KEYWORD DETECTION FOR THE ACTIVATION OF SPEECH DIALOGUE SYSTEMS

Hans-Günter Hirsch[1], *Alexander Micheel*[1], *Michael Gref*[1,2]
[1]*Niederrhein University,* [2]*Fraunhofer Institut IAIS*
hans-guenter.hirsch@hs-niederrhein.de

Abstract: The detection and recognition of a spoken keyword is an adequate and comfortable method to activate a speech dialogue system. A low false acceptance rate (FAR) is needed to avoid the erroneous activation of a speech assistant and the erroneous activation of a speech controlled device as consequence of recognizing a legal command after the keyword. The false rejection rate (FRR) should also be low to guarantee good user acceptance. Often, the keyword recognition has to be realized in an embedded system with limited computational resources. Therefore, the detection and recognition algorithm has to fulfill the requirements of a low FAR and FRR on the one hand and the need of a low computational load on the other hand. We designed a two stage algorithm to meet these expectations. The first stage consists of a GMM-HMM (Gaussian Mixture Model - Hidden Markov Model) based recognizer with one or several HMMs for the keyword and a set of so-called filler HMMs to model speech segments that do not contain the keyword. To reduce the FAR of the first stage, the MEL spectrum of the pretended keyword segment is analyzed by employing a neural network. The task of the neural network as second stage of the recognition process is either to accept or the reject the keyword as pretended in the first stage. It turns out that the FAR of the first stage can considerably be reduced by the second stage.

1 Introduction

The recognition of a predefined keyword is an essential feature in improving the usability and the acceptance of a speech assistant system. Instead of pressing a button, the keyword is used to wake up the speech dialogue system and to enable further speech input, e.g. to address an information inquiry or to control a device as a component of a home automation system. The voice activation brings about the need of permanently listening for a spoken keyword. The task is the detection of segments in a continuous audio stream that may contain the keyword. Pattern matching is needed to classify the segment as belonging to one of two categories, keyword or non-keyword. But this classification has to work extremely reliable. There should be no false acceptance of an audio segment that does not contain the keyword but a similar sounding sequence of phonemes. In case of an automation system, this could lead to false activation of a device. To avoid such a false acceptance, the keyword should be chosen carefully. It should contain a sequence of phonemes that usually does not occur in the common vocabulary of a language. Furthermore, it can be of advantage to have a phoneme with high energy, usually a vowel, at the beginning and/or at the end of the keyword so that the word boundaries can be detected better in a noisy background. For example, this is the case with the keyword "Alexa" used by the Amazon devices.

The usual setup of a keyword detection was based on the parallel usage of a Hidden Markov Model (HMM) for the keyword and a set of so-called filler models. Quite often, a set of monophone or triphone HMMs has been taken as filler models [1], so that all speech segments not containing the keyword could be modeled as a sequence of the corresponding phonemes. Nowadays, neural networks are applied to detect a keyword by taking a sequence of feature vectors as input for the network. Different types of neural networks from multi-layer perceptrons over networks with convolutional layers up to recurrent networks like LSTM (long short-term memory) [2, 3, 4] have been trained on the specific task to find and detect the keyword within the continuous stream of feature vectors.

Nowadays, most of the speech recognition systems work in a distributed setup. The speech is recorded and played back by a client system like e.g. the Amazon Echo or the Google Home devices. Probably, the speech recognition is done with an extremely powerful set of server machines somewhere in the cloud to allow the application of neural networks as part of the pattern matching. Therefore, the speech utterance that should be recognized has to be transmitted from the client to the server via the internet. In the opposite direction, the audio output has to be transmitted to realize a speech dialogue. When using the distributed setup also for the recognition of the keyword, this would cause a continuous transmission of the recorded audio stream. To avoid this continuous transmission, devices like Amazon Echo or Google Home usually realize the keyword detection on the client system. Google has presented an approach where they combine a keyword detection on the client side with a keyword verification on the server side [5]. This approach can be applied in case the keyword is spoken in combination with a speech inquiry or command. The computational power on the client side is usually much lower than on the server side. Especially when applying a neural network a high number of multiplications is needed to realize the continuous detection of the keyword. In [6] the relationship between the network topology and the number of multiplications as well as the power consumption of a Raspberry Pi device is presented.

