transfer motion from a source pair to the target image, without requiring any form of supervision (see Fig. 15). As shown in Table 2, our algorithm out-performs that by Reed *et al.* [41].

7.2 Extrapolation

Our network can also be used to generate video sequences by extrapolation. In Fig. 16, we synthesize videos from a single image by simply repeatedly applying the sample motion. Given an input image frame I_1 and sampled motion vector z, we first synthesize the second frame I_2 from the first frame I_1 and motion vector z, and then synthesize the third frame I_3 from the second using the same motion vector. We see that our framework generates plausible video sequences. Recent work on modeling transitions across possible motions [48], could serve as alternative way to extend our framework to longer-term video generation. Compared with most of previous deterministic video synthesis networks [1], [32], our approach can sample multiple possible ways a person can move, as shown Fig. 16. One limitation of this approach is that artifacts gradually emerge over time. It may be possible to reduce these artifacts using a learned image prior [49].

8 CONCLUSION

In this paper, we have proposed a novel framework that samples future frames from a single input image. Our

Fig. 16. Generating video sequences by repeatedly applying a sampled motion representation z. Our model is able to synthesize reasonable videos of human moving in various ways, though artifacts gradually emerge over time (*e.g.* the thighs become bigger in the top-right example).

method incorporates a variational autoencoder for learning compact motion representations and layer-based synthesis algorithm to generate realistic movement of objects. We have demonstrated that our framework works well on both synthetic and real-life videos.

The key component of our frame synthesis model is to decompose an input image into different layers and model the movement of each layer through a simple convolutional network. This motion can well model the motion of a single deformable object, like human body. In the future, we would also like to extend it to handle more complicated motion and stochastic motion, *e.g.*, water flowing.

Current results also suggest that our probabilistic visual dynamics model may be useful for additional applications, such as inferring objects' higher-order relationships by examining correlations in their motion distributions. Furthermore, this learned representation could be potentially used as a sophisticated motion prior in other computer vision and computational photography applications.

9 VERIFICATION OF EQ. (1) AND EQ. (5)

In this section, we formally derive how we obtain the training objective function in Eq. (1), following similar derivations in [7], [8], [36]. As mentioned in Section 3.3, the generative process that samples a difference image v from a θ-parametrized model, conditioned on an observed image I, consists of two steps. First, the algorithm samples the hidden variable z from a prior distribution $p_z(z)$. Then, given a value of z, the algorithm samples the intensity difference image v from the conditional distribution $p_\theta(v|I, z)$. This process is also described in the graphical model in Fig. 2d.

Given a set of training pairs $\{I^{(i)}, v^{(i)}\}$, the algorithm maximizes the log-likelihood of the conditional marginal distribution during training

$$\sum_i \log p(v^{(i)}|I^{(i)}). \tag{7}$$

Recall that I and z are independent as shown in the graphical model in Fig. 2. Therefore, based on the Bayes' theorem, we have

$$p(v^{(i)}|I^{(i)}) = \frac{p_z(z)p_\theta(v^{(i)}|I^{(i)}, z)}{p(z|v^{(i)}, I^{(i)})}. \tag{8}$$

It is hard to directly maximizing the marginal distribution in Eq. (7). We therefore maximize its variational upper-bound instead, as proposed by Kingma and Welling [7].

Let $q_\phi(z|v^{(i)}, I^{(i)})$ be the variational distribution that approximates the posterior $p(z|v^{(i)}, I^{(i)})$. Then each term in the marginal distribution is upper bounded as

$$\log p(v^{(i)}|I^{(i)})$$
$$= \mathbb{E}_{q_\phi}\left[\log p(v^{(i)}|I^{(i)})\right]$$
$$= \mathbb{E}_{q_\phi}\left[\log \frac{p_z(z)p_\theta(v^{(i)}|I^{(i)}, z)}{p(z|v^{(i)}, I^{(i)})}\right]$$
$$= \mathbb{E}_{q_\phi}\left[\log \frac{p_z(z)}{q_\phi(z|v^{(i)}, I^{(i)})}\right] + \mathbb{E}_{q_\phi}\left[\log \frac{q_\phi(z|v^{(i)}, I^{(i)})}{p(z|v^{(i)}, I^{(i)})}\right]$$
$$\quad + \mathbb{E}_{q_\phi}[\log p_\theta(v^{(i)}|I^{(i)}, z)]$$
$$= -D_{\mathrm{KL}}(q_\phi(z|v^{(i)}, I^{(i)})||p_z(z))$$
$$\quad + D_{\mathrm{KL}}(q_\phi(z|v^{(i)}, I^{(i)})||p(z|v^{(i)}, I^{(i)}))$$
$$\quad + \mathbb{E}_{q_\phi}[\log p_\theta(v^{(i)}|I^{(i)}, z)]$$
$$\geq -D_{\mathrm{KL}}(q_\phi(z|v^{(i)}, I^{(i)})||p_z(z)) + \mathbb{E}_{q_\phi}[\log p_\theta(v^{(i)}|I^{(i)}, z)]$$
$$\triangleq \mathcal{L}(\theta, \phi, v^{(i)}|I^{(i)}) \tag{9}$$

