Consistent Generative Query Networks

Ananya Kumar
DeepMind
London N1C4AG
ananyak@google.com

S. M. Ali Eslami
DeepMind
London N1C4AG
aeslami@google.com

Danilo J. Rezende
DeepMind
London N1C4AG
danilor@google.com

Marta Garnelo
DeepMind
London N1C4AG
garnelo@google.com

Fabio Viola
DeepMind
London N1C4AG
fviola@google.com

Edward Lockhart
DeepMind
London N1C4AG
locked@google.com

Murray Shanahan
DeepMind
London N1C4AG
mshanahan@google.com

Abstract

Stochastic video prediction is usually framed as an extrapolation problem where the goal is to sample a sequence of consecutive future image frames conditioned on a sequence of observed past frames. For the most part, algorithms for this task generate future video frames sequentially in an autoregressive fashion, which is slow and requires the input and output to be consecutive. We introduce a model that overcomes these drawbacks – it learns to generate a global latent representation from an arbitrary set of frames within a video. This representation can then be used to simultaneously and efficiently sample any number of temporally consistent frames at arbitrary time-points in the video. We apply our model to synthetic video prediction tasks and achieve results that are comparable to state-of-the-art video prediction models. In addition, we demonstrate the flexibility of our model by applying it to 3D scene reconstruction where we condition on location instead of time. To the best of our knowledge, our model is the first to provide flexible and coherent prediction on stochastic video datasets, as well as consistent 3D scene samples. Please check the project website https://bit.ly/2jX7Vyu to view scene reconstructions and videos produced by our model.

1 Introduction

The ability to fill in the gaps in high-dimensional data is a fundamental cognitive skill. Suppose you glance out of the window and see a person in uniform approaching your gate carrying a letter. You can easily imagine what will (probably) happen next. The person will walk up the path and push the letter through your door. Now suppose you glance out of the window the following day and see a person in the same uniform walking down the path, away from the house. You can easily imagine what (probably) just happened. The person came through your gate, walked up the path, and delivered a letter. Moreover, in both instances, you can visualize the scene from different viewpoints. From your vantage point at the window, you can imagine how things might look from the gate, or from the front door, or even from your neighbour's roof. Essentially, you have learned from your past experience of unfolding visual scenes how to extrapolate and interpolate in both time and space.

Preprint. Work in progress.

Replicating this ability is a significant challenge for artificial intelligence. Building on the recently developed Generative Query Network (GQN) framework (Eslami et al., 2018), we here present a neural network architecture that learns models that can flexibly extrapolate in the visual domain. Moreover, as well as models operating in time, our method can learn models that operate in space. To achieve this, we had to overcome a significant difficulty that does not arise with autoregressive models, namely the need to generate consistent sets of samples for a given temporal context even when the problem is non-deterministic. Similarly, when conditioned on camera position, our models can sample consistent sets of images for an occluded region of a scene, even when there are multiple possibilities for what that region might contain.

To make this more precise, let's first consider video prediction, a task that has been widely studied in machine learning and computer vision. At its core any video prediction task involves learning a model that can sample a set of future frames $f_{t+1}, ..., f_T$ conditioned on a sequence of previous frames $f_1, ..., f_t$. State-of-the-art models often carry out the prediction sequentially in an autoregressive manner by sampling the frame f_{t+n} from $p(f_{t+n}|f_1, ..., f_{t+n-1})$. While autoregressive models are able to generate accurate sequences of predicted frames, they are usually restricted with regards to the structure and order of their inputs – the input and output frames must be consecutive. Further, their autoregressive one-frame-at-a-time nature renders them slow to sample from.

Instead, we cast the video prediction problem as a "query" task. Generative Query Networks (GQNs) (Eslami et al., 2018) are spatial prediction models that are "queried" at test time given a set of conditioning input pairs. In its original setting, a trained GQN is given frames $f_1, ..., f_n$ from a single 3D scene together with camera positions $v_1, ..., v_n$ from which those frames were rendered. These input pairs are referred to as the *context* $C = (V, F) = ((v_1, f_1), ..., (v_n, f_n))$. The model is then asked to sample a plausible frame rendered from an arbitrary position v'. In this work, we apply this framework to temporal as well as spatial data. When we apply this framework to video prediction, the context C corresponds to the pairs of individual frames $f_1, ..., f_t$ and the timestamps $t_1, ..., t_t$ at which they occur. The query contains the timestamps of the frames that we want to sample, $t_{t+1}, ..., t_T$.

In many cases there are multiple possible future frames that are consistent with the context, making the problem stochastic. For example, a car moving towards an intersection could turn left or turn right. Given the context (timestamps and frames) of the car moving to the intersection, and a single query timestamp, GQN can sample a plausible frame at the specified time. However, unlike autoregressive models, each frame is sampled independently. GQN will sometimes sample a frame of the car on the left and sometimes sample a frame of the car on the right, but cannot capture a coherent sequence of frames where the car turns left. To address these issues, we introduce Consistent Generative Query Networks (CGQN). Given the context, CGQN samples a stochastic latent *scene representation* that models the global stochasticity in the scene. Given an arbitrary query, the model uses the sampled scene representation to render a frame corresponding to the query. The model captures correlations over multiple target frames.

To test CGQN we develop two synthetic datasets. Our first dataset consists of procedurally generated 2D shapes enacting a non-deterministic narrative of events. The goal of the prediction model is to reconstruct the narrative from a few random context frames. Quantitatively, the predictions generated by our model match those obtained by state-of-the-art video prediction algorithms, while being flexible with regards to the input and output structures as well as being able to generate several frames at once. To showcase the flexibility of our model we also apply it to consistent 3D scene reconstruction. To this end we introduce an additional dataset that consists of images of 3D scenes containing a cube with random MNIST digits engraved on each of its faces. We show quantitatively and qualitatively that CGQN outperforms GQN on this dataset, as GQN is unable to capture several correlated frames of occluded regions of the scene. We strongly encourage the reader to check the project website `https://bit.ly/2jX7Vyu` to view actual videos of experiments.

2 Related Work

Generative Query Networks (GQN) Our model builds on GQN (Eslami et al., 2018), a conditional DRAW (Gregor et al., 2015, 2016) network used for spatial prediction. The architectural changes we add to facilitate consistency include the rendering network M_γ, reconstructing multiple target frames, giving the posterior network an encoding of multiple targets instead of a single target frame, and using

a global latent to capture stochastic elements of the scene. In addition, GQN was used for (primarily deterministic) spatial prediction. We cast stochastic video prediction as a similar querying task and highlight its advantages. Concurrent work on neural processes (Garnelo et al., 2018), which extend GQN, also have architectural similarities to CGQN. Neural processes are tested on 2D function regression, 2D Thomson Sampling, and 2D contextual bandit problems. CGQN's architecture and evaluation focuses on high dimensional datasets like video prediction and scene prediction.

