Conformer-based Automatic Speech Recognition for Arabic Dialects

1st Amin HASSAIRI
dept. of Telecommunications and Artificial Intelligence
Budapest University of Technology and Economics
Budapest, Hungary
amin-hassairi@edu.bme.hu

2nd Péter MIHAJLIK
dept. of Telecommunications and Artificial Intelligence
Budapest University of Technology and Economics
Budapest, Hungary
mihajlik@tmit.bme.hu

***Abstract*—Automatic Speech Recognition has shown a significant upward trend in recent years. This paper investigates an ASR system for the Arabic language, developed using the Conformer-CTC character-based model within the NeMo framework. The system leverages the latest deep learning techniques, focusing on the conformer architecture combined with Connectionist Temporal Classification for sequence-to-sequence learning. The model is supervised, using labeled training data to map the input audio to text. The Mozilla Common Voice 11.0 dataset, which offers diverse spoken Arabic samples, is used for training. This paper details the model training process, including configuration setup, data processing, and optimization strategies. The performance of the model is evaluated, offering insights into the challenges and effectiveness of the Conformer-CTC character-based model for Arabic speech recognition tasks.**

***Index Terms*—ASR, Conformer-CTC, Character-based, Mozilla Common Voice, Arabic language, Deep Learning**

I. Introduction

Automatic Speech Recognition (ASR) has become a pivotal technology in various sectors such as healthcare, customer service, and entertainment.

However, while significant advances have been made in ASR for high-resource languages such as English, a considerable gap remains in the performance of ASR systems for low-resource languages, including Arabic [1] [2] [3] [4] [5]. Arabic presents unique challenges for ASR systems due to its rich morphology, complex sentence structures, and significant dialectal variations [5]. Unlike languages with standardized forms such as English, Arabic has multiple regional dialects that differ substantially from each other in phonology, vocabulary, and syntax. This variation complicates the task of creating a unified ASR system that can handle both Modern Standard Arabic (MSA) and its various dialects [5].

The Conformer-CTC model [6] [7] has emerged as one of the most promising approaches addressing challenges in ASR. The conformer network, which is designed to capture both short- and long-range dependencies in audio signals, is particularly well suited for speech recognition tasks. Its ability to process sequential data effectively, combined with the power of the Connectionist Temporal Classification (CTC) [7] loss function, makes it ideal for ASR tasks. CTC is particularly advantageous in speech recognition, as it allows the model to learn directly from raw audio features without the need for predefined alignments.

This work focuses on the development of an ASR system for the Arabic language, leveraging the Conformer-CTC architecture to improve performance in both Modern Standard Arabic and its dialects. The primary objective is to bridge the gap in the accuracy of ASR for Arabic, which is often hindered by the dialectal and morphological complexity.

The structure of the work is organized as follows: we first review the challenges in Arabic ASR, then introduce the Conformer-CTC model, followed by the methodology used in our experiments, and conclude with an analysis of the results and future directions for this research.

II. Related Work

Automatic Speech Recognition (ASR) for Arabic has seen significant advancements over the past few decades. Early approaches to Arabic ASR relied heavily on traditional methods such as Hidden Markov Models (HMMs) [8] combined with Gaussian Mixture Models (GMMs). In recent decade, however, there has been a notable shift in the field of ASR towards more advanced, fully data driven techniques. Deep learning, particularly end-to-end neural networks, has largely replaced traditional methods–Recurrent Neural Networks (RNNs) [9] and Convolutional Neural Networks (CNNs) have demonstrated substantial improvements in performance [1] [10].

Another breakthrough is that Transformer-based models [11], originally developed for machine translation, have revolutionized ASR systems and become the architecture of choice for various natural language processing (NLP) tasks. By utilizing self-attention [12] mechanisms, Transformers can capture long-range dependencies in sequential data, making them highly effective for speech recognition tasks. Many studies show that Transformer-based models, such as Whisper-Large [13] for Arabic, outperform others with a Word Error Rate (WER) of 13.28 and Character Error Rate (CER) of 4.23 on the Mozilla Common Voice

https://doi.org/10.3311/WINS2025-004

v11.0 test set, although results are not directly comparable to ours because of differences in training data sizes.