In our application, we use keyword detection as component of a speech recognition system in the field of home automation. The recognition of the commands for controlling devices at home will be realized on a server that is located inside the house. The recorded microphone signal will not leave the home to take care of data privacy. The speech input will be done with very small devices that could be placed in standard outlet sockets for example. The client as well as the server system have to be realized very cost-effectively. Furthermore, the client system has to work with low power consumption. Both aspects, low energy and low cost, were the main requirements for the design of a keyword detection algorithm that should work reliably with low computational resources. We keep in mind a realization with small computer devices like the PI-Zero for example.

Based on an earlier approach [7] we are presenting a modified algorithm in this paper. We could improve our first algorithm with respect to an easier implementation on devices with low computational resources and with respect to a higher recognition performance under noisy conditions.

2 Algorithm

The algorithm for the detection and the recognition of the keyword consists of two stages as shown in Figure 1. Below, we will present details about the extracted acoustic features, the GMM-HMM recognition and the neural network.

2.1 Feature Extraction

The speech signal is sampled at a rate of 16 kHz. The short term DFT spectra are calculated for frames containing 400 samples (= 25 ms) every 10 ms after applying a preemphasis filtering and weighting the 400 samples of each segment with a Hamming window. The 400 filtered and

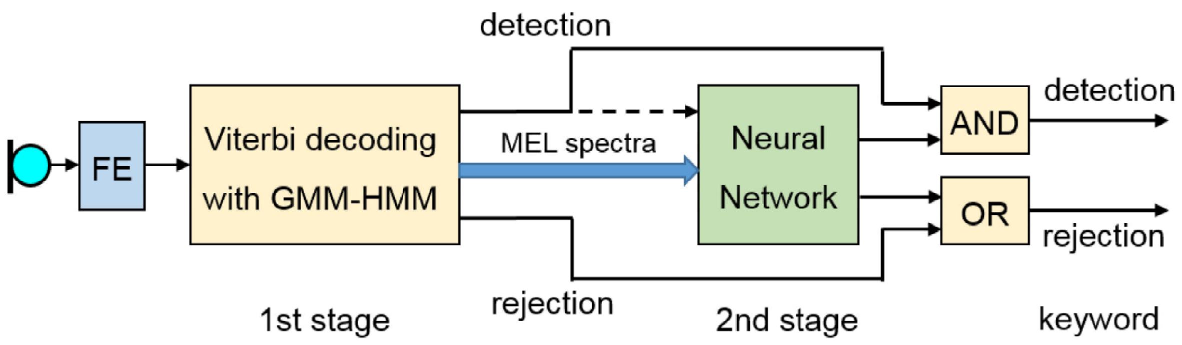

Figure 1 – Two-stage keyword recognition

weighted samples are transformed by means of a DFT with length 512. In our first approach [7] we applied a noise reduction algorithm to the DFT spectrum. The intention was the reduction of stationary background noise based on an adaptive filtering in the frequency domain [8]. However, in the practical implementation and realization it turned out, that the estimation of the noise spectrum as well as the speech artifacts introduced by the adaptive filtering can have a negative impact on the keyword recognition in noisy conditions. Furthermore, we can reduce the needed processing power by omitting the filtering stage. The short term logarithmic energy logE is calculated by taking the logarithm of the sum of the squared DFT magnitude coefficients in the range between 200 Hz and 7 kHz. Furthermore 12 cepstral coefficients C1 to C12 are determined by transforming the logarithmic MEL spectrum with a DCT. 36 MEL filters have been defined in the range from 200 Hz to about 7 kHz to calculate the MEL spectrum from the magnitude DFT coefficients. The Delta coefficients (ΔlogE, ΔC1, .., ΔC12) and the second derivative of the energy contour ΔΔlogE are calculated according to the filtering scheme defined in [9]. The vector containing the 26 components (C1, ..., C12, ΔlogE, ΔC1, .., ΔC12, ΔΔlogE) is used as feature vector for the GMM-HMM recognizer. The energy coefficient logE is omitted due to its varying value in case of background noise.

2.2 Keyword HMMs

We selected the personal name "Esmeralda" as keyword for our investigations that is rarely used as a given name in Germany and it rarely occurs in German conversations. The word begins and ends with a vowel. It contains a fairly long sequence of phonemes that supports a better recognizability. We created two Hidden Markov Models representing the keyword that are applied for the realization of the first recognition stage.