Video Prediction A common limitation of video prediction models is their need for determinism in the underlying environments (Lotter et al., 2017; Srivastava et al., 2015; Boots et al., 2014; Finn et al., 2016; Liu et al., 2017). Creating models that can work with stochastic environments is the motivation behind numerous recent papers: On one end of the complexity spectrum there are models like Video Pixel Networks (Kalchbrenner et al., 2017) and its variants (Reed et al., 2017b) that are powerful but computationally expensive models. These models generate a video frame-by-frame, and generate each frame pixel-by-pixel. SV2P (Babaeizadeh et al., 2018) does not model each individual pixel, but autoregressively generates a video frame-by-frame. On the other end of the spectrum, sSSM (Buesing et al., 2018) is a faster model that generates an abstract state at each time step which can be decoded into a frame when required. All these stochastic models still generate frames (or states) one time-step at a time and the input and output frames must be consecutive. By bypassing these two issues CGQN extends this spectrum of models as a flexible and even faster stochastic model. Additionally, unlike prior methods, our models can be used for a wider range of tasks – we demonstrate our approach on non-deterministic 3D scene reconstruction.

Meta-Learning Finally, our task can be framed as a few-shot density estimation problem and is thus related to some ongoing research in the area of meta learning. Meta learning is often associated with few-shot classification tasks (Vinyals et al., 2016; Koch et al., 2015) but these algorithms can be extended to few-shot density estimation for image generation (Bartunov and Vetrov, 2018). Additional approaches include models with variational memory (Bornschein et al., 2017), attention (Reed et al., 2017a) and conditional latent variables (J. Rezende et al., 2016). Crucially, while these models can sample the estimated density at random they cannot query it at specific target points and their application is limited to visually less challenging datasets like omniglot.

3 Methods

3.1 Problem Description

We consider problems where we have a collection of "scenes". Scenes could be videos, spatial scenes, or in general any key-indexed collection. A scene $S^{(i)}$ consists of a collection of viewpoint-frame (key-value) pairs $(v_1^{(i)}, f_1^{(i)}), ..., (v_n^{(i)}, f_n^{(i)})$ where $v_j^{(i)}$ refers to the indexing 'viewpoint' information and $f_j^{(i)}$ to the frame. For videos the 'viewpoints' are timestamps. For spatial scenes the 'viewpoints' are camera positions and headings. For notational simplicity, we assume that each scene has fixed length n (but this is not a requirement of the model). The viewpoint-frame pairs in each scene are generated from a data generative process D, as formulated below. Note that the viewpoint-frame pairs are typically not independent.

$$(v_1^{(i)}, f_1^{(i)}), ..., (v_n^{(i)}, f_n^{(i)}) \sim D = P((v_1, f_1), ..., (v_n, f_n)) \tag{1}$$

Each scene S is split into a context and a target. The context C contains m viewpoint-frame pairs $C = \{(v_i, f_i)\}_{i=1}^{m}$. The target T contains the remaining $n - m$ viewpoints $V = \{v_i\}_{i=m+1}^{n}$ and corresponding target frames $F = \{f_i\}_{i=m+1}^{n}$. At evaluation time, the model receives the context C and target viewpoints V and should be able to sample possible values $\hat{F}$ corresponding to the viewpoints V. In particular, the model parameterizes a (possibly implicit) conditional distribution $P_\theta(F|C, V)$, from which the frames are sampled.

Given a training set of N example scenes from data distribution D, the training objective is to find model parameters θ that maximize the log probability of the data

$$\mathop{\mathbb{E}}_{C,V,F\sim D}[\log P_\theta(F|C, V)]. \tag{2}$$

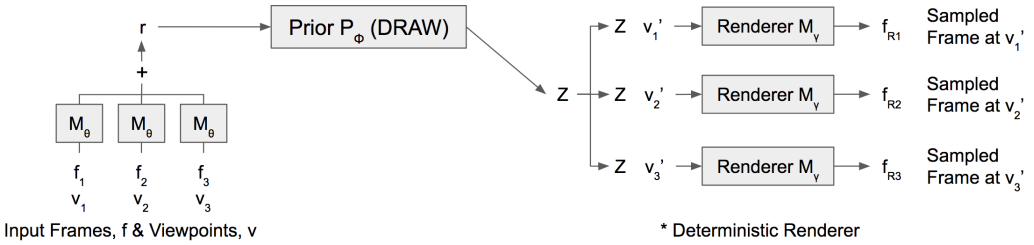


Figure 1: CGQN uses a prior P_ϕ to sample a latent z, conditioned on the input frames and viewpoints. CGQN then uses a rendering network M_γ to render z at arbitrary viewpoints e.g. v'_1, v'_2, v'_3.

3.2 Model (Generation)

We implement CGQN as a latent variable model. For each scene S, CGQN encodes the m viewpoint-frame pairs of the context C by applying a representation function M_θ to each pair independently. The resulting representations $r_1, ..., r_m$ are aggregated in a permutation-invariant way to obtain a single representation r. The latent variable z is then sampled from a prior P_ϕ that is conditioned on this aggregated representation r. The idea behind z is that it can capture global dependencies across the $n - m$ target viewpoints, which is crucial to ensure that the output frames are generated from a single consistent plausible scene. For each corresponding target viewpoint v_i, the model applies a deterministic rendering network M_γ to z and v_i to get an output frame f_i. Our model, CGQN, can thus be summarized as follows.

$$r_j = M_\theta(v_j, f_j) \text{ for all } j \in 1, ..., m$$
$$r = r_1 + ... + r_m$$
$$z \sim P_\phi(z|r)$$
$$f_i = M_\gamma(z, v_i) \text{ for all } i \in m+1, ..., n$$

In CGQN, the representation network M_θ is implemented as a convolutional network. The latent z is sampled using a convolutional DRAW (Gregor et al., 2015, 2016) prior, an LSTM-like model that recurrently samples latents over several iterations. The rendering network M_γ is implemented as an LSTM where the inputs z and v are fed in at each recurrent step. We give details of these implementations in the Appendix. Note that these building blocks are easily customizable. For example, DRAW can be replaced with a regular variational autoencoder, albeit at a cost to the visual fidelity of the generated samples.

