Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

Probabilistic Programming for Scientific Discovery

Lecture 3

Ludger Paehler

Lviv Data Science Summer School

July 29, 2020

Table of Contents

Outline

Bayesian Deep Learning

Type 2 Systems: Marrying Deep Learning with Probabilistic Programming
- Generative Adversarial Networks
- Variational Autoencoder

Bayesian Deep Learning

Core-Purpose

- Model posterior distributions over the weights of the neural networks
- Better predictions and well-calibrated uncertainties
  - Recommender systems
  - Security-critical applications
  - ...

Bayesian Deep Learning

Theory

In the Jupyter Notebook we will now sketch the Bayesian neural networks in probabilistic programming systems.

Bayesian Deep Learning

Challenges

Bayesian inference extremey challenging for deep networks

- Intractable posterior
  - Could switch to approximate inference, which has less of a problem there
- Millions of parameters
- Large datasets
- Unclear which priors to use

Outline

Bayesian Deep Learning

Type 2 Systems: Marrying Deep Learning with Probabilistic Programming

- Generative Adversarial Networks
- Variational Autoencoder

What is type 2?

Following the definition of Yoshua Bengio

System 1

- Intuitive
- Fast
- Unconscious
- Non-linguistic
- Habitual
- Current DL

System 2

- Slow
- Logical
- Sequential
- Conscious
- Linguistic
- Algorithmic
- Planning
- Reasoning
- Future DL

What is type 2?

Key here is the move to manipulations of **high-level/semantic concepts**, which can be recombined **combinatorially**

What is type 2?

Key targets here are:

- Out-of-distribution generalization
- Out-of-distribution transfer
- Compositionality
- Causality
- Knowledge-seeking
- Better world model

What is type 2?

Key ingredients of these systems are

- Attention [1]
- Sparse factor graphs
- Meta-learning [2]
- Causal discovery [3]
- Compositional architectures
- Probabilistic programming

[1] https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
[2] https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html
[3] https://www.inference.vc/untitled/

Including Generative Models

Generative models are a prime example of one of those composable building blocks, which we seek to include in out probabilistic programming systems

- Can be used to construct oracles or surrogates
- GANs
  - Overview of theory
  - Sketch of a useful embedding in a probabilistic programming system
- Variational Autoencoders
  - Overview of theory
  - Sketch of a useful embedding in a probabilistic programming system

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

GANs for Probabilistic Programming

Generative Models in General

Learn a model of the **true** underlying data distribution from samples.

GANs for Probabilistic Programming

Overview of Theory

- Generative adversarial networks (GANs) are an instance of an implicit model, which hence aims to learn an implicit distribution from data
- The model is then learned in a two-player min-max game between the generator and discriminator
- Generator
  - Aims to generate as realistic data as possible to trick the discriminator
- Discriminator
  - Distinguishes between real and generated data
- Essentially defines a bi-level optimization, where we would ideally like to find the Nash equilibrium

GANs for Probabilistic Programming

Overview of Theory

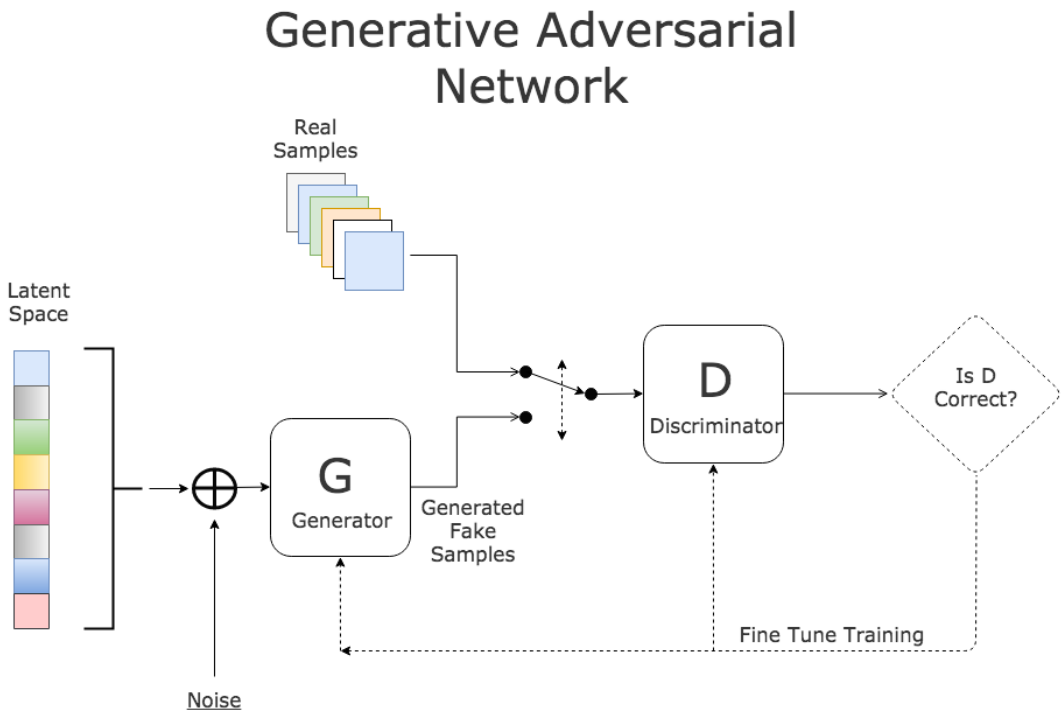


Figure: Generative Adversarial Network architecture. Source: kndnuggets

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

GANs for Probabilistic Programming

Overview of Theory

- Objective of original GAN [4] given by

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right]$$

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

[4] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

GANs for Probabilistic Programming

Overview of Theory

- Objective of the generative model can be described as divergence minimization, where the distance between the true distribution and the generated, internal, distribution is to be minimized
  - GANs do not do divergence minimization in practice, but instead learn a distance between the data and the model distribution, which can then provide the learning signal to the model
- If the discriminator is optimal it is minimizing the Jensen Shannon divergence between the true and generated distributions
- But the discriminator is not optimal and lacks the computational resources and access to the true data distribution
- For training to be successful the support of the distribution needs to overlap
- There exist many other types of divergences in literature [5]

[5] Interpretable comparison of distributions and models. Gretton et al.

