# Compositional Video Prediction

Yufei Ye[1] Maneesh Singh[2] Abhinav Gupta[13*] Shubham Tulsiani[3*]
[1]Carnegie Mellon University [2]Verisk Analytics [3]Facebook AI Research
{yufeiy2,abhinavg}@cs.cmu.edu maneesh.singh@verisk.com shubtuls@fb.com
https://judyye.github.io/CVP/

## Abstract

*We present an approach for pixel-level future prediction given an input image of a scene. We observe that a scene is comprised of distinct entities that undergo motion and present an approach that operationalizes this insight. We implicitly predict future states of independent entities while reasoning about their interactions, and compose future video frames using these predicted states. We overcome the inherent multi-modality of the task using a global trajectory-level latent random variable, and show that this allows us to sample diverse and plausible futures. We empirically validate our approach against alternate representations and ways of incorporating multi-modality. We examine two datasets, one comprising of stacked objects that may fall, and the other containing videos of humans performing activities in a gym, and show that our approach allows realistic stochastic video prediction across these diverse settings. See project website for video predictions.*

## 1. Introduction

A single image of a scene allows us humans to make a remarkable number of judgments about the underlying world. For example, consider the two images on the left in Fig 1. We can easily infer that the top image depicts some stacked blocks, and the bottom shows a human with his arms raised. While these inferences showcase our ability to understand what is, even more remarkably, we are capable of predicting what will happen next. For example, not only do we know that there are stacked blocks in the top image, we understand that the blue and yellow ones will topple and fall to the left. Similarly, we know that the person in the bottom image will lift his torso while keeping his hands in place. In this work, we aim to build a model that can do the same – from a *single* (annotated) image of a scene, predict at a pixel level, what the future will be.

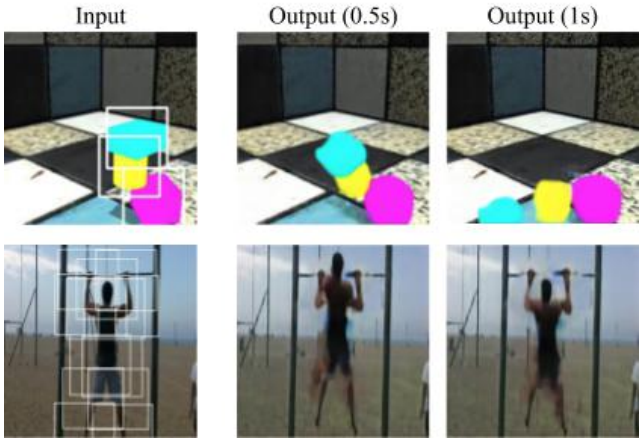


Figure 1. Given a still image with locations of entities (objects or joints), we predict a sequence of future frames. We visualize two frames from the predicted sequence for the given inputs.

A key factor in the ability to make these predictions is that we understand scenes in terms of 'entities', that can move and interact e.g. the blocks are separate objects that move; the human body's motion can similarly be understood in terms of the correlated motion of the limbs. We operationalize this ideology and present an approach that instead of directly predicting future frames, learns to predict the future locations and appearance of the entities in the scene, and via these composes a prediction of the future frame. The modeling of appearance and the learned composition allows our method to leverage the benefits of independent per-entity representations while allowing for reasoning in pose changes or overlap/occlusions in pixel space.

Although our proposed factorization allows learning models capable of predicting the future frames via entity-based reasoning, this task of inferring future frames from a single input image is fundamentally ill-posed. To allow for the inherent multi-modality of the prediction space, we propose to use a trajectory-level latent random variable that implicitly captures the ambiguities over the whole video and

* The last two authors were equally uninvolved.

train a future predictor conditioned of this latent variable. We demonstrate that modeling the ambiguities using this single latent variable instead of per-timestep random variables allows us to make more realistic predictions as well as sample diverse plausible futures.

We validate our approach using two datasets where the 'entities' either represent distinct objects, or human body joints, and demonstrate that the same method allows for predicting future frames across these diverse settings. We demonstrate: (a) the benefits of our proposed entity-level factorization; (b) ability of the corresponding learned decoder to generate future frames; (c) capability to sample different futures.

## 2. Related Work

**Modeling Physical Interaction.** Many recent works [38, 21, 2, 28, 4, 15] study modeling multiple objects in physical systems. Similar to us, they reason using the relationship between objects, and can predict the trajectories over a long time horizon. However, these approaches typically model deterministic processes under simple visual (or often only state based) input, while often relying on observed sequences instead of a single frame. Although some recent works take raw image as input [38, 10], they also only make prediction in state, and not pixel space. In contrast to these approaches, while we also use insights based on modeling physical interaction, we show results for video frame generation in a stochastic setup, and therefore also need to (implicitly) reason about other properties such as shape, lighting, color. Lastly, a related line of work is to predict stability of configurations [25, 13, 24, 19, 23]. Our video forecasting task also requires this understanding, but we do not pursue this as the end goal.

**Video Factorization.** It is challenging to directly predict pixels due to high dimensionality of the prediction space, and several methods have been used to factorize this output space [34, 32, 31, 7]. The main idea is to separate dynamic foreground from static background and generate pixels correspondingly. While these approaches show promising results to efficiently model one object motion, we show the benefits of modeling multiple entities and their interactions.

Another insight has been to instead model the output space differently, e.g. optical flow [35, 26], or motion transformation [40, 5, 18, 9]. This enables generating more photo-realistic images for shorter sequences, but may not be applicable for longer generation as new content becomes visible, and we therefore pursue direct pixel generation. Another line of work proposes to predict future in a predefined structured representation space, such as human pose [36, 33]. While our approach also benefits from predicting an intermediate structured representations, it is not our end goal as we aim to generate pixels from this representation.

**Object-centric video prediction.** A line of work explicitly enumerates the state of each object as location, velocity, mass, etc, then applies planning algorithm to unroll movement under reward [22, 16], or leverage Newtonian dynamics [42, 39]. However, these explicit representation based methods may not be applicable when the state space is hard to define, or pixel-wise predictions are not easily inferred given such a state e.g. human motions on complex background.

