UNSUPERVISED HIERARCHICAL VIDEO PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long term video prediction is a challenging machine learning problem, whose solution would enable intelligent agents to plan sophisticated interactions with their environment and assess the effect of their actions. Much recent research has been devoted to video prediction and generation, but mostly for short-scale time horizons. The hierarchical video prediction method by Villegas et al. (2017) is an example of a state of the art method for long term video predictions. However, their method has limited applicability in practical settings as it requires annotation of structures (e.g., pose) at every time step. This paper presents a long term hierarchical video prediction model that does not have such a restriction. We show that the network learns its own higher-level structure (e.g., pose equivalent hidden variables) that works better in cases where that higher-level structure does not capture all of the information needed to predict the next frames. This method gives sharper results than other video prediction methods which do not require a groundtruth pose, and its efficiency is shown on the Humans 3.6M and Robot Pushing datasets.

1 INTRODUCTION

It is hypothesized that learning to predict the future and the effect of their actions is an important quality for an intelligent agents that interact with their environment. This is a complicated task, as typical use-cases require predicting the outcome of interactions between the agent and objects, over multiple time-steps. Further issues arise due to the fact that interactions can be stochastic and could lead to potentially multimodal distributions of potential futures.

Concretely in this work we are looking at the task of predicting the pixels of future video frames given the first few observed frames. We also consider the action-conditional setting, in which we are given the action that the agent is taking and are tasked to predict the pixel-level outcome of that action in the future.

Modern techniques have been applied to this problem with success and they are generally good at producing relatively sharp video frames over a short number of time-steps into the future (Kalchbrenner et al., 2016; Finn et al., 2016). The method from Villegas et al. (2017) can generate long term video predictions, but requires ground truth pose annotations. In this work we explore ways to generate videos using a hierarchical model without requiring a ground truth pose or other high-level structure annotations for each frame. The method is hierarchical in the sense that it generates a high level structure, then makes next frame predictions based on that structure.

In Section 3 we discuss different ways to train our network without requiring the ground truth pose. In short, we show how the network can be trained end to end, as explained in Section 3.2.1 or how one can train different parts separately, as explained in Section 3.2.2. We show the advantages of not relying on a ground truth pose on the Robot Push Dataset (Finn et al., 2016) in Section 4.1. In Section 4.2 we demonstrate that our method produces sharper images than the non-hierarchical baseline of Finn et al. (2016) on the Humans 3.6M Dataset (Ionescu et al., 2014; Catalin Ionescu, 2011).

2 RELATED WORK

Patch-level prediction The video prediction problem has been initially studied at the patch level (Sutskever et al., 2009; Michalski et al., 2014; Mittelman et al., 2014; Srivastava et al., 2015).
This work showed promising results on synthetic data (e.g., bounding balls), but did not scale to predicting higher-resolution videos.

**Frame level prediction on realistic videos.** More recently, the video prediction problem have been formulated at the entire frame level. Most of the recent work is based on the convolutional encoder/decoder framework. [Finn et al. (2016)] proposed a network that can perform next-level video frame prediction by explicitly predicting movement. This work achieved reasonable short-term prediction, but the prediction typically blurs out very quickly on datasets with high stochasticity. [Mathieu et al. (2016)] proposed adversarial training with multiscale convolutional networks to generate sharper pixel-level predictions in comparison to conventional L2 loss. [Villegas et al. (2017)] proposed a network that decomposes motion and content in video prediction and showed improved performance over [Mathieu et al. (2016)]. [Lotter et al. (2017)] proposed a deep predictive coding network where each layer learns to predict the lower-level difference between the future frame and current frame. As an alternative approach to convolutional encoder/decoder networks, [Kalchbrenner et al. (2016)] proposed an auto-regressive generation scheme for improved prediction performance. Despite promise, all these work have not been demonstrated for long-term prediction on high-resolution natural videos beyond ≈20 frames.