GANs for Probabilistic Programming

Overview of Theory

Wasserstein [6]

- Wasserstein distance

$$W(p, p^\star) = \sup_{||f||_L \leq 1} \mathbb{E}_{p(x)} f(x) - \mathbb{E}_{p^\star(x)} f(x)$$

- Wasserstein GAN

$$\min_G \max_{||D||_L \leq 1} \mathbb{E}_{p^\star(x)} D(x) - \mathbb{E}_{p(z)} D(G(z))$$

- Becomes computationally intractable for complex cases

[6] Arjovsky, M., Chintala, S. and Bottou, L., 2017. Wasserstein gan. arXiv preprint arXiv:1701.07875.

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

GANs for Probabilistic Programming

Overview of Theory

Moment-matching distributions [7]

- MMD

$$MMD(p^\star, p) = \sup_{||f||_{\mathcal{H}} \leq 1} \mathbb{E}_{p^\star} f(x) - \mathbb{E}_{p(x)} f(x)$$

- MMD-GAN

$$\min_G \max_{||D||_{\mathcal{H}} \leq 1} \mathbb{E}_{p^\star(x)} D(x) - \mathbb{E}_{p(z)} D(G(z))$$

[7] Li, C.L., Chang, W.C., Cheng, Y., Yang, Y. and Póczos, B., 2017. MMD GAN Towards deeper understanding of moment matching network. In Advances in Neural Information Processing Systems (pp. 2203-2213).

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

GANs for Probabilistic Programming

Overview of Theory

The family of f-divergences [8]

- f-divergences

$$D_f(p^\star||p) = \int p(x) f\left(\frac{p^\star(x)}{p(x)}\right) dx$$

- Which also directly define a variational lower bound

$$\int p(x) f\left(\frac{p^\star(x)}{p(x)}\right) dx \geq \sup_{T \in \mathcal{T}} \left(\mathbb{E}_{p(x)} T(x) - \mathbb{E}_{p^\star(x)} f^\star(T(x))\right)$$

- f-GAN

$$\min_G \max_D \mathbb{E}_{p(z)} D(G(z)) - \mathbb{E}_{p^\star(x)} f^\star(D(x))$$

[8] Nowozin, S., Cseke, B. and Tomioka, R., 2016. f-gan: Training generative neural samplers using variational divergence minimization. In Advances in neural information processing systems (pp. 271-279).

GANs for Probabilistic Programming

Overview of Theory

So why should we not just do divergence minimization?

GAN

- \+ good sample
- \+ learned loss function
- \- hard to analyze
- \- no optimal convergence guarantees

Divergence minimization

- \+ optimal convergence guarantees
- \+ easy to analyzable loss properties
- \- hard to get good samples
- \- loss functions don't correlate with human evaluation

GANs for Probabilistic Programming

Overview of Theory

Also have to consider we want to use unconditional, or conditional GANs, where conditional GANs have a much clearer interleaving with the application of choice

- Unconditional
  - Generates samples from a noise vector with no user-control
- Conditional
  - Are able to add additional information to specfy which types of samples we are interested in
- Conditional GANs have some a lot of promise in science [9]

There exists no single evaluation metric, which would be able to distinguish between all desired properties [10]

[9] Luke de Oliveira, M.P. and Nachman, B., 2017, December. Tips and tricks for training gans with physics constraints. In Workshop at the 31st Conference on Neural Information Processing Systems (NIPS), Deep Learning for Physical Sciences.(December 2017).

[10] Theis, L., Oord, A.V.D. and Bethge, M., 2015. A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844.

GANs for Probabilistic Programming

Conditional GANs [11]

- Generalization to the conditional setting, where we have access to some additional information for each data point
- I.e. random noise + task-relevant information input

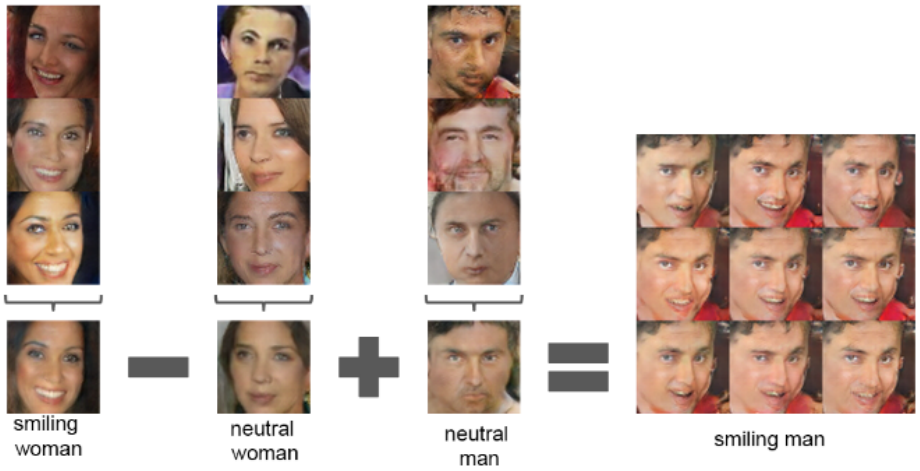
	User tags + annotations	Generated tags
	montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car	taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails
	food, raspberry, delicious, homemade	chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes
	water, river	creek, lake, along, near, river, rocky, treeline, valley, woods, waters
	people, portrait, female, baby, indoor	love, people, posing, girl, young, strangers, pretty, women, happy, life

[11] Mirza, M. and Osindero, S., 2014. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

GANs for Probabilistic Programming

Deep Convolutional GANs [12]

- Uses deep convnets as generator and discriminator
- Produces semantically reasonable images
- Latent space of the generator develops meaningful semantics

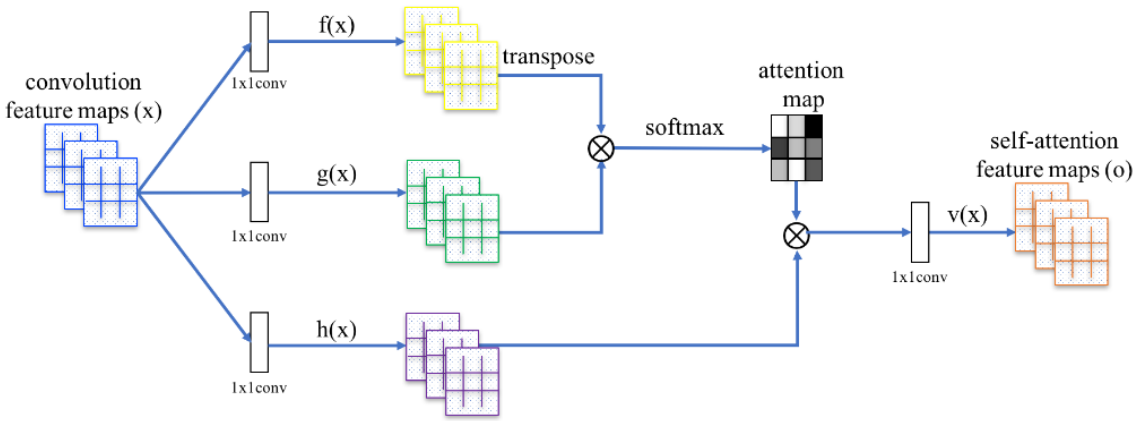


[12] Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

GANs for Probabilistic Programming

Self-Attention GANs [13]

- Self-attention equips the images with global structure and coherence
- First introduction of attention mechanisms into the GAN domain



[13] Zhang, H., Goodfellow, I., Metaxas, D. and Odena, A., 2019, May. Self-attention generative adversarial networks. In International Conference on Machine Learning (pp. 7354-7363).

