Pulsar Classification using Deep Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Pulsars are rapidly rotating, highly magnetized, neutron stars and white dwarfs that emit focused electromagnetic radiation in a beam. The beam is only visible to us when it is directly facing Earth and is the reason for their pulsed nature. This paper approaches classification of pulsars and non-pulsars with the intent of learning more about TensorFlow. The classification method used is Deep Neural Networks and an acceptable level of accuracy was obtained.

1 Introduction

We were tasked with picking a dataset and training a deep learning network. First thing to do was to pick a dataset. Being an open-ended assignment there was uncertainty as to what particular dataset to explore. Initial looks involved different image classification datasets such as CIFAR-10, Caltech 101, and NORB. Down the road there is no desire to work with image data classification, work with files and datasets of continuous and discrete data are preferred. After thorough search of the available datasets it was decided pulsar classification best suited this paper.

The last decision before pressing on to data format was to pick the correct deep learning framework for the job. There were several options but TensorFlow seemed like a good choice. Keras and PyTorch are other tools that utilize TensorFlow but were not used because those are often considered a "front end" for TensorFlow, which was not part of the goals of the assignment.

2 Method

2.1 Main

Main.py is where the project is ran from. There are several hyperparameters that a user can pick, beginning with $\_TEST\_PERCENTAGE$, which sets the percentage of comma-separated values (CSV) data to be utilized for testing. Industry standard is 20%, so that is the default setting. The training data $\_TRAIN$ is, as a result, 80% of the CSV data.

$$\_\mathcal{TRAIN} = \_NUM\_ELEMENTS * (1 - \_TEST\_PERCENTAGE)$$

It was necessary to hard-code in the number of elements in the CSV file because there was trouble when trying to run update on the CSV class due to an incomplete understanding of python abilities such as $class.update()$.

$\_BATCH\_SIZE$ is set as

$$\_\mathcal{BATCH\_SIZE} = max\{\_TRAIN * 0.1, 1\}$$

This is part of ensuring the training does not get stuck in local minima. _MODEL is then set as whichever optimizer the user would like, in this case, RMSPropOptimizer was selected. A CSV class is created based on the dataset to be classified.

Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017). Do not distribute.

Hyperparameter choices are part of the art of machine learning but in this project it was not the focus to make perfect hyperparameters. That being said, that was explored briefly for this project as seen in section 2.5. $\_NODES\_PER\_LAYER$ was one of such hyperparameters, as was $\_LAYERS$, and $\_HIDDEN\_LAYERS$. The final implementation of these parameters are as follows:

$$\_\mathcal{NODES\_PER\_LAYER} = (num\_features - 1) * 1.5$$

$$\_\mathcal{LAYERS} = max\{sqrt(\_NODES\_PER\_LAYER), 2\}$$

$$\_\mathcal{HIDDEN\_LAYERS} = [\_NODES\_PER\_LAYER] * \_LAYERS$$

This lead to 3 hidden layers with 12 nodes each for the HTRU_2 Dataset.

After all parameters have been setup main gets run. Users can select the number of runs to perform as an argument when they begin the program e.g.

python main.py *X*

The program will then run the classifier *X* times. After finishing the program will output classification accuracy of the given dataset.

2.2 Data format

After determining the dataset and the framework to be utilized it is necessary to figure out how to read data from the dataset. A modular approach was taken with this project. Users will be able to select a CSV file to load in. By specifying a class to contain the CSV data it is possible to run multiple datasets while making only minor modifications to the code. The format is also much easier to read as a result of the class format.

Rather than code up a custom method of moving CSV data into a format usable by the deep learning framework it was determined to "stand on the shoulders of giants". Pandas was used to bring CSV data into usable form. After pulling the data into a single structure it was necessary to split the data into training and testing data. Again, rather than code that from scratch it was determined best practice to utilize existing tools.

For that endevour scikit-learn was used, specifically the $train\_test\_split()$ method. As is standard for classification problems the data was segmented into 80% training and 20% testing data. It was then necessary to place the $class$ label into a data structure for later. For both the train and test data, associated labels are passed, along with the data, as tuples. The psuedocode for this section can be found under Algorithm 1.

Algorithm 1 $input\_fn()$

$data \leftarrow CSV$
$test, train \leftarrow train\_test\_split(data)$
$test\_label \leftarrow test.pop(class)$
$train\_label \leftarrow train.pop(class)$
return $(train, train\_label), (test, test\_label)$

2.3 TensorFlow estimator

TensorFlow has built in estimators such as:

- DNNClassifier
- BaselineClassifier
- BaselineRegressor
- DNNLinearCombinedClassifier
- LinearClassifier
- LinearRegressor

You also have the ability to build your own estimator, or classifier, and this is the route that was selected for this project. The custom estimator selected utilizes the $model$ specified at the start of the program, a $model\_dir$ for TensorBoard output which will be discussed in section 2.6, and the $params$ which consist of $feature\_columns$, $hidden\_units$, and $n\_classes$.