Building upon these advancements, the Conformer model, introduced in 2020, combines the strengths of both convolutional layers and self-attention mechanisms. This hybrid approach is particularly suited for sequence-based tasks like ASR. The convolutional layers capture local temporal features, while the self-attention mechanism is effective at modeling long-range dependencies. This architecture has proven to be highly effective in speech recognition tasks for languages such as English and Mandarin, achieving state-of-the-art results [12] [14] across various benchmark datasets.

However, despite the success of Conformer-based models in languages like English and Mandarin, their application to Arabic ASR [15] [16] [17] [18] remains underexplored. Arabic presents unique challenges for ASR, including its rich morphology, dialectal variation, and limited availability of labeled data. With significant dialectal diversity ranging from Modern Standard Arabic (MSA) to various regional dialects, each with its distinct pronunciation and vocabulary, ASR systems face considerable difficulty in achieving high accuracy across all dialects. Moreover, the scarcity of large-scale labeled datasets for Arabic speech has further hampered the development of robust ASR systems for the language.

To address these challenges, this work focuses on bridging the gap in Arabic ASR research by developing and evaluating a Conformer-CTC character-based model specifically designed for Arabic. Leveraging the Conformer's ability to model both local and long-range dependencies in speech, along with the flexibility of CTC (Connectionis Temporal Classification [7]), our goal is to create an effective Arabic-specific ASR system.

III. Methodology

In this work we have created an Automatic Speech Recognition (ASR) system based on a Conformer-CTC character-based model for Arabic language. The implementation steps of this system are as follows:

A. Data Collection, Analysis and Preprocessing

The dataset used in this study is the Mozilla Common Voice 11.0 Arabic language dataset [19], publicly available, including diverse Arabic audio recordings with transcriptions, ideal for training an ASR system.

The dataset is already divided by Mozilla Common Voice into train, validation, and test dataset. We kept it with this division in order to have a comparison with others previous work using the same dataset and division. The division ensures that the model is not evaluated on data it has already seen during training, providing a more reliable indication of its performance on unseen data.

The original audio format of our dataset is mp3. We convert all audio files to wav. format with sampling frequency 16000 Hz (Hertz), so it will align with our selected model Conformer-CTC character-based model, and would be better in the training process.

	Train	**Validation**	**Test**
Sentence Count	28043	10438	10440
Word Count	162962	57835	54301
Duration	31h 54m 22s	12h 44m 24s	12h 37m 25s

TABLE I: Overview of the Mozilla Common Voice dataset.

To ensure that the dataset is well-organized and easy to access during training, manifest files were created for train, test, and validation dataset, each manifest file contains multiple json objects and each json object contains three-fields as follows: **audio_filepath** field for the corresponding file path of the audio and a **duration** field which contains the duration of the audio file in seconds calculated during the process of the audio file, for the last field is the **text** field which contains the text transcript of the audio file. This clear organization streamlines the training process and minimizes potential issues when loading data into the model and to adapt this structure for the configuration of our selected model.

To ensure consistency in input data we standardize the data for efficient training, specifically the transcribed **text** input, all the last underwent several preprocessing steps. The preprocessing pipeline was implemented using the **PYARABIC** library, which provides specialized tools for Arabic language text processing and is well-suited for handling strings input. The specific preprocessing steps included:

Normalization: Removing Diacritics–Diacritical marks from all transcriptions was removed to ensure consistent input data, crucial for effective model training. Flattening characters reduces computational cost by simplifying the data while preserving essential details. Removing diacritics ensures that the system treats word variations as equivalent, enabling consistent recognition despite different accents in the dataset. Example: removing diacritics from this text
"إِنَّ الَّذِينَ آمَنُوا وَعَمِلُوا الصَّالِحَاتِ وَأَخْبَتُوا إِلَى رَبِّهِمْ أُولَئِكَ أَصْحَابُ الْجَنَّةِ هُمْ فِيهَا خَالِدُونَ"

to be like the following.
"إن الذين آمنوا وعملوا الصالحات وأخبتوا إلى ربهم أولئك أصحاب الجنة هم فيها خالدون"

Removing Elongation: the use of extra characters (like "___") in Arabic language are often introduced for aesthetic or emphasis purposes in written scripts or calligraphy. We ensure that all text instances are standardized into a more uniform form–ensuring consistency in our large dataset. When elongation is removed, all the words in the text dataset are reduced to their core form–phonetic form. This will facilitate text matching, search indexing, and retrieval since two instances of the same word, whether written with elongation or not, are treated as equivalent–make it easier to process for our ASR model.
Example: deleting elongation from this text

"يـــا ربنـــا احفـــظ لنـــا"

to be like the following.

