MUSIC RECONSTRUCTION USING DYNAMIC EPISODIC MEMORY

Vidya Rangasayee

Center for Computer Research in Music and Acoustics (CCRMA), Stanford University
vidya@ccrma.stanford.edu

Chaofei Fan

Dept of Computer Science
Stanford University
stfan@stanford.edu

ABSTRACT

In this paper we present a novel approach to music generation that uses episodic memory to reconstruct or denoise a partial input from memory. In particular, we are using an implementation of Sparse Distributed Memory called Dynamic Kanerva Machine (DKM). DKM is a probabilistic generative model that is very effective in denoising. Our model was trained on the Bach Chorales dataset and tested on episodes for which the second half is masked. We show that DKM can display behaviors similar to human episodic memory (e.g., pattern storage and retrieval given a noisy or partial cue), with the model achieving good results in denoising and reconstruction with partial cues. Future work involves more accurate modeling of human episodic memory and using it for music generation.

1. INTRODUCTION

Humans can remember many episodes of music. Episodic memory is an autobiographical storage of our experiences [1], whereby the experiences are sliced into short episodes and stored in episodic memory. Episodic memory enables us to travel back in time to relive a previous experience, such as a moment of experiencing a Bach concert, together with the immediate moment. You may hear the same Bach tune at another time and your mind brings you back to the previous concert. Episodic memory forms rapidly, can be retrieved given a partial cue, and is presented in the order of occurrence. These properties allow us to make better decisions when a situation similar to the past recurs.

There are many proposed computational models of memory. Hopfield network is a content addressable auto-associative memory network [2]. It can retrieve stored patterns given a partial or noisy input but has limited capacity. Differentiable Neural Computer [3] uses a neural network to learn how to read and write to a memory. It has achieved remarkable performance in some algorithmic tasks, but also suffers from a capacity problem [4]. Moreover, to learn a new task, the neural network controller needs to be retrained. Sparse distributed memory (SDM) [5] has a much larger capacity than Hopfield network. However, the input to the model is limited to randomly distributed binary vectors. Dynamic Kanerva Machine (DKM) is a probabilistic variant of SDM. Memory in DKM becomes a global random matrix that can be read and written using Bayesian inference. Unlike DNC, which uses a fixed trained neural network as controller for memory addressing, DKM's addressing mechanism is dynamic in the sense that it searches for the best address at run-time. DKM can also store more data than its memory size given that the data share some similarities.

In this paper, we describe a novel approach to music generation using DKM as a memory augmented generative model. We describe the use of sparse memory representation to improve the long-term dependencies. We begin with a discussion of the general DKM model in Section 2 and a particular implementation we use in Section 3. The experimental setup and results are discussed in Section 4. Finally the interpretation of these results and future research are presented in Section 5.

2. DYNAMIC KANERVA MACHINE

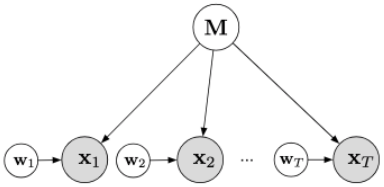

Figure 1: *Probabilistic graphical model of DKM [4].*

Dynamic Kanerva Memory (DKM) is a probabilistic generative model that views memory as a global latent variable. The memory $\mathbf{M}$ is implemented as a column independent $K \times C$ random matrix $p(\mathbf{M}; \mathbf{R}, \mathbf{U})$ with mean $\mathbf{R}$ and covariance $\mathbf{U}$. Given T episodes of experience $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_T}$, the conditional distribution is:

$$p(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_1}|\mathbf{M}) = \prod_{t=1}^{T} p(\mathbf{x_t}|\mathbf{M})$$

.

The underlying assumption is that episodes are exchangeable, i.e., $\mathbf{x_t}$ can be retrieved regardless of the order it was stored in memory. Reading from memory is done via a random Gaussian addressing variable $p(\mathbf{w}) = \mathcal{N}(\mu_w, \sigma_w^2)$,

which linearly combines rows of $\mathbf{M}$:

$$\mathbf{x} = \sum_{k=1}^{K} \mathbf{w}(k)\mathbf{M}(k) + \xi \tag{1}$$

where k is index of rows of $\mathbf{M}$ and ξ is observation noise added to ensure proper model's well-defined likelihood. $\mathbf{w}$ is chosen such that the readout of $\mathbf{x}$ from $\mathbf{M}$ is most likely. Since both $\mathbf{w}$ and $\mathbf{M}$ are Gaussian distributions, $p(\mathbf{x}|\mathbf{w}, \mathbf{M})$ can be solved analytically. Unlike previous models where the addressing is done via a fixed trained neural network [3, 6], DKM's addressing is dynamic so that it can adapt to different data distributions at test time. Memory writing is done via Bayesian inference, i.e., to calculate the posterior $p(\mathbf{M}|\mathbf{x}_{\leqslant T})$. Since exact inference is intractable, a sequential approximate inference is used.

