DOI: 10.15439/2018F140
ISSN 2300-5963 ACSIS, Vol. 15

A neural framework for online recognition of handwritten Kanji characters

Małgorzata Grębowiec, Jarosław Protasiewicz
National Information Processing Institute, Warsaw, Poland

***Abstract*—The aim of this study is to propose an efficient and fast framework for recognition of Kanji characters working in a real-time during their writing. Previous research on online recognition of handwritten characters used a large dataset containing samples of characters written by many writers. Our study presents a solution that achieves fine results, using a small dataset containing a single sample for each Kanji character from only one writer. The proposed system analyses and classifies the stroke types appearing in a Kanji and then recognises it. For this purpose, we utilise a Convolutional Neural Network and a hierarchical dictionary containing Kanji definitions. Moreover, we compare the histograms of Kanjis to solve the problem of distinguishing character having the same number of strokes of the same type, but arranged in a different position in relation to each other. The proposed framework was validated experimentally on online handwritten Kanjis by beginners and advanced learners. Achieved accuracy up to 89% indicates that it may be a valuable solution for learning Kanji by beginners.**

I. Introduction

KANJI, along with the syllabic kana - hiragana and katakana, belong to the Japanese writing system. They are adopted logographic Chinese characters and literally mean "Han characters" (漢字). The number of Kanji characters is vast and amounts to over 500,000 [1]. However, in order to understand Japanese newspapers and books, it is usually enough to know 2,136 of *jouyou kanji* from the official list of Kanji determined by the Ministry of Education of Japan in 2010.

A large number of Kanji characters and a complex structure could require a lot of time to learn them. One of the solutions that may help the students in that may be an intelligent application. Its main idea is that it analyses online Kanji characters when a student is writing them stroke by stroke. The analysis relies on checking the correctness of the drawn characters or suggesting their meaning. To make this possible, such a system has to recognize the handwritten Kanjis online.

There are several approaches to this issue. Early solutions for online recognition of handwritten Kanji presented in [2] emphasize the similarity of the sequence of pen movements to the problem of speech recognition. Due to this similarity, they utilised a Hidden Markov Model (HMM) for a sub-stroke of Kanji and built a hierarchical dictionary defining the characters' structure. For further improvement of the recognition accuracy, [3] used pen pressure to propose writer-independent handwriting recognition. The further studies of [4], [5] have expanded this solution, taking into account the relative position of the strokes in Kanji. Study of [6] also is based on HMM recognition for structured character pattern representation.

We have to underline that the approaches mentioned above deal very well with the issue of Kanji recognition. However, some improvements may be required to gain better efficiency, especially when considering a solution that is able to work in real-time. One of the above approaches refers to a hierarchical dictionary containing definitions of Kanji characters. Creating such a dictionary required manual preparation of rules that could contain errors, and it was a time-consuming task. Further work on its construction could include automatic generation of rules that would be helpful in adding new definitions to the dictionary. Current research assumes that strokes in Kanji characters are drawn in the correct order, considering that the errors could occur while writing a character. Thus, new studies should include this problem to develop algorithms that are less sensitive to strokes order.

Having in mind these two problems, namely (i) the definition dictionary of Kanji characters and (ii) sensitivity of algorithms to strokes order, we propose a neural framework for online recognition of handwritten Kanji characters. More specifically, the main objectives of this study are as follows:

1) To propose a framework for online recognition of handwritten Kanji characters utilising convolutional neural networks, which are currently one of the state-of-the-art approaches in image recognition.
2) To automatically generate a dictionary containing definition of Kanji characters, which simple construction allows to easily supplement it with new definitions and its further development.
3) To assess the effectiveness of proposed solutions by experiments carried out using own implementation of the framework.

The novelty of our research is based on the implementation of a framework that allows to suggest Kanji characters in real time. The user can draw a character from the first stroke to the last and the system returns a sorted list of proposed Kanji characters after each line drawn. The results returned by the system are sorted by a measure of similarity between the character being drawn and the characters contained in the Kanji dictionary. This makes it possible to find the target character even before finishing the drawing of all the strokes.

