

## Article

# Lightweight End-to-End Diacritical Arabic Speech Recognition Using CTC-Transformer with Relative Positional Encoding

Haifa Alaqel \*  and Khalil El Hindi 

Department of Computer Science, College of Computer and Information Science, King Saud University, Riyadh 11451, Saudi Arabia; khindi@ksu.edu.sa

\* Correspondence: haifaaqel@gmail.com

## Abstract

Arabic automatic speech recognition (ASR) faces distinct challenges due to its complex morphology, dialectal variations, and the presence of diacritical marks that strongly influence pronunciation and meaning. This study introduces a lightweight approach for diacritical Arabic ASR that employs a Transformer encoder architecture enhanced with Relative Positional Encoding (RPE) and Connectionist Temporal Classification (CTC) loss, eliminating the need for a conventional decoder. A two-stage training process was applied: initial pretraining on Modern Standard Arabic (MSA), followed by progressive three-phase fine-tuning on diacritical Arabic datasets. The proposed model achieves a WER of 22.01% on the SASSC dataset, improving over traditional systems (best 28.4% WER) while using only  $\approx 14$  M parameters. In comparison, XLSR-Large (300 M parameters) achieves a WER of 12.17% but requires over  $20\times$  more parameters and substantially higher training and inference costs. Although XLSR attains lower error rates, the proposed model is far more practical for resource-constrained environments, offering reduced complexity, faster training, and lower memory usage while maintaining competitive accuracy. These results show that encoder-only Transformers with RPE, combined with CTC training and systematic architectural optimization, can effectively model Arabic phonetic structure while maintaining computational efficiency. This work establishes a new benchmark for resource-efficient diacritical Arabic ASR, making the technology more accessible for real-world deployment.



Academic Editor: Jonathan Blackledge

Received: 19 September 2025

Revised: 15 October 2025

Accepted: 18 October 2025

Published: 21 October 2025

**Citation:** Alaqel, H.; El Hindi, K. Lightweight End-to-End Diacritical Arabic Speech Recognition Using CTC-Transformer with Relative Positional Encoding. *Mathematics* **2025**, *13*, 3352. <https://doi.org/10.3390/math13203352>

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** transformer encoder; relative positional encoding; connectionist temporal classification; modern standard Arabic speech recognition; transfer learning; log mel-spectrogram

**MSC:** 68T10; 68T50; 68T07

## 1. Introduction

Automatic Speech Recognition (ASR) has achieved remarkable progress in recent years for well-resourced languages like English. However, developing high-accuracy speech recognition systems for morphologically complex and under-resourced languages remains a significant challenge. Arabic is one of the world's most widely spoken languages that has these difficulties. Arabic's complex morphology, flexible word order, and dependence on diacritical marks create major obstacles for ASR development. These diacritical marks are challenging, as they are essential for avoiding ambiguity and indicating grammatical relationships [1,2].

The Arabic language is spoken by over 400 million individuals in more than 20 nations [3]. It has three main varieties: Classical Arabic, used in the Quran and Islamic literature; Modern Standard Arabic (MSA), which is the formal written and spoken language throughout the Arab world; and Dialectal Arabic, consisting of various regional dialects used for everyday conversation. The Arabic writing system is largely consonantal, with short vowels generally absent in standard text, which leads to significant ambiguity that native speakers interpret through contextual understanding [4].

MSA refers to Arabic text without diacritical marks, as used in newspapers, books, and digital content. Diacritical Arabic, in contrast, represents the same MSA language but enhanced with explicit diacritical marks that indicate short vowels and other phonetic features [2,5].

The diacritical marks used in the MSA are shown in Table 1 along with their corresponding marks [5]. For instance, the non-diacritical word كُتِبَ in MSA can be read as كَتَبَ (kataba—he wrote), كُتُبَ (kutub—books), and كُتِبَ (kuteba—written). Therefore, diacritics can affect the pronunciation and meaning of Arabic words. In non-diacritical MSA, a single form can have multiple interpretations, which may lead to ambiguity and increase computation when processed by computers. Native Arabic speakers use context to resolve ambiguities, whereas ASR systems depend on diacritics to transcribe speech accurately. Diacritics are important in applications that require high accuracy, such as educational tools, language learning platforms, and voice-controlled systems [6–8].

**Table 1.** Diacritics and their symbols [5].

| Diacritic | Symbol | Diacritic    | Symbol |
|-----------|--------|--------------|--------|
| Fatha     | َ      | Tanween Fath | ً      |
| Kasra     | ِ      | Tanween Kasr | ٍ      |
| Damma     | ُ      | Tanween Damm | ٌ      |
| Sukun     | ◌ْ     | Shadda       | ◌ّ     |

Several factors make diacritical Arabic ASR challenging, including the scarcity of diacritical Arabic ASR datasets, the computational complexity, and the need for strategies that capture both linguistic structures and vowelization patterns efficiently [1,2].

Research efforts in Arabic ASR have traditionally relied on conventional feature extraction techniques such as log-mel spectrograms, combined with statistical models like Hidden Markov Models (HMMs), or more recently deep neural networks [9–13]. In recent years, with the advancements in deep learning, end-to-end (E2E) architectures based on Recurrent Neural Networks [14–23] and Transformers [24] have significantly improved recognition accuracy. Notably, models like XLSR [25] and HuBERT [26] leverage self-supervised pretraining on vast amounts of unlabeled speech data, enabling the extraction of highly rich and contextualized speech representations [5].

The study in [5] pretrained a model, called XLSR, that made a significant advancement in diacritical Arabic ASR. Utilizing transfer learning and data augmentation, XLSR showed remarkable results. Although transfer learning and data augmentation demonstrate that Transformers can be successfully adapted for this purpose, many methods depend on either extensive pretraining datasets or direct E2E training with diacritical data, which may not fully utilize the hierarchical connection between MSA and diacritical Arabic.

Transfer learning has become a key paradigm in modern ASR, especially given its success in enabling models trained on high-resource languages or massive unlabeled corpora to generalize to under-resourced languages with only limited fine-tuning. A recent survey [27] highlights how such techniques help models better generalize by initializing lower layers with rich acoustic-phonetic representations, then adapting higher layers to

language-specific phenomena. Meanwhile, the rise in Transformer-based and large language model (LLM) architectures in ASR has shifted the field toward encoder–decoder frameworks and massive self-supervised pretraining. However, these models often demand heavy computational resources. Our work seeks to bring the power of transfer learning and Transformer efficiency into reach for diacritical Arabic ASR by designing a lightweight, effective architecture that can perform well without the cost of full-scale pretraining [27,28].

This paper presents a lightweight Arabic Automatic Speech Recognition system that predicts diacritical Arabic using a two-stage training approach: first, training a Transformer-based architecture enhanced with RPE [29,30] within a CTC [31] framework on an MSA dataset, followed by a fine-tuning stage that uses a diacritical Arabic dataset. This methodology leverages the foundational knowledge learned from MSA to enhance diacritical prediction performance.

While both this work and XLSR [5] employ a similar two-stage fine-tuning approach (MSA pre-training followed by diacritical Arabic fine-tuning), the key differences lie in computational scale and architecture. XLSR uses a massive 300 M+ parameter model with extensive multilingual pre-training on 53 languages, requiring weeks of training on GPU clusters. Our approach employs a lightweight 14 M parameter transformer encoder with relative positional encoding (RPE) and CTC loss, eliminating the need for a decoder component. This targeted architecture focuses specifically on Arabic without multilingual pre-training, requiring only 15 h of training on a single GPU. The encoder-only design with CTC achieves competitive performance while dramatically reducing computational requirements and improving accessibility for resource-constrained researchers.