About 280 recordings of the keyword from about 100 speakers (male and female) are taken as initial data to train the parameters of the first HMM. At the beginning we collected spoken keywords with a recording tool where speakers uttered the keyword several times in hands-free mode at a distance of about 0.5 to 3 m from the microphone. Later on, we added further recordings where the keyword was spoken during a first application phase of the recognition system. For data augmentation, we applied a tool [10] to create modified versions of parts of the collected recordings. This tool allows the creation of modified signals that contain background noise at a desired signal-to-noise ratio (SNR) and the effects of a hands-free speech input in reverberant environments. Noise was only added to signals with a good SNR. Noise segments were randomly extracted from several recordings in different babble noise scenarios. The effects of recording in hands-free mode is simulated by folding the speech signal with a room impulse response (RIR). The applied RIR is randomly selected from a collection of responses that have been determined with an experimental setup in real rooms [11]. Thus, the number of about 280 initial recordings could be increased to a total of about 850 utterances containing the spoken keyword. The appropriate tools of HTK [12] have been applied to train the parameters of a simple left-to-right HMM consisting of 27 states with a mixture of 4 Gaussian distributions per state. The number of 27 states is chosen as the adequate number to the second triphone based HMM described below.

The second keyword HMM is built as a concatenation of the 9 triphone HMMs that are defined by the corresponding pronuncation of the keyword. This leads to a simple left-to-right HMM consisting of 27 states with a mixture of 8 Gaussian distributions per state. The second HMM is added as a model that has been trained from more than 100 hundred hours of speech from different databases. The introduction of the second HMM shall counteract the risk that the first HMM is trained on a limited number of utterances from a limited number of speakers so that it does not contain all possible variations of the acoustic features.

2.3 GMM-HMM based recognition with filler models

The sequence of feature vectors as described in section 2.1 and the keyword HMMS as described in the previous section are taken to set up a GMM-HMM recognizer as first stage of the recognition algorithm. A set of 25 monophone HMMs is included as filler models [1] to perform a recognition with a grammar that is visualized by the transitions between two states in Figure 2.

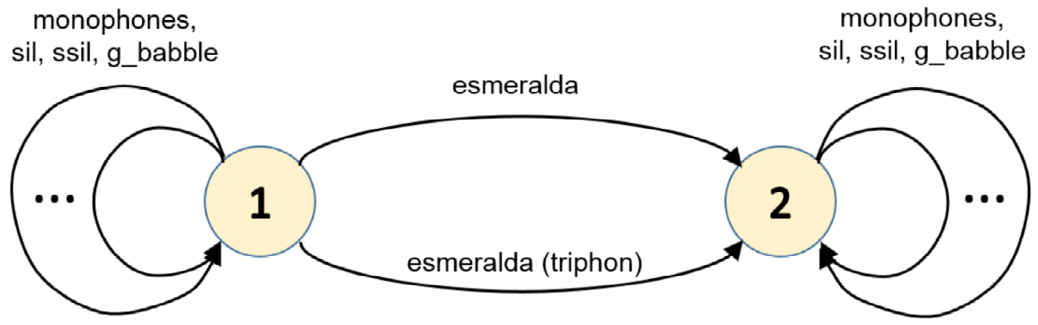

Figure 2 – Grammar of first recognition stage

The desired keyword recognition is included as transition from state 1 to state 2 by one of the two keyword HMMs. The filler models as well as two silence and one babble noise HMM enable the remaining in state 1. One of the silence models and the babble noise model consist of 3 HMM states. The other silence model has a single state only. The intention is the modeling of speech not containing the keyword as sequence of filler and/or pause HMMs. This should lead to the calculation of a higher probability in state 1 in case the speech input does not contain the keyword. In case of a spoken keyword the probability should become higher in state 2 due to a better match to one of the keyword HMMs. A critical point is the fast reaction and feedback in case a keyword is spoken. Therefore, we combine the fulfillment of two conditions as first detection criterium. The first condition is the determination of a higher probability in state 2 in comparison to state 1 so that we can assume that a keyword has been spoken. For the second one we look at this HMM that leads to the calculation of the probability in state 2 according to the Viterbi decision. In case this HMM is one of the filler or pause models for a few successive frames we assume that the keyword has been completely spoken. So, we achieve a fast detection after just a few tens of milliseconds. As second criterium, we look at the number of frames that lead to the calculation of a higher probability in state 2 in case one of the keyword HMMs is the starting point according to the Viterbi decision. If this number is larger than 70 frames, we assume that the keyword has been spoken. In this case, we might even get a reaction already before the end of the keyword.

