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 doesn't have such a restriction. We show that the network learns its own learned higher-level structure (e.g., pose equivalent hidden variables) that works better in cases where that higher-level structure doesn't capture all of the information needed to predict the next frames. This method gives sharper results than other video prediction methods which don't 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 the 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 relative 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). In this work we explore ways 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 the choices to be made for the network architecture and different ways to train it 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 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).

These work showed promising results on synthetic data (e.g., bounding balls), but didnt scale to predicting higher-resolution videos.

Frame level prediction on realistic videos. More recently, 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. 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 different ways of training the network that don't require a ground truth pose. Instead, the network learns its own equivalent of the pose as it is trained.

3.1 Inference

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.

$$e_{t-1} = CNN(img_{t-1})$$

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 N context frames, the following is used to generate the encoding at step t.

$$\begin{cases} [p_t, H_t] = LSTM(e_{t-1}, H_{t-1}) & if\ t <= N \\ [p_t, H_t] = LSTM(p_{t-1}, H_{t-1}) & if\ t > N \end{cases}$$

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 much lower dimensionality than the image, usually less than 256 dimensions.

Once p_t is obtained, a visual analogy network (VAN) (Reed et al., 2015) 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 n-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 aren't 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

$$\widehat{img}_t = VAN(e_1, p_t, img_1)$$

where the VAN is defined as

$$VAN(e_1, p_t, img_1) = f_{dec}(f_{enc}(p_t) - f_{enc}(e_1) + f_{img}(img_1))$$

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

3.2 Training

There are several ways these networks can be trained. In Villegas et al. (2017), 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 frame-level high-level structures that are annotation. Each of the methods described below uses the procedure described above during 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:

$$\min(\sum_{t=1}^{T} L_2(\widehat{img}_t, img_t))$$

There are no constraints on what kind of encoding the encoder produces, or what kind of predictions the predictor makes. In principle, the encoder is trained to produce an encoding which the predictor can predict into the future easily. Likewise, the predictor will be trained to make predictions which the VAN can use to produce images which are similar to the ground truth. The encoder and predictor won't 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 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: $L_2(\widehat{img}_{e_t}, img_t)$ where $\widehat{img}_{e_t} = VAN(e_1, e_t, img_1)$. The network is trained to minimize the sum of these two losses:

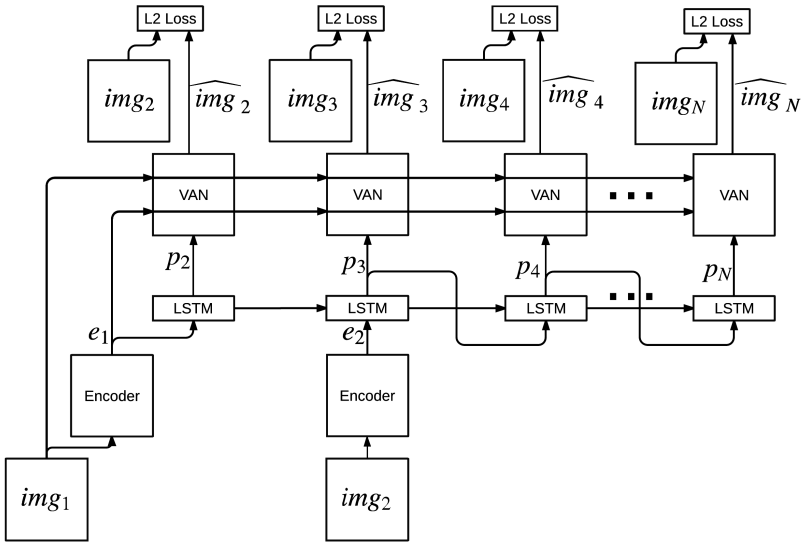


Figure 1: The E2E method. The first few frames are encoded and fed into the predictor as context. The predictor predicts the subsequent encodings, which the VAN uses to produce the pixel level predictions. The average of the losses is minimized. This is also the configuration of every method at test time, even if the predictor and VAN are trained separately.

$$\min(\sum_{t=1}^{T} L_2(\widehat{img}_{e_t}, img_t) + \alpha L_2(e_t, p_t))$$

where α 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 $L_2(\widehat{img}_{e_t}, img_t)$ and $L_2(e_t, p_t)$, but we find that minimizing the sum works better experimentally.

With this method, the predictor will predict the encoder outputs in future time steps, and the VAN will use the encoder output to produce the frame.

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 theE2Eapproach. 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 $e_t = [e_{pose_t}, e_{remaining_t}]$ and $p_t = [p_{pose_t}, p_{remaining_t}]$, the following losses are minimized:

The loss representing how well the encoder infers the pose:

$$\min(\sum_{t=1}^{T} L_2(e_{pose_t}, pose_t))$$

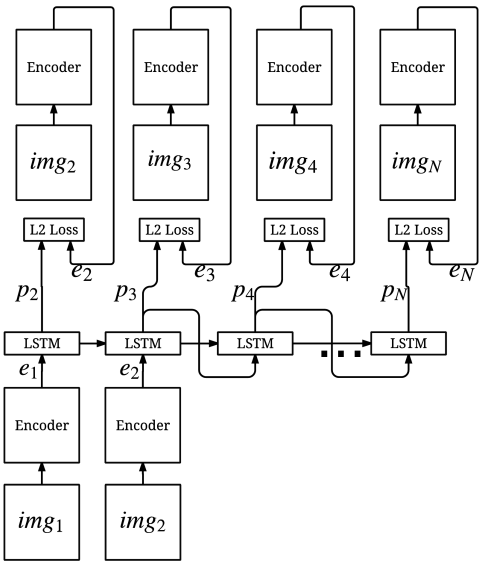


Figure 2: The part of the EPEV method where the encoder and predictor are trained together. The encoder is trained to produce an encoding that's easy to predict, and the predictor is trained to predict that encoding into the future. The average of the losses is minimized.

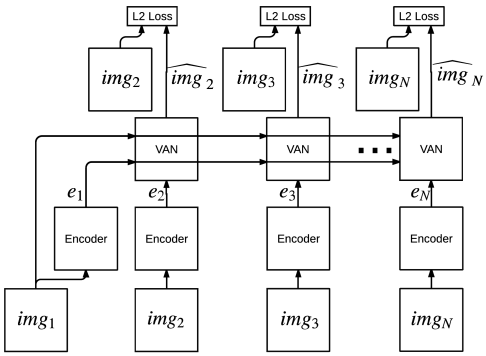


Figure 3: The part of the EPEV method where the encoder and VAN are trained together. The encoder is trained to produce an encoding that is informative to the VAN, while the VAN is trained to output the image given the encoding. The average of the losses is minimized. This is similar to an autoencoder.

The loss representing how well the predictor predicts the pose:

$$\min(\sum_{t=1}^{T} L_2(p_{pose_t}, pose_t))$$

The end to end loss

$$\min(\sum_{t=1}^{T} L_2(\widehat{img}_t, img_t))$$

These losses are minimized with separate optimizers in this method. Minimizing the end to end loss ensures that the VAN 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 VAN.

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 VAN will be able to use both the predicted pose, and other information the network has encoded to generate the video.

3.2.4 INDIVIDUAL

This method is very similar to the method 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 VAN is trained to generate the image given the pose. The main difference between this method and Villegas et al. (2017) is we don't 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.

