dBaby: Grounded Language Learning through Games and Efficient Reinforcement Learning

Guntis Barzdins
AI-Lab at IMCS
University of Latvia
Riga LV-1459, Latvia
guntis.barzdins@lu.lv

Renars Liepins
Innovation Labs
LETA
Riga LV-1050, Latvia
renars.liepins@leta.lv

Didzis Gosko
Robotics Lab
Accenture
Riga LV-1050, Latvia
didzis@gmail.com

Abstract

This paper outlines a project proposal to be submitted to EC H2020 call ICT-29-2018. The purpose of the project is to create a digital Baby (dBaby) - an agent perceiving and interacting with the 3D world and communicating with its Teacher via natural language phrases to achieve the goals set by the Teacher. The novelty of the approach is that neither language nor visual capabilities are hard-coded in the dBaby - instead, the Teacher defines a grounded language learning Game, and dBaby learns the language as a byproduct of the reinforcement learning from raw pixels and character strings while maximizing the rewards in the Game. So far such approach successfully has been demonstrated only in the virtual 3D world with pre-programmed Games where it requires millions of episodes to learn a dozen words. Moving to human Teacher and real 3D environment requires an order-of-magnitude improvement to data-efficiency of the reinforcement learning. A novel Memory Neuron based Episodic Control is demonstrated as a promising approach for bootstrapping the data-efficient reinforcement learning.

1 Introduction

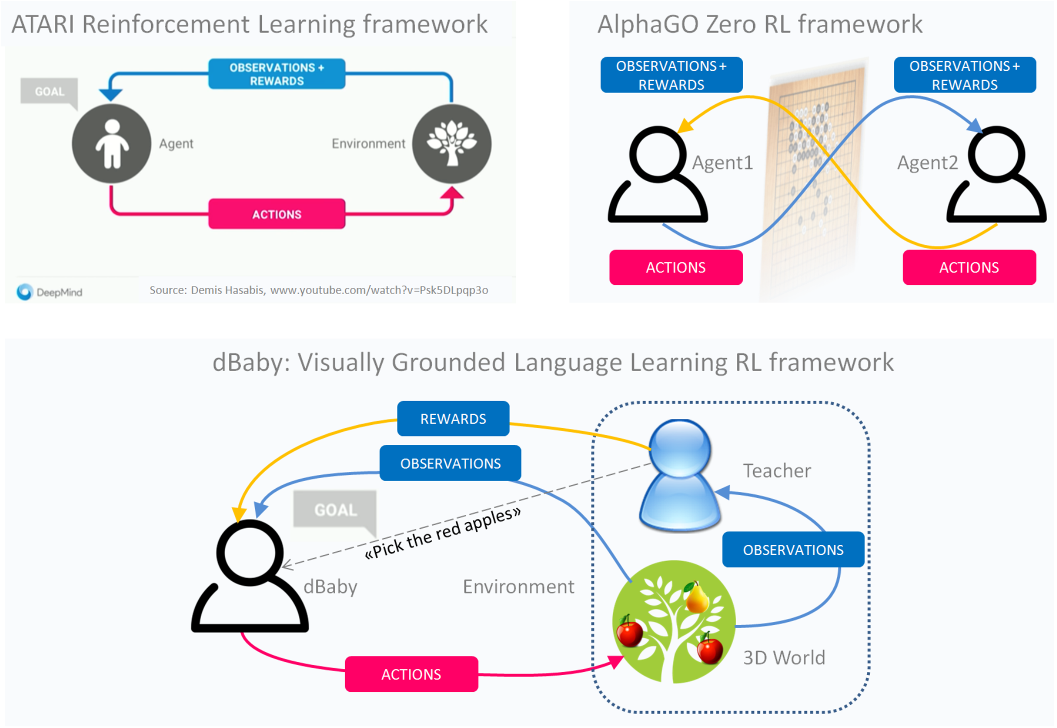
The digital Baby (dBaby) project proposal to EC H2020 call ICT-29-2018 described in this paper is a follow-up to the already running ICT-16-2015 project SUMMA.[1] During the SUMMA project, it has become apparent that the current state-of-art NLP approaches to ASR, MT, NER, NEL, AMR, KBP, Summarizing are suitable only for gisting purposes but will never achieve actual natural language understanding expected by the project end-users (BBC and DW) in the news-monitoring use-case. The current statistical NLP approaches remain in word-symbols curse [] and due to their supervised learning nature are constrained by the training corpora inter-annotator agreement ratio. The deep learning and embedding methods have recently allowed to nearly reach this ratio, but they are helpless to overcome it.

