*INTERSPEECH 2020*
October 25–29, 2020, Shanghai, China

# End-to-End Keyword Search Based on Attention and Energy Scorer for Low Resource Languages

*Zeyu Zhao, Wei-Qiang Zhang*

Beijing National Research Center for Information Science and Technology
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
zzy17@mails.tsinghua.edu.cn, wqzhang@tsinghua.edu.cn

## Abstract

Keyword search (KWS) means searching for the keywords given by the user from continuous speech. Conventional KWS systems based on automatic speech recognition (ASR) decode input speech by ASR before searching for keywords. With deep neural network (DNN) becoming increasingly popular, some end-to-end (E2E) KWS emerged. The main advantage of E2E KWS is to avoid speech recognition. Since E2E KWS systems are at the very beginning, the performance is currently not as good as traditional methods, so there is still loads of work to do. To this end, we propose an E2E KWS model consists of four parts, including speech encoder-decoder, query encoder-decoder, attention mechanism and energy scorer. Different from the baseline system using auto-encoder to extract embeddings, the proposed model extracts embeddings that contain character sequence information by encode-decoder. Attention mechanism and a novel energy scorer are also introduced in the model, where the former can locate the keywords, and the latter can make the final decision. We train the models on low resource condition with only about 10-hour training data in various languages. The experiment results show that the proposed model outperforms the baseline system.

**Index Terms**: Automatic speech recognition, deep neural network, end-to-end, keyword search, low resource language

## 1. Introduction

Keyword search (KWS), or keyword spotting is to detect and locate keywords input by users in continuous speech and to output confidence scores indicating how confident the KWS system is about its decision. The input keywords can be either in text form or spoken terms. In this paper, we merely discuss searching for text-form keywords. Any sequences of characters in the object language are allowed to be input as keywords that we want to search in speech.

Conventional KWS systems [1–5] based on ASR usually decodes speech into word lattices or multiple decoding results in other forms first. After that, a backend KWS system [6, 7] is performed to find keywords of interest. However, with DNN becoming more and more popular, KWS system can also be developed in an end-to-end fashion. First of all, as to an E2E KWS system, there is no need to train an ASR system as the frontend and the corresponding decoding step neither. Besides, to deal with out-of-vocabulary (OOV) queries, conventional ASR-based KWS system usually needs large sub-word lattices, confusion networks or other techniques such as proxy word [8]. However, fully-neural E2E systems make it possible to avoid the above [9]. Last but not least, training an ASR system can sometimes become difficult in low resource situation but, intuitively, training an E2E KWS system is much easier.

A primary method is to extract embeddings or features from input speech and queries using encoders and perform KWS with them [9, 10]. In [9], the authors use an auto-encoder to recover the original input speech features from speech embeddings. If this works well, there is little information lost during compressing input speech to the embeddings.

However, we argue that the most crucial information that speech embeddings should contain is what words occur in input speech because the following KWS operation demands nothing but the information of word sequences for input speech. Thus, we try to convert the speech embeddings into the corresponding character sequences instead. Besides, the authors of [9, 10] transform input speech into fixed-length vectors where the temporal dimension is lost, which we suppose is vital for speech though. On the other hand, we found sequence-to-sequence (Seq2Seq) method works better than CNN-RNN character language model (LM) in [9]. The last phase of [9] is KWS system, which is a multilayer perceptron (MLP) that takes speech and query embeddings as input. We believe this can be replaced by a more sophisticated structure to achieve better performance. Thus, we introduce the attention mechanism and a new component, called *energy scorer*.

The main contribution of our work is that 1) we do not transform the input audio into a fixed-length vector but a variable-length matrix instead, where we keep the time dimension, 2) we improve the performance of recovering accuracy for the input query with a more efficient network structure and 3) we propose attention mechanism and energy scorer to achieve better performance.

## 2. Method

In this section, we first give the overall architecture of our model and then introduce each component in depth, respectively.