Table 1: Average PSNR for different methods

Method	Average PSNR across all frames
E2E	27.018
Individual	26.56
E2E with pose	27.34
EPEV	26.64

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 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 α 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 make that encoding easy to predict. If α is too high initially, the encoder reduces to outputting the same encoding at every timestep so it's 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.

The robot push dataset with actions is relatively deterministic, so PSNR is a reasonable metric to use to compare different methods.

We split the dataset into training, validation and test randomly. Results from our test set are show here. We don't test on novel objects as in Finn et al. (2016).

We used 64x64 images, and the same frame rate as the original dataset. 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.

We can gain more insight into the performance by analyzing the PSNR at different frame numbers as show 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.

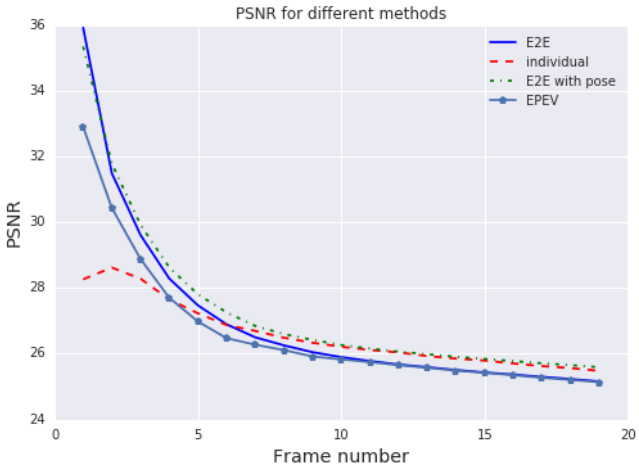


Figure 4: PSNR compared with ground truth for different methods on the robot push dataset.

Table 2: Results from manual comparison of object predictions in 40 videos. The methods perform similarly in the remaining videos.

Comparison	**Number of videos**
EPEV better than individual	6
Individual better thanEPEV	3
E2E better than individual	6
Individual better thanE2E	6

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, and show the results 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, theEPEV method had better object predictions than the individual. This supports our hypothesis that the network defining it's own pose equivalent can better represent how objects will move. Figure 5 shows one video where theEPEV method predicted the object movement better than E2E. Other videos from this analysis are in the appendix.

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 theE2Emethod, 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 aren't trained together, so they can't 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.

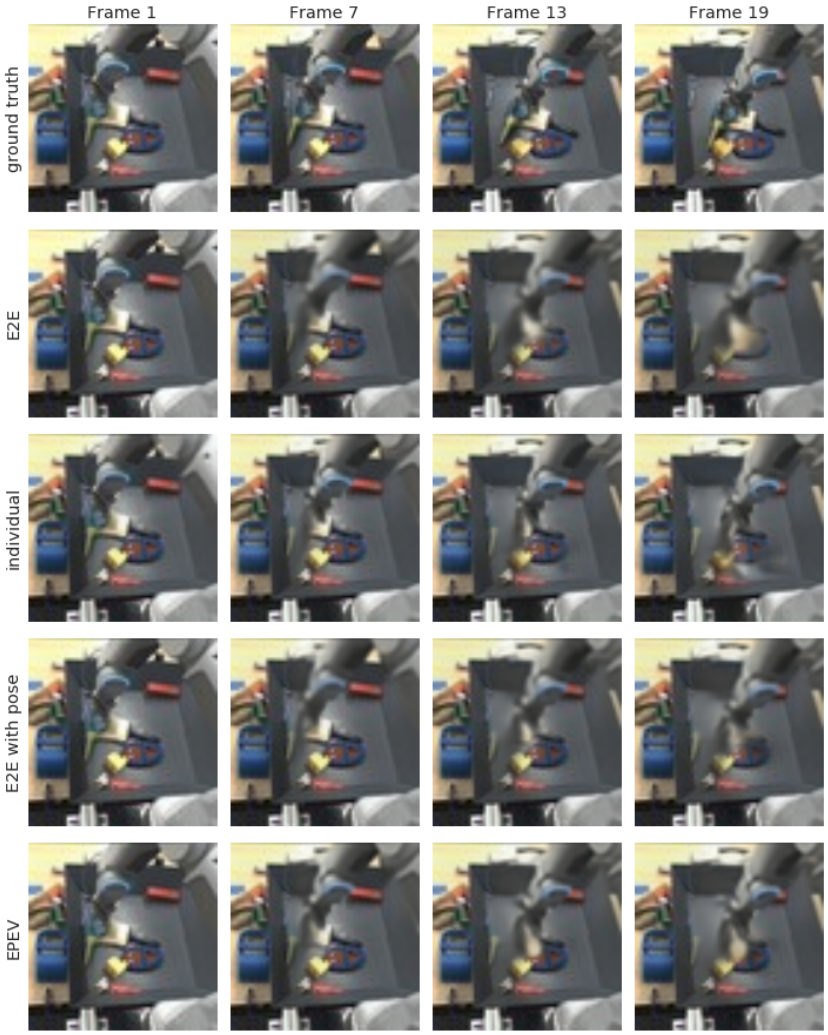


Figure 5: A visual comparison of the different methods on the robot push dataset. In the E2E and EPEV methods, the yellow square objects moves, where it doesn't in the individual method.

4.2 HUMANS 3.6M

Our method was also tested on the Humans 3.6M Dataset. Only theE2EandEPEV 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 theE2Emethod and a encoding size of 64 for the EPEV method on this dataset.

InE2Emethod, 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 theE2Emethod, the predictor can

