Small Language Modelling to Generate Playlists from Song Sequences

Mark Green
Data Science Program
Indiana University
Bloomington, IN 47405
margree@iu.edu

Sahil Dhingra
Data Science Program
Indiana University
Bloomington, IN 47405
sahdhin@iu.edu

Rahul Jain
Data Science Program
Indiana University
Bloomington, IN 47405
rjdharmc@iu.edu

Abstract

Large Language Models (LLMs) such as GPT and BERT have led to revolutionary advances in language applications such as machine translation, sentiment analysis, and text generation. However, these models rely on enormous datasets of texts derived from spoken language which are prohibitively resource-intensive to train. Although pre-trained models solve this to a degree, their scope is limited to the pre-trained spoken language so they cannot be applied to more niche language modeling tasks. This paper explores the application of neural networks for small language modelling where the dataset is small in both size and vocabulary and is composed of non-spoken language "words". Specifically, this approach explores the music of the band Phish, leveraging historical song appearances and setlists from their shows to model the band's performance tendencies. By decoding intricate patterns from their extensive performance history using a targeted small language model, this approach offers a unique solution for personalized music recommendations. The research aims to enhance the user experience for dedicated fans with more artist-centric music recommendation systems and to test the performance of neural network language models for small, niche applications.

1 Introduction

As of 2022, music streaming is an $18 Billion dollar industry.[1] Playlist generation is a key differentiator for streaming platforms and previous works have used reinforcement learning [1], convolutional neural networks (CNNs) [2], and recurrent neural networks (RNNs) [3] as the generative mechanisms. These works have generally spanned a broad scope of artists and genres to generate playlists composed of multiple bands or band-specific playlists ordered on an arbitrary learned feature. While this may work well for a generic music listener, this approach may not be effective for "dedicated fans" - a subset of users whose listening habits tend to be more focused around an individual band and their compositional tendencies.

One such type of "dedicated fan" are those followers of Jam bands. Since the 1960s, Jam bands have captivated audiences with their improvisational Rock and Roll style, reminiscent of Jazz. Notably, these groups do not pre-plan setlists for their live performances and instead weave together a pseudo-random selection of songs such that no two shows are alike. The choice of songs for each show is partially influenced by spontaneous notions such as crowd energy or a band member's mood, but it is also influenced by constraints like temporal frequency and positional sequencing. These shows also generally follow a standard pattern composed of two sets separated by an intermission and a brief encore composed of just a song or two. Beyond the music itself, the sequencing of songs in

[1] https://www.weforum.org/agenda/2023/03/charted-the-impact-of-streaming-on-the-music-industry/

Preprint. Under review.

shows is reminiscent of natural language – song names are analogous to words, setlists to sentences, show starts and stops to parts-of-speech – with the repertoire of each band forming its own unique language full of syntactical intricacies.

We purport that constructing small language models from the song sequences of live performances to generate playlists for a given band could increase the engagement of "dedicated fan" type users by generating playlists that more closely mimic a band's musical tendencies than a more generalized algorithm. In this report, we explore the music of the Jam band Phish for this purpose – who has not only a large catalog of live shows and songs, but also a very dedicated fanbase. A previous endeavor[2] employed a Long Short-Term Memory (LSTM) model with Word2Vec [4] algorithm for token embeddings to achieve a approxmately 21% accuracy for next song prediction given a sequence of Phish songs. This paper builds upon this work by recreating it four years later and exploring the prediction task with alternate embedding methods, and RNN model architectures, and attention model architectures.

2 Related Work

The first stage of experimentation was an attempt to reproduce the original experiment[3] from November 2019 with the latest setlist data from December 2023. Experiments were run for multiple model architectures, both with and without Word2Vec embedding. For both architectures, a grid search was performed with song sequence lengths of 50 and 150 with varying LSTM unit sizes and dropout rates. Although the original experiment reported the best next song accuracy of approximately 21.8%, We were unable to reproduce these results and none of the 16 models attempted in the grid search produced satisfactory results with Dec 2023 dataset. The model without Word2Vec embedding suffered from an exploding gradient problem, and the validation accuracy quickly approached 0 after just a few training epochs. Models with Word2Vec embedding did marginally better, but would also become saturated at low validation accuracy within 4-5 training epochs. These results indicate the model used in the original experiment was overfitting the data from Nov 2019, and hence did not generalize well to the latest data. The highest validation accuracies observed were approximately 12% for models without Word2Vec and 10% for the models with Word2Vec embedding - both significantly less than the originally reported accuracy of 21.8%. This highlights the difficulty of using only the unidirectional songs sequences to predict new song sequences, and suggests more context and data is needed to obtain a more meaningful inference. This conclusion is in line with the original Author's assessment as well, and we attempt to address this in further experiments.

