UnderCurrent: Deep Normalizing Flows for Robotics Applications

Harrison Delecki[*]
hdelecki@stanford.edu

Liam Kruse[*]
lkruse@stanford.edu

Marc Schlichting[*]
mschl@stanford.edu

1 Introduction and Novelty

One of the most important tasks in robotics applications is the estimation of states of a system. Examples of state variables include position, velocity, or the orientation of the robot. System measurements or robot dynamics might be stochastic in nature, i.e., we do not know the exact state and assume the states are random variables distributed according to some unknown distribution. The core state estimation task is to find the underlying distribution. Furthermore, the task is assumed to be autoregressive—that is, the estimation of the state $\mathbf{x}_t$ can only use the observations $\mathbf{o}_{1:t}$ and no future observations. In more formal terms, state estimation is a regression problem where the goal is to fit a probability density function $p_\theta(\mathbf{x}_t)$ that is parameterized by θ to current and previous observations $\mathbf{o}_{1:t}$. Unfortunately, classical state estimation techniques often lack the expressive power to represent complex underlying distributions, especially if the system dynamics are highly nonlinear or if outcomes are multi-modal.

Normalizing flows are a deep generative method for constructing probabilistic models of complex distributions using parametric variable transforms [1]. Transforms can be parameterized by simple deep neural networks, invertible neural networks, or recurrent networks. Normalizing flows have been applied to model high dimensional and multi-modal distributions in image generation and time-series modeling [2]. In this project we seek to improve upon existing architectures for normalizing flows by using more expressive deep neural network architectures. Furthermore, we apply our deep normalizing flow framework to robotics environments such as autonomous driving—an area that has received little attention in the normalizing flow literature thus far. Furthermore, we contribute a direct comparison between different deep recurrent architectures for sequential observations. This includes a novel approach of using transformers for learning embeddings in the context of normalizing flows.

2 Related Work

Classical filtering techniques such as the ubiquitous Kalman filter [3] are well-suited for applications in which unimodal Gaussian distributions represent the state distributions. To deal with non-linear system dynamics, extensions such as the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) [4] have been introduced. However, many complex systems—especially highly stochastic multi-agent systems—induce multi-modal distributions over outcomes for which classical Gaussian filter techniques fail.

Given the potential multi-modality of the distribution $p(\mathbf{x}_t)$, we must consider more advanced models for density estimation such as Gaussian mixture models (GMM) or mixture density networks (MDN) [5]. Mixture density networks (MDN) use deep neural networks to parameterize a conditional GMM which is optimized with a maximum likelihood objective. MDNs have found applications in many domains such as human pose estimation [6], trajectory prediction [7], and speech synthesis [8]. While they perform well in lower dimensional space, they struggle with high-dimensional and non-Gaussian target distributions.

In recent years, deep generative models have been applied to the state estimation task. However, not all deep generative models provide explicit access to the probability density function. For example, generative adversarial networks (GANs) [9] can only sample from the latent distribution and do not explicitly compute its density. Variational autoencoders (VAE) [10] can provide access to the latent distribution while separately training an encoder and decoder. The training goal for VAEs has two components: learning a transform $Z = f(X)$ such that $Z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and learning the approximate inverse of f. In the context of state estimation tasks, VAEs have been used for estimating and sampling the distribution of the quantum states of a system [11] or for representing distributions over electroencephalogram measurements [12]. However, VAEs often suffer from poor reconstruction errors because of the two-objective training goal.

The inherent issue with an encoder/decoder structure—namely, that the decoder is merely an approximate inverse of the encoder—is improved upon by the normalizing flow architecture, which learns a single invertible mapping. The interpretable yet highly expressive nature of normalizing flows has garnered increased research interest in the robotics community. Deng *et al.* [13] use normalizing flows to decode a base continuous stochastic process into a complex observable process. Ma, Gupta, and Kochenderfer [14] represent multi-modal policies with conditional normalizing flows and demonstrate their effectiveness on a series of multi-agent games. Normalizing flows can also improve importance sampling, as Zhang, Sun, and Tian [15] learn the distribution of risk events for autonomous vehicles and fit an importance sampling distribution with a masked autoregressive flow. Zanfir *et al.* [16] conduct human

[*]H. Delecki, L. Kruse, and M. Schlichting are with the Stanford Intelligent Systems Laboratory in the Department of Aeronautics and Astronautics at Stanford University, Stanford, CA 94305, USA

CS230: Deep Learning, Autumn 2022, Stanford University, CA.