The remainder of this paper is as follows. Section II introduces the problem to solve and presents the proposed framework for Kanji recognition. Next, Section III validates the framework performance with illustrative examples. Finally, the findings are concluded, and references are provided.

II. Kanji recognition framework

In this section, we define the problem of online recognition of handwritten Kanji characters, and we propose and describe the framework that solves the indicated issue.

A. Problem definition

Let us assume that a single Kanji character, K consists of n strokes:

$$K = \{s_1, ..., s_i, ..., s_n\}. \tag{1}$$

Information about the number of strokes which compose a character is not sufficient to recognise the Kanji correctly. For a given stroke s_i, it is required to specify its type s_t, order s_o in which it appears in the sign K, and its position s_p in relation to other strokes:

$$s_i = (s_t, s_o, s_p) \tag{2}$$

The examples illustrating the described problem are the signs 犬 (dog) and 太 (fat) . Each of them consists of four lines. There are many more signs that are built by using the same number of strokes. Additional information about the strokes type is also not sufficient to indicate correctly the character. Although they have the same number of dashes, and they are of the same type written in the same order, the last short slashing line is in a different position in each of these Kanjis. Therefore, information about the placement of each stroke in a character is an issue that must be considered.

The task is to recognise the character, K during writing its strokes s_i, $i = 1, ..., n$, including their parameters s_t, s_o, s_p.

B. Framework overview

The simplified construction of our framework that tries to solve the above task is presented in Figure 1.

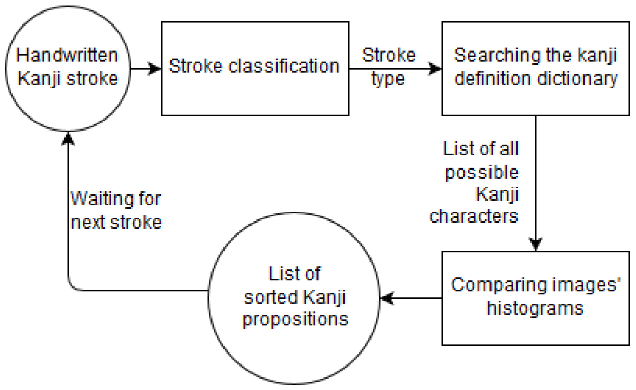
Handwritten Kanji stroke
Stroke classification
Stroke type
Searching the kanji definition dictionary
List of all possible Kanji characters
Comparing images' histograms
List of sorted Kanji propositions
Waiting for next stroke

Figure 1. Overview of the framework for online recognition handwritten Kanji characters.

In the first step, after receiving the first handwritten stroke, the system saves it to a PNG file and tries to classify it to a proper type. The classification of the stroke type is done by a convolutional neural network. Having classified the stroke type, the system searches the Kanji definitions dictionary. This dictionary covers definitions of all Kanjis from our dataset. The result of that is a list of all Kanjis that start with the previously classified type. Since the list of such possible characters is extensive, in the next step, the system tries to sort them by the most likely ones. For this purpose, we compare the histograms of Kanjis' images returned by the dictionary with the image containing the handwritten strokes. Based on these comparisons, the list of Kanjis is sorted by the most similar to the drawn character. When the next stroke appears, the whole process is repeated from the beginning as described above by keeping the order of drawing lines. Finally, a sorted list of the proposed Kanji characters is returned.

Detailed implementation of individual modules are presented in the following subchapters.

C. Classification method

We utilise a Convolutional Neural Network (CNN) to classify Kanji strokes. The CNN contains several layers processing signals feed-forward. In the input layer ($INPUT$), neurons are arranged in three dimensions (width, height, depth) to process pictures. They are transferred through the set of hidden layers which are of three types, namely: (i) a convolutional layer ($CONV$), (ii) a pooling layer ($POOL$), and (iii) a fully-connected layer (FC) with an ReLU activation function. These layers produce class scores z. The architecture of the networks can be simplified as follows:

$$\begin{array}{c} INPUT \rightarrow (CONV * N \rightarrow POOL) * M \rightarrow \\ FC * K \rightarrow softmax \end{array}, \tag{3}$$

where the asterisk, $*$ indicates repetition N, M, K times respectively. In the final layer, $softmax$ the vector of class scores, z returned from the last fully-connected layer are integrated using the following softmax function:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}. \tag{4}$$

