# Utterance-level Code-Switching Identification using Transformer Network

*Krishna D N, Ankita Patil*

HashCut Inc.

krishna@sizzle.gg, ankita@sizzle.gg

## Abstract

A continuous change between two or more languages in a single utterance is known as code-switching. In this work, we develop a novel model to predict whether given audio is mono-lingual or contains code-switching. We propose to use a convolutional encoder in combination with transformer architecture for utterance-level code-switching detection. The convolution encoder processes the spectral features from audio with the help of 1D convolutional layers. The convolutional layer output features are processed by the Transformer network, consisting of a sequence of multi-head self-attention layers. The multi-head self-attention layers use attention mechanisms and help in selecting relevant information for better code-switching detection. Finally, our model aggregates frame-level features from the transformer network to create an utterance-level feature vector to predict the class label. We train and evaluate our model using the dataset provided by the Microsoft code-switching challenge. The training data contains three different languages, and each language contains both mono-lingual and code-switched utterances. Our model achieves significantly better accuracy compared to the baseline model. Our experiments show that, compared to the baseline, we obtain approximately 8%, 7%, and 3% improvement in accuracy for Telugu, Gujarati, and Tamil language, respectively.

**Index Terms**: code-switch detection,transformer, 1D-CNNs.

## 1. Introduction

Code-switching detection is about identifying whether an utterance contains multiple languages. Code-switching speech is defined as a speech that contains multiple languages within a single conversation [17]. Code-switching is known to occur in mostly low resource languages. In India, almost all of the spoken languages are multilingual in nature. Code-switching can sometimes cause problems to speech recognition systems, which are built using mono-lingual data. Many researchers in the speech community are trying to develop systems that can handle code-switching speech. Recently, robust acoustic-modeling techniques[8] are developed to handle code-switching. [5] proposes to map English phonemes to German phonemes to create a robust acoustic model for German speech recognition. [6] propose to use language-independent speech recognition and have shown to improve the system performance for Chinese speech recognition system. Recently, [7] built an end-to-end sequence-based neural network model for improving Chinese code-switching speech recognition accuracy by predicting language labels for every frame. Recently techniques [19,20] have shown to use language modeling in order to build robust speech recognition systems against code-switching. Language identification (LID) also seems to be an important technique to deal with code-switching [9,10,11,18]. Some of the interesting works[1,2,3,4] have shown different methods to detect the code-switching in single conversation/utterance. Multilingual training of DNN-based speech recognition systems has shown some improvements in the low resource languages[13,14,15,16].

Recent developments in the area of deep learning [27] have shown tremendous improvement in many areas of speech, including speech recognition [21], language identification [22], speaker identification [23], etc. Sequence to sequence [24] models have shown to be the best models for sequence prediction or sequence mapping problems like machine translation and speech recognition. Attention models [25] have become the dominating models in both NLP and speech problems due to their ability to focus on specific part of the signal to extract relevant information. Attention models have been the state of the art models for most of the sequence mapping problems like machine translation, speech recognition [21], text-to-speech, etc. Today state of the art speech recognition uses attention based sequence to sequence model due to its efficiency and capacity to align and predict. Transformers [26] is a special type of attention model that uses only attention technique without any other layers of convolution or RNNs, and they have been the best models for machine translation. Recently, attention models have also been extensively used in problems like emotion recognition [28], language identification and they are shown to be the best models for these problems.

Motivated by the work [26], this paper proposes to use transformer architecture in combination with a 1D convolutional neural network for code-switching identification. Our proposed model sequence of 257-dimensional spectral features as input and applies a sequence of 1D convolutional layers followed by a sequence of multi-head self-attention layers. We use the statistics pooling layer as an utterance-level aggregator to pool the frame-level features into an utterance-level feature vector. The utterance-level feature is used to predict if the input is mono-lingual or code-switched utterance. We conduct all our experiments on the Microsoft code-switching challenge dataset for three Indic languages Telugu, Tamil and Gujarati. Our experimental study shows that the transformer architecture can significantly improve the baseline for all three languages.

The organization of the paper is as follows. In section 2, we explain our proposed approach in detail. In section 3, we give a detailed analysis of the dataset, and in section 4, we explain our experimental setup in detail. Finally, in section 5, we describe our results.

