Generating ocean glider trajectories: a comparison of variational recurrent autoencoders and autoregressive models

Peter Boennighausen, Lucia Zheng, Jerry Hong
`pboennig`, `zlucia`, `jerryh2`

1 Introduction

Sequential data present unique challenges for generative models. In this report, we compare and contrast two types of generative models, a variational recurrent autoencoder (VRAE) and a simpler autoregressive approach, on a dataset of ocean glider trajectories.

Ocean gliders are unmanned underwater vehicles used to collect information such as pressure, salinity, and conductivity at multiple ocean depths and over a long period of time. This allows scientists to inexpensively monitor the state of a particular region of the ocean, observing features such as the strength of currents and the warming of the seas. The data gliders collect are time series data, with each observation coming at a regular interval. Thus, the trajectory of a glider is naturally sequential, as the current position, depth, and observations of a glider influence the glider's state at the next time step.

In this project, we apply generative modelling techniques to this sequential data in two ways. First, we seek to learn a compact representation of a given trajectory. Such a representation allows for exploratory data analysis of the underlying variation in trajectories. While some of this variation is due to human decisions — where exactly the scientists decide to send a glider — other sources of variation may reflect some underlying properties about the region of ocean studied. Second, we seek to generate plausible samples that fit nicely with the overall dataset of trajectories. Such a model would allow scientists to augment their datasets without having to conduct additional experiments.

This report considers two approaches to modelling ocean glider trajectories. The first model is a VRAE, which generalizes a variational autoencoder (VAE) to sequential data. Through its encoder, the VRAE is able to learn compact representations of trajectories in addition to generating sequences. The second is an autoregressive approach in which we learn a model that predicts future positions based on past time-steps, a natural approach for sequential data. This allows for the generation of samples, but does not learn a compact latent representation. Though the VRAE is able to learn a representation that detects outliers in the data, we find that the autoregressive approach is superior for generating sequences.

2 Previous and related works

VRAEs combine the ELBO objective and reparameterization trick of VAEs with the structure of recurrent neural networks (RNNs) and were introduced in Fabius & van Amersfoort (2015). The authors introduce recurrence in the encoder, so that the encoding of point x_t depends both on the features of x_t and the hidden state h_{t-1} that was calculated in the previous time step. Thus, the relationship between elements of the sequence is captured in the latent space. A similar exposition of variational RNNs (VRNNs) appeared the same year in which the authors apply their model to speech data (Chung et al. (2015)).

The technical implementation of VRAEs, which is our model of choice for this project, is further specified in Section 4. Our choice of VRAEs over VRNNs is justified by what the latent space represents in each model; whereas VRNNs learn the latent space of the *time steps* of a sequence, VRAEs learn the latent space of *entire* individual sequences. Given our interest in uncovering underlying structure and similarities amongst whole trajectories, VRAEs stand out as the more appropriate choice.

In contrast to the relatively recent innovation of VRAEs, autoregressive models are among the most well-established techniques in forecasting time series data. For example, an autoregressive integrated moving average (ARIMA) model forecasts sequences of a stationary random variable by averaging together past observations. For an overview, see Box et al. (2015). More recently, autoregressive models have been extended by the use of deep neural networks as predictors (rather than moving averages) and have also been applied to non-sequential data such as images. For an example of both of these trends, see the PixelCNN described in Oord et al. (2016).

The field of oceanography has not seen much application of generative modelling techniques, though some previous works exist. For example, Murray & Perera (2021) used VRAEs and ship tracking data from a region of Sweden to cluster and predict ship behavior. This paper, by focusing on trajectories in a defined geographical region, is naturally related to our project. In the specific field of ocean gliders, we were unable to find any papers applying deep generative modelling to the field. Ocean gliders are an area of research in control systems, as the utilization of existing ocean dynamics can enable the glider to move from one location to another, so some literature exists applying control techniques to the field, e.g. Inanc et al. (2005).