D. Kanji dictionary

The Kanji dictionary is constructed from the definitions of 7,365 Kanjis appearing in the dataset of files describing the characters. We extract all strokes from each file that maintain information about the appearance order of strokes. The dictionary is based on a tree structure, in which the number of trees is determined by the number of stroke types. The first stroke in the character determines the selection of trees to which its definition will be saved. Each subsequent stroke forms the next node in this tree. Only unique stroke types can exist at the same depth of the tree. An example of Kanji definitions is presented in Figure 2.

E. Histogram comparison

The list of suggested Kanjis returned by the dictionary may be extensive. Moreover, they are not sorted according to the similarity to the character under examination. To solve this problem, we decided to use the comparison of histograms as the similarity measure of images.

A histogram is the graphical representation of the tonal distribution in a digital image. To compare two histograms (H_1,H_2), we choose Chi-Square as a distance measure $d(H_1, H_2)$ which evaluates how well both histograms match:

$$d(H_1, H_2) = \sum_I \frac{(H_1(I)) - H_2(I))^2}{H_1(I)} \qquad (5)$$

The smaller the distance between histograms is, the more similar they are. By calculating the distance between the image containing drawn lines and the images representing Kanjis returned by the list, we can sort the Kanji list according to their similarity to the tested image.

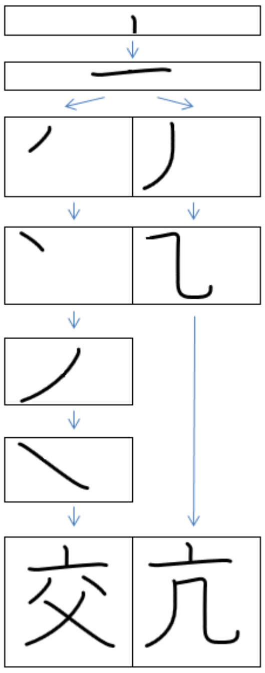

Figure 2. An example of Kanji definitions stored in the dictionary. The straight vertical stroke at the top is the root of a tree. Each subsequent stroke is added to the dictionary as a new node. If a stroke duplicates on a given depth of the tree, it is not added again to the tree.

III. Experiments

In this section, we (i) characterise the dataset for experiments, (ii) evaluate various configurations of the CNN network classifying the stroke types, and (iii) test online recognition of Kanjis written out by two groups of testers.

A. Dataset

The experiments are based on the KanjiVG database, which contains descriptions of Kanji, hiragana, and katakana. In this study, we use only Kanji, namely the version, kanjivg-r20160426 containing 7,365 characters. Each of them is described as an SVG file, which is a universal format of two-dimensional vector graphics. Essential information in this database is the order of strokes from which a particular Kanji character is constructed. Each stroke type is labelled by an identifier. Figure 3 shows 26 stroke types available the database, and samples size for each type. Unfortunately, the distribution of strokes over their types is unequal.

B. Parameters of the Convolutional Neural Network

To select an optimal classifier of Kanji strokes, we decided to test three configurations of a CNN. They are based on the architectures proposed by [7], [8] and our pre-experiments. Table I depicts the architectures in detail, namely CNN1, CNN2, and CNN3. Each of them contains several convolutional layers followed by an activation layer and a max-pooling layer. Next, there are up to three fully-connected layers in which the final result is calculated by a softmax classifier.

Table I
CONFIGURATIONS OF CNN WHICH WHERE EVALUATED. EACH COLUMN CORRESPONDS TO A LAYER OF THE NETWORK, E.G. CONV3-32 STANDS FOR THE CONVOLUTIONAL LAYER WITH KERNEL 3X3 AND 32 FILTERS, FC-1024 STANDS FOR THE FULLY CONNECTED LAYER WITH SIZE 1024.

