# Handwriting Generation with Recurrent Neural Networks

**Jing Leng**
University of Michigan
lengjing@umich.edu

**Shayan Masooman**
University of Michigan
masooman@umich.edu

**Blake Smith**
University of Michigan
blakesm@umich.edu

## Abstract

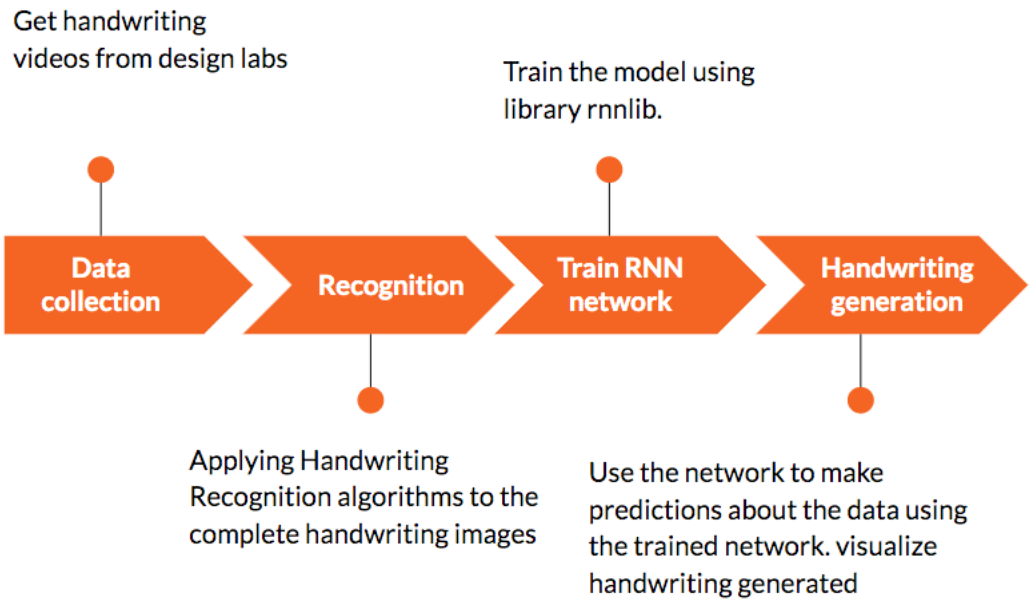
In this report we talk about using recurrent neural networks to generate a font that resembles human handwriting. There have been various methods proposed to make machines mimick handwriting in previous research however we chose to approach this problem through the use of Long Short-term Memory recurrent neural networks to make predictions dealing with sequential data. To collect data, we employed video processing software and then used handwriting recognition algorithms on an image of the handwriting to link it with its print-text equivalent. As you will see, the experiments we completed produced satisfactory results in the sense that our algorithm produces an eligible handwriting font for a given input.

## 1 Introduction

Optical character recognition (OCR) is a widely used form of human-computer interaction that can be seen in society today. It is the conversion of images of typed or printed text into machine-encoded text often used for data entry, text-to-speech, and text mining. In this project we have implemented Intelligent Character Recognition (ICR) software that has the capability to take OCR to the next level and recognize handwritten text one character at a time. This is all done with the intention of using handwriting recognition to construct an exclusive and accurate handwriting font tailored to unique input in order to grant out users the ability to add a personal touch to the banal style of standard electronic fonts.

This complex input will be seamlessly fed to our software through the utilization of the Wacom Intuos Draw Tablet Stylus here on the University of Michigan campus. The Wacom Intous Draw Tablet grants the capability to track dynamic motion of the pen tip in real-time as our users input their handwriting, which will be used to construct a data set and recreate the handwriting to a digital image. It is at this point that the handwriting recognition software can be used to analyze the handwriting and generate a text file of matching text ultimately feeding these two correlated inputs to the rest of our software to begin creating a realistic font simply off a sample of our users handwriting.