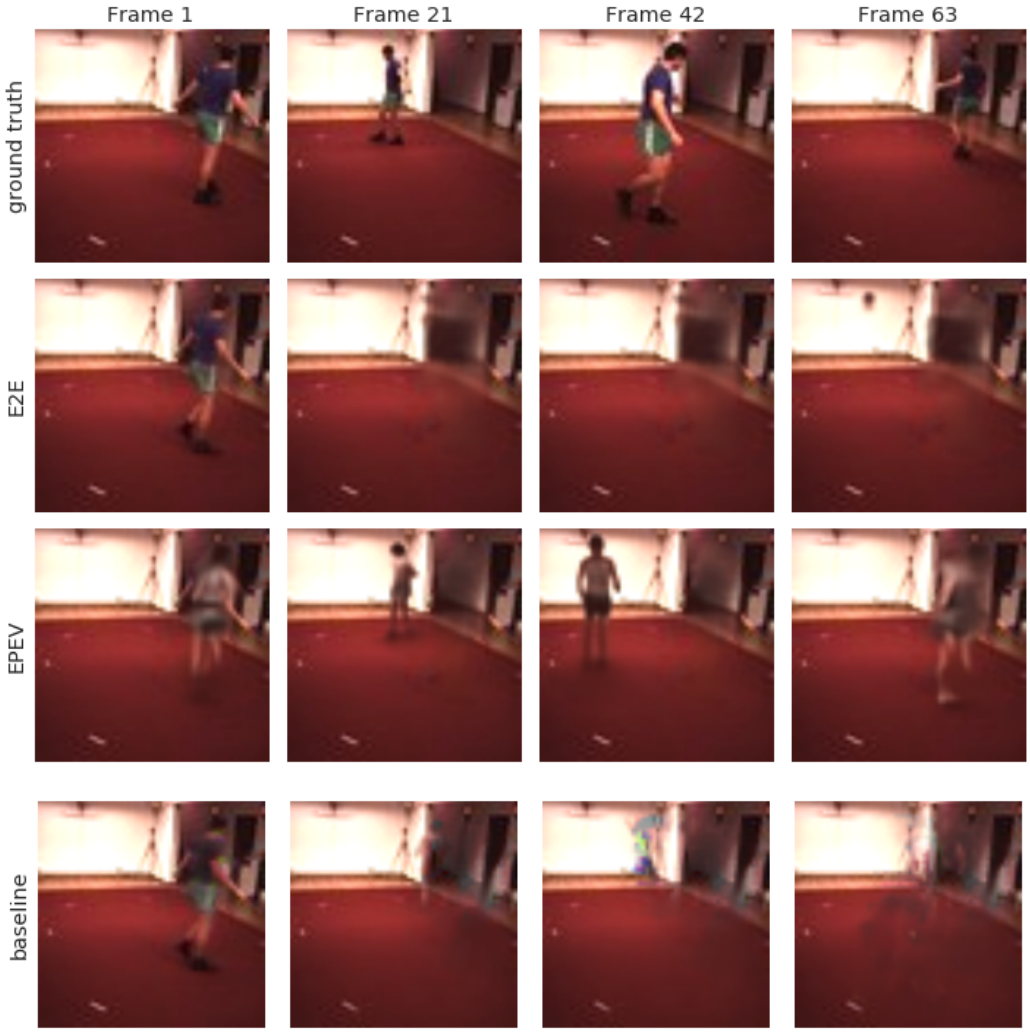


Figure 6: A visual comparison of the different methods on the Humans Dataset. the baseline method CDNA from Finn et al. (2016). This example is cherry picked to highlight the problem of the the figure disappearing in theE2Eapproach. See the appendix for non cherry picked results. The contrast of these images was increased to make the humans easier to see.

encode its confidence in the prediction, and the VAN won't generate the person if the predictor isn't 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 don't learn to do this. As a result, as long as the predictor network predicts a valid encoding, the person is generated. When theEPEV does generate a image where the human disappears, it's probably because the predictor predicted an invalid encoding so the VAN couldn't generate a reasonable human image.

In the predictions from the EPEV approach the person moves around the room in a circle similar to the ground truth. In the later frames, the prediction differs more and more from the ground truth. On the 63rd generated frame, the person is in the back of the room, while the person is near the center in the ground truth frame. This shows that the method can make reasonably sharp predictions even though the dataset is too complex for the predictor network to predict precisely where the person will be.

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 theE2Emethod 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, their predictor is trained on the ground truth pose, that makes the predictor less likely to generate a invalid pose which the VAN won't know how to interpret.

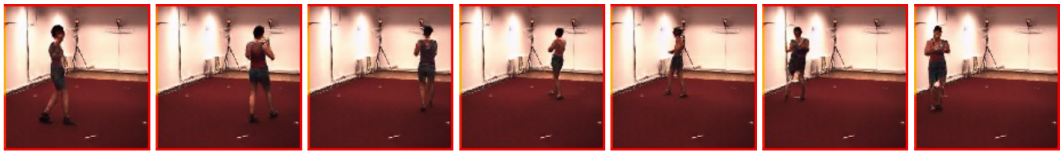


Figure 7: Results taken from the paper Villegas et al. (2017) predicting about 10 seconds into the future.

To see what the encoder has learned in theEPEV 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 8. The results show that the encoder encodes where the person is in

Figure 8: Results from theEPEV approach when the VAN is given the output of the encoder on the ground truth frame.

the image, as well as the orientation of the arms, legs and head to some extent. The results aren't 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.

5 Conclusion

This method is an improvement upon Villegas et al. (2017) on datasets where collecting a ground truth pose is impractical, or the pose doesn't fully capture the information needed to produce a frame from the first one.

On the robot push dataset theE2Emodel works better thanEPEV, but on the Humans Dataset, theEPEV 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 theEPEV method, the predictor and VAN are trained separately, so they can't learn to make the images more blurry over time.

In theEPEV method, the network learned a high level encoding that is both easy to predict, and informative enough to generate the nth frame from the first one. This could have applications in video compression because the network can recreate the video given only the first frame and the encodings as shown in figure 8. This high level encoding can also be used for motion classification. The encoding represents what has changed from the first frame so the VAN can use it to construct the nth frame. Therefore a motion classifier could probably be trained with fewer training examples if it was trained to classify the encodings instead of the images themselves.

6 FUTURE WORK

The reason the person sometimes disappears in the Humans DatasetEPEV might be because the predictor isn't predicting a valid encoding. The dataset is non deterministic, so the predictor network will not know for sure what the encoding of the next frame will be. In order to minimize the L2 loss, the best the predictor can do is average the possible predictions, which might not be a valid encoding. The VAN won't know what to do with this prediction and may generate an image without a person.

To fix this, 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. 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). TheEPEV 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 computed 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.

ACKNOWLEDGMENTS

REFERENCES

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pp. 343–351, 2016.

Cristian Sminchisescu Catalin Ionescu, Fuxin Li. Latent structured models for human pose estimation. In *International Conference on Computer Vision*, 2011.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, pp. 64–72, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *ArXiv e-prints*, July 2016.

William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*. 2017.

Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*. 2016.

V. Michalski, R. Memisevic, and Kishore Konda. Modeling deep temporal dependencies with recurrent "grammar cells". In *NIPS*, 2014.

Roni Mittelman, Benjamin Kuipers, Silvio Savarese, and Honglak Lee. Structured recurrent temporal restricted boltzmann machines. In *ICML*. 2014.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*. 2015.

Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1252–1260. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5845-deep-visual-analogy-making.pdf`.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15 (1):1929–1958, January 2014. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=2627435.2670313`.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*. 2015.

Ilya Sutskever, Geoffrey E. Hinton, and Graham W. Taylor. The recurrent temporal restricted boltzmann machine. In *NIPS*. 2009.

Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 3560–3569, 2017.

Appendices

A Implementation details

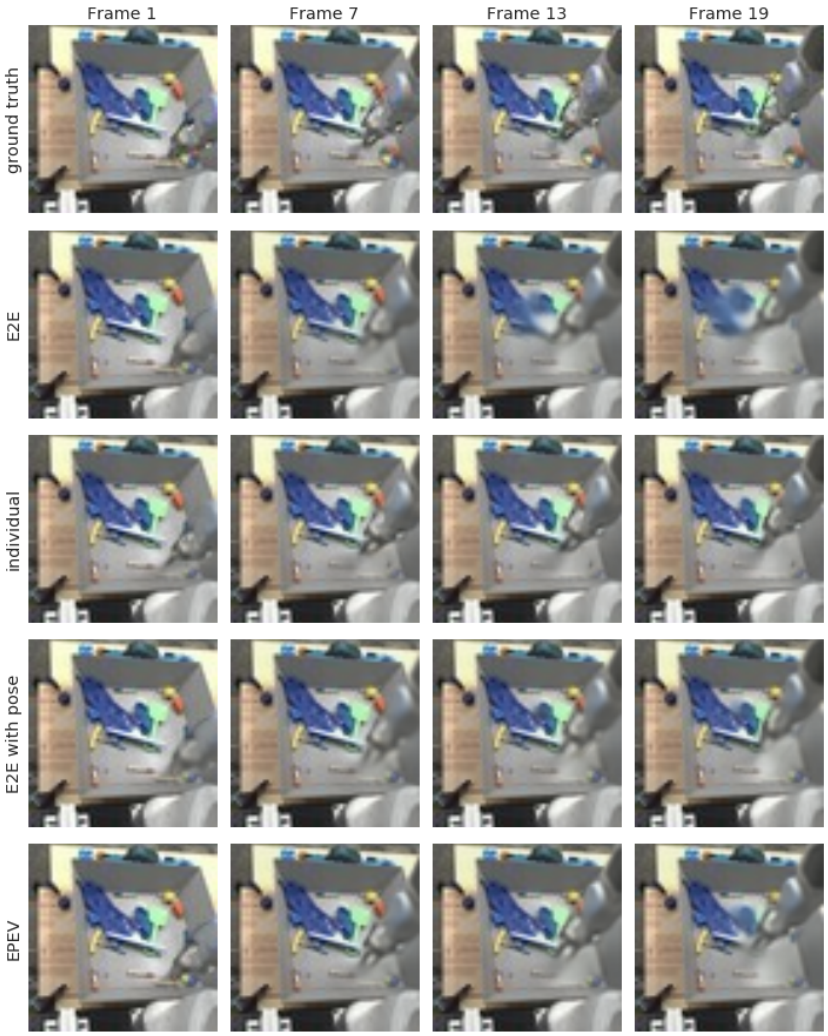
The convolutional neural net encoder is a VGG-16 network (Simonyan & Zisserman, 2014). In the EPEV method, the encoder needs to be pre-trained on Imagenet (Deng et al., 2009). If that is not done, the VAN will ignore the output from the encoder. All other configurations are run without pretraining.