**Stochastic prediction.** Predicting the future is an inherently multi-modal task. Given a still image or a sequence of frames, there are multiple plausible futures that could happen. The uncertainty is usually encoded as a sequence of latent variables, which are then used in a generative model such as GAN [12] based [27, 34, 5, 31], or, similar to ours, VAE [20] based [35, 6]. These methods [11, 6, 41] often leverage an input sequence instead of a single frame, which helps reduce the ambiguities. Further, the latent variables are either per-timestep [6], or global [1, 41] whereas our model leverages a global latent variable, which in turn induces per-timestep variables.

## 3. Approach

Given an input image along with (known or detected) locations of the entities present, our goal is to predict a sequence of future frames. Formally, given a starting frame $f^0$ and the location of $N$ entities $\{b_n^0\}_{n=1}^N$, we aim to generate $T$ future frames $f^1, f^2, ..., f^T$. This task is challenging mainly for two reasons: a) the scene may comprise of multiple entities, making it necessary to account for their different dynamics and interactions, and b) the inherently multi-modal nature of the prediction task.

To overcome the first challenge, our insight is that instead of modeling how the scene changes as a whole, we should pursue prediction by modeling how the entities present change. We do so using an *entity predictor* that predicts per-entity representations: $\{x_n^t\}_{n=1}^N \equiv \{(b_n^t, a_n^t)\}_{n=1}^N$, where $b_n^t$ denotes the predicted location, and $a_n^t$ denotes predicted features that implicitly capture appearance for each entity. While this factorization allows us to efficiently predict the future in terms of these entities, an additional step is required to infer pixels. We do so using a *frame decoder* that is able to retain the properties of each entity, respect the predicted location, while also resolving the conflicts e.g. occlusions when composing the image.

To account for the fundamental multi-modality in the task, we incorporate a global random latent variable $u$ that implicitly captures the ambiguities across the whole video. This latent variable $u$, in turn deterministically (via a learned network) yields per-timestep latent variables $z_t$ which aid the per-timestep future predictions. Concretely, the predictor $\mathcal{P}$ takes as input the per-entity representation $\{x_n^t\}$ along with the latent variable $z_t$, and predicts

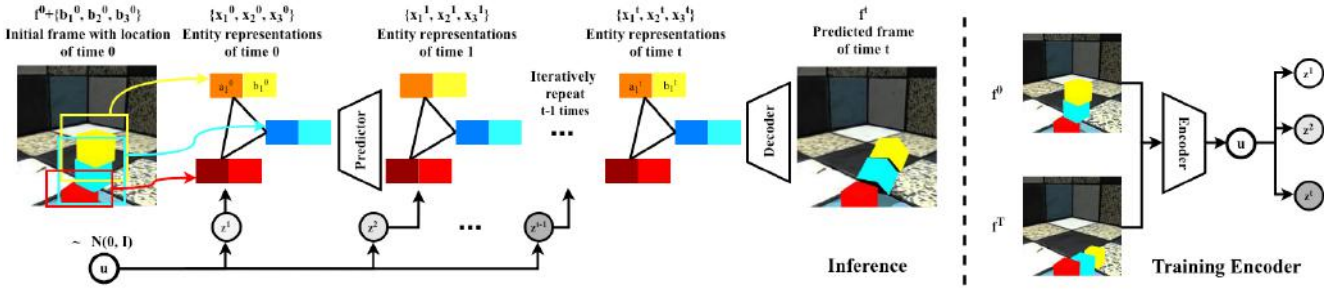


Figure 2. Our model takes as input an image with known/detected location of entities. Each entity is represented as its location and an implicit feature. Given the current entity representations and a sampled latent variable, our prediction module predicts the representations at the next time step. Our learned decoder composes the predicted representations to an image representing the predicted future. During training, a latent encoder module is used to infer the distribution over the latent variables using the initial and final frames.

the entity representations at the next timestep $\{x_n^{t+1}\} \equiv \mathcal{P}(\{x_n^t\}, z_t)$. The decoder $\mathcal{D}$, using these predictions (and the initial frame $f^0$ to allow modeling background), composes the predicted frame $f^t \equiv \mathcal{D}(\{x_n^t\}, f^0)$.

We train our model to maximize the likelihood of the training sequences, comprising of terms for both the frames and the entity locations. As is often the case with optimizing likelihood in models with unobserved latent variable models e.g. VAEs [20], directly maximizing likelihood is intractable, and we therefore maximize a variational lower bound. Towards this, we train another module, a *latent encoder*, which predicts a distribution over the latent variable $u$ using the target video. Note that the annotation of future frames/locations, as well as the latent encoder, are all only used during training. During inference, however, as illustrated in Fig 2, we take in input only a single frame along with (predicted/known) locations of the entities present, and can generate multiple plausible future frames. We first describe the predictor, decoder, and encoder modules in more detail, and the present the overall training objective.

### 3.1. Entity Predictor

Given per-entity locations and implicit appearance features $\{x_n^t\}_{n=1}^N \equiv \{(b_n^t, a_n^t)\}_{n=1}^N$, the predictor outputs the predictions for the next time step using the latent variable $z_t$. An iterative application of this predictor therefore allows us to predict the future frames for the entire sequence using the encodings from the initial frame. To obtain this initial input to the predictor i.e. the entity encodings at the first time step $\{x_n^0\}_{n=1}^N$, we use the known/detected entity locations $\{b_n^0\}$, and extract the appearance features $\{a_n^0\}$ using a standard ResNet-18 CNN [14] on the cropped region from $f^0$.

While the predictor $\mathcal{P}$ infers per-entity features, the prediction mechanism should also allow for the interaction among these entities rather than predicting each of them independently e.g. a block may or may not fall depending on the other ones around it. To enable this, we leverage a model in the graph neural network family, in particular based on 'Interaction Networks' which take in a graph $G = (V, E)$ with associated features for each node, and update these via iterative message passing and message aggregation. See [3] for a more detailed review. Our predictor $\mathcal{P}$ that infers $\{x_n^{t+1}\}$ from $(\{x_n^t\}, z_t)$ comprises of 4 interaction blocks, where the first block takes as input the entity encodings concatenated with the latent feature: $\{x_n^t \oplus z_t\}_{n=1}^N$. Each of these blocks performs a message passing iteration using the underlying graph, and the final block outputs predictions for the entity features for the next timestep $\{x_n^t\}_{n=1}^N \equiv \{(b_n^t, a_n^t)\}_{n=1}^N$. This graph can either be fully connected as with our synthetic data experiments, or more structured e.g. skeleton in our human video prediction experiments. See appendix for more details on the message passing operations.