CNN1	**CNN2**	**CNN3**
input (108 x 108 gray-scale image)		
conv3-32	conv3-64	conv3-64
maxpool	maxpool	maxpool
conv3-64	conv3-128	conv3-128
maxpool	maxpool	maxpool
	conv3-192	conv3-192
	maxpool	maxpool
	conv3-256	conv3-256
	maxpool	maxpool
		conv3-512
		maxpool
FC-256	FC-1024	
FC-n classes		
softmax		

An input layer holds the raw normalised pixel values of an image with the the size of $108x108$ pixels with one colour channel. All convolution layers compute by using the kernel with the size of $3x3$ with the stride equal to 1. The number of filters varies from 32 to 512. All max-pooling layers utilise kernels with the size of $2x2$ with the stride of two downsamples. The fully-connected layers are the size of 256 or 1024 outputs. However, the last fully-connected layer has the same number of outputs as the number of classes. It is followed by the softmax classifier. Each of fully-connected and convolution layers contains a ReLU non-linear layer with a dropout equal to 0.5 [9].

All the networks were trained by using the Adam optimiser with the parameters equal to $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\epsilon = 10^{-8}$ and the learning rate equal to 10^{-4} [10]. The weights were initialised by the Glorot uniform initializer, which is also called the Xavier uniform initializer [11]. The size of mini-batch of 16 was performed. For training, we use 80 of all samples in dataset, 20% for validation at each epoch, and an untrained 20% of examples were used for testing. The models were implemented by using the Keres interface.

C. Stroke classification

The first set of experiments involved the selection of an optimal model for Kanji strokes classification to their types. The results of experiments are covered in Table II, which includes typical classification quality indicators such as by accuracy, precision, recall, and F1-score.

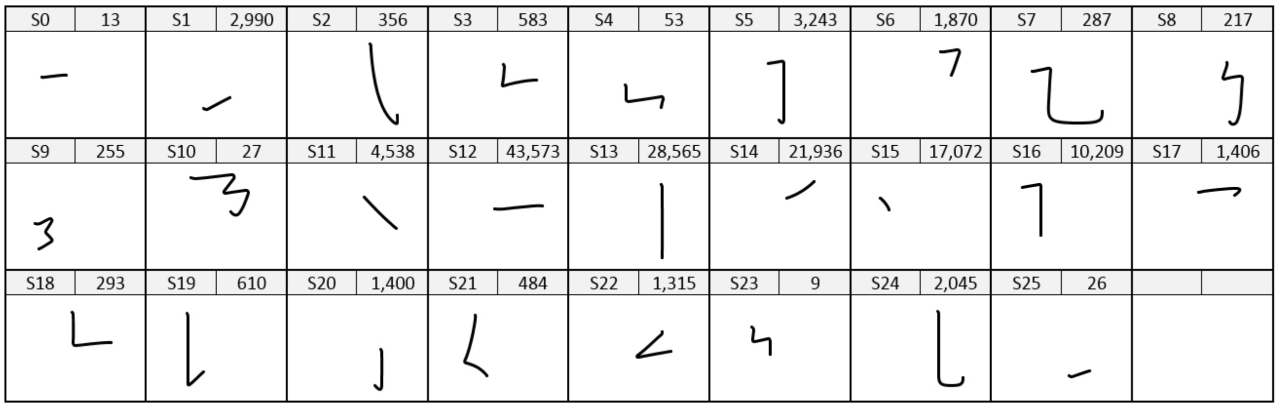

Figure 3. Kanji stroke types available in the KanjiVG database, and distribution of stroke examples over the stroke types.

We implemented and tested three network architecture configurations, namely CNN1, CNN2, and CNN3, which are described in Subsection III-B. We also used a callback function for monitoring the training process. It works as follows: if the loss function does not improve in two epochs, the process is interrupted, and the algorithm does not execute the next epoch. The results of these experiments are covered in the section, "Configuration selection" in Table II. A simple construction of the Kanji character strokes suggested starting from a simple architecture of the convolution neural network (CNN1). However, we decided to increase the number of network parameters by adding more layers, in order to see if this would improve the classification results. Subsequent attempts to deepen the network (CNN2, CNN3) did not bring any improvement. Moreover, the additional complexity of the network configuration adversely affected the network learning time. Comparing the obtained results, we decided to use the CNN1 network in the succeeding experiments.