Genuine natural language understanding requires a different approach. Grounded language learning through reinforcement learning has recently emerged [] as a potential alternative and is at the core of the dBaby project proposal. The attractiveness of this approach is strengthened by the deep reinforcement learning recently demonstrating super-human capabilities both in Atari games [] and the very challenging GO game[]. This progress allows us to speculate that someday dBaby might achieve super-human fluency in natural language understanding and generation, although in this project we aim only to demonstrate the basic viability of the approach.

In Figure 1 illustrates how the dBaby approach relates and differs from other reinforcement learning frameworks. dBaby is a reinforcement learning agent perceiving and interacting with the 3D world

[1] http://summa-project.eu

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



Source: Demis Hasabis, www.youtube.com/watch?v=Psk5DLpqp3o

Figure 1: Three reinforcement learning frameworks. (a) Atari, (b) AlphaGO, (c) dBaby.

and communicating with its Teacher via natural language phrases to achieve the goals set by the Teacher. The novelty of the approach is that neither language nor visual capabilities are hard-coded in the dBaby - instead, the Teacher defines a language learning Game grounded in the 3D world, and dBaby learns the language as a byproduct of the reinforcement learning from raw pixels and character strings while maximizing the rewards in the Game. The dBaby framework differs from the previous approaches in that Environment is split into 3D world and Teacher, where the Teacher is the one defining a verbal goal and providing the rewards according to the set Goal. In this way, the Game to be learned by the dBaby is effectively defined by the Teacher, and 3D world merely acts as a visual "language" in which the natural language gets grounded.

The goal of the Game is conveyed by the Teacher to dBaby as a character string (e.g. "Pick the red apples"). The dBaby treats this character string as part of the observation from the environment along with the raw pixels observed from the 3D world. The reason in Figure 1 (c) we separate goal from the observation is because they come from different sources - from the Teacher and from the 3D world respectively and thus play rather different roles in the overall RL framework. It is interesting to note that the concept of a "goal" separate from "observation" appears already in Figure 1 (a) depicting classic Atari reinforcement learning as presented by D.Hasabis[2] [], where he describes it as "agent finds itself in some sort of environment and it is trying to achieve a goal in that environment ... Goal is simply to maximize the score". In case of dBaby framework the "goal" is any natural language phrase spelled by the Teacher, so it can be either "Maximize the score" or it can be a more high-level goal like "Pick the red apples".

So far the dBaby approach successfully has been demonstrated only in the virtual 3D world with pre-programmed Games [] where it requires millions of episodes to learn a dozen words. Section 2 illustrates the dBaby approach and shows that moving to human Teacher and real 3D environment or self-play between dBabies requires an order-of-magnitude improvement to data-efficiency of the reinforcement learning. In Section 3 we present Memory Neuron based Episodic Control as a promising approach we are exploring towards bootstrapping the data-efficient reinforcement learning.

[2] youtu.be/Psk5DLpqp3o

2 dBaby as a Teacher for another dBaby

For the baseline implementation[3] of the dBaby framework in the simulated 3D world and pre-programmed Teacher we refer to [] as we are still in the process of replicating these results. An interesting aspect of this baseline implementation is that the agent (dBaby) not only listens to the goals expressed as the natural language phrases by Teacher, but it is also able to output the names of objects in its view eventually leading to the two-way natural language communication. Although this aspect is not essential for the core dBaby framework illustrated in Figure 1 (c), eventually it is of interest for natural language generation and for self-play where one dBaby acts as a Teacher for another dBaby.