Although our prediction module falls under the same umbrella as Interaction Networks(IN) [2], which are in turn related to Graph Convolution Networks(GCN) [21], there are subtle differences, both in the architecture and application. While [2] use a single interaction block to update node features, we found that stacking multiple interaction blocks for each timestep is particularly helpful. In contrast to GCNs which use a predefined mechanism to compute edge weights and use linear operations for messages, we find that using non-linear functions as messages allows better performance. Finally, while existing approaches do apply variants of GNNs for future prediction, these are restricted to predefined state-spaces as opposed to pixels, and do not account for uncertainties using latent variables.

### 3.2. Frame Decoder

The decoder aims to generate pixels of the frame $f^t$ from a set of predicted entity representations. While the entity representations capture the moving aspects of the scene, we also need to incorporate the static background, and additionally use the initial frame $f^0$ to do so. Our decoder $\mathcal{D}$, as depicted in Fig 3, therefore predicts $f^t \equiv \mathcal{D}(\{x_n^t\}, f^0)$. To compose frames from this factored input representation, there are several aspects that our decoder must consider: a) the predicted location of the entities should be respected,

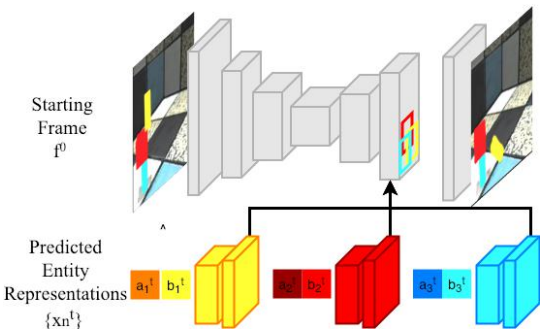


Figure 3. Our frame decoder takes in the initial frame $f^0$ and the predicted entity representations at time $t$, and outputs the frame corresponding to the predicted future $f^t$.

b) the per-entity representations may need to be fused e.g. when entities occlude each other, and c) different parts of background may become visible as objects move.

To account for the predicted location of the entities when generating images, we propose to decode a normalized spatial representation for each entity, and warp it to the image coordinates using the predicted 2D locations. To allow for the occlusions among entities, we predict an additional soft mask channel for each entity, where the value of masks are supposed to capture the visibility of the entities. Lastly, we overlay the (masked) spatial features predicted via the entities onto a canvas containing features from the initial frame $f^0$, and then predict the future frame pixels using this composed feature.

More formally, let us denote by $\phi_{bg}$ the spatial features predicted from the frame $f^0$ (using a CNN with architecture similar to UNet), and let $\{(\bar{\phi}_n, \bar{M}_n) = g(a_n)\}_{n=1}^N$ denote the features and spatial masks decoded per-entity using an up-convolutional decoder network $g$. We first warp, using the predicted locations $b_n$, these features and masks into image coordinates at same resolution as $\phi_{bg}$. Denoting by $\mathcal{W}$ a differentiable warping function e.g. in Spatial Transformer Networks [17], we can obtain the entity features and masks in the image space:

$$\phi_n = \mathcal{W}(\bar{\phi}_n, b_n); \;\; M_n = \mathcal{W}(\bar{M}_n, b_n) \tag{1}$$

Note that the warped mask and features $(\phi_n, M_n)$ for each entity are zero outside the predicted bounding box $b_n$, and the mask $M_n$ can further have variable values within this region. Using these independent background and entity features, we compose frame level spatial features $\phi$ by combining these via a weighted average. Denoting by $M_{bg}$ a constant spatial mask (with value 0.1), we obtain the composed features as:

$$\phi = \frac{\phi_{bg} \odot M_{bg} \;\; \oplus \;\; \sum_n \phi_n \odot M_n}{M_{bg} \;\; \oplus \;\; \sum_n M_n} \tag{2}$$

These composed features $\phi$ incorporate information from all entities at the appropriate spatial locations, allow for occlusions using the predicted masks, and incorporate the information from background. We then decode the pixels for the future frame from these composed features. Note that one has a choice over the spatial level where this feature composition happens e.g. it can happen in feature space at near the image resolution (late fusion), or even directly at pixel level (where the variables $\phi$ all represent pixels), or alternatively at a lower resolution (mid/early fusion). We find that late fusion in implicit (and not pixel) space yields most promising results, and also find that the inferred masks end up correspond to instance segmentations.

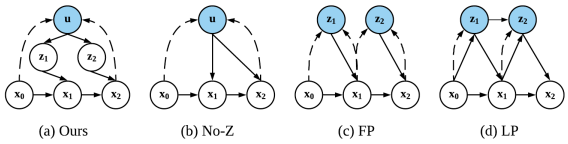


Figure 4. Our encoder (a) and baseline encoder (b-d). At test time, variables in blue are sampled randomly. At training, encoders model the posterior by all $x$s connected with dotted lines.

### 3.3. Latent Representation

We described in Sec 3.1 how our prediction module is conditioned on a latent variable $u$, which in turn generates per-timestep conditioning variables $z_t$ that are used in each prediction step – this is depicted in Fig 4(a). Intuitively, the global latent variable would capture video-level ambiguities e.g. where the blocks fall, the variables $z_t$ resolve the corresponding ambiguities in the per-timestep motions. While previous approaches for future prediction similarly use latent variables to resolve ambiguities (see Fig 4(c-d)), the typical idea is to use independent per-timestep random variables, whereas in our model the $z_t$'s are all correlated.

During training, instead of marginalizing the likelihood of the sequences over all possible values of the latent variable $u$, we instead minimize the variational lower bound of the log-likelihood objective. This is done via training another module, a *latent encoder*, which (only during training) predicts a distribution over $u$ conditioned on the ground-truth video. In practice, we find that simply conditioning on the first and last frame of the video (using a feed-forward neural network) is sufficient, and denote by $q(u|f^0, \hat{f}^T)$ the distribution predicted. Given a particular $u$ sampled from this distribution, we recover the $\{z_t\}$ via a one-layer LSTM which, using $u$ as the cell state, predicts the per-timestep variables for the sequence.