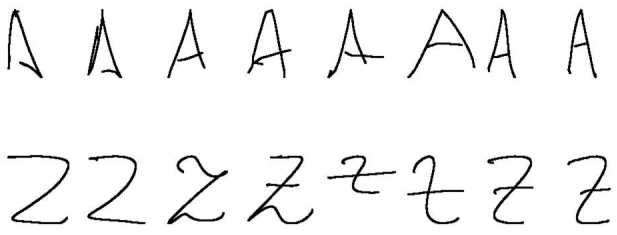
Creation of a realistic font is most efficiently done through the process of machine learning in which the computer trains a model from extensive previous input to make data-driven decisions (i.e. predictions) rather than following strictly static program instructions. With the ability to process and store handwritten text by comparison between scanned text and learned text, our software will ultimately construct a font that is particular to the handwriting input to be used for future use on the system. A requisite for a font to resemble actual handwriting is that each letter must be displayed diversely each time it is used, and its size and shape will depend on the values of surrounding letters. Conclusively, this project has provided us with useful experience in the practice of video processing, handwriting recognition, and machine learning using video as input to create a unique tool to use in a vast variety of circumstances. A concept figure illustrating the method described is shown below.




## 2 Previous work

### 2.1 Handwriting recognition

Recognizing handwriting from an image poses several problems and is no simple task. The most important obstacle is that there is a significant amount of variation within classes, as you can see below, there are several different ways the letters ”A” and ”Z” can be handwritten.



The MATLAB OCR function in the Computer Vision toolbox works extremely well with recognizing typed text in an image, as shown below.


However, the OCR function works very poorly with actual handwriting recognition, making it unsuitable for this project. The method we implemented is similar to the paper "Optical Character Recognition for Handwritten Hindi". This paper recognizes hindi text by using Histogram of Oriented Gradient (HOG) Features with three different classifiers: Naive Bayes, Support Vector Machines, and Adaboost. The Support Vector Machines gave the highest test accuracy (1.2 percent misclassification rate), so we decided to use the SVM classifier for our method.

### 2.2 Handwriting generation

There have been plenty of ways to approach the problem of handwriting synthesis. They can be categorized into Perturbation-based techniques, Fusion-based generation, and Model-based generation. Perturbation-based techniques generate new samples by altering geometric features such as the size, thickness, and slant of a given sample. Perturbation-based operations can be seen as the inverse of the preprocessing steps employed in text recognition. Perturbation-based techniques are easy to apply, but the results may be unnatural due to random and non-calibrated parameter settings.

Fusion-based techniques take few input samples and combine them into new synthesized outputs. They differ from concatenation techniques in that they generate scripting units at the same level as their inputs; e.g., characters generate new characters. Shape-matching algorithms are necessary for fusionbased techniques to make sure that segments are properly aligned. The number of unique outputs is limited in fusionbased techniques as compared to that of other generation techniques.

Model-based techniques capture the statistics of natural handwriting variations into models. Although model-based techniques are profoundly established in theory, they may often be challenging to implement due to the large number of samples they require. Models resulting from these techniques can also be utilized in recognition systems.

## 3 Technical details

### 3.1 Data Collection

The first step to our software process is for our user to input their handwriting. We need to process this input in real time in order to gather sufficient data of where the pen tip is at all times while it is writing and an indication of when it is picked up. This will ensure that with the help of our trained machine, the font we generate is realistic in the sense of all inconsistencies that normal handwriting possesses. Therefore we set out to obtain x and y of the pen-tip corresponding to its timestamp and when the pen is lifted so that the next x and y pair that you will see will be the initial point of a new stroke.

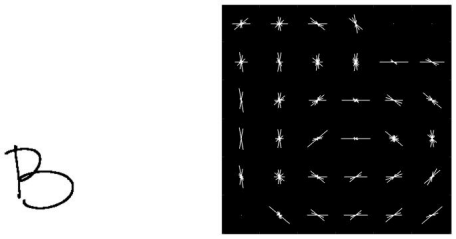
To capture the users handwriting we chose to utilize the accessible and accurate Wacom Intuos Draw Tablet and Stylus available on North Campus, with Adobe Photoshop. After opening a blank canvas and selecting the pen tool, the individual uses the stylus to write out a phrase and with Apple QuickTime Players built in screen capture we obtain a video of the handwriting phrase frame by frame as it is written. The input to be used by the initial stage of our software has thus been obtained.