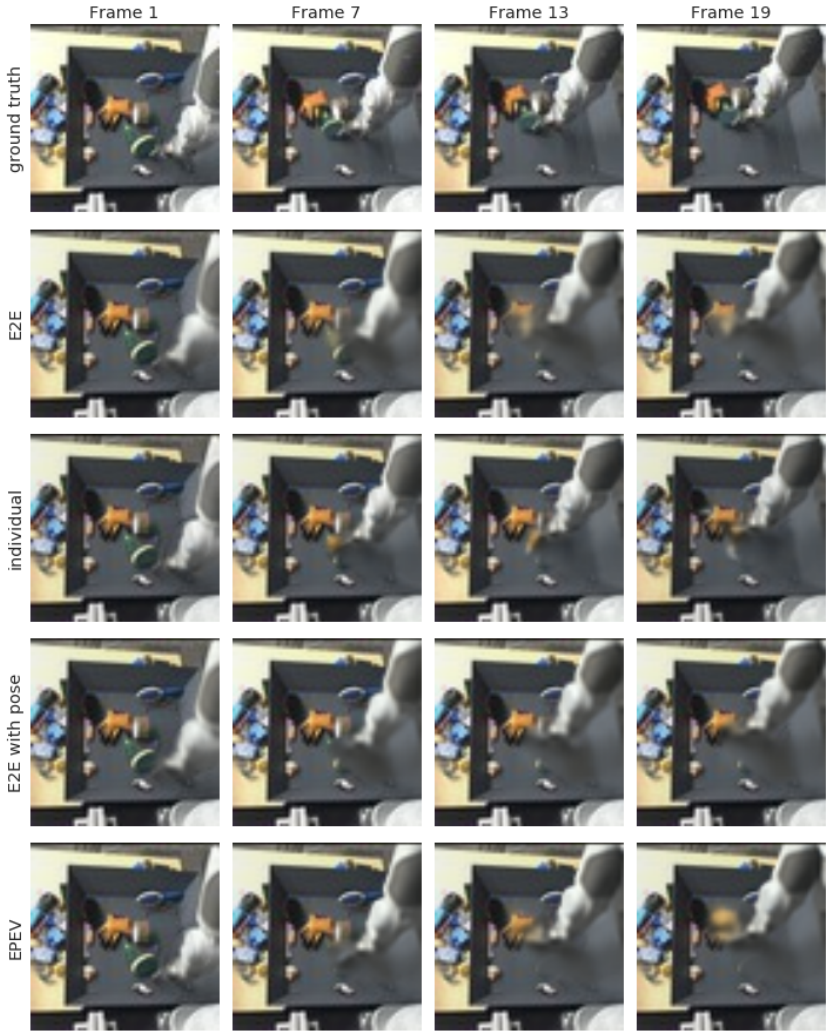
The predictor network is a 20 layer LSTM, while the analogy network uses the deep analogy transformation described in Reed et al. (2015). The VAN outputs a mask which controls whether to use its own output, or the pixels of the first frame. This allows the network to focus on learning the changing parts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

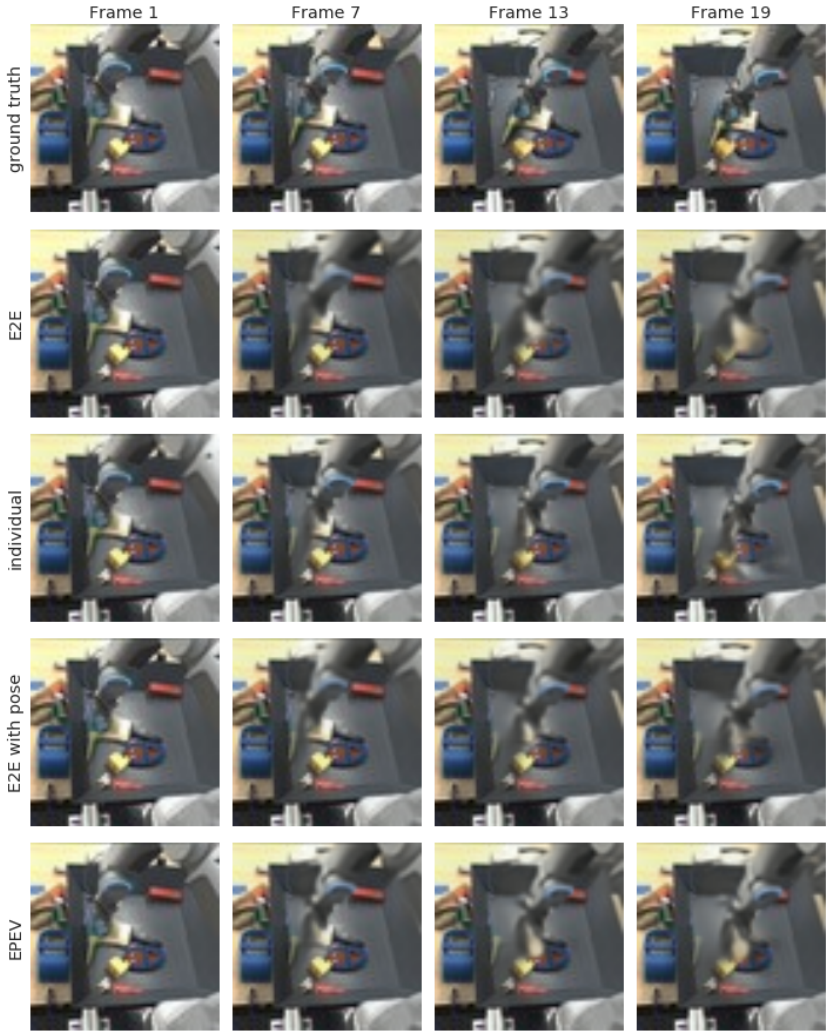
The end effector orientation is converted to quaternions (Bousmalis et al., 2016) in order to calculate the loss if the pose is used. Layer normalization (Lei Ba et al., 2016) is used between every other layer in the VAN and predictor networks. We used dropout (Srivastava et al., 2014) on the encoder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