## 2. Proposed approach

In this section, we explain our proposed approach in detail. The detailed model architecture is shown in Figure 1. Our model consists of 3 main stages, 1) An Convolutional Encoder layer, which consists of series of 1D convolutional layers, 2) Transformer block, consisting of multiple self-attention layers to select important and relevant features for code-switching detection using attention weighting and 3)Utterance-level aggregation layer, which contains statistics pooling layer to obtain ut-

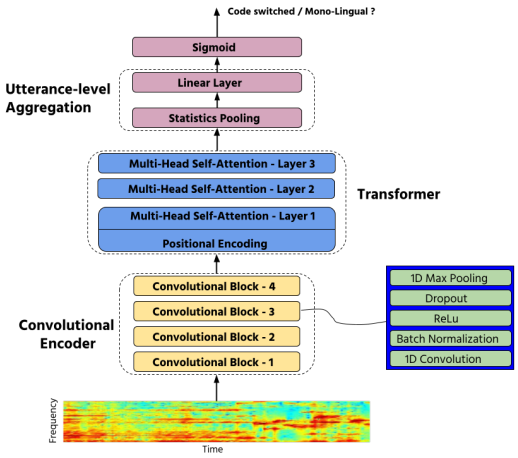


Figure 1: *Proposed model architecture.*

terance level feature vector for classification. The model takes a spectrogram feature matrix as input, and it is fed to the Convolutional Encoder as shown in Figure 1. The features from the convolutional encoder layer will go to the transformer block, which consists of a series of multi-head self-attention to extract relevant features from different parts of the input. The statistics pooling layer generates an utterance level feature vector by concatenating the mean and standard deviation vectors. The output of the statistics pooling layer gives us a single feature vector called the utterance level feature vector. This utterance level feature vector is fed into a projection layer followed by a Sigmoid layer to predict whether the input audio is monolingual or code-switched. We explain the details of each of these blocks in the following section.

### 2.1. Convolutional Encoder

Convolutional neural networks have been used a lot by computer vision community due to their nature of learning higher-level representation through the hierarchical structure. In speech, the audio waveform also contains both lower and higher-level abstractions. The higher-level abstractions like phonemes, syllables, or words are important in code-switching, whereas lower-level representations are may not of great importance. We use a convolutional neural network to learn these higher-level representations from lower-level spectral features. The Convolution encoder stage contains a series of 4 Convolution blocks, as shown in Figure 1. Each Convolutional block consits of 1D convolution operation, 1D Batch Normalization, Relu, Dropout and 1D max-pooling as depicted in Figure 1. Each of the convolution blocks operates at the same kernel size, but the number of filters varies between them. The convolution encoder takes a spectrogram feature matrix as input. The spectrogram contains a series of T spectral features with a feature dimension of 257. The input processed by the first convolution block *Convolution Block -1* consists of 64 1D convolution filters of kernel size 1x3, and the max-pooling operation is applied with a kernel size of 1x3 and stride 2. Similarly, the *Convolution Block -2*, *Convolution Block -3* and *Convolution Block -4* have 128,256,and 256 filters respectively. Each of the convolution blocks is operated with the same kernel size of 1x3. Also, each of the blocks is operated with a max-pooling kernel size of 1x3 and stride of 2. The convolution filter sizes and max-pooling filter sizes are chosen in such a way that the final output from the convolution encoder gives us a feature vector for every 200 ms. Let $\boldsymbol{X_n} = [\boldsymbol{x_1}, \boldsymbol{x_2}..\boldsymbol{x_n}, ...\boldsymbol{x_T}]$ be a spectrogram of an utterance consisting of T feature vectors. The input vector $\boldsymbol{x_n}$ is a spectral feature vector extracted from the raw audio.

$$\boldsymbol{H^C} = \text{Convolution-Encoder}(\boldsymbol{X_n}) \tag{1}$$

Where, Convolution-Encoder is a mapping function which consisting of series 1D Convolution blocks *Convolution Block -1*,*Convolution Block -2*, *Convolution Block -3*, and *Convolution Block -4*. After this operation, we obtain a feature sequence $\boldsymbol{H^C} = [\boldsymbol{h_1}, \boldsymbol{h_2}.....\boldsymbol{h^{T'}}]$ of length $\boldsymbol{T'}$ and $\boldsymbol{T'} << \boldsymbol{T}$. The Convolution-Encoder operation can be thought of as a feature learning neural network, and it also helps frame-rate reduction. The reason is as follows, typically the spectral features are extracted at a 10ms frame rate from the raw audio, but the code-switching can happen at a word or sub-word level, typically occurring for around 200ms. We design the network in such a way that we obtain a single feature vector approximately for every 200 ms so that the transformer model can look for variations in the features and learn to classify whether an utterance is mono-lingual or code-switched.