3 Dataset and preprocessing

Our data come from the Coastal Pioneer Array of the Ocean Observatories Institute off the coast of Martha's Vineyard. In data preprocessing, we deduplicated real-time and delayed-mode data for trajectories so our dataset only included either the real-time or delayed-mode data for a trajectory (not both), with a preference for real-time data if available. We also deduplicated each trajectory on the `profile_id` feature, a unique sequential profile number that identifies the order of an observation in a single trajectory, by taking the first observation for each `profile_id` for each trajectory. Finally, we filtered out observations with null latitude or longitude. After these preprocessing steps, we had 90 unique trajectories.

Since the existing literature truncates input data to a fixed length, we tried to find a reasonable fixed length for our data by analyzing the distribution over the trajectory lengths. We aimed to set the fixed length to a value that produced a good tradeoff between keeping a majority of the unique trajectories and having adequate training data to generate from for each trajectory. Using these two criteria, we chose a fixed length of 500 and filtered the data to include only trajectories with at least 500 observations, truncating those with more than 500 observations to 500. This gave us 72 trajectory sequences of length 500, which were used to create our dataset X. For simplicity, we only considered the observed latitude and longitude, so X is a tensor with dimension $72 \times 500 \times 2$. For the pre-processed data, see our Github.

4 Methods

4.1 Variational recurrent autoencoder (VRAE)

At a high level, a VRAE can be considered as a VAE with RNNs for its encoder and decoder. More explicitly, the encoder creates a set of hidden states h_t which depend deterministically on the previous hidden state and the data of the corresponding step, x_t. The final hidden state h_T is then used to define the probability distribution $q_\phi(z|x)$, which we constrain to be a Gaussian $\mathcal{N}(\mu_z, \sigma_z)$:

$$\begin{aligned} h_t &= f_\phi(h_{t-1}, x_t) \\ \mu_z &= \mu_\phi(h_T) \\ \sigma_z &= \exp(\sigma_\phi(h_T)) \end{aligned}$$

Note that the structure of a RNN allows us to both generate and capture the latent space of variable-length paths. To generate samples with the decoder, Z is sampled from the prior. Then, similarly to

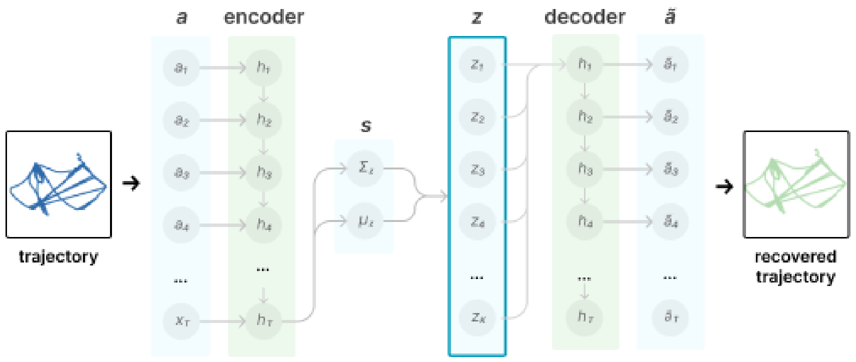


Figure 1: Summary of VRAE architecture

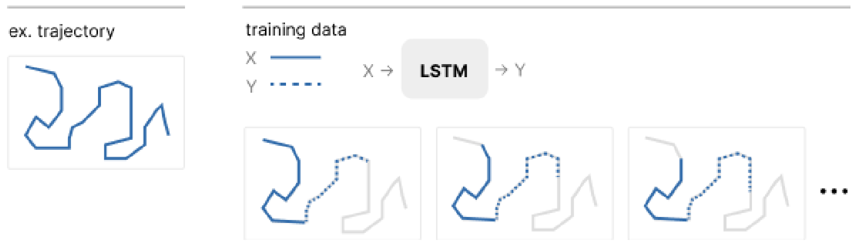


Figure 2: Deriving supervised data from trajectories to train the autoregressive model

the encoder, the hidden states and output states are generated in the manner of a traditional RNN:

$$h_0 = g_{z,\theta}(z)$$
$$h_t = g_{h,\theta}(h_{t-1}, x_{t-1})$$
$$\hat{x}_t = g_{out,\theta}(h_t)$$