### 3.4. Training Objective

Overall, our training objective can be viewed as maximizing the log-likelihood of the ground-truth frame sequence $\{\hat{f}^t\}_{t=1}^T$. We additionally use training-time supervision for the locations of the entities $\{\{\hat{b}_n^t\}_{n=1}^N\}_{t=1}^T$. While this objective has an interpretation of log-likelihood maxi-

mization, for simplicity it is described as a loss $L$ with different terms, where the first $L_{pred}$ encourages the frame and location predictions to match the ground-truth:

$$L_{pred} = \sum_{t=1}^{T}(\|\mathcal{D}(\{x_n^t\}, f^0) - \hat{f}^t\|_1 + \lambda_1 \sum_{n=1}^{N} \|b_n^t - \hat{b}_n^t\|^2)$$

The second component corresponds to enforcing an information bottleneck on the latent variable distribution:

$$L_{enc} = KL[q(u) \parallel \mathcal{N}(0, I)]$$

Lastly, to further ensure that the decoder generates realistic composite frames, we add an auto-encoding loss that enforces it generates the correct frame when given entities representations $\{\hat{x}_n^t\}$ extracted from $\hat{f}^t$ (and not the ones predicted) as input.

$$L_{dec} = \sum_{t=0}^{T} \|\mathcal{D}(\{\hat{x}_n^t\}, f^0) - \hat{f}^t\|_1$$

The total loss is therefore $L = L_{dec} + L_{pred} + \lambda_2 L_{enc}$ with hyper-parameter $\lambda_2$ determining the trade-offs among accurate predictions and information bottleneck in random variable. See appendix for additional details. We will release our code for reproducibility.

# 4. Experiments

We aim to show qualitative and quantitative results highlighting the benefits of various components (predictor, decoder, and latent representation) in our approach, and aim to highlight that our approach is general to accommodate various scenarios. See generated videos in supplementary.

## 4.1. Experiment Setup

**Dataset.** We demonstrate our results on both the synthetic (ShapeStacks [13]) and real (Penn Action [44]) dataset. Shapestacks is a synthetic dataset comprised of stacked objects that fall under gravity with diverse blocks and configurations. The blocks can be cubes, cylinders, or balls with different colors. In addition to evaluating generalization ability, we further test with similar setups with videos comprised of 4, 5 or 6 blocks.

Penn Action [44] is a real video dataset of people playing various indoor and outdoor sports with annotations of human joint locations. The Penn Action dataset is challenging because of a) diverse backgrounds, view angles, human poses and scales b) noise in annotations, and c) multiple activity classes with different dynamics. We use a subset of the categories related to gym activities because most videos in these classes do not have camera motion and their backgrounds are similar within these categories. We adopt the recommended train/test split in [44]. Beyond that, we argue

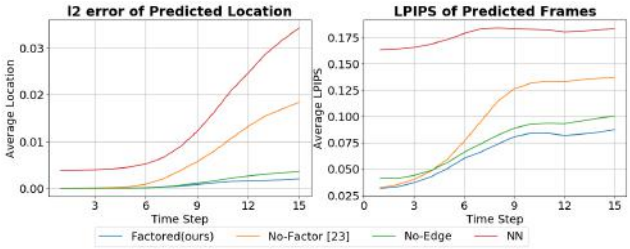


Figure 5. Error for location (Left) and frame (Right) prediction using our entity predictor and baselines. For each sequence, the best score of 100 random samples is drawn.

it is not impractical to assume known locations – we substitute ground truth annotation $\hat{b}_n^t$ with key-points location from off-the-shelf detector [8] in both training and testing.

In both scenarios, we train our model to generate video sequence of 1 second given an initial frame, using exactly the same architecture despite the two diverse scenarios – entities correspond to objects in Shapestacks and correspond to joints of human body in Penn Action.

**Evaluation Metrics.** In both of these settings, we evaluate the predicted entity locations using average mean square error and the quality of generated frames using the Learned Perceptual Image Patch Similarity (LPIPS) [43] metric. A subtle detail in the evaluation is that at inference, the prediction is dependent on a random variable $u$, and while only a single ground-truth is observed, multiple predictions are possible. To account for this, we draw 100 samples of latents and record the best scores as in [6]. When we ablate non-stochastic modules (e.g. decoders), we use the mean $u$ predicted by the latent encoder (after seeing the 'ground-truth' video). Without further specification, the curves are plotted in the 'best of 100' setup; the qualitative results visualize the best predictions in terms of LPIPS.

**Baselines.** There are three key components in our model, i.e. the entity predictor, frame decoder, and latent representation. Various baselines are provided to highlight our choices in each of the components. Among them, some variant specifically points to previous approaches as the following:

- No-Factor [23] only predicts on the level of frames. Here we provide supervision from entity locations and pixels instead of segmentation masks;
- LP [6] implements the stochastic encoder module in SVG-LP to compare different dependency of latent variables;
- Pose Knows [36] is most related to our Penn Action setting which also predicts poses as intermediate representation, but it predicts location jointly and generates videos in a different way.

Besides the above which are strongly connected to previous works, we also present other baselines whose details are discussed in Section 4.2.

