Interspeech 2022
18-22 September 2022, Incheon, Korea

Accent Conversion using Pre-trained Model and Synthesized Data from Voice Conversion

Tuan Nam Nguyen[1], *Ngoc Quan Pham*[1], *Alexander Waibel*[1,2]

[1]Karlsruhe Institute of Technology
[2]Carnegie Mellon University
tuan.nguyen@kit.edu

Abstract

Accent conversion (AC) aims to generate synthetic audios by changing the pronunciation pattern and prosody of source speakers (in source audios) while preserving voice quality and linguistic content. There has not been a parallel corpus that contains pairs of audios having the same contents yet coming from the same speakers in different accents, the authors hence work on a solution to synthesize one as training input. The training pipeline is conducted via two steps. First, a voice conversion (VC) model is constructed to synthesize a training data set, containing pairs of audios in the same voice but two different accents. Second, an AC model is trained with the synthesized data to convert a source accented speech to a target accented speech. Given the recognized success of self-supervised learning speech representation (wav2vec 2.0) on certain speech problems such as VC, speech recognition, speech translation, and speech-to-speech translation, we adopt this architecture with some customization to train the AC model in the second step. With just 9-hour synthesized training data, the encoder initialized by the weight of the pre-trained wav2vec 2.0 model outperforms the LSTM-based encoder.

Index Terms: Accent Conversion, Voice Conversion, seq2seq, wav2vec 2.0

1. Introduction

The main challenge in AC is that ground-truth data for the desired output is not available. Previous solutions [1] [2] tackle this issue by introducing a reference utterance from target accented speakers at inference. However, it is not always possible to get references from those target objects, which limits the applicability of this method. As an alternative solution, we propose an end-to-end AC approach capable of converting accent from source accented utterances without using any target reference during the inference phase. The proposed approach includes two major steps: (1) Generating synthetic data by a VC model; and (2) Training a seq2seq AC model with those synthetic data. In the first step, we use a VC model which can preserve pronunciation pattern and prosody of the source audios and only convert the identity (i.e., timbre and pitch) of a source speaker into a target one. Since prosody and pronunciation pattern are the most important constituents of an accent, if we can preserve both of them in an audio, the accent will remain the same. Accordingly, we can synthesize a pair of utterances having the same voice and same context, yet different accents. The process of generating synthetic data is described in section 3.1. In the second step, we train a seq2seq AC model with audio data created in the first phase.

Our approach is inspired by a number of prevailing speech methodologies. First, speech recognition and accent conversion models share the input type in common, which is audio data. That is why we can investigate the architecture of a speech recognition encoder to apply on an AC model. Also, using a pre-trained audio encoder is proven to improve the performance of speech recognition models without requiring much labeled data [3]. Second, an AC model and a speech synthesis system share the same output modality so we can employ the architecture of a speech synthesis decoder for an AC model. In the end, our experiment seq2seq AC model consists of a pre-trained audio encoder and a speech synthesis decoder. In brief, we use VC to resolve the ground-truth data challenge in AC problem and then prove the advantages of using a pre-trained audio encoder in training the AC model without relying on a great deal of labelled audio data.

2. Related work

The goal of any voice conversion system is to convert the voice of a source speaker into the voice of another target speaker. Traditional techniques based on Gaussian Mixture Models performed well, but those based on Artificial Neural Networks even outperformed them. Several architectures have been used to solve VC problems such as GAN, VAE, and seq2seq. There exist also many kinds of VC systems: one-to-one, one-to-many, many-to-many, any-to-any, and so on. The any-to-any VC system, which can convert any source speaker to any target speaker even if a speaker has not been seen in training data, is the most challenging one and has been addressed in several experiments like VQMIVC [4], AutoVC [5], Adain [6], FragmentVC [7]. We adopt the two most recent architectures VQMIVC and FragmentVC, to train a VC system and generate synthetic data, then find out which is the better data synthesizing method.