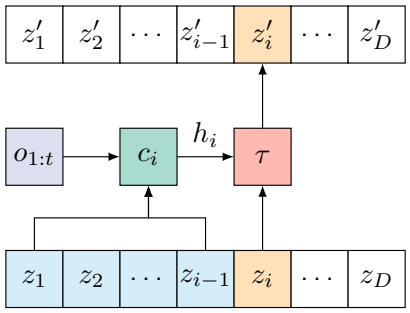


Figure 1: The i-th step of our conditional autoregressive normalizing flow, as adapted from [1]. The conditioning operator c_i takes as input latent variables z and observation sequence $o_{1:t}$. The operator τ uses the output h_i to transform latent variables.

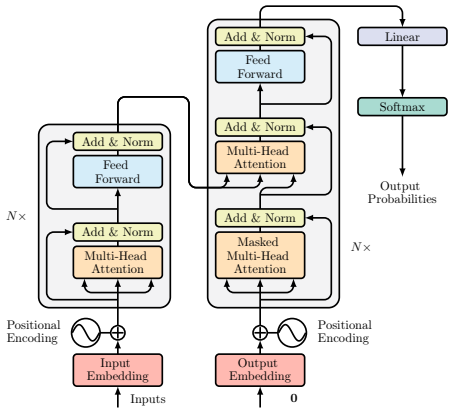


Figure 2: Transformer architecture, as adapted from [17]. We propose the use of transformers for the conditioning operator in a normalizing flow.

pose and shape estimation using kinematic latent normalizing flows. However, the use of deep normalizing flows in robotic state estimation applications such as autonomous driving and UAV pose estimation remains largely unexplored. We propose the use of our *UnderCurrent* framework to represent the complex and possibly multimodal beliefs over underlying robot states, mapping complex real-world distributions to highly interpretable latent representations using more advanced deep neural network architectures.

3 Methodology

Normalizing flows represent a target distribution $p(\mathbf{x})$ by transforming an easy-to-evaluate latent distribution $p(\mathbf{z})$ through the change of variable formula:

$$p(\mathbf{x}) = p(\mathbf{z})|\det J_f(\mathbf{z})|^{-1} \text{ where } \mathbf{z} = f^{-1}(\mathbf{x}) \tag{1}$$

where J_f is the Jacobian of the transformation f, which is invertible and differentiable. f is typically composed of i transformations $f = f_i \circ ... \circ f_1$, parameterized by θ. The parameters are optimized by minimizing the KL-Divergence between the target distribution and the latent space under the transform f.

3.1 Normalizing Flows

Autoregressive flows are one of the most popular classes of flow. In autoregressive flows, the transformations are constructed in an autoregressive manner. For a single transformation step, the transform of the i-th component of $\mathbf{z}$ to $\mathbf{z}'$ is

$$z'_i = \tau(z_i, \mathbf{h}_i) \text{ where } \mathbf{h}_i = c_i(\mathbf{z}_{<i}) \tag{2}$$

where τ is the *transformation operator* and c_i is the *conditioning operator*. This step is illustrated Fig. 1. The transformation operator must be invertible. The conditioning operator has no restrictions and can be parameterized by an arbitrary deep network.

For the classical setup that mimics the GAN or VAE functionality of sampling from the distribution, normalizing flows require a tractable inverse of the flow function. However, the state estimation task only requires the forward pass through the flow function and the guarantee that the flow is a diffeomorphism. This encourages us to explore various deep neural network architectures for the conditioning operator.