Our training approach moves from undiacritized MSA to fully diacritized Arabic that effectively captures both the underlying linguistic structure and the specific vowelization patterns. RPE enhances the model's ability to capture long-range temporal dependencies in speech sequences, while the CTC framework enables training without explicit alignment data.

These limitations highlight an important research gap: Can effective Arabic diacritical ASR systems be built from scratch, without relying on extensive pretraining? Addressing this question is motivated by the realities of resource-limited contexts, initiatives for language preservation, and the need for transparent, lightweight models that can be easily adapted and deployed on devices with limited capabilities.

In summary, the key contributions of this work are:

- **Novel Architectural Design:** Propose an encoder-only Transformer architecture with RPE and CTC designed for diacritical ASR that captures long-range temporal dependencies in speech sequences and enables efficient sequence alignment without explicit alignment data.
- **Efficient Two-Stage Transfer Learning Implementation:** Demonstrate how MSA-to-diacritical transfer learning can be effectively implemented in a lightweight architecture, achieving competitive performance with significantly reduced computational requirements.
- **Efficiency–Performance Balance:** We introduce a lightweight model with only 14 M parameters (vs. 300 M reported in [5]) that achieves a WER of 22.01%. While this is higher than the 12.17% WER reported in [5], it substantially outperforms traditional approaches, highlighting a practical trade-off between efficiency and accuracy.

## 2. Diacritical Arabic ASR

Despite the importance of diacritical Arabic in various applications, research in diacritical Arabic ASR remains underexplored, with only a limited number of studies addressing

this challenging domain. This section provides a comprehensive review of the existing literature, which encompasses most of the research conducted in diacritical Arabic ASR to date. The scarcity of work in this area highlights both the complexity of the problem and the significant research opportunities that remain largely untapped.

Among the few research groups that have ventured into this domain, several have conducted comparative analyses between diacritized and non-diacritized Arabic speech recognition systems. In one of the studies, Abed et al. [32] developed five distinct models using identical corpora with and without diacritical marks, and found that incorporating diacritical information increased the WER by 0.59% to 3.29%. In contrast, Al-Anzi et al. [33,34] presented evidence supporting the beneficial effects of diacritics on recognition performance.

The CMU Sphinx toolkit [35] has been extensively utilized in diacritical Arabic ASR research. Al-Anzi et al. [33] achieved 63.8% accuracy when processing diacritized text, while non-diacritized versions reached 76.4% accuracy. In a follow-up investigation, Al-Anzi et al. [32] observed that although diacritics enhanced phoneme-level recognition capabilities, the overall system accuracy for diacritical processing (69.1%) remained below that of non-diacritical systems (81.2%). This performance gap was attributed to the increased computational complexity introduced by diacritic variability.

Given the limited research landscape, innovative approaches to phonetic representation and linguistic modeling represent particularly valuable contributions to this nascent field. Among the few studies exploring novel methodologies, AbuZeina et al. [36] incorporated sophisticated linguistic frameworks, including parts-of-speech tagging methodologies, to generate contextually appropriate word representations. This strategy resulted in a notable 2.39% reduction in WER when evaluated on Modern Standard Arabic datasets. In another investigation of alternative representations, Boumehdi et al. [37] proposed a novel semi-syllabic representation scheme that integrated diacritical information by treating Sukun and Shadda as distinct phonemic units. Their configuration-based approach demonstrated superior recognition performance, particularly when multiple linguistic representations were combined.

Advanced feature extraction methodologies have been explored by only a handful of studies. Ding et al. [38] implemented MFCC [39] and HMM-based [40] processing techniques on a limited diacritical Arabic vocabulary, achieving impressive results with 7.08% WER and 92.92% recognition accuracy. Among the few researchers pursuing more sophisticated feature extraction approaches, Abed et al. [32] and Alsayadi et al. [41] employed Linear Discriminant Analysis (LDA) [42], Maximum Likelihood Linear Transform (MLLT) [43], and feature-space Maximum Likelihood Linear Regression (fMLLR) [44]. These techniques proved particularly effective for Gaussian Mixture Model (GMM) [45]-based architectures, representing some of the most advanced work in this limited domain.

The exploration of E2E learning paradigms for diacritical Arabic ASR has been extremely limited, with only one notable study pioneering this approach. Alsayadi et al. [41] investigated the combination of MFCC and filter bank (FBANK) features [46] with advanced neural architectures, including CNN-LSTM [47] and CTC-attention [48] models. Their experimental results demonstrated that CNN-LSTM architectures with attention mechanisms surpassed traditional ASR systems by 5.24% and outperformed joint CTC-attention approaches by 2.62%, representing the sole investigation of modern neural approaches in this underexplored domain.

The review of existing studies underscores a critical research gap in diacritical Arabic ASR, indicating the need for further investigation. The complexity inherent in Arabic speech recognition stems from multiple linguistic factors, including rich morphological structures, extensive lexical variation, and constrained availability of diacritized training materials [49,50]. Computational linguistics research focused on Arabic remains limited

compared to other languages [9–12]. The scarcity of studies specifically addressing diacritical Arabic ASR is particularly striking—while substantial advances have been made in general Arabic ASR technology, diacritical Arabic recognition has received minimal attention from the research community.

Given this significant research gap and the promising results of transformer architectures in related domains, recent work has begun exploring transformer models for diacritical Arabic ASR. The emergence of transformer-based approaches represents a paradigm shift from traditional neural architectures toward more sophisticated attention mechanisms capable of capturing long-range dependencies in speech sequences. Initial investigations in this direction leveraged XLSR-based approaches that achieved promising recognition performance, representing the first steps in applying transformer technology to this challenging domain.

The study in [5] introduced two novel transformer-based models employing strategic architectural modifications and transfer learning techniques. The DAASR 1 model implemented direct fine-tuning of the XLSR-53 foundation model on diacritical Arabic data, while DAASR 2 adopted a two-stage transfer learning approach, initially fine-tuning on MSA before adapting to diacritical Arabic recognition tasks. A key innovation in that work involved modifying the XLSR tokenizer to fully support diacritical mark generation, enabling properly diacritized Arabic transcriptions rather than standard MSA outputs. Our comprehensive evaluation on the SASSC dataset demonstrated substantial performance improvements, with DAASR 1 and DAASR 2 achieving WERs of 12.17% and 14%, respectively, representing gains of more than 7% over previous state-of-the-art methods. Furthermore, our investigation of hybrid data augmentation techniques—combining speed adjustment, pitch shifting, and volume modification—yielded additional performance gains, with the parallel hybrid approach achieving an optimal WER of 12.17%. While XLSR-based models achieve high recognition accuracy and set a new performance benchmark, they also reveal the growing need for computationally efficient solutions in transformer-based diacritical Arabic ASR. In this work, we introduce a novel transformer-based approach that integrates relative positional encoding (RPE) and replaces the traditional decoder with connectionist temporal classification (CTC). Our design reduces complexity by using fewer parameters, enables faster training, and requires less memory. As a result, the proposed model is more practical for resource-constrained environments while maintaining competitive performance.