GANs for Probabilistic Programming

LOGAN [14]

- Utilize latent optimization with gradients to improve the dynamics of the game between the generator and discriminator
- Improves must in terms of fidelity and variety

Figure: BigGAN vs LOGAN.

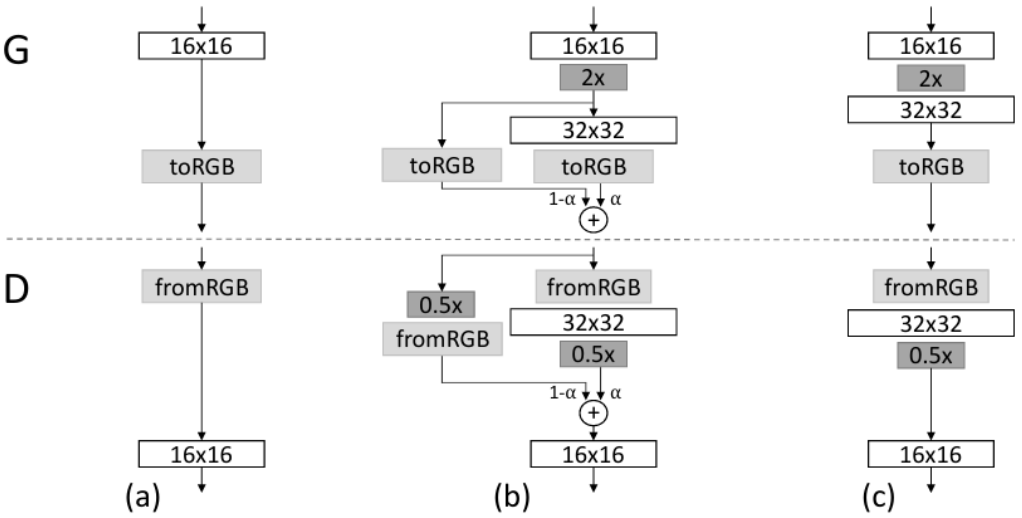
[14] Wu, Y., Donahue, J., Balduzzi, D., Simonyan, K. and Lillicrap, T., 2019. Logan: Latent optimisation for generative adversarial networks. arXiv preprint arXiv:1912.00953.

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

GANs for Probabilistic Programming

Progressive GANs [15]

- Starts with a small image, which then trained until convergence
- Once it is converged add a new layer in 4 times the resolution
- Train until convergence
- Repeat the process until you arrive at your desired resolution

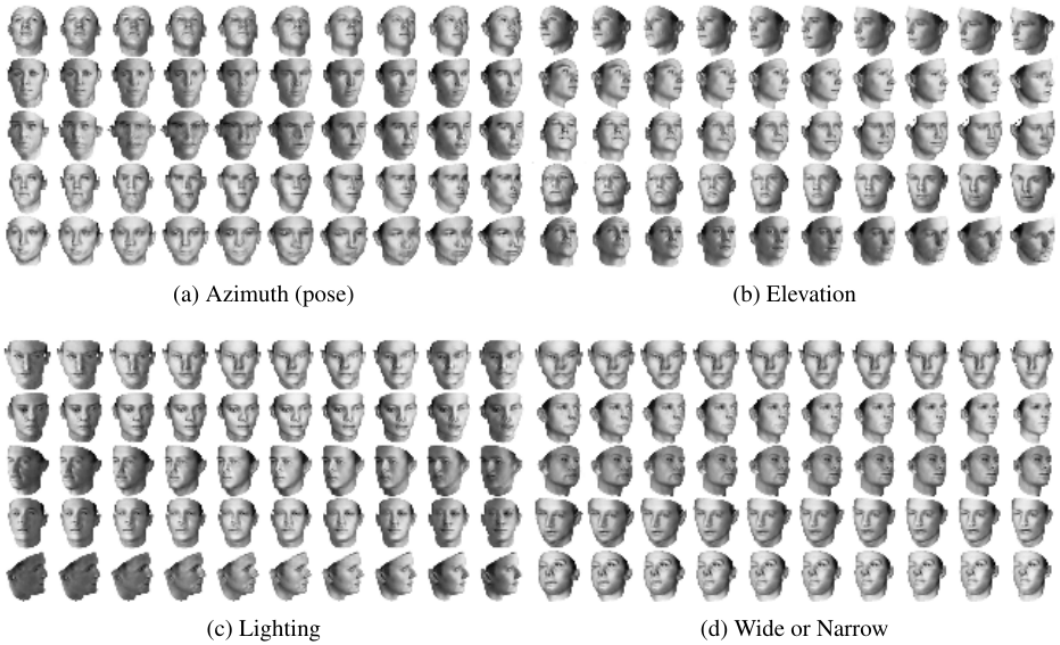


[15] Karras, T., Aila, T., Laine, S. and Lehtinen, J., 2017. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196.

GANs for Probabilistic Programming

InfoGANs [16]

- Information maximizing GAN
- Introduces inference network to solve the inverse problem of latent code z given generator output $G(z)$
- Learns to associate discrete latent variable with each category



[16] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. and Abbeel, P., 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Advances in neural information processing systems (pp. 2172-2180).