**Long-term prediction.** [Oh et al. (2015)] proposed action-conditional convolutional encoder-decoder architecture that has demonstrated impressive long-term prediction performance on video games (e.g., Atari games), but it has not been applied for predicting challenging real videos. [Villegas et al. (2017)] demonstrated a long-term prediction method using hierarchical prediction. In this work, high-level structures are assumed to be given as supervision. The network learns to predict the high-level structures (e.g., human pose) from the input frames, which is used for predicting the high-level structure trajectories in the long term. Finally the predicted future structures are used for generating pixel-level predictions through a visual analogy network ([Reed et al., 2015]). The limitation of this work is that (1) the high-level structure needs to be defined a priori and (2) the network requires the full supervision of high-level structures. Our proposed work eliminates these constraints and show that it is possible to perform long-term prediction without high-level structure annotations.

3 **PROPOSED METHOD**

Our method uses a similar network architecture to [Villegas et al. (2017)] but we present ways of training the network that do not require a ground truth pose, with the network learning its own representation of the pose as it is trained.

3.1 **INFERENC**

At inference time the proposed method is the same as [Villegas et al. (2017)]. To generate the image at timestep $t$, the following procedure is used. First, a convolutional neural network encoder generates an embedding vector from the previous image: $e_{t-1} = \text{CNN}(img_{t-1})$. In [Villegas et al. (2017)], this encoding represents the pose of a person. In our work, it can represent any high level structure.

Next, a multilayer LSTM predictor network predicts what that encoding will be in a future time step. For some number of context frames, the predictor makes its prediction based off of the encoding from the ground truth image. After the predictor network has enough context, it makes its predictions based off of its previous predictions. For example if there are $C$ context frames, the following is used to generate the encoding at step $t$.

$$
\begin{align*}
[p_t, H_t] &= \text{LSTM}(e_{t-1}, H_{t-1}) & \text{if } t \leq C \\
[p_t, H_t] &= \text{LSTM}(p_{t-1}, H_{t-1}) & \text{if } t > C
\end{align*}
$$

where $H_t$ is the hidden state of the LSTM at timestep $t$. Similar to $e_t$ in the above, $p_t$ is the predicted encoding (pose, in [Villegas et al. (2017)]’s case) which can also represent any high level structure. $e_t$ and $p_t$ have the same dimensionality, which is much lower than the image, usually less than 256 dimensions.
Once \( p_t \) is obtained, a visual analogy network (VAN) \cite{reed2015} is used to generate the corresponding image at time \( t \). The VAN applies the transformation that occurred between two images to a given query image. In this case, we want the first frame of the video to be transformed in the same way as the encoding was transformed from the first to \( t \)-th time step. The VAN does this by mapping images to a space where analogies can be represented by additions and subtractions, and then mapping the result back to image space.

In our case, \( p_t \) is a one dimensional vector as opposed to a 2D feature map, so the part of the VAN that maps the encoding is a fully connected network instead of a convolutional neural network. As a result, the weights are not shared between the fully connected network which processes the encoding, and the conv-net which processes the image.

The equation used to obtain the predicted image at timestep \( t \) using the VAN is \( \hat{\text{img}}_t = \text{VAN}(e_1, p_t, \text{img}_1) \), where the VAN is defined as

\[
\text{VAN}(e_1, p_t, \text{img}_1) = f_{\text{dec}}(f_{\text{enc}}(p_t) - f_{\text{enc}}(e_1) + f_{\text{img}}(\text{img}_1))
\]

Note that \( f_{\text{enc}} \) is a fully connected network, and \( f_{\text{img}} \) is a conv net. \( f_{\text{dec}} \) is a deconv network.

### 3.2 Training

There are several ways these networks can be trained. In \cite{villegas2017}, they are each trained separately with the ground truth human pose. In this work, we explore alternative ways of training these networks in the absence of any ground truth pose or other high-level structure annotations; we use the procedure in Section 3.1 at inference time.