In the same manner as a VAE, a VRAE is trained by optimizing a lower bound to the log likelihood. More specifically, we seek to optimize the ELBO:

$$ELBO(x^{(i)};\phi,\theta) = \mathrm{E}_{q_\phi(z|x^{(i)})}\left[\frac{\log p_\theta(z, x^{(i)})}{q_\phi(z|x^{(i)})}\right] \leq \log p(x^{(i)};\theta)$$

which we can re-express as the sum of a KL-divergence term and reconstruction term:

$$ELBO(x^{(i)};\phi,\theta) = -D_{KL}(q_\phi(z|x^{(i)})||p(z)) + \mathrm{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]$$

Since $p(z)$ is Gaussian, the KL-divergence has a tractable closed form.

To optimize the ELBO via SGD or Adam, we need to compute the gradients with respect to θ and ϕ. Gradients with respect to θ are straightforward, but since the expectation is taken over q_ϕ, gradients with respect to ϕ are more difficult. This is addressed via the reparametrization trick; in particular, we express $z = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$.

We leverage a previous implementation of the VRAE [1] which defines the specifics of the architecture. In contrast to Fabius & van Amersfoort (2015), this architecture does not opt for simple $\tanh$-activated linear transformations but instead employs stacked LSTMs or GRUs for the transition functions f_ϕ and $g_{h,\theta}$. We hypothesized this would allow us to train on longer time sequences, as Fabius & van Amersfoort (2015) notes their implementation only captures time dependencies up to 100 steps. Additionally, the generation of μ_z and $\log\sigma_z$ with a linear layer is retained, but the sigmoid activation for generating $\hat{x}_t$ is removed.

4.2 Autoregressive approach

We considered an autoregressive approach as an alternative model for the reconstruction task. The model takes as input $x_{t:t+M} = [x_t, \ldots x_{t+M}]$, which we refer to as the *conditioning range*, and

[1]https://github.com/tejaslodaya/timeseries-clustering-vae

outputs a prediction for the next $x_{t':t'+N} = [x_{t'}, \dots x_{t'+N}]$ data points in the time series, which we refer to as the *prediction range*, using the terminology from Salinas et al. (2020). This model satisfies the autoregressive property since predictions for future data points in the time series sequence depend only on past data points.

We transform the dataset in order to train the baseline model with the supervised training task described above. Each sequence in the dataset is cut into conditioning range windows of length M and prediction range windows of length N, with the starting time step t for the conditioning range window increasing by 1 until the end of the sequence is reached. The transformed dataset has 35,000 input output pairs. We split the transformed dataset into train, validation, and test set with a 70:15:15 ratio.

We chose to use an LSTM for the model, since it has greater capacity to learn long range time dependencies in the time series data, though another sequence model could have been used instead. Since our focus is on reconstructing a reasonable glider trajectory from an initial conditioning range window (in our validation set), rather than generating a completely random sample glider trajectory, we feed the conditioning range window as input to the LSTM and then feed the last hidden state from the LSTM to a linear layer to output a direct prediction for the prediction range window. This is in contrast to using the hidden state of the LSTM to model a probability distribution over outputs, as is done in the encoder-decoder LSTM architecture described in Salinas et al. (2020).

We train the model with mean squared error loss between the true prediction range window, $x_{t':t'+N}$, and the predicted prediction range window, $\hat{x}_{t':t'+N}$.

$$L(x_{t':t'+N}, \hat{x}_{t':t'+N}) = \frac{1}{N}(x_{t'+i} - \hat{x}_{t'+i})^2$$

To reconstruct a glider trajectory, we use the first N data points of a sequence from the validation / test set as the initial conditioning range window and predict one data point, then reset the conditioning range window by increasing the starting index by 1 to include the new predicted data point.

