PREDICTION UNDER UNCERTAINTY WITH ERROR-ENCODING NETWORKS

Mikael Henaff, Junbo Zhao and Yann LeCun
Facebook AI Research
Courant Institute, New York University

ABSTRACT

In this work we introduce a new framework for performing temporal predictions in the presence of uncertainty. It is based on a simple idea of disentangling components of the future state which are predictable from those which are inherently unpredictable, and encoding the unpredictable components into a low-dimensional latent variable which is fed into a forward model. Our method uses a supervised training objective which is fast and easy to train. We evaluate it in the context of video prediction on multiple datasets and show that it is able to consistently generate diverse predictions without the need for alternating minimization over a latent space or adversarial training.

1 INTRODUCTION

Learning forward models in time series is a central task in artificial intelligence, with applications in unsupervised learning, planning and compression. A major challenge in this task is how to handle the multi-modal nature of many time series. When there are multiple valid ways in which a time series can evolve, training a model using classical ℓ_1 or ℓ_2 losses produces predictions which are the average or median of the different outcomes across each dimension, which is itself often not a valid prediction.

In recent years, Generative Adversarial Networks (Goodfellow et al., 2014) have been introduced, a general framework where the prediction problem is formulated as a minimax game between the predictor function and a trainable discriminator network representing the loss. By using a trainable loss function, it is in theory possible to handle multiple output modes since a generator which covers each of the output modes will fool the discriminator leading to convergence. However, a generator which covers a single mode can also fool the discriminator and converge, and this behavior of mode collapse has been widely observed in practice. Some workarounds have been introduced to resolve or partially reduce mode-collapsing, such as minibatch discrimination, adding parameter noise (Salimans et al., 2016), backpropagating through the unrolled discriminator (Metz et al., 2016) and using multiple GANs to cover different modes (Tolstikhin et al., 2017). However, many of these techniques can bring additional challenges such as added complexity of implementation and increased computational cost. The mode collapsing problem becomes even more pronounced in the conditional generation setting when the output is highly dependent on the context, such as video prediction (Mathieu et al., 2015; Isola et al., 2016).

In this work, we introduce a novel architecture that allows for robust multimodal conditional predictions in time series data. It is based on a simple intuition of separating the future state into a deterministic component, which can be predicted from the current state, and a stochastic (or difficult to predict) component which accounts for the uncertainty regarding the future mode. By training a deterministic network, we can obtain this factorization in the form of the network's prediction together with the prediction error with respect to the true state. This error can be encoded as a low-dimensional latent variable which is fed into a second network which is trained to accurately correct the determinisic prediction by incorporating this additional information. We call this model the Error Encoding Network (EEN). In a nutshell, this framework contains three function mappings at each timestep: (i) a mapping from the current state to the future state, which separates the future state into deterministic and non-deterministic components; (ii) a mapping from the non-deterministic component of the future state to a low-dimensional latent vector; (iii) a mapping from the current state to

the future state conditioned on the latent vector, which encodes the mode information of the future state. While the training procedure involves all these mappings, the inference phase involves only (iii).

Both networks are trained end-to-end using a supervised learning objective and latent variables are computed using a learned parametric function, leading to easy and fast training. We apply this method to video datasets from games, robotic manipulation and simulated driving, and show that the method is able to consistently produce multimodal predictions of future video frames for all of them. Although we focus on video in this work, the method itself is general and can in principle be applied to any continuous-valued time series.

2 MODEL

Many natural processes carry some degree of uncertainty. This uncertainty may be due to an inherently stochastic process, a deterministic process which is partially observed, or it may be due to the complexity of the process being greater than the capacity of the forward model. One natural way of dealing with uncertainty is through latent variables, which can be made to account for aspects of the target that are not explainable from the observed input.

