Knowledge and Decision Systems with Applications

www.kadsa.org
eISSN: 3079-6318

Arabic Text Diacritization Using Deep Neural Networks and Transformer-Based Architectures

Mohamed Cherradi[1,*], Hajar El Mahajer[2]

[1] Abdelmalek Essaâdi University (UAE), ENSAH, Tetouan, Morocco
[2] Abdelmalek Essaâdi University (UAE), FSTT, Tetouan, Morocco

ARTICLE INFO

Article history:
Received 03 June 2025
Received in revised form 27 July 2025
Accepted 18 August 2025
Available online 28 August 2025

Keywords:
Arabic Diacritization; Natural Language Processing (NLP); Sequence-to-Sequence Models; Transformer Architecture

ABSTRACT

This study investigates the application of deep learning architectures for automatic Arabic text diacritization, with a particular focus on character-level neural networks. Four architectures were implemented: a Transformer encoder-decoder, a BiGRU model, a baseline stacked BiLSTM, and a CBHG model. Diacritic Error Rate (DER) and Word Error Rate (WER) were used as evaluation metrics, with training and evaluation conducted on the Tashkeela corpus. The results show that the CBHG model achieved faster inference times while slightly outperforming the Transformer encoder-decoder in diacritic accuracy. However, the findings also suggest that the Transformer model may yield better performance with larger datasets, improved parameter tuning, and increased model capacity.

1. Introduction

Arabic is currently the fourth most widely used language on the internet, largely due to the rapid expansion of digital Arabic content [1]. Spoken by hundreds of millions of people and recognized as an official language in more than a dozen countries [2], Arabic has become increasingly important in the global digital landscape. As the volume and diversity of Arabic electronic corpora grow, there is a pressing need for robust technologies capable of processing this data for various critical applications [3].

However, the advancement of Arabic Natural Language Processing (NLP) faces significant challenges due to the language's unique linguistic and orthographic features. Arabic is written in a cursive, right-to-left script with context-sensitive character forms and relies heavily on diacritical marks to convey meaning. These characteristics complicate the development of computational models that can effectively handle Arabic text [4].

In response, numerous automatic diacritization methods have been proposed, ranging from rule-based and statistical techniques to more recent approaches based on Artificial Intelligence (AI), particularly Deep Learning (DL) and Neural Networks (NN) [5]. Recent state-of-the-art systems, such

* *Corresponding author.*
E-mail address: m.cherradi@uae.ac.ma

https://doi.org/10.59543/kadsa.v1i.15077

as those used in speech recognition or machine translation, demonstrate the effectiveness of DL in Arabic language tasks [6].

This study investigates and evaluates deep learning-based approaches for automatic Arabic diacritization, with a particular focus on Transformer-based architectures. By comparing the performance of various models, this work aims to identify the most effective strategies for restoring diacritics in Arabic text and to contribute to the advancement of NLP tools for this morphologically rich language.

The remainder of this paper is structured as follows: Section 2 reviews previous research and existing approaches to Arabic diacritization. Section 3 details the proposed methodology, including the architectures and training settings used. Section 4 presents the experimental results and discusses the performance of the evaluated models. Finally, Section 5 concludes the paper and outlines directions for future research.

2. Related works

Automatic diacritization of Arabic text is one of the most challenging tasks in Arabic Natural Language Processing (NLP). Researchers have explored several approaches to tackle this problem, including rule-based, statistical, hybrid, and deep learning methods.

Rule-based techniques rely on a deep understanding of Arabic grammar and morphology to construct a predefined set of linguistic rules for diacritic restoration. Although such systems can be effective in specific contexts, they often lack generalizability. Some systems have combined rule-based approaches with statistical or machine learning methods. For example, El-Imam [7] employed this strategy in a text-to-speech system, while Cherradi and El Haddadi [8] used it in various NLP applications such as machine translation and named entity recognition.

Purely statistical approaches have also been widely explored. One early study [9] developed a Hidden Markov Model (HMM) for both Hebrew and Arabic. Using the Quranic corpus for Arabic, the system achieved a Word Error Rate (WER) of 14% with case endings (CE). Nelken and Shieber [10] proposed a system based on weighted finite-state transducers and a combination of three probabilistic language models: a word-based model, a letter-based model, and an orthographic model. Their best configuration, using trigram word models with clitic concatenation and four-gram letter models, achieved 23.61% WER and 12.79% Diacritic Error Rate (DER) with CE, and 7.33% WER and 6.35% DER without CE [11].

In recent years, Deep Neural Networks (DNNs) have shown considerable improvements in Arabic diacritization. Many models are built using sequential architectures based on Recurrent Neural Networks (RNNs) combined with fully connected layers. Al Sallab et al. [12] developed a system using DNNs with Confused Sub-Classes Resolution (CSR), achieving 12.7% WER and 3.8% DER with CE. Abandah et al. [13] introduced a model based on Long Short-Term Memory (LSTM) networks that significantly improved performance, obtaining 5.82% WER and 2.09% DER with CE, and 3.54% WER and 1.28% DER without CE.