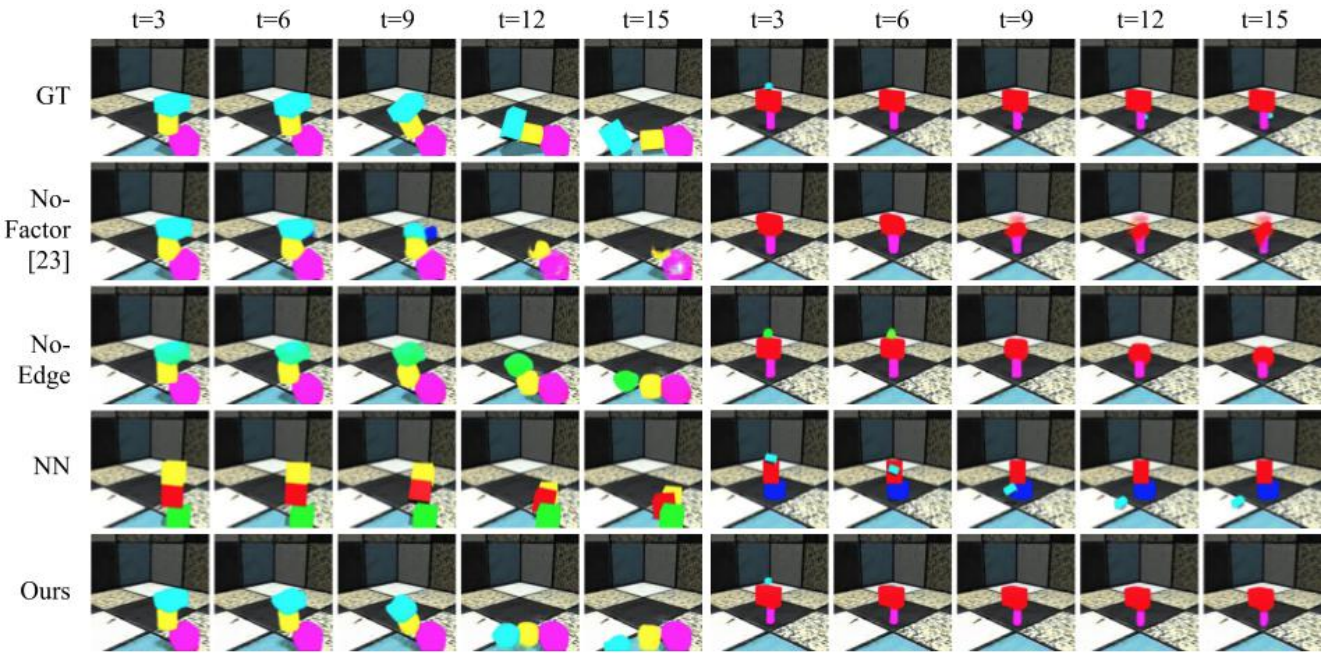


Figure 6. Video predictions using our predictor compared to baselines. We visualize the generated sequence after every 3 time steps.

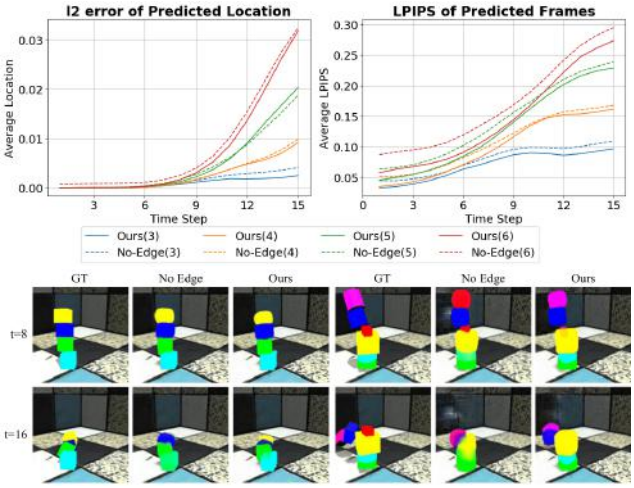


Figure 7. Above: Quantitative evaluation of the entity predictor when generalized to different number of blocks. The number in the bracket indicates the number of blocks in the subset. Below: Video predictions. The middle and last step are visualized.

### 4.2. Analysis using Shapestacks

We use Shapestacks to validate the different components of the proposed approach i.e. the entity predictor, frame decoder, and the modeling choices for the latent variables.

**Entity Predictor.** We aim to show that our proposed predictor, which is capable of factorizing prediction over per-entity locations and appearance, as well as allowing reasoning via GNNs, improves prediction. Towards this, we compare against three alternate models: a) No-Factor [23], b) No-Edge and c) Nearest Neighbor(NN). The No-Factor model does not predict a per-entity appearance but simply outputs a global feature that is decoded to foreground appearance and mask. To leverage the same supervision as box locations, it also takes as input (and outputs) the per-entity bounding boxes. The No-Edge does not allow for interactions among entities when predicting the future. The NN computes the features of the initial frame using a CNN. During inference, it retrieves the training video that is most similar in terms of those features. See appendix for details.

Figure 6 shows the prediction using our model and the baselines. The No-Factor generates plausible frames at the beginning and performs well for static entities. However, at later time steps, entities with large range of motion diffuse because of the uncertainty. In contrast, entities generated by ours have clearer boundary over time. The No-Edge does not accurately predict block orientations as it requires more information about relative configuration, and further changes the colors over time. In contrast, blocks generated by our approach gradually rotate and fall over and are learned to retain colors. The NN baseline shows our model does not simply memorize the training set. Figure 5 reports quantitative evaluations, and similarly observe the benefits of our approach.

Figure 7 shows the results when the model generalizes to different number of entities (4, 5, and 6) at test time. The No-Factor uses fully connected layers to predict which cannot be directly adapted to variable number of blocks. We show methods that are able to accommodate the number of entities changes, i.e. No-Edge and ours. Our method predicts locations closer to the truth with more realistic appearance, and is able to retain the blocks color across time. Note that we train all models with only three blocks.

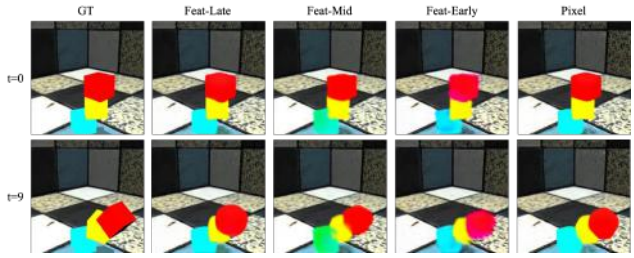


Figure 8. Qualitative results for composing entity representations into a frame. We visualize the outputs from variants of the decoder performing Late/Mid/Early fusion in feature space, or directly in pixel space. The first row depicts decoding of the initial representation; the second row depicts decoding of the predicted entities at a later time step.

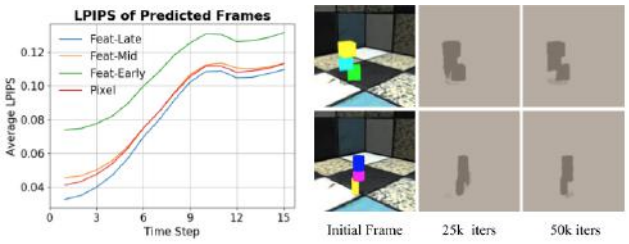


Figure 9. Left: Average Perceptual error for predicted frames via variants of the decoder. Right: Visualization of the composition of the foreground masks predicted for the entities.