GANs for Probabilistic Programming

Pix2Pix [17]

- Trains the generator to construct a mapping between two different image domains
- Only needs to complement the plain-vanilla GAN objective with the reconstruction error

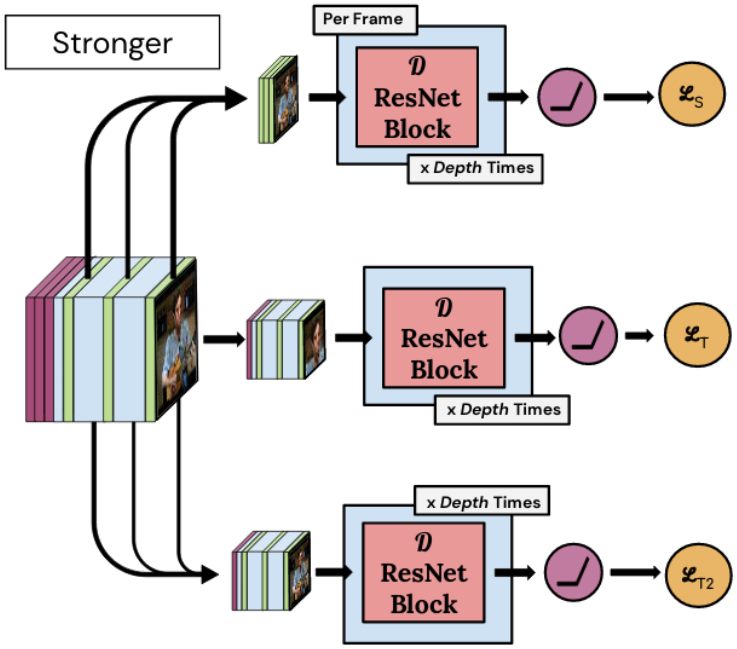
[17] Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A., 2017. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

GANs for Probabilistic Programming

TriVD-GAN [18]

- Systematically deconstruct the discriminator to derive insights into performance-relevant features and possible improvement
- Introduce a new recurrent unit inside of the video prediction algorithm to make algorithm much more efficient and achieve new benchmark results

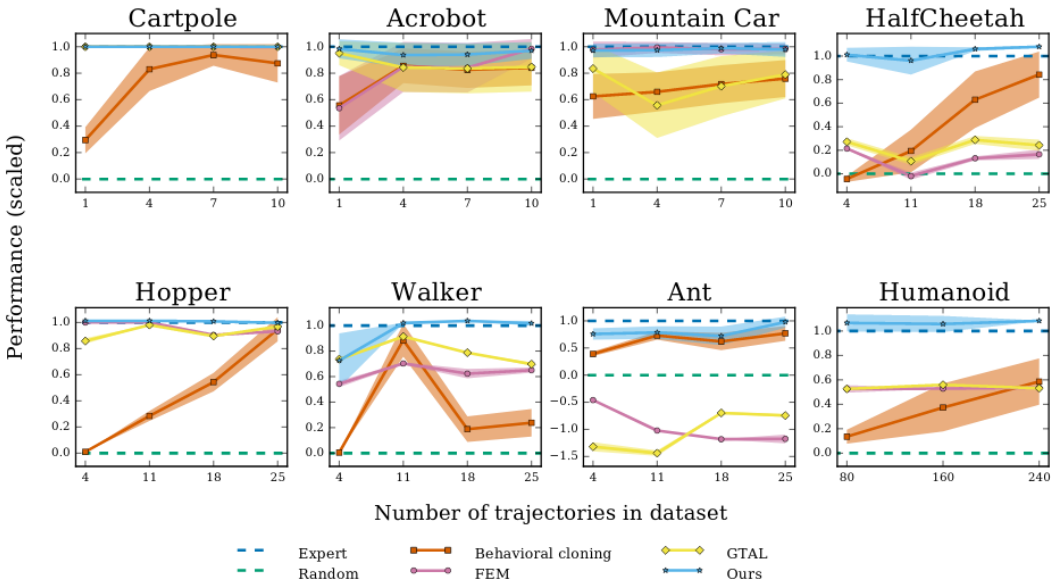


[18] Luc, P., Clark, A., Dieleman, S., Casas, D.D.L., Doron, Y., Cassirer, A. and Simonyan, K., 2020. Transformation-based adversarial video prediction on large-scale data. arXiv preprint arXiv:2003.04035.

GANs for Probabilistic Programming

GAIL [19]

- Applies GANs to inverse reinforcement learning to obtain a model-free imitation algorithm
- The target distributions in this case are the distributions of state and actions
- Method not very sample-efficient
  - Behavioural cloning for policy parameter initialization?

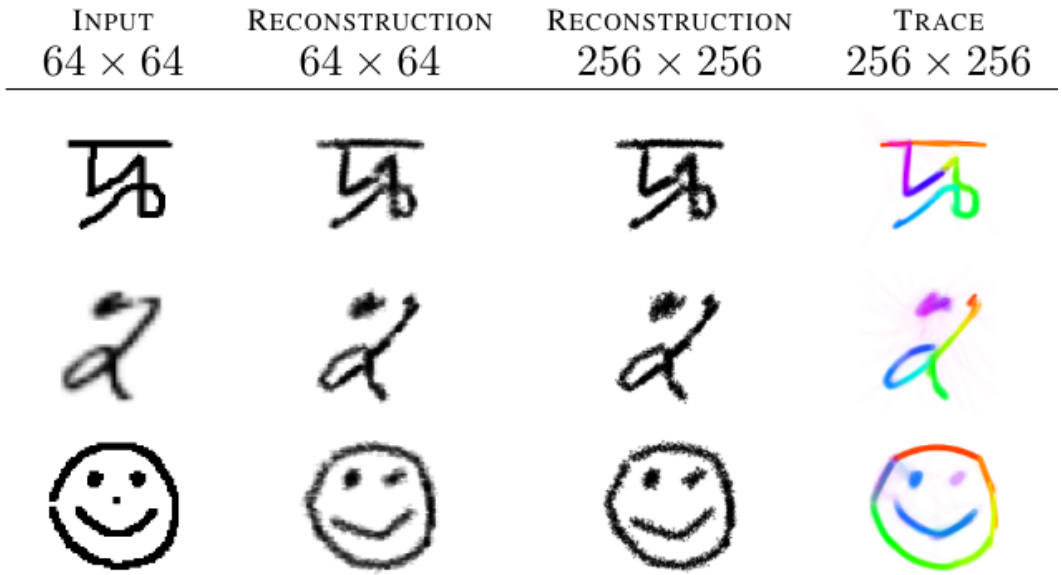


[19] Ho, J. and Ermon, S., 2016. Generative adversarial imitation learning. In Advances in neural information processing systems (pp. 4565-4573).