#### 3.2.1 End to End

One option is to connect the networks the same way as in inference time and train them end to end (E2E). In this method, the L2 loss of of the predicted image is optimized:

\[
\min(\sum_{t=1}^{T} L_2(\hat{\text{img}}_t, \text{img}_t))
\]

There are no constraints on what kind of encoding the encoder produces, or what kind of predictions the predictor makes. Because of how the networks are connected, the encoder will produce an encoding which the predictor can easily predict into the future. Likewise, the predictor will make predictions which the VAN can use to produce images which are similar to the ground truth. The encoder and predictor will not have to represent information that is present in the first ground truth frame, since the VAN will have access to the first frame. The size of \( e_t \) and \( p_t \) is a hyper parameter of this approach. Figure 1 represents a diagram of this method.

#### 3.2.2 Encoder Predictor with Encoder VAN

An alternative way to train the combined network is to explicitly train the encoder so that \( e_t \) is easy to predict into the future, and so that the VAN can use \( e_t \) to produce the next frame. We call this method Encoder Predictor with Encoder VAN, or EPEV. The encoder and predictor are trained together so the \( e_t \) is easy to predict and the predictor predicts that encoding into the future. To accomplish this, the difference between \( e_t \) and \( p_t \), \( L_2(e_t, p_t) \) is minimized. The encoder is also trained with the VAN so the VAN can use \( e_t \) to produce the image and so that the encoder generates an informative encoding. This is done by minimizing the loss of the VAN given the encoder output:

\[
L_2(\hat{\text{img}}_{e_t}, \text{img}_t) \text{ where } \hat{\text{img}}_{e_t} = \text{VAN}(e_1, e_t, \text{img}_1).
\]

The network is trained to minimize the sum of these two losses: \( \min(\sum_{t=1}^{T} L_2(\hat{\text{img}}_{e_t}, \text{img}_t) + \alpha L_2(e_t, p_t)) \), where \( \alpha \) is a hyper-parameter that controls the degree to which the \( e_t \) will be easy to predict vs. informative enough so the VAN can produce a good image.

See figure 2 for a diagram of the encoder and predictor trained together and figure 3 for the encoder and VAN trained together.

Separate gradient descent procedures (or optimizers, in TensorFlow parlance) could be used to minimize \( L_2(\hat{\text{img}}_{e_t}, \text{img}_t) \) and \( L_2(e_t, p_t) \), but we find that minimizing the sum works better experimentally.
3.2.3 E2E WITH POSE

The end to end approach can also be augmented if the dataset has information about the ground truth pose or any other high-level frame annotations. In this method, the first part of the $e_t$ and $p_t$ is optimized to represent the pose, and the rest of $e_t$ and $p_t$ is trained the same way as in the E2E approach. At each training step a separate optimizer minimizes each loss. In this method, we can think of $e_t$ and $p_t$ as the concatenation of two vectors, one representing the pose, and another
representing other information the network can represent. If \( e_t = [e_{\text{pose}_t}, e_{\text{remaining}_t}] \) and \( p_t = [p_{\text{pose}_t}, p_{\text{remaining}_t}] \), the following losses are minimized:

The loss representing how well the encoder infers the pose: \( \min(\sum_{t=1}^{T} L_2(e_{\text{pose}_t}, \text{pose}_t)) \). The loss representing how well the predictor predicts the pose: \( \min(\sum_{t=1}^{T} L_2(p_{\text{pose}_t}, \text{pose}_t)) \). The end to end loss: \( \min(\sum_{t=1}^{T} L_2(\hat{\text{img}}_t, \text{img}_t)) \).

These losses are minimized with separate optimizers in this method. Minimizing the end to end loss ensures that the V AN will learn to use the pose provided by the predictor network, and that the encoder and predictor will learn to produce additional information besides the pose that is useful to the V AN.

This method will likely work the best on datasets where the ground truth pose captures some, but not all of the information needed to predict the next video (for instance, where there is movement in the background or interactions with the objects whose pose is not measured). In this method, the V AN will be able to use both the predicted pose, and other information the network has encoded to generate the video.