3.2 *UnderCurrent* Flow Architecture

Our *UnderCurrent* framework consists of stacked layers of *permutation*, *linear*, and *masked affine autoregressive* flows. Permuting input variables between flow layers has been shown to help the model learn the target transformation [1]. A linear flow is an invertible linear transformation of the form $\mathbf{z} = \mathbf{W}\mathbf{x}$, where $\mathbf{W}$ is an invertible matrix that parameterizes the transformation. Finally, a masked affine autoregressive layer in the style of Fig. 1 passes outputs to subsequent layers. A deep conditioning operator conditions the flow on the provided *context*, while a context encoder encodes the conditional information to the latent distribution parameters.

3.3 Conditioner Scheme

The i-th conditioning operator takes $\mathbf{z}_{1:i-1}$ and $\mathbf{o}_{1:t}$ as an input and outputs the parameters $\mathbf{h}_i$ that parameterize the transformation operator τ. In a naive approach, we could train i different conditioners; however, this is inefficient as all conditioners have a similar task. Using a single conditioning operator for all i flow transformations requires deep architectures that take sequential inputs such as recurrent neural networks (RNN) [18], gated recurrent units (GRU) [19], long short-term memory (LSTM) [20], or transformers [17]. Sequences of observations $\mathbf{o}_{1:t}$, possibly of varying length, necessitate a recurrent architecture.

4 Dataset

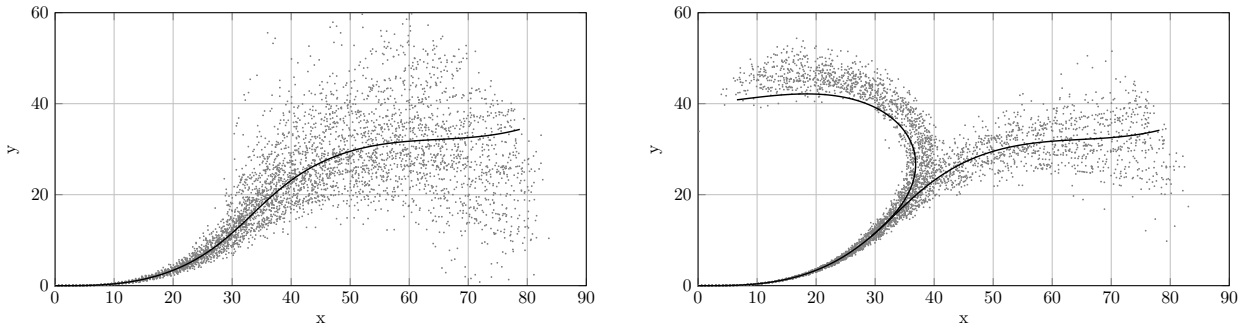
We construct two driving datasets in simulation to validate the proposed architecture. Consider a nonholonomic car with position $(p_x,\ p_y)$ and heading angle θ, and control over its velocity v and angular acceleration ϕ. The car's equations of motion are given by

$$p_x^{t+1} = p_x^t + \Delta t \cdot v^t \cdot \cos(\theta^t) \qquad p_y^{t+1} = p_y^t + \Delta t \cdot v^t \cdot \sin(\theta^t) \qquad \theta^{t+1} = \theta^t + \Delta t v^t \phi^t$$

The angular acceleration is given by

$$\phi^{t+1} = \phi^t + \Delta t \cdot \psi \cdot c_1 \cdot \cos(c_2 \cdot t)$$

where c_1 and c_2 are constants and ψ is a *switching parameter* used to induce multimodalities into the dataset. Time-stamped coordinates from individual rollouts, (p_x, p_y, t), with noise-corrupted inputs v and ϕ are shown in Fig. 3. The black lines indicate the nominal noise-free trajectory. A unimodal dataset, shown in Fig. 3a, is generated by fixing $\psi = 1$ for every rollout. A bimodal dataset representing paths through a traffic circle, shown in Fig. 3a, is generated by randomly selecting $\psi \sim \mathcal{U}([-1, 1])$ at a fixed time index and thereafter holding ψ constant for the remainder of the rollout. Each dataset contains over 1.5 million data points.



(a) Unimodal driving dataset

(b) Bimodal driving dataset

Figure 3: Driving datasets generated in simulation.

5 Experimental Results

We demonstrate the representational capabilities of a deep normalizing flow-based approach on two state-estimation tasks using our autonomous driving datasets. The source code is available online[2].