We propose a lightweight transformer for diacritical ASR that provides a good balance between accuracy and computational cost. In this study, computational cost is measured in terms of training time and the number of GPUs used, without considering detailed hardware-level metrics such as memory usage or floating-point operations per second. Additionally, inference efficiency is reported separately in terms of latency and real-time factor. This represents an important advancement in making transformer-based diacritical Arabic ASR obtainable where computational resources are limited.

### 3. Proposed Speech Transformer

#### 3.1. Overall Design

The proposed architecture for Arabic ASR with diacritics is illustrated in Figure 1. The model follows an encoder-only Transformer architecture that processes log mel-spectrograms and outputs character-level predictions through CTC. A linear projection layer maps the log mel-spectrograms to the model's hidden dimension, followed by layer normalization and dropout for regularization. RPE replaces absolute positional encoding to better capture long-range dependencies in speech sequences and replaces the traditional Transformer decoder with a CTC head for efficient sequence-to-sequence alignment.

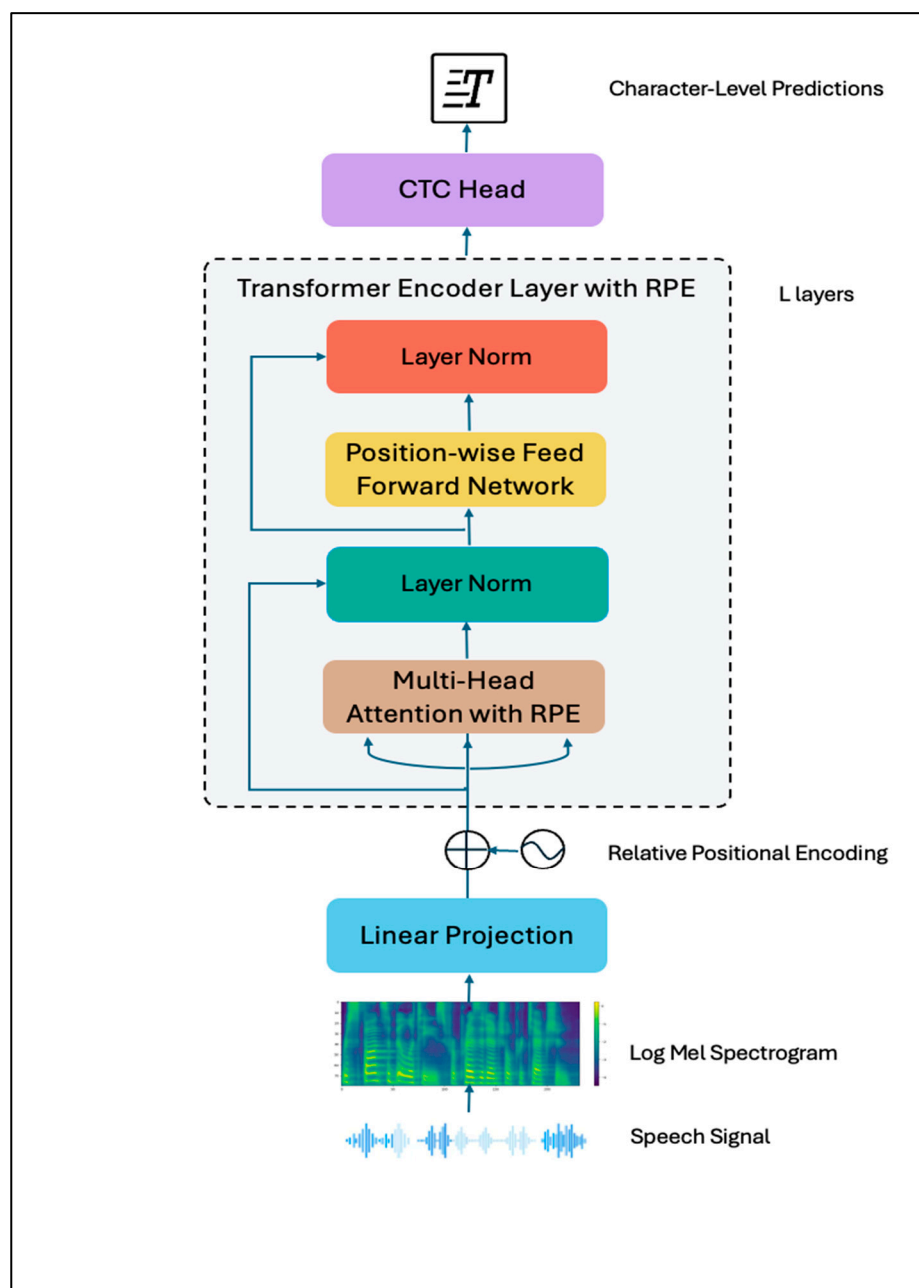


**Figure 1.** Overall architecture of the proposed lightweight Transformer-based Arabic ASR model. The encoder-only design integrates relative positional encoding (RPE) and a CTC head to generate character-level predictions directly from log-Mel spectrogram inputs.

The Transformer Encoder block is composed of two main sub-layers: multi-head attention (MHA) layer and feedforward layer connected via residual connections. RPE replaces the absolute positional embedding (APE) to capture long dependencies. Normalization and dropout are added for regularization to avoid overfitting.

### 3.2. Input Processing and Feature Projection

The input log mel-spectrogram  $X \in \mathbb{R}^{(T \times 80)}$  are first projected through a linear transformation to the hidden dimension of the model:

$$H_0 = XW^{proj} + b^{proj} \quad (1)$$

where  $W^{proj} \in \mathbb{R}^{(80 \times d_{model})}$  and  $H_0 \in \mathbb{R}^{(T \times d_{model})}$ .

Layer normalization and dropout ( $p = 0.2$ ) are applied for regularization:

$$H_0 = \text{Dropout}(\text{LayerNorm}(H_0)) \quad (2)$$

A slightly higher dropout rate ( $p = 0.2$ ) is used only at the input projection layer to mitigate overfitting on raw acoustic features, which typically contain more noise variability.

Within the Transformer encoder blocks, a lower dropout ( $p = 0.1$ ) is applied, as confirmed by the ablation study, since moderate regularization preserves representational capacity while ensuring stable convergence.

### 3.3. Encoder Layer Structure

The encoder architecture consists of  $L$  stacked transformer layers, each containing the key components detailed below. The encoder processes log mel-spectrogram inputs through a linear projection layer, followed by  $L$  identical encoder transformer blocks that each comprise MHA with relative positional encoding, position-wise feedforward networks, with residual connections, layer normalization applied to each sub-layer and dropout for regularization.

#### 3.3.1. Relative Positional Encoding (RPE)

The APE used by conventional Transformer architectures determines fixed position embeddings according to the precise location of the element in the sequence. RPE, however, works better for ASR tasks, particularly in Arabic, because diacritical Arabic is dependent on contextual relationships rather than absolute placement. RPE captures the relative distance between elements and allows the model to understand temporal relationships regardless of absolute sequence position [27,28].

The standard self-attention mechanism computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{D_K}}\right)V \quad (3)$$

RPE modifies this by incorporating relative position bias:

$$e_{ij}^h = (q_i^h)(k_j^h)^T + (q_i^h)(r_{ij}^k)^T \quad (4)$$

where  $e_{ij}^h$ : attention logit between positions  $i$  and  $j$  for head  $h$ ,  $(q_i^h)$  is a query vector for head  $h$ ,  $(k_j^h)$  is a key vector for head  $h$  and  $r_{ij}^k$  is a learnable relative position embedding for keys, representing the distance between positions  $i$  and  $j$ .