Assume we have a set of continuous vector-valued input-target pairs (x_i, y_i), where the targets depend on both the inputs and some inherently unpredictable factors. For example, the inputs could be a set of consecutive video frames and the target could be the following frame. Classical latent variable models such as k-means or mixtures of Gaussians are trained by alternately minimizing the loss with respect to the latent variables and model parameters; in the probabilistic case this is the Expectation-Maximization algorithm (Dempster et al., 1977). In the case of a neural network model $f_\theta(x_i, z)$, continuous latent variables can be optimized using gradient descent and the model can be trained with the following procedure:

Algorithm 1 Train latent variable model with alternating minimization

Require: Learning rates α, β, number of iterations K.
repeat
  Sample (x_i, y_i) from the dataset
  initialize $z \sim \mathcal{N}(0, 1)$
  $i \leftarrow 1$
  while $i \leq K$ **do**
    $z \leftarrow z - \alpha \nabla_z \mathcal{L}(y_i, f_\theta(x_i, z))$
    $i \leftarrow i + 1$
  $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}(y_i, f_\theta(x_i, z))$
until converged

Our approach is based on two observations. First, the latent variable z should represent what is not explainable using the input x_i. Ideally, the model should make use of the input x_i and only use z to account for what is not predictable from it. Second, if we are using gradient descent to optimize the latent variables, z will be a smooth function of x_i and y_i, although a possibly highly nonlinear one. We train two functions to predict the targets: a deterministic model $g(x)$ and a latent variable model $f(x, z)$. We first train the deterministic model g to minimize the following loss over the training set:

$$\mathcal{L}_g = \sum_i \|y_i - g(x_i)\| \tag{1}$$

Here the norm can denote ℓ_1, ℓ_2 or any other loss which is a function of the difference between the target and the prediction. Given sufficient data and capacity, g will learn to extract all the information possible about each y_i from the corresponding x_i, and what is inherently unpredictable will be contained within the residual error, $y_i - g(x_i)$. This can be passed through a learned parametric function ϕ to produce a low-dimensional latent variable $z = \phi(y_i - g(x_i))$ which encodes the identity of the mode to which the future state belongs. This then be used as input to f to more accurately predict y_i conditioned on knowledge of the proper mode.

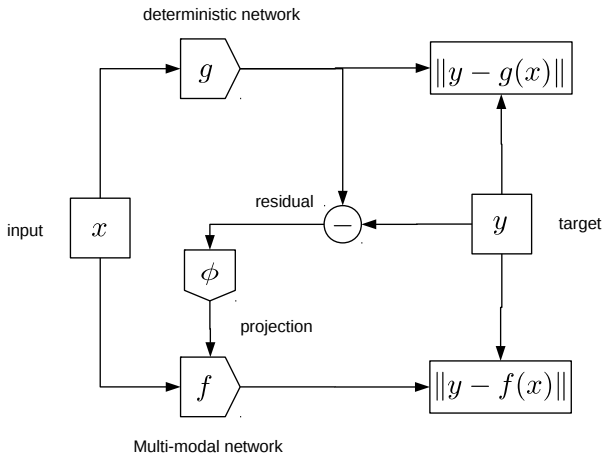


Figure 1: Model Architecure.

Once g is trained, we then minimize the following loss over the training data:

$$\mathcal{L}_f = \sum_i \|y_i - f(x_i, \phi(y_i - g(x_i))\| \tag{2}$$

The fact that z is a function of the prediction error of g reflects the intuition that it should only account for what is not explainable by the input, while still being a smooth function of x and y. Note that z is typically of much lower dimension than $y_i - g(x_i)$, which prevents the network from learning a trivial solution where f would simply invert ϕ and cancel the error from the prediction. This forces the network to map the errors to general representations which can be reused across different samples and correspond to different modes of the conditional distribution.