Initially, the only element of data preprocessing was normalisation to the range of (0,1). Additionally, we utilised a ZCA (Zero Components Analysis) whitening transformation of the input images. It is a linear algebra operation that reduces redundancies in the matrix of pixel images. The operation is intended to better highlight structures and features in images for the learning algorithm. However, in our case, the improvements were insignificant (see the results in the section, "Additional ZCA whitening transformation" in Table II). Thus, we decided to skip this technique not to increase the complexity of the algorithm.

Due to the relatively small number of samples in the case of some types of strokes, e.g. S0, S4, S10, S23, we tried to extend the dataset by more samples. Unfortunately, the image augmentation techniques may not produce the expected results. This is due to the characteristic construction of Kanji characters, in which even a slight rotation of the image could change a stroke type. A good example highlighting this problem is the case of S11 and S12. Turning the stroke S11 in too high angle to the left can cause it to become too similar to the stroke S12. A similar case appears in the pair of strokes S0 and S1. These concerns have been confirmed in the research. Setting the rotation range parameter to 20 degrees degraded the results compared to the original dataset (see the results in the section, "Sensitivity to rotation" in Table II).

Looking at the shape of 26 classes, we noticed that several pairs of strokes are very similar in regard to each other. We combined the most similar stroke types by reducing the number of classes from 26 to 12. Table III presents the stroke types that we decided to merge into new classes. This approach allowed to achieve the best results in comparison to all previous experiments (see the results in the section, "Classes reduction" in Table II).

Table II
RESULTS OF THE EXPERIMENTS INVOLVING THE SELECTION OF AN OPTIMAL MODEL FOR KANJI STROKES CLASSIFICATION TO THEIR TYPES.

Architecture	Image augmentation	Accuracy	Precision	Recall	F1-score
Configuration selection					
CNN1	- normalization	96,14	96,14	96,14	96,11
CNN2	- normalization	95,79	96,23	95,25	95.72
CNN3	- normalization	94,91	95,34	94,55	94.93
Additional ZCA whitening transformation					
CNN1	- normalization - ZCA whitening	95,97	95,96	95,98	95.93
Sensitivity to rotation					
CNN1	- normalization - range rotation	93,24	94,00	92,31	93.12
Classes reduction					
CNN1	- normalization - 12 classes	**96,89**	**96,92**	**96,89**	**96,89**

Table III
LISTING OF THE NEW MERGED CLASSES.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
S0 S12 S17	S1 S14 S25	S2	S3 S18	S4 S8 S23	S5 S6 S16	S7	S9 S10	S11 S15	S13 S19 S20	S21 S22	S24

D. Kanji recognition

The second set of experiments involved Kanji characters recognition in real-time during their writing by an individual. In order to perform these tests, we have created a simple web interface. It contains a space, in which a user can draw Kanji by using a mouse cursor. The real-time tests involved two groups of individuals, namely (i) ten people who have never studied Japanese before, (ii) one expert familiar with Japanese language and Japanese calligraphy. Each tester had to draw 100 randomly selected Kanji characters. Prior to that, each non-expert individual had performed a small tutorial, which instructed how to write ten Kanjis.

During the tests, especially with non-expert group, we encountered two problems. The first issue resulted from unfamiliarity of Kanji structure by testers. Some individuals were not able to notice differences between some strokes, because they were so similar in regard to the others. The second issue was of technical nature. Our tests were carried out on a laptop with a mouse device. Testers complained about the shape of the mouse device, because it was too small and uncomfortable to use. It affected the quality of drawn lines. Users could not draw in the way that they wanted to. We were suggested that it would be more convenient to do tests on a tablet with a pen device.

It has to be noted that we assumed that a target Kanji was correctly identified if the following conditions had been met:

1) The target character had to appear in the first five characters proposed by the system.
2) A tester had only two attempts to draw the mark; any subsequent attempts were not taken into account.