Accent conversion is often related to the similar problem: voice conversion (VC). While VC mainly focuses on changing the speaker identity in audios, AC only changes or corrects the pronunciation pattern while trying to retain the speaker identity. AC is a more challenging problem because there are generally no ground-truth data with the same voice and different accents. Phonetic posteriorgrams (PPGs) [2, 1], which have been successfully used for VC tasks, can also be used in AC despite various limitations of the systems. For example, in order to convert a non-native accent to a native accent, these systems require reference native-accented utterances at the conversion stage, which ultimately restrict their practical applications in real life. AC without reference utterance is known as a reference-free process [8]. Our literature research could only highlight two prior works on reference-free AC, namely [9], [8]. The second research has addressed some limitations in the first one by eliminating some complicated components such as TTS, ASR and is more related to our proposed methodology. They used speech synthesizer with speech embedding as an input to do synthetic data and an LSTM-based seq2seq AC model was trained from

10.21437/Interspeech.2022-10729

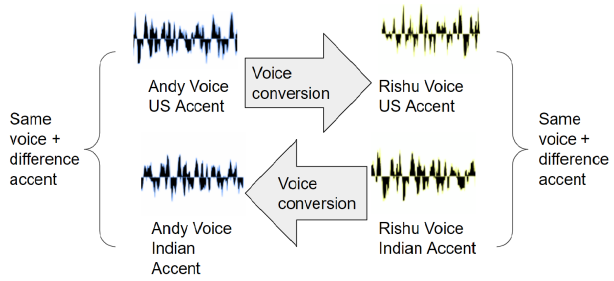

Figure 1: *Synthetic data process*

scratch. While their speech synthesizer requires to be trained specifically for each speaker in the training set, our approach uses a single VC model for synthesizing training data of all speakers and is a more convenient way. As a result, we can refer to their AC model as the baseline approach against which our model is compared.

Most previous research focuses mostly on converting non-native into native speech. The reverse conversion could be more challenging since non-native speech data is not as available as native speech. In this research, beside correcting the pronunciation of non-native speakers, we also concentrate on experimenting native-to-non-native AC. This kind of conversion could help to generate more non-native accented speech from native speakers, enabling the enrichment of the non-native data resource.

Recently, pre-trained models have been applied to a success in speech translation [10], speech-to-speech translation [9], speech recognition [3], and voice conversion [7]. These models have the potential to provide another way to overcome the insufficiency of labeled data. Inspired by the success of wav2vec 2.0, our research proposes an AC model with wav2vec 2.0 pre-trained that can run effectively in low-resource condition.

3. Method

3.1. Generating synthetic data

3.1.1. General idea of generating synthetic data

The purpose of generating synthetic data is to create pairs of audios that can be used in training the seq2seq AC model. For example, there are two audios of the same content recorded by two different speakers: Andy with an American accent and Rishu with an Indian accent. When we apply voice conversion on Andy's original audio, the expected result is an audio having Rishu's voice yet remaining in American accent. This is consistent to a VC system which only changes a speaker's identity while preserves an original accent. At the end of the process, we have a pair of audios with Rishu's voice in two different accents - (original) Indian and (converted) American. If we do the voice conversion other way round on Rishu's audio, we expect to have an audio pair in Andy's voice but two different accents - (original) American and (converted) Indian. These audio pairs can be used as source and target for training the seq2seq AC system. The voice conversion process is visualized in Figure 1.

3.1.2. Voice conversion

Our desired architecture should have the ability to change a speaker's identity and preserve the unique accent features (pronunciation pattern and prosody). We investigate two promising VC architectures, which are Fragment VC based on seq2seq architecture [7] and VQMIVC [4] based on variational autoencoder architecture. After the model is finetuned thoroughly by our data and with the same hyperparameters as in the original papers, we use this VC model to generate synthetic data.

FragmentVC. FragmentVC, a seq2seq architecture consists of a source encoder, a target encoder, and a decoder. The decoder takes one output feature from the source encoder (Q) and two output features from the target encoder (K, V). The output feature sequence of the target encoder (K) is then attended by the source encoder's output (Q). The cross-attention module in the decoder learns to align the source features to the target features with similar phonetic content in this architecture. The decoder then converts the Mel-spectrogram from the attention-augmented features. In the training phase, the same utterance is fed to the source, the target encoder and the decoder's reconstruction target. The encoders automatically learn to decouple the content and the speaker information without any explicit constraint. As the converted speech shares the same content (phonetic) with the source speech, the pronunciation and prosody (or the accent) will be preserved.