Scaling the dBaby framework beyond the above baseline implementation towards human Teacher and real 3D environment requires an order-of-magnitude improvement to the data-efficiency of the reinforcement learning because neither human Teacher nor robot in a real 3D environment can sustain millions of training episodes. Scaling must also avoid the catastrophic forgetting of the previously acquired language skills. To that end here we discuss some extensions to the above baseline framework.

As was demonstrated in the baseline implementation (mentioned above) it is quite easy to create a pre-programmed Teacher to teach the embodied meaning of simple sentences like "pick blue object" or "pick the red TV next to a green object in the magenta room". It would be quite easy to extend the Teacher to teach more extended tasks like moving objects around or placing objects into some arrangements. However to scale it more complex tasks become hard because not only do we need to write a Teacher that can recognize when a task has been done, but also we need to program a language generation module to describe the result. The task generation and recognition are quite doable if we have a 3D environment. To generate a task we merely randomly generate two environment states and the task of the dBaby is to transform the environment from the first state to the second. However, the generation of natural language description for this task is much harder. What we can do here, is to let the dBaby and the Teacher learn their own language for communication. To force this language to be close to English we can add extra constraints. E.g., add a network that is trained on english language model and give dBaby and Teacher a penalty to how far their language is from the english language model. We can also add an object recognition network that is first used to associate object names with virtual objects in the environment. Another option is to use the information from the environment, i.e., when we are generating tasks we know which objects are in the environment and which of those objects have moved or changed between the initial state and the target state. Thus we can enforce that the dBaby and Teacher communication must involve the names of some of these objects. Maybe the hardest part would be to impose a consistent use of English verbs because it is not easy to recognize them automatically (as opposed to the nouns that correspond to objects in the environment). Here we propose to use a relatively small set of human-labeled inputs where real people describe the tasks that the dBaby and Teacher are doing and them as training samples to force dBaby and Teacher to use the same verbs in their communication.

3 temporal Auto Encoder and Memory Neuron

The dBaby baseline implementation [] used temporal Auto Encoder (tAE) [] to improve the data-efficiency of the reinforcement learning process - without tAE authors were not even able to train the network.

In this section we propose an additional pre-training technique which can be used together with tAE to further improve the data-efficiency of the reinforcement learning. The integrated tAE and pre-training implementation here is demonstrated on the Atari Pong game in the OpenAI Gym environment, under the assumption that similar data-efficiency gains can also be achieved in the 3D environment of the dBaby framework.

Figure 2 illustrates the reinforcement learning data-efficiency gains achieved. The graph (a) shows the baseline policy gradient reinforcement learning algorithm [Karpathy] performance on the raw pixels (8400dim) of the Pong game as input. The graph (b) shows the performance of the same policy gradient reinforcement learning algorithm when the raw pixels are replaced by the low-dimensional