Estimator also requires three modes depending on the goal:

- ModeKeys.TRAIN
- ModeKeys.EVAL
- ModeKeys.PREDICT

Lastly, the estimator requires a return of $EstimatorSpec$. Documentation for the EstimatorSpec can be found on TensorFlow's website. It is important to note that each of the different $ModeKeys$ requires a different set of parameters for the returned $EstimatorSpec$.

2.4 Training

TensorFlow offers the ability to train estimators by calling a $train()$ method. For this paper the $input\_fn$ and $steps$ arguments are set by a function call to $train\_input\_fn$ and $csv.num\_examples['train']$ respectively. This allows the data to be segmented for training.

The model that was instantiated during estimator creation is detailed under model.model_RMSProp. For each of the layers that was passed to it under $params['hidden\_units']$ there is a fully connected layer built with selu activation. Klambauer et al. [2017] found with the proper math you can obtain self-normalizing exponential linear units, meaning they don't require batch normalization and that is why they were chosen over all of the other activation functions. There is also the final fully connected layer which has no activation layer. This final layer aids in determining the classification.

During training ModeKeys.TRAIN is called, which specifies RMSProp gradient descent as our optimizer and then calls $minimize()$. The $minimize()$ method will automatically compute the gradients and apply them to the variables, but if you need to tune the network there are additional steps that can be taken instead:

1. Compute the gradients with $compute\_gradients()$
2. Process the gradients as you wish
3. Apply the processed gradients with $apply\_gradients()$

Further documentation on this process can be found on TensorFlow's website under the class Optimizer. It was not necessary to have that level of fidelity with this particular dataset, so the base $minimize()$ was all that was necessary.

With $minimize()$ minimizing the loss we are able to make progress towards a classification. This dataset in particular minimizes loss to near-zero after just the first episode of 100 training steps as can be seen in Figure 5.

2.5 Hyperparameters

Hyperparameters are normally very important but in this dataset it was discovered that they had very little impact on the efficacy of this classifier.

An example of hyperparameter randomization can be found under:

python hyperparams.py

That file will automatically generate randomized hyperparameters. During testing of this project various hyperparameters were tested with no significant improvements over the initial parameters set so the associated code was removed from primary operation of the classifier. The file described would allow for random testing of hyperparameters across any number of computers. Bergstra and Bengio [2012] found random search of hyperparameters to be superior to grid search and was the method used while exploring hyperparameter tuning during this project.

2.6 TensorBoard

One of the other reasons for utilizing TensorFlow is the TensorBoard project. TensorBoard allows for visualization of various parameters as defined by the user. Figure 1 and 2 are the same results as shown in Figure 4 and 5 respectively. The smooothing helps in understanding what may be going on, especially in more turbulant graphs. Another benefit is being able to visualize the graph structure. To use TensorBoard on this project ensure it was installed with TensorFlow and type

tensorboard --logdir model

and go to the link it provides (by ctrl clicking or copy/paste).