3.2.4 Individual

In order to compare to baseline we also implemented the method where each of the networks are trained individually as in Villegas et al. (2017). The encoder is trained to produce the pose given the image. The predictor is trained to predict that pose into the future. The V AN is trained to generate the image given the pose. We call this method INDIVIDUAL. The main difference between this method and Villegas et al. (2017) is we do not use a adversarial loss (Goodfellow et al., 2014). See future work for a discussion of how an adversarial loss could be added to our method.

4 Experiments

These methods were tested on two different datasets, the Robot Push dataset (Finn et al., 2016) and the Humans 3.6M dataset (Ionescu et al., 2014; Catalin Ionescu, 2011). Videos of the results of our method are available by visiting the following URL: https://goo.gl/WA8uxc.

We trained all of the methods including Finn et al. (2016) for 3 million steps using async SGD, across 32 worker machines. We used a minibatch size of 8 sequences in each step. The minibatch size could be so small because there were multiple frames per sequence. In methods with multiple optimizers, a step is defined as running each optimizer once. The hyperparameters are optimized separately for both datasets on a validation set. We used the best learning rates for each method.

The EPEV method works best experimentally if \( \alpha \) starts small, around 1e-7, and is gradually increased to around .1 during training. Then the encoder will first be optimized to produce an informative encoding, then gradually optimized to also make that encoding easy to predict. If \( \alpha \) is too high initially, the encoder reduces to outputting the same encoding at every timestep so it is easy to predict. Our conjecture is that the encoder has not learned to generate encoding that are useful to the predictor yet, so outputting the same encoding every time is a simple way to minimize the loss.

4.1 Robot Push Dataset

This dataset contains videos of a robot arm pushing objects on a table. The current joint angles and the location of the end effector are given and we use these as the pose/high-level structures to be predicted, for the methods which require it. The action the robot arm is taking is fed into the predictor.

Each of the methods considered was given two frames of context, and then trained to predict 17 subsequent frames. An encoding size of 16 was used for the E2E method. The size of the pose is 12 so the encoding size of the INDIVIDUAL method is 12. The other methods used an encoding size of 32.

We split the dataset into training, validation and test randomly. We used 64x64 images, and the same frame rate as the original dataset. Results from our test set are shown here. We do not test on novel objects as in Finn et al. (2016). The robot push dataset with actions is relatively deterministic,
so PSNR is a reasonable metric to use to compare different methods. The PSNR numbers are from averaging across about 54k runs.

As shown in Table 1, the E2E WITH POSE method is the best on average and the INDIVIDUAL method is the worst. This shows that there is some benefit of the network creating its own equivalent of the pose instead of using the predefined pose.

Figure 4: PSNR compared with ground truth for different methods on the robot push dataset.

We can gain more insight into the performance by analyzing the PSNR at different frame numbers as shown in Figure 4. In the first few frames, the other methods are much better than the INDIVIDUAL method. This may be because the high level structure the other methods create is fine grained enough to produce an image very close to frame 0 with slight modifications. In the later frames, the INDIVIDUAL method does better than the E2E and EPEV methods. This could be because the pose is a very good representation of the arm, and this dataset is dominated by arm movements. The E2E WITH POSE method obtained the best PSNR at nearly every frame.

We hypothesized that the methods where the network could learn its own pose equivalent would predict objects better than the INDIVIDUAL method. To test this we manually compared the E2E and EPEV methods to the INDIVIDUAL method and evaluated which object prediction most closely matched the ground truth. We evaluated 40 videos where objects move, the results are in Table 2.