**Primitive Decoder.** While the No-Factor baseline shows the benefits of composing features for each entity while accounting for their predicted spatial location, we ablate here whether this composition should directly be at a pixel-level or at some implicit feature level (early, mid, or late). Across all these ablations, the number of layers in the decoder remain the same; only the level at which features from entities are composed differs.

The qualitative result are shown in Fig 8 where the first row visualizes decodings from the initial frame, and the second row demonstrates decoding from predicted features for a later timestep. While both late/pixel-level fusion reconstructs the initial frame faithfully, the pixel-level fusion introduces artifacts for future frames. The mid/early fusion alternates do not capture details well. We also observe similar trends in the quantitative results visualized in Figure 9. Note the latent $u$ is encoded by the ground truth videos.

To further analyze the decoder, we visualize the generated soft masks in Figure 9. The values indicate the probability of the pixel belongs to a foreground of the entity. Note that this segmentation emerges despite of no direct supervision, but only location and frame-level pixels.

**Latent Representation.** Our choice of the latent variables in the prediction model differs from the common choice of using a per-timestep random variable $z_t$. We compare our approach (Figure 4a) with such other alternatives (Figure 4 b-e). The No-Z baseline (Figure 4b) directly

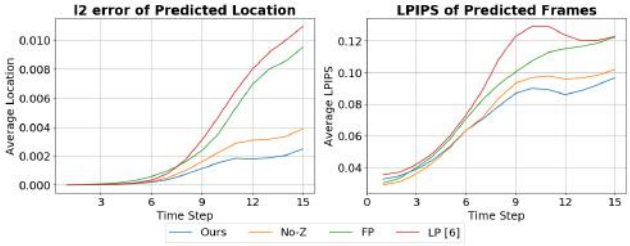


Figure 10. Error for location (Left) and frame (Right) prediction using our encoder and baselines. For each sequence, the best score of 100 random samples is drawn.

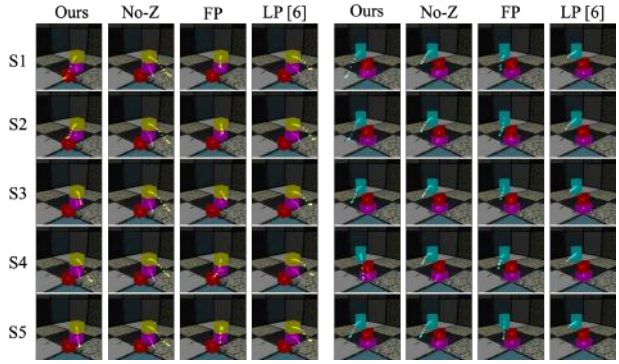


Figure 11. Visualizing five randomly sampled predictions by our method and other baselines. The predicted centers of entities over time overlay on top of the initial frame.

uses $u$ across every time step, instead of predicting a per-timestep $z^t$ from it. In both Fixed Prior (FP) and Learned Prior(LP) [6] baselines, the random variables are sampled per time step, either independently (FP), or depending on previous prediction (LP). During training, both FP and LP models are trained using an encoder similar to ours, but this encoder that predicts $z_t$ using the frames $f^t$ and $f^{t+1}$ (instead of our approach using $f^0$ and $f^T$ to predict $u$).

We visualize using five *random* samples in form of trajectories of entity locations in Figure 11. We notice that the direction of trajectories from No-Z model do not change across samples. The FP model has issues maintaining consistent motions across time-steps as during each timestep, an independent latent variable is sampled. The LP method performs well compared to FP, but still has similar issues. Compared to baselines, the use of a global latent variable allows us to sample and produce consistent motions across a video sequence, while also allowing diverse predictions across samples. The quantitative evaluations in Figure 10 show similar benefits where our method does well for both location error and frame perceptual distance over time.

### 4.3. Penn Action

Our model used in this dataset is exactly the same as that in the Shapestacks. Only the graph in the predictor is based on the human skeleton, instead of fully-connected. See project page for generated videos.

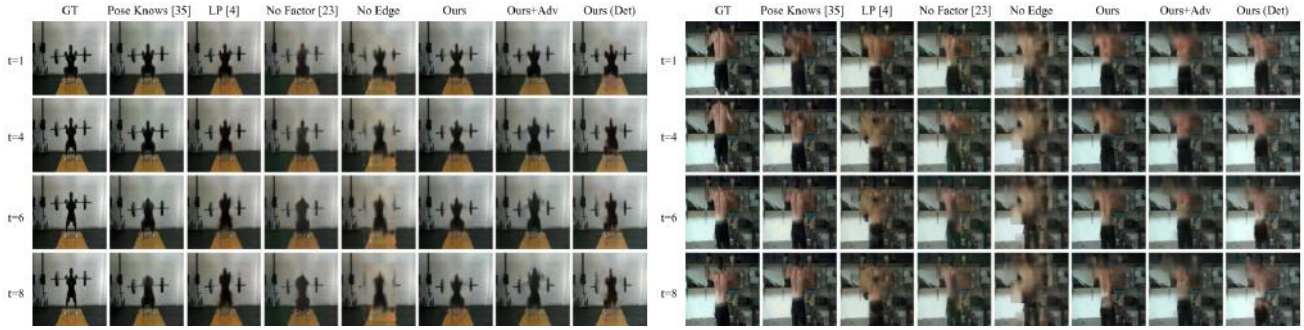


Figure 12. Video prediction results with best LPIPS latent using our approach compared to baselines. The last column visualizes results when the entity (joints) locations are replaced by the detection in both training and testing. Videos are in supplementary.