Figure 1: Smoothed Accuracy

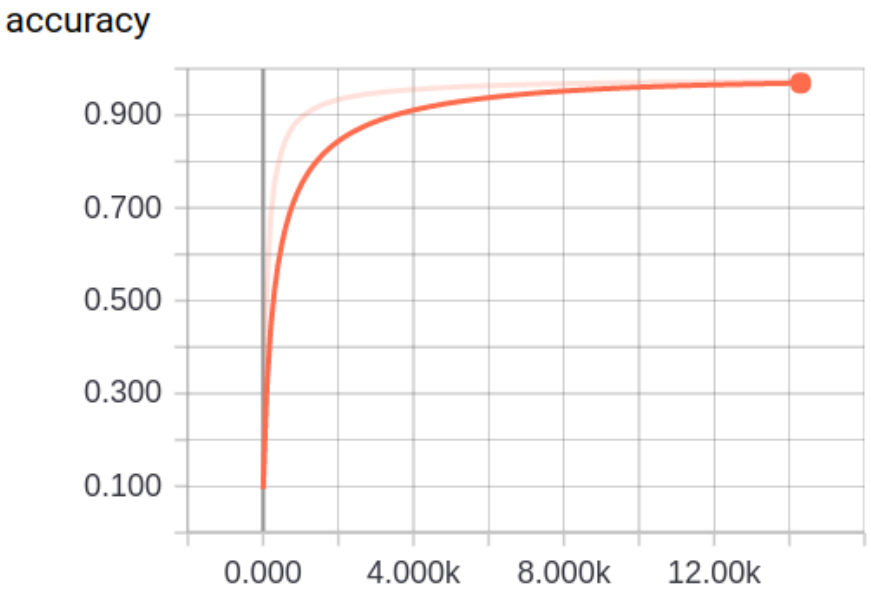
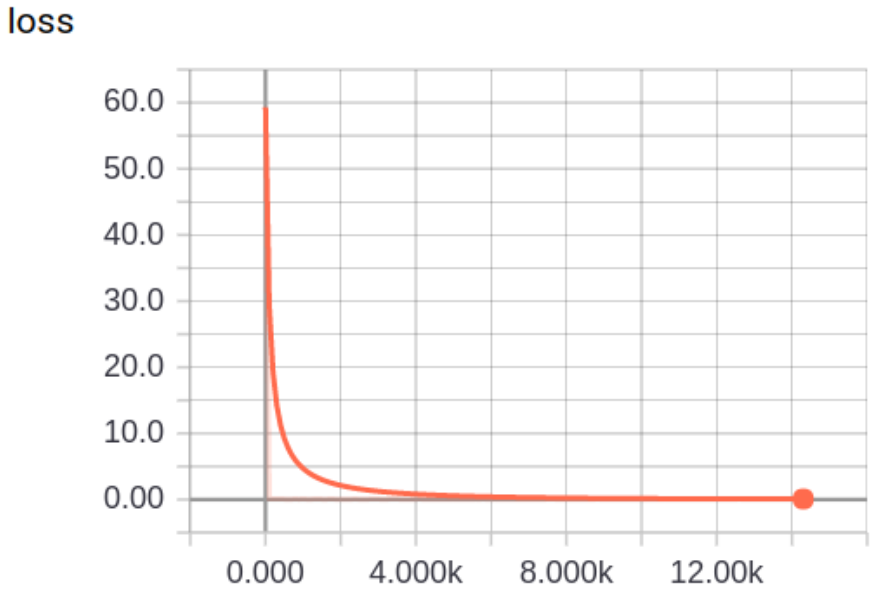


Figure 2: Smoothed Loss



3 Results

RMSProp gradient descent with 0.001 learning rate, performing batch runs on 10% of the training samples every 100 steps produced an accuracy of 97.8% averaged over 20 tests. These were taken with 80% train and 20% test samples. Other hyperparameters were three fully connected layers with 12 nodes each using selu activation.

Figure 3: Accuracy

```
INFO:tensorflow:Finished evaluation at 2018-04-19-01:46:51
INFO:tensorflow:Saving dict for global step 14318: accuracy
Accuracy =  0.978687149286
kyle@:ResearchProject_HTRU2$
```