To perform inference after the network is trained, we first extract and save the $z_i = \phi(y_i - g(x_i))$ from each sample in the training set. Given some new input x', we can then generate different predictions by computing $f(x', z')$, for different $z' \in \{z_i\}$. In this work, we adopt a simple strategy of sampling uniformly from this set to generate new samples, however more sophisticated methods could be used such as fitting a conditional distribution over $p(z|x)$ and sampling from it.

The model architecture is shown in Figure 1. To speed up training, we can choose f and g to have similar architecture and initialize f with the parameters of g. For example, we can have $g(x) = g_2(g_1(x))$ and $f(x, z) = f_2(f_1(x)+Wz)$ and initialize f_1, f_2 with the weights of g_1, g_2 respectively. Thus, if ϕ and W are initialized such that Wz has zero mean and small variance, the network is already able to extract all the signal from the input x and can only improve further by adapting its weights to make use of the mode information contained in z.

3 Related Work

In recent years a number of works have explored video prediction. These typically train models to predict future frames with the goal of learning representations which disentangle factors of variation and can be used for unsupervised learning (Srivastava et al., 2015; Villegas et al., 2017; Denton & Birodkar, 2017), or learn action-conditional forward models which can be used for planning (Oh et al., 2015; Finn et al., 2016; Agrawal et al., 2016; Kalchbrenner et al., 2016). In the first case, the predictions are deterministic and ignore the possibly multimodal nature of the time series. In the second, it is possible to make different predictions about the future by conditioning on different actions, however this requires that the training data includes additional action labels. Our work

makes different predictions about the future by instead conditioning on latent variables which are extracted in an unsupervised manner from the videos themselves.

Several works have used adversarial losses in the context of video prediction. The work of (Mathieu et al., 2015) used a multiscale architecture and a combination of several different losses to predict future frames in natural videos. They found that the addition of the adversarial loss and a gradient difference loss improved the generated image quality, in particular by reducing the blur effects which are common when using ℓ_2 loss. However, they also note that the generator learns to ignore the noise and produces similar outputs to a deterministic model trained without noise. This observation was also made by (Isola et al., 2016) when training conditional networks to perform image-to-image translation.

Other works have used models for video prediction where latent variables are inferred using alternating minimization. The model in (Vondrick et al., 2015) includes a discrete latent variable which was used to choose between several different networks for predicting hidden states of future video frames obtained using a pretrained network. This is more flexible than a purely deterministic model, however the use of a discrete latent variable still limits the possible future modes to a discrete set. The work of (Goroshin et al., 2015) also made use of latent variables to model uncertainty, which were inferred through alternating minimization. In contrast, our model infers continuous latent variables through a learned parametric function. This is related to algorithms which learn to predict the solution of an iterative optimization procedure (Gregor & LeCun, 2010).

Recent work has shown that good generative models can be learned by jointly learning representations in a latent space together with the parameters of a decoder model (Bojanowski et al., 2017). This leads to easier training than adversarial networks. This generative model is also learned by alternating minimization over the latent variables and parameters of the decoder model, however the latent variables for each sample are saved after each update and optimization resumes when the corresponding sample is drawn again from the training set. This is related to our method, with the difference that rather than saving the latent variables for each sample we compute them through a learned function of the deterministic network's prediction error.

Our work is related to predictive coding models (Rao & Ballard, 1999; Spratling, 2008; Chalasani & Principe, 2013; Lotter et al., 2016) and chunking architectures (Schmidhuber, 1992), which also pass residual errors or incorrectly predicted inputs between different parts of the network. It differs in that these models pass errors upwards to higher layers in the network at each timestep, whereas our model passes the compressed error signal from the deterministic network backwards in time to serve as input to the latent variable network at the previous timestep.

4 EXPERIMENTS

We tested our method on five different video datasets from different areas such as games (Atari Breakout, Atari Seaquest and Flappy Bird), robot manipulation (Agrawal et al., 2016) and simulated driving (Zhang & Cho, 2016). These have a well-defined multimodal structure, where the environment can change due to the actions of the agent or other stochastic factors and span a diverse range of visual environments. For each dataset, we trained our model to predict the following 1 or 4 frames conditioned on the previous 4 frames. We also trained a deterministic baseline model and a GAN to compare performance. Code to train our models and obtain video generations is available at `https://github.com/mbhenaff/EEN`.