#### 3.3.2. Multi-Head Self-Attention (MHA) with RPE

MHA mechanism forms the core of our encoder, enhanced with RPE to improve temporal modeling. Each attention head computes attention weights between all pairs of positions in the input sequence, modified by relative position bias terms. Building upon the RPE formulation from Section 3.3.1, MHA mechanism computes multiple attention heads in parallel, using the same relative position attention for each head:

Each encoder layer implements MHA enhanced with RPE:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}^1, \text{head}^2, \dots, \text{head}_H)W^O \quad (5)$$

where

$$\text{head}_h = \text{softmax}\left(\frac{e_{ij}^h}{\sqrt{d_k}}\right)V_h \quad (6)$$

### 3.3.3. Feedforward Network

Each encoder layer includes a position-wise feedforward network that consists of two linear transformations with Swish activation [51]. The feedforward network expands the representation to a higher dimension ( $6\times$  the model dimension based on our ablation study), applies non-linear transformation, and projects back to the original dimension:

$$FFN(x) = \text{Swish}(xW_1 + b_1)W_2 + b_2 \quad (7)$$

where  $\text{Swish}(x) = x \cdot \text{sigmoid}(x)$ ,  $W_1 \in \mathbb{R}^{d_{\text{model}} \times 6d_{\text{model}}}$  and  $W_2 \in \mathbb{R}^{6d_{\text{model}} \times d_{\text{model}}}$

Based on our ablation study (Section 4.2), both the  $6\times$  expansion ratio and Swish activation consistently outperformed alternative configurations for diacritical Arabic speech recognition. This component provides the model with the capacity to learn complex non-linear mappings between acoustic features and linguistic representations. The larger expansion ratio ( $6\times$  vs. typical  $4\times$ ) and smooth, differentiable nature of Swish activation prove particularly effective for Arabic ASR, enabling better gradient flow when learning the intricate relationships between spectral features and diacritical patterns.

### 3.3.4. Residual Connection and Layer Normalization

Each sub-layer (attention and feedforward) employs residual connections with post-layer normalization:

$$H_l = \text{LayerNorm}(H_{l-1}) + \text{SubLayer}(H_{l-1}) \quad (8)$$

Each sub-layer is followed by layer normalization. For diacritical Arabic speech recognition, this post-norm setup performs better than the more popular pre-norm technique, according to our ablation study (Section 4.2). Layer normalization is more suited for sequence modeling jobs where batch sizes may change, as it calculates statistics across the feature dimension for each individual sample, unlike batch normalization.

In the context of Arabic speech recognition, the post-norm configuration helps maintain stable activations across different speakers, dialects, and acoustic conditions while providing superior gradient flow during training. This is useful for diacritical Arabic ASR, where the model learns to distinguish phonetic variations that indicate different vowelization patterns.

Finally, the complete encoder layer  $l$  processes input  $H_{l-1}$  as:

$$H'_l = H_{l-1} + \text{Dropout}(\text{MultiHeadAttention}_{\text{RPE}}(H_{l-1})). \quad (9)$$

$$H'_l = \text{LayerNorm}(H'_l) \quad (10)$$

$$H_l = H'_l + \text{Dropout}(FFN(H'_l)) \quad (11)$$

$$H_l = \text{LayerNorm}(H_l) \quad (12)$$

## 3.4. Connectionist Temporal Classification (CTC) Head and Output Layer

CTC [29] head is used to align the sequence of spectrograms and their corresponding character sequences instead of a decoder. The CTC head consists of a linear projection, which translates the hidden representations from the encoder into a vocabulary-sized output that includes all Arabic characters and their variations, diacritical marks, and a special blank symbol. Without the need for explicit alignment between audio frames and characters, CTC enables the model to predict character sequences of any length from input sequences. Since it does not require frame-level alignment but only sentence-level transcriptions, this is beneficial for Arabic ASR where precisely aligned annotated data is lim-

ited. The model can automatically learn the best character-to-frame mappings since the CTC loss function marginalizes over all potential alignments between the input and target sequences. During inference, the most likely character sequence from the CTC output probabilities is decoded using a greedy decoding method. After decoding, a post-processing step ensures valid Arabic orthography. Each predicted diacritic is appended to the immediately preceding base character, maintaining proper character–diacritic alignment. Consecutive diacritics without an intervening base letter are filtered to retain only the last valid one, and any isolated or non-Arabic symbols are removed. This step guarantees that the final output text follows correct Arabic writing conventions and avoids invalid sequences.

The CTC head converts encoder outputs to character-level predictions as:

$$P = \text{softmax}(H_L W^{\text{out}} + b^{\text{out}}) \quad (13)$$

where  $W_{\text{out}} \in \mathbb{R}^{(d_{\text{model}} \times |V| + 1)}$ ,  $|V| = 48$  (vocabulary size), and +1 accounts for the CTC blank token.

Vocabulary compositions: Arabic letters: 28 basic letters + 8 variations (أ، إ، آ، ؤ، ع، هـ، و، ز)، representing phonemic and orthographic variants rather than positional letter shapes; diacritical marks: 8 marks and special tokens: <blank>, <pad>, <unk>, <space>.

CTC loss: The CTC loss marginalizes over all valid alignments  $\pi$  between input sequence  $X$  and target sequence  $Y$ :

$$L_{\text{CTC}} = -\log \sum_{\pi \in \phi(Y)} \prod_{t=1}^T P(\pi_t | X) \quad (14)$$

where  $\phi(Y)$  represents all valid CTC paths that decode to  $Y$ ,  $\pi$  is alignment path (one possible character sequence),  $t$ : Time step index and  $T$  is the total number of time steps (total input sequence length).

### 3.5. Training Hyperparameters and Implementation Details

All key hyperparameters and preprocessing settings were defined explicitly in the training script:

- Optimizer: AdamW with  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ,  $\epsilon = 1 \times 10^{-8}$
- Weight decay: 0.01
- Gradient clipping: applied at max norm = 1.0
- Learning-rate schedule: cosine annealing with  $T_{\text{max}} = \text{epochs per phase}$  and Initial LR:  $1 \times 10^{-3}$  (Phase 1),  $3 \times 10^{-4}$  (Phase 2),  $1 \times 10^{-4}$  (Phase 3)
- Batch size: 6
- Sampling rate: 16 kHz
- Spectrogram parameters: 80 Mel bins, FFT = 512, window length = 400 (25 ms), hop length = 160 (10 ms),  $f_{\text{min}} = 0$ ,  $f_{\text{max}} = 8$  kHz

### 3.6. Transfer Learning

Transfer learning is a core technique in machine learning that allows models to leverage knowledge gained from large-scale pretraining to improve performance on tasks with limited data. It is significant for low-resource languages and specialized domains, where collecting huge training data is challenging or expensive. Transfer learning means that the knowledge gained from one task is used for similar tasks effectively. In our task, transfer learning refers to pretraining the model on a large MSA dataset of unlabeled audio and then fine-tuning it on a diacritical Arabic dataset. Thus, low-level acoustic features and patterns learned during pretraining phase are transferred across different languages, speakers, and domains [52].