As mentioned, this video needs to be processed and converted to a formatted data file. To accomplish this we utilized the video processing capabilities of Matlabs videoReader functionality. Employing frame by frame comparisons to detect pixel discrepancies, a competently accurate representation of the handwriting is exported to an excel document in all necessary variables. With this data in its appropriate format, the last frame of the video is used as image input to another portion of our software to recognize the handwriting and construct a text file to pair with the output of data collection.

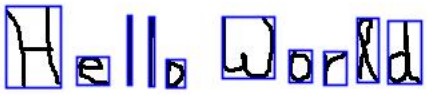
### 3.2 Handwriting recognition

In order to use the Histogram of Oriented Gradients (HOG) Features and Support Vector Machine Classifier (SVM), we needed a large dataset which contained images of handwritten letters. The UCI Machine Learning Repository had sample penstroke data from 60 different writers, each ASCII character as well as a few spanish characters were included in the dataset. We only used the uppercase and lowercase letters of the English alphabet, and converted the penstroke data into $120 \times 120$ pixel images for each letter. This gave us a training set of 60 $120 \times 120$ pixel images for each letter.

On each of the images of the training set, we found the HOG features using a window size of 20, and input these into an SVM in order to get a classifier. The visualization of the HOG features is shown in the image below. The window size of 20 gave adequate information on the shape of each letter while still providing a reasonable runtime.



On any new image, the MATLAB function **improps** was used to separate letters, then our classification was used on each separate letter.



### 3.3 Handwriting generation

Neural networks (NN) are know to do a good job in capturing complex, non-linear connections between input and output. A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs

can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition or speech recognition.

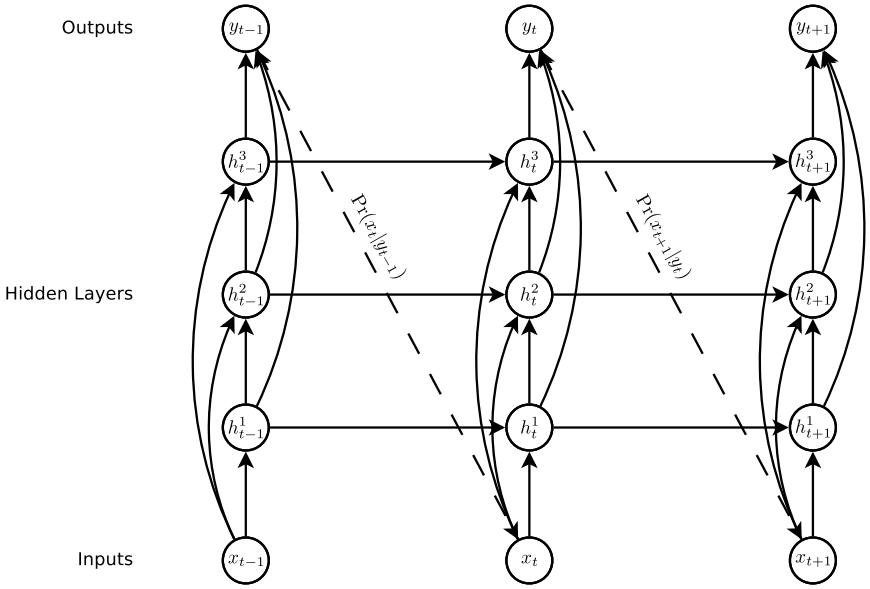



Figure 1: **Deep recurrent neural network prediction architecture.** The circles represent network layers, the solid lines represent weighted connections and the dashed lines represent predictions.

In Fig. 1, an input vector sequence $\mathbf{x} = (x_1, \ldots, x_T)$ is passed long a weighted connections to a stack of $N$ recurrently interconnected hidden layers to compute the first hidden vector sequences $\mathbf{h}^n = (h^n_1, \ldots, h^n_T)$, and then the output vector sequence $\mathbf{y} = (y_1, \ldots, y_T)$

Each output vector $y_t$ is used to parameterise a predictive distribution $\Pr(x_{t+1}|y_t)$ over the possible next inputs $x_{t+1}$.