The deterministic model and EEN were trained using the ℓ_2 loss. Although more sophisticated losses exist, such as the Gradient Difference loss (Mathieu et al., 2015), our goal here was to evaluate whether our model could capture multimodal structure such as objects moving or appearing on the screen or perspective changing in multiple different realistic ways. We used the same architecture across all tasks, namely a 3-layer convolutional network followed by a 3-layer deconvolutional network, all with 64 feature maps at each layer and batch normalization. We did not use pooling and instead used strided convolutions, similar to the DCGAN architecture (Radford et al., 2015). The parametric function ϕ mapping the prediction error to latent variables was also a multilayer convolutional network followed by two fully-connected layers. For Atari Breakout we used 2 latent variables, for Seaquest and the Robot dataset we used 8, for Flappy Bird 16 and for driving 32. To train our network we used the ADAM optimizer (Kingma & Ba, 2014) with default parameters

and learning rate 0.001 or 0.0005. The deterministic baseline model and the GAN had the same encoder-decoder architecture as the EEN. Details can be found in our code release.

4.1 DATASETS

We now describe the video datasets we used.

Atari Games We used a pretrained A2C agent (Mnih et al., 2016) [1] to generate episodes of gameplay for the Atari games Breakout and Seaquest (Bellemare et al., 2012) using a standard video preprocessing pipeline, i.e. downsampling video frames to 84×84 pixels and converting to grayscale. We then trained our forward model using 4 consecutive frames as input to predict either the following 1 frame or 4 frames.

Flappy Bird We used the OpenAI Gym environment Flappy Bird [2] and had a human player play approximately 50 episodes of gameplay. In this environment, the player controls a moving bird which must navigate between obstacles appearing at different heights. We trained the model to predict the next 4 frames using the previous 4 frames as input, all of which were rescaled to 128×72 pixel color images.

Robot Manipulation We used the dataset of (Agrawal et al., 2016), which consists of 240×240 pixel color images of objects on a table before and after manipulation by a robot. The robot pokes the object at a random location with random angle and duration causing it to move, hence the manipulation does not depend of the environment except for the location of the object. Our model was trained to take a single image as input and predict the following image.

Simulated Driving We used the dataset from (Zhang & Cho, 2016), which consists of color videos from the front of a car taken within the TORCS simulated driving environment. This car is driven by an agent whose policy is to follow the road and pass or avoid other cars while staying within the speed limit. Here we again trained the model to predict 4 frames using the 4 previous frames as input. Each image was rescaled to 160×72 pixels as in the original work.

4.2 RESULTS

Our experiments were designed to test whether our method can generate multiple realistic predictions given the start of a video sequence. We first report qualitative results in the form of visualizations. In addition to the figures in this paper, we provide video links which facilitate viewing. An example of generated frames in Atari Breakout is shown in Figure 2. For the baseline model, the image of the paddle gets increasingly diffuse over time which reflects the model's uncertainty as to its future location while the static background remains well defined. The residual, which is the difference between the ground truth and the baseline prediction, only depicts the movement of the ball and the paddle which the deterministic model is unable to predict. This is encoded into the latent variables z through the learned function ϕ which takes the residual as input. By sampling different z vectors from the training set, we obtain three different generations for the same conditioning frames. For these we see a well-defined paddle executing different movement sequences starting from its initial location. Additional results can be found here: `https://youtu.be/3R6BQA6wCKc`.