[3] See video at youtu.be/wJjdu1bPJ04

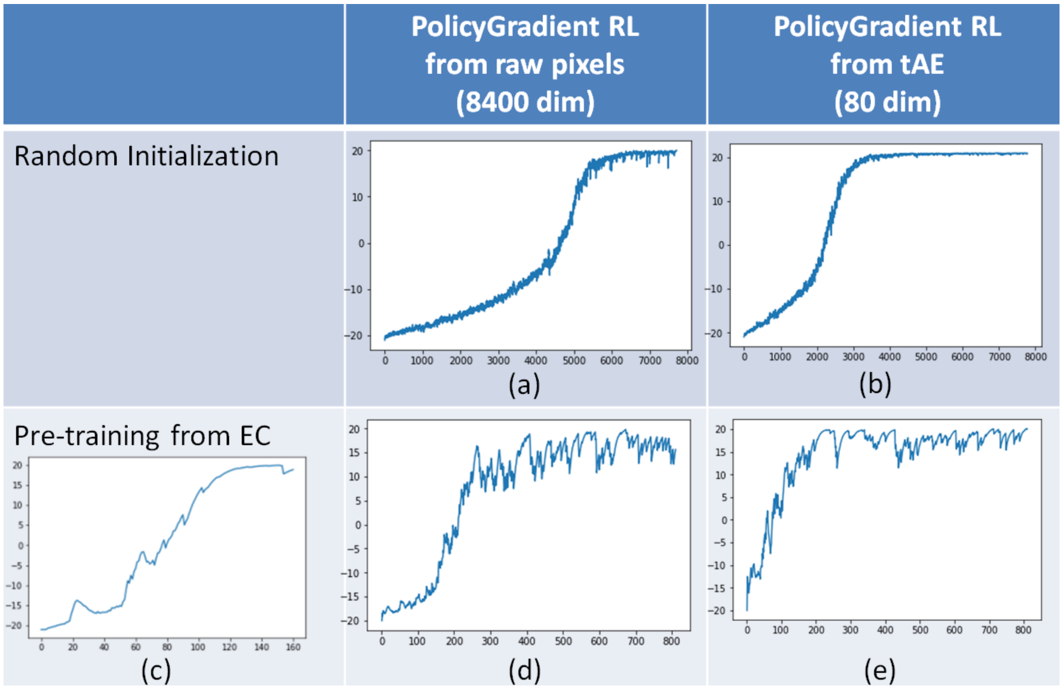


Figure 2: Data-efficiency on Atari Pong game for various RL setups: (a) Policy-gradient from raw pixels, (b) Policy-gradient with tAE, (c) Episodic Control RL, (d) Policy-gradient pre-trained by EC, (e) Policy-gradient with tAE pretrained by EC. Horizontal axis shows episode count and vertical axis shows average score achieved.

(80dim) tAE output[tAE paper]. Finally, (c) shows the simplified EC [] performance om the raw pixels (8400dim).

As can be seen from graphs (a), (b), (c) of Figure 2, the baseline (a) and tAE (b) approaches have similar data-efficiency, while EC (c) is substantially more efficient. The problem is that RL is not really a neural network, but a computationally inefficient KNN approach with a full memory about the previous actions and their outcomes and thus is not a practical substitute for deep reinforcement learning.

To overcome this problem, the novelty of our approach is to use EC game-play recordings for pre-training the RL wights in a supervised manner. The graphs (d) and (e) in Figure 2 show the resulting improved data-efficiency of both baseline and tAE approaches from (a) and (b) respectively. The data-efficiency gain can be measured by the number of episodes required to reach score 0 (RL agent plays as good as the Atari built-in agent) - the gain is about 10x taking into account the 160 episodes played by EC for graphs (d) and (e).

Due to space limitations we are omitting the details of the implementation for this novel method, and refer to the actual code provided online.[4] All components of the integrated solution are described in the respective publications referenced above and the novelty is only in the way they are integrated together. The key trick of the integrated solution which makes or breaks it is that only the positive examples from the EC game-play (state/action sequences leading to positive reward in the short term) should be used for supervised pre-training of RL weights.

[4] www.github.com/abc

4 Conclusions

As a project proposal outline this paper clearly is not a finished work - it is rather our perspective on future directions in the grounded language learning. Nevertheless, we hope it being fruitful sharing our perspective already in this early proposal preparation stage.

Acknowledgments

This work was supported in part by the Latvian National research program SOPHIS under grant agreement Nr.10-4/VPP-4/11 and in part by H2020 SUMMA project under grant agreement 688139/H2020-ICT-2015.