The first element $x_1$ of every input sequence is always a null vector whose entries are all zero; the network therefore emits a prediction for $x_2$, the first real input, with no prior information. The network is 'deep' in both space and time, in the sense that every piece of information passing either vertically or horizontally through the computation graph will be acted on by multiple successive weight matrices and nonlinearities.

How each hidden node is calculated can be shown in the following formula:

$$h^1_t = \mathcal{H}\left(W_{ih^1}x_t + W_{h^1h^1}h^1_{t-1} + b^1_h\right) \tag{1}$$

$$h^n_t = \mathcal{H}\left(W_{ih^n}x_t + W_{h^{n-1}h^n}h^{n-1}_t + W_{h^nh^n}h^n_{t-1} + b^n_h\right) \tag{2}$$

where the $W$ terms denote weight matrices (e.g. $W_{ih^n}$ is the weight matrix connecting the inputs to the $n^{th}$ hidden layer, $W_{h^1h^1}$ is the recurrent connection at the first hidden layer, and so on), the $b$ terms denote bias vectors (e.g. $b_o$ is output bias vector) and $\mathcal{H}$ is the hidden layer function.

Given the hidden sequences, the output sequence can be calculated:

$$\hat{y}_t = b_o + \sum_{n=1}^{N} W_{h^ny}h^n_t \tag{3}$$

$$y_t = \mathcal{Y}(\hat{y}_t) \tag{4}$$

The Long short-term memory (LSTM) network, developed by Hochreiter & Schmidhuber in 1997, is an artificial neural net structure that unlike traditional RNNs doesn't have the vanishing gradient problem (compare the section on training algorithms below). It works even when there are long delays, and it can handle signals that have a mix of low and high frequency components. LSTM RNN outperformed other methods in numerous applications such as language learning and connected handwriting recognition.

That is how the network is built.

In this project, the data is the sequence

To enable the network to make prediction about a sequence, the output sequence is used as a parameter of the probability distribution $\Pr(x_{t+1}|y_t)$ of the next input. The form of distribution should be selected carefully to give high performing prediction.

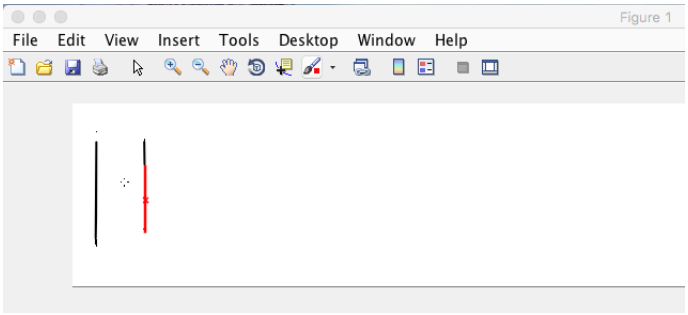
The probability given by the network to the input sequence x is

$$\Pr(\mathbf{x}) = \prod_{t=1}^{T} \Pr(x_{t+1}|y_t) \tag{5}$$

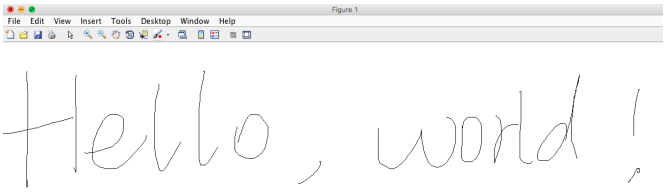
# 4 Experiments

## 4.1 Data collection

As discussed in Section 3.1, the input to be translated to collected data is in video format and is processed by detecting pixel discrepancies between each frame. An example of detected pixel discrepancies is shown below where detected discrepancies between a frame range was highlighted to show accuracy in the construction of the letter H. Note that the cursor does not interfere with detection due to disregard of pixels that have been accurately identified as pixels changed due to cursor for every iteration/frame.