VQMIVC. VQMIVC (Vector quantization mutual information voice conversion) uses a straightforward autoencoder architecture to address the VC process. The framework is made up of four modules: a content encoder that produces a content embedding from speech, a speaker encoder that produces a speaker embedding (D-vector) from speech, a pitch encoder that produces prosody embedding from speech, and a decoder that reconstructs speech from content, prosody, and speaker embeddings. The phonetic and prosody are represented through content and prosody embedding. The content embedding is discretized by the vector quantization module and used as target for the contrastive predictive coding loss [11].

The mutual information (MI) loss measures the dependencies between all representations and can be effectively integrated into the training process to achieve speech representation disentanglement. During the conversion stage, the source speech is put into the content encoder and pitch encoder to extract content embedding and prosody embedding. To extract target speaker embedding, the target speech is sent to the speaker encoder. Finally, the decoder reconstructs the converted speech using the source speech's content and prosody embedding and the target speech's speaker embedding. Like in FragmentVC architecture, the converted speech will share the same content and prosody with the source speech, the accent is thus expected to be preserved by the model.

3.2. Accent conversion

3.2.1. Baseline approach

The baseline system is based on an encoder-decoder paradigm (Figure 2) with an attention mechanism. The encoder has two pyramid bi-LSTM layers, similiar to [9]. Each layer has 768 units and the down sample factor is 2. Input of the baseline encoder is 80 mel-filterbanks feature. On top of the encoder, we add a phoneme classifier layer and compute a connectionist temporal classification (CTC) loss.

The baseline decoder has a similar neural-network structure as the Tacotron speech synthesizer decoder [12] with an attention mechanism. The attention mechanism comprises three parts: query layer, key-value layer, and alignment layer. The first two layers produce query vector and sequence of key-value vectors which are of the same dimension. In other words, query, key, and value are processed to be in the same vector space. After that, the alignment layer learns the attention weights via a method deployed in this vector space. The first step of the de-

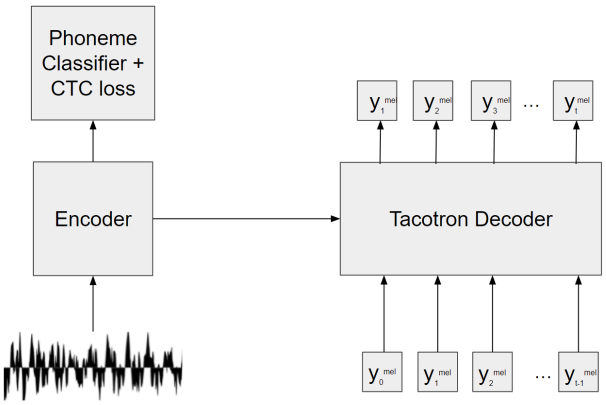

Figure 2: *Accent conversion architecture*

coding process composes query for the attention mechanism, defined as Eq 2 where s_i is the hidden-state of the attention-RNN layer, s_{i-1} and c_{i-1} are the previous hidden-state and context vector respectively. Then, the alignment layer of our attention mechanism is defined as Eq 3, where α_i is the attention weights at ith the decoding step, Query layer, Memory Layer, and Location Layer are modules of the alignment block. The attention mechanism considers three terms to calculate the attention weights. Beside the fundamental query and key-value, previous predictions of attention weights α_{i-1} are also fed into the alignment block. Thus, the attention mechanism in the Tacontron decoder is not only content-based but also location-sensitive. By this time, current attention weights are available, hence we calculate a weighted sum of the key-value sequence as shown in Eq 4. This weighted sum represents the context information and thus is called a context vector. After that, a decoder-RNN layer is defined as Eq 5, in which s_i and c_i are concatenated to be fed into this RNN-layer; d_i is the RNN hidden-state. Finally, the RNN hidden-state is fed to the final prediction layer and we get y_i^{mel} as the mel-spectrogram prediction at the decoding step i in Eq 6. To improve the spectrogram quality, the mel-spectrogram prediction is passed through a convolutional PostNet module as in Eq 7. The output of the tacotron decoder is 80 mel-filterbanks feature. We set the hyperparemeters of Tacotron decoder similar to this implementation [1].