5 Renara teikumi

Intro/Problem

- current approaches to NLP have a low upper limit to performance because need human annotations but inter-annotator agreement only 80%

- deep reinforcement learning and self-play has demonstrated superhuman performance in limited domains (e.g. AlphaGo Zero)

- how to apply Deep RL and Self-play to NLP

What we need to map from tasks like GO to NLP

- GO has a clear Goal (win game)

- NLP does not have such a simple goal, but we can reformulate that the goal of Language is communication. I.e. one agent want some change in the environment and uses language to communicate this change to the second agent. The second agent performs actions in the environment. Meanwhile, the first agent monitors the environment to check when the change he desired has been achieved. Therefore we need two players. One (Teacher) who communicates using language and recognizes when the goal is met. And second (dBaby) who manipulates environment using actions until he achieves the state that the Teacher wants.

- Question how to bootstrap. DeepMind demonstrated that for simple tasks we can create a Teacher programmatically. Generate task description (e.g. "find pink spoon"), and monitor the environment until the dBably's position overlaps that object.

- Question how to generalize further...

Bootstrapping and Self-play

Some simple tasks for which it is easy to write a hand-coded teacher.

- First generation dBaby and Teacher - learning to names of basic actions (e.g. move-forward, move-right, move-left, etc.)

- Second generation dBaby and Teacher - learning name of objects (e.g. find objectX, find adjectiveY objectZ)

These tasks are not directly applicable to bootstrapping but they might come handy for a base on which to build a bootstrapping process. These tasks might also be useful because in subsequent tasks dBaby could describe what it's trying to achieve. E.g. if dBabe is in the middle of a room (no objectX nearby), and it uses future unrolling to imagine a state where it is near objectX, than dBaby can verbalize all the steps in between as "find objectX". So in a way, dBaby has acquired a new higher-level action "find X".

5.1 Style

Papers to be submitted to NIPS 2017 must be prepared according to the instructions presented here. Papers may only be up to eight pages long, including figures. This does not include acknowledgments and cited references which are allowed on subsequent pages. Papers that exceed these limits will not be reviewed, or in any other way considered for presentation at the conference.

The margins in 2017 are the same as since 2007, which allow for $\sim$15% more words in the paper compared to earlier years.

Authors are required to use the NIPS LaTeX style files obtainable at the NIPS 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.

5.2 Retrieval of style files

The style files for NIPS and other conference information are available on the World Wide Web at

`http://www.nips.cc/`

The file `nips_2017.pdf` contains these instructions and illustrates the various formatting requirements your NIPS paper must satisfy.

The only supported style file for NIPS 2017 is `nips_2017.sty`, rewritten for LaTeX 2_ε. **Previous style files for LaTeX 2.09, Microsoft Word, and RTF are no longer supported!**

The new LaTeX style file contains two optional arguments: `final`, which creates a camera-ready copy, and `nonatbib`, which will not load the `natbib` package for you in case of package clash.

At submission time, please omit the `final` option. 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 `nips_2017.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 6, 7, and 8 below.

6 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 8 regarding figures, tables, acknowledgments, and references.

7 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.

7.1 Headings: second level

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

7.1.1 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.

8 Citations, figures, tables, references

These instructions apply to everyone.

8.1 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 nips_2017 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]{nips_2017}
```

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."

8.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[5] 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.[6]

8.3 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.

[5] Sample of the first footnote.

[6] As in this example.

Table 1: 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

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.

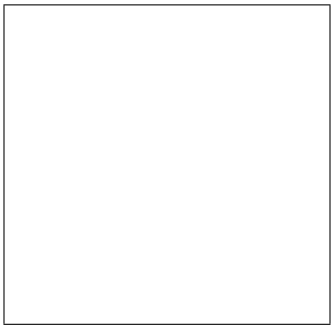


Figure 3: Sample figure caption.

8.4 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 1.

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 1.

9 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.

10 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.

- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NIPS. Please see `http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf`
- `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.

10.1 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

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

References

References follow the acknowledgments. 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. **Remember that you can go over 8 pages as long as the subsequent ones contain *only* cited references.**

[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.