We iterate through a set of hyperparameters $M, N \in \{2, 5, 10, 20, 50\}$ and hidden dimensions $\{5, 20, 50, 100, 200\}$ to optimize model performance. We find that transforming the data with $M = 10$, $N = 10$ and training the LSTM with a hidden state size of 50 minimizes validation loss and produces reasonable reconstructions of our data.

4.3 Code

The code for this project, including data and plots, can be found on Github. Our implementation of the VRAE is a fork of a previous implementation. The original implementation can be found here while the fork can be found here.

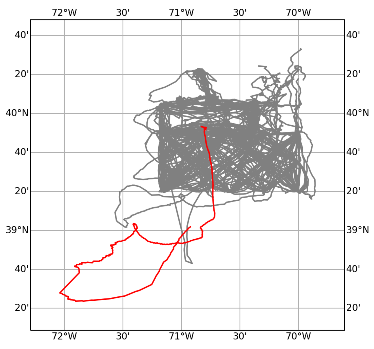
5 Results

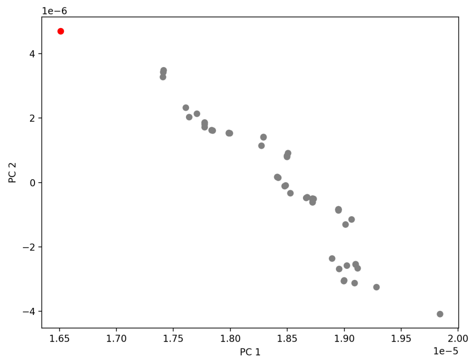
5.1 VRAE

5.1.1 Outlier detection

As a qualitative heuristic for the quality of our VRAE's compact representations, we consider whether an "outlying" representation corresponds with an "outlying" trajectory. We initially use an encoder/decoder constructed by the stacking of $L = 4$ GRU cells, generating hidden states of dimension $H = 200$ and outputting a latent state of dimension $Z = 50$. This is trained with a learning rate of $\eta = 0.01$ over 100 epochs with a conventional ELBO objective.

In this configuration of hyperparameters, the representations of our 72 trajectories mapped to PCA space, shown in Figure 3b, produce a clear outlier highlighted in red. This outlier corresponds to the trajectory in Figure 3 highlighted in red, which clearly meanders in a manner unlike the remaining trajectories. Despite this prominent outlier, however, we note that viewing the original trajectory in PCA space, as shown in Figure 3c, captures the outlier to a greater extent, suggesting that our representations may not capture uniquely salient properties of a trajectory. We also note the unusual correlation between the two principal values of our representations, suggesting that there may be a


(a) Outlier as a trajectory.


(b) Outlier in representation PCA space.

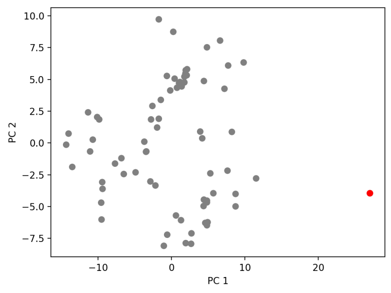

(c) Outlier in trajectory PCA space.

Figure 3: VRAE encodings of trajectories identify outliers to a similar, but lesser, extent than directly viewing trajectories in PCA space.

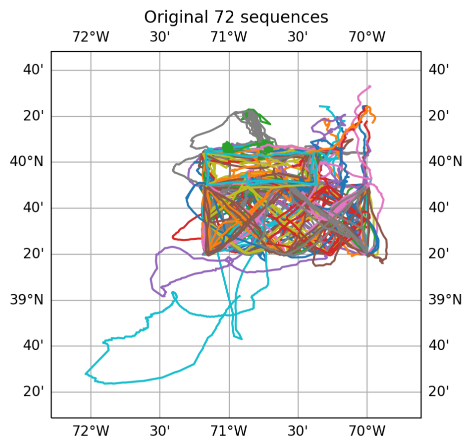
further possible reduction of representation dimensionality or, as we expand on later, a total collapse of the posterior distribution. However, these initial experiments suggest some capacity of the VRAE to capture a meaningful latent state for our trajectories.