In the INDIVIDUAL method, the predictor network can only produce the pose, so the VAN has to infer how the objects will move based on the start and end state of the arm. We were surprised by how well the VAN could infer this. However, from examining the videos, the EPEV method had better object predictions than the INDIVIDUAL. This supports our hypothesis that the network defining it own pose equivalent can better represent how objects will move. Figure 5 shows a few frames where the EPEV method predicted the object movement better than E2E. More such analysis is in the appendix.
Table 2: Results from manual comparison of object predictions in 40 videos. The methods perform similarly in the remaining videos.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Number of videos</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPEV better than INDIVIDUAL</td>
<td>6</td>
</tr>
<tr>
<td>INDIVIDUAL better than EPEV</td>
<td>3</td>
</tr>
<tr>
<td>E2E better than INDIVIDUAL</td>
<td>6</td>
</tr>
<tr>
<td>INDIVIDUAL better than E2E</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 5: A visual comparison of the different methods on the robot push dataset. In the E2E and EPEV methods, the yellow square objects moves, where it does not in the INDIVIDUAL method.

Another observation from figure 5 is that the INDIVIDUAL and EPEV methods give the sharper arm predictions than E2E. One reason for this is that the pose provides very good information for predicting the robot arm so the INDIVIDUAL method does well. Another reason is that in the E2E method, the predictor can give information about its confidence in the prediction to the VAN. The VAN can use this to make the image more blurry as the predictor gets less confident over time. In the INDIVIDUAL and EPEV methods the VAN and predictor are not trained together, so they cannot do this.

The model from [Finn et al. (2016)] gets a better PSNR than ours on this dataset. It gets about 26.7 PSNR on the 20th frame. The movement in this dataset can easily be represented by the movement of pixels, and it is relatively deterministic. So the model from [Finn et al. (2016)] which explicitly predicts the movement of objects and minimizes the L2 loss directly works well here.

4.2 Humans 3.6M

Our method was also tested on the Humans 3.6M Dataset. Only the E2E and EPEV methods were tested here, since [Villegas et al. (2017)] has already shown the results using the ground truth pose.
We used subjects 1, 5, 6, 7 and 8 for training, subject 9 for validation. Subject 11 results are reported in this paper for testing. We used 64 by 64 images. We subsampled the dataset to 6.25 frames per second. We trained the methods to predict 32 frames and the results in this paper show predicting 64 frames. Each method is given the first 5 frames as context frames. So the in these images, the model predicts about 10 seconds into the future from .8 seconds of context. This setup is comparable to Villegas et al. (2017) because they also show predicting about 10 seconds into the future given .8 seconds of context. The results of different methods on this dataset are shown in figure 6.

We used an encoding size of 32 for the E2E method and a encoding size of 64 for the EPEV method on this dataset.

![Figure 6: A visual comparison of the different methods on the Humans Dataset. The baseline method CDNA is from Finn et al. (2016). This example is cherry picked to highlight the problem of the figure disappearing in the E2E approach. See the appendix for non cherry picked results. The contrast of these images was increased to make the humans easier to see.](image)

In the E2E method, the person often disappears part way through the prediction. In the EPEV method this happens more rarely. We think this is caused the same reason the EPEV method produced sharper images on the robot push dataset than the E2E method. In the E2E method, the predictor can encode its confidence in the prediction, and the VAN will not generate the person if the predictor is not confident. This minimizes the L2 loss since there is a heavy penalty for generating the person in the wrong place in the image. In the EPEV method, the predictor and VAN are trained separately, so they do not learn to do this. When EPEV does generate a image where the human is blurry, it is probably because the predictor predicted an invalid encoding so the VAN could not generate a reasonable human image. One reason the predictor would predict an invalid encoding is because it is trained on L2 loss so if there are two possible encodings, it will predict the average.

We compare our method to the CDNA method in Finn et al. (2016) shown in figure 6. We trained this method on the same data as ours. We found the best learning rate for this method. This method has a similar problem to the E2E method where the figure usually disappears or blurs out part way through prediction. This is because the L2 loss is directly optimized in this method so it encodes uncertainty as blur (Finn et al., 2016).