### 2.2. Transformer

Attention-based models have shown significant improvement in many speech problems these days. Specifically, the transformer [26] architecture is shown to the best neural network model for many sequence mapping problems like machine translation and speech recognition due to their ability to select important and relevant information during prediction. In this work, we propose to use a transformer for code-switching detection task. We expect the transformer model to learn to attend and select relevant features for better classification once we extract higher-level representations using convolutional layers. We assume that after training, the transformer will be able to focus and select the feature where code-switching occurs through an attention mechanism. Hence, we will be able to predict if the given utterance is monolingual or code-switched.

In this section, we describe the transformer block in detail. The transformer block contains three multi-head self-attention layers and an initial positional encoding layer [26], as shown in Figure 1. The positional encoding layer uses the sinusoidal positional embedding technique, as proposed in [26]. The multi-head self-attention layers in the transformer use scaled dot product attention mechanism to select the relevant features from the input feature sequence $\boldsymbol{H^C} = [\boldsymbol{h_1}, \boldsymbol{h_2}.....\boldsymbol{h^{T'}}]$ at multiple levels. The transformer helps attend to different parts of the input to detect if there is code-switching occurring in between monolingual words. The multi-head self-attention block is described, as shown in Figure 2. It consists of 3 different linear blocks, one for query $\boldsymbol{Q}$ , one for key $\boldsymbol{K}$, and another for value $\boldsymbol{V}$. Each linear block consists of M independent linear layers, as shown in Figure 2, where M is the number of heads. The first multi-head attention block *Multi-Head Self-Attention -Layer 1* takes features $\boldsymbol{H^C} = [\boldsymbol{h_1}, \boldsymbol{h_2}.....\boldsymbol{h^{T'}}]$ from Convolution block after positional encoding operation and applies linear transformation to create $\boldsymbol{Q_i}$, $\boldsymbol{K_i}$ and $\boldsymbol{V_i}$ using $\boldsymbol{i}^{\text{th}}$ linear layers where, $\boldsymbol{i} = [\boldsymbol{1}, \boldsymbol{2}.....\boldsymbol{M}]$ and M is the total number of attention heads. The $\boldsymbol{Q_i}$, $\boldsymbol{K_i}$ and $\boldsymbol{V_i}$ are fed into scaled dot product attention layer. The scaled dot product attention $\boldsymbol{A_i}$ for $\boldsymbol{i}^{\text{th}}$ head is defined as follows.

$$\boldsymbol{A_i} = \text{Softmax}(\frac{\boldsymbol{Q_i K_i}}{\boldsymbol{d_q}})\boldsymbol{V_i} \tag{2}$$

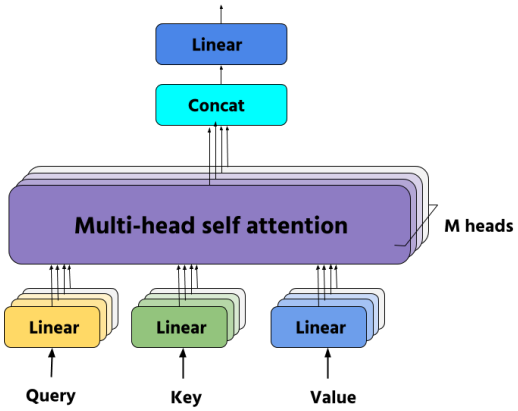


Figure 2: *multi-head self-attention*

Where $d_q$ is the dimension of the query vector. We combine the attention output from all the heads using simple concatenation and feed into the feed-forward layer.

$$A = \text{Concat}(A_1, A_2, A_3 ... A_i ..... A_M)W_0 \quad (3)$$

Where, $A_i$ is a $d_q$x$T$ dimensional matrix. Since the Concat operation is applied to the feature dimension of all the matrices, the final output attention matrix $A$ from multi-head attention block will have $Md_q$x$T$ matrix dimensions. The output of the first layer $A$ goes to the second layer *Multi-Head Self-Attention -Layer 2*, and this time the query, Key and Value vectors are generated from $A$. Similarly, the output from *Multi-Head Self-Attention -Layer 2* are fed *Multi-Head Self-Attention -Layer 3* to select more important features at a higher level. It can be seen that the output of the last layer of the transformer block will have the same temporal dimension as $H^C$ because multi-head self-attention layers do not change the dimension. Overall, the transformer layer helps in finding regions that are more for detecting code-switch. The scaled dot product attention achieves this by giving more weighting to the features which are more relevant and less weighting to less relevant features. This process selects features from different parts of the input and helps in obtaining better classification performance due to the presence of multiple heads in the attention layer.