"يا ربنا احفظ لنا"

Removing extra spaces: Ensuring that each word is separated by exactly one space–making the data more consistent and uniform. The presence of extra spaces can create inconsistencies in feature extraction. Representing each word as intended by deleting extra spaces–guaranteeing more accurate input to our Conformer-CTC model. Reducing redundancy, extra spaces add unnecessary characters to the text, which may only lead to space inefficiency and slow down the process. Apart from removing these extra spaces, we also remove the extra spaces that might be presented at the beginning and at the end of the text dataset.
Example: removing extra spaces between words, and at the beginning and the end of this text

"   ارجعي   الى ربك راضية مرضية   ."

to be like the following.

"ارجعي الى ربك راضية مرضية."

The preprocessing steps, using the **PYARABIC** library, ensured that the data were model-ready and optimized for efficiency, saving time and resources.

In summary, the Mozilla Common Voice 11.0 Arabic dataset [19] was pre-processed for optimal input of the Conformer-CTC model, including normalized transcriptions and efficient handling of train, test, and validation data, ensuring effective learning and evaluation.

B. Model Architecture

The Conformer-CTC character-based model used in this study combines convolutional and self-attention mechanisms, effectively handling local and global speech data dependencies for ASR tasks. As a non-autoregressive model built on the CTC loss function, it enables end-to-end training without explicit alignment between input and output sequences. Implemented using the NeMo framework [20] [21], this version replaces the original LSTM-based decoder with a simpler linear decoder, improving efficiency and speed, particularly for real-time applications.

The Conformer-CTC architecture uses convolutional layers to capture local speech patterns and multi-head self-attention layers for long-range dependencies. The self-attention mechanism, with absolute or relative positional encoding, helps handle varying input lengths and long-term dependencies.

The CTC loss function is a key component of the model. To map audio input sequences $X = [x_1, x_2, ..., x_T]$ to output transcription sequences $Y = [y_1, y_2, ..., y_U]$, challenges arise, such as the varying lengths of X and Y and a lack of alignment between them. CTC overcomes these issues by providing an output distribution over possible Y's for a given X. The goal is to maximize the probability $p(Y|X)$, which is differentiable to allow gradient descent. After training, the aim is to infer the most likely Y using $Y^* = argmaxp(Y|X)$. CTC offers an efficient, approximate solution for this inference process. Conformer-CTC is efficient for speech-to-text tasks, bypassing auto-regressive decoding to reduce inference time. The encoder output is passed through a linear decoder, generating character probabilities, which are decoded using methods like greedy decoding or beam search.

Conformer-CTC supports both character-level and subword-level encodings. The character-level variant maps acoustic features to individual characters, while the subword-level variant uses Byte Pair Encoding (BPE) to reduce model size and improve performance on unseen words. A schematic of the Conformer-CTC character-based model for Arabic is shown in Figure 1.

By combining convolution and self-attention, Conformer-CTC effectively balances local and global feature extraction. It captures local acoustic patterns through convolution and models long-range dependencies with self-attention, while maintaining the simplicity and efficiency of CTC-based decoding.