Other works have explored variations of deep learning models. For instance, [14] evaluated several architectures and configurations, with their best model—a three-layer bidirectional LSTM—achieving 8.14% WER and 5.08% DER with CE. Fadel et al. [15] used Feedforward Neural Networks (FFNNs) and RNNs combined with techniques such as one-hot encoding, embeddings, Conditional Random Fields (CRF), and Block-Normalized Gradient (BNG), reaching 7.69% WER and 2.60% DER with CE, and 4.57% WER and 2.11% DER without CE.

Mubarak et al. [16] proposed a character-level sequence-to-sequence model using a Neural Machine Translation (NMT) framework applied to overlapping word windows. Their system achieved

state-of-the-art results with 4.49% WER and 1.21% DER with CE. The encoder-decoder architecture proposed in this paper is similar to theirs, but is adapted from a text-to-speech (TTS) model with significant modifications to suit the diacritization task. Notably, our model employs location-sensitive attention [17], whereas Mubarak et al. used content-based attention [18]. Finally, Al-Thubaity et al. [19] developed a model combining bidirectional LSTM networks with a CRF layer, achieving 4.92% WER and 1.34% DER on the Holy Quran corpus.

3. Methodology

In this section, we present the implementation of four character-level deep learning architectures for Arabic text diacritization. The first model is a baseline sequence model based on stacked bidirectional Long Short-Term Memory (BiLSTM) layers combined with an embedding layer, referred to as the RNN model. It serves as a reference to evaluate the performance of basic recurrent architectures on the Tashkeela corpus. The second model, named Stacked BiGRU, replaces the LSTM units with Gated Recurrent Units (GRUs) to reduce computational complexity while preserving temporal dependencies. The third model integrates a CBHG module (Convolutional Bank, Highway network, and Bi-GRU), inspired by the Tacotron architecture originally developed for speech synthesis. This model combines convolutional, recurrent, and highway layers to capture both local and long-range sequential patterns effectively. Finally, we implement a full Transformer Encoder-Decoder model, adapted from sequence-to-sequence architectures used in neural machine translation, where the encoder processes the undiacritized input and the decoder predicts the corresponding diacritics sequence. In the following subsections, we describe the corpus and preprocessing steps, training strategies, and evaluation metrics used to assess the effectiveness of each model on the Arabic diacritization task.

3.1 Diacritization Corpora

Training data is a critical component for any deep learning model. Unfortunately, there is no widely accepted standard corpus for diacritic restoration in the Arabic language. Among the available datasets, two notable corpora exist: the ATL corpus (which is proprietary and requires payment) and the freely accessible Tashkeela corpus. In this study, we utilize the Tashkeela corpus, which consists of 97 classical Arabic texts totaling approximately 75 million fully vocalized words. This rich dataset provides a substantial resource for training and evaluating automatic Arabic diacritization models.

3.2 Corpus Preprocessing

Effective preprocessing of the corpus is crucial to ensure the quality and consistency of the input data for deep learning models. This process involves multiple steps designed to clean, normalize, and structure the raw text data before feeding it into the models. As depicted in Figure 1, the preprocessing pipeline includes data cleaning, transliteration, diacritic normalization, tokenization, special token management, and feature extraction.

- Data Cleaning: All characters except Arabic letters, diacritics, and punctuation marks are removed from the corpus. This step is essential to retain only the characters relevant to the diacritization task, allowing the models to focus on meaningful inputs.
- Transliteration: The cleaned Arabic text is then transliterated into Buckwalter encoding, a system that maps Arabic letters to ASCII characters. For example, the Arabic letter ك is transliterated as "k", and ت as "t". This transliteration facilitates character-level tokenization and standardizes the input format for neural network models.

- Diacritic Normalization: This step ensures a consistent ordering of combined diacritics. For instance, sequences such as "a~" (fatha followed by shadda) are normalized to "~a" (with shadda preceding fatha). Normalizing the diacritic order prevents alignment errors during tokenization and improves the accuracy of the model's diacritic tagging.
- Tokenization: The transliterated text is segmented into (character, diacritic) pairs. If a character does not carry a diacritic, a special tag such as <NT> (No Tashkeel) is assigned. This token-level annotation is vital for training sequence labeling models that predict the correct diacritic for each character.
- Special Token Management: Special tokens are added to define the boundaries and structure of sequences, including: <BOS> (Beginning of Sequence), <EOS> (End of Sequence), <PAD> (used to pad shorter sequences), <MASK> (optional, used in models like Transformers); These tokens help the model understand sequence limits and support batch training with inputs of variable length.
- Feature Extraction: During model training, embeddings are learned automatically through backpropagation. Each input character or word is initially transformed into a dense vector via an embedding layer initialized with numeric values. These vectors serve as inputs to recurrent layers such as LSTM or GRU, which make predictions. The error between predicted and actual labels is calculated and propagated backward through the network, updating all model parameters, including the weights of the embedding layer.