The first KL-divergence term in Eq. (9) has an analytic form [7]. To make the second term tractable, we approximate the variational distribution, $q_\phi(z|x^{(i)}, I^{(i)})$, by its empirical distribution. We have

$$\mathcal{L}(\theta, \phi, v^{(i)}|I^{(i)})$$
$$\approx -D_{\mathrm{KL}}(q_\phi(z|v^{(i)}, I^{(i)})||p_z(z))$$
$$\quad + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(v^{(i)}|z^{(i,l)}, I^{(i)})$$
$$= -D_{\mathrm{KL}}(q_\phi(z|v^{(i)}, I^{(i)})||p_z(z))$$
$$\quad - \frac{1}{2\sigma^2 L}\sum_{l=1}^{L} \|v^{(i)} - f_{\mathrm{mean}}(z^{(i)}, I^{(i)})\| + C, \tag{10}$$

where $z^{(i,l)}$ are samples from the variational distribution $q_\phi(z|v^{(i)}, I^{(i)})$ and C is a constant. Eq. (10) is the variation lower bound that our network minimizes during training.

In practice, we simply generate one sample of $z^{(i,l)}$ at each iteration (thus $L = 1$) of stochastic gradient descent, and different samples are used for different iterations. By defining $\lambda = 1/(2\sigma^2)$ and taking the negative of the right-hand side of Eq. (10) , we get the objective function to minimize in

training (Eq. (5)):

$$D_{KL}(q_\phi(z|v^{(i)}, I^{(i)})||p_z(z)) + \lambda||v^{(i)} - f_{mean}(z^{(i)}, I^{(i)})||. \tag{11}$$

ACKNOWLEDGMENTS

REFERENCES

[1] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *International Conference on Learning Representations*, 2016. 1, 3, 12

[2] J. Walker, A. Gupta, and M. Hebert, "Patch to the future: unsupervised visual prediction," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 1, 2

[3] D. J. Fleet, M. J. Black, Y. Yacoob, and A. D. Jepson, "Design and use of linear models for image motion analysis," *International Journal of Computer Vision*, vol. 36, no. 3, pp. 171–193, 2000. 1, 2

[4] Y. Weiss and E. H. Adelson, "Slow and Smooth: a Bayesian theory for the combination of local motion signals in human vision," *Center for Biological and Computational Learning Paper*, vol. 158, no. 1624, pp. 1–42, 1998. 1, 2

[5] S. Roth and M. J. Black, "On the spatial statistics of optical flow," in *IEEE International Conference on Computer Vision*, 2005. 1, 2

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations*, 2016. 1, 2

[7] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014. 1, 2, 4, 13

[8] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*, 2016. 1, 2, 4, 13

[9] H. Lu and A. L. Yuille, "Ideal Observers for Detecting Motion: Correspondence Noise," in *Advances in Neural Information Processing Systems*, 2006. 2

[10] C. Liu, J. Yuen, and A. Torralba, "SIFT flow: Dense correspondence across scenes and its applications," *IEEE Transactions on Pattern Analysis and Machine intelligence*, vol. 33, no. 5, pp. 978–994, 2011. 2

[11] S. L. Pintea, J. C. van Gemert, and A. W. Smeulders, "Dejavu: Motion prediction in static images," in *European Conference on Computer Vision*, 2014. 2

[12] T. Xue, M. Rubinstein, N. Wadhwa, A. Levin, F. Durand, and W. T. Freeman, "Refraction wiggles for measuring fluid depth and velocity from video," in *European Conference on Computer Vision*, 2014. 2

[13] J. Walker, A. Gupta, and M. Hebert, "Dense optical flow prediction from a static image," in *IEEE International Conference on Computer Vision*, 2015. 2

[14] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in *European Conference on Computer Vision*, 2016. 2

[15] J. Wu, I. Yildirim, J. J. Lim, W. T. Freeman, and J. B. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," in *Advances in Neural Information Processing Systems*, 2015. 2

[16] J. Wu, E. Lu, P. Kohli, W. T. Freeman, and J. B. Tenenbaum, "Learning to see physics via visual de-animation," in *Advances in Neural Information Processing Systems*, 2017. 2

[17] D. Zheng, V. Luo, J. Wu, and J. B. Tenenbaum, "Unsupervised learning of latent physical properties using perception-prediction networks," in *Conference on Uncertainty in Artificial Intelligence*, 2018. 2

[18] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman, "Physics 101: Learning physical object properties from unlabeled videos," in *British Machine Vision Conference*, 2016. 2

[19] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating visual representations from unlabeled video," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[20] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 49–70, 2000. 2

[21] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," in *International Conference on Machine Learning*, 2015. 2

[22] J. Xie, S.-C. Zhu, and Y. N. Wu, "Synthesizing dynamic textures and sounds by spatial-temporal generative convnet," *arXiv preprint arXiv:1606.00972*, 2016. 2