The effectiveness of transfer learning in ASR stems from the hierarchical nature of speech representations. Lower layers of neural networks learn general acoustic features such as spectral patterns, while higher layers capture more linguistic information. This hierarchical organization enables effective knowledge transfer, where pre-trained lower layers provide a strong foundation for learning task-specific upper layers. Transfer learning has proven especially valuable for cross-lingual speech recognition, where models trained on high-resource languages are adapted for low-resource ones. Research has shown that acoustic-phonetic similarities between languages influence transfer effectiveness, with closely related languages benefiting more than distant language pairs [52,53].

#### 4. Ablation Study

We conducted a comprehensive ablation study using a CommonVoice 6.1 Arabic dataset [54] to systematically assess the effects of various architectural decisions and hyperparameters on the performance of our Transformer-based Arabic ASR system. The experiments required more than 110 GPU hours of training time because they included 30 different tests spread over seven progressive phases. Each phase built upon the best configuration from the previous phase, ensuring that improvements are cumulative and interactions between architectural components are properly evaluated.

Our progressive methodology began with a baseline configuration of an 8-layer, 6-head, 192-model dimensional architecture using ReLU activation, Post-LN normalization, a 0.1 dropout rate, a  $4\times$  feedforward expansion ratio, and a  $1.5 \times 10^{-4}$  learning rate. For efficient exploration, we applied 40 epochs for initial evaluation, with the best-performing model from each phase then fully trained the final model for 200 epochs with early stopping patience of 20 for the ablation study only. All architectural configurations and hyperparameter variations examined during the seven ablation study phases are reported in Table 2, and the results are comprehensively visualized in Figure 2.

**Table 2.** Complete evaluation of all seven phases. Bold is for the best result in each phase.

| Phase | Architecture |            | Dimension  | Activation   | Regularization |            | FF Ratio                    | LR                                     | Parameters     | Training Time (h) | WER           |
|-------|--------------|------------|------------|--------------|----------------|------------|-----------------------------|----------------------------------------|----------------|-------------------|---------------|
|       | # of Layers  | # of Heads |            |              | Normalization  | Dropout    |                             |                                        |                |                   |               |
| 1     | 4            | 4          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 02.20 M        | 4.0               | 92.77%        |
|       | 4            | 6          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 02.20 M        | 2.6               | 94.07%        |
|       | 4            | 8          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 02.20 M        | 3.1               | 92.58%        |
|       | 6            | 4          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 03.09 M        | 3.3               | 89.56%        |
|       | 6            | 6          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 03.09 M        | 4.5               | 87.31%        |
|       | 6            | 8          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 03.09 M        | 5.7               | 89.35%        |
|       | 8            | 4          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 03.98 M        | 4.5               | 90.88%        |
|       | <b>8</b>     | <b>6</b>   | <b>192</b> | <b>ReLU</b>  | <b>Post-LN</b> | <b>0.1</b> | <b><math>4\times</math></b> | <b><math>1.5 \times 10^{-4}</math></b> | <b>03.98 M</b> | <b>5.5</b>        | <b>86.67%</b> |
|       | 8            | 8          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 03.98 M        | 6.1               | 87.52%        |
|       | 10           | 4          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 04.87 M        | 5.8               | 99.78%        |
|       | 10           | 6          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 04.87 M        | 6.9               | 100.00%       |
|       | 10           | 8          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 04.87 M        | 7.6               | 100.00%       |
| 2     | 8            | 6          | 132        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 01.98 M        | 4.7               | 99.53%        |
|       | 8            | 6          | 192        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 03.98 M        | 5.5               | 86.67%        |
|       | 8            | 6          | 252        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 06.68 M        | 6.6               | 90.62%        |
|       | <b>8</b>     | <b>6</b>   | <b>318</b> | <b>ReLU</b>  | <b>Post-LN</b> | <b>0.1</b> | <b><math>4\times</math></b> | <b><math>1.5 \times 10^{-4}</math></b> | <b>10.44 M</b> | <b>9.4</b>        | <b>81.02%</b> |
| 3     | 8            | 6          | 318        | ReLU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 10.44 M        | 9.4               | 81.02%        |
|       | <b>8</b>     | <b>6</b>   | <b>318</b> | <b>Swish</b> | <b>Post-LN</b> | <b>0.1</b> | <b><math>4\times</math></b> | <b><math>1.5 \times 10^{-4}</math></b> | <b>10.44 M</b> | <b>9.5</b>        | <b>79.78%</b> |
|       | 8            | 6          | 318        | GELU         | Post-LN        | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 10.44 M        | 3.4               | 100.00%       |
| 4     | 8            | 6          | 318        | Swish        | Pre-LN         | 0.1        | $4\times$                   | $1.5 \times 10^{-4}$                   | 10.44 M        | 9.4               | 88.43%        |
|       | <b>8</b>     | <b>6</b>   | <b>318</b> | <b>Swish</b> | <b>Post-LN</b> | <b>0.1</b> | <b><math>4\times</math></b> | <b><math>1.5 \times 10^{-4}</math></b> | <b>10.44 M</b> | <b>9.5</b>        | <b>79.78%</b> |

Table 2. Cont.

| Phase | Architecture |            | Dimension | Activation | Regularization |         | FF Ratio | LR                   | Parameters | Training Time (h) | WER     |
|-------|--------------|------------|-----------|------------|----------------|---------|----------|----------------------|------------|-------------------|---------|
|       | # of Layers  | # of Heads |           |            | Normalization  | Dropout |          |                      |            |                   |         |
| 5     | 8            | 6          | 318       | Swish      | Post-LN        | 0.05    | 4×       | $1.5 \times 10^{-4}$ | 10.44 M    | 4.7               | 100.00% |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 4×       | $1.5 \times 10^{-4}$ | 10.44 M    | 9.5               | 79.78%  |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.15    | 4×       | $1.5 \times 10^{-4}$ | 10.44 M    | 9.5               | 99.76%  |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.25    | 4×       | $1.5 \times 10^{-4}$ | 10.44 M    | 9.5               | 83.74%  |
| 6     | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 2×       | $1.5 \times 10^{-4}$ | 07.20 M    | 4.5               | 100.00% |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 3×       | $1.5 \times 10^{-4}$ | 08.82 M    | 9.3               | 79.64%  |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 4×       | $1.5 \times 10^{-4}$ | 10.44 M    | 9.4               | 81.02%  |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 6×       | $1.5 \times 10^{-4}$ | 13.68 M    | 10.2              | 79.10%  |
| 7     | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 6×       | $1.4 \times 10^{-4}$ | 13.68 M    | 9.5               | 73.92%  |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 6×       | $1.5 \times 10^{-4}$ | 13.68 M    | 10.2              | 79.11%  |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 6×       | $2.9 \times 10^{-4}$ | 13.68 M    | 4.4               | 100.00% |
|       | 8            | 6          | 318       | Swish      | Post-LN        | 0.1     | 6×       | $4.9 \times 10^{-4}$ | 13.68 M    | 8.8               | 100.00% |