GANs for Probabilistic Programming

SPIRAL [20]

- Uses a probabilistic programming approach, which can in some sense be compared to the DreamCoder algorithm
- Adversarially trains an agent, who then generated the program which describes the scene
- First demonstration of end-to-end, unsupervised and adversarial inverse graphics agent on a real worl dataset



[20] Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S.M. and Vinyals, O., 2018. Synthesizing programs for images using reinforced adversarial learning. arXiv preprint arXiv:1804.01118.

GANs for Probabilistic Programming

SPIRAL [21]

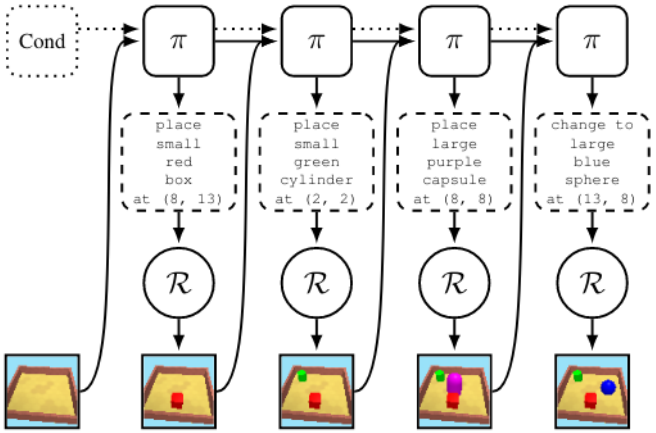


Figure: Execution traces.

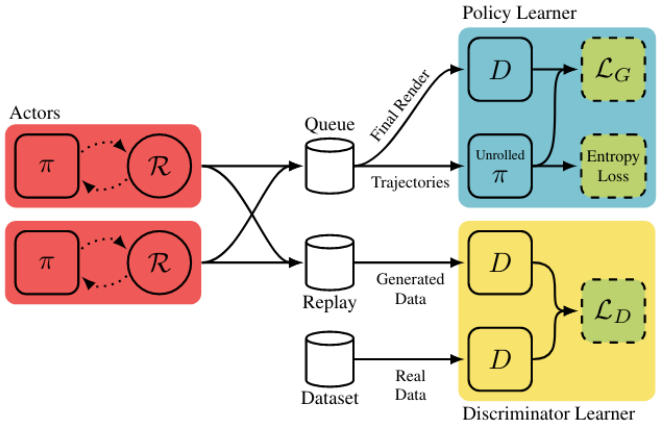


Figure: Distributed training of the algorithm.

[21] Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S.M. and Vinyals, O., 2018. Synthesizing programs for images using reinforced adversarial learning. arXiv preprint arXiv:1804.01118.

Chair of Aerodynamics and Fluid Mechanics
Department of Mechanical Engineering
Technical University of Munich

GANs for Probabilistic Programming

Jupyter Notebook

In the Jupyter Notebook we will now sketch the introduction of GANs into probabilistic programming systems.

VAEs for Probabilistic Programming

Theory [24]

- A generative model with continous latent variables
  - The likelihood $p(x|z)$ and the variational posterior $q_\phi(z|x)$ get parameterized with neural networks
  - Prior and variational posterior tend to be fully factorized Gaussians
  - Exploiting the reparameterization trick, one usually applies amortized variational inference
- Introduction in 2014 by Kingma and Welling [22]
- A further review of variational auto-encoders is available here [23]

[22] Kingma, D.P. and Welling, M., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
[23] Kingma, D.P. and Welling, M., 2019. An introduction to variational autoencoders. arXiv preprint arXiv:1906.02691.
[24] Subsection modeled after Andriy Minh's UCL x Deepmind lecture

VAEs for Probabilistic Programming

Theory

- Prior: $p(z) = \mathcal{N}(0, I)$
- Likelihood, i.e. the decoder:
  - For binary data:

$$p_\theta(x|z) = \text{Bernoulli}(\text{NeuralNetwork}_\theta(z))$$

  - For real-valued data:

$$p_\theta(x|z) = \mathcal{N}(\text{NeuralNetwork}_\theta(z), \text{diag}(\text{NeuralNetwork}_\theta(z)))$$

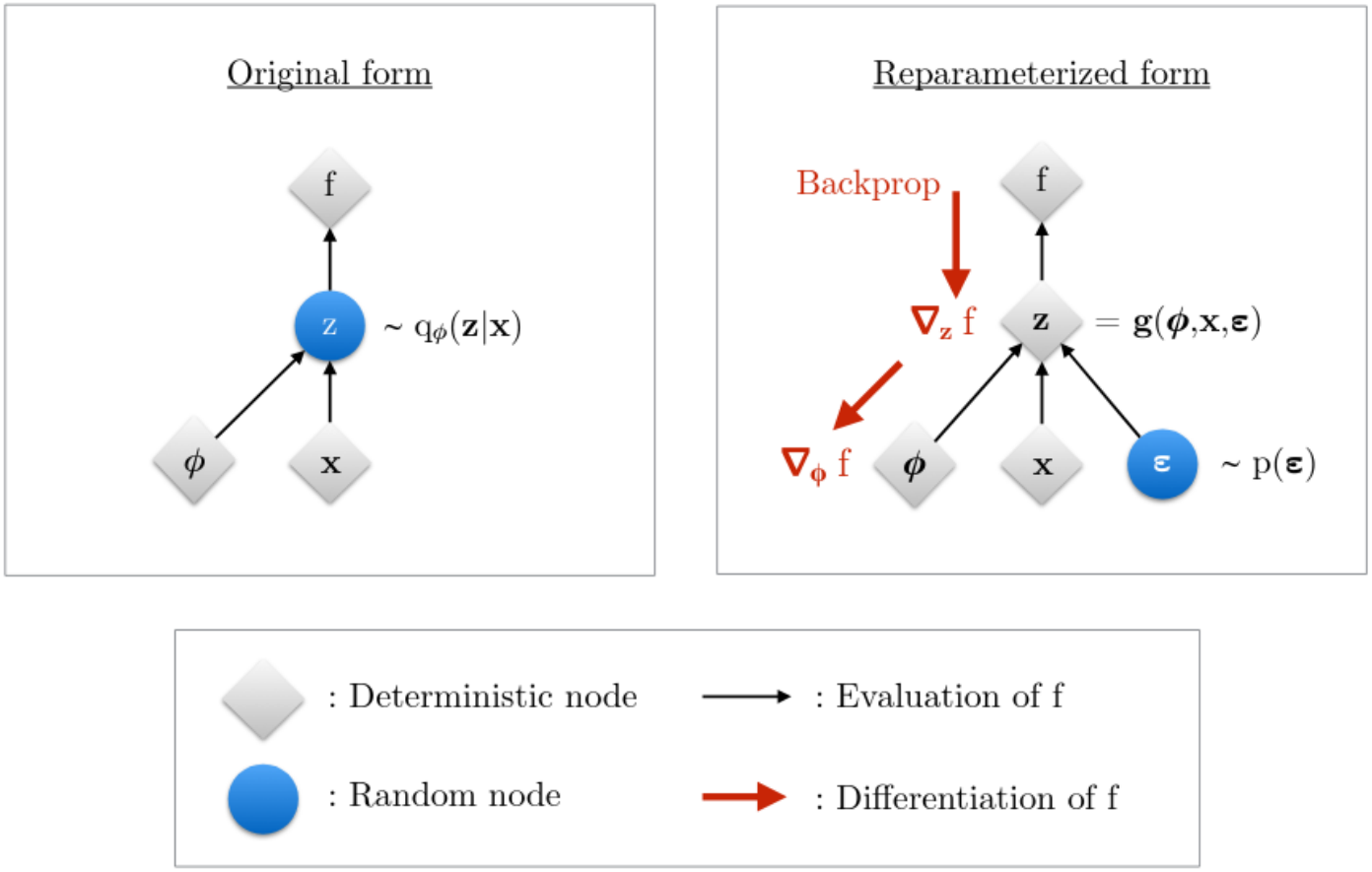
- Variational posterior, i.e. the encoder is given by