5.1 Temporal Conditioning Task

For the first state estimation task we consider the unimodal dataset and condition on the time stamp of each data point, estimating the conditional distribution $p(p_x, p_y \mid t)$. An MLP context encoder encodes the conditional information to the parameters of a latent conditional diagonal normal distribution, while the conditioning operator is simply the identity function. We benchmark our proposed approach against three baselines:

- **Unscented Kalman Filter**: A UKF is a classical recursive Bayesian filter that relies on a deterministic sampling strategy to approximate the effect of a distribution undergoing a nonlinear transformation [4].
- **Mixture Density Network**: An MDN represents a conditional Gaussian mixture model where the components are parameterized by a deep neural network. We use an MDN with 5 mixture components parameterized by a two-layer network with 8 neurons per layer and *tanh* activations.
- **VAE**: We use a conditional VAE architecture with identical encoder and decoder architectures, each consisting of 8 hidden layers with 64 neurons each and *tanh* as the activation function.

The results of the baseline methods and our current implementation of normalizing flows are depicted in Fig. 4 where we show the 1σ, 2σ, and 3σ confidence regions. Additionally, we estimate the KL divergence between the dataset and 1000 samples drawn from the fitted models using a two-sample KL divergence estimator [21] in Table 1. Both from the visual evidence and the quantitative analysis, our normalizing flow implementation outperformed the baselines.

5.2 Observation History Conditioning Task

For the second task we condition on sequences of noisy position observations and estimate the next state of the vehicle; that is, we estimate the conditional distribution $p(\mathbf{x}_t \mid \mathbf{o}_{1:t-1})$. Due to the time-series nature of the data, we consider recurrent conditioning operators. The following conditioning operator architectures are tested:

[2] `https://github.com/MarcSchlichting/UnderCurrent.git`

Method	$\hat{D}_{KL}(p(\mathbf{x})\|\hat{p}(\mathbf{x}))$
Unscented Kalman Filter	0.7578
Mixture Density Network	0.2612
Variational Autoencoder	0.7978
Normalizing Flows	**0.0684**

Table 1: Estimated KL divergence between models and dataset. Lower KL divergence indicates better performance.

- **Recurrent Neural Network**: A single-layer RNN with three input features and four output features serves as a deep baseline.
- **Gated Recurrent Unit**: A single-layer GRU with three input features, four hidden features, and four output features is employed to improve upon the RNN baseline by solving the vanishing gradient problem.
- **Long Short-Term Memory**: A single-layer LSTM with three input features, four hidden features, and four output features is employed to improve upon the RNN baseline by solving the vanishing gradient problem while providing more expressivity than the GRU.
- **Transformer**: We use a transformer network and follow the same architecture as [22]. Since we use the transformer to learn an embedding, no ground truth for the target sequence is available. As learning an embedding is a many-to-one task, only one prediction step in "inference" mode is necessary. We initialize the target sequence required by the transformer with zeros, which is common practice for other recurrent methods.