### 2.1. Overview

The overall structure of our model is shown in Figure 1. First, we obtain speech and query embeddings by speech and query encoder, respectively. After that, we apply the attention mechanism to output a set of attention weights. Finally, the energy scorer takes all of them as input and outputs the final results.

### 2.2. Speech Encoder-Decoder

The overall process of our speech encoder-decoder is shown in Figure 2.

We first pass input speech features through 1-D CNNs followed by a 1-D max pooling with a stride of 2, which means the

This work was supported by the National Natural Science Foundation of China under Grant U1836219.

The corresponding author is Wei-Qiang Zhang.



http://dx.doi.org/10.21437/Interspeech.2020-2613

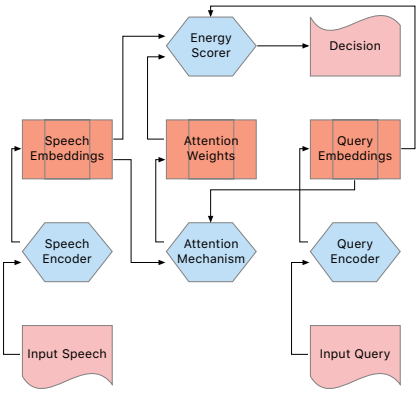


Figure 1: *The overall structure of our E2E KWS system.*

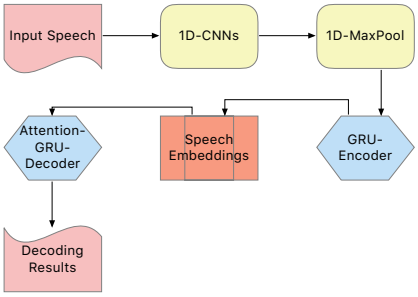


Figure 2: *The process of speech embeddings extraction and using an attention-based decoder (Attention-GRU-Decoder) to predict the corresponding character sequence (Decoding Results) from the speech embeddings. GRU-Encoder and Attention-GRU-Decoder are both multilayer GRU-RNN with the same hyper-parameters.*

number of time-steps will be halved after the above processing. This makes it quicker to train a recurrent neural network (RNN) later. After that, we apply the framework of Attention-based Sequence-to-Sequence [11] to extract our speech embeddings, which is a matrix but not a fixed-length vector. Note that when doing KWS, we only need to compute the speech embeddings given the input speech features but ignoring the part of Attention-GRU-Decoder in Figure 2.

Denote input speech features as $X^* = \{x_1^*, x_2^*, \ldots, x_T^*\}$ and after the process of 1-D CNNs and 1-D maxpooling the we obtain $X = \{x_1, x_2, \ldots, x_{T/2}\}$. The speech embeddings $E_s = \{e_1, e_2, \ldots, e_{T/2}\}$ are computed by GRU-Encoder, which is a multilayer GRU [12] followed by a fully connected layer with ReLU activation, as

$$E_s = \text{Encoder}(X). \tag{1}$$

Given $E_s$, using the GRU-Decoder with Global Attention [11], at each decoding time step $t$, we calculate the attention weights for every $e_s$ in $E_s$, according to

$$\begin{aligned} \alpha_t(s) &= \text{align}(h_t, e_s) \\ &= \frac{\exp\left(\text{score}(h_t, e_s)\right)}{\sum_{s'} \exp\left(\text{score}(h_t, e_{s'})\right)}, \end{aligned} \tag{2}$$

where $\alpha_t(s)$ denotes the attention weight for the $s$-th speech embedding at the $t$-th decoding time step, $h_t$ is the hidden state of the GRU-Decoder at time $t$ and $\text{score}(a, b)$ is the dot product of vector $a$ and $b$.