3.3 Model (Training)

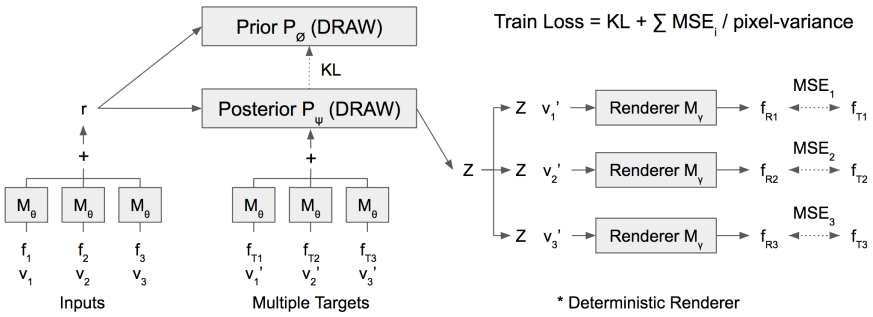


Figure 2: CGQN is trained using an approximate posterior P_ψ that has access to multiple targets.

We wish to find parameters $\theta^*, \phi^*, \gamma^*$ that maximize the log probability of the data under our model:

$$\theta^*, \phi^*, \gamma^* = \arg\max_{\theta,\phi,\gamma} f(\theta, \phi, \gamma) = \arg\max_{\theta,\phi,\gamma} \mathop{\mathbb{E}}_{C,V,F\sim D}[\log P_{\theta,\phi,\gamma}(F|C, V)]. \quad (3)$$

Since this optimization problem is intractable we introduce an approximate posterior $P_{\psi,\theta}(z|C,V,F)$. We train the model by maximizing the evidence lower bound (ELBO), as in (Rezende et al., 2014; Kingma and Welling, 2014), see derivation details in the Appendix. The resulting formulation for the ELBO of our model is as follows.

$$\mathop{\mathbb{E}}_{C,V,F\sim D}\left[\mathop{\mathbb{E}}_{z\sim P_{\psi,\theta}(z|C,V,F)}\left[\log\prod_{i=m+1}^{n}P_{\gamma}(f_i|z,v_i)\right]-\mathrm{KL}(P_{\psi,\theta}(z|C,V,F)\;||\;P_{\phi,\theta}(z|C))\right]. \quad (4)$$

Note that this expression is composed of two terms: the reconstruction probability and the KL-divergence from the the approximate posterior $P_{\psi,\theta}(z|C,V,F)$ to the conditional prior $P_{\phi,\theta}(z|C)$. We use the reparameterization trick to propagate gradients through the reconstruction probability. As we are considering Gaussian probability distributions, we compute the KL in closed form. For training, to ensure that our model's likelihood has support everywhere, we add zero-mean, fixed variance Gaussian noise to the output frame of our model. This variance (referred to as the *pixel-variance*) is annealed down during the course of training.

4 Experiments

We evaluate CGQN against a number of strong existing baselines on two tasks: a synthetic, combinatorial video prediction task and a 3D scene reconstruction task.

4.1 Video Prediction

In video prediction, given the first m frames $\{f_i\}_{i=1}^{m}$ of a video, the model's goal is to sample plausible frames that follow: $\{\hat{f}_i\}_{i=m+1}^{n}$. We encourage the reader to view actual videos of our experiments at `https://bit.ly/2jX7Vyu`.

Narratives Dataset: We present quantitative and qualitative results on a set of synthetic datasets that we call "narrative" datasets. Each dataset is parameterized by a "narrative" which describes how a collection of shapes interact. For example, in the "Traveling Salesman" narrative (Figure 3), one shape sequentially moves to (and "visits") 4 other shapes. A dataset consists of many videos which represent different instances of a single narrative. In each instance of a narrative, each shape has a randomly selected color (out of 12 colors), size (out of 4 sizes), and shape (out of 4 shapes), and is randomly positioned. While these datasets are not realistic, they are a useful testbed. In our Traveling Salesman narrative with 5 shapes, the number of distinct instances (ignoring position) is over 260 billion. With random positioning of objects, the real number of instances is higher.

Training CGQN: For CGQN, we associate frame $\hat{f}_i$ with corresponding timestamp $t_i = \frac{i}{n}$. For training, the model is given between m_1 and m_2 frames (and corresponding timestamps) randomly selected out of the first m frames, and tasked to predict k randomly selected frames. When evaluating metrics, the model is given the first m frames, and tasked to predict all n frames in the video.

GQN vs CGQN: We qualitatively show the difference between CGQN and GQN. As described in the introduction, GQN samples each frame independently. The frames sampled therefore do not form a coherent video, because each frame is from a different possible continuation. We illustrate this in Figure 3. In this Traveling Salesman narrative, an actor shape (in Figure 3, the green square) sequentially visits four other shapes. The context includes the first 6 frames, where the actor visits the first shape. The actor then visits the other three shapes in a random order. GQN, however, cannot capture a coherent video. In the GQN sample in Figure 3, the green square never visits the white pentagon. In the project website, `https://bit.ly/2jX7Vyu`, we also show examples of a narrative we call "DAG Chase", which has 4 shapes all of which move during the narrative.

Flexibility of CGQN: CGQN is more flexible than existing video prediction models. When trained with arbitrary contexts, it can take arbitrary sets of frames as input, and directly predict arbitrary sets of output frames. We illustrate this in Figure 4. In this "Color Reaction" narrative, shape 1 moves to shape 2 over frames 1 - 6, shape 2 changes color and stays that color from frames 7 - 12. Figure 4A shows the ground truth narrative. CGQN can take two frames at the start of the video, and sample frames at the end of the video (as in Figures 4B, 4C). Alternatively, CGQN can go

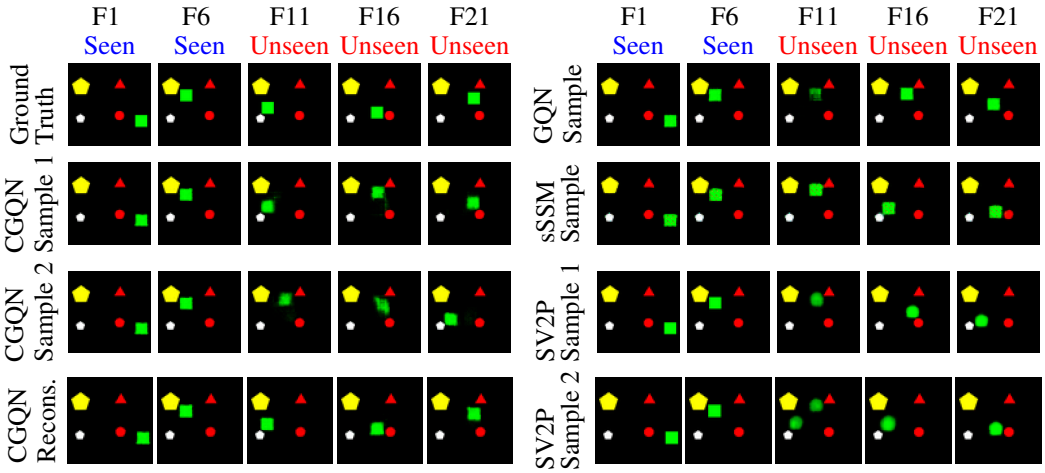


Figure 3: In the Traveling Salesman narrative, one shape (in this case the green square) sequentially moves towards and visits the other four shapes in some stochastic order. CGQN, sSSM and SV2P generally capture this, except with shape and color artifacts. GQN, on the other hand, samples each frame independently, and does not capture a coherent narrative where the green square visits all four objects (in this case the white hexagon is not visited).