5.1.2 Reconstruction

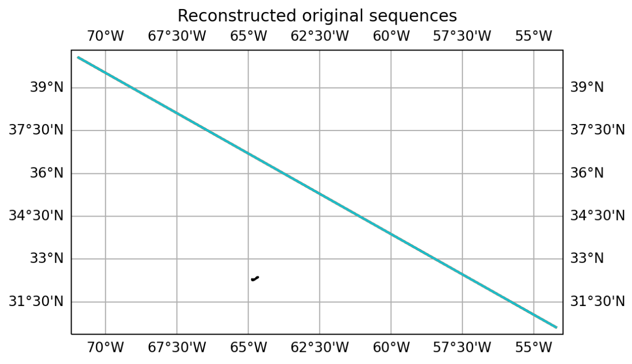
After our outlier detection experiments, we moved onto generating samples. While our eventual goal is *de novo* generation from noise sampled from $p(z)$, as a first step we attempted to reconstruct the 72 sequences in our dataset. As shown in Figure 4, the VRAE is not able to recover the original sequences nor even create unique sequences for each embedding. The latter is an example of *posterior collapse* (Goyal et al. (2017)). This occurs when the posterior approximation provides too weak or noisy signal. As a result, the decoder may learn to ignore the learned latent representation, z, and rely solely on the autoregressive properties of the data, x, for generation, causing x and z to be independent. In our case, the weak signal from the latent variable causes the decoder to minimize reconstruction loss by generating the same generic output $\hat{x}$ that is a crude representation of all of the training data.

Much of our time in this project was spent trying to ameliorate this posterior collapse. What follows is a brief summary two attempted fixes, the theory behind them, and the results.

First, we tried to scaling the data. Since all trajectories are in a small region of the ocean, their latitude and longitude measurements have quite low variance. Quantitatively, the average standard deviation in latitude for a trajectory in the dataset is 0.235 and in longitude is 0.132. We thus hypothesized that increasing the variance by scaling the data linearly by $k = 10$ in the sequences would improve reconstruction. However, the model still exhibits posterior collapse, even if the sequences are slightly more coherent. The results shown in 4 are from a model trained with scaled data.



(a) Original data



(b) Reconstructed data

Figure 4: The VRAE is unable to recover the original 72 sequences from their encodings. Note the axes: the reconstructed sequences do not even reside in the same space. In addition, the VRAE exhibits posterior collapse: all 72 reconstructed sequences are more or less the same.

Next, we focused on weighting the two terms in the ELBO. As shown in Figure 5, the reconstruction loss is orders of magnitude larger than the KL divergence over the course of training. Since our model struggles with reconstruction, one approach would be a simple β-VAE with loss function

$$ELBO_\beta(x^{(i)}; \phi, \theta) = -\beta D_{KL}(q_\phi(z|x^{(i)})||p(z)) + \mathrm{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)].$$

as described in Higgins et al. (2017). A larger weight on β has been empricially found to lead to better samples by forcing representations to fit $p(z)$, but a large β did not lead to better results with our data.

Another approach is KL annealing, in which we in effect schedule β as a function of the epoch t such that

$$ELBO_{\text{KL annealing}}(x^{(i)}; \phi, \theta, t) = -\beta(t) D_{KL}(q_\phi(z|x^{(i)})||p(z)) + \mathrm{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)].$$

In contrast to the β-VAE, in which $\beta > 1$ typically leads to better results, KL cost annealing begins training with $\beta = 0$ such that the loss is equivalent to that of a vanilla autoencoder and then gradually increases β to 1 to recover the true ELBO. This incents the model to first learn to reconstruct the data and then gradually add further constraints to allow for sampling. For an example of this technique, see Bowman et al. (2016). While theoretically well motivated in our case, since the model struggles