We also compare our method to Villegas et al. (2017). This method gives sharper results than ours. We think there are two reasons for this. One is that Villegas et al. (2017) uses a adversarial loss (Goodfellow et al., 2014) which produces sharper images. Also, since nothing else is moving in this dataset besides the human, the pose works well as a high level structure.
5 CONCLUSION

On datasets where the pose does not capture all of the information needed to predict future frames, letting the network define its own high level structure in addition to the pose is an improvement upon Villegas et al. (2017). The EPEV method generates sharper images than Finn et al. (2016) on non-deterministic datasets.

On the robot push dataset the E2E model works better than EPEV, but on the Humans Dataset, the EPEV model is better. This is because the robot push dataset is relatively deterministic, so optimizing L2 loss on the image directly makes sense. However on the Humans Dataset, optimizing the L2 loss directly produces blurry images. In the EPEV method, the predictor and VAN are trained separately, so they cannot learn to make the images more blurry over time.

6 FUTURE WORK

To fix the problem of the EPEV method generating invalid encodings, an adversarial loss (Goodfellow et al., 2014) can be used where a network will discriminate between the results from the encoder vs the predictor. The predictor and encoder will be optimized to confuse the discriminator instead of minimizing L2 loss. This way, the predictor will always be motivated to produce valid encoding, even if there is more than one possible encoding.

Another solution do this problem is to use an approach similar to a variational autoencoder (Kingma & Welling, 2013). The EPEV approach will be modified so the output from the encoder will be reparameterized to means and standard deviations before it is fed into the VAN. The KL divergence will be minimized between the reparameterized encoder output and the normal distribution. As a result, the VAN will be able to generate an image from anything that has a small KL divergence from the normal distribution. The predictor network output will also be reparameterized to output the mean and standard deviation. The KL divergence between the predictor network output and the encoder output will be minimized so that the predictor outputs encodings that are similar to the outputs from the encoder, and are also similar to the normal distribution. This will ensure that the predictor network always produces valid encodings.

REFERENCES


Appendices

A IMPLEMENTATION DETAILS

The convolutional neural net encoder is a VGG-16 network [Simonyan & Zisserman, 2014]. In the EPEV method, the encoder needs to be pre-trained on Imagenet [Deng et al., 2009]. If that is not done, the VAN will ignore the output from the encoder. All other configurations are run without pretraining.

The predictor network is a 20 layer LSTM, while the analogy network uses the deep analogy transformation described in [Reed et al., 2015]. The VAN outputs a mask which controls whether to use its own output, or the pixels of the first frame. This allows the network to focus on learning the changing parts of the image instead of the background.

The end effector orientation is converted to quaternions [Bousmalis et al., 2016] in order to calculate the loss if the pose is used. Layer normalization [Lei Ba et al., 2016] is used between every other layer in the VAN and predictor networks. We used dropout [Srivastava et al., 2014] on the encoder and VAN to prevent overfitting. The network overfits less in the INDIVIDUAL method, so we used less dropout.
B  EPEV METHOD WITH ENCODER OUTPUT FED TO VAN

To see what the encoder has learned in the EPEV method, we can obtain results from the visual analogy network given the input from the encoder. The encoder is given the ground truth image. The results are shown in figure 7. The results show that the encoder encodes where the person is in the image, as well as the orientation of the arms, legs and head to some extent. The results are not as good as one would expect from an autoencoder, since the encoder has the constraint that the encoding also has to be easy to predict.

![Figure 7: Results from the EPEV approach when the VAN is given the output of the encoder on the ground truth frame.](image)
C MORE RESULTS ON ROBOT PUSH DATASET.

C.1 FRAMES WHERE EPEV PREDICTS OBJECTS BETTER THAN INDIVIDUAL.
C.2 Frames where individual predicts objects better than EPEV.
Under review as a conference paper at ICLR 2018
C.3 Frames where EPEV and individual predict objects with the same quality.
Under review as a conference paper at ICLR 2018

Frame 1
Frame 7
Frame 13
Frame 19

ground truth
E2E
individual
E2E with pose
BEV
D More results on humans dataset.