At the completion of the video processing there should be two accessible outputs for use later in software. The first is the last frame of the handwriting input and the second is the formatted data. Examples of these are shown below.



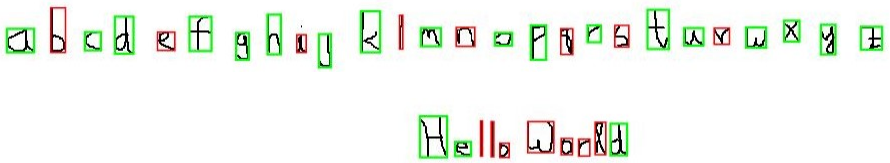
| Δxcor | Δycor | time stamp | end stroke |
|---|---|---|---|
| 1073 | 1058 | 769.05 | 0 |
| -1 | 27 | 769.07 | 0 |
| -6 | 32 | 769.08 | 0 |
| -14 | 35 | 769.1 | 0 |
| -22 | 44 | 769.12 | 0 |
| -21 | 46 | 769.13 | 0 |
| -15 | 44 | 769.14 | 0 |
| -14 | 31 | 769.16 | 0 |
| -9 | 19 | 769.18 | 0 |
| -3 | 8 | 769.19 | 0 |
| -2 | -5 | 769.2 | 0 |
| 6 | 1 | 769.22 | 0 |
| 6 | -20 | 769.24 | 0 |
| 13 | -22 | 769.25 | 0 |
| 12 | -32 | 769.25 | 0 |
| 13 | -35 | 769.27 | 0 |
| 5 | -47 | 769.28 | 0 |
| 9 | -41 | 769.3 | 0 |
| 10 | -35 | 769.31 | 0 |
| 9 | -31 | 769.33 | 0 |
| 6 | -28 | 769.34 | 0 |
| 3 | -28 | 769.36 | 0 |
| 6 | -15 | 769.37 | 0 |
| 7 | 0 | 768.38 | 0 |
| 3 | 7 | 769.4 | 0 |
| 9 | 29 | 769.42 | 0 |
| 14 | 40 | 769.43 | 0 |
| 17 | 42 | 769.45 | 0 |
| 20 | 48 | 769.46 | 0 |
| 24 | 53 | 769.48 | 0 |
| 23 | 55 | 769.49 | 0 |
| 19 | 47 | 769.51 | 0 |
| 18 | 38 | 769.52 | 0 |
| 2 | 8 | 769.54 | 0 |
| 3 | 3 | 769.55 | 0 |
| -2 | -4 | 769.57 | 0 |
| -1 | -3 | 769.58 | 0 |
| -5 | -16 | 769.6 | 0 |
| -11 | -23 | 769.64 | 1 |

### 4.2 Handwriting recognition

As discussed in Section 3.2, letters were separated using the **improps** MATLAB function, and our SVM classifier was used on each separate letter. The results were extremely accurate on typed text, as shown below (green=correct classification was made, red=incorrect classification was made).

However, there was still a very high misclassification rate on handwritten text (30 percent). As you can see below, several letters are still being misclassified. While this is a fairly good handwriting recognizer, it still causes a huge problem in our project, since some of the letters from the ”Hello World” example from Section 3.1 will be misclassified,

the error will propogate to the handwriting generator, as dicussed in the following section.



### 4.3 Handwriting generation

First we collected a trace of the pen stroke using a tracking pad and screen capture. Then we used the handwriting recognition technology to generate the contents of handwriting. With the data collected we trained a Recurrent Neural Network, which is later used for handwriting generation.

After we collected and formatted the data, we use the data to train a recurrent neural network. The idea of RNN is to encode sequential correlation of ordered data points, in our case, the pen stroke data introduced earlier. The input and output variable will be the same sequence, except they are offsetted by one timestamp to make prediction possible. By training the recurrent neural network we are basically telling the program how to write.

Using the well-implemented C++ library rnnlib, we were able to train an RNN that takes in the sequential data of the pen stroke, which is then used for making predictions about handwriting, i.e. handwriting generation. The training of the network took longer than we thought. Each epoch takes 30 hours or so, and therefore we only managed to run 7 epochs. The training error was still dropping when we stop the training process early because of time issues, so the network is not yet well trained. Nevertheless it has given us eligible results.