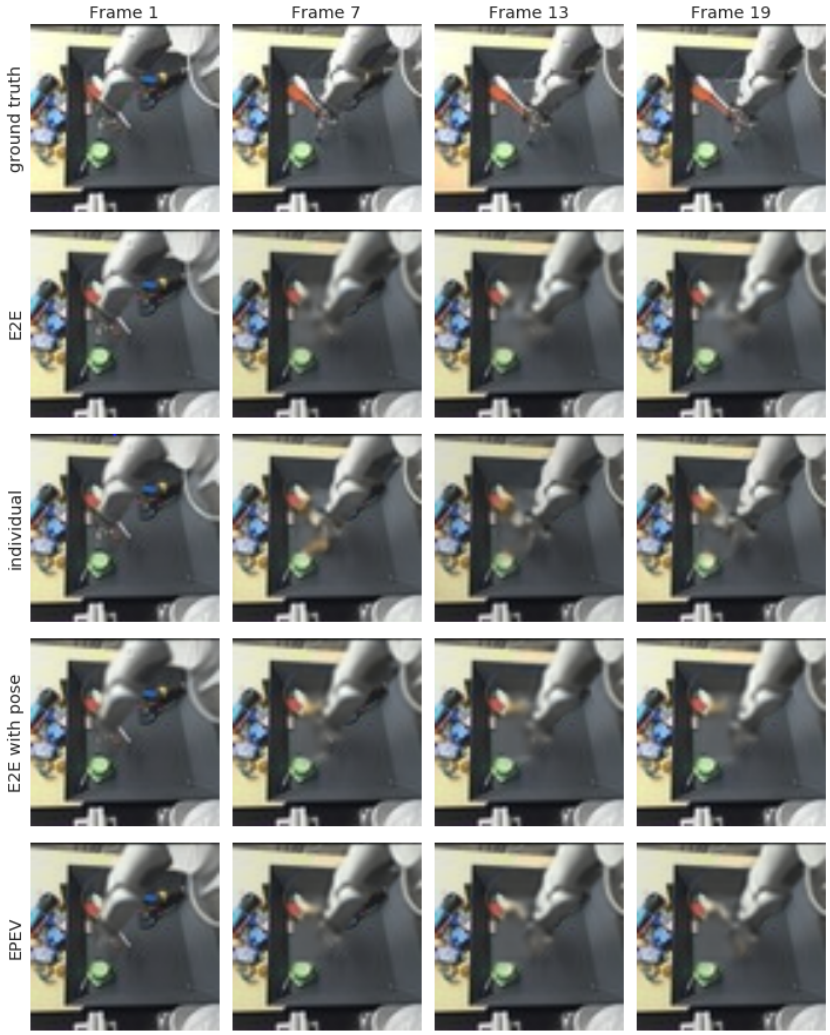
B MORE RESULTS ON ROBOT PUSH DATASET.

B.1 FRAMES WHEREEPEV PREDICTS OBJECTS BETTER THAN INDIVIDUAL.