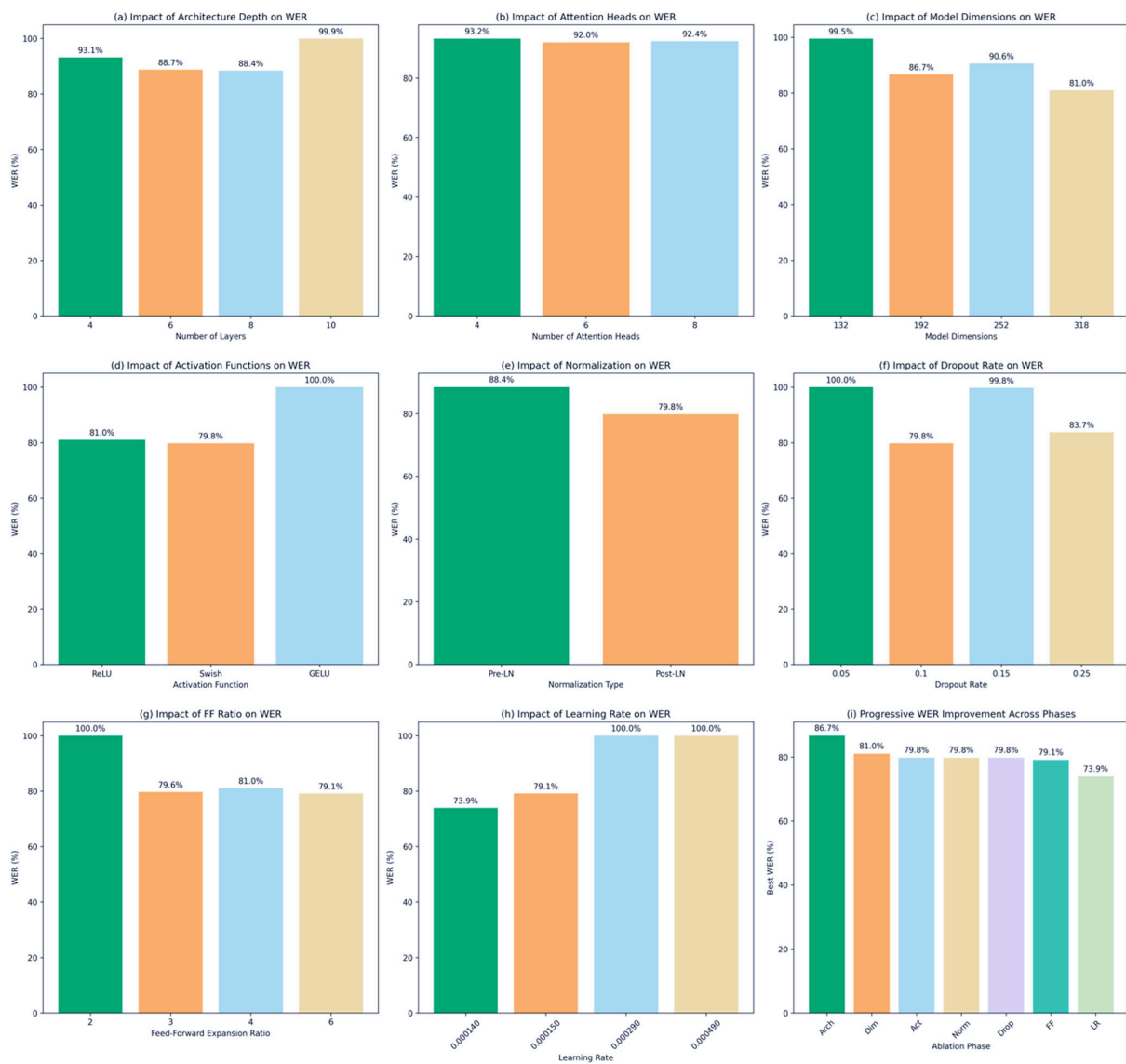


**Figure 2.** Comprehensive ablation study showing systematic optimization of transformer components for Arabic ASR. (a) Architecture depth analysis, (b) attention head optimization, (c) model dimension scaling, (d) activation function comparison, (e) layer normalization strategy, (f) dropout rate optimization, (g) feed-forward ratio analysis, (h) learning rate optimization, and (i) progressive improvement summary.

#### 4.1. Architecture Ablation

##### Phase 1: Core Architecture Scaling

In Phase 1, we systematically investigated the transformer architecture by varying the number of layers (L) and attention heads (H), while keeping the model dimension fixed at 192. This phase included evaluation of 4-, 6-, and 8-layer architectures with attention head counts of 4, 6, and 8.

**Depth vs. Width Analysis:** Our experiments revealed that deeper architectures with moderate width provided optimal performance for ASR, with clear limits to depth scaling (Figure 2a,b). The optimal balance between model depth and attention width was achieved with the 8L6H configuration, yielding a WER of 86.67%. While 10L6H192D and 10L8H192D both completely failed with 100.00% WER, 10L4H192D achieved 99.78% WER, indicating considerable degradation in deeper designs (Figure 2a). For later stages, this serves as the baseline setup.

**Parameter Efficiency:** We analyzed the relationship between parameter count and performance across all architectural variants. The findings demonstrate distinct failure points and diminishing rewards as depth increases. With a reasonable training period of 5.5 h and optimal performance with 3.98 M parameters, 8L6H192D was the most parameter-efficient configuration. However, despite the higher computational cost, deeper 10 L configurations with 4.87 M parameters showed dramatic performance reduction.

**Scaling Patterns:** We observed that deeper architectures failed to converge effectively. The 10L4H192D configuration achieved a WER of 99.78% with 4.87 M parameters and 5.8 h training time, while 10L6H192D and 10L8H192D both failed completely with WER scores of 100.00% (6.9 and 7.6 h training time, respectively). This implies that the 8L6H layout offers the best architectural balance, as shown in Figure 2a,b.

##### Phase 2: Model Dimension Scaling

In Phase 2, we explored the effect of model dimensionality by testing values of 132, 252, and 318, chosen to align with the optimal head count. The results indicate that higher dimensionality improves performance (Figure 2c). The optimal model dimension of 318 achieved a WER of 81.03%, with a 6.51% improvement over the baseline 192-dimensional model. While the 132 D performed poorly with a WER of 99.54%, 252 D showed moderate performance with a WER of 90.63%. This indicates that adequate model capacity is crucial to capture the complexity of Arabic speech.

##### Phase 3: Activation Function Analysis

We compared three activation functions within the feed-forward layers: ReLU (baseline), GELU, and Swish. With a WER of 79.78%, the experimental results indicate that Swish activation performed the best, followed by ReLU baseline performance (Figure 2d). GELU activation failed to converge (WER: 100.00%). The improved performance of Swish can be attributed to its smooth functional form, which facilitates more effective optimization in transformer-based Arabic ASR. The unstable convergence of GELU was likely caused by gradient fluctuations under limited data, as its input-dependent curvature can amplify small training noise. Additionally, the deeper model variants combined with GELU tended to overfit quickly, suggesting that the model capacity exceeded what the low-resource dataset could effectively support.

##### Phase 4: Normalization Strategy

We evaluated the impact of normalization placement by comparing Post-LayerNorm (baseline) with Pre-LayerNorm configurations. The results showed that Post-LayerNorm remained optimal with a WER of 79.78% compared to 88.44% for Pre-LayerNorm, showing that Pre-LayerNorm significantly degraded performance (Figure 2e).

#### 4.2. Hyperparameter Ablation

##### Phase 5: Regularization Analysis

To find the ideal regularization strength, we systematically changed the dropout rate between 0.05, 0.10 (baseline), 0.15, and 0.25.