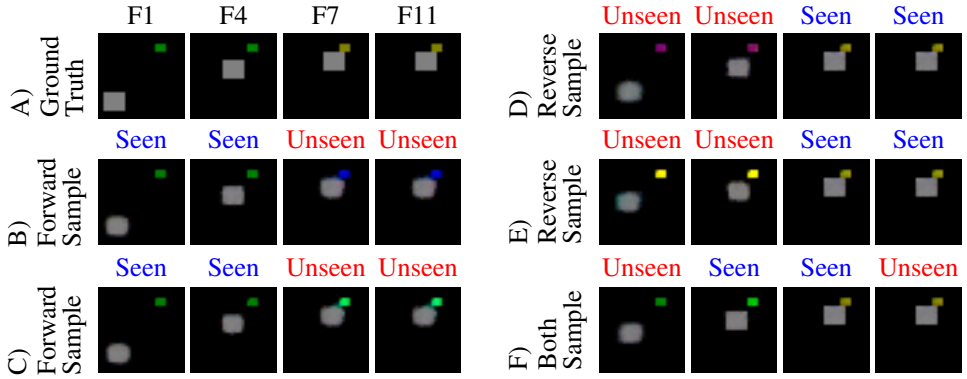
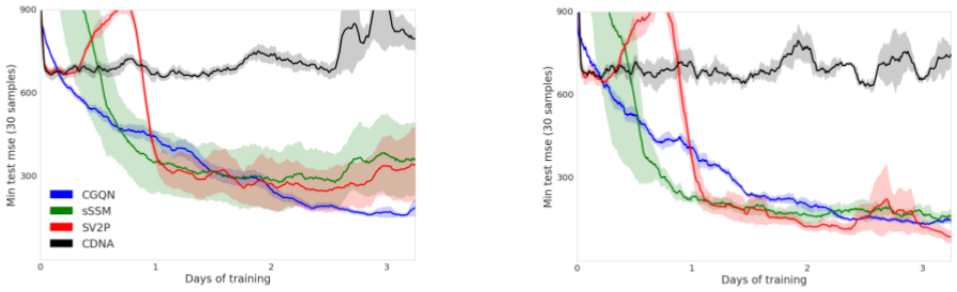


Figure 4: In the "Color Reaction" narrative, shape 1 moves to shape 2 over frames 1 - 6, shape 2 changes color and stays that color from frames 7 - 12. The ground truth is shown in sub-figure A. Our model sees 2 frames, labeled 'seen', and samples the remaining 10, of which we show 2 'unseen' samples. Our model is flexible with respect to input-output structure, and can roll a video forwards (Figures 4B, 4C), backwards (Figures 4D, 4E), or both ways (Figure 4F).

"backwards" and take two frames at the end of the video, and sample frames at the start of the video (as in Figures 4D, 4E).

Quantitative Comparisons: We quantitatively compare CGQN with sSSM (Buesing et al., 2018), SV2P (Babaeizadeh et al., 2018), and CDNA (Finn et al., 2016) on the Traveling Salesman narrative dataset, where one shape sequentially moves to (and "visits") 4 other shapes. The training set contains 98K examples, and the test set contains 1K examples. To evaluate each model, we take 30 sample continuations for each video and compute the minimum mean squared error between the samples and the original video. This measures that the model can (in a reasonable number of samples) sample the true video. A model that exhibits mode collapse would fare poorly on this metric because it would often fail to produce a sample close to the original video.

We swept over model size, learning rate, and stochasticity parameters for the models (see the Appendix for more details). We selected the best hyperparameter configuration, and ran the model with that hyperparameter configuration with 15 random seeds (10 for CDNA). We ran all models for 3 days using distributed ADAM on 4 Nvidia K80 GPUs. For sSSM, we discarded runs where the KL loss



(a) All runs of CGQN, sSSM, SV2P, CDNA.

(b) Best 30% runs of CGQN, sSSM, SV2P, CDNA.

Figure 5: Test-set multi-sample MSE loss for CGQN, sSSM, SV2P, CDNA, lower is better.

became too low or too high (these runs had very bad metric scores), and for SV2P we discarded a run which had especially poor metric scores. This discarding was done to the benefit of SV2P and sSSM – for CGQN and CDNA we used all runs. The runs for sSSM and SV2P had high variance, so we also compared the best 4/15 runs for sSSM, SV2P, CGQN and 3/10 runs for CDNA. The plots, with error bars of $\sqrt{2}$ times the standard error of the mean, are shown in Figure 5.

We draw three main conclusions from the plots.

1. Figure 5b shows that our dataset is able to distinguish between the performance of video prediction models. CDNA, a deterministic model, does much worse than the other three models. SV2P, a dedicated stochastic video prediction model, performs the best.
2. Figure 5a shows that our model converges very reliably. Averaged across all runs, our model performs significantly better than sSSM, SV2P, and CDNA.
3. Figure 5b shows that our model is competitive with modern video prediction models. Our model is best compared with sSSM. SV2P is a much slower model that generates each output frame sequentially. sSSM is a faster, state-space model that bypasses generating intermediate output frames (but still does not have the flexibility of our model).

4.2 Scene Reconstruction

Our model is also capable of consistent 3D scene reconstruction. In this setup, CGQN is provided with context frames $f_1, ..., f_m$ from a single 3D scene together with camera positions $v_1, ..., v_m$ from which those frames were rendered. The model is then asked to sample plausible frames rendered from a set of arbitrary camera positions $v_{m+1}, ..., v_n$. Often the model is asked to sample frames in an occluded region of the scene, where there are multiple possibilities for what the region might contain. Even in the presence of this uncertainty, CGQN is able to sample consistent frames that form a coherent scene. We encourage the reader to view videos visualizations of our experiments at `https://bit.ly/2jX7Vyu`.