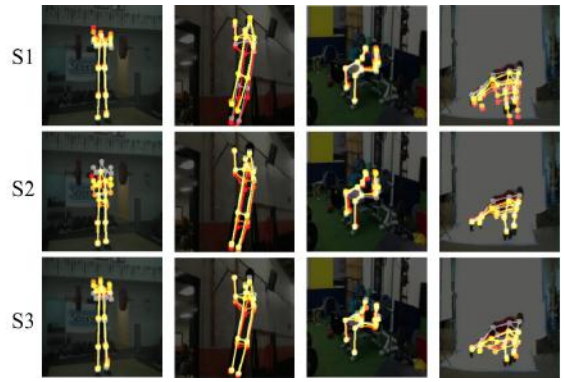


Figure 13. Visualizing joint positions in three randomly sampled predictions by our method. The initial skeletons are plotted as white. Skeletons at time 0.25s, 0.5s, and 1s are plotted as yellow, orange, and red, respectively.

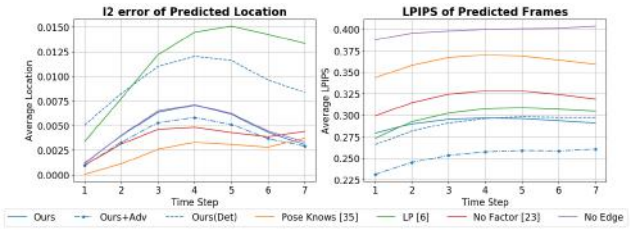


Figure 14. Error for location (Left) and frame (Right) prediction using our model and baseline methods. For each sequence, the best score of 100 samples is drawn.

We compare with Pose-Knows [36] which leverages entities as intermediate representation and generates pixel-level prediction. However, they a) do not predict feature for appearance but only location of each entity (joint); b) do not involve interaction mechanism; c) adopt a different generation method (GAN) where they stick sequence of rendered pose figures to the initial frame, and fuse them by a spatial-temporal 3D convolution network [30]. In their paper, the adversarial loss is posed to improve realism. We present that our method also benefits from the adversarial loss (Ours+Adv).

Figure 12 and Figure 14 show qualitative and quantitative results using the best latent variable among 100 samples. The No-Factor cannot generate plausible foreground while the No-Edge does not compose well. Our results improve to be sharper if adversarial loss is added (Ours+Adv).

We also visualize predictions when, during both *training* and inference, annotated key-points are replaced with detected key-points using [8]. We note that the performance is competitive to the setting using annotated key-point locations, indicating that our method is robust to annotation noise. It also indicates that the requirement of entity locations is not a bottle-neck, since automatically inferred location suffice in our experiment.

Figure 13 visualizes different sample futures using the predicted joint locations across time. Our model learns the boundary of the human body against the background as well as how the entities compose the human body even when they heavily overlap. More interestingly, the model learns different types of dynamics for different sports. For example, in pull ups, the legs move more while the hands are still; in clean and jerk, the legs almost remain at the same place.

## 5. Discussion

In this work we proposed a method that leverages compositionality across entities for video prediction. However, the task of video prediction in a general setting is far from being solved, and many challenges still remain. In particular, we rely on supervision of the entity locations, either from human or automatic annotations. It would be interesting to relax this requirement and allow the entities to emerge as pursued in some recent works [4, 15], although in simpler settings. Additionally, GAN-based auxiliary losses have been shown to improve image synthesis quality, and these could be explored in conjunction with our model. Lastly, developing metrics to evaluate the diversity and accuracy of predictions is also challenging due to the multi-modal nature of the task, and we hope some future efforts will also focus on this aspect.

**Acknowledgements.** This research is partly sponsored by ONR MURI N000141612007 and the ARO W911NF-18-1-0019.

## References


[1] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *ICLR*, 2017. 2

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, 2016. 2, 3

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 3

[4] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *ICLR*, 2016. 2, 8

[5] Baoyang Chen, Wenmin Wang, and Jinzhuo Wang. Video imagination from a single image with transformation generation. In *ACMMM Workshop*, 2017. 2

[6] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *ICML*, 2018. 2, 5, 7

[7] Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *NeurIPS*, 2017. 2

[8] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *ICCV*, 2017. 5, 8

[9] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, 2016. 2

[10] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *ICLR*, 2015. 2

[11] Katerina Fragkiadaki, Jonathan Huang, Alex Alemi, Sudheendra Vijayanarasimhan, Susanna Ricco, and Rahul Sukthankar. Motion prediction under multimodality with conditional stochastic networks. *arXiv preprint arXiv:1705.02082*, 2017. 2

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2

[13] Oliver Groth, Fabian Fuchs, Ingmar Posner, and Andrea Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. *ECCV*, 2018. 2, 5

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[15] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. *NeurIPS*, 2018. 2, 8

[16] De-An Huang, Amir-massoud Farahmand, Kris M Kitani, and James Andrew Bagnell. Approximate maxent inverse optimal control and its application for mental simulation of human interactions. In *AAAI*, 2015. 2

[17] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NeurIPS*, 2015. 4

[18] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NeurIPS*, 2016. 2

[19] Zhaoyin Jia, Andrew C Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3d reasoning from blocks to stability. *IEEE transactions on pattern analysis and machine intelligence*, 2015. 2

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014. 2, 3

[21] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *ICML*, 2019. 2, 3

[22] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *ECCV*, 2012. 2

[23] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. *ICML*, 2016. 2, 5, 6

[24] Wenbin Li, Seyedmajid Azimi, Aleš Leonardis, and Mario Fritz. To fall or not to fall: A visual approach to physical stability prediction. *arXiv preprint arXiv:1604.00066*, 2016. 2

[25] Wenbin Li, Aleš Leonardis, and Mario Fritz. Visual stability prediction and its application to manipulation. *AAAI*, 2016. 2

[26] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017. 2

[27] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *ICLR*, 2016. 2

[28] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. A simple neural network module for relational reasoning. In *NeurIPS*, 2017. 2

[29] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012. 11

[30] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 8, 11

[31] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. *CVPR*, 2017. 2

[32] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *ICLR*, 2017. 2

[33] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. *ICML*, 2017. 2

[34] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NeurIPS*, 2016. 2

[35] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, 2016. 2

[36] Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. The pose knows: Video forecasting by generating pose futures. In *ICCV*, 2017. 2, 5, 8, 11

[37] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 11

[38] Nicholas Watters, Andrea Tacchetti, Theophane Weber, Razvan Pascanu, Peter Battaglia, and Daniel Zoran. Visual interaction networks. *arXiv preprint arXiv:1706.01433*, 2017. 2