3 Data Preparation

The `https://phish.net` fan forum curates an API[4] cataloguing every setlist of songs for every show the band has played. First queried from this endpoint were song-related details encompassing song name, ID, artist, play frequency, last performance date, and debut date. Further, we meticulously filter shows attributed exclusively to Phish, forming a chronological compendium of their extensive concert history. Next queried were setlist data such as showdate, set number, song positions, ID, name, transition markers, song gaps since the last performance, and jam categorization. Incomplete information were eliminated to ensure the dataset's integrity, resulting in a comprehensive dataframe. This curated dataset, exclusively featuring "full" shows with two sets and an encore, serves as the foundation for the predictive modeling task and is saved as a csv file[5] in the project's code repository.

In the initial exploratory data analysis of the Phish concert dataset, it was revealed the frequency of songs played resembled a power-law distribution whereby almost half of the songs were only played once or twice throughout the band's entire history. To address the challenge posed by rarely performed songs, we consolidated these one-off and two-off songs into a unified song category termed the "wildcard". For prediction, this wildcard song is treated like any other song Phish plays but it represents an intractable uncertainty to relax the problem for new song debuts or unpredictably random cover songs.

[2] `https://towardsdatascience.com/predicting-what-song-phish-will-play-next-with-deep-learning-947ccce3824`

[3] `https://github.com/andrewrreed/phish-setlist-modeling`

[4] `https://docs.phish.net/`

[5] `https://github.iu.edu/rjdharmc/dlProject/blob/main/data/allphishsets.csv`

Furthermore, for predictive modeling, we systematically aggregated songs into sequences by grouping them based on show date and set number. These sequences were then concatenated into a unified string with additional separators in the string to indicate a song's position in the show. For example, the beginning of every string starts with 'set-1'. Then after all the 1st set songs appear in the string, the 'set-2' separator appears, then the 'set-e' separator before the encore, and finally the 'eos' separator at the end of the string after all the songs.

These preprocessing steps refined the dataset and established a structured framework for developing models focused on predicting Phish setlists. Following this modification and the original data cleaning, the dataset is composed of 1,550 shows, 33,533 total songs played, with 482 unique songs and separators.

3.1 Interpretting Song Sequences as Language

In order to mathematically interpret the song names as a language, it was necessary to tokenize the song names and separators into numerical values suitable for input into a neural network model. This was predominantly done using unigram tokens, or integer representations mapped to each full song name. In this way, each setlist is similarly decomposed into a list of integer values.

Recall, the aim of this project is to generate song sequences (IE playlists) which are representative of the style in which the band plays. This problem can be framed in terms of language modelling in either a *sequence-to-token* perspective, using a sequence to predict the next token, or a *sequence-to-sequence* perspective, using a sequence to predict the next sequence. Since the models will have to either predict the next n^{th} song or the next n songs, multiple data splitting techniques for input and output data were applied to evaluate the model performance under both of these paradigms. These splitting mechanisms are described in the following section, specific for each model.

4 Models

Since the language modelling is trying to find good features rooted in the sequential aspect of the data, both RNN LSTM and attention architectures were modeled so as to focus on the sequential information to make better predictions. For RNN experiments, we used both uni-directional and bi-directional models were attempted. For the attention experiments, self-attention and cross-attention were attempted. These model architectures are discussed in more detail below.

4.1 Model 1: Unidirectional LSTM with N-dimensional Embedding

The initial model architecture first includes a trainable embedding layer to represent songs numerically as N-dimensional vectors which are then passed to two Long Short-Term Memory (LSTM) layers for capturing sequential dependencies. Dropout is used for regularization and a final Dense layer with softmax activation is used for multi-class classification. The model is trained on variable-length input sequences of songs, and the embedding layer allows it to learn meaningful representations for each song. The use of masking and padding ensures proper handling of sequences with varying lengths.

The training sequences for this model is designed to predict either the next n^{th} or n songs, and so to assess the model's performance the dataset was partitioned in three distinct ways, each tailored to a specific model evaluation strategy.

Row-Level Splitting A standard 80/20 split was applied at the row level, allocating 80% of the data for training and reserving 20% for testing. This method ensures an adequate distribution for both training and testing.

Time-Based Splitting In this approach, the dataset was chronologically split into training and testing sets using a predetermined date, e.g., '2015-12-31'. By reflecting the evolving nature of song sequences over time, this strategy enhances the model's ability to capture temporal dependencies.