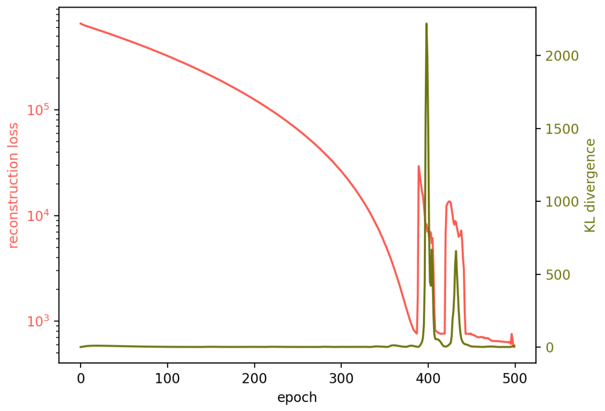


Figure 5: Evolution of two components of VRAE loss over time. Observe that the reconstruction loss is much larger and that the model appears to converge despite the lack of performance as shown in 4

with reconstruction, KL annealing did not bear fruit. Figures 4 and 5 were trained with a linear annealing function $\beta(t) = \frac{t}{T}$ where T is the total number of epochs. We also tried a sigmoid annealing function, like the one proposed in Bowman et al. (2016), to no avail.

Thus, the VRAE performed poorly despite many changes in the architecture, hyperparameters, and cost function. While our dataset was quite small, the inability of the model to even overfit on training data indicates a lack of model capacity. If given more time, we would attempt to increase the capacity by trying out different encoder/decoder modules, optimizers, and perhaps building out our own *tabula rasa* implementation of the VRAE.

5.2 Autoregressive approach

5.2.1 Reconstruction

For the training task described in Section 4.2, the LSTM-based autoregressive approach achieves an validation MSE of 0.023 after training, which suggests that the model achieves good generalization on unseen data.

Below is a plot of a ground truth trajectory from the validation set and the autoregressive predicted trajectory, which uses the first M time steps of the ground truth trajectory to initialize the conditional range window.

Compared to the VRAE, the autoregressive model generates reconstructions that are more realistic, in terms of modeling reasonable glider dynamics compared to the ground truth glider trajectories, and that are more varied, meaning the prediction distribution is closer to the ground truth glider trajectory distribution.

Even though the model is able to optimize for the training task quite well, it is worth noting that the generated reconstructions, like the one in Figure 6, do not exactly match the ground truth trajectories from which the conditional range window is initialized, since the training objective optimizes for prediction of a short prediction range window of length N, not necessarily for prediction of the full sequence length.

It has been noted in the literature that language generation models can suffer from similar issues, in the sense that these models are trained on “positive” examples that model the proper dynamics of the environment (i.e., sentences with good grammar and syntax), so when the model makes a prediction that takes it off the path of the ground truth example, it is difficult for the model to self-correct (see Xie et al. (2018)). In our case, this issue is likely exacerbated by the deterministic output predictions, so when the model predicts one data point off the path of the ground truth glider trajectory, it may begin to resemble a different group of glider trajectory in the training set and reconstruct to look like

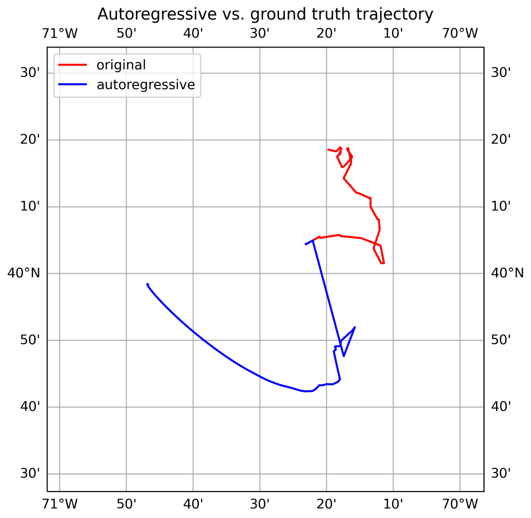


Figure 6: Autoregressive reconstruction vs. ground truth trajectory for one sequence in the test set. Note that the sequences are identical for the first 10 time steps in the center of the region plotted, corresponding to the seed input steps to the autoregressive model, but diverge afterwards. However, the autoregressive reconstruction is relatively plausible, especially when compared to the posterior collapse evidenced in Figure 4.

the mean of that group, instead of correcting back to the ground truth glider trajectory, since a glider trajectory that looks like the ground truth glider trajectory with this incorrect predicted data point would not be in the training data.