Putting everything together, the model maximizes the log likelihood of data:

$$\begin{aligned} \ln p(\mathbf{x} \leqslant T) = \mathcal{L}_T \\ + \sum_{t=1}^{T} \left\langle \mathrm{D_{KL}}\left(q\left(\mathbf{w}_t\right) \| p\left(\mathbf{w}_t|\mathbf{x}_t, \mathbf{M}\right)\right)\right\rangle_{q(\mathbf{M})} \\ + \mathrm{D_{KL}}\left(q(\mathbf{M}) \| p\left(\mathbf{M}|\mathbf{x}_{\leqslant T}\right)\right) \end{aligned} \tag{2}$$

with its variational lower-bound:

$$\begin{aligned} \mathcal{L}_T = \\ \sum_{t=1}^{T} \Big(\left\langle \ln p\left(\mathbf{x}_t|\mathbf{w}_t, \mathbf{M}\right)\right\rangle_{q(\mathbf{w}_t)q(\mathbf{M})} \\ - \mathrm{D_{KL}}\left(q\left(\mathbf{w}_t\right) \| p\left(\mathbf{w}_t\right)\right) \\ - \mathrm{D_{KL}}(q(\mathbf{M}) \| p(\mathbf{M})) \end{aligned} \tag{3}$$

The learning algorithm is analogous to expectation maximization (EM): it tightens the bound by minimizing the KL-divergences (relative entropy) in Equation 2 and maximizes the lower-bound $\mathcal{L}_T$ in Equation 3 using gradient descent.

The trained model has attractor dynamics that can be used to denoise input. Given a noised input $\mathbf{x}_q$, the denoise operation is defined as:

$$\begin{aligned} \mathbf{x}_{denoised} &= \mathrm{argmax}_{\hat{\mathbf{x}}}\, p\left(\hat{\mathbf{x}}|\mathbf{x}_q, \mathbf{M}\right) \\ &= p\left(\mathbf{x}|\mathbf{w}^*, \mathbf{M}\right)|_{\mathbf{w}^* = \mathrm{argmax}_{\mathbf{w}}\, p(\mathbf{w}|\mathbf{x}_q, \mathbf{M})} \end{aligned} \tag{4}$$

Intuitively, it uses the maximum a posteriori (MAP) estimate $\mathbf{w}^*$ to reconstruct $\mathbf{x}$ and then take the mode of the resulting distribution. This can be done iteratively until convergence:

$$\mathbf{x}_n = \mathrm{argmax}_{\hat{\mathbf{x}}}\, p\left(\hat{\mathbf{x}}|\mathbf{x}_{n-1}, \mathbf{M}\right) \tag{5}$$

3. MODEL ARCHITECTURE

In this work, we use a symbolic music representation. Each episode is represented as $\mathbf{x} \in \mathbb{R}^{S \times I}$, where S is the number of time-steps and I is the number of instruments. We set $S = 32$ and $I = 4$ so an episode can be flattened into vector $\mathbf{x} \in \mathbb{R}^{128 \times 1}$, which is small enough to store in memory directly. We thus omit encoder and decoder. The only trainable parameters in our model address the variable's variance $\sigma_{\mathbf{w}}$ and random memory matrix's mean $\mathbf{R}$ and covariance $\mathbf{U}$. Training these variables can be seen as adapting the memory prior to the training data distribution so that at test time the posterior distribution of the memory matrix is close to the prior.

3.1. Modified Dynamic Kanerva Machine

The denoising procedure of DKM works well if noise is within a certain threshold. However, in our case, we would like the memory to complete an episode when parts of it are masked out. This implies a much larger noise level compared to noise used in [4]. As a result, the memory can barely complete an episode even if we mask out just the last time step in an episode. To solve this problem, we make the following observations. When finding the MAP estimate of $\mathbf{w}$, it is essentially solving a least-squares problem:

$$\mathrm{argmin}_{\mu_{\mathbf{w}}} \frac{\left\|\mathbf{x}_q - \mathbf{M}^\top \mu_{\mathbf{w}}\right\|^2}{2\sigma_\xi^2} + \frac{1}{2}\left\|\mu_{\mathbf{w}}\right\|^2 \tag{6}$$

where σ_ξ^2 is the variance of observation noise. If parts of $\mathbf{x}_q$ are masked out (e.g., setting some notes to NaN), those masked out parts should not be included when solving the least-squares problem. In other words, the original least-squares problem becomes a least-squares problem with missing data. And the way to solve this problem is to also mask out corresponding parts of $\mathbf{M}$.