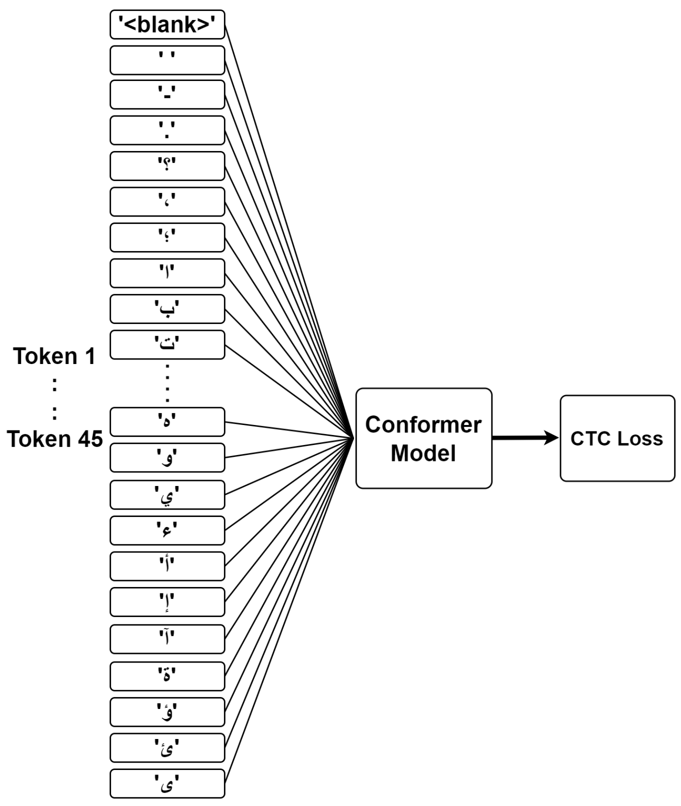

Fig. 1: The overall architecture of the Conformer-CTC Arabic language character-based model.

C. Configuration

The model configuration, written in YAML, is designed for character-based transcription at a 16,000 Hz sample rate. It uses the small Conformer version with a 16-layer encoder, 176 model dimensions, and a 4x feed-forward expansion. The encoder employs striding sub-sampling to reduce feature sequence length by a factor of 4 for efficiency.

The decoder is a **ConvASRDecoder**, designed for CTC tasks, where the output layer corresponds to labels defined in the **model.labels** section. It bypasses complex recurrent networks, directly producing probability distributions over characters for each time step.

D. Training Procedure

The model is trained on a single device using mixed-precision 16-bit computation for faster processing and accuracy. Training lasts up to 100 epochs with validation at each epoch. The system supports distributed data parallel (DDP) for faster training on multiple GPUs, though it's optimized for single-device operation.

Model checkpoints are saved throughout training, with the best models selected based on validation WER. The experiment manager logs progress and stores checkpoints in the designated directory. The integration of the **WandbLogger** [22] facilitates real-time monitoring and logging of training metrics.

The optimizer used is **Novograd** [23], chosen for its suitability in training large-scale models. The learning rate is set with different values 0.0001, 0.001 and 0.01, with the betas parameters defined differently depending on different experiments. Weight decay is applied with a value of 1×10^{-3} to reduce overfitting. The learning rate schedule is governed by **CosineAnnealing**, which adjusts the learning rate dynamically during training. A warm-up period of 1000 steps is defined, with the minimum learning rate set to 1×10^{-6} to avoid getting stuck in poor local minima during later stages of training.

The preprocessor converts raw audio into Mel spectrograms for ASR, using a 25 ms window size and a 10 ms stride.

E. Evaluation

This work uses Word Error Rate (WER) as the primary evaluation metric, as it is straightforward to calculate and enables flawless comparison between models. WER measures transcription errors, including substitutions, insertions, and deletions. The calculation formula is defined as follows:

$$\mathrm{WER} = \frac{S + D + I}{N}$$

where, S represents the number of substituted words, D represents the number of deleted words, I represents the number of inserted words and N represents the total number of words in the reference.
Example: the ground truth is.

"لم نسكن هنا قط"

the transcription by the model is.

"لم نعيش هنا قطة"

WER calculation:
S=2

نسكن -> نعيش

قط -> قطة

I=0
D=0

$$\mathrm{WER} = \frac{S + D + I}{TotalWords} = \frac{2 + 0 + 0}{4} = \frac{2}{4} = 0.5$$

To evaluate the model more, the Character Error Rate (CER) is used as well, CER is similar to WER but operates at the character level–it is calculated using S, I, and D of character instead of words. Calculating the CER for the same previous example: characters in the ground truth.

[ل,م" "ن,س,ك,ن" "ه,ن,ا" "ق,ط]

Total number of characters is equal to 14.
Characters in the model transcription.

[ل,م" "ن,ع,ي,ش" "ه,ن,ا" "ق,ط,ة]

S= 3

نسكن -> نعيش | س->ع | ك->ي | ن->ش

I=1 adding

ة to قط

D=0

$$\mathrm{CER} = \frac{S + D + I}{TotalCharacters} = \frac{3 + 0 + 1}{15} = \frac{4}{14} = 0.29$$

F. Hardware Configuration

The experiments in this study were conducted on the Ubuntu 20.04 operating system, utilizing the PyTorch Lightning deep learning framework.