Figure 4: Accuracy

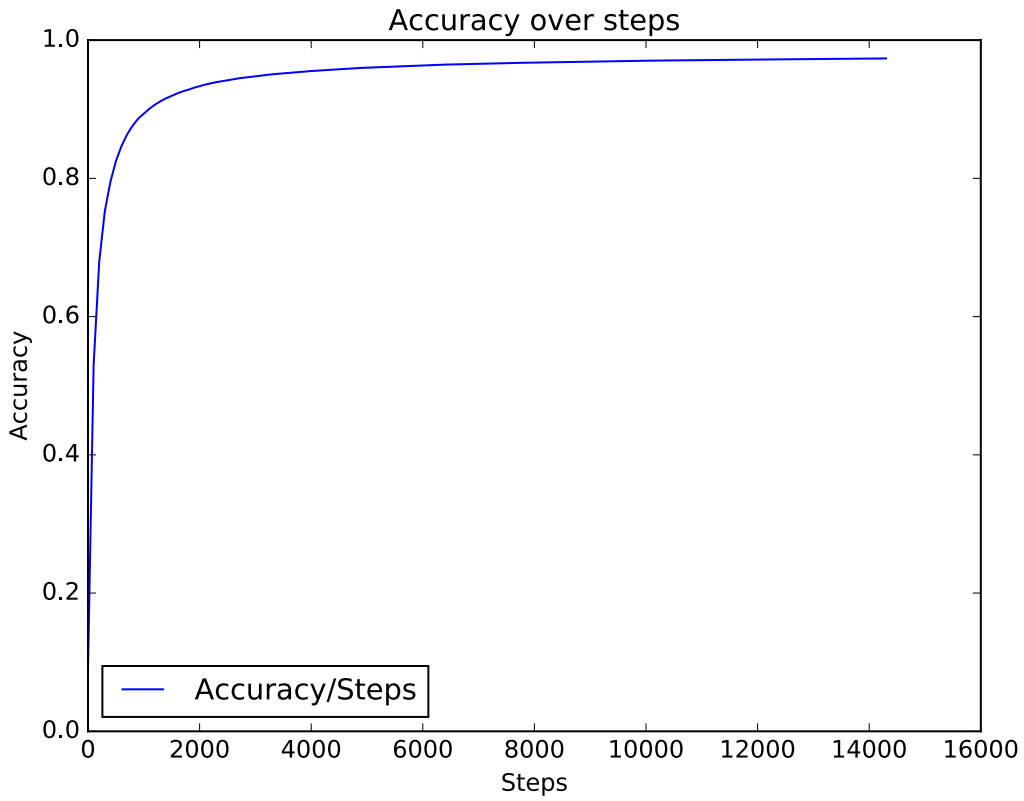
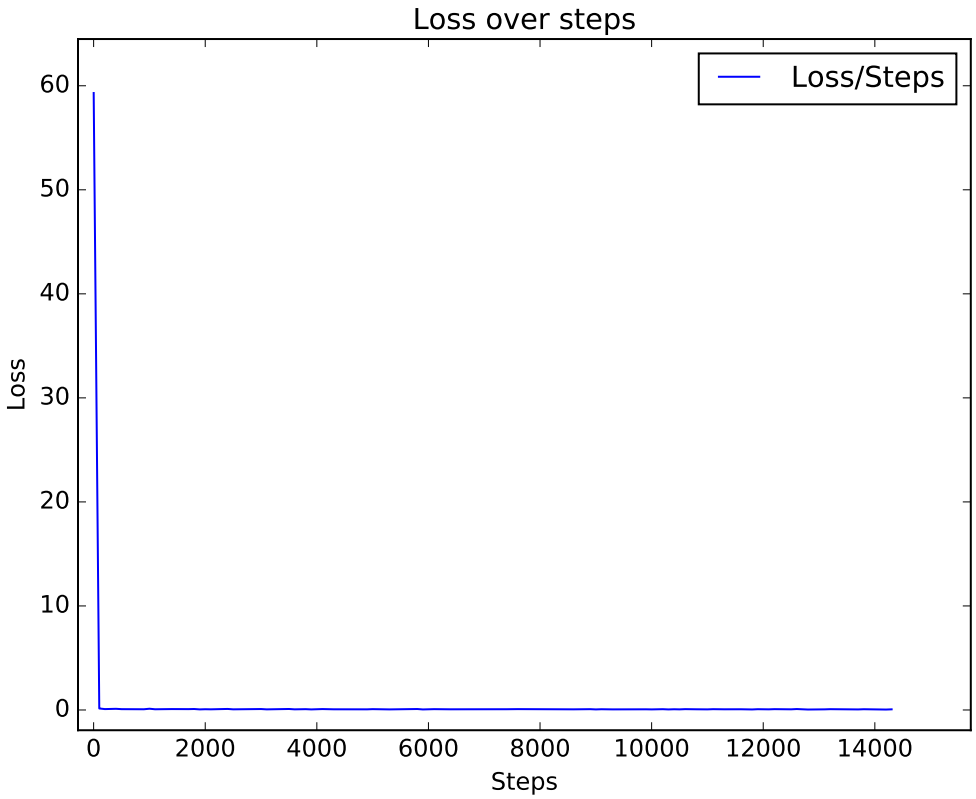


Figure 5: Loss



4 Conclusion

5 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 7 regarding figures, tables, acknowledgments, and references.

6 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.

6.1 Headings: second level

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

6.1.1 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.

7 Citations, figures, tables, references

These instructions apply to everyone.

7.1 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."

7.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, 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).

Note that footnotes are properly typeset *after* punctuation marks.[2]

[1] Sample of the first footnote.

[2] 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

7.3 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.

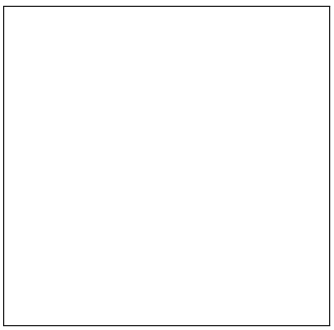


Figure 6: Sample figure caption.

7.4 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.

8 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.

9 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.

9.1 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.

References

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Machine Learning Research*, 13(10):281–305, 2012. doi: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *arXiv*, 2017. doi: https://arxiv.org/pdf/1706.02515.pdf.