The Cholesky decomposition used by the original DKM implementation does not give accurate results for completion due to the missing data. In all of our denoising and completion experiments, we use the orthogonal decomposition version of $matrix\_solve\_ls$, which is a slower but more accurate way to achieve better results.

4. EXPERIMENT AND RESULTS

4.1. Dataset and preprocessing

For training we use the Bach chorales dataset published by Boulanger-Lewandowski, Bengio and Vincent [7]. The dataset consists of 382 four-part harmonized tracks split by instruments. Each track is represented as a matrix of shape $S \times I$ where S is the length of track after quantization and I is the number of voices or instruments. The dataset contains 46 pitches and 4 voices. Each time step represents a 16th note and contains exactly 4 numbers; one pitch for each of the SATB (Soprano-Alto-Tenor-Bass) voices. If a voice is silent at a given time step, its pitch is NaN. Each episode is generated using a 32 time-step slice. Episodes

are uniformly sampled from the training set. All data are normalized to $[-1, 1]$ before feeding into memory. For reconstruction, the observation noise ξ is a zero mean Guassian with $\sigma_\xi = 0.2$. The memory size is set to $K = 32$, $C = 128$. Episode length is $T = 32$. We used Adam optimizer [8] with learning rate 0.001 and batch size 32. All of these values are hyperparameters and have been chosen to provide the best results so far. The model is trained to convergence after 3×10^4 iterations. There are two tasks that the model is trained for - denoising and completion. These tasks are explained in Sections 4.3 and 4.4.

4.2. Data Log-likelihood

Since our model is a generative model, we can evaluate its log-likelihood on the test set. Following [4], we evaluate the negative conditional evidence lower-bound (ELBO). It is defined as $-\mathcal{L}_T - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{M})\|p(\mathbf{M}))$. The evaluation is done by first storing T episodes into memory and retrieving episodes using the weight samples that are generated when storing. The retrieved episodes' negative conditional ELBO are calculated and averaged per time-step per instrument, which is 0.76 ± 0.002. The log-likelihood shows that our model learns the statistical structure of the data. We can compress and transmit each note without loss, using 0.76 bits, given that both sides of communication have access to the populated memory. The results of the reconstruction are published online[1].

4.3. Denoising

For denoising, we first store T episodes into memory. Then, for each original episode $\mathbf{x}_t$, we add random Gaussian noise $\mathcal{N}(0, 0.01)$ to the normalized $\mathbf{x}_t$. This is equivalent to randomly shifting notes by one or two semitones in $\mathbf{x}_t$. Noised episodes are used as cues to retrieve the original ones (eq. 4 and eq. 5). We run the denoise operation for 10 iterations. The denoising procedure is visualized in Figure 2. The procedure converges after around 3 iterations. The denoised episodes are almost the same as the original ones. Readers can listen to the denoised samples here[2].

4.4. Completion

The original DKM fails at completion task [5]. The modified DKM that we propose, however, can complete an episode. We mask out the second half of each episode (i.e., set s from 16 to 32 to NaN in each episode). The masked-out episodes are completed for 10 iterations similar to denoising. The completion procedure is visualized in Figure 3. Most of the structures of the original episodes are recovered. Some notes are slightly off usually by no more than a whole step. Readers can listen to sample completed episodes here[3]. If we increase the mask size to the last 3 quarters of each episode (i.e., set s from 8 to 32 to NaN in each episode), the completed excerpt is no longer similar to the original. This is probably because there are not enough constraints to solve the least-squares problem, resulting in a sub-optimal reading from memory.

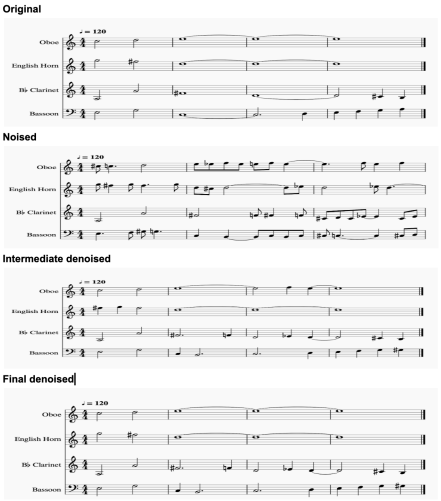

Figure 2: *Stages of denoising. From top to bottom: Original score, Noised input to the model, Intermediate denoised, Final output.*