Here are two examples:

Figure 2: ”hello, it’s me”

Figure 3: ”I am your father”

Fig. 4.3, which says ”hello, it’s me”, is very legible, and looks like a sensible handwriting version of the text. Looking closely, we can see the e at the end was not finished. That is because the end of stroke flag appeared too soon. In another case where the algorithm did not do so well, Fig. 4.3 is less handwriting-like, and less legible.

## 5 Conclusions

In this project, Long Short-Term Memory recurrent neural networks have displayed a strong ability in making predictions about human handwriting sequence. Although it did not reach its full potential, the RNN we trained is able to generate convincing handwriting data as expressed in our experimental data above. There are a few ways we could improve the project, however. A simple improvement given more time would be to train the network more to improve our output. Additionally, we have come to realize that the rnn library we are using is not optimized for multiprocessors,

or gpu, so a strive to improving on their code would go a long way. And finally if we really dove into this project we could also try to improve the architecture of the neural networks, however if that succeeded we would be famous. All in all, this project has given us great insight to video processing, handwriting recognition, and machine learning all contributing to a better understanding of computer vision and the frontier of discovery and improvement that it proposes.

### 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.

[4] Vincent, N., Seropian, A., Stamon, G.: Synthesis for handwriting analysis. Pattern Recognit. Lett. 26(3), 267275 (2005)

[5] Goyal, A., Khandelwal, K., Keshri, P., (2010) "Optical Character Recognition for Handwritten Hindi". Stanford, CA.


Below is not part of the report

## 6 Submission of papers to NIPS 2015

NIPS requires electronic submissions. The electronic submission site is

`http://papers.nips.cc`

Please read carefully the instructions below, and follow them faithfully.

### 6.1 Style

Papers to be submitted to NIPS 2015 must be prepared according to the instructions presented here. Papers may be only up to eight pages long, including figures. Since 2009 an additional ninth page *containing only cited references* is allowed. Papers that exceed nine pages will not be reviewed, or in any other way considered for presentation at the conference.

Please note that this year we have introduced automatic line number generation into the style file (for LaTeX 2$\varepsilon$ and Word versions). This is to help reviewers refer to specific lines of the paper when they make their comments. Please do NOT refer to these line numbers in your paper as they will be removed from the style file for the final version of accepted papers.

The margins in 2015 are the same as since 2007, which allow for $\approx 15\%$ more words in the paper compared to earlier years. We are also again using double-blind reviewing. Both of these require the use of new style files.

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.

### 6.2 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 `nips2015.pdf` contains these instructions and illustrates the various formatting requirements your NIPS paper must satisfy. LaTeX users can choose between two style files: `nips15submit_09.sty` (to be used with LaTeX version 2.09) and `nips15submit_e.sty` (to be used with LaTeX2e). The file `nips2015.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 file `nips2015.rtf` is provided as a shell for MS Word users.

The formatting instructions contained in these style files are summarized in sections 7, 8, and 9 below.

# 7 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 of 11 points. Times New Roman is the preferred typeface throughout. Paragraphs are separated by 1/2 line space, with no indentation.

Paper title is 17 point, initial caps/lower case, bold, centered between 2 horizontal rules. Top rule is 4 points thick and bottom rule is 1 point thick. Allow 1/4 inch space above and below 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 9 regarding figures, tables, acknowledgments, and references.

# 8 Headings: first level

First level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 12. One line space before the first level heading and 1/2 line space after the first level heading.

## 8.1 Headings: second level

Second level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the second level heading and 1/2 line space after the second level heading.

### 8.1.1 Headings: third level

Third level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the third level heading and 1/2 line space after the third level heading.

# 9 Citations, figures, tables, references

These instructions apply to everyone, regardless of the formatter being used.

## 9.1 Citations within the text

Citations within the text should be numbered consecutively. The corresponding number is to appear enclosed in square brackets, such as [1] or [2]-[5]. The corresponding references are to be listed in the same order at the end of the paper, in the **References** section. (Note: the standard BIBTEX style `unsrt` produces this.) As to the format of the references themselves, any style is acceptable as long as it is used consistently.