MNIST Dice Dataset To demonstrate this, we develop a 3D dataset where each scene consists of a cube in a room. Each face of the cube has a random MNIST digit (out of 100 digits) engraved on it. In each scene, the cube is randomly positioned and oriented, the color and textures of the walls are randomly selected, and the lighting source (which creates shadows) is randomly positioned. The context frames show at most three sides of the dice, but the model may be asked to sample camera snapshots that involve the unseen fourth side. We show quantitatively and qualitatively that CGQN performs better than GQN on this dataset, because GQN is unable to capture a coherent scene.

GQN vs CGQN (Qualitative) GQN samples each frame independently, and does not sample a coherent scene. We illustrate this in Figure 7, where we show an example scene from our test-set. The context frames (blue cones) see three sides of the cube, but the model is queried (red cones) to sample the occluded fourth side of the cube. Figure 7 also shows the samples for CGQN and GQN. GQN (right column) independently samples a 7 and then a 0 on the unseen side of the dice. CGQN samples a coherent scene, where the sampled unseen digit is consistently rendered across different viewpoints. CGQN's reconstruction accurately captures the ground truth digit, which shows that the model is capable of sampling the target. Note that all frames are captured from a circle with fixed

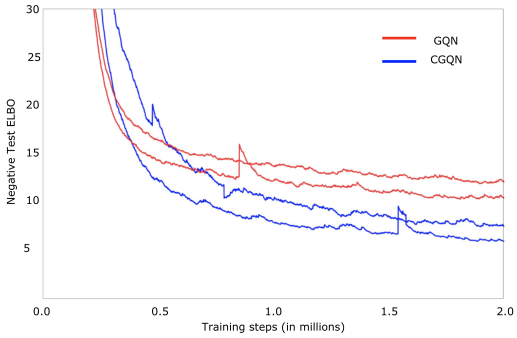


Figure 6: Test-set negative ELBO against number of training steps (lower is better).

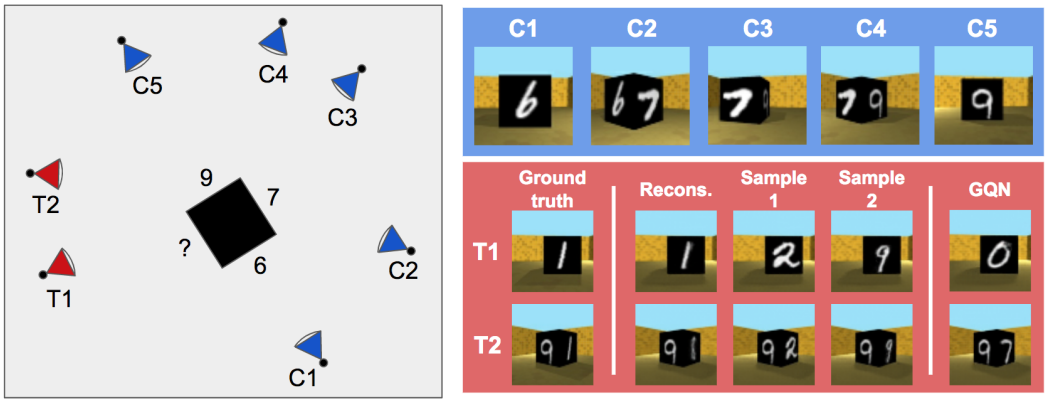


Figure 7: A cube in a room, with MNIST digits engraved on each face (test-set scene). The blue cones are where the context frames were captured from. The red cones are where the model is queried. The context frames see three sides of the cube, but the models are tasked to sample from the fourth, unseen, side. GQN (right column) independently samples a 0 and 7 for the unseen side, resulting in an inconsistent scene. CGQN samples a consistent digit (2 or 9) for the unseen cube face.

radius, with the camera facing the center of the room. However, the frames are not equally spaced, which distinguishes this from video prediction tasks.

GQN vs CGQN (Quantitative) We can compare GQN and CGQN by analyzing the test-set negative ELBO (as a proxy for the test-set negative log likelihood) over multiple target frames, each showing different viewpoints of the same unseen face of the cube. This serves as a quantitative measure for the quality of the models' scene reconstruction. To motivate why CGQN should do better, imagine that we have a perfectly trained GQN and CGQN, which captures all the nuances of the scene. Since there are 100 possible digits engraved on the unseen side, there is a 1/100 chance that each sampled frame captures the ground truth digit on the unseen face. GQN samples the unseen digit independently for each viewpoint, so the probability that a set of three frames all capture the ground truth digit is 1/1000000. On the other hand, CGQN captures the correlations between frames. If the digit is correct in one of three frames, it should be correct in the other two frames. So the probability that a set of three frames all capture the ground truth digit is 1/100. In other words, a perfectly trained consistent model will have better log likelihoods than a perfectly trained factored model.

In practice, the benefits of consistency may trade off with accuracy of rendering the scene. For example, the more consistent model could produce lower quality images. So it is important to compare the models' performance by comparing the test-set ELBOs. Figure 6 compares the test-set ELBOs for CGQN and GQN. We ran 4 runs for each model, and picked the best 2/4 runs for each model. The results suggest that CGQN does achieve quantitatively better scene reconstruction than GQN. We repeated this experiment twice more, with different 'pixel-variance' values and obtained similar plots, as shown in the Appendix.

CGQN Consistency Analysis We also analyze the consistency of CGQN. We measure the KL divergence from the posterior to the prior network in a trained CGQN model. We give CGQN a context comprising of frames that show three sides of the cube. We condition the posterior on one additional target frame that shows the unseen side of the cube, and compute the KL divergence from the posterior to the prior, KL_1. Alternatively, we condition the posterior on three additional target frames that show the unseen side of the dice, and compute the KL divergence from the posterior to the prior, KL_3. The 2 extra target frames added for KL_3 do not add any information, so $\mathrm{KL}_3 \approx \mathrm{KL}_1$ for a consistent model. On the other hand, for a factored model like GQN, $\mathrm{KL}_3 = 3\mathrm{KL}_1$. We trained 12 CGQN models, and the mean KL_3 was 4.25, the mean KL_1 was 4.19, and the standard deviation of $\mathrm{KL}_3 - \mathrm{KL}_1$ was 0.092. This suggests that CGQN is consistent, in the intended sense.