The baseline dropout rate remained appropriate, according to the experimental results (Figure 2f). A dropout rate of 0.1 achieved the best performance of a WER of 79.78%, while higher rates of 0.15 achieved a WER of 99.77% and 0.25 achieved a WER of 83.75%, leading to performance degradation. A 0.05 dropout rate resulted in complete training failure. This implies that moderate regularization at 0.1 prevents overfitting while preserving model capacity.

##### Phase 6: Feed-Forward Dimension Ratio

We investigated the impact of feed-forward layer dimensionality by testing ratios of  $2\times$ ,  $3\times$ ,  $4\times$  (baseline), and  $6\times$  relative to the model dimension.

Our findings indicate that higher feed-forward capacity improves performance (Figure 2g). The  $6\times$  ratio achieved the best performance with a WER of 79.11% with a 0.84% improvement over the  $4\times$  baseline at the cost of an additional 3.24 M parameters with a total of 13.68 M. The  $2\times$  ratio failed to converge, while  $3\times$  showed competitive performance with a WER of 79.64%. These findings underscore the importance of sufficient feed-forward capacity for handling task complexity.

##### Phase 7: Optimization Strategy

We explored different learning rates ( $1.4 \times 10^{-4}$ ,  $1.5 \times 10^{-4}$  baseline,  $2.9 \times 10^{-4}$ ,  $4.9 \times 10^{-4}$ ) to optimize the training dynamics.

The results show that fine-tuning the learning rate provided substantial final improvements (Figure 2h). A learning rate of  $1.4 \times 10^{-4}$  achieved the best convergence, reaching a WER of 73.92%, representing a 6.56% relative improvement over the previous best. The higher learning rates ( $2.9 \times 10^{-4}$ ,  $4.9 \times 10^{-4}$ ) led to training instability and convergence failure, while the baseline  $1.5 \times 10^{-4}$  was better.

#### 4.3. Overall Results

Our progressive ablation study systematically optimized each component of the initial baseline configuration (Figure 2i). We started from 8L6H192D + ReLU + Post-LN + Dropout = 0.1 + FF =  $4\times$  + LR =  $1.5 \times 10^{-4}$  (WER: 86.67) and achieved the final optimized configuration of 8L6H318D + Swish + Post-LN + Dropout = 0.1 + FF =  $6\times$  + LR =  $1.4 \times 10^{-4}$  (WER: 73.92) that represents a substantial 14.71% relative improvement.

The systematic optimization revealed the relative contribution of each component: the largest improvement (6.51%) came from model dimension scaling, followed by learning rate adjustment (6.56%), activation function optimization (1.54%) and feedforward expansion (0.84%). Normalization and regularization settings remained optimal at their baseline values (Figure 2a–h). Table 3 summarizes the progressive optimization across all phases and presents the cumulative improvements achieved through systematic hyperparameter tuning.

The study reveals several key insights: moderate depth with sufficient attention heads (8 layers, 6 heads) provides the best performance (Figure 2a,b). Expanding the model dimensionality to 318 improved performances despite higher computational costs, see Figure 2c. The Swish activation outperforms traditional ReLU (Figure 2d), Post-LN normalization and moderate dropout (0.1) are optimal for regularization (Figure 2e,f), expanded feedforward capacity ( $6\times$ ) further enhances performance (Figure 2g), and careful learning rate tuning provides substantial final improvements (Figure 2h). Figure 2i summarizes the progressive improvement.

**Table 3.** Summary of the progressive optimization across all phases.

| Phase | Architecture | Activation | Normalization | Dropout | FF Ratio | Learning Rate        | WER    | Improvement |
|-------|--------------|------------|---------------|---------|----------|----------------------|--------|-------------|
| 1     | 8L6H192D     | ReLU       | Post-LN       | 0.1     | 4×       | $1.5 \times 10^{-4}$ | 86.67  | -           |
| 2     | 8L6H318D     | ReLU       | Post-LN       | 0.1     | 4×       | $1.5 \times 10^{-4}$ | 81.02% | 6.51%       |
| 3     | 8L6H318D     | Swish      | Post-LN       | 0.1     | 4×       | $1.5 \times 10^{-4}$ | 79.78% | 1.54%       |
| 4     | 8L6H318D     | Swish      | Post-LN       | 0.1     | 4×       | $1.5 \times 10^{-4}$ | 79.78% | 0.00%       |
| 5     | 8L6H318D     | Swish      | Post-LN       | 0.1     | 4×       | $1.5 \times 10^{-4}$ | 79.78% | 0.00%       |
| 6     | 8L6H318D     | Swish      | Post-LN       | 0.1     | 6×       | $1.5 \times 10^{-4}$ | 79.11% | 0.84%       |
| 7     | 8L6H318D     | Swish      | Post-LN       | 0.1     | 6×       | $1.4 \times 10^{-4}$ | 73.92% | 6.56%       |

**Computational Efficiency Analysis:** We analyzed the trade-offs between model performance and computational requirements across all experimental configurations. The parameter efficiency frontier shows that the final optimized model achieved the best performance-efficiency balance. The architecture (8L6H with 13.68 M parameters in the final configuration) achieved superior performance while requiring reasonable computational resources.

Training time analysis reveals that the best configuration required 10.2 h of training time and achieved convergence with early stopping patience of 10 epochs. Larger dimensionalities showed good returns in performance per additional training hour, which demonstrated practical feasibility for this task and dataset size.

#### Interpretation of the Configuration

The final configuration (8 layers, 6 heads, 318 model dimensions, and 6× feed-forward ratio) achieved superior accuracy since it effectively models the acoustic and linguistic structure of Arabic. The moderate depth and number of heads (8L6H) give enough context without causing overfitting or instability. The larger model dimension (318) helps the network capture small sound differences such as short vowels and consonant emphasis that define diacritics. The 6× feed-forward ratio improves how the model learns short, detailed sound patterns. These settings work together to find a balance between expressiveness, stability, and efficiency. The result is a lightweight Transformer that is suitable for diacritical Arabic ASR.

## 5. Training and Results

This section presents the training methodology and experimental evaluation of our proposed diacritical Arabic ASR system. We employ a two-stage training strategy that leverages the linguistic relationship between MSA and diacritical Arabic, followed by the specific optimization configuration and architectural advantages that make our approach particularly effective for this challenging task.

The evaluation involved extensive 10-fold cross-validation experiments. In this setup, the data were partitioned differently in each fold, with separate training, validation, and test splits to ensure a robust assessment of model stability and generalization. No fixed independent test set was used; instead, testing was performed within each fold to obtain an averaged performance across all folds. Three distinct training approaches were compared: direct training on diacritical Arabic datasets, traditional full fine-tuning, and three-phase progressive fine-tuning. In addition, the effectiveness of various data augmentation techniques was examined, and the results were compared with state-of-the-art diacritical Arabic ASR systems. The chapter concludes with an analysis of computational efficiency, practical deployment implications, and future research directions.

Our findings demonstrate that the three-phase progressive fine-tuning approach combined with data augmentation achieves the best Word Error Rate (WER) of 22.01% on the Diacritical Arabic dataset. This represents a substantial improvement over traditional

training methods while maintaining computational efficiency and practical viability for real-world deployment scenarios.

### 5.1. The Training Data

The limited availability of annotated datasets is one of the main obstacles to developing diacritical Arabic ASR systems. To mitigate this limitation, we employed transfer learning to enhance recognition performance. Specifically, we first trained the model on non-diacritical datasets and then fine-tuned it on a diacritical Arabic dataset. This approach helped compensate for the scarcity of annotated diacritical data.