After that, at each decoding time step $t$, a context vector $c_t$, which is the weighted sum of the speech embeddings, can be computed by

$$c_t = \sum_s \alpha_t(s) e_s. \tag{3}$$

Then, the prediction for the $t$-th time step $o_t$ is obtained by concatenating the context vector $c_t$ and the hidden state $h_t$ and passing them through a fully connected layer (FCL) with softmax activation, where the number of units in FCL is equal to the number of characters in the target language. This can be expressed by

$$o_t = \text{FCL}([h_t^{\mathrm{T}}, c_t^{\mathrm{T}}]^{\mathrm{T}}). \tag{4}$$

We only use speech embeddings to do the following KWS operation, and GRU-Decoder here is just a mechanism to lead GRU-Encoder to extract the speech embeddings that contain the information about the corresponding character sequences.

### 2.3. Query Encoder-Decoder

Similarly, to extract query embeddings from input query, a sequence of characters, we also apply a Seq2Seq method consists of a GRU-Encoder and a GRU-Decoder, but there is no attention mechanism here. The structure of query encoder-decoder is shown in Figure 3.

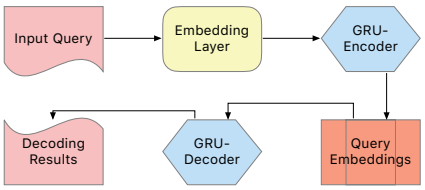


Figure 3: *The process of query embeddings extraction. The hidden state of the last time step of the GRU-Encoder is output as query embeddings. GRU-Encoder and GRU-Decoder are both multilayer GRU-RNN with the same hyper-parameters so that the query embeddings can be taken as initial states by GRU-Decoder.*

Again, note that when performing KWS, we only use GRU-Encoder to extract embeddings for input queries but ignoring GRU-Decoder.

### 2.4. Attention Mechanism

The speech embeddings, which is a matrix, keeps the temporal dimension, and we want to know that which part contains the keywords. That is why we introduce the attention mechanism, which helps to locate the keywords. To be more specific, it outputs a weight for each time step of speech embeddings indicating the probability of the occurrence of the keyword.

The structure of the attention mechanism is shown in Figure 4.

The overall purpose of the attention mechanism can be described as

$$\alpha = \text{Attend}(E_s, E_q^*), \tag{5}$$

where $\alpha = \{a_1, a_2, \ldots, a_{T/2}\}$ and $a_t \in (0, 1)$ denotes the attention weight for time step $t$ and $E_s$, $E_q^*$ are speech and repeated query embeddings respectively. The value of $a_t$ indi-

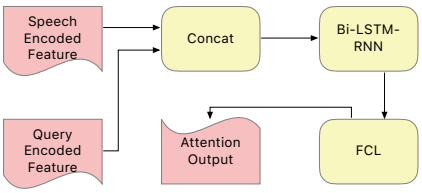


Figure 4: *The structure of the attention mechanism. The speech and query embeddings are first concatenated together (we should repeat the query embeddings first before concatenation, which is not shown in this figure for simplicity.). Bi-LSTM-RNN denotes bi-directional LSTM RNN.*

cates how likely the attention mechanism supposes that a keyword occurs at time step $t$.

Firstly, to concatenate the speech and query embeddings, we should repeat query embeddings to make it the same length of time as speech embeddings. After that, we pass concatenation result through a bi-directional LSTM-RNN [13]. Finally, the last layer is an FCL with a sigmoid activation and one single neural unit.

### 2.5. Energy Scorer

Till now, we have obtained speech embeddings $E_s$, query embeddings $E_q$ and attention weights $\alpha$. The attention weights indicate when the keywords occur, but we still cannot make the final decision because we are not sure about the duration of the keywords.

We are about to combine them to give out a final decision score indicating how likely the keyword occurs in input speech. Thus, we introduce a novel component called *energy scorer*. The structure is shown in Figure 5.