5. DISCUSSION

The results above show that our model is very accurate in recovering the original by denoising and reconstructing. Example denoising and completion are shown in Figures 2 and 4. However, if the loss of original data is $> 50\%$, the reconstruction yields poor results as mentioned in the previous section. Another limitation of this model is its capacity. Since our memory has size 32×128, The model can remember 32 episodes, each of which is size 128. In [4] DKM has been demonstrated to be able to store more patterns than it has been trained on. Our implementation does not seem to have the ability to store more episodes than the training set. This discrepancy may be caused by two reasons. First, the original DKM's capacity experiment increases episode length by adding episodes of similar patterns, but our musical episodes are different. The second reason is that unlike our model, the original DKM uses a trained encoder and decoder which may cluster similar patterns into similar codes.

6. CONCLUSION AND FUTURE WORK

Given a melodic cue, humans can recall the corresponding melody and the context in which it was heard. With episodic

[1] `http://bit.ly/wimp2019recons`
[2] `http://bit.ly/wimp2019denoise`
[3] `http://bit.ly/wimp2019complet`

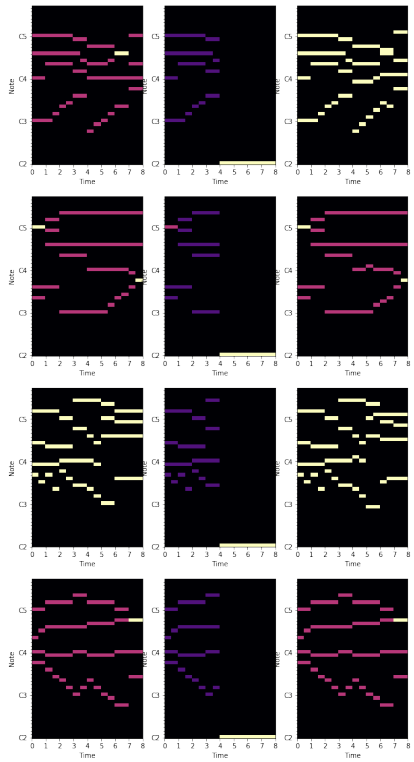

Figure 3: *Episode completion visualization, with original episode (left column), half-masked episode (middle column; the masked notes are set to* 0 *instead of NaN for visulaization), and completed episode (right column).*

memory we can experience the current moment and the past at the same time. Our DKM memory model shows some similarities with human musical episodic memory. It can store and recall musical episodes, and it can complete an episode given a partial cue. DKM's perfect recall ability is impossible for a human, since each note in an episode has the same importance to DKM. Humans on the other hand recall musical episodes differently—we tend to remember melody and lyrics much more than the harmony, timbre or expression [9]. Such an augmented human-like model, which accounts for such differences in recall across components of the memory, could become a very powerful generative tool for music.

7. REFERENCES

[1] E. Tulving, "Episodic memory: From mind to brain," *Annual review of psychology*, vol. 53, no. 1, pp. 1–25, 2002.

[2] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

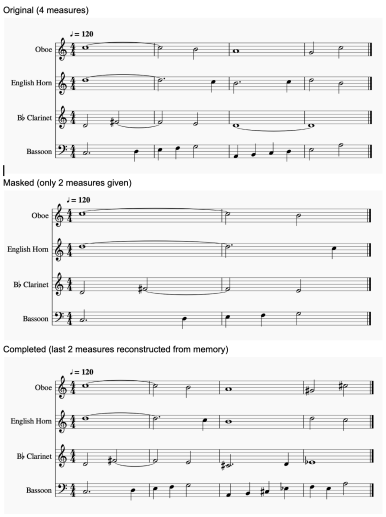

Figure 4: *Reonstruction using partial input. From top to bottom: Original score with 4 measures, Partial input with 2 measures, Final output with 4 measures*

[3] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[4] Y. Wu, G. Wayne, K. Gregor, and T. Lillicrap, "Learning Attractor Dynamics for Generative Memory," no. Nips, 2018.

[5] P. Kanerva, *Sparse distributed memory*. MIT press, 1988.

[6] Y. Wu, G. Wayne, A. Graves, and T. Lillicrap, "The kanerva machine: A generative distributed memory," *arXiv preprint arXiv:1804.01756*, 2018.

[7] P. Boulanger-Lewandowski, Nicolas; Bengio, Yoshua; Vincent, "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription Nicolas," *A Priori*, no. Cd, pp. 1–229, 2009.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] F. Bailes, "The prevalence and nature of imagined music in the everyday lives of music students," *Psychology of Music*, vol. 35, no. 4, pp. 555–570, 2007.