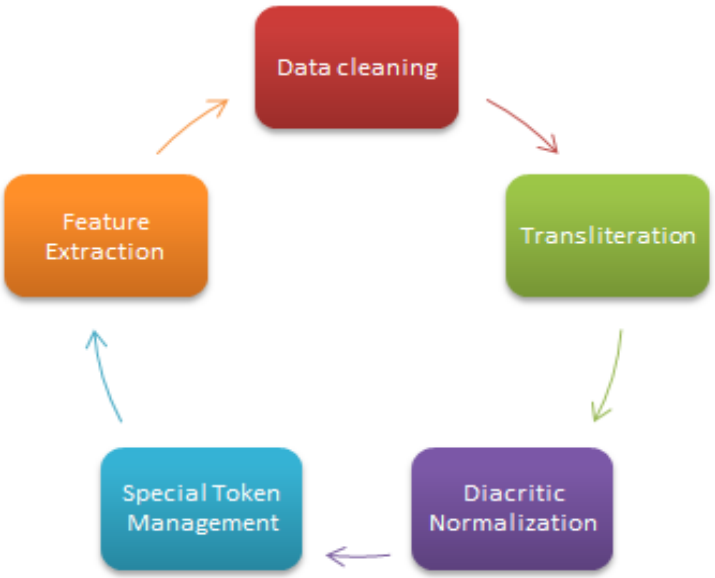

Fig. 1. Data preprocessing pipeline.

3.3 Baseline Model

The objective of this model is to evaluate the performance of a lightweight recurrent neural network architecture for Arabic text diacritization. The model begins with a 128-dimensional embedding layer that converts each input token into a dense vector representation. This is followed by one or more stacked bidirectional Long Short-Term Memory (BiLSTM) layers, each consisting of 128 hidden units in both the forward and backward directions. These recurrent layers are designed to capture contextual dependencies from both the past and future within the character sequence.

The output of the final BiLSTM layer is passed through a fully connected (dense) layer equipped with a softmax activation function, producing a probability distribution over a predefined set of diacritic classes. In total, the model predicts among 15 classes, including all common Arabic diacritics as well as a special class for characters without diacritics. The total number of trainable parameters depends on the number of BiLSTM layers used. For a single-layer configuration, the model contains approximately X million parameters (to be computed based on the actual vocabulary size and input sequence length). This architecture serves as a baseline for evaluating the performance of more sophisticated models such as the CBHG module and Transformer-based encoder-decoder frameworks. Figure 2 depicts the architecture of the baseline model, illustrating the embedding layer, stacked BiLSTM layers, and the final softmax classification layer.

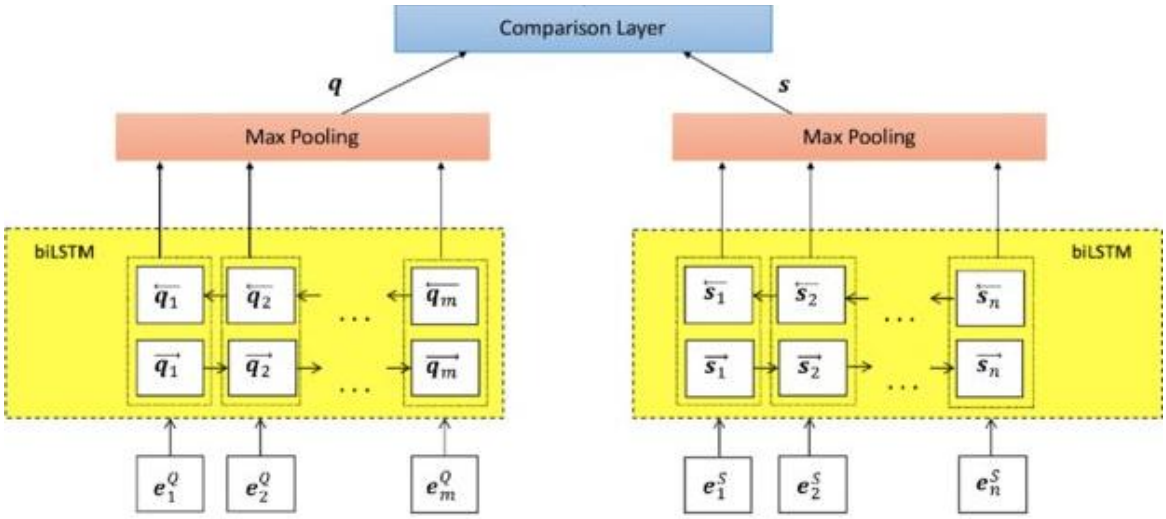

Fig. 2. Architecture of the baseline model [20]

Mat *et al*., [7] has performed a comprehensive flow visualization study on blunt-edge delta wing. The primary vortex is developed at certain chordwise position and progress upstream with angle of attack; however, there is no data in VFE-2 indicating that the vortex progressed up to the Apex region with angle of attack increases.