2.4 Keyword verification by means of a neural network

In case the GMM-HMM recognition stage indicates the detection of a keyword, we try to verify this assumption by a second stage. It is known that the modeling of speech by means of filler models [1] works quite well, but the expected FAR of this approach is too high for the intended application. Therefore, we apply a neural network as main component of the second stage. As input we use the sequence of logarithmic MEL spectra within the speech segment that should contain the keyword according to the recognition result of the first stage. To get a fixed number of input coefficients for the neural network we reduce the number of Mel spectra to 50 by the following procedure. We calculate the spectral differences between all pairs of consecutive logarithmic MEL spectra by means of the City block distance. The average of the two spectra with the smallest difference is determined. Next, the differences between the new average spectrum and the preceding and the succeeding spectrum are recalculated. We repeat the decrement of the number of spectra by averaging two spectra until the number of 50 spectra is reached. The number of 50 is adequate for the fairly long keyword chosen in this work where we found no utterance with a duration less than 500 ms. The average spectrum is presented in Figure 3 that has been calculated over the 850 utterances of the keyword used to create one of the keyword HMMs. We observe a very characteristic spectral pattern where the spectral characteristics of each individual phoneme get clearly visible.

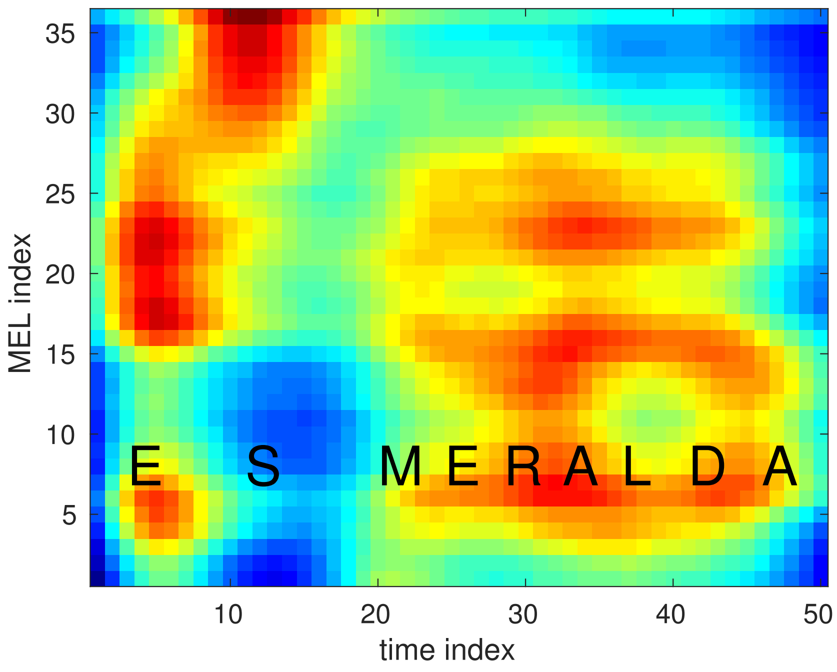

Figure 3 – Average MEL spectrogram of the keyword

The input to the neural network consists of 1800 spectral amplitudes from 50 spectra with 36 MEL coefficients. We apply a mean and variance normalization to each spectral pattern by calculating the mean and the variance over all 1800 spectral parameters of each individual pattern. We apply a fully connected multi-layer perceptron, consisting of 3 layers. The first layer consists of 200 nodes, the second of 50 nodes and the output of 2 nodes for the two cases of a keyword and a non-keyword.