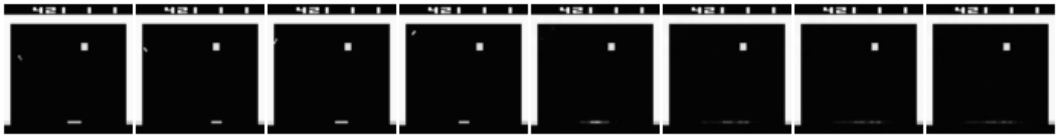
Figure 3 shows generations for Atari Seaquest. Again we see the baseline model captures most of the features on the screen except for the agent's movement, which appears in the residual. This is the information that will be encoded in the latent variables, and by sampling different latent variables we obtain the generations below where the submarine changes direction. Additional results can be found here: `https://youtu.be/UPx31nIQP_o`

We next evaluated our method on the Robot dataset. For this dataset the robot pokes the object with random direction and force which cannot be predicted from the current state. The prediction of the baseline model blurs the object but does not change its location or angle. In contrast, our model is able to produce a diverse set of predictions where the object is moved to different adjacent locations, as shown in Figure 4. More results can be found here: `https://youtu.be/HmlhdJAr4E0`.

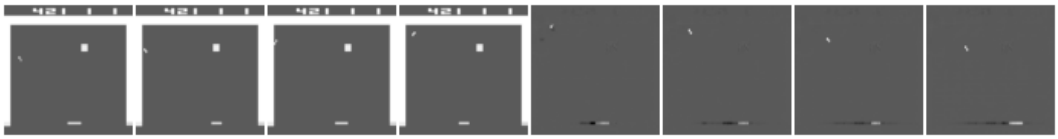
[1] https://github.com/ikostrikov/pytorch-a2c-ppo-acktr

[2] https://gym.openai.com/envs/FlappyBird-v0/

a) Ground truth



b) Deterministic Baseline



c) Residual

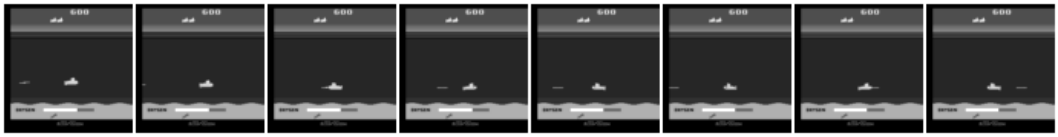
d) Generations with different z

Figure 2: Generations on Breakout. Left 4 frames are given, right 4 frames are generated. Best viewed with zoom.

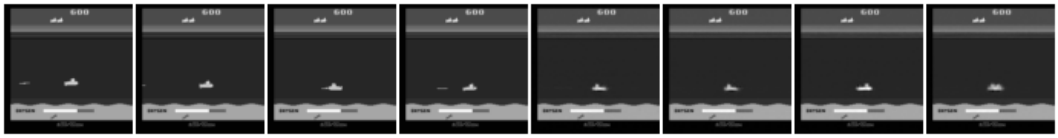
Figure 6 shows generated frames on Flappy Bird. Flappy Bird is a simple game which is deterministic except for two sources of stochasticity: the actions of the player and the height of new pipes appearing on the screen. In the first example, we see that by changing the latent variable we generate two sequences with pipes entering at different moments and heights and one sequence where no pipe appears. In the second example, changing the latent variable changes the height of the bird. The EEN is thus able to model both sources of uncertainty in the environment. Additional examples can be found here: `https://youtu.be/aJaSPzAUw74`.

The last dataset we evaluated our method on was the TORCS driving simulator. Here we found that generating frames with different z samples changed the location of stripes on the road and also produced shifts left and right as happens when turning the steering wheeel. These effects are best viewed though the following video link: `https://youtu.be/gY2PxHy2s7k`.