In spite of the problems mentioned above, we managed to obtain the following results. 79% of the drawn Kanji characters by the non-experts were correctly recognised by the system. For the expert's drawings, the system recognised correctly 89% of characters. The detailed results showing the number of correctly recognised Kanjis are in Table IV.

Table IV
THE NUMBER OF CORRECTLY RECOGNISED KANJIS IN RESPECT TO A POSITION ON WHICH THEY WERE PROPOSED BY THE SYSTEM (IT PROPOSES FIVE THE MOST LIKELY KANJI CHARACTERS).

Group of testers	Position					Sum
	1st	2nd	3rd	4th	5th	
Non-experts	43	18	7	5	6	79%
Expert	78	9	1	1	0	89%

In both groups, the correctly recognised character was, in most cases, in the first position of the suggested Kanjis. In only one case for the non-expert group, the correctly identified Kanji was farther than the fifth position. In other cases, the type of stroke was incorrectly classified so that the Kanji could not be recognised. The most frequent mistakes occurred between lines S0 and S1 and S9 and S1. An imprecisely drawn line caused an incorrect classification. Due to this, searching of a dictionary with the definitions of Kanji did not bring the expected result. Another encountered problem, which also affected the results, consisted of incorrectly marked strokes in the KanjiVG database. We had found that some of the characters in our sets for testers had a stroke marked as S8, when it should be marked as S1.

IV. CONCLUSIONS

In the study, we presented and implemented a complete framework that recognises Kanji characters in a real-time, based on successively drawn strokes. In comparison to the previous works, our solution did not assume writing characters in cursive. For this reason, the comparison of prior results with those achieved by us is not entirely reliable. However, we were able to find out that even the characters written by a group that has never before had contact with the Japanese writing system can be correctly recognised by the system. Another advantage of our solution is the uncomplicated construction of the system, as well as an automatically generated dictionary definition of Kanji, which can be easily expanded with new meanings. The problems encountered during the tests included the incorrect classification of strokes, which in the training set were present in a small amount. In order to improve the quality of CNN classification, it would be necessary to expand the collection in the least numerous classes. One of the limitations mentioned in earlier studies was the problem of writing strokes in the wrong order. This will be the subject of our further research.

REFERENCES

[1] T. Morohashi, *Dai Kan-Wa jiten*. Tokyo : Taishukan Shoten, Showa 59-61, 1986.

[2] M. Nakai, N. Akira, H. Shimodaira, and S. Sagayama, "Substroke approach to hmm-based on-line kanji handwriting recognition," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 491–495. [Online]. Available: http://dx.doi.org/10.1109/ICDAR.2001.953838

[3] M. Nakai, T. Sudo, H. Shimodaira, and S. Sagayama, "Pen pressure features for writer-independent on-line handwriting recognition based on substroke hmm," in *Object recognition supported by user interaction for service robots*, vol. 3, 2002, pp. 220–223. [Online]. Available: http://dx.doi.org/10.1109/ICPR.2002.1047834

[4] J. Tokuno, M. Nakai, H. Shimodaira, S. Sagayama, and M. Nakagawa, "On-line Handwritten Character Recognition Selectively Employing Hierarchical Spatial Relationships among Subpatterns," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, G. Lorette, Ed., Université de Rennes 1. La Baule (France): Suvisoft, Oct. 2006, http://www.suvisoft.com. [Online]. Available: https://hal.inria.fr/inria-00104751

[5] I. Ota, R. Yamamoto, S. Sako, and S. Sagayama, "Online handwritten kanji recognition based on inter-stroke grammar," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, Sept 2007, pp. 1188–1192. [Online]. Available: http://dx.doi.org/10.1109/ICDAR.2007.4377103

[6] M. Nakagawa, J. Tokuno, B. Zhu, M. Onuma, H. Oda, and A. Kitadai, "Recent results of online japanese handwriting recognition and its applications," in *Arabic and Chinese Handwriting Recognition*, D. Doermann and S. Jaeger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 170–195.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[8] C. Tsai, "Recognizing handwritten japanese characters using deep convolutional neural networks," 2015.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://dx.doi.org/10.1145/3065386

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[11] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: http://dx.doi.org/10.1.1.207.2059