To train the weights of the neural network, we need spectral patterns for spoken keywords as well as for segments where the keyword was erroneously detected by the first stage. About 850 spectral patterns for the spoken keyword can be determined from the utterances that have been used for training the keyword HMM. To get spectrograms for speech segments where the keyword was not spoken we applied the detection algorithm of the first stage to German speech data from different databases [13, 14]. Several thousands of segments were erroneously detected. Thus, we had about 850 examples of the keyword spectrogram and several thousand examples of the non-keyword spectrogram available. We applied the tools of [15] to estimate the weights of the neural network. We trained several networks with an increasing number of non-keyword spectrograms. Results are presented in the next section.

3 Evaluation

As already mentioned, the performance of keyword detection can be measured by FAR and FRR. In our application where we want to apply the keyword detection for the activation of a home automation system, we put a higher priority on lowering the FAR. We have to avoid any command recognition after an erroneous keyword detection because this could lead to the uncontrolled activation of devices at home. At the beginning, we investigated the performance of both recognition stages by means of speech signals available in different databases. After implementing the algorithm on small computer devices as component of a home automation system we were able to evaluate the keyword recognition in real application scenarios.

3.1 Speech signals from databases

At the beginning, we applied the first recognition stage to about 30 hours of speech from a German database [14]. About 7000 segments were erroneously detected that should contain the keyword. This corresponds to the high FAR of about 190 keywords per hour of speech. When performing the keyword detection on the 852 utterances containing a keyword, 844 keywords are recognized. This corresponds to a fairly low FRR. We use the approximately 850 keyword spectrograms and the approximately 7000 non-keyword spectrograms to train a first neural network NN_V1.

Then, we applied the first recognition stage to about 52.5 hours of speech from another German database [13]. About 2400 segments were erroneously detected that should contain the keyword. This corresponds to a FAR of about 46 keywords per hour of speech that is lower in comparison to the FAR for the first database. But it is still too high for a practical application. Applying network NN_V1 on the spectrograms of the 2400 erroneously detected segments we observe that about 80% of the 2400 segments are correctly rejected eventhough we did not use any data from this database for training the network. As next experiment we randomly selected 1200 non-keyword spectrograms as additional non-keyword examples to train a further network NN_V2. Applying the remaining 1200 non-keyword spectrograms to this network, only 2.9% of these 1200 segments are still erroneously recognized as keywords. This would correspond to a FAR of less than 0.7 keywords per hour for the whole database.

3.2 Practical application

We implemented the algorithm for the keyword recognition on some of the client devices we have developed within a project to control devices at home by voice. Two of these devices are shown in Figure 4. They are used to record and play back speech, to control the speech dialogue and to send commands to a home automation system or directly to controllable devices. The client devices contain a PI or a PI-zero as computing unit to perform the keyword detection. The device in the left picture is an integration of all needed components including two microphones in an outlet socket. The device in the right picture contains an array of 4 microphones that can e.g. be integrated in the niche of furniture.

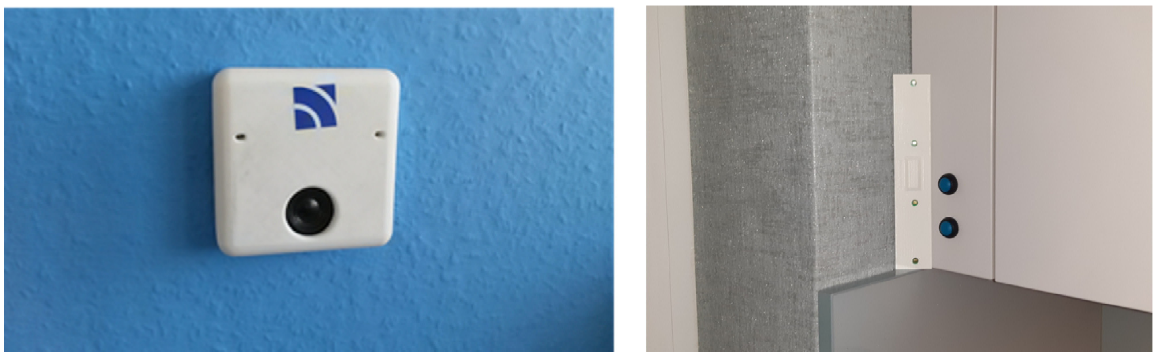
Figure 4 – Two client devices for the voice control of a home automation system