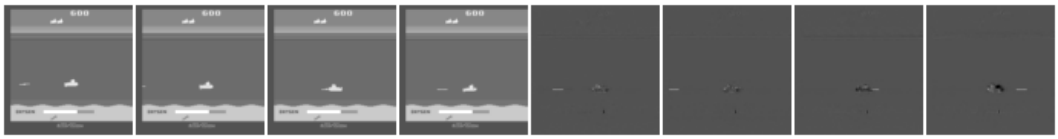
We next report quantitative results. Quantitatively evaluating multimodal predictions is not obvious, since the ground truth sample is drawn from one of several possible modes and the model may generate a sample from a different mode. In this case, simply comparing the generated sample to the ground truth sample may give high loss even if the generated sample is of high quality. We therefore



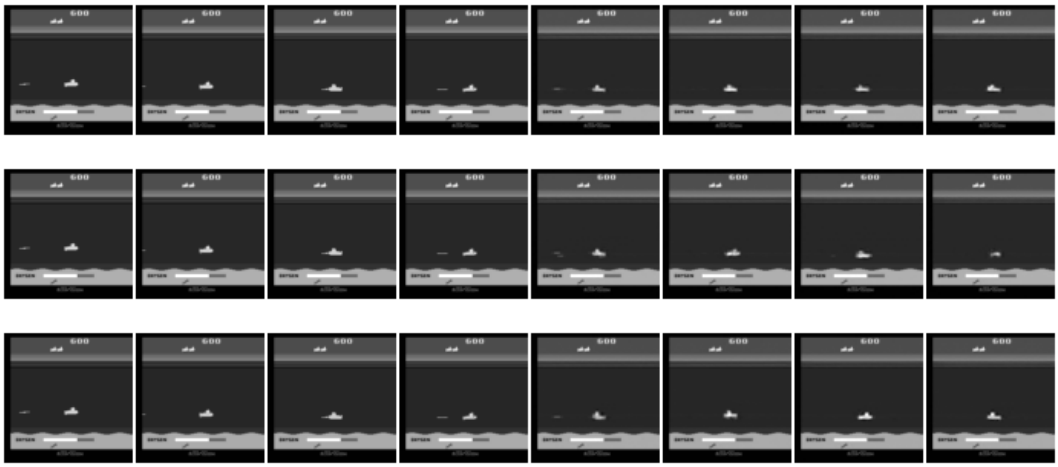
a) Ground Truth



a) Deterministic Baseline



c) Residual



b) EEN with different z

Figure 3: Generations on Seaquest. Left 4 frames are given, right 4 frames are generated. Best viewed with zoom.

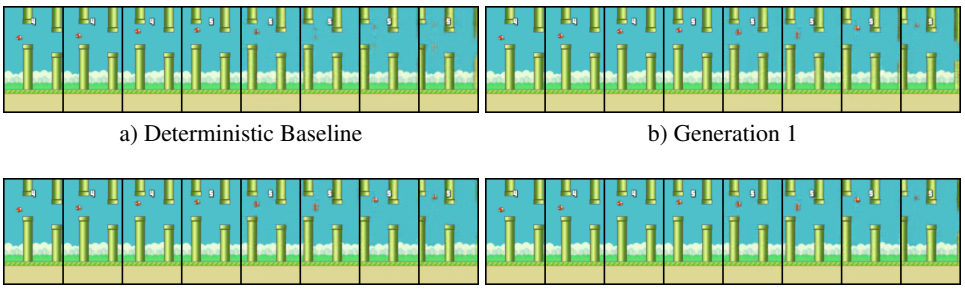
report the best score across different generated samples: $\min_k \mathcal{L}(y, f(x, z_k))$. If the multimodal model is able to use its latent variables to generate predictions which cover several modes, generating more samples will improve the score since it increases the chance that a generated sample will be from the same mode as the test sample. If however the model ignores latent variables or does not capture the mode that the test sample is drawn from, generating more samples will not improve the loss. Note that if $\mathcal{L}$ is a valid metric in the mathematical sense (such as the ℓ_1 or ℓ_2 distance), this is a finite-sample approximation to the Earth Mover or Wasserstein-1 distance between the true and generated distributions on the metric space induced by $\mathcal{L}$.