Name	**Model/Parameter**
Linux Version	5.4.0-200-generic
Operating System	Ubuntu 20.04
GPU	1 × RTX2080 Super
CUDA Version	12.2
Deep Learning Framework	PyTorch 2.5.1
Programming Language	Python 3.10

TABLE II: Experimental Environment Configuration

G. Hyperparameter Tuning

Hyperparameter tuning was performed to optimize model performance through experimentation with several key parameters on the validation dataset:

Learning Rate: A range of learning rates was tested, between 1×10^{-4} and 1×10^{-2} , to determine the best value for stable and efficient convergence.

The model was trained for up to 100 epochs, with early stopping based on validation performance to prevent overfitting and save time.

Other hyperparameters, such as the optimizer's betas and configurations, were fine-tuned to balance training speed and accuracy.

IV. Results, Evaluation and Discussion

A. Results

We trained a different Conformer-CTC Arabic language character-based model, with different hyperparameter. The performance of the model was quantitatively evaluated using the Word Error rate (WER), as shown in Table III below. These models are represented with different colors following the output of the **wandb** logger. We noted the best-performed hyperparameter model (based on the Word Error Rate) in italic. The "Yellow" (Model D) Conformer-CTC model consistently outperforms in terms of transcription accuracy.

Model	Learning Rate	Betas	Best WER
Orange (Model A)	0.001	[0.9,0.999]	0.61141
Pink (Model B)	0.001	[0.8,0.6]	0.69448
Purple (Model C)	0.001	[0.8,0.9]	0.69775
Yellow (Model D)	*0.01*	*[0.9,0.999]*	*0.43563*
Seafoam (Model E)	0.0001	[0.6,0.8]	0.83302

TABLE III: Overview of the performance results of Conformer-CTC Character-based model with various hyperparameters, evaluated on the **validation** dataset (see Table I.)

We obtained these chart results after training using wandb—The validation loss and the Validation WER, as shown in the Figure 2 and 3, respectively. From both figures, we can observe that the curve of the "Yellow" (Model D) model is below all the other models, in both metrics: Validation loss and Validation WER. Despite the "Yellow" model's instability during training, indicated by the large spikes in the curves, it achieved the lowest Validation WER, making it the best-performing model.

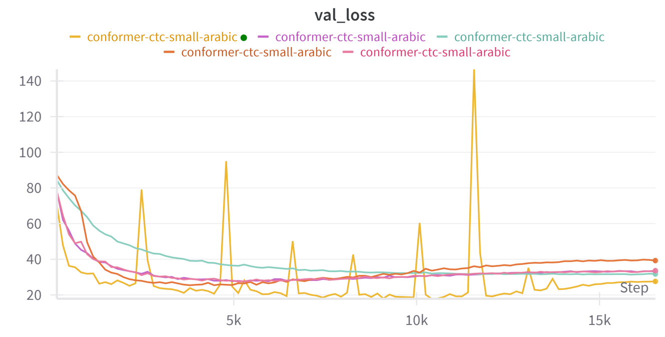

Fig. 2: Validation Loss over Training Progress.

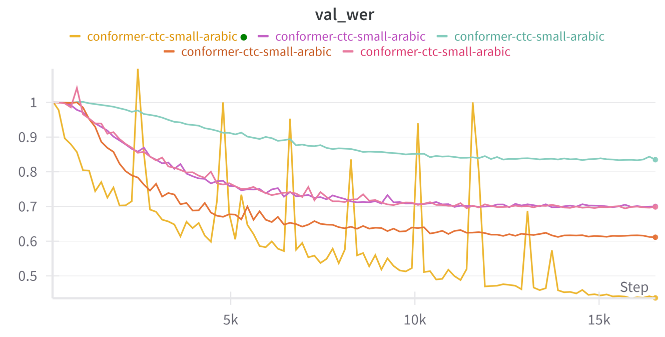

Fig. 3: Validation WER over Training Progress.

Building on this success, and due to time constraints, a more exhaustive hyperparameter optimization—such as further tuning of the learning rate—was not pursued after the "Yellow" model's strong performance.

