PREDICTION UNDER UNCERTAINTY WITH ERROR ENCODING NETWORKS

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 the forward model. Our method uses a simple 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 of the environment is a central task in artificial intelligence, with applications in unsupervised learning, planning and compression. A major challenge in learning a forward model 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 $\ell_1$ or $\ell_2$ losses produces predictions which are the average or median of the different outcomes across each dimension, which is itself often not a realistic outcome.

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, notably in the context of video prediction (Mathieu et al., 2015; Isola et al., 2016). 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, none of these techniques has been proven to work in conditional generation, especially video prediction in which the context that the discriminator is conditioned on involves a very complex distribution. Another drawback in some of these approaches is the complexity of implementation and increased computational cost.

In this work, we introduce a novel architecture that allows for a robust multimodal conditional prediction 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 estimate) component which accounts for uncertainty of 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 deterministic prediction by incorporating this additional information. We call this model the Error Encoding Network (EEN). In a nutshell, this framework contains three function mappings in a recursive manner: (i)-mapping from the current state to the deterministic part of the future state; (ii)-mapping from the prediction error from (i) to a low-dimensional latent vector; (iii)-mapping from the latent vector back to pixel space, fixing the prediction in (i). While the training procedure involves all these mappings, the inference phase involves only (i) and (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 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 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 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. In the case of a neural network model \(f_\theta(x, z)\), 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

1: repeat
2: Sample \((x_i, y_i)\) from the dataset
3: initialize \(z \sim \mathcal{N}(0, 1)\)
4: \(i \leftarrow 1\)
5: while \(i \leq K\) do
6: \(z \leftarrow z - \alpha \nabla_z \mathcal{L}(y_i, f_\theta(x_i, z))\)
7: \(i \leftarrow i + 1\)
8: \(\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}(y_i, f_\theta(x_i, z^*))\)
9: 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:

\[
\sum_i \| y_i - g(x_i) \| \tag{1}
\]

Here the norm can denote \(\ell_2\), \(\ell_1\) 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 each \(x_i\), and what is inherently unpredictable will be contained within the residual error, \(y_i - g(x_i)\). This can be encoded into a latent variable \(z\) using a learned parametric function \(\phi\). The latent variable can then be used as input to \(f\) to more accurately predict \(y_i\). Once \(g\) is trained, we then minimize the following loss over the training data:

\[
\sum_i \| y_i - f(x_i, \phi(y_i - g(x_i)) \| \tag{2}
\]

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 \(y\). After
training, we can extract the $z_i = \phi(y_i - g(x_i))$ from each sample in the training set. Given some new input $x'$, we can generate different targets 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 $f(x) = f_2(f_1(x) + z)$ and initialize $f_1, f_2$ with the weights of $g_1, g_2$ respectively. Thus, if $z = \phi(g(x) - y)$ is initialized to have 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 predictions to make use of the mode information contained in $z$.

3 RELATED WORK

In recent years several works have explored video prediction, either in the context of unsupervised learning (Srivastava et al., 2015; Denton & Birodkar, 2017) or planning through prediction conditioned on actions (Oh et al., 2015; Finn et al., 2016; Kalchbrenner et al., 2016). When labels such as actions performed by an agent are available, it is possible to make different predictions about the future by conditioning on the action, effectively making multimodal predictions using additional knowledge.

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 $\ell_2$ loss. However, they also note that the generator learns to ignore the noise and produce 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 different frames in video. Although more flexible than a purely deterministic model, 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 the 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 cached 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 caching the latent variables for each sample we compute them through a learned function of the deterministic network’s residual error.

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 factors and span a diverse range of visual environments. Our experimental setup was the following: for each dataset, we trained our model to predict the following 1 or 4 frames conditioned on the previous 4 frames. We compared the performance of a deterministic feed-forward model, the Error Encoding Network, and a conditional GAN.

The deterministic model and EEN were trained using the $\ell_2$ 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. The parametric function $\phi$ mapping the prediction error to latent variables was also a multilayer convolutional network followed by two fully-connected layers. For the Atari games we used 2 latent variables, for Flappy Bird 8 and the other two tasks 32. To train our network we used the ADAM optimizer with default parameters and learning rate 0.001 or 0.0005. Full details will be provided in a code release.

The deterministic baseline model and the GAN had the same encoder-decoder architecture as the EEN. We also experimented with giving these twice as many feature maps to allow them additional capacity, which proved helpful for the baseline model in the Atari tasks.

4.1 DATASETS

We now describe the video datasets we used.

**Atari Games** We used a pretrained A2C agent\footnote{https://github.com/ikostrikov/pytorch-a2c-ppo-acktr} to generate episodes of gameplay for the Atari games Breakout and Seaquest using a standard video preprocessing pipeline, i.e. downsampling video frames to $84 \times 84$ 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\footnote{https://gym.openai.com/envs/FlappyBird-v0/} 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 frame using 4 frames as input, all of which were rescaled to $128 \times 72$ pixel color images.

**Robot Manipulation** We used the dataset of (Agrawal et al., 2016), which consists of $240 \times 240$ 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 ma-
Figure 2: Generations on Breakout. Left 4 frames are given, right 4 frames are generated. Best viewed with zoom.
Figure 3: Generations on Seaquest. Left 4 frames are given, right 4 frames are generated. Best viewed with zoom.
Manipulation does not depend on 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. Each image was rescaled to $160 \times 72$ 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 $\phi$ which takes the residual as input. By sampling different $z'$s 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 from its initial location. Additional results can be found here: [YouTube Link](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: [YouTube Link](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: [YouTube Link](https://youtu.be/HmlhdJAr4E0).

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: [YouTube Link](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 wheel. These effects are best viewed though the following video link: [YouTube 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 begin with reporting the best score across different generated samples: $\min_k \mathcal{L}(y, f(x, z_k))$. This score gives a multimodal model the chance to generate a sample from the same mode as the test sample, which should give low error. If however the model ignores latent variables or does not capture the mode that the test sample is drawn from, this will result in high loss. Note that if $\mathcal{L}$ is a valid metric in the mathematical sense (such as the $\ell_2$ or $\ell_1$ 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 $\mathcal{L}$. 

7
Figure 4: Generations on Robot Task. Left frame is given, right frame is generated.

Figure 5: Generated frames on Flappy Bird. First 4 are given, last 4 are generated.

Figure 6: Generated frames on Flappy Bird. First 4 are given, last 4 are generated.
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 $\ell_2$ loss on which 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 it 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 adversarial 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

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES


6 Appendix