Figure 7 shows the best PSNR for different numbers of generated samples. We see that our model's best performance increases as more samples are generated, indicating that its generations are diverse enough to cover at least some of the modes of the test set. Also note that the GAN's performance does not change as we increase the number of samples generated, which indicates that its latent variables have little effect on the generated samples. This is consistent with findings in other work (Mathieu et al., 2015; Isola et al., 2016). We also note that the different models are not quite comparable to each other using PSNR since the baseline model is directly optimizing the ℓ_2 loss on which

a) Deterministic Baseline b) Generation 1

c) Generation 2 d) Generation 3

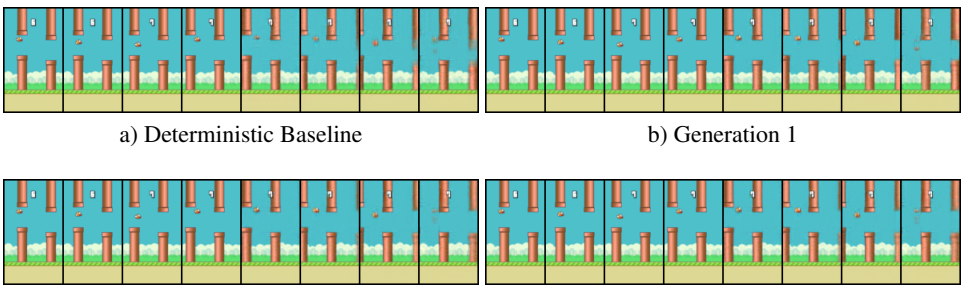
Figure 4: Generations on Robot Task. Left frame is given, right frame is generated.



a) Deterministic Baseline b) Generation 1

c) Generation 2 d) Generation 3

Figure 5: Generated frames on Flappy Bird. First 4 are given, last 4 are generated. Note that the pipe in the last frame appears at different heights. Best viewed with zoom.



a) Deterministic Baseline b) Generation 1

c) Generation 2 d) Generation 3

Figure 6: Generated frames on Flappy Bird. First 4 are given, last 4 are generated. Note that the bird changes height in the last generated frame. Best viewed with zoom.

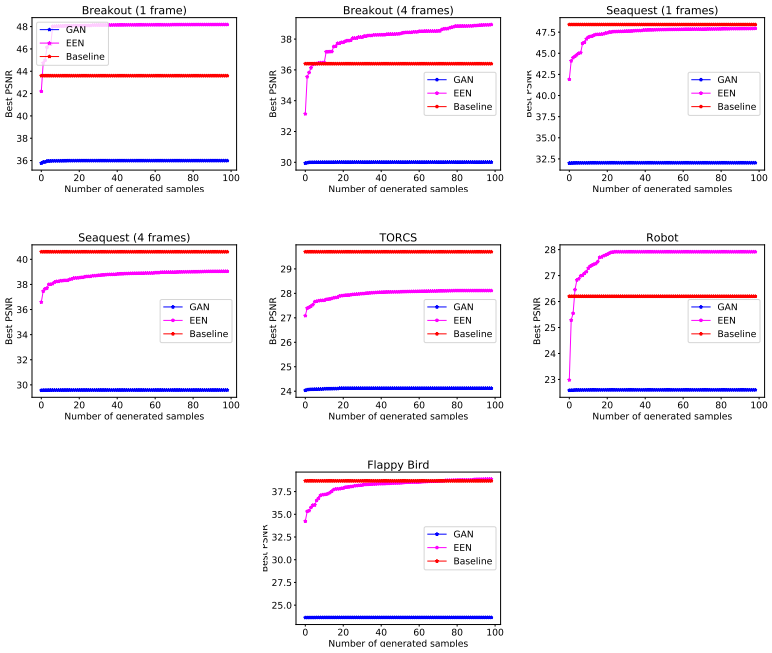


Figure 7: Top PSNR for different models over varying numbers of different samples.

it is based, the EEN is optimizing it conditioned on knowledge of a specific test sample, and the GAN is optimizing a different loss altogether. Our main goal is to illustrate that our model's performance improves by this approximate measure as is generates more samples, whereas the GAN does not due to mode collapse.