$$q_\theta(z|x) = \mathcal{N}(\text{NeuralNetwork}_\theta(z), \text{diag}(\text{NeuralNetwork}_\theta(z)))$$

- The neural network can be replaced with all types of networks, recently graph-nets have come more and more into fashion

VAEs for Probabilistic Programming

Theory



VAEs for Probabilistic Programming

Theory

- VAEs are trained, like variational inference algorithms, by maximizing the ELBO
- Where the positive part measures the quality of the model's prediction/reconstruction when given an observation of a sample from the variational posterior → reconstruction error
- The KL-divergence then acts as a regularizer, which pushes the variational posterior in the direction of the prior.
  - Measure for information about the observation in the latent variables

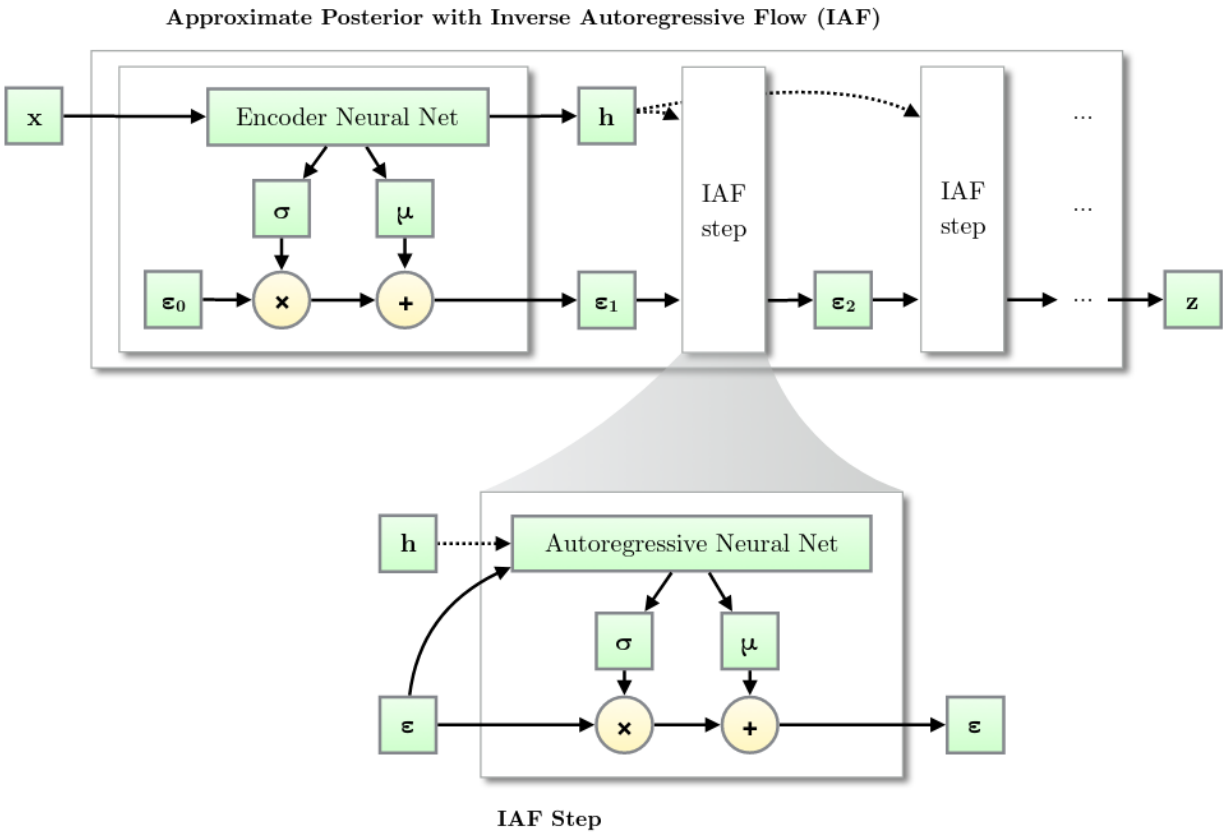
VAEs for Probabilistic Programming

Theory

- Theory has been softened up so that by now VAEs refers mostly to a framework of a continuous latent variable model which is being trained with amortized variational inference
- There exist a great many extensions to the original VAE framework:
  - Multiple latent layers
  - Non-Gaussian latent variables
  - Non-Gaussian Posteriors
    - Inverse Autoregressive Transformations
    - Inverse Autoregressive Flows
  - Mixtures, autoregressive, flow-based, implicit priors and posteriors
  - More advanced networks in the decoder
    - ResNet
    - Autoregressive Models
    - Graph-Nets
  - Improvements to the amortized inference