Sequential Training Data The training data was structured sequentially as [n], [n, n+1], [n, n+1, n+2], and so on, with corresponding labels as [n+1], [n+2], [n+3], and so forth. Emphasizing continuity, this method enables the model to learn and predict the sequential progression of song sequences.

4.2 Model 2: Bidirectional LSTM on N-grams with Adjacency Matrix Embedding

The next LSTM model architecture that was tried uses an *N-gram* sequencing approach. Although similar to other sequence-to-token approaches, the N-gram tokenization splits a given setlist into as many vectors as the setlist contains songs and set markers. Each of these vectors is composed of a padded sequence of 0s and an increasing number of songs and set markers from the setlist. For example: the first vector is composed of just "set-1", the next also contains the 1st song, and the next also contains the first and second song – etc. In this way, each sequence is composed of N tokens (the N-grams) which are used to predict the next target token. This technique focuses on the LSTM's ability to predict song sequences from positionings within a given show.

In addition to this N-gram splitting technique, this method embeds the tokens using their co-occurrence matrix. This co-occurrence matrix is given by the LaPlace-smoothed adjacency matrix of the directed graph created by overlaying all the setlists as walks. In this graph, each song is a node, and each song transition contributes to a directed weighted edge. This embedding layer is not trainable and encodes long-term transition frequency data to reperesent likelihoods of choosing the next song.

The model is constructed with the special embedding layer, followed by two Bi-directional LSTM layers and a feed forward layer is used to compute the final output, with softmax activation. Although the setlists are unidirectional sequences, the bi-directional LSTM method looks for patterns more wholistically for better latent feature interpretation.

4.3 Model 3: Self-Attention Model with N-dimensional + Positional Embedding

In contrast to the sequence-to-token models attempted with the LSTM architectures, sequence-to-sequence models are attempted with multi-head attention architectures. These models process the entire song sequence for a setlist into offset pairs to form the inputs and target outputs, where the original sequence is broken down into [0,...,n-1] tokens for the input sequence and [1,...,n] tokens for the output sequence. The sequences can be any size, although sequence lengths of 125 songs performed the best. The token sequences are encoded by an embedding layer composed of the sum of a learned N-dimensional vector embedding layer and a positional embedding layer. The positional embedding layer applies sine and cosine functions for fixed positional encodings. This is the same mechanism as is used in the original attention model [5].

The model architecture itself is similar to the decoder-only transformer used in the GPT model[6, 7]. It consists of a causal self-attention layer using masked multi-head attention to attend to the sequencing. This layer is added and normalized, then passed to a feed forward layer to form the Transformer block. These transformer blocks are then layered sequentially and an additional feed forward layer computes the final output, then activated by softmax.

4.4 Model 4: Cross-Attention Model with N-dimendsional + Positional Embedding

This final model architecture is setup similarly to the original transformer model proposed in "Attention is All You Need" [5]. Here, a sequence of songs is concatenated from the previous 5 or more setlists to be used as the context for the global self-attention layer, which is passed to the decoder via the cross-attention layer. Inputs to be predicted upon are first sent through the causal self-attention layer as part of the decoder, before then being sent through the cross-attention layer for the given context sequence to form the transformer block. These transformer blocks are similarly layered and fed forward to compute the final output, activated by softmax to generate the multi-class probability classification.

5 Results

Summary of the Results.

6 Discussion

Discussion of what was accomplished.

Table 1: Summary of Model Training Results

Model	Best Next-Song Validation Accuracy	Generated Sequence Quality
2019 Model Recreation	∼12.5%	NA[6]
Model 1	∼18%	Moderate
Model 2	19.5%	Poor
Model 3	18.6%	Good
Model 4	15.9%	Moderate

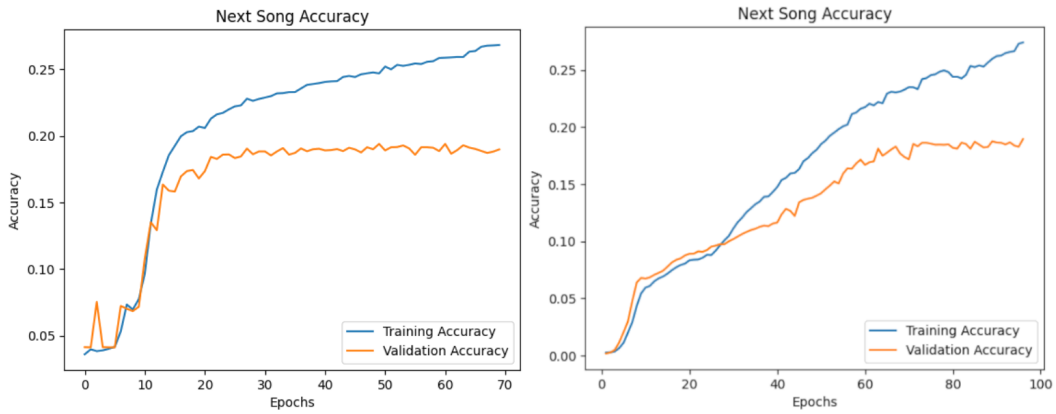

Figure 1: Validation accuracies during training for the bi-directional LSTM model (model 2; *left*) and for the self-attention model (model 3; *right*)

7 Conclusion

Some concluding words.