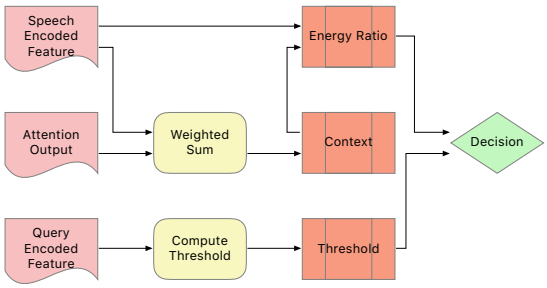


Figure 5: *The structure of the energy scorer. Firstly, we obtain Context, which is the weighted sum of speech embeddings according to the attention outputs. Then we compute the energy ratio of the Context and speech embeddings. Finally, we make the final decision by comparing the energy ratio and the threshold computed by passing the query embeddings through an MLP.*

First the context vector is computed by summing the speech embeddings $E_s = \{e_1, e_2, \ldots, e_{T/2}\}$ according to attention output $\alpha = \{a_1, a_2, \ldots, a_{T/2}\}$, as

$$c = \sum_t a_t e_t, \tag{6}$$

where $a_t \in (0, 1)$ and $c$ denotes the context vector.

The energy of context vector,

$$\begin{aligned} \text{Energy}_c &:= c^{\mathrm{T}} c \\ &= (\sum_t a_t e_t)^{\mathrm{T}} (\sum_t a_t e_t) \\ &= \sum_t a_t^2 e_t^{\mathrm{T}} e_t + \sum_{t_1 \neq t_2} a_{t_1} a_{t_2} e_{t_1}^{\mathrm{T}} e_{t_2} \end{aligned} \tag{7}$$

has an upper bound, namely

$$\text{Energy}_c \leq \sum_t e_t^{\mathrm{T}} e_t + \sum_{t_1 \neq t_2} e_{t_1}^{\mathrm{T}} e_{t_2} =: \text{Energy}_s. \tag{8}$$

Note that (8) holds because after ReLU, $e_t \geq 0, t = 1, 2, \ldots, T/2$.

Finally, we compute the energy ratio $r$ between the context vector and the speech embedding according to

$$r = \text{Energy}_c / \text{Energy}_s. \tag{9}$$

The energy ratio represents the energy proportion of the context vector. The larger the energy ratio is, the more likely that the keyword occurs in the speech. We apply the concept of energy because we want to make it more robust against noise. Usually, the energy of noise is smaller than that of real speech. Thus, even though the attention mechanism outputs a big weight for a time step that is full of noise, $\text{Energy}_c$ may not be large enough, compared with $\text{Energy}_s$, to result in a false alarm decision.

The one last thing is to pass the query embedding through an MLP to get a threshold $r_{\text{th}}$ and compare it with the energy ratio $r$. This means that we generate a specific threshold relating to each query. The reason is that the query embeddings can be almost precisely transformed to the original input query by the GRU-Decoder in Figure 3, implying that nearly all the information of input query has been compressed into this query embedding.

Finally, we make the final decision by

$$r \lessgtr_Y^N r_{\text{th}}. \tag{10}$$

## 3. Experiment

We use Babel-102 Assamese, Babel-103 Bengali and Babel-104 Pashto Dataset to train and evaluate both our proposed model and the baseline model [9]. In order to illustrate the performance on low resource condition, we only use LimitedLP, which contains only about 10-hour speech and the corresponding annotation for each language. To make the training and evaluation procedure easier, we cut all input speech into one-second pieces so that the keyword searching precision is one second.

We implement the proposed and the baseline model on TensorFlow [14] and Keras [15] deep learning framework. Training processed is divided into three parts, speech encoder-decoder, query encoder-decoder, attention mechanism and energy scorer. Namely, we train them separately using the cross-entropy loss function and Adam optimiser [16] with a learning rate of $0.001$.