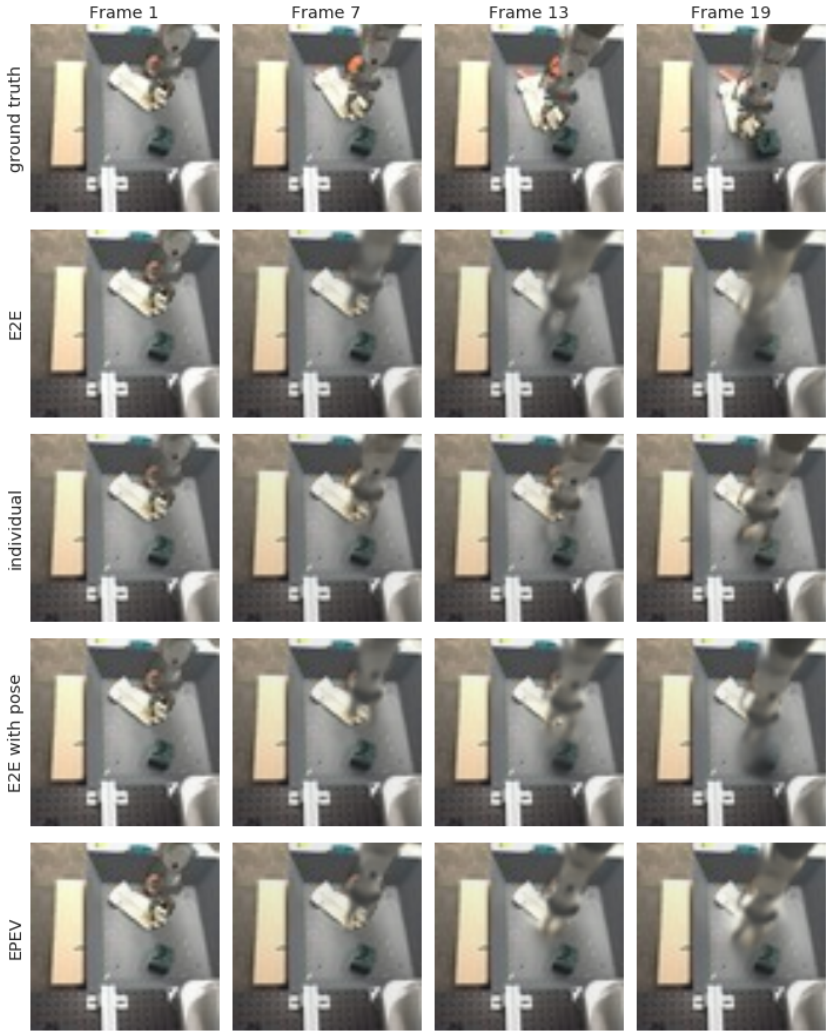



Frame 1
Frame 7
Frame 13
Frame 19
ground truth
E2E
individual
E2E with pose
EPEV

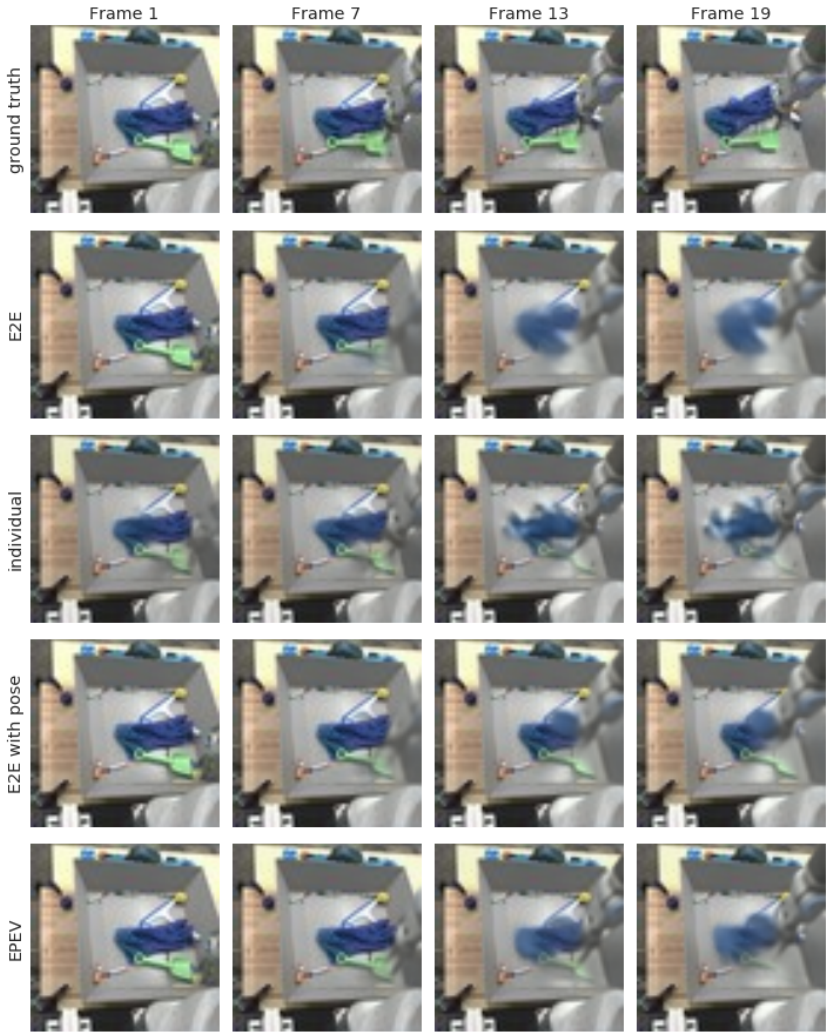

Frame 1
Frame 7
Frame 13
Frame 19
ground truth
E2E
individual
E2E with pose
EPEV

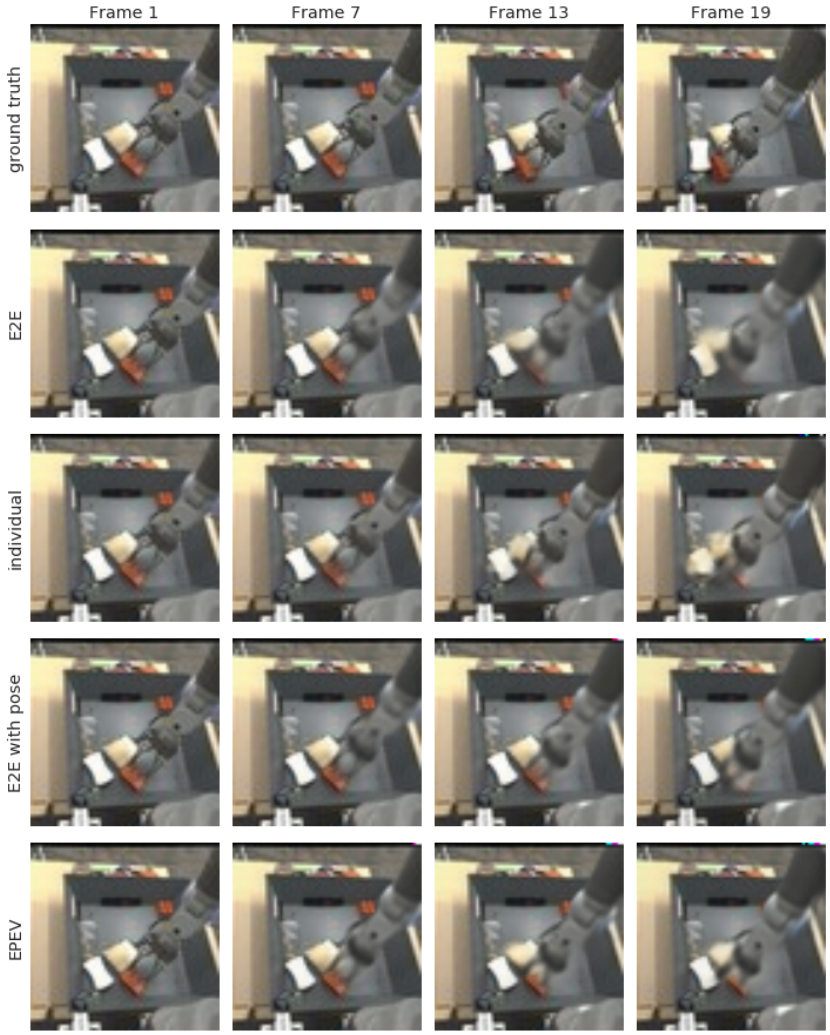

Frame 1
Frame 7
Frame 13
Frame 19
ground truth
E2E
individual
E2E with pose
EPEV

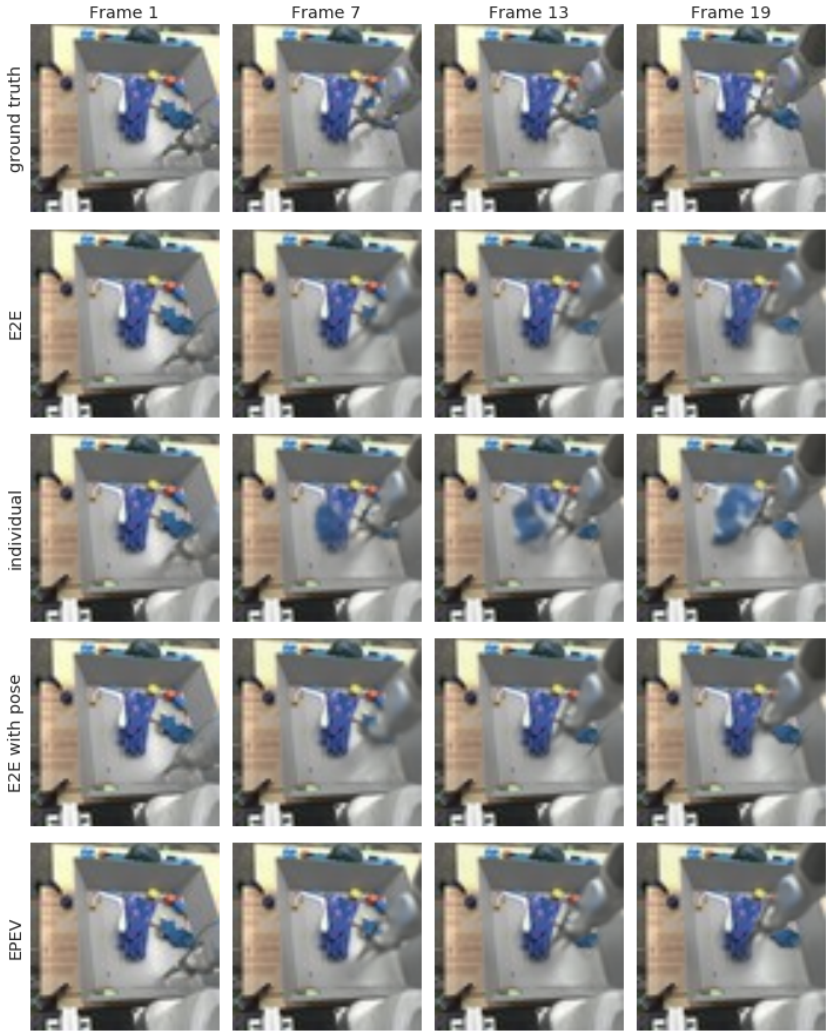
Frame 1
Frame 7
Frame 13
Frame 19
ground truth
E2E
individual
E2E with pose
EPEV