The code we used to acquire and process the dataset, train and evaluate our models is available at `https://github.iu.edu/rjdharmc/dlProject`. Note that access to this codebase will require an active account on the IU Enterprise Github instance.

References

[1] F. Tomasi, J. Cauteruccio, S. Kanoria, K. Ciosek, M. Rinaldi, and Z. Dai, "Automatic Music Playlist Generation via Simulation-based Reinforcement Learning," Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, Aug. 04, 2023. doi: 10.1145/3580305.3599777.

[2] R. T. Irene, C. Borrelli, M. Zanoni, M. Buccoli, and A. Sarti, "Automatic playlist generation using Convolutional Neural Networks and Recurrent Neural Networks," 2019 27th European Signal Processing Conference (EUSIPCO). IEEE, Sep. 2019. doi: 10.23919/eusipco.2019.8903002.

[3] K. Choi, G. Fazekas, and M. Sandler, "Towards Playlist Generation Algorithms Using RNNs Trained on Within-Track Transitions." arXiv, 2016. doi: 10.48550/ARXIV.1606.02096.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space." arXiv, 2013. doi: 10.48550/ARXIV.1301.3781.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, "Attention Is All You Need." arXiv, 2017. doi: 10.48550/ARXIV.1706.03762.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. "Language models are unsupervised multitask learners". 2019.

[7] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever. "Improving Language Understanding by Generative Pre-Training". 2017.

8 Submission of papers to NeurIPS 2023

Please read the instructions below carefully and follow them faithfully. **Important:** This year the checklist will be submitted separately from the main paper in OpenReview, please review it well ahead of the submission deadline: `https://neurips.cc/public/guides/PaperChecklist`.

8.1 Style

Papers to be submitted to NeurIPS 2023 must be prepared according to the instructions presented here. Papers may only be up to **nine** pages long, including figures. Additional pages *containing only acknowledgments and references* are allowed. Papers that exceed the page limit will not be reviewed, or in any other way considered for presentation at the conference.

The margins in 2023 are the same as those in previous years.

Authors are required to use the NeurIPS LaTeX style files obtainable at the NeurIPS website as indicated below. Please make sure you use the current files and not previous versions. Tweaking the style files may be grounds for rejection.

8.2 Retrieval of style files

The style files for NeurIPS and other conference information are available on the website at

`http://www.neurips.cc/`

The file `neurips_2023.pdf` contains these instructions and illustrates the various formatting requirements your NeurIPS paper must satisfy.

The only supported style file for NeurIPS 2023 is `neurips_2023.sty`, rewritten for LaTeX 2_ε. **Previous style files for LaTeX 2.09, Microsoft Word, and RTF are no longer supported!**

The LaTeX style file contains three optional arguments: `final`, which creates a camera-ready copy, `preprint`, which creates a preprint for submission to, e.g., arXiv, and `nonatbib`, which will not load the `natbib` package for you in case of package clash.

Preprint option If you wish to post a preprint of your work online, e.g., on arXiv, using the NeurIPS style, please use the `preprint` option. This will create a nonanonymized version of your work with the text "Preprint. Work in progress." in the footer. This version may be distributed as you see fit, as long as you do not say which conference it was submitted to. Please **do not** use the `final` option, which should **only** be used for papers accepted to NeurIPS.

At submission time, please omit the `final` and `preprint` options. This will anonymize your submission and add line numbers to aid review. Please do *not* refer to these line numbers in your paper as they will be removed during generation of camera-ready copies.

The file `neurips_2023.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own.

The formatting instructions contained in these style files are summarized in Sections 9, 10, and 11 below.

9 General formatting instructions

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing (leading) of 11 points. Times New Roman is the preferred typeface throughout, and will be selected for you by default. Paragraphs are separated by ½ line space (5.5 points), with no indentation.

The paper title should be 17 point, initial caps/lower case, bold, centered between two horizontal rules. The top rule should be 4 points thick and the bottom rule should be 1 point thick. Allow ¼ inch space above and below the title to rules. All pages should start at 1 inch (6 picas) from the top of the page.