B. Evaluation

Using the test dataset we evaluated our trained models using this unseen dataset. For each model we generated a prediction text from the audio file. Then comparing the last with the ground truth–text to calculate Word Error Rate (WER) as well as the Character Error Rate (CER), we got the following results in table IV:

Model	WER	CER
Orange (Model A)	68.78	25.86
Pink (Model B)	75.64	29.46
Purple (Model C)	75.46	29.46
Yellow (Model D)	*52.28*	*18.29*
Seafoam (Model E)	99.95	84.72

TABLE IV: Overview of the evaluation results of Conformer-CTC Character-based models, evaluated on the **test** dataset (see Table I.)

C. Discussion

The Conformer-CTC model's performance on the Arabic ASR task was evaluated using the Word Error Rate (WER), the standard ASR accuracy metric. Along with WER, the Character Error Rate (CER) was used to provide a more comprehensive evaluation of the model's performance at the character level, crucial for assessing its effectiveness in Arabic ASR. Trained over 100 epochs with regular validation, the model showed competitive performance, particularly in handling Arabic's rich morphology and phonetic variability. One of its key strengths was managing the complexities of Arabic pronunciation, including dialectal variations, enabling effective recognition across different forms and conditions while maintaining reasonable accuracy.

Challenges: We encountered several challenges during the training of the Conformer-CTC model such as the phonetic complexity–Arabic language contains a variety of homophones, where words sound the same but have different meaning. The last with the combination of multiple regional dialects from the Mozilla Common Voice 11.0 dataset (which is rich in dialects and including variety of voices that represent different ages, which contribute to its richness in terms of morphological diversity) adds more complexity to the ASR system.

To address this issue, a character-level tokenizer was used – simplifying tokenization by focusing on individual characters as well as removing diacritics, thus reducing complexity associated with diacritics.

Limitation and Future Work: While the Conformer-CTC Arabic language character-based model has shown competitive results, there is significant potential for further improvement. Future enhancements could focus on incorporating more advanced data augmentation techniques, which would help the model become more robust to variations in speech patterns and noise levels. Although the initial training was successful, fine-tuning the model with more data would likely boost its performance, yielding better results. Additionally, exploring larger architecture versions of the Conformer-CTC model presents another avenue for improvement. For instance, we implemented the "Small" version with 13 million parameters, but there are also "Medium" (30 million parameters), "Large" (121 million parameters), and "XLarge" (635 million parameters) versions available. Using these larger versions of the Conformer-CTC architecture may improve performance by allowing the model to capture more complex features of speech audio. With more parameters, the model can generalize better to unseen data, especially in noisy environments or scenarios involving speech overlap, ultimately leading to more accurate transcriptions and better handling of variations in speech input. Furthermore, a more exhaustive hyperparameter optimization approach could further fine-tune the model's performance, enhancing its ability to adapt to different speech conditions and providing more accurate results across a wider range of inputs.

V. Conclusion

In this study, we explored the potential of Conformer-CTC for Arabic language character-based architecture model in Automatic Speech Recognition, specifically focusing on the small model architecture. By training this model on the Mozilla Common Voice 11.0 Arabic language dataset, we could build a relatively well performing Conformer-CTC model which has demonstrated a significant potential for Arabic ASR, but there are several avenues for further improvement. By implementing data augmentation techniques, fine tuning the model with larger datasets, exploring larger Conformer-CTC architectures, and conducting more exhaustive hyperparameter optimization, the model's performance can be significantly enhanced. These strategies will facilitate to create a more robust ASR system, allowing to handle complexities of Arabic speech across multiple dialects.

Future studies might concentrate on the continuation of leveraging the model architecture and data diversity to enhance and boost the performance of the Conformer-CTC model in the context of Arabic language.

References

[1] S. Nasr, R. Duwairi, and M. Quwaider, "End-to-end speech recognition for arabic dialects," *Arab J. Sci. Eng.*, vol. 48, no. 12, pp. 10 617–10 633, 2023.

[2] F. Z. Besdouri, I. Zribi, and L. H. Belguith, "Arabic automatic speech recognition: Challenges and progress," *Speech Communication*, vol. 163, p. 103110, 2024.

[3] A. Rahman, M. M. Kabir, M. F. Mridha, M. Alatiyyah, H. F. Alhasson, and S. S. Alharbi, "Arabic speech recognition: Advancement and challenges," *IEEE Access*, vol. 12, pp. 39 689–39 716, 2024.