Frame 1
Frame 7
Frame 13
Frame 19
ground truth
E2E
individual
E2E with pose
EPEV

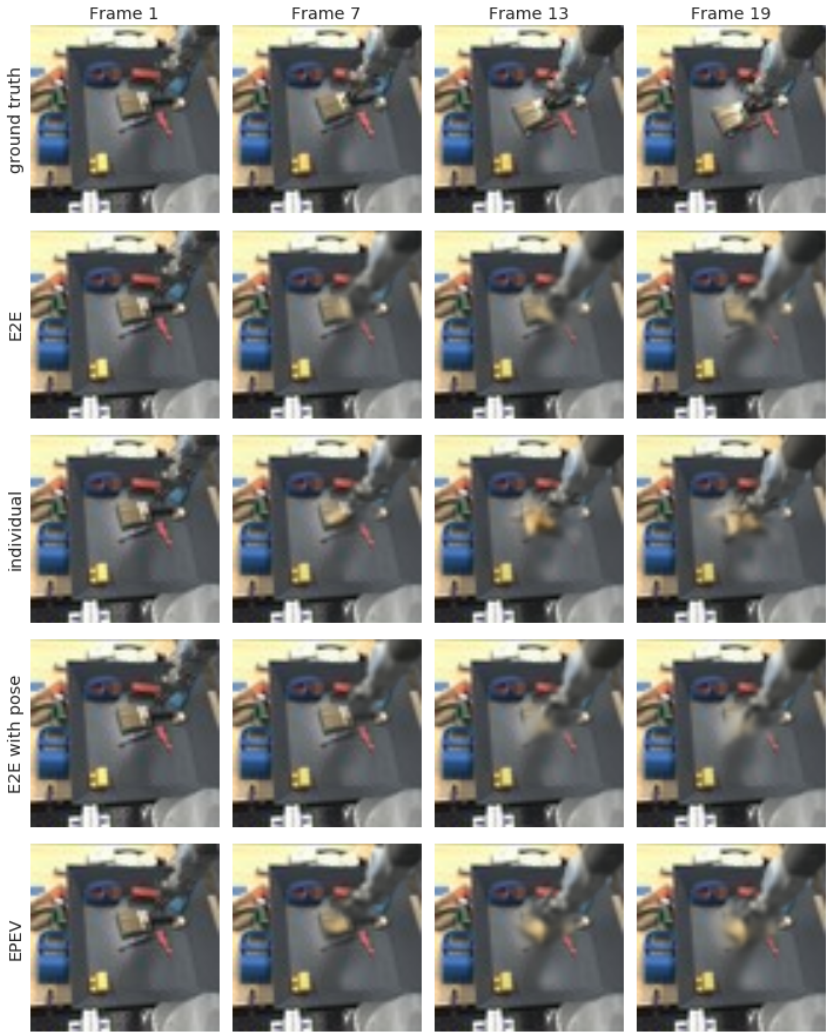
B.2 FRAMES WHERE INDIVIDUAL PREDICTS OBJECTS BETTER THAN EPEV.



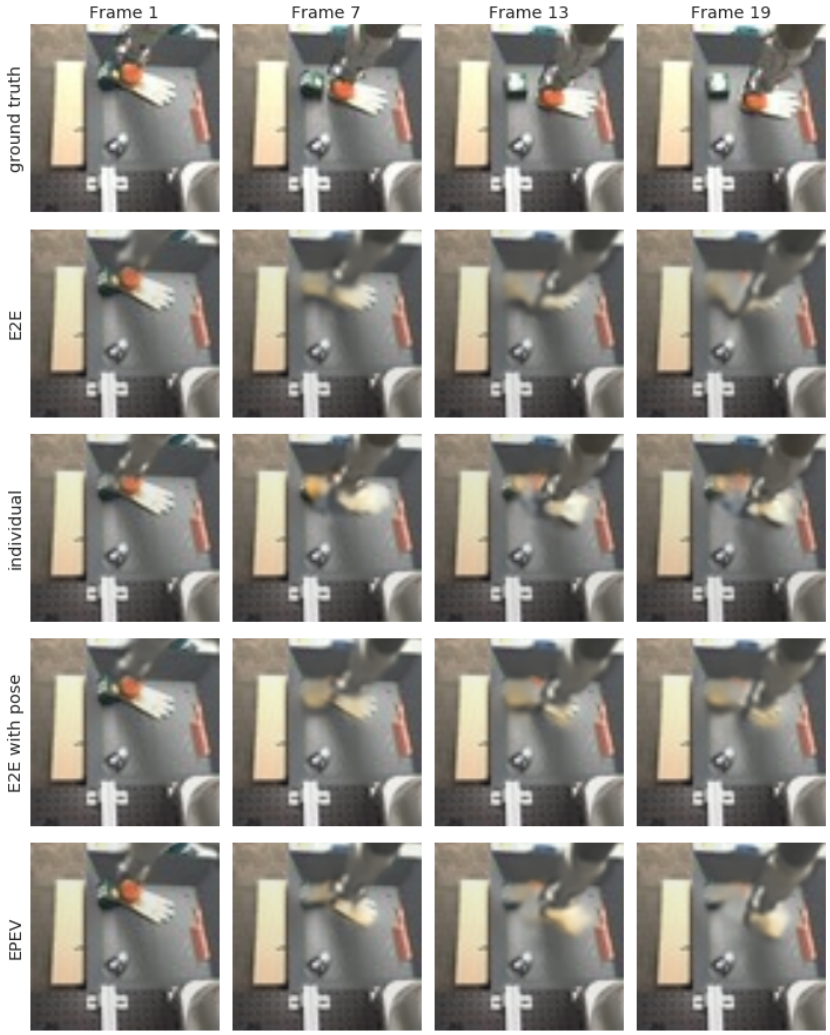

Frame 1
Frame 7
Frame 13
Frame 19
ground truth
E2E
individual
E2E with pose
EPEV


Frame 1
Frame 7
Frame 13
Frame 19
ground truth
E2E
individual
E2E with pose
EPEV

B.3 FRAMES WHEREEPEV AND INDIVIDUAL PREDICT OBJECTS WITH THE SAME QUALITY.