For the final version, authors' names are set in boldface, and each name is centered above the corresponding address. The lead author's name is to be listed first (left-most), and the co-authors' names (if different address) are set to follow. If there is only one co-author, list both author and co-author side by side.

Please pay special attention to the instructions in Section 11 regarding figures, tables, acknowledgments, and references.

10 Headings: first level

All headings should be lower case (except for first word and proper nouns), flush left, and bold.

First-level headings should be in 12-point type.

10.1 Headings: second level

Second-level headings should be in 10-point type.

10.1.1 Headings: third level

Third-level headings should be in 10-point type.

Paragraphs There is also a `\paragraph` command available, which sets the heading in bold, flush left, and inline with the text, with the heading followed by 1 em of space.

11 Citations, figures, tables, references

These instructions apply to everyone.

11.1 Citations within the text

The `natbib` package will be loaded for you by default. Citations may be author/year or numeric, as long as you maintain internal consistency. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

The documentation for `natbib` may be found at

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

Hasselmo, et al. (1995) investigated…

If you wish to load the `natbib` package with options, you may add the following before loading the `neurips_2023` package:

```
\PassOptionsToPackage{options}{natbib}
```

If `natbib` clashes with another package you load, you can add the optional argument `nonatbib` when loading the style file:

```
\usepackage[nonatbib]{neurips_2023}
```

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]," not "In our previous work [4]." If you cite your other papers that are not widely available (e.g., a journal paper under review), use anonymous author names in the citation, e.g., an author of the form "A. Anonymous" and include a copy of the anonymized paper in the supplementary material.

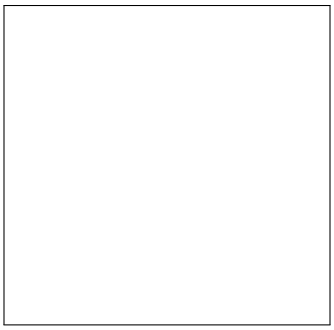

Figure 2: Sample figure caption.

Table 2: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

11.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[7] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[8]

11.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

11.4 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 2.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

Note that publication-quality tables *do not contain vertical rules.* We strongly suggest the use of the `booktabs` package, which allows for typesetting high-quality, professional tables:

`https://www.ctan.org/pkg/booktabs`

This package was used to typeset Table 2.

[7] Sample of the first footnote.

[8] As in this example.

11.5 Math

Note that display math in bare TeX commands will not create correct line numbers for submission. Please use LaTeX (or AMSTeX) commands for unnumbered display math. (You really shouldn't be using $$ anyway; see `https://tex.stackexchange.com/questions/503/why-is-preferable-to` and `https://tex.stackexchange.com/questions/40492/what-are-the-differences-between-align-equation-and-displaymath` for more information.)

11.6 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

12 Preparing PDF files

Please prepare submission files with paper size "US Letter," and not, for example, "A4."

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You should directly generate PDF files using `pdflatex`.
- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- `xfig` "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
- The `\bbold` package almost always uses bitmap fonts. You should use the equivalent AMS Fonts:

  ```
  \usepackage{amsfonts}
  ```

  followed by, e.g., `\mathbb{R}`, `\mathbb{N}`, or `\mathbb{C}` for $\mathbb{R}$, $\mathbb{N}$ or $\mathbb{C}$. You can also use the following workaround for reals, natural and complex:

  ```
  \newcommand{\RR}{I\!\!R} %real numbers
  \newcommand{\Nat}{I\!\!N} %natural numbers
  \newcommand{\CC}{I\!\!\!\!C} %complex numbers
  ```

  Note that `amsfonts` is automatically loaded by the `amssymb` package.

If your file contains type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

12.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width as in the example below:

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

See Section 4.4 in the graphics bundle documentation (`http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command when necessary.

Acknowledgments and Disclosure of Funding

Use unnumbered first level headings for the acknowledgments. All acknowledgments go at the end of the paper before the list of references. Moreover, you are required to declare funding (financial activities supporting the submitted work) and competing interests (related financial activities outside the submitted work). More information about this disclosure can be found at: `https://neurips.cc/Conferences/2023/PaperInformation/FundingDisclosure`.

Do **not** include this section in the anonymized submission, only in the final paper. You can use the `ack` environment provided in the style file to autmoatically hide this section in the anonymized submission.

13 Supplementary Material

Authors may wish to optionally include extra information (complete proofs, additional experiments and plots) in the appendix. All such materials should be part of the supplemental material (submitted separately) and should NOT be included in the main submission.

References

References follow the acknowledgments in the camera-ready paper. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. Note that the Reference section does not count towards the page limit.

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.