3.4 BiGRU Model

This architecture is designed to perform Arabic text diacritization using a compact and computationally efficient recurrent neural network. It starts with a 128-dimensional embedding layer, which transforms each input token into a dense vector representation suitable for processing by the network. The embedded sequence is then passed through two stacked bidirectional Gated Recurrent Unit (BiGRU) layers, each comprising 128 hidden units in both the forward and backward directions. These recurrent layers enable the model to capture contextual dependencies from both preceding and succeeding tokens, which is essential for accurately predicting diacritics in context-sensitive scripts like Arabic. The final output is produced by a fully connected layer, which maps the output of the last BiGRU layer to a set of diacritic classes. Instead of applying a softmax activation, the model outputs raw logits, making it compatible with training using loss functions such as sparse categorical cross-entropy with logits. The total number of trainable parameters depends on the vocabulary size and input sequence length, but remains relatively low compared to more complex architectures. This makes the BiGRU model an attractive option when balancing performance and

computational efficiency. Figure 3 illustrates the overall architecture of the BiGRU model, highlighting the embedding layer, the stacked bidirectional GRU layers, and the output layer producing raw logits for classification.

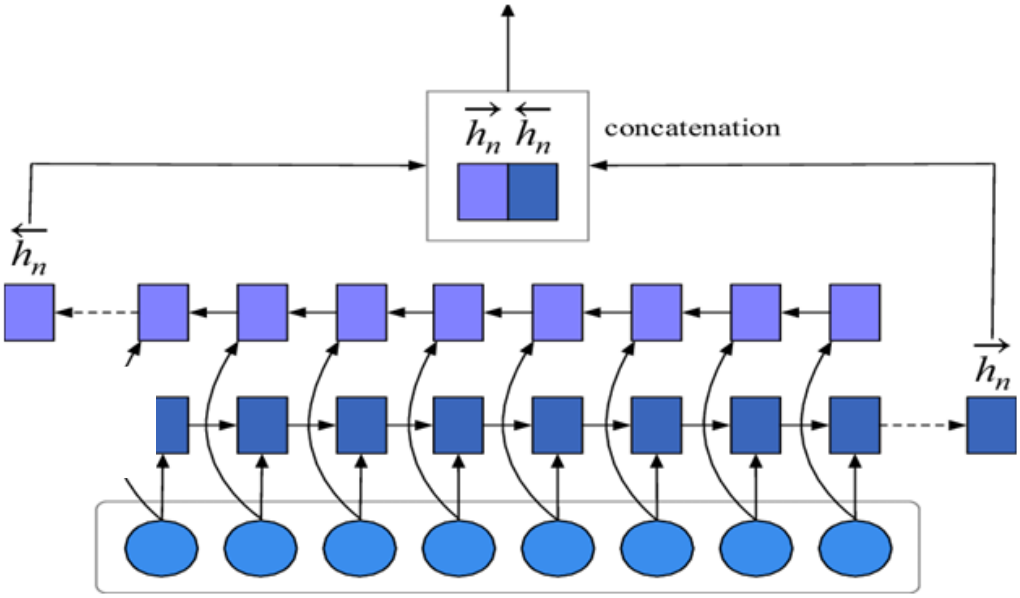

Fig. 3. Architecture of the BiGRU model [21]

3.5 CBHG Model

The CBHG model is designed as an efficient encoder-based architecture for Arabic text diacritization. Unlike typical encoder-decoder frameworks used in sequence-to-sequence tasks—where the input and output sequence lengths may differ—diacritization has a one-to-one alignment between input and output tokens. This allows us to simplify the architecture by using only the encoder component, without the need for an autoregressive decoder. The core of this model is the CBHG module (Convolutional Bank, Highway network, and Bidirectional GRU), originally proposed in the Tacotron architecture for speech synthesis. We adapted it for the diacritization task by integrating a fully connected projection layer followed by a softmax layer, enabling the model to predict a probability distribution over possible diacritics for each character (see Figure 4).

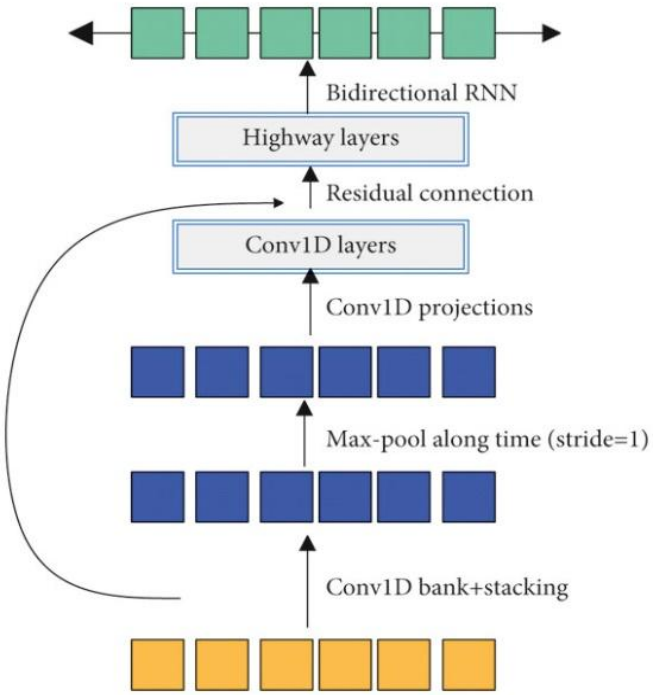

Fig. 4. Architecture of the CBHG model [22]