[23] J. Xie, R. Girshick, and A. Farhadi, "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks," in *European Conference on Computer Vision*, 2016. 2

[24] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," *European Conference on Computer Vision*, 2016. 2

[25] 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*, 2014. 2

[26] E. L. Denton, S. Chintala, and R. Fergus, "Deep generative image models using an laplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems*, 2015. 2

[27] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa, "Video textures," *ACM Transactions on Graphics*, vol. 7, no. 5, pp. 489–498, 2000. 2

[28] Y. Wexler, E. Shechtman, and M. Irani, "Space-time video completion," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2004. 3

[29] N. Joshi, S. Mehta, S. Drucker, E. Stollnitz, H. Hoppe, M. Uyttendaele, and M. Cohen, "Cliplets: juxtaposing still and dynamic imagery," in *ACM Symposium on User Interface Software and Technology*, 2012. 3

[30] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski, "Panoramic video textures," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 821–827, 2005. 3

[31] Z. Liao, N. Joshi, and H. Hoppe, "Automated video looping with progressive dynamism," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 77, 2013. 3

[32] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," in *International Conference on Machine Learning*, 2015. 3, 12

[33] 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*, 2015. 3

[34] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in Neural Information Processing Systems*, 2016. 3, 6

[35] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," *Advances in Neural Information Processing Systems*, 2016. 3

[36] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, 2014. 4, 13

[37] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, 2015, pp. 3483–3491. 4

[38] J. Y. Wang and E. H. Adelson, "Layered representation for motion analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1993. 5

[39] B. D. Brabandere, X. Jia, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," *Advances in Neural Information Processing Systems*, 2016. 6

[40] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016. 6

[41] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Advances in Neural Information Processing Systems*, 2015. 6, 7, 12

[42] C. Liu, "Beyond pixels: exploring new representations and applications for motion analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 2009. 7

[43] W. Zhang, M. Zhu, and K. G. Derpanis, "From actemes to action: A strongly-supervised representation for detailed action understanding," in *IEEE International Conference on Computer Vision*, 2013. 8, 9

[44] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2387–2395. 8

[45] S. Huang and D. Ramanan, "Expecting the unexpected: Training detectors for unusual pedestrians with adversarial imposters," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2017. 8

[46] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Conference On Learning Theory*, 1993. 10

[47] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early visual concept learning with unsupervised deep learning," *arXiv preprint arXiv:1606.05579*, 2016. 10

[48] Y.-W. Chao, J. Yang, B. Price, S. Cohen, and J. Deng, "Forecasting human dynamics from static images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 12

[49] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, "Real-time user-guided image colorization with learned deep priors," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 119, 2017. 12

William T. Freeman is the Thomas and Gerd Perkins professor of electrical engineering and computer science with MIT, and a member of the Computer Science and Artificial Intelligence Laboratory (CSAIL) there. He was an associate department head from 2011-2014. His current research interests include machine learning applied to computer vision, Bayesian models of visual perception, and computational photography. He received outstanding paper awards at computer vision or machine learning conferences in 1997, 2006, 2009 and 2012, and test-of-time awards for papers from 1990 and 1995. Previous research topics include steerable filters and pyramids, orientation histograms, the generic viewpoint assumption, color constancy, computer vision for computer games, and belief propagation in networks with loops. He is active in the program or organizing committees of computer vision, graphics, and machine learning conferences. He was the program co-chair for ICCV 2005, and for CVPR 2013.

Tianfan Xue received his Ph.D. in EECS from MIT, working with William T. Freeman. Before that, he received his B.E. degree from Tsinghua Universtiy, and M.Phil. degree from The Chinese University of Hong Kong. His research interests include computer vision, image processing, and machine learning. Specifically, he is interested in motion estimation and image and video processing based on the motion information.
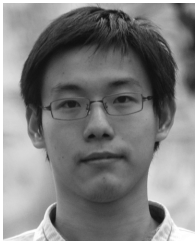
Jiajun Wu received the BEng degree in computer science from Tsinghua University, China, in 2014, and the SM degree in electrical engineering and computer science from MIT, in 2016. He is working toward the PhD degree in MIT, under the supervision of William T. Freeman and Joshua B. Tenenbaum. His research interests lie on the intersection of computer vision, machine learning, and computational cognitive science. He received the Facebook Fellowship, the Nvidia Fellowship, and the Adobe Fellowship.

Katherine L. Bouman received the BSE degree in electrical engineering from the University of Michigan, Ann Arbor, Michigan, in 2011, and the SM and PhD degrees in electrical engineering and computer science from Massachusetts Institute of Technology (MIT), Cambridge, Massachusetts, in 2013 and 2017, under the supervision of William T. Freeman. Her masters thesis on estimating material properties of fabric from video won the Ernst A. Guillemin Thesis Prize for outstanding S.M. thesis in electrical engineering with MIT. She received the US National Science Foundation Graduate Fellowship, the Irwin Mark Jacobs and Joan Klein Jacobs Presidential Fellowship, and is a Goldwater Scholar. Her research interests include computer vision, computational photography, and computational imaging.