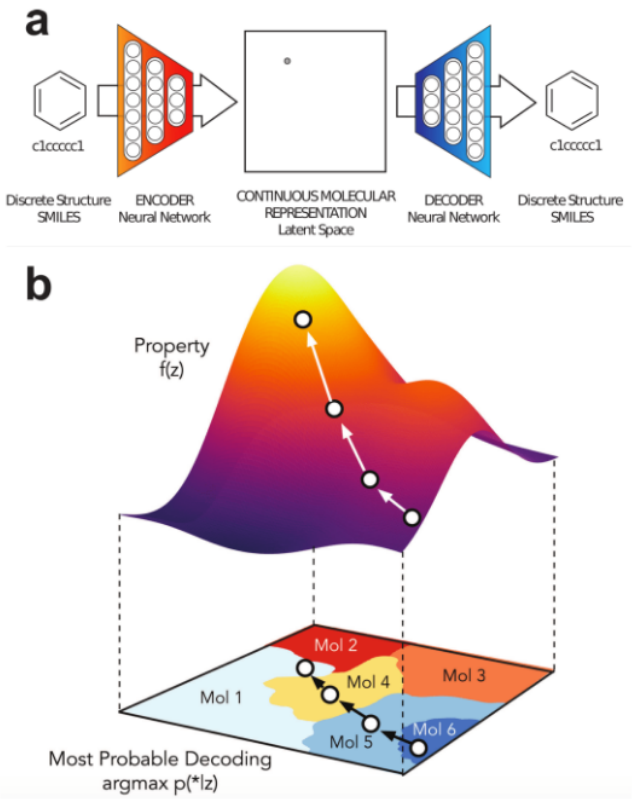
VAEs for Probabilistic Programming

VAEs for Chemical Design



VAEs for Probabilistic Programming

VAEs for Chemical Design



VAEs for Probabilistic Programming

Mixed-Curvature Variational Autoencoders

- Transforms the latent space and the associated prior distribution onto a Riemannian manifold
- The VAE is then trained on this curved space
- Positioned as a generalization of classical VAE theory

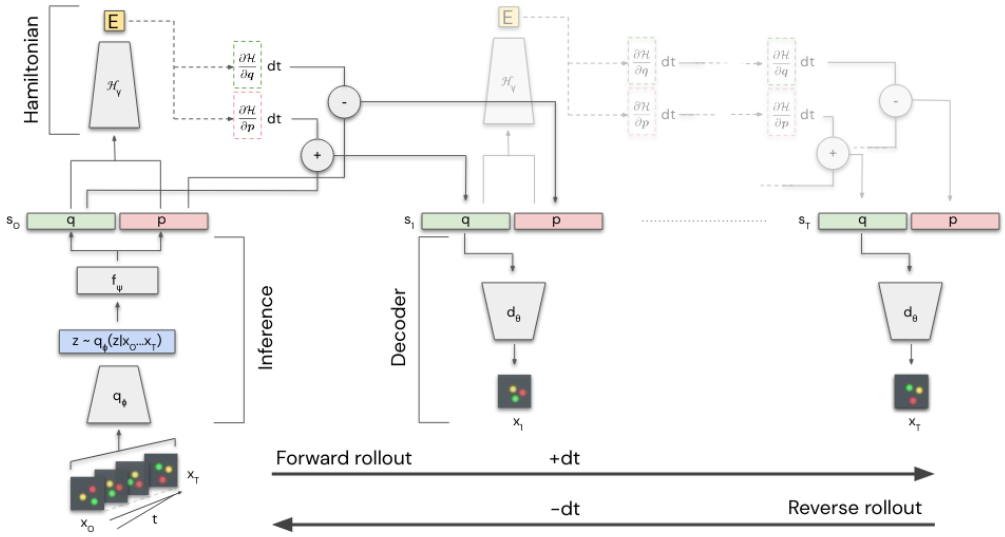
Table 2: Summary of operations in projected spaces $\mathbb{D}_K$ and $\mathbb{P}_K$.

Distance	$d(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\sqrt{\lvert K\rvert}} \cos_K^{-1}\left(1 - \frac{2K \lVert \boldsymbol{x} - \boldsymbol{y}\rVert_2^2}{(1 + K \lVert \boldsymbol{x}\rVert_2^2)(1 + K \lVert \boldsymbol{y}\rVert_2^2)}\right)$
Gyrospace distance	$d_{\mathrm{gyr}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{2}{\sqrt{\lvert K\rvert}} \tan_K^{-1}(\sqrt{\lvert K\rvert} \lVert -\boldsymbol{x} \oplus_K \boldsymbol{y}\rVert_2)$
Exponential map	$\exp_{\boldsymbol{x}}^K(\boldsymbol{v}) = \boldsymbol{x} \oplus_K \left(\tan_K\left(\sqrt{\lvert K\rvert}\frac{\lambda_{\boldsymbol{x}}^K \lVert \boldsymbol{v}\rVert_2}{2}\right)\frac{\boldsymbol{v}}{\sqrt{\lvert K\rvert}\lVert \boldsymbol{v}\rVert_2}\right)$
Logarithmic map	$\log_{\boldsymbol{x}}^K(\boldsymbol{y}) = \frac{2}{\sqrt{\lvert K\rvert}\lambda_{\boldsymbol{x}}^K} \tan_K^{-1}\left(\sqrt{\lvert K\rvert} \lVert -\boldsymbol{x} \oplus_K \boldsymbol{y}\rVert_2\right)\frac{-\boldsymbol{x} \oplus_K \boldsymbol{y}}{\lVert -\boldsymbol{x} \oplus_K \boldsymbol{y}\rVert_2}$
Parallel transport	$\mathrm{PT}_{\boldsymbol{x} \to \boldsymbol{y}}^K(\boldsymbol{v}) = \frac{\lambda_{\boldsymbol{x}}^K}{\lambda_{\boldsymbol{y}}^K}\mathrm{gyr}[\boldsymbol{y}, -\boldsymbol{x}]\boldsymbol{v}$

VAEs for Probabilistic Programming

Hamiltonian Generative Networks

- Able to learn the Hamiltonian from the image space
- Hamiltonian can subsequently be used for rollouts and for the guidance of trajectory sampling
- It can furthermore be modified into a normalising flow model, which uses Hamiltonian dynamics to model expressive densities