[39] Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, 2016. 2

[40] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NeurIPS*, 2016. 2

[41] Xinchen Yan, Akash Rastogi, Ruben Villegas, Kalyan Sunkavalli, Eli Shechtman, Sunil Hadap, Ersin Yumer, and Honglak Lee. MT-VAE: learning motion transformations to generate multimodal human dynamics. In *ECCV*, 2018. 2

[42] Tian Ye, Xiaolong Wang, James Davidson, and Abhinav Gupta. Interpretable intuitive physics model. *ECCV*, 2018. 2

[43] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[44] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *ICCV*, 2013. 5

## A. Architecture Details

**Entity Predictor.** Our predictor leverages the graph neural network family, whose learning process can be abstracted to iterative message passing and message aggregation. In each round of message passing, each node (edge) is a parameterized function of their neighboring node and edges, which updates their parameters by back propagation. We introduce the predictor architecture by instantiating the message passing and aggregation operation as following:

For the $l$-th layer of message passing, it consists of two operations:

$$v \to e : e_{i,j}^{(l)} = f_{v\to e}^{(l)}[v_i^{(l)} \oplus v_j^{(l)}]$$
$$e \to v : v_i^{(l+1)} = f_{e\to v}^{(l)}[\text{POOL}[e_{i,j}^{(l)}|(i,j)$$

We first perform node-to-edge passing $f_{v\to e}^{(l)}$ where edge embeddings are implicitly learned. Then we perform edge-to-node $f_{v\to e}^{(l)}$ operation given the updated edge embeddings. The message passing block can be stacked to arbitrary layers to perform multiple rounds of message passing between edge and node. In our experiment, we stack four blocks of the above module. For each block, $f_{v\to e}^{(l)}, f_{e\to v}^{(l)}$ are both implemented as a single fully connected layer. The aggregation operator is implemented as a average pooling. Note that connection expressed in the edge set can be either from explicitly specified graph, or a fully connected graph when the relationship is not explicitly observed.

**Frame Decoder.** We use the backbone of Cascaded Refinement Networks. Given feature in shape of $(N, D, h_0, w_0)$ either from entity predictor or background feature, the frame decoder upsamples it at the end of every unit. Each unit comprises of $Conv \to Batch \to LeakyRelu$. When the entity features are warped to image coordinates, the spatial transformation is implemented as a forward transformation to sharpen entities.

**Latent Encoder.** At training, the encoder takes in the concatenated features of two frames and apply a one layer neural network to obtain mean and variance of $u$, where we resample with reparameterization trick at training time. The resampled $u'$ is fed into a one-layer LTSM as cell unit to generates a sequence of $z^t$.

**Training Details.** We optimize the total loss with Adam optimizer in learning rate $1e-4$. $\lambda_1 = 100, \lambda_2 = 1e-3$. The dimensionality of latent is 8, i.e. $|u| = |z^t| = 8$. Location feature is represented as the center of entities $|b| = 2$, appearance feature $|a| = 32$. The region of each entity is set to a large enough fixed width and height to cover the entity, $d = 70$ in all of our experiment. All generated frame are in resolution of $224 \times 224$.

## B. Dataset

In Shapestacks, the 'entities' correspond to distinct objects, among which the graph used for interaction is fully connected since no explicit relationships are observed. The videos are generated by simulating the given initial configurations in in mujoco [29] for 16 steps. While the setting is deterministic under perfect state information (precise 3D position and pose, mass, friction, etc), the prediction task is ambiguous given an image input. The subset is split to 1320 clips for training, 281 clips for validation, and 296 clips for testing. When we evaluate the generalization ability, the test set further includes 221 (136 / 93) clips of videos comprised of 4 (5 / 6) blocks.

In Penn Action, 'entities' correspond to joints of human body and the graph is built based on prior knowledge of skeletons. If some joint is missing in the video, we instead link the edge to its parent if possible. We train our model to generate video sequences of 1 second at 8 fps given an initial frame. The categories we used are bench press, clean and jerk, jumping jacks, pull up, push up, sit up, and squat. To reduce overfitting, we augment data on the fly, including randomly selecting the starting frame for each clip, random spatial cropping, etc.

## C. Baseline Model

The No-Factor model does not predict a per-entity appearance but simply outputs a global feature that is decoded to foreground appearance and mask. To ensure the use of the same supervision as box locations, the No-Factor model also takes as input (and outputs) the per-entity bounding boxes. Thus, the foreground is represented as the extracted feature of the entire frame concatenated by all locations and they are directly predicted together with fully connected layers. To decode them to pixels, an additional binary mask is applied. However, no mechanism in No-Factor baseline guarantees the generated pixels of entities respect the predicted locations.

In the No-Edge baseline, we remove all but self-link edges between nodes so that all the nodes are predicted independently.

Pose-Knows [36] consists of two models: a Pose-VAE that takes input as the initial frame together with joint location and outputs joint location in the future, a Pose-GAN with skip layers that takes input as the initial frame together with rendered predicted poses and generate frames. The original work uses 3D convolutional [30] network to generate low resolution videos ($80 \times 64$). However, with progress in GAN techniques in recent years [37], we find that 2D convolution with frame-wise adversarial loss improves performance when generating high resolution videos ($224 \times 224$) in terms of both qualitative and quantitative evaluation.

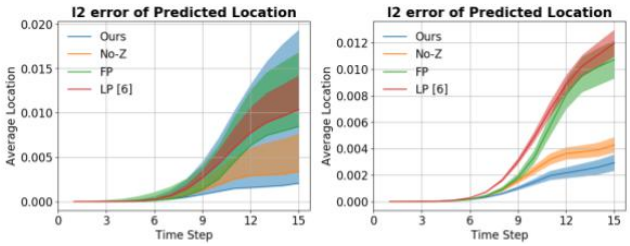


Figure 15. Left: For all 100 samples, $\sigma$ (shade) and best samples (line at lower boundary of shade) are plotted. Right: For best 5 samples of 100, mean and $\sigma$ are plotted.

## D. Standard Deviation

Prediction task is multi-modal, a model that correctly handles uncertainty will predict diverse future states (and therefore should) in the error across samples (as $\sigma = 0$ implies mean prediction). while generate enough good samples. We plot the $\sigma$ in location error (Shapestacks) over 100 samples in Figure 15 (Left). We also report the $\sigma$ across top 5 samples in Figure 15 (right)