The architecture consists of several sequential components. It begins with a 128-dimensional embedding layer that processes the input sequence, transforming each character into a dense vector representation. The resulting embeddings are then passed through a two-layer pre-net, where each layer contains 128 units with ReLU activation and a dropout rate of 0.3. This pre-net introduces non-linearity and regularization, preparing the representations for the core CBHG module. The CBHG block, which combines a convolutional bank, max-pooling, highway networks, and a bidirectional GRU, captures both local and long-range dependencies in the character sequence. The output of the CBHG module is then passed through a fully connected layer that projects the sequence representations onto the space of possible diacritic classes. Finally, a softmax layer is applied to produce a probability distribution over the diacritic labels for each character in the sequence.

The model contains approximately 14 million trainable parameters, which is significantly more than the Transformer-based encoder-decoder model. However, it offers faster inference because it predicts all diacritics in parallel, as opposed to the sequential decoding used in typical encoder-decoder setups.

3.6 Transformer encoder-decoder Model

The Transformer Encoder-Decoder (ED) model employed in this study follows a standard sequence-to-sequence architecture. It consists of an encoder and a decoder, each comprising multiple layers with an internal dimensionality of 512 and 16 attention heads. The model is initialized randomly and trained from scratch, without requiring any pretraining on external corpora. This architecture directly leverages fully diacritized data, such as the Tashkeela corpus, to model the diacritization task as a translation problem. The input sequence consists of unvocalized Arabic text, while the output sequence represents the corresponding diacritics. At each decoding step, the model generates one diacritic mark, conditioned on the entire unvocalized input and all previously

predicted diacritics. This auto-regressive generation allows the decoder to take into account both global context and historical predictions, thereby improving the quality of sequential labeling. Figure 5 illustrates the Transformer encoder-decoder architecture, showing the flow from unvocalized input through the encoder, and the step-by-step generation of diacritics by the decoder.

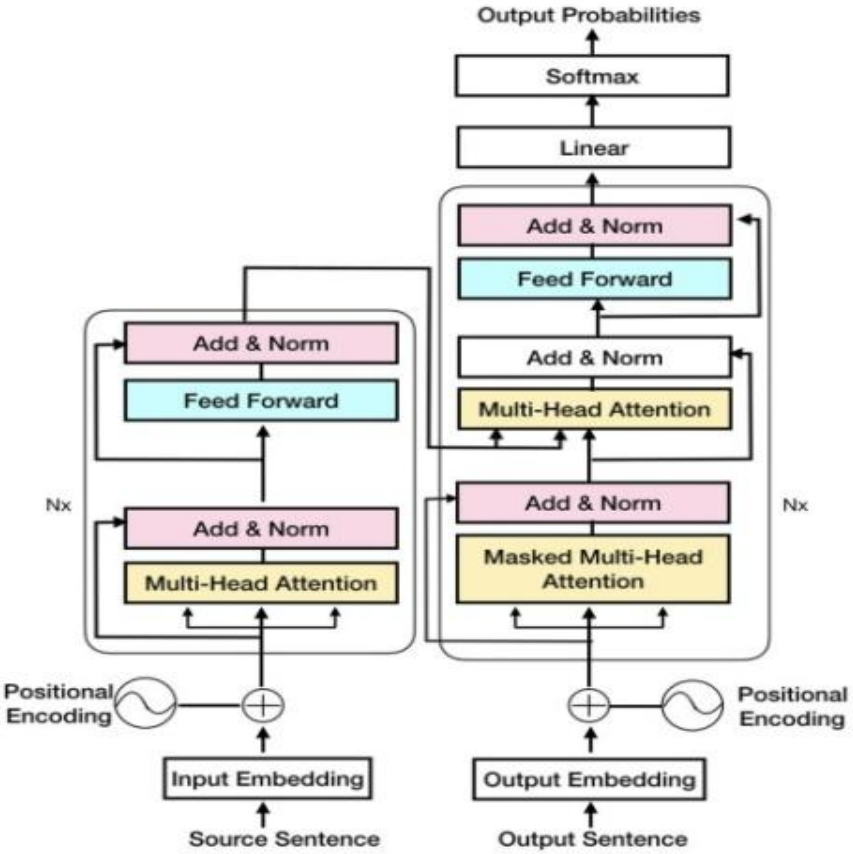

Fig. 5. Architecture of the Transformer encoder-decoder model [23]

3.7 Diacritization systems evaluation

The performance of Arabic diacritization systems is commonly assessed using two standard metrics: Diacritization Error Rate (DER) and Word Error Rate (WER). Both metrics can be computed with or without Case Ending (CE), depending on whether the final character of each word is included in the evaluation. When evaluated without CE, the last character of each word is excluded, since its diacritic is generally determined by grammatical and syntactic context. As a result, DER and WER without CE usually yield lower (i.e., better) error rates, as they reflect the system's ability to recover internal diacritics independent of syntactic rules. In contrast, metrics with CE provide a measure of overall performance, including grammatical accuracy.