5 Conclusion

We have presented an architecture for learning generative models in the visual domain that can be conditioned on arbitrary points in time or space. Our models can extrapolate forwards or backwards in time, without needing to generate intermediate frames. Moreover, given a small set of contextual frames they can be used to render 3D scenes from arbitrary camera positions. In both cases, they generate consistent sets of frames for a given context, even in the presence of stochasticity. One limitation of our method is that the stochastic latent representation is of a fixed size, which may limit its expressivity in more complicated applications – fixing this limitation and testing on more complex datasets are good avenues for future work. Among other applications, video prediction can be used to improve the performance of reinforcement learning agents on tasks that require lookahead (Racanière et al., 2017; Buesing et al., 2018). In this context, the ability to perform "jumpy" predictions that look many frames ahead in one go is an important step towards agents that can explore a search space of possible futures, as it effectively divides time into discrete periods. This is an avenue we will be exploring in future work.

6 Acknowledgements

We would like to thank Dumitru Erhan and Mohammad Babaeizadeh for the code for SV2P and helping us in getting SV2P working on our datasets, and Lars Buesing, Yee Whye Teh, Antonia Creswell, Chongli Qin, Jonathan Uesato, and Valentin Dalibard for their very useful feedback in preparing this manuscript.

References

Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R. H., and Levine, S. (2018). Stochastic variational video prediction. In *International Conference on Learning Representations*.

Bartunov, S. and Vetrov, D. (2018). Few-shot generative modelling with generative matching networks. In *International Conference on Artificial Intelligence and Statistics*, pages 670–678.

Boots, B., Byravan, A., and Fox, D. (2014). Learning predictive models of a depth camera amp; manipulator from raw execution traces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4021–4028.

Bornschein, J., Mnih, A., Zoran, D., and J. Rezende, D. (2017). Variational memory addressing in generative models. In *Advances in Neural Information Processing Systems*, pages 3923–3932.

Buesing, L., Weber, T., Racanière, S., Eslami, S. M. A., Rezende, D. J., Reichert, D. P., Viola, F., Besse, F., Gregor, K., Hassabis, D., and Wierstra, D. (2018). Learning and querying fast generative models for reinforcement learning. *CoRR*, abs/1802.03006.

Eslami, S. M. A., Jimenez Rezende, D., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., Reichert, D. P., Buesing, L., Weber, T., Vinyals, O., Rosenbaum, D., Rabinowitz, N., King, H., Hillier, C., Botvinick, M., Wierstra, D., Kavukcuoglu, K., and Hassabis, D. (2018). Neural scene representation and rendering. *Science*, 360(6394):1204–1210.

Finn, C., Goodfellow, I. J., and Levine, S. (2016). Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 64–72.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., and Teh, Y. W. (2018). Neural processes. In *Theoretical Foundations and Applications of Deep Generative Models Workshop, ICML*.

Gregor, K., Besse, F., Jimenez Rezende, D., Danihelka, I., and Wierstra, D. (2016). Towards conceptual compression. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3549–3557. Curran Associates, Inc.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471, Lille, France. PMLR.

J. Rezende, D., Danihelka, I., Gregor, K., Wierstra, D., et al. (2016). One-shot generalization in deep generative models. In *International Conference on Machine Learning*, pages 1521–1529.

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2017). Video pixel networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1771–1779.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Liu, Z., Yeh, R. A., Tang, X., Liu, Y., and Agarwala, A. (2017). Video frame synthesis using deep voxel flow. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4473–4481.

Lotter, W., Kreiman, G., and Cox, D. D. (2017). Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations*.

Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Jimenez Rezende, D., Puigdomènech Badia, A., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., and Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems 30*, pages 5690–5701.

Reed, S. E., Chen, Y., Paine, T., van den Oord, A., Eslami, S. M. A., Rezende, D. J., Vinyals, O., and de Freitas, N. (2017a). Few-shot autoregressive density estimation: Towards learning to learn distributions. *CoRR*, abs/1710.10304.

Reed, S. E., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Chen, Y., Belov, D., and de Freitas, N. (2017b). Parallel multiscale autoregressive density estimation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2912–2921.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1278–1286. PMLR.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 843–852. JMLR.org.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

A Appendix A: ELBO Derivation

The model is trained by maximizing an evidence lower bound, as in variational auto-encoders. We begin by expressing the log probability of the data in terms of the model's latent variable z.

$$\mathop{\mathbb{E}}_{C,V,F\sim D}[\log P(F|C,V)] = \mathop{\mathbb{E}}_{C,V,F\sim D}\left[\log \mathop{\mathbb{E}}_{z\sim P_{\phi,\theta}(z|C)}\left[P_\gamma(F|z,V)\right]\right]$$

The derivative of this objective is intractable because of the $\log$ outside the expectation. Using Jensen's inequality and substituting the $\log$ and expectation leads to an unbiased estimator that collapses to the mean of the distribution. Instead, we use the standard trick of parameterizing an approximate posterior distribution $P_{\psi,\theta}$. Instead of sampling z from the prior $P_{\phi,\theta}(z|C)$ we sample z from the approximate posterior $P_{\psi,\theta}(z|C,V,F)$ to get an equivalent objective.

$$f(\theta,\phi,\gamma) = f(\theta,\phi,\gamma,\psi) = \mathop{\mathbb{E}}_{C,V,F\sim D}\left[\log \mathop{\mathbb{E}}_{z\sim P_{\psi,\theta}(z|C,V,F)}\left[\frac{P_{\phi,\theta}(z|C)}{P_{\psi,\theta}(z|C,V,F)}P_\gamma(F|z,V)\right]\right]$$

Note that for efficiency, we often sample a subset of the target viewpoint-frame pairs (V,F) instead of conditioning the posterior and training on the entire set. We now apply Jensen's inequality to get a lower bound (ELBO) that we maximize as a surrogate.

$$g(\theta,\phi,\gamma,\psi) = \mathop{\mathbb{E}}_{C,V,F\sim D}\left[\mathop{\mathbb{E}}_{z\sim P_{\psi,\theta}(z|C,V,F)}\left[\log\left(\frac{P_{\phi,\theta}(z|C)}{P_{\psi,\theta}(z|C,V,F)}P_\gamma(F|z,V)\right)\right]\right]$$
$$g(\theta,\phi,\gamma,\psi) \le f(\theta,\phi,\gamma,\psi)$$

We can split the ELBO into 2 terms, the reconstruction probability and the KL-divergence between the prior and posterior.