Several techniques introduced for mitigating this issue in natural language generation could be useful in this context as well. One way to mitigate this problem might be to using noising data augmentation techniques to introduce errors into the training data, to help the model learn how to self-correct as described in Xie et al. (2018). In an encoder-decoder architecture with probabilistic modeling over outputs, using beam search, as suggested in Cho et al. (2014), would allow the model to consider the maximal probability over a complete sample, instead of greedily maximizing at each time step, which could also improve reconstruction.

6 Conclusion

In sum, this paper compares a LSTM-based autoregressive approach to a VRAE on a dataset composed of ocean glider trajectories. We found that the autoregressive model far outperformed the VRAE in reconstructing sequences. Concretely, the autoregressive model is able to better reconstruct a given sequence from the first 10 time steps than the VRAE is when given that sequence's 20-dimensional latent representation. This even holds when the sequence of interest is outside the set of sequences used to train the autoregressive model. Despite its shortcomings, the VRAE is still somewhat useful, as it is able to detect outlier sequences via the latent space representations.

These results show that autoregressive models, despite their relative simplicity, still provide compelling results in a sequential setting. While it may be the case that there exists a specific VRAE initialization that outperforms the autoregressive model, the robustness and simplicity of the autoregressive approach makes it an attractive option. As for the VRAE, we ran into model capacity issues in terms of reconstruction, though its ability to learn embeddings represents a unique advantage of the approach.

Sequential data admits many different approaches, and the two considered in this paper just scratch the surface. We hope that this report can help practitioners understand the trade-offs of autoregressive and VAE-based models and guide the development of new ones that combine the ability to learn latent space representations with simple initialization and coherent sequence generation.

References

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating Sentences from a Continuous Space. *arXiv:1511.06349 [cs]*, May 2016. URL http://arxiv.org/abs/1511.06349. arXiv: 1511.06349.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A Recurrent Latent Variable Model for Sequential Data. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/hash/b618c3210e934362ac261db280128c22-Abstract.html.

Fabius, O. and van Amersfoort, J. R. Variational Recurrent Auto-Encoders. *arXiv:1412.6581 [cs, stat]*, June 2015. URL http://arxiv.org/abs/1412.6581. arXiv: 1412.6581.

Goyal, A., Sordoni, A., Côté, M.-A., Ke, N. R., and Bengio, Y. Z-forcing: Training stochastic recurrent networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6716–6726, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK. pp. 22, 2017.

Inanc, T., Shadden, S., and Marsden, J. Optimal trajectory generation in ocean flows. In *Proceedings of the 2005, American Control Conference, 2005.*, pp. 674–679, Portland, OR, USA, 2005. IEEE. ISBN 978-0-7803-9098-0. doi: 10.1109/ACC.2005.1470035. URL http://ieeexplore.ieee.org/document/1470035/.

Murray, B. and Perera, L. P. An AIS-based deep learning framework for regional ship behavior prediction. *Reliability Engineering & System Safety*, 215:107819, November 2021. ISSN 0951-8320. doi: 10.1016/j.ress.2021.107819. URL https://www.sciencedirect.com/science/article/pii/S0951832021003409.

Oord, A. v. d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional Image Generation with PixelCNN Decoders. *arXiv:1606.05328 [cs]*, June 2016. URL http://arxiv.org/abs/1606.05328. arXiv: 1606.05328.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.07.001. URL https://www.sciencedirect.com/science/article/pii/S0169207019301888.

Xie, Z., Genthial, G., Xie, S., Ng, A., and Jurafsky, D. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 619–628, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1057. URL https://aclanthology.org/N18-1057.