$$h_{1...n}^{x} = Encoder(x_1...x_N) \tag{1}$$

$$s_i = RNNAtt(s_{i-1}, [c_i - 1; Prenet(y_{i-1}^{mel})]) \tag{2}$$

$$\alpha_i = Att(Query(s_i), Mem(h_{1...n}^{x}), Location(\alpha_{i-1})) \tag{3}$$

$$c_i = \sum_j \alpha_{i,j} * h_j \tag{4}$$

$$d_i = RNNDecoder(d_{i-1}, [s_i; c_i]) \tag{5}$$

$$y_i^{mel} = Linear([d_i; c_i]) \tag{6}$$

$$y_i^{Postnet} = Postnet(y_i^{mel}) \tag{7}$$

3.2.2. Proposed approach

Our proposed model leverages a pre-trained encoder from wav2vec 2.0 and and a speech synthesis decoder as the baseline's decoder.

[1]https://github.com/NVIDIA/tacotron2

Wav2vec 2.0 is a decent framework to learn high-quality speech representation from unlabeled audio data. It consists of two components: feature encoder and context encoder. The feature encoder contains temporal convolution layers, takes raw waveform as input and conducts speech representation. In the next step, they are fed to the transformer-based-context encoder to generate context representations with sequence-level information. In the pre-training phase, the model is optimized with a contrastive loss [11] to distinguish the true target from distractors. The input to the context encoder is masked partially. The speech representation is discretized by the vector quantization module and used as targets for the contrastive loss. The pre-trained model of Wav2Vec 2.0 Base trained on LibriSpeech that extracts 768-dimensional speech representations is used in this research. Unlike the baseline encoder, the input of the wav2vec encoder is a raw audio waveform. We use the implementation of Wav2vec from Fairseq [2]. On top of the wav2vec encoder, we add a phoneme classifier layer and compute CTC loss again. In the training phase, we freeze the feature encoder and fine-tune the whole parameters of the Wav2vec context encoder and Tacotron Decoder.

In the training phase, the final loss function of both baseline system and proposed system is Eq 8. We do not include the stop loss token in loss function because we assume an output audio has the same length as an input audio in the inference stage.

$$L = ||Y_{mel} - Y_{mel}^{Decoder}||_2 + ||Y_{mel} - Y_{Postnet}^{decoder}||_2 + Loss_{ctc} \tag{8}$$

In this paper, the WaveGlow network [13] is used as the neural vocoder. We also use the open-sourced Pytorch implementation. Since the mel-spectrogram captures all of the relevant details needed for high-quality speech synthesis, we simply use ground-truth 80 mel-filterbanks spectrograms to train the WaveGlow.

4. Experiments

4.1. Dataset and training description

To train the VC model, we used audio data from CMU-ARCTIC corpus [14] and L2-ARCTIC [15] Hindi-accented corpus. These corpuses have 8 speakers in total (4 native English speakers and 4 Indian accented speakers), and each speaker has audios of same 1152 sentences. In order to get better VC output on all speakers, we build on the pre-trained VC model in [4], [7] by finetuning them on these data corpuses without having to train the VC model from scratch.

In order to synthesize parallel data to train our AC model, we firstly split 1152 sentences into a training set (1052 sentences), a validation set (50 sentences), and a test set (50 sentences). Each sentence is spoken by 8 speakers (4 native and 4 Indian-accented speakers). After applying the data synthesizing process described in 3.1, we get 8 audio pairs for each sentence. In total, we have around 9000 audio pairs of approximately 9 hours in length. All of them are sampled at 16khz.