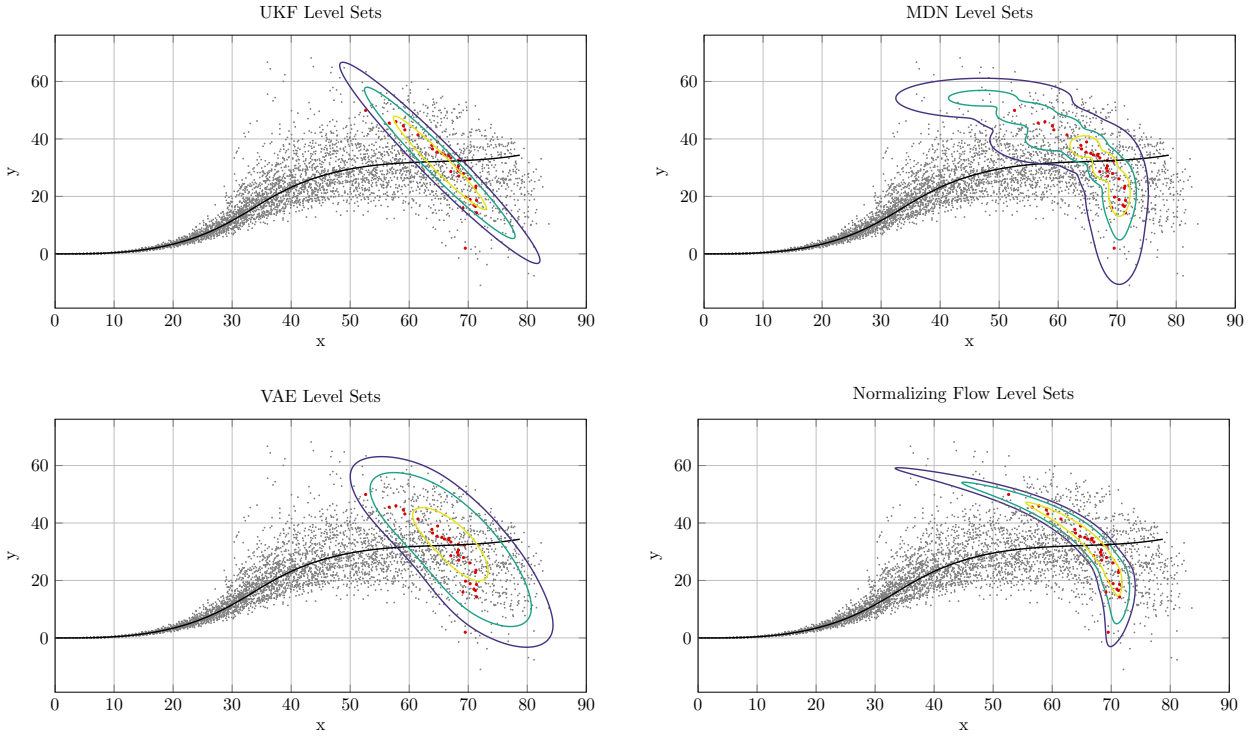


Figure 4: Confidence regions of baseline models and normalizing flows conditioned for $t = 13s$. The yellow (1σ), turquoise (2σ), and the purple (3σ) contours represent confidence regions of the model. The red dots are datapoints at $t = 13s$.

The results of the four deep recurrent conditioning operator architectures are depicted in Fig. 5 where we show the 1σ, 2σ, and 3σ confidence regions over the predicted next state. Additionally, we compute the mean log likelihood of the ground truth positions given the associated context. The model attempts to maximize the log likelihood of the noise-free data points, and thus a higher mean log likelihood indicates that the learned distribution more closely matches the true underlying belief. The Transformer architecture yields the highest mean log likelihood value, as shown in Table 2.

6 Discussion and Future Work

In this work we evaluate four deep conditioning operator architectures for conditioning a normalizing flow and an MLP context encoder for encoding conditional information to the parameters of a latent base distribution. Normalizing flows outperformed

Conditioning Operator	Log Likelihood.
RNN	1.4255
GRU	1.4818
LSTM	1.5057
Transformer	**1.5736**

Table 2: Mean log likelihood of undisturbed data points under the learned distribution, represented by the normalizing flow. Higher log likelihood indicates better performance.