### 3.1. Speech Encoder-Decoder

We use two 1D-CNNs with 128 and 256 filters, respectively. Besides, the number of units in GRU for both encoder and decoder is 256. The performance of our speech encoder-decoder is shown in Table 1, where the accuracy is character-wise. As we can see, given the speech embeddings extracted by the speech encoder, the speech decoder can predict the character sequences

with an accuracy around 70 percent for each language in our experiment. The results imply that the speech embeddings extracted by speech encoder contain character sequence information.

Table 1: *Character accuracy on training and validation set.*

| | Train | Valid |
|---|---|---|
| Assamese | 0.7758 | 0.7465 |
| Bengali | 0.7780 | 0.7438 |
| Pashto | 0.7266 | 0.6719 |

### 3.2. Query Encoder-Decoder

There are two GRUs for both encoder and decoder and the numbers of units are all 128. The method we apply in this paper is Seq2Seq, and the baseline model used a CNN-RNN character LM [9]. Finally, both of them compress an input query into a 256-element vector. The results are shown in Table 2. We found that Seq2Seq method is more suitable as a query encoder to extract query embeddings from input queries.

Table 2: *Query recovering performance on training and validation set.*

| (Baseline/**Seq2Seq**) | Train | Valid |
|---|---|---|
| Assamese | 0.8451/**0.9939** | 0.8347/**0.9918** |
| Bengali | 0.8159/**0.9942** | 0.8110/**0.9916** |
| Pashto | 0.8737/**0.9990** | 0.8693/**0.9973** |

### 3.3. Attention Mechanism

We visualise the attention outputs, i.e., $\alpha$ in (5) for both positive and negative sample, as shown in Figure 6. We found that for the negative sample, the attention weights are suppressed almost zero while for the positive sample, the prediction is close to the ground truth though there are some noisy points.

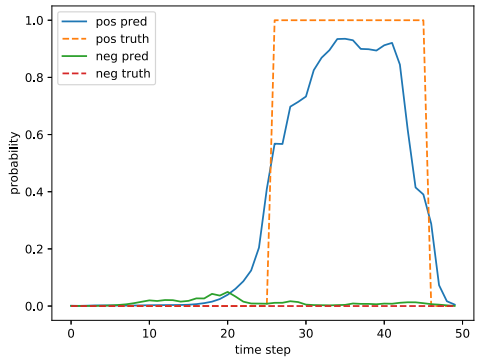


Figure 6: *The visualisation of the attention weights output for negative and positive sample, where "pos pred" and "pos truth" denote the prediction of attention mechanism and the ground truth for the positive sample. Similarly, "neg pred" and "neg truth" are for the negative sample.*

### 3.4. Keyword Search

The loss function applied for both attention mechanism and energy scorer are binary cross-entropy, and we set their loss weights equally.

The training process of the baseline system is similar, namely training each component separately. We train our proposed and the baseline model on the 10-hour training set and evaluate them on a 5-hour evaluation set for each language dataset. For Assamese Bengali and Pashto, the number of IV and OOV are (5285, 2088), (4957, 2368) and (3228, 975), respectively.

For baseline system [9], accuracy was applied as the evaluation metric. Thus, in this paper, we also do so. Besides, we use the metric of AUC (Area Under Curve) [17] as an additional metric to marginalise the threshold and obtain an overall performance. Note that the metric of AUC we mentioned here is the area under the curve, where the x-coordinate denotes false alarm rate, and the y-coordinate represents recall rate.

The results are shown in Table 3. Our proposed model outperforms the baseline model. We suppose there are some reasons for this result. 1) To recover the original input speech features from speech embeddings is not as efficient as transforming speech embeddings into sequences of characters, which is crucial for KWS. 2) As for query embeddings extractor, Seq2Seq method is a better choice as demonstrated in section 3.2. 3) Attention mechanism helps to locate the keywords as illustrated in Figure 6. 4) Energy scorer is much more effective than KWS system in the baseline model.