Both baseline and proposed AC models are trained on a single GPU by grouping with batch size 64 and the gradients are updated every 8 mini-batches with the Adam optimizer in using the base learning rate 0.5 and 4000 warm-up steps with the same learning rate schedule as Transformer model [16]. Both models are trained for at most 20000 steps

[2]https://github.com/pytorch/fairseq/blob/main/fairseq/models/wav2vec

4.2. Evaluation metrics

We conduct the evaluation on all models by using two kinds of metrics: objective and subjective, which are described in the following section. We use 50 sentences (200 American-accented audio files and 200 Indian-accented audio files) in the test set for evaluation. Sample evaluation audios can be found here [3].

4.2.1. Subjective tests

Accentedness test and speaker similarity test : Two tests are conducted by 10 Indian participants who listen to the provided audios and evaluate their accent features on a 5-point scale: 1-bad, 2-poor, 3-fair, 4-good, 5-excellent. To generate converted audios with an Indian accent, we apply our AC models on 200 American-accent audio files. For the accentedness test, the participants give a score out of 5 for the degree to which the converted audios sound like Indian-accented speeches. For the speaker similarity test, they rate the similarity between the voice identity of the input audio and the converted audio, then give a score out of 5 as above.

4.2.2. Objective tests

Word error rates (WER) and Accent classifier accuracy (ACC) : We compute a WER of both original Indian-accented audios and converted Indian-accented audios by our competitive ASR system [17]. If the gap between two WERs is lower, the quality of the converted audio is implied to be better. In addition, we design an accent classifier, which takes audio as an input and predict the accent of that input. The accent classifier is a 4-LSTM-layer network with 512 hidden units with an accent classifier layer on the top and trained on our training data set. We compute ACC on both converted and original audios. A better conversion model is expected to have a smaller gap between two ACCs.

4.3. Results

Models	Accentness	Speaker Simlarity
Original Indian Accented audio	4.8	
LSTM-based		
+FragmentVC Synthetic Data	2.5	3.8
+VQMIVC Synthetic Data	2.3	**4.1**
Wav2vec-based		
+FragmentVC Synthetic Data	**3.8**	3.1
+VQMIVC Synthetic Data	**3.7**	**3.6**

Table 1: *Subjective metrics*

The survey participated by Indian native speakers produces several subjective metrics as in Table 1. For the Accentedness test, the Wav2vec-based model outperforms the LSTM-based model on all synthetic data conditions (3.8 vs 2.5 and 3.6 vs 2.3). On the other hand, the LSTM-based model seems to have a better result in the speaker similarity test. Such observation implies that the Wav2vec model might have changed the audios to a greater extent compared to the LSTM-based model, which actually produces a better accent output yet makes the participants find the speaker identity altered more. The FragmentVC and VQMIVC receives fairly similar assessment in the Accentedness test (2.5 vs 2.3 and 3.8 vs 3.7). However, in the speaker similarity test, FragmentVC method does not have as good performance as VQMIVC. This means VQMIVC model is more effective than FragmentVC in VC; hence it can generate target ground-truth audios with similar voice to the source audios.

[3]https://tuannamnguyenkit.github.io

Models	WER	ACC
Original Indian Accented audio	9.7	96
Original US Accented audio	6.3	98
LSTM-based		
+FragmentVC Synthetic Data	42	63
+VQMIVC Synthetic Data	35	47
Wav2vec-based		
+FragmentVC Synthetic Data	24	**88**
+VQMIVC Synthetic Data	**18**	**87**

Table 2: *Objective metrics*

To determine the performance of the speech recognition and accent classifier, we apply them to the original Indian and American accented audios. On an Indian accented test set, the speech recognition model achieves a WER of 9.7%, whereas it is only 6.3% on an American accented test set. The accent classifier is also highly accurate, detecting 96% and 98% on Indian and American accented test sets respectively. We apply these models on our AC's audio results to compare different synthetic data and AC models. To compare the synthetic data models, we use the output provided by the same AC model. Training using VQMIVC Synthetic Data always results in an improved WER for both LSTM-based and Wav2vec-based models. This observation implies that the VQMIVC generates higher-quality training data than FragmentVC. In terms of ACC, the Wav2vec-based model yields comparable results for two synthetic models (88% and 87% for Fragment VC and VQMIVC respectively). However, the results from an LSTM-based model are somewhat different (63% and 47% correspondingly). However, this does not suggest that the Fragment VC preserves the accent better than the VQMIVC, since the LSTM-based model is not powerful enough to generate a good quality output. When comparing AC models, once again the Wav2vec-based has a better subjective result than LSTM-based on all data synthetic conditions.