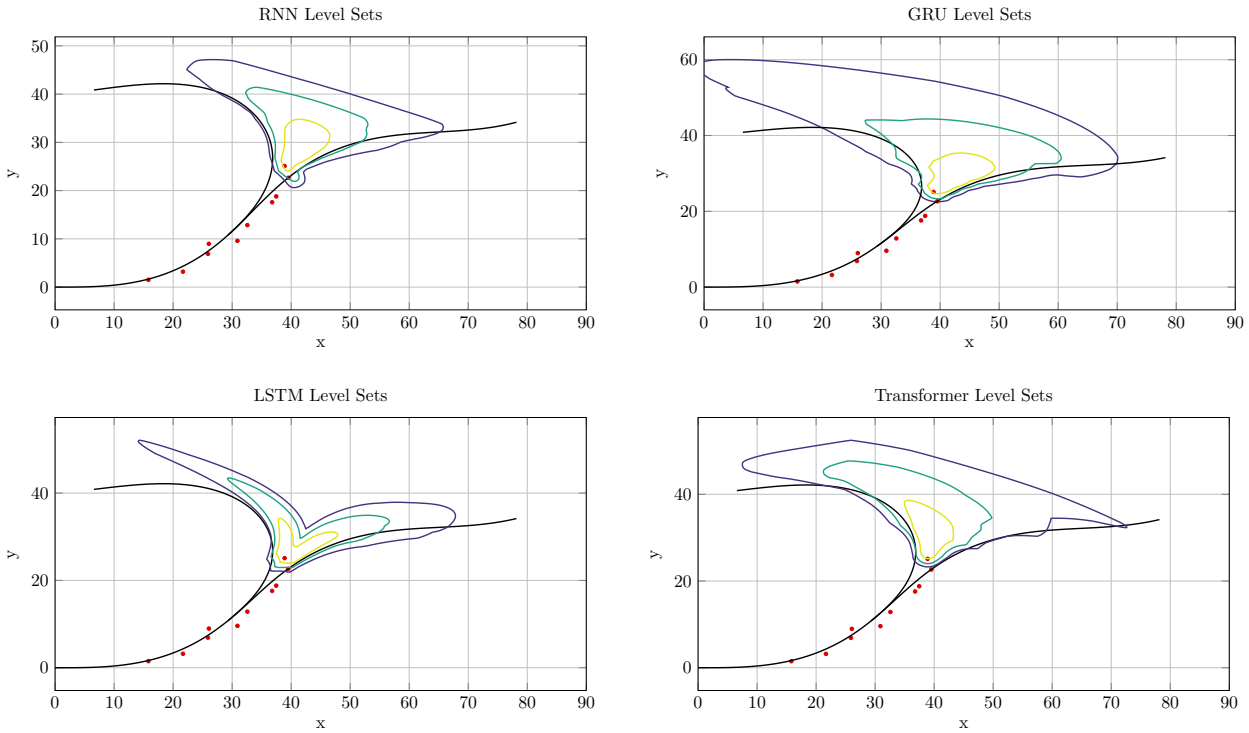


Figure 5: Confidence regions of four conditioning network architectures conditioned on a sequence of noisy observations. The yellow (1σ), turquoise (2σ), and the purple (3σ) contours represent confidence regions of the model. The red dots are noisy observations for a single trajectory. The trajectory is a left-branching trajectory. Both RNNs and GRU predict a rather right-branching trajectory, the LSTM is indecisive, and only the transformer has a slightly left-branching prediction.

both classical state-estimation techniques and deep conditional networks on a simple state-estimation task—even when an identity function conditioning operator was used—due to their expressive representational capabilities. Furthermore, we demonstrated that deep recurrent conditioning operator architectures are well-suited for time-series context on a complex state-estimation task with a bimodal underlying belief distribution. A Transformer network was experimentally shown to provide the best quantitative results in the *UnderCurrent* framework. To the author's knowledge, this is the first time that a transformer architecture has been employed as a normalizing flow conditioning operator, and represents one of the first applications of normalizing flows to state-estimation tasks.

In future work we will consider alternative flow architectures such as Unconstrained Monotonic Neural Network autoregressive flows [23]. Furthermore, additional conditioning operator architectures—such as diffusion models—will be considered. Finally, we will validate the *UnderCurrent* framework on high-dimensional state-estimation tasks such as UAV pose estimation.

7 Contributions

This project is a team effort and implementing the normalizing flows was a highly collaborative effort. For the temporal conditioning task, L. Kruse contributed the UKF baseline, H. Delecki the MDN, and M. Schlichting the VAE. For the observation history conditioning task, L. Kruse contributed the RNN and GRU, H. Delecki the LSTM implementation, and M. Schlichting the Transformer embedding network.