$$\text{RP}(\theta,\gamma,\psi) = \mathop{\mathbb{E}}_{C,V,F\sim D}\left[\mathop{\mathbb{E}}_{z\sim P_{\psi,\theta}(z|C,V,F)}\left[\log P_\gamma(F|z,V)\right]\right]$$

$$\text{KL}(\theta,\phi,\psi) = \mathop{\mathbb{E}}_{C,V,F\sim D}\left[\text{KL}(P_{\psi,\theta}(z|C,V,F)\ ||\ P_{\phi,\theta}(z|C))\right]$$
$$g(\theta,\phi,\gamma,\psi) = \text{RP}(\theta,\gamma,\psi) - \text{KL}(\theta,\phi,\psi)$$

Since we consider Gaussian probability distributions, the KL can be computed in closed form. For the reconstruction probability, we note that each of the $n-m$ target frames f_i are generated independently conditional on z and the corresponding viewpoint v_i.

$$\text{RP}(\theta,\gamma,\psi) = \mathop{\mathbb{E}}_{C,V,F\sim D}\left[\mathop{\mathbb{E}}_{z\sim P_{\psi,\theta}(z|C,V,F)}\left[\log \prod_{i=m+1}^{n} P_\gamma(f_i|z,v_i)\right]\right]$$

We can then apply the standard reparameterization trick (where we sample from a unit Gaussian and scale the samples accordingly). This gives us a differentiable objective where we can compute derivatives via backpropagation and update the parameters with stochastic gradient descent.

B Appendix B: Model Details and Hyperparameters

We first explain some of the hyper-parameters in our model. For reproducibility, we then give the hyper-parameter values that we used for the narrative concepts task and the 3D scene reconstruction task.

For CGQN, recall that we added Gaussian noise to the output of the renderer to ensure that the likelihood has support everywhere. We call the variance of this Gaussian distribution the "pixel-variance". When the pixel-variance is very high, the ELBO loss depends a lot more on the KL-divergence between the prior and the posterior, than the mean squared error between the target and

predicted images. That is, a small change δ in the KL term causes a much larger change in the ELBO than a small change δ in the mean squared error. As such, the training objective forces the posterior to match the prior, in order to keep the KL low. This makes the model predictions deterministic. On the other hand, when the pixel-variance is near zero, the ELBO loss depends a lot more on the mean squared error between the target and predicted images. In this case, the model allows the posterior to deviate far from the prior, in order to minimize the mean squared error. This leads to good reconstructions, but poor samples since the prior does not overlap well with the (possible) posteriors.

As such, we need to find a good "pixel-variance" that is neither too high, nor too low. In our case, we linearly anneal the pixel-variance from a value α to β over 100,000 training steps. Note that the other models, sSSM and SV2P, have an equivalent hyper-parameter, where the KL divergence is multiplied by a value β. SV2P also performs an annealing-like strategy (Babaeizadeh et al., 2018).

For the traveling salesman dataset, we used the following parameters for the DRAW conditional prior/posterior net (Gregor et al., 2015, 2016). The rendering network was identical, except we do not have a conditional posterior, making it deterministic.

Name	Value	Description
nt	4	The number of DRAW steps in the network.
stride_to_hidden	$[2, 2]$	The kernel and stride size of the conv. layer mapping the input image to the LSTM input.
nf_to_hidden	64	The number of channels in the LSTM layer.
nf_enc	128	The number of channels in the conv. layer mapping the input image to the LSTM input.
stride_to_obs	$[2, 2]$	The kernel and stride size of the transposed conv. layer mapping the LSTM state to the canvas.
nf_to_obs	128	The number of channels in the hidden layer between LSTM states and the canvas
nf_dec	64	The number of channels of the conv. layer mapping the canvas state to the LSTM input.
nf_z	3	The number of channels in the stochastic latent in each DRAW step.
α	2.0	The initial pixel-variance.
β	0.5	The final pixel-variance.

For the encoder network, M_θ, we apply a convolutional net to each image separately. The convolution net has 4 layers, with a ReLU non-linearity between each layer (but not after the last layer). The first layer has 8 channels, kernel shape of 2x2, and stride lengths of 2x2. The second layer has 16 channels, kernel shape of 2x2, and stride lengths of 2x2. The third layer has 32 channels, kernel shape of 3x3, and stride length of 1x1. The final layer has 32 channels, kernel shape of 3x3, and stride length of 1x1.

For all other datasets, we use the same encoder network, and similar hyper-parameters. For the MNIST Cube 3D scene reconstruction task, the main differences are that we use nt: 6, nf_to_hidden: 128, nf_dec: 128. We also had to use a slightly different annealing strategy for the pixel-variance. Simply annealing the variance down led to the KL-values collapsing to 0, and never rising back up. In other words, the predictions became deterministic. We use an annealing strategy somewhat similar to (Babaeizadeh et al., 2018). We keep the pixel-variance at 2.0 for the first 100,000 iterations, then keep it at 0.2 for 50,000 iterations, then keep it at 0.4 for 50,000 iterations, and then finally leave it at 0.9 until the end of training. The intuition is to keep the KL high first so that the model can make good deterministic predictions. Then, we reduce the pixel-variance to a low value (0.2) so that the model can capture the stochasticity in the dataset. Finally, we increase the pixel-variance so that the prior and the posteriors are reasonably similar.

Note that for each stochastic video prediction model we tested (CDNA, SV2P, sSSM), we swept over hyper-parameters, doing a grid search. We swept over size parameters, the learning rate, and the

parameter used to control the KL divergence between the conditional posterior and the conditional prior. We ensured that we had tested hyper-parameter values slightly above, and below, the ones that we found worked best.

C Appendix C: Additional Consistent ELBO Experiments

We compare the negative ELBO over 3 target frames for CGQN and GQN for different pixel-variance values. The pixel-variance is effectively a hyper-parameter, that we can tune based on visual inspection of the samples, or an alternative metric. Good pixel-variance values for GQN and CGQN are different. As such, we included the pixel-variance values we found to work well for GQN, and the values we found to work well for CGQN. We tried pixel-variance values of $v\sqrt{T}$, where $T = 3$ is the number of targets and we tried $v = 0.4, 0.5, 0.6$. The plot for $v = 0.5$ is shown in the main paper. We show the best half (two out of four) runs for GQN and CGQN for $v = 0.4$, $v = 0.5$, and $v = 0.6$ below. Note that the ELBO for CGQN is better than the ELBO for GQN in all the plots below. If CGQN and GQN are equally good models (null hypothesis), the probability than CGQN did better than GQN in all these 6 curves is upper bounded by $1/2^6 \approx 1.6\%$. So we can reject the null hypothesis with high statistical significance.