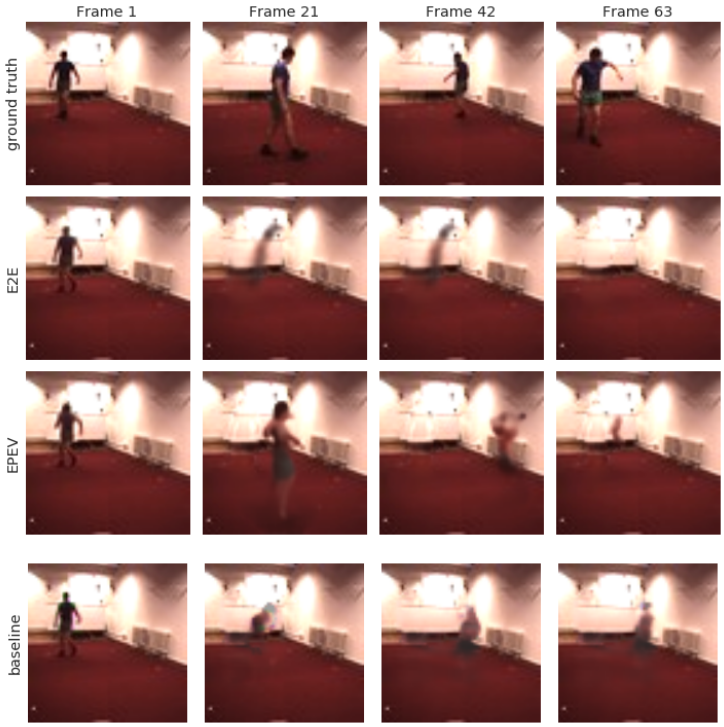


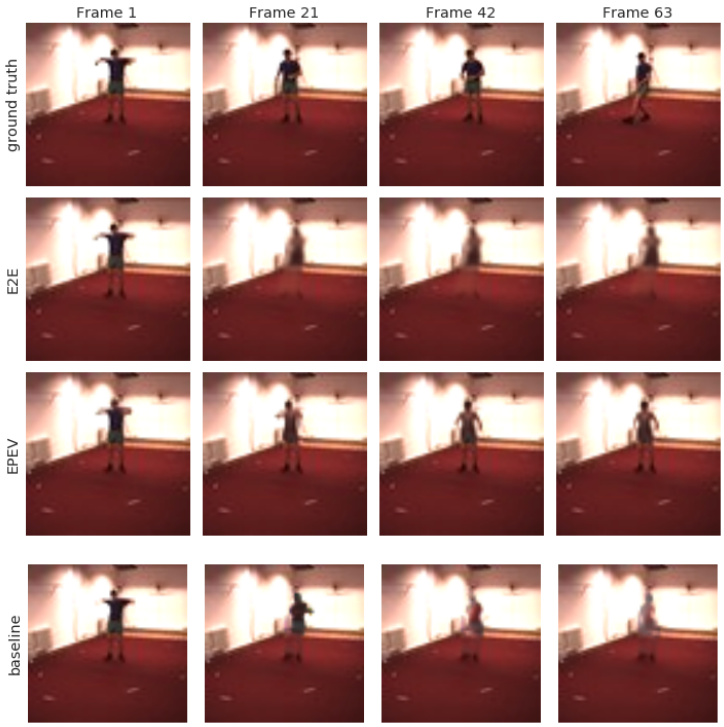
Frame 1
Frame 7
Frame 13
Frame 19
ground truth
E2E
individual
E2E with pose
EPEV

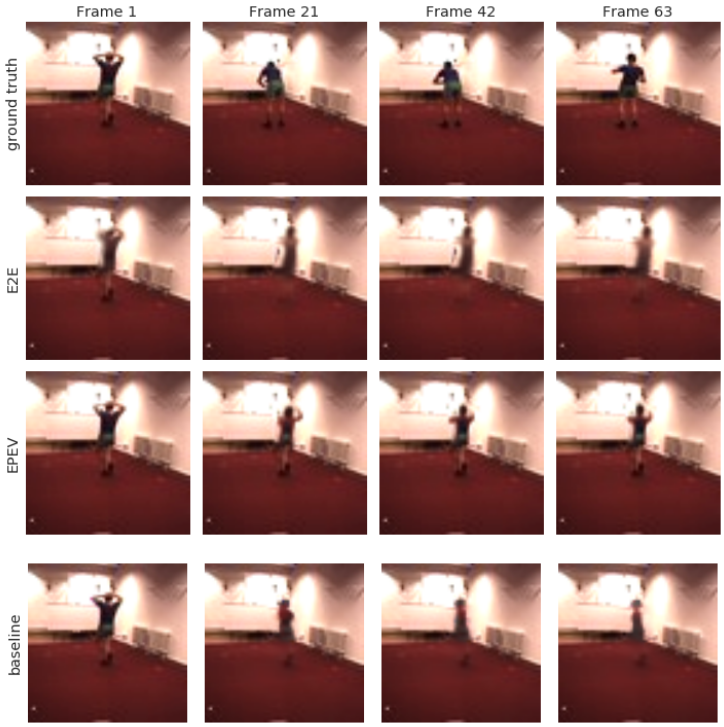


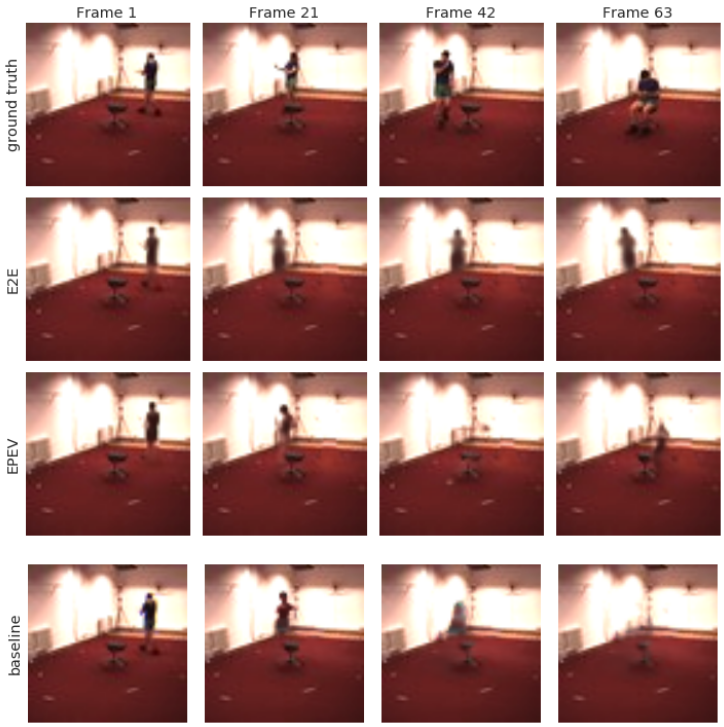
C MORE RESULTS ON HUMANS DATAEST.

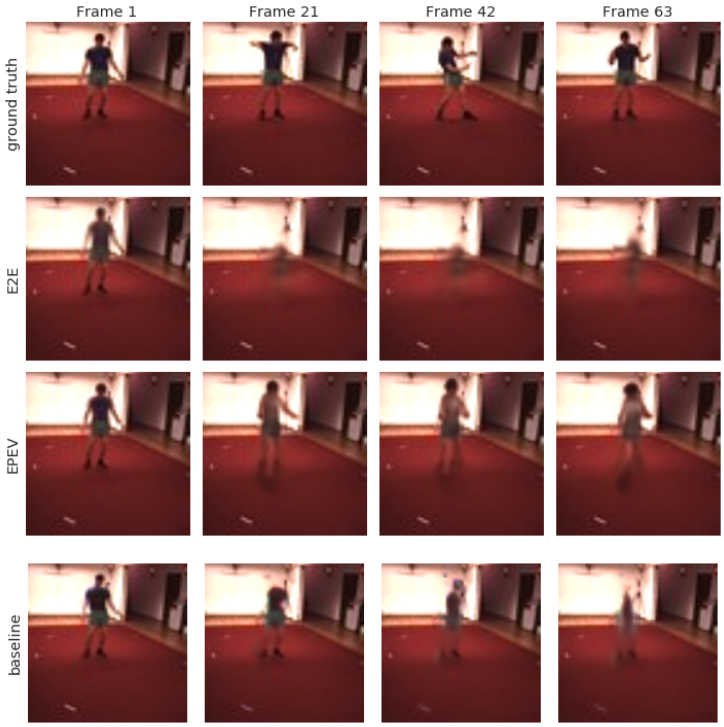
TODO (Dumitru) Group these by the action the human is taking.


Frame 1
Frame 21
Frame 42
Frame 63
ground truth
E2E
EPEV
baseline


Frame 1
Frame 21
Frame 42
Frame 63
ground truth
E2E
EPEV
baseline


Frame 1
Frame 21
Frame 42
Frame 63
ground truth
E2E
EPEV
baseline


Frame 1
Frame 21
Frame 42
Frame 63
ground truth
E2E
EPEV
baseline


Frame 1
Frame 21
Frame 42
Frame 63
ground truth
E2E
EPEV
baseline