VAEs for Probabilistic Programming

Hamiltonian Generative Networks

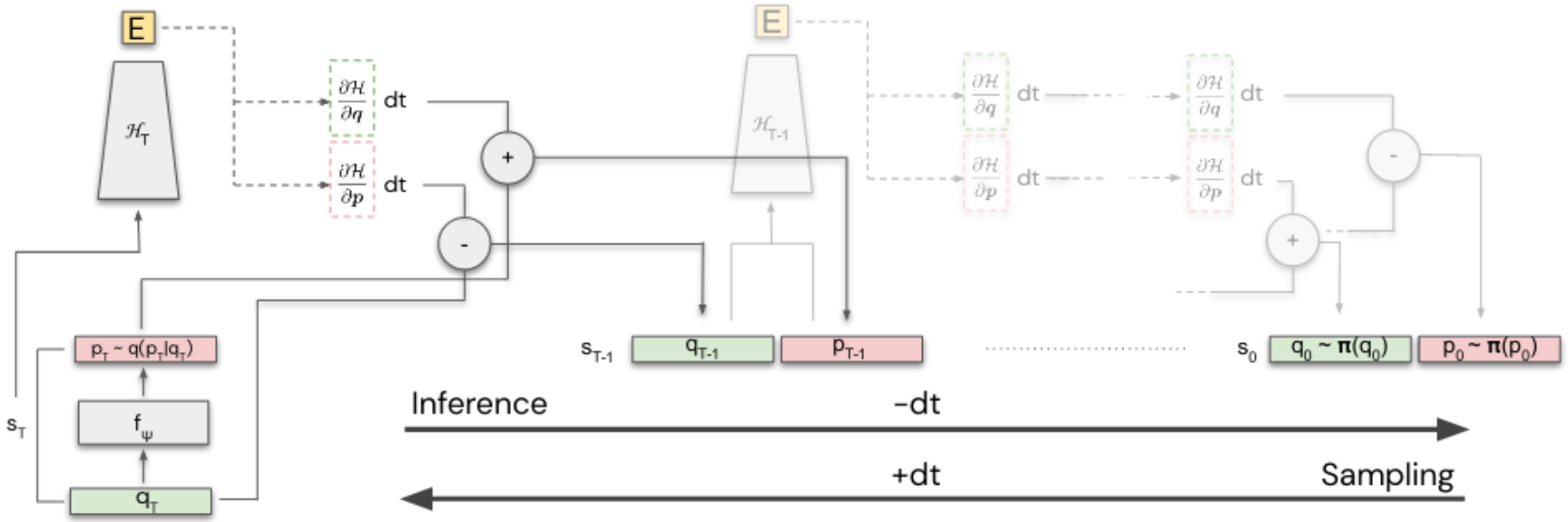
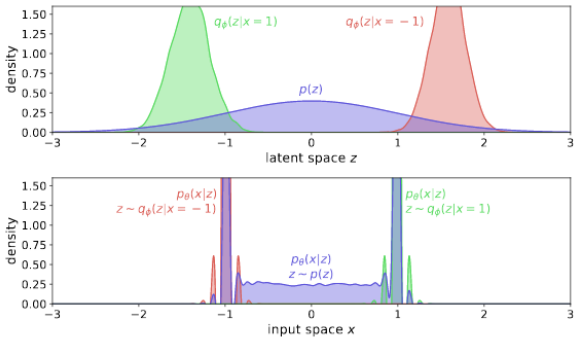


Figure: Using the learned Hamiltonian as a normalising flow.

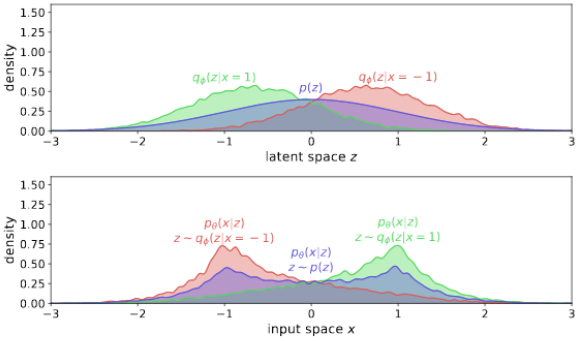
VAEs for Probabilistic Programming

InfoVAE: Information Maximizing Variational Autoencoders

- Proposes a new class of training objectives, which can make efficient use of the latent features
- At its core the approach centers around the maximum-mean discrepancy



ELBO

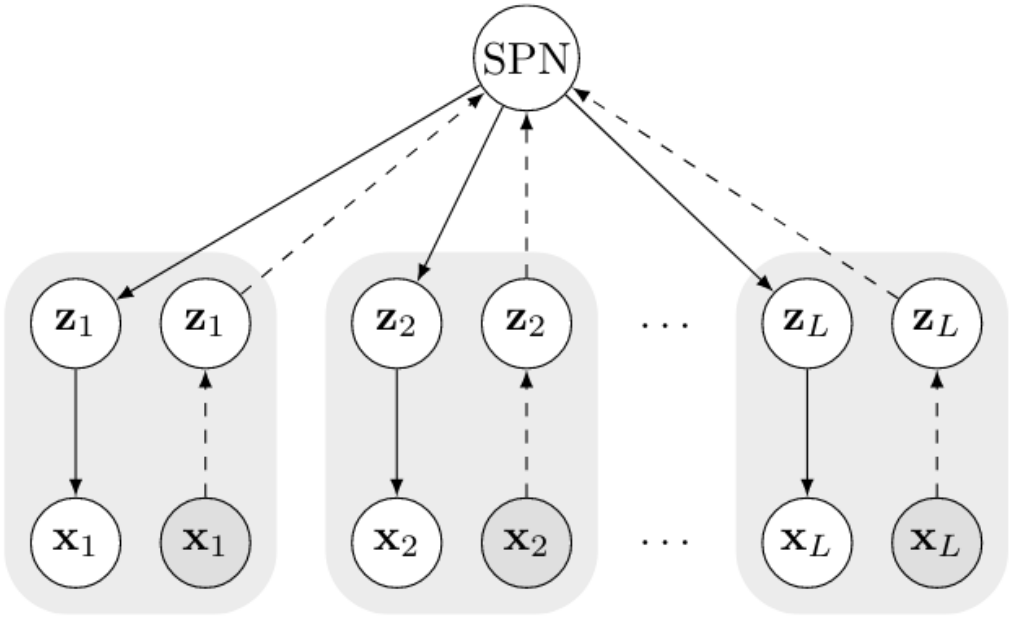


InfoVAE ($\lambda = 500$)

VAEs for Probabilistic Programming

Hierarchical Decompositional Mixtures of Variational Autoencoders

- Proposes the use of a hierarchical mixture model over low-dimensional experts
- Can essentially be likened to a sparse Mixture-of-Experts layer
- Decomposes the inference down into smaller inference routines, hence improving performance



VAEs for Probabilistic Programming

Jupyter Notebook

In the Jupyter Notebook we will now sketch how VAEs can be integrated into probabilistic programming systems.