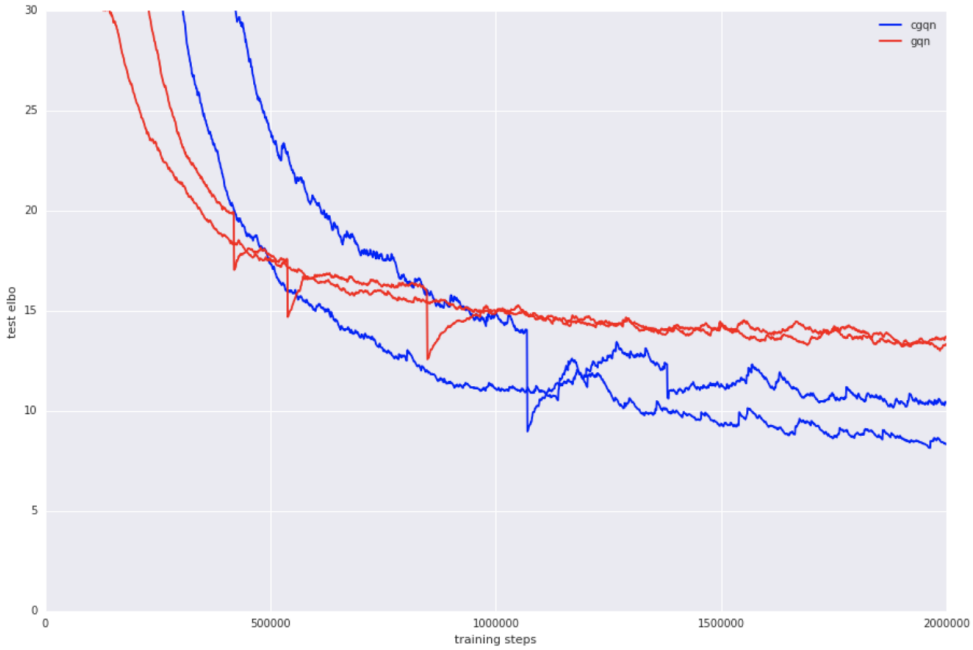


Figure 8: Test-set negative ELBO against number of training steps (lower is better). CGQN in blue, GQN in red. $v = 0.4$ in this case.

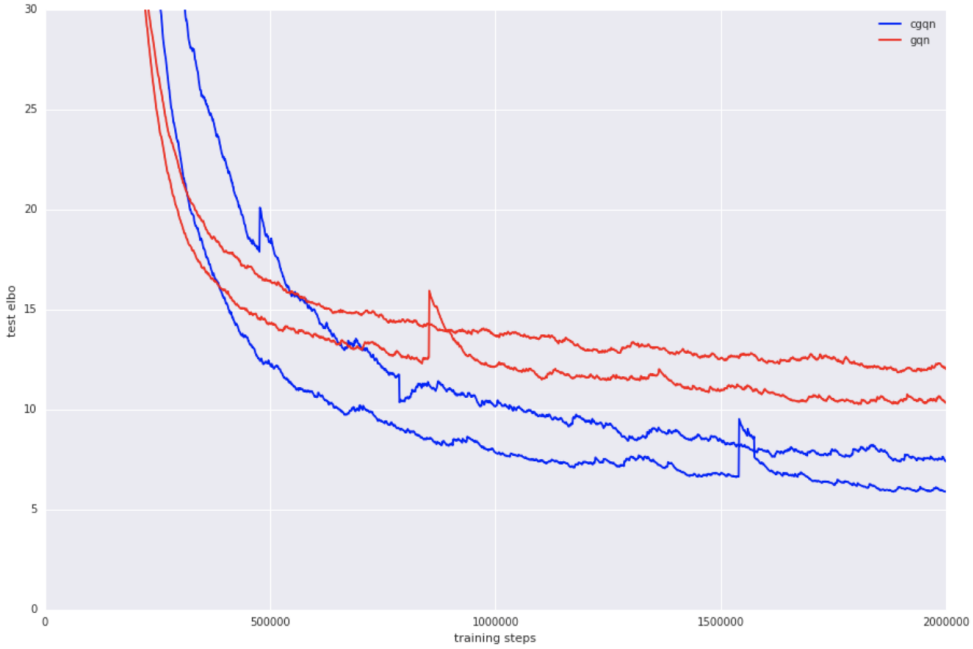


Figure 9: Test-set negative ELBO against number of training steps (lower is better). CGQN in blue, GQN in red. $v = 0.5$ in this case.

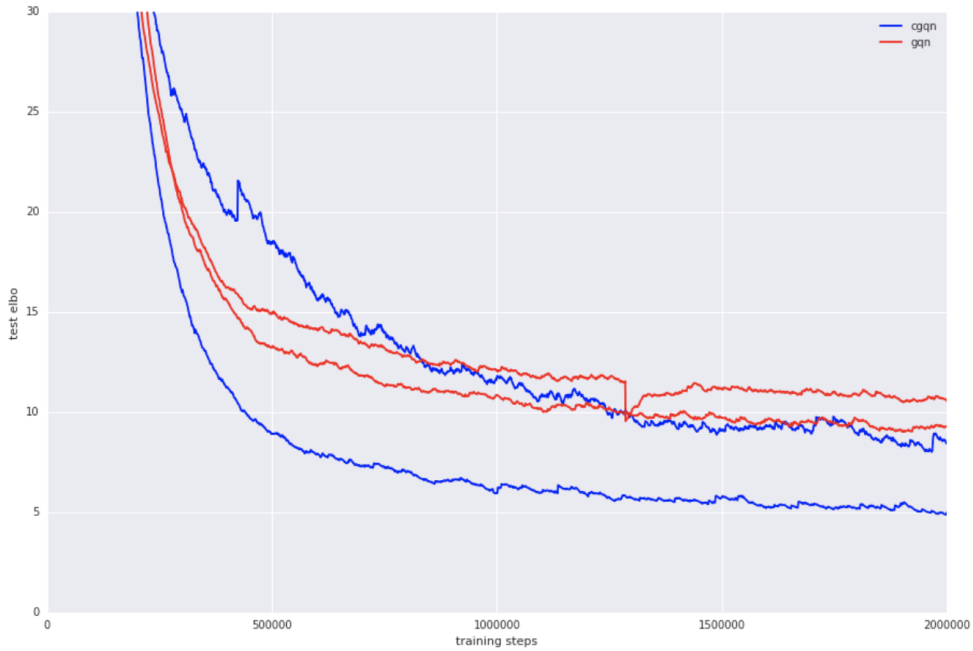


Figure 10: Test-set negative ELBO against number of training steps (lower is better). CGQN in blue, GQN in red. $v = 0.6$ in this case.