5. Conclusions

In this paper, we have shown that a decent VC model could provide a convenient way to address the ground-truth data problem in AC. Additionally, how a Wav2vec pre-trained encoder can contribute to an AC model being trained in low-resource condition is also described. Throughout the experiment, we only focus on uni-directional (one-to-one) AC model. Given the advantage of the pre-trained audio encoder, our future goal is to apply this approach in training a multi-directional (many-to-many) AC with more accent data.

6. Acknowledgements

The projects on which this paper is based were funded by the Federal Ministry of Education and Research (BMBF) of Germany under the numbers 01IS18040A and 01EF1803B.

7. References

[1] G. Zhao, S. Sonsaat, J. Levis, E. Chukharev-Hudilainen, and R. Gutierrez-Osuna, "Accent conversion using phonetic posteriorgrams," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5314–5318.

[2] G. Zhao, S. Ding, and R. Gutierrez-Osuna, "Foreign accent conversion by synthesizing speech from phonetic posteriorgrams," in *Proc. Interspeech*, 2019. [Online]. Available: https://psi.engr.tamu.edu/wp-content/uploads/2019/07/zhao2019interspeech.pdf

[3] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[4] D. Wang, L. Deng, Y. T. Yeung, X. Chen, X. Liu, and H. Meng, "Vqmivc: Vector quantization and mutual information-based unsupervised speech representation disentanglement for one-shot voice conversion," 2021.

[5] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, "Autovc: Zero-shot voice style transfer with only autoencoder loss," 2019.

[6] J. chieh Chou, C. chieh Yeh, and H. yi Lee, "One-shot voice conversion by separating speaker and content representations with instance normalization," 2019.

[7] Y. Y. Lin, C.-M. Chien, J.-H. Lin, H. yi Lee, and L. shan Lee, "Fragmentvc: Any-to-any voice conversion by end-to-end extracting and fusing fine-grained voice fragments with attention," 2021.

[8] G. Zhao, S. Ding, and R. Gutierrez-Osuna, "Converting foreign accent speech without a reference," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2367–2381, 2021.

[9] S. Liu, D. Wang, Y. Cao, L. Sun, X. Wu, S. Kang, Z. Wu, X. Liu, D. Su, D. Yu, and H. Meng, "End-to-end accent conversion without using native utterances," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6289–6293.

[10] X. Li, C. Wang, Y. Tang, C. Tran, Y. Tang, J. Pino, A. Baevski, A. Conneau, and M. Auli, "Multilingual speech translation from efficient finetuning of pretrained models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 827–838. [Online]. Available: https://aclanthology.org/2021.acl-long.68

[11] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018. [Online]. Available: http://arxiv.org/abs/1807.03748

[12] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[13] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3617–3621.

[14] J. Kominek and A. Black, "The cmu arctic speech databases," *SSW5-2004*, 01 2004.

[15] G. Zhao, S. Sonsaat, A. Silpachai, I. Lucic, E. Chukharev-Hudilainen, J. Levis, and R. Gutierrez-Osuna, "L2-arctic: A non-native english speech corpus," in *Proc. Interspeech*, 2018, p. 2783–2787. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1110

[16] 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.

[17] T. N. Nguyen, T. S. Nguyen, C. Huber, N.-Q. Pham, T.-L. Ha, F. Schneider, and S. Stüker, "KIT's IWSLT 2021 offline speech translation system," in *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*. Bangkok, Thailand (online): Association for Computational Linguistics, Aug. 2021, pp. 125–130. [Online]. Available: https://aclanthology.org/2021.iwslt-1.13