A Appendix

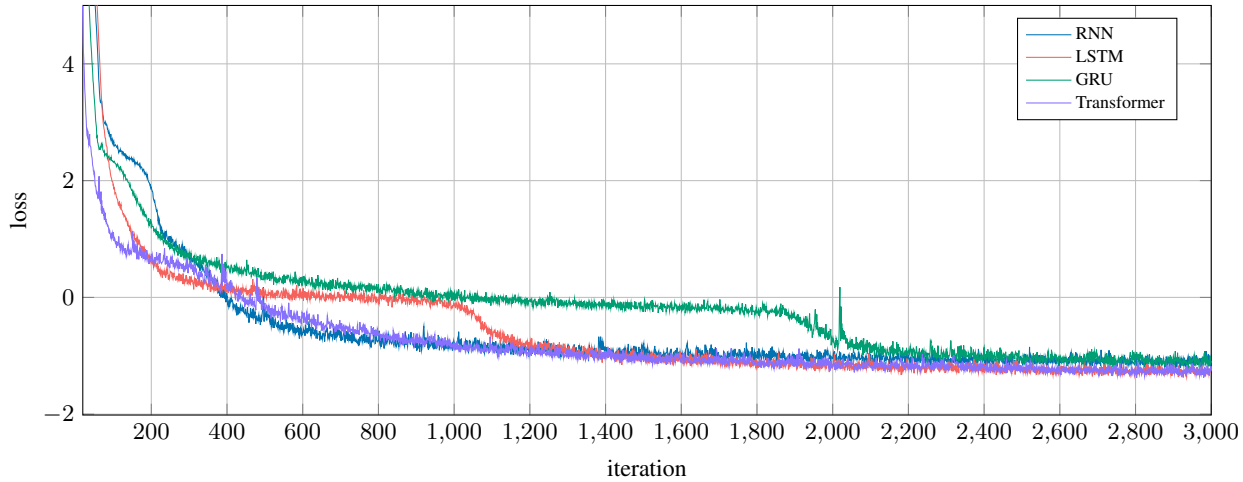


Figure 6: Loss for normalizing flows based on different embedding networks.

References

[1] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference.," *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.

[2] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, 2021.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[4] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, Spie, vol. 3068, 1997, pp. 182–193.

[5] C. M. Bishop, "Mixture density networks," 1994.

[6] C. Li and G. H. Lee, "Generating multiple hypotheses for 3d human pose estimation with mixture density network," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2019, pp. 9887–9895.

[7] O. Makansi, E. Ilg, O. Cicek, and T. Brox, "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[8] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 3844–3848.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[11] A. Rocchetto, E. Grant, S. Strelchuk, G. Carleo, and S. Severini, "Learning hard quantum distributions with variational autoencoders," *npj Quantum Information*, vol. 4, no. 1, pp. 1–7, 2018.

[12] D. Bethge, P. Hallgarten, T. Grosse-Puppendahl, *et al.*, "Eeg2vec: Learning affective eeg representations via variational autoencoders," *arXiv preprint arXiv:2207.08002*, 2022.

[13] R. Deng, B. Chang, M. A. Brubaker, G. Mori, and A. Lehrmann, "Modeling continuous stochastic processes with dynamic normalizing flows," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7805–7815, 2020.

[14] X. Ma, J. K. Gupta, and M. J. Kochenderfer, "Normalizing flow policies for multi-agent systems," in *International Conference on Decision and Game Theory for Security*, Springer, 2020, pp. 277–296.

[15] H. Zhang, J. Sun, and Y. Tian, "Accelerated testing for highly automated vehicles: A combined method based on importance sampling and normalizing flows," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2022, pp. 574–579.

[16] A. Zanfir, E. G. Bazavan, H. Xu, W. T. Freeman, R. Sukthankar, and C. Sminchisescu, "Weakly supervised 3d human pose and shape reconstruction with normalizing flows," in *European Conference on Computer Vision*, Springer, 2020, pp. 465–481.

[17] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[19] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*, arXiv:1409.1259 [cs, stat], Oct. 2014.

[20] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[21] F. Pérez-Cruz, "Kullback-leibler divergence estimation of continuous distributions," in *2008 IEEE international symposium on information theory*, IEEE, 2008, pp. 1666–1670.

[22] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *arXiv preprint arXiv:2001.08317*, 2020.

[23] A. Wehenkel and G. Louppe, "Unconstrained monotonic neural networks," *Advances in neural information processing systems*, vol. 32, 2019.