The Diacritization Error Rate (DER) measures the proportion of characters for which the diacritic is incorrectly predicted. It is computed as shown in Eq. 1:

$$DER = \frac{D_w}{D_w + D_c} \tag{1}$$

Where: D_w is the number of incorrectly predicted diacritics; D_c is the number of correctly predicted diacritics. This calculation includes all characters, including punctuation and whitespace.

If a character has multiple diacritics, they are treated as a single composite unit. When DER is computed without CE, the final character of each word is excluded from the error count.

The Word Error Rate (WER) measures the percentage of words that contain at least one incorrectly diacritized character. It is defined as illustrated in Eq. 2:

$$WER = \frac{W_u}{W_u + W_c} \tag{2}$$

Where: W_u is the number of incorrectly diacritized words; W_c is the number of fully correct words. A word is considered incorrect if any of its characters has a wrong or missing diacritic. For WER without CE, two words are considered equal if all characters except the final one have identical diacritics.

3. Results and Discussions

We trained the three models using three different subsets extracted from the Tashkeela corpus. Both the baseline and the CBHG models were trained on a Tesla V100 GPU with a batch size of 64. For the Transformer encoder-decoder model, the same GPU was used but with a smaller batch size of 32, due to its higher computational demand. Table 1 presents the experimental results for all three models.

Table 1. Performance Comparison of Diacritization Models on the Tashkeela Corpus

Model	DER	WER
Baseline	**0.4918**	**0.00010**
BiGRU	**0.2622**	**0.00014**
CBHG	**0.2915**	**0.00017**

Since the CBHG model outperformed the baseline and Transformer models, we performed a random search to optimize its hyperparameters. After tuning, the CBHG model showed a significant improvement in performance: the Diacritization Error Rate (DER) decreased from approximately 0.29 to 0.26, indicating a reduction in diacritic prediction errors. Meanwhile, the Word Error Rate (WER), which was already very low at 0.00017, remained nearly unchanged. This suggests that hyperparameter tuning mainly enhanced the character-level diacritic accuracy without substantially affecting the word-level structure, which was already well preserved. Figure 6 illustrates the training accuracy and loss curves for the CBHG model, demonstrating stable convergence and consistent improvement throughout the training process.

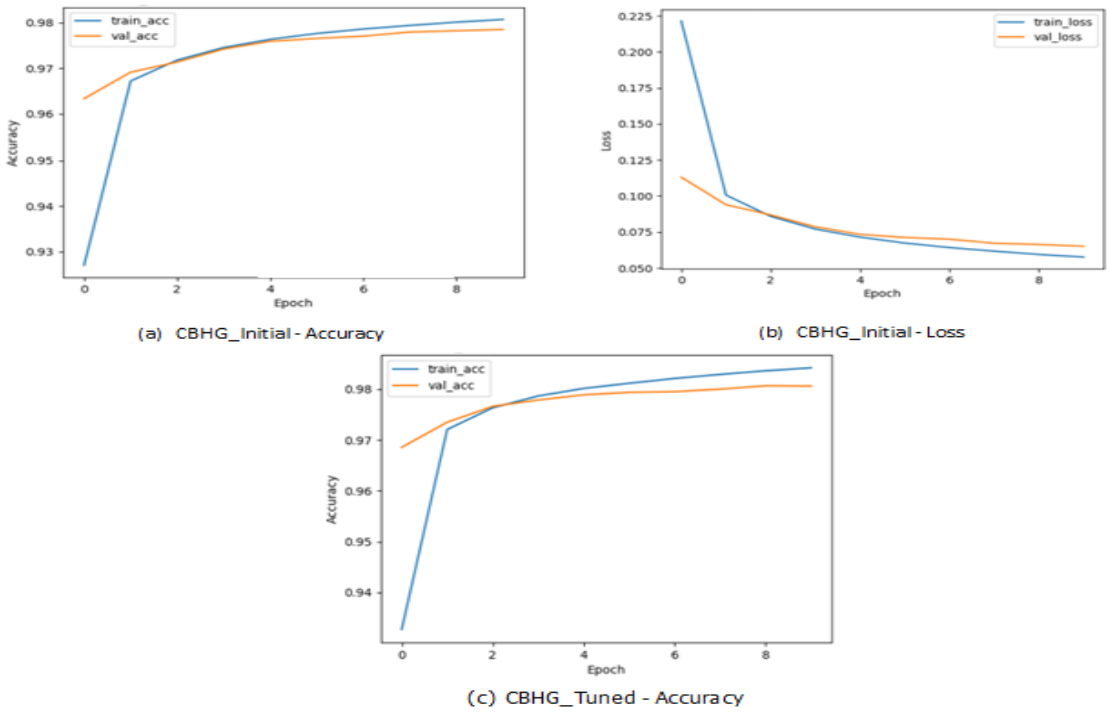

Fig. 6. Learning Progression: Accuracy and Loss over Training Epochs for the CBHG Model.