[4] I. Hamed, P. Denisov, C.-Y. Li, M. Elmahdy, S. Abdennadher, and N. T. Vu, "Investigations on speech recognition systems for low-resource dialectal arabic–english code-switching speech," *Comput. Speech Lang.*, vol. 72, Mar. 2022.

[5] A. Ali, S. Chowdhury, M. Afify, W. El-Hajj, H. Hajj, M. Abbas, A. Hussein, N. Ghneim, M. Abushariah, and A. Alqudah, "Connecting arabs: bridging the gap in dialectal speech recognition," *Commun. ACM*, vol. 64, no. 4, pp. 124–129, Apr. 2021.

[6] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, May 2020. [Online]. Available: https://arxiv.org/abs/2005.08100

[7] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," pp. 369–376, 2006.

[8] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[9] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," vol. 32, pp. 1764–1772, 2014.

[10] Y. Zhan, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent, Y. Bengio, and A. Courville, "Towards end-to-end speech recognition with deep convolutional neural networks," 2016.

[11] M. Sameer, A. Talib, A. Hussein, and H. Husni, "Arabic speech recognition based on encoder-decoder architecture of transformer," *Journal of Techniques*, vol. 5, no. 1, pp. 178–182, 2023.

[12] J. Salazar, K. Kirchhoff, and Z. Huang, "Self-attention networks for connectionist temporal classification in speech recognition," *arXiv preprint arXiv:1901.10055*, 2019. [Online]. Available: https://arxiv.org/abs/1901.10055

[13] B. Talafha, A. Waheed, and M. Abdul-Mageed, "N-shot benchmarking of whisper on diverse arabic speech recognition," in *INTERSPEECH 2023*, 2023. [Online]. Available: https://arxiv.org/abs/2306.02902

[14] M. Chen, P. Liu, H. Yang, and H. Wang, "Towards end-to-end unified recognition for mandarin and cantonese," in *Proc. Interspeech*, Kos, Greece, 2024, pp. 1–5.

[15] O. Aitoulghazi, A. Jaafari, and A. Mourhir, "Darspeech: An automatic speech recognition system for the moroccan dialect," in *Proc. 2022 Int. Conf. on Intelligent Systems and Computer Vision (ISCV)*, 2022, pp. 1–6.

[16] D. Wang, S. Ye, X. Hu, S. Li, and X. Xu, "An end-to-end dialect identification system with transfer learning from a multilingual automatic speech recognition model," in *Proc. Interspeech*, 2021, pp. 3266–3270.

[17] M. Labied, A. Belangour, and M. Banane, "Delve deep into end-to-end automatic speech recognition models," in *Proc. 2023 Int. Seminar on Application for Technology of Information and Communication (iSemantic)*, 2023, pp. 164–169.

[18] S. A. Chowdhury, A. Hussein, A. Abdelali, and A. Ali, "Towards one model to rule all: Multilingual strategy for dialectal code-switching arabic asr," *arXiv preprint arXiv:2105.14779*, 2021. [Online]. Available: https://arxiv.org/abs/2105.14779

[19] Mozilla, "Common voice: A mozilla project to build a global open-source voice dataset," 2024, accessed: Nov. 29, 2024. [Online]. Available: https://commonvoice.mozilla.org/en

[20] NVIDIA, "Nemo framework," 2024, accessed: Nov. 29, 2024. [Online]. Available: https://github.com/NVIDIA/NeMo/tree/main

[21] ——, "Nemo toolkit asr models user guide," 2024, accessed: Nov. 27, 2024. [Online]. Available: https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/asr/models.html

[22] W. . Biases, "Weights and biases: A platform for machine learning experiment tracking," 2024, accessed: Nov. 29, 2024. [Online]. Available: https://wandb.ai/site

[23] B. Ginsburg, P. Castonguay, O. Hrinchuk, O. Kuchaiev, R. Leary, V. Lavrukhin, J. Li, H. Nguyen, Y. Zhang, and J. M. Cohen, "Training deep networks with stochastic gradient normalized by layerwise adaptive second moments," *arXiv preprint arXiv:1905.11286*, 2019. [Online]. Available: https://arxiv.org/abs/1905.11286