We have run these devices for several weeks in the environment of a laboratory respectively in a living room. We stored the speech segments and the corresponding MEL spectrograms when a keyword was correctly or erroneously detected applying the neural network NN_V1 as second stage of the recognition. We observed false detections in cocktail party situations where a lot people are talking in the background or in situations with an active TV or radio. We collected 348 segments where a keyword was detected but not spoken. Applying network NN_V2 instead of NN_V1 about 67% of the segments could be correctly rejected. We trained a third network NN_V3 after extending the training set used to train NN_V2 by half of the 348 non-keyword spectrograms. Applying network NN_V3 to the other half of the non-keyword spectrograms we observed a correct rejection of about 95% of the 174 non-keyword segments. This shows that the performance of the keyword detection can be considerably improved by adding more data to train the weights of the neural network.

4 Conclusions

The algorithmic details are presented to realize a reliable keyword detection and recognition for the activation of a speech controlled system. The focus was put on the achievement of a low false acceptance rate and on the implementation on devices with low computing power. The algorithm consists of a two stage approach where the first stage is based on a GMM-HMM recognition including a set of filler models. The FAR of the first stage can be considerably reduced by analyzing the spectrogram of the pretended keyword segment by means of a neural network as second stage of the recognition process. It turns out that the FAR can be reduced further by collecting and including more data for training the neural network. In the future, besides further increasing the amount of training data we will investigate other types of network layers like e.g. convolutional layers with the intention of reducing the computational load.

References

[1] ROSE, R. and D. PAUL: *A hidden markov model based keyword recognition system*. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 129–132, 1990.

[2] SAINATH, T. and C. PARADA: *Convolutional neural networks for small-footprint keyword spotting*. *Proceedings of Interspeech*, pp. 1478–1482, 2015.

[3] Chen, G., C. Parada, and G. Heigold: *Small-footprint keyword spotting using deep neural networks*. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 4087–4091, 2014.

[4] Hwang, K., M. Lee, and W. Sung: *Online keyword spotting with a character-level recurrent neural network*. 2015. `https://arxiv.org/pdf/1512.08903.pdf`.

[5] Michaely, A. H., X. Zhang, G. Simko, C. Parada, and P. Aleksic: *Keyword spotting for google assistant using contextual speech recognition*. 2017. `https://storage.googleapis.com/pub-` tools-public-publication-data/pdf/be2559f953dce47e69f4d06692df1184719c4d4b.pdf.

[6] Tang, R., W. Wang, Z. Tu, and J. Lin: *An experimental analysis of the power consumption of convolutional neural networks for keyword spotting*. 2017. `https://arxiv.org/pdf/1711.00333.pdf`.

[7] Hirsch, H.-G. and M. Gref: *Keyword detection for the activation of speech assistants*. *ITG Fachtagung Sprachkommunikation*, 2018.

[8] Breithaupt, C., T. Gerkmann, and R. Martin: *Cepstral smoothing of spectral filter gains for speech enhancement without musical noise*. *IEEE Signal Processing Letters*, 2007.

[9] ETSI: *Speech processing, transmission and quality aspects; distributed speech recognition; advanced front-end feature extraction algorithm; compression algorithm*. *ETSI document ES 202 050 v1.1.3 (2003-11)*, 2003.

[10] Hirsch, H.-G. and H. Finster: *The simulation of realistic acoustic input scenarios for speech recognition systems*. *Proceedings of the 9th European Conference on Speech Communication and Technology*, 2005.

[11] Hirsch, H.-G., A. Kitzig, and K. Linhard: *Simulation of the hands-free speech input to speech recognition systems by measuring room impulse responses*. *ITG Fachtagung Sprachkommunikation*, 2010.

[12] Young, S., G. Evermann, M. Gales, T. Hain, D. Kershaw, X. G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland: *The HTK Book*. version 3.4, 2006.

[13] Burger, S. and F. Schiel: *Rvg 1 - a database for regional variants of contemporary german*. *Proceedings of the 1st LREC conference*, 1998.

[14] Radeck-Arneth, S., B. Milde, A. Lange, E. Gouvea, S. Radomski, M. Muehlhaeuser, and C. Biemann: *Open-source german distant speech recognition: Corpus and acoustic model*. *Proceedings of the 18th International Conference TSD2015*, 2015.

[15] Chollet, F. et al.: *Keras*. `https://keras.io`, 2015.