The experimental results show that the CBHG model slightly outperforms the other architectures in terms of diacritization accuracy. This can be attributed to its effective combination of convolutional layers, highway networks, and bidirectional GRUs, which enable it to capture both local and mid-range linguistic patterns inherent in Arabic's orthographic and morphological structure. Additionally, the CBHG's non-autoregressive design facilitates parallel inference, significantly reducing prediction time—a practical advantage over the Transformer model for real-time applications.

In contrast, the Transformer model, leveraging its self-attention mechanism, excels at modeling long-range dependencies and complex contextual relationships. Despite this strength, in our experiments, the Transformer did not surpass the CBHG's performance, likely due to the absence of pretraining and the limited size of the training corpus. We anticipate that its capabilities could be greatly enhanced through training on larger datasets or fine-tuning with pretrained Arabic language models such as AraBERT or mBART.

The baseline BiLSTM model exhibited the lowest performance, mainly due to its limited capacity to capture long-term dependencies. The BiGRU model improved upon this by replacing LSTM units with more efficient gated recurrent units, resulting in better accuracy with fewer parameters and shorter training times. Qualitative analysis revealed that the Transformer occasionally produced inconsistencies, particularly with short or ambiguous words lacking clear grammatical cues. This is likely because the model's reliance on self-attention may struggle when contextual signals are sparse. Meanwhile, the CBHG's convolutional front-end proved more robust in these cases, effectively capturing local character-level patterns critical in morphologically rich languages like Arabic. Overall, while the CBHG model demonstrates superior accuracy and efficiency in the current experimental setup, the Transformer remains a highly promising approach, especially for future work involving larger corpora, advanced fine-tuning, and domain adaptation.

4. Conclusions

In this study, we investigated several deep learning architectures for automatic Arabic diacritization, including a baseline stacked BiLSTM, a stacked BiGRU, a CBHG-based model, and a Transformer encoder-decoder. Our experiments, conducted on the Tashkeela corpus and evaluated using Diacritization Error Rate (DER) and Word Error Rate (WER), demonstrate that the CBHG model slightly outperforms the Transformer encoder-decoder in terms of diacritic prediction accuracy. Nevertheless, the Transformer architecture remains a promising approach due to its superior ability to capture long-range dependencies through self-attention mechanisms. We anticipate that with larger training datasets, increased model capacity, and more extensive hyperparameter optimization, the Transformer model's performance could surpass that of the CBHG. Future work will explore these directions and investigate the integration of external linguistic knowledge to further enhance Arabic diacritization accuracy.

Author Contributions

Conceptualization, Mohamed CHERRADI and Hajar EL MAHAJER; methodology, Mohamed CHERRADI and Hajar EL MAHAJER; formal analysis, Mohamed CHERRADI and Hajar EL MAHAJER; writing—original draft preparation, Mohamed CHERRADI and Hajar EL MAHAJER; writing—review and editing, Mohamed CHERRADI and Hajar EL MAHAJER; Both authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding

Data Availability Statement

The article does not report any data.

Conflicts of Interest

There are no known competing financial interests or personal relationships that could have influenced the work reported in this article.

References

[1] Guellil, I., Saâdane, H., Azouaou, F., Gueni, B., & Nouvel, D. (2021). Arabic natural language processing : An overview. Journal of King Saud University - Computer and Information Sciences, 33(5), 497-507. https://doi.org/10.1016/j.jksuci.2019.02.006

[2] Fadel, A., Tuffaha, I., Al-Jawarneh, B., & Al-Ayyoub, M. (2019). Arabic Text Diacritization Using Deep Neural Networks. International Conference on Computer Applications & Information Security (ICCAIS), 1-7. https://doi.org/10.1109/CAIS.2019.8769512

[3] Roberts, A., Al-Sulaiti, L., & Atwell, E. (2006). aConCorde : Towards an open-source, extendable concordancer for Arabic. Corpora, 1(1), 39-60. https://doi.org/10.3366/cor.2006.1.1.39

[4] Almanea, M. M. (2021). Automatic Methods and Neural Networks in Arabic Texts Diacritization : A Comprehensive Survey. IEEE Access, 9, 145012-145032. https://doi.org/10.1109/ACCESS.2021.3122977

[5] Alqahtani, S. (2020). Full and Partial Diacritic Restoration : Development and Impact on Downstream Applications.