### 2.3. Statistics pooling

The idea of the statistics pooling layer is similar to max pooling. In the case of statistics pooling, we compute the mean and standard deviation from feature vectors generated by the transformer model. The mean and standard deviation features are concatenated in order to create the utterance level feature vector, as described in the equation below. Let $A = [a_1, a_2 ..... a_{T'}]$ is the output from the transformer block.

$$P = \text{Concat}(\text{mean}(A), \text{std}(A)) \quad (4)$$

Where, $a_i$ is a feature vector of dimension $M * d_q$ and $P$ is final pooled feature vector using statistics pooling layer. Since the dimension of the utterance level feature vector is $P$ become bigger when M is large, we add a projection layer on top to the statistics pooling layer (Figure 1) in order to reduce the dimension of $P$. The project layer output will be passed to the Sigmoid layer in order to predict if the input utterance contains any code-switching.

Table 1: *Train and evaluation splits for different languages*

| Datset | Train | | Evaluation | |
|---|---|---|---|---|
| | Duration(Hrs) | Utterances | Duration(Hrs) | Utterances |
| Gujarati | 31.59 | 16780 | 3.59 | 2091 |
| Telugu | 31.59 | 16991 | 4.0 | 2135 |
| Tamil | 30.24 | 10933 | 7.312 | 2642 |

## 3. Dataset

In this section, we briefly discuss the dataset. We obtain the dataset from Code-switching spoken language identification challenge[1]. The task of the challenge is to detect if an audio file is monolingual or code-switched. The shared task contains two subtasks, 1) Utterance-level identification of monolingual vs. code-switched utterances, and 2)Frame-level identification of language in a code-switched utterance. In this work, we chose to work on the first subtask. The dataset contains 3 Indic languages, Gujarati, Telugu, and Tamil. Each language contains audio files that are monolingual and code-switched along with their labels. The statistics of the dataset is given in Table 1. In India, most of the languages are code-switched with English languages. In this dataset also, each utterance can be completely monolingual, or it can be mixed with English. Each language pair contains training data and evaluation dataset. The dataset contains labels of every 200ms in the audio, but in our case, we convert all the frame-level labels utterance level labels. We train a separate model for each of the languages, and we report our results on the evaluation dataset. We also test our model on the blind dataset provided during the challenge.

## 4. Experiments

We conduct all our experiments on Microsoft's Code-switching spoken language identification challenge dataset. The dataset is described in the previous section in detail. We develop our model for an utterance-level code-switching identification task. Our model consists of a Convolutional encoder operating with 1D convolution kernels. The convolution encoder has 4 convolution block with [64,128,256,256] 1D convolution kernels of kernel size 1x3. Each layer also has a max-pooling layer operating with a kernel size of 1x3 with stride 2. The output of the Convolutional encoder goes to the transformer block, which consists of 3 multi-head self-attention blocks. Each self-attention block has eight attention heads. Finally, the statistics pooling layer computes the mean and standard deviation to pool frame-level features into utterance level feature. We use the sigmoid layer to predict the binary class label. In our case, we use 0-(class-1) corresponds to code-switched, and 1(class-1) corresponds to monolingual. The input to our model is a 2D magnitude spectrogram (we refer magnitude spectrogram as a spectrogram). We take the maximum length of the audio to be 25sec. We crop the audio if the duration of the audio is more than 25sec, and we pad zeros if the duration is less than 25sec. We compute a 257-dimensional spectral feature for every 25ms window with a frame-shift of 10ms. So, the spectrogram contains 2500

[1]https://www.microsoft.com/en-us/research/event/workshop-on-speech-technologies-for-code-switching-2020/

Table 2: *Acc (%) and EER(%) on Evaluation dataset: bold represents the best method*

| Language | Baseline | | Proposed | |
|---|---|---|---|---|
| | Acc (%) | EER(%) | Acc (%) | EER(%) |
| Te-En | 71.2 | 14.4 | **79.34** | **12.40** |
| Ta-En | 74.0 | 13.0 | **76.70** | **12.55** |
| Gu-En | 76.8 | 11.6 | **83.54** | **9.9** |

frames, and each frame will be of 257 dimensions. We can think of the spectrogram as an image with a single channel. We also normalize the spectrogram by computing the mean and standard deviation. We use binary cross-entropy loss function to train our model. We use Adam [30] optimizer to train all our models with a learning rate of 0.0001 for up to 80 epochs. We use a batch size of 32 during training. We train all our models using Pytorch [29] toolkit. We compare our results with the baseline model, and we observe that our approach gives 8%, 7%, and 3% improvement in accuracy for Telugu, Gujarati, and Tamil language respectively over the baseline results. We use 3 RTX 2080Ti GPU cards for our experiments. We will open-source our codes and models soon.