5 Conclusion

In this work, we have introduced a new framework for performing temporal prediction in the presence of uncertainty by disentangling predictable and non-predictable components of the future state. It is fast, simple to implement and easy to train without the need for an adverserial network or alternating minimization. We have provided one instantiation in the context of video prediction using convolutional networks, but it is in principle applicable to different data types and architectures. There are several directions for future work. Here, we have adopted a simple strategy of sampling uniformly from the z distribution without considering their possible dependence on the state x, and there are likely better methods. In addition, one advantage of our model is that it can extract latent variables from unseen data very quickly, since it simply requires a forward pass through a network. If latent variables encode information about actions in a manner that is easy to disentangle, this could be used to extract actions from large unlabeled datasets and perform imitation learning. Another interesting application would be using this model for planning and having it unroll different possible futures.

Acknowledgments

We would like to thank Jiakai Zhang and Kyunghyun Cho for sharing their dataset with us, and Martin Arjovsky, Arthur Szlam and Gabriel Synnaeve for helpful discussions.

References

Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419, 2016. URL `http://arxiv.org/abs/1606.07419`.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012. URL http://arxiv.org/abs/1207.4708.

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *CoRR*, abs/1707.05776, 2017. URL https://arxiv.org/abs/1707.05776.

Rakesh Chalasani and Jose C. Principe. Deep predictive coding networks. *CoRR*, abs/1301.3541, 2013. URL http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3541.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. *CoRR*, abs/1705.10915, 2017. URL http://arxiv.org/abs/1705.10915.

Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016. URL http://arxiv.org/abs/1605.07157.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Ross Goroshin, Michaël Mathieu, and Yann LeCun. Learning to linearize under uncertainty. *CoRR*, abs/1506.03011, 2015. URL http://arxiv.org/abs/1506.03011.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 399–406, 2010. URL http://www.icml2010.org/papers/449.pdf.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. URL http://arxiv.org/abs/1611.07004.

Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *CoRR*, abs/1610.00527, 2016. URL http://arxiv.org/abs/1610.00527.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

William Lotter, Gabriel Kreiman, and David D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *CoRR*, abs/1605.08104, 2016. URL http://arxiv.org/abs/1605.08104.

Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015. URL http://arxiv.org/abs/1511.05440.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016. URL http://arxiv.org/abs/1611.02163.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder P. Singh. Action-conditional video prediction using deep networks in atari games. *CoRR*, abs/1507.08750, 2015. URL http://arxiv.org/abs/1507.08750.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL http://arxiv.org/abs/1511.06434.

R.P.N. Rao and D.H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2:79–87, 1999.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL http://arxiv.org/abs/1606.03498.

Jrgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. 4:234–242, 03 1992.

M.W. Spratling. Predictive coding as a model of biased competition in visual attention. *Vision Research*, 48(12):1391 – 1408, 2008. ISSN 0042-6989. doi: https://doi.org/10.1016/j.visres.2008.03.009. URL http://www.sciencedirect.com/science/article/pii/S0042698908001466.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015. URL http://arxiv.org/abs/1502.04681.

Ilya O. Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. *CoRR*, abs/1701.02386, 2017. URL http://arxiv.org/abs/1701.02386.

Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *CoRR*, abs/1706.08033, 2017. URL http://arxiv.org/abs/1706.08033.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating the future by watching unlabeled video. *CoRR*, abs/1504.08023, 2015. URL http://arxiv.org/abs/1504.08023.

Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *CoRR*, abs/1605.06450, 2016. URL http://arxiv.org/abs/1605.06450.