Table 3: *Evaluation results for IV and OOV*

| (Baseline/**Proposed**) | | Accuracy | AUC |
|---|---|---|---|
| Assamese | IV | 0.6042/**0.6255** | 0.6384/**0.7243** |
| | OOV | 0.6035/**0.6188** | 0.6320/**0.7239** |
| Bengali | IV | 0.5892/**0.6388** | 0.6578/**0.7432** |
| | OOV | 0.5787/**0.6371** | 0.6533/**0.7422** |
| Pashto | IV | 0.6084/**0.6735** | 0.6527/**0.7972** |
| | OOV | 0.5936/**0.6646** | 0.6355/**0.7858** |

## 4. Conclusion

We propose an E2E KWS model consists of four components, speech encoder-decoder, query encoder-decoder, attention mechanism and energy scorer. The speech and query encoders extract embeddings from inputs and pass them through attention mechanism and energy scorer. In order to extract embeddings from input speech, a better method is using a decoder to lead the encoder to extract character sequence information which is vital in the downstream KWS operation. With the same length of query embeddings, Seq2Seq method outperforms CNN-RNN character LM. Energy scorer with attention mechanism is much more powerful than KWS system in the baseline model. As we illustrated, the attention mechanism helps to locate the keywords. Energy scorer makes the final decision by concerning speech embeddings, query embeddings and attention outputs in parallel. The experiment results show that our proposed model outperforms the baseline system.

# 5. References


[1] B. Kingsbury, J. Cui, X. Cui, M. J. F. Gales, K. Knill, J. Mamou, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran, R. Schlüter, A. Sethy, and P. C. Woodland, "A high-performance cantonese keyword search system," in *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8277–8281.

[2] A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny, "End-to-end speech recognition and keyword search on low-resource languages," in *Proceedings of 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5280–5284.

[3] W. Hartmann, V. B. Le, A. Messaoudi, L. Lamel, and J. L. Gauvain, "Comparing decoding strategies for subword-based keyword spotting in low-resourced languages," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2014, pp. 2764–2768. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2014/i14_2764.html

[4] J. Trmal, M. Wiesner, V. Peddinti, X. Zhang, P. Ghahremani, Y. Wang, V. Manohar, H. Xu, D. Povey, and S. Khudanpur, "The Kaldi OpenKWS System: Improving low resource keyword search," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2017-Augus, 2017, pp. 3597–3601. [Online]. Available: http://www.isca-speech.org/archive/Interspeech_2017/abstracts/0601.html

[5] P. Fousek and H. Hermansky, "Towards ASR based on hierarchical posterior-based keyword recognition," in *Proceedings of 2006 IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP 2006, Toulouse, France, May 14-19, 2006*. IEEE, 2006, pp. 433–436. [Online]. Available: https://doi.org/10.1109/ICASSP.2006.1660050

[6] D. Can and M. Saraçlar, "Lattice indexing for spoken term detection," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 8, pp. 2338–2347, 2011.

[7] L. Mangu, B. Kingsbury, H. Soltau, H. Kuo, and M. Picheny, "Efficient spoken term detection using confusion networks," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*. IEEE, 2014, pp. 7844–7848. [Online]. Available: https://doi.org/10.1109/ICASSP.2014.6855127

[8] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proceedings of 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2013*, 2013, pp. 416–421.

[9] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-End ASR-Free Keyword Search from Speech," *IEEE Journal on Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1351–1359, dec 2017.

[10] N. Sacchi, A. Nanchen, M. Jaggi, and M. Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2019-Septe, 2019, pp. 3362–3366.

[11] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL), aug 2015, pp. 1412–1421. [Online]. Available: http://arxiv.org/abs/1508.04025

[12] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL), jun 2014, pp. 1724–1734.

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, nov 1997.

[14] M. Abadi, A. Agarwal, P. Barham *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

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

[16] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of 3rd International Conference on Learning Representations, ICLR 2015*. International Conference on Learning Representations, ICLR, dec 2015.

[17] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.