## 5. Results

In this section, we describe the evaluation of different models and their performances. Our baseline model is a five layer LSTM model, each layer having a hidden dimension of 1024. The baseline model uses CTC loss function during training. The workshop organizers already provide the baseline model results for the evaluation dataset. We train our proposed model for three different languages Gujarati, Telugu, and Tamil. We compare both accuracy and equal error rate performance of our model with the baseline approach. The results are shown in Table 2. We represent the languages as Gu-En, Te-En, and Ta-En. Where Gu-En represents Gujarati code mixed with English, Te-En represents Telugu code mixed with English, and Ta-En represents Tamil code mixed with English. It can be seen that our approach obtains 8.74% absolute improvement in accuracy and 1.69% improvement in EER for Gu-En. Similarly, we obtain an 8.14% improvement in accuracy and 2.0% improvement in EER for Te-En. For Ta-En, we obtain 2.7% absolute improvement in accuracy and 0.5% improvement in EER.

We also test our models on the blind test data provided for the challenge, and the results are shown in Table 3. We can see that the model performance not as good as on the evaluation dataset, and this may be due to various reasons like speaker variability, perturbation like speed and volume, etc. We do not know the exact reason why there is a significant difference between the evaluation test set and the blind test set. We want to explore the reason and make improvements in the future.

## 6. Conclusions

In this work, we propose a new approach for utterance-level code-switching detection using transformer architecture. We use a convolutional encoder to extract higher-level abstractions at phoneme or syllable levels. We use a sequence of multi-head self-attention layers to find to select and learn relevant information to detect if the utterance is monolingual or contains code-switching. We also utterance-level aggregation module, which uses a statistics pooling approach to pool frame-level features into utterance-level features. Our experimental results show that a transformer-based model can outperform the LSTM baseline due to their ability to attend and select different parts of the input to detect code-switching. Our method consistently obtains huge improvement on Microsoft's Code-switching spoken language identification challenge dataset for Telugu, Tamil and Gujarati languages. We also see that the model suffers accuracy on the blind test data due to the speaker and other local variations. We would like to explore the reasons and make improvements in the future.

Table 3: *Accuracy and EER on blind test data*

| Language | Proposed | |
|---|---|---|
| | Acc (%) | EER(%) |
| Te-En | 69.23 | 15.38 |
| Ta-En | 69.08 | 15.45 |
| Gu-En | 48.46 | 25.76 |

## 7. Acknowledgements

We want to thank Microsoft India for providing the dataset for this research work. We would also like to thank HashCut Inc. for supporting this work. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of HashCut Inc.

## 8. References

[1] E. Yılmaz, H. van den Heuvel and D. van Leeuwen, "Code-switching detection using multilingual DNNS," 2016 *IEEE Spoken Language Technology Workshop (SLT)*, San Diego, CA, 2016, pp. 610-616, doi: 10.1109/SLT.2016.7846326.

[2] C. Wu, H. Shen and C. Hsu, "Code-Switching Event Detection by Using a Latent Language Space Model and the Delta-Bayesian Information Criterion," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 11, pp. 1892-1903, Nov. 2015, doi: 10.1109/TASLP.2015.2456417.

[3] E. Yılmaz, H. Van den Heuvel, and D. A. Van Leeuwen, "Exploiting untranscribed broadcast data for improved code-switching detection," in *Proc. INTERSPEECH*, Aug. 2017, pp. 42–46.

[4] Emre Yılmaz, Henk van den Heuvel, and David van Leeuwen, "Code-switching detection with dataaugmented acoustic and language models," in the *Sixth International Workshop on Spoken Language Technology for Under-resourced Languages (SLTU)*. Procedia Computer Science, 2018, pp. 127–131.

[5] G. Stemmer, E. Noth, and H. Niemann, "Acoustic modeling of foreign words in a German speech recognition system," in *Proc.EUROSPEECH*, 2001, pp. 2745–2748

[6] D.-C. Lyu, R.-Y. Lyu, Y.-C. Chiang, and C.-N. Hsu, "Speech recognition on code-switching among the Chinese dialects", in *Proc. ICASSP*, vol. 1, May 2006, pp. 1105–1108

[7] N. Luo, D. Jiang, S. Zhao, C. Gong, W. Zou, and X. Li, "Towards end-to-end code-switching speech recognition," *arXiv preprint* arXiv:1810.13091, 2018.