[6] Cherradi, M., & El Mahajer, H. (2025). Malicious URL detection using machine learning techniques. International Journal of Data Informatics and Intelligent Computing, 4(2), Article 2. https://doi.org/10.59461/ijdiic.v4i2.187

[7] El-Imam, Y. A. (2004). Phonetization of Arabic : Rules and algorithms. Computer Speech & Language, 18(4), 339-373. https://doi.org/10.1016/S0885-2308(03)00035-4

[8] Cherradi, M., & El Haddadi, A. (2024). Comparative Analysis of Machine Learning Algorithms for Sentiment Analysis in Film Reviews. https://doi.org/10.56578/ataiml030301

[9] Hifny, Y. (2021). Recent Advances in Arabic Syntactic Diacritics Restoration. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 7768-7772. https://doi.org/10.1109/ICASSP39728.2021.9414500

[10] Madhfar, M. A. H., & Qamar, A. M. (2021). Effective Deep Learning Models for Automatic Diacritization of Arabic Text. IEEE Access, 9, 273-288. https://doi.org/10.1109/ACCESS.2020.3041676

[11] Alasmary, F., Zaafarani, O., & Ghannam, A. (2024). CATT : Character-based Arabic Tashkeel Transformer. In N. Habash, H. Bouamor, R. Eskander, N. Tomeh, I. Abu Farha, A. Abdelali, S. Touileb, I. Hamed, Y. Onaizan, B. Alhafni, W. Antoun, S. Khalifa, H. Haddad, I. Zitouni, B. AlKhamissi, R. Almatham, & K. Mrini (Éds.), Proceedings of the Second Arabic Natural Language Processing Conference (p. 250-257). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.arabicnlp-1.21

[12] Al Sallab, A., Rashwan, M., M. Raafat, H., & Rafea, A. (2014). Automatic Arabic diacritics restoration based on deep nets. Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP), 65-72. https://doi.org/10.3115/v1/W14-3608

[13] Abandah, G. A., Graves, A., Al-Shagoor, B., Arabiyat, A., Jamour, F., & Al-Taee, M. (2015). Automatic diacritization of Arabic text using recurrent neural networks. Int. J. Doc. Anal. Recognit., 18(2), 183-197.

[14] Belinkov, Y., & Glass, J. (2015). Arabic Diacritization with Recurrent Neural Networks. In L. Màrquez, C. Callison-Burch, & J. Su (Éds.), Conference on Empirical Methods in Natural Language Processing (p. 2281-2285). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1274

[15] Fadel, A., Tuffaha, I., Al-Jawarneh, B., & Al-Ayyoub, M. (2019). Neural Arabic Text Diacritization : State of the Art Results and a Novel Approach for Machine Translation. In T. Nakazawa, C. Ding, R. Dabre, A. Kunchukuttan, N. Doi, Y. Oda, O. Bojar, S. Parida, I. Goto, & H. Mino (Éds.), Proceedings of the 6th Workshop on Asian Translation (p. 215-225). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-5229

[16] Mubarak, H., Abdelali, A., Sajjad, H., Samih, Y., & Darwish, K. (2019). Highly Effective Arabic Diacritization using Sequence to Sequence Modeling. In J. Burstein, C. Doran, & T. Solorio (Éds.), Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers) (p. 2390-2395). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1248

[17] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. International Conference on Neural Information Processing Systems - Volume 1, 1, 577-585.

[18] Bahdanau, D., Cho, K. H., & Bengio, Y. (2015, janvier 1). Neural machine translation by jointly learning to align and translate : 3rd International Conference on Learning Representations, ICLR 2015. Computer Science. Computer Science. http://www.scopus.com/inward/record.url?scp=85062889504&partnerID=8YFLogxK

[19] Al-Thubaity, A., Alkhalifa, A., Almuhareb, A., & Alsanie, W. (2020). Arabic Diacritization Using Bidirectional Long Short-Term Memory Neural Networks With Conditional Random Fields. IEEE Access, 8, 154984-154996. https://doi.org/10.1109/ACCESS.2020.3018885

[20] Lai, T. M., Bui, T., Lipka, N., & Li, S. (2018). Supervised Transfer Learning for Product Information Question Answering. 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 1109-1114. https://doi.org/10.1109/ICMLA.2018.00180

[21] Zhang, Y., Lu, W., Ou, W., Zhang, G., Zhang, X., Cheng, J., & Zhang, W. (2020). Chinese medical question answer selection via hybrid models based on CNN and GRU. Multimedia Tools and Applications, 79. https://doi.org/10.1007/s11042-019-7240-1

[22] Wang, S., & Shi, X. (2021). Research on Correction Method of Spoken Pronunciation Accuracy of AI Virtual English Reading. Advances in Multimedia, 2021, 1-12. https://doi.org/10.1155/2021/6783205

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. ukasz, & Polosukhin, I. (2017). Attention is All you Need. Advances in Neural Information Processing Systems, 30. https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html