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".

## 9.2 Footnotes

Indicate footnotes with a number[1] 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).[2]

## 9.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction; art work should not be hand-drawn. The figure number and caption always appear after the figure. Place one line

[1] Sample of the first footnote

[2] Sample of the second footnote

Table 1: Sample table title

| PART | DESCRIPTION |
|---|---|
| Dendrite | Input terminal |
| Axon | Output terminal |
| Soma | Cell body (contains cell nucleus) |

space before the figure caption, and one line space after the figure. The figure caption is lower case (except for first word and proper nouns); figures are numbered consecutively.

Make sure the figure caption does not get separated from the figure. Leave sufficient space to avoid splitting the figure and figure caption.

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

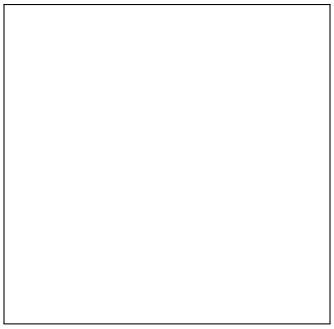


Figure 4: Sample figure caption.

### 9.4 Tables

All tables must be centered, neat, clean and legible. Do not use hand-drawn tables. 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.

## 10 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.

## 11 Preparing PostScript or PDF files

Please prepare PostScript or PDF files with paper size "US Letter", and not, for example, "A4". The -t letter option on dvips will produce US Letter files.

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 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`

- LaTeX users:
  - Consider directly generating PDF files using `pdflatex` (especially if you are a MiKTeX user). PDF figures must be substituted for EPS figures, however.
  - Otherwise, please generate your PostScript and PDF files with the following commands:

    ```
    dvips mypaper.dvi -t letter -Ppdf -G0 -o mypaper.ps
    ps2pdf mypaper.ps mypaper.pdf
    ```

    Check that the PDF files only contains Type 1 fonts.
  - xfig "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
  - The `\bbold` package almost always uses bitmap fonts. You can try the equivalent AMS Fonts with command

    ```
    \usepackage[psamsfonts]{amssymb}
    ```

    or 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
    ```
  - Sometimes the problematic fonts are used in figures included in LaTeX files. The ghostscript program `eps2eps` is the simplest way to clean such figures. For black and white figures, slightly better results can be achieved with program `potrace`.
- MSWord and Windows users (via PDF file):
  - Install the Microsoft Save as PDF Office 2007 Add-in from `http://www.microsoft.com/downloads/details.aspx?displaylang=en&familyid=4d951911-3e7e-4ae6-b059-a2e79ed87041`
  - Select "Save or Publish to PDF" from the Office or File menu
- MSWord and Mac OS X users (via PDF file):
  - From the print menu, click the PDF drop-down box, and select "Save as PDF..."
- MSWord and Windows users (via PS file):
  - To create a new printer on your computer, install the AdobePS printer driver and the Adobe Distiller PPD file from `http://www.adobe.com/support/downloads/detail.jsp?ftpID=204` *Note:* You must reboot your PC after installing the AdobePS driver for it to take effect.
  - To produce the ps file, select "Print" from the MS app, choose the installed AdobePS printer, click on "Properties", click on "Advanced."
  - Set "TrueType Font" to be "Download as Softfont"
  - Open the "PostScript Options" folder
  - Select "PostScript Output Option" to be "Optimize for Portability"
  - Select "TrueType Font Download Option" to be "Outline"
  - Select "Send PostScript Error Handler" to be "No"
  - Click "OK" three times, print your file.
  - Now, use Adobe Acrobat Distiller or ps2pdf to create a PDF file from the PS file. In Acrobat, check the option "Embed all fonts" if applicable.

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

### 11.1 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 using .eps graphics

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.eps}
```

or

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

for .pdf graphics. See section 4.4 in the graphics bundle documentation (`http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.ps`)

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

## 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 third 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 this year you can use a ninth page as long as it contains *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.