[8] E. Yılmaz, H. Van den Heuvel, and D. A. Van Leeuwen, "Investigating bilingual deep neural networks for automatic speech recognition of code-switching Frisian speech," in *Proc. Workshop on Spoken Language Technology for Under-resourced Languages (SLTU)*, May 2016, pp. 159–166.

[9] J. Weiner, N. T. Vu, D. Telaar, F. Metze, T. Schultz, D.-C. Lyu, E.-S. Chng, and H. Li, "Integration of language identification into a recognition system for spoken conversations containing codeswitches," in *Proc. SLTU*, May 2012.

[10] K. R. Mabokela, M. J. Manamela, and M. Manaileng, "Modeling code-switching speech on under-resourced languages for language identification," in *Proc. SLTU*, 2014, pp. 225–230.

[11] D.-C. Lyu, E.-S. Chng, and H. Li, "Language diarization for codeswitch conversational speech," in *Proc. ICASSP*, May 2013, pp.7314–7318.

[12] T. I. Modipa, M. H. Davel, and F. De Wet, "Implications of Sepedi/English code switching for ASR systems," in *Pattern Recognition Association of South Africa*, 2015, pp. 112–117.

[13] S. Thomas, S. Ganapathy, and H. Hermansky, "Multilingual MLP features for low-resource LVCSR systems," in *Proc. ICASSP*, March 2012, pp. 4269–4272.

[14] Z. Tuske, J. Pinto, D. Willett, and R. Schluter, "Investigation on cross- and multilingual MLP features under matched and mismatched acoustical conditions," in *Proc. ICASSP*, May 2013, pp.7349–7353.

[15] N. T. Vu, F. Metze, and T. Schultz, "Multilingual bottle-neck features and its application for under-resourced languages," in *Proc.SLTU*, May 2012.

[16] K. Vesely, M. Karafiat, F. Grezl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *Proc. SLT*, Dec 2012, pp. 336–341.

[17] Peter Auer, "Code-switching in conversation: Language, interaction and identity", Routledge, 2013.

[18] K Bhuvanagiri and Sunil Kopparapu, "An approach to mixed language automatic speech recognition," *Oriental COCOSDA*, Kathmandu, Nepal, 2010.

[19] H. Adel, N. Vu, F. Kraus, T. Schlippe, H. Li, and T. Schultz, "Recurrent neural network language modeling for code switching conversational speech," in *Proc. ICASSP*, 2013, pp. 8411–8415.

[20] Heike Adel, Katrin Kirchhoff, Dominic Telaar, Ngoc Thang Vu, Tim Schlippe, and Tanja Schultz, "Features for factored language models for code-switching speech," in *Spoken Language Technologies for Under-Resourced Languages*, 2014.

[21] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition" in *ICASSP*, 2016

[22] E. Variani, X. Lei, E. McDermott, I. Lopez-Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, May 2014.

[23] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur,"X-vectors: Robust dnn embeddings for speaker recognition," in 2018 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018

[24] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[25] D. Bahdanau, K. Cho, and Y. Bengio. "Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv:1409.0473*, 2014

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp.5998–6008.

[27] Schmidhuber, J. (2015). "Deep learning in neural networks: An overview". *Neural Networks*, 61 ,85–117.

[28] M. Sarma, P. Ghahremani, D. Povey, N. K. Goel, K. K. Sarma, and N. Dehak, "Emotion identication from raw speech signals using dnns", *Proc. Interspeech* 2018, pp. 3097–3101, 2018.

[29] Paszke, Adam and Gross, Sam and Chintala, Soumith and Chanan, Gregory and Yang, Edward and DeVito, Zachary and Lin, Zeming and Desmaison, Alban and Antiga, Luca and Lerer, Adam "Automatic differentiation in PyTorch" in *NIPS*, 2017

[30] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda and K. Takeda, "Weakly-Supervised Sound Event Detection with Self-Attention," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain*, 2020, pp. 66-70, doi: 10.1109/ICASSP40776.2020.9053609.

[31] B. Kim and S. Ghaffarzadegan, "Self-supervised Attention Model for Weakly Labeled Audio Event Classification," *27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain*, 2019, pp. 1-5, doi: 10.23919/EUSIPCO.2019.8902567