- Non-Diacritical Dataset: Common Voice 6.1 [54] is an MSA corpus containing 128 h of speech.
- Diacritical Dataset: SASSC [55] is a diacritical Arabic single-speaker corpus comprising 51,432 discretized words and more than seven hours of audio recordings.

### Dataset Splits and Preprocessing

For Common Voice 6.1, an 80%/20% random split was applied for training–validation and testing. Because the corpus is multi-speaker, speaker overlap may occur between subsets. However, this dataset was used primarily for pretraining, so overlap does not affect the final evaluation.

For the SASSC corpus, which is a single-speaker dataset, the data were divided into 80% training–validation and 20% testing in each fold, ensuring that different test splits were used across folds to evaluate the model’s robustness and generalization performance.

After splitting:

- Transcripts were normalized by removing special characters and non-Arabic symbols while retaining diacritical marks.
- Audio recordings were used without additional preprocessing (e.g., silence trimming, noise reduction, or amplitude normalization) to preserve natural acoustic variability.

### 5.2. Training Strategy

We employed a Two-Stage Training Process:

1. Stage 1—MSA Pre-training: The model was first trained on a Modern Standard Arabic dataset that does not contain diacritics, CommonVoice 6.1.
2. Stage 2—Diacritical Fine-tuning: The pretrained model was then fine-tuned on the diacritical Arabic dataset (SASSC). Two different fine-tuning strategies were applied:
  - Full Fine-Tuning: the model obtained from Stage 1 was fully trained on the diacritical Arabic dataset.
  - Three-phase fine-tuning: To avoid catastrophic forgetting during adaptation, we implemented a gradual three-phase fine-tuning process with systematic parameter unfreezing. This approach preserved acoustic-phonetic representations while enabling vocabulary expansion. In the first phase (10 epochs), only the CTC head was trained, while all encoder layers were frozen using a learning rate of  $1 \times 10^{-3}$ . This allowed the CTC head to learn diacritic mappings without disrupting the established feature extraction process. In the second phase (10 epochs), the final two encoder layers and final normalization layer were unfrozen and trained with a learning rate of  $3 \times 10^{-4}$ . This strategy preserved audio processing in initial layers while allowing the higher layers to capture linguistic patterns related to diacritics. The final phase involved full model fine-tuning with 100 epochs with a learning rate of  $1.5 \times 10^{-4}$  and early stopping using a patience of 10 epochs.

This approach leverages the linguistic relationship between MSA and diacritical Arabic, enabling the model to learn first basic Arabic phonetic patterns before specializing in diacritical recognition.

The number of epochs and learning rates for the three-phase fine-tuning were determined empirically through preliminary validation experiments. A small portion (10%) of the SASSC training data was reserved as a validation set to monitor convergence speed and generalization stability during pilot runs. We found that 10 epochs with a learning rate of  $1 \times 10^{-3}$  allowed the CTC head to adapt rapidly without overfitting, while another 10 epochs with  $3 \times 10^{-4}$  enabled stable adaptation of the upper encoder layers. The final phase with 100 epochs and  $1.5 \times 10^{-4}$ , combined with early stopping (patience = 10 epochs), consistently produced the lowest validation loss and best WER across folds. These hyperparameters therefore represent an optimal balance between convergence rate, stability, and computational efficiency for the diacritical fine-tuning process.

### 5.3. Model Advantages

This architecture offers several advantages that make it well-suited for diacritical Arabic ASR. The incorporation of CTC provides flexibility, as it allows the system to handle alignments implicitly during the training [29,56]. Furthermore, the use of RPE improves the model's capacity to capture contextual relationships, as it generalizes to sequences of unseen lengths and facilitates the acquisition of position-relative patterns [27,28].

The two-stage training methodology enhances model robustness and generalization capabilities by leveraging the hierarchical relationship between MSA and diacritical Arabic. Fine-tuning is applied directly to the complete model architecture and further strengthened through a three-phase progressive training strategy that produces superior performance compared to conventional single-phase approaches. This progressive approach prevents knowledge loss by slowing updates to important weights from previous tasks and allows the model to learn diacritical recognition while preserving its foundational MSA knowledge.

### 5.4. Experimental Evaluation and Performance Analysis

We conducted extensive experiments with 10-fold cross-validation to ensure robust and reliable performance assessment, where three distinct training approaches were evaluated to determine the most effective strategy for diacritical Arabic ASR.

#### 5.4.1. Performance Metrics

The performance of the proposed model is evaluated using the SASSC dataset. Two standard metrics—Word Error Rate (WER) and Character Error Rate (CER)—are employed to assess the accuracy of the models and to guide adjustments when necessary.

Word Error Rate (WER) is a common metric used to evaluate the performance of speech recognition systems. It measures the proportion of incorrectly predicted words compared to the reference transcription. The metric typically ranges from 0 to 1, where lower values indicate higher accuracy. WER is calculated as:

$$WER = \frac{S + D + I}{N} \quad (15)$$

where S denotes the number of substitutions, D the number of deletions, I the number of insertions, and N the total number of words in the reference transcript (the sum of correct and incorrect words).

Character Error Rate (CER) is a similar measure computed at the character level, reflecting how well individual characters are recognized. Like WER, CER values range from

0 (perfect match) to 1 (completely different), with lower values indicating better accuracy. CER is computed as:

$$CER = \frac{S + D + I}{N} \quad (16)$$

To further analyze model performance, Precision, Recall, and the F1-score are calculated to quantify the balance between false positives and false negatives:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \quad F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (17)$$

where TP, FP, and FN represent the counts of true positives, false positives, and false negatives, respectively. High Precision and Recall values indicate accurate and consistent predictions, while the F1-score provides a single measure that balances both.

The performance of the proposed model will be compared with that reported in [41], which uses the same dataset. Based on the experimental findings, the results will be disseminated through international conferences and peer-reviewed journals.

#### 5.4.2. Performance Comparison of Training Strategies

The experimental results demonstrated significant performance variations across the training strategies. Direct training on the diacritical dataset achieved an average WER of 34.09% and CER of 8.88%, representing the baseline performance when no transfer learning is applied. The traditional full fine-tuning approach, which involves pre-training on non-diacritic data followed by complete model fine-tuning on diacritical data, improved performance to an average WER of 30.15% and CER of 5.35%, corresponding to relative improvements of 11.56% and 39.75%, respectively. In contrast, the three-phase progressive fine-tuning strategy achieved the best performance across all metrics, with an average WER of 23.99% and CER of 4.49%. This represents remarkable improvements of 29.63% in WER and 49.44% in CER compared to direct training, and 20.43% in WER and 16.07% in CER compared to traditional fine-tuning. Table 4 shows the WER and CER results for the three training strategies.

**Table 4.** Performance Comparison of Different Training Strategies Using 10-Fold Cross-Validation.

| Fold    | Direct Training on Diacritical Dataset |       | Training on Non-Diacritic Dataset then Apply Full Fine-Tuning on Diacritical Dataset |      | Training on Non-Diacritic Dataset then Applying 3-Phase Fine-Tuning |      |
|---------|----------------------------------------|-------|--------------------------------------------------------------------------------------|------|---------------------------------------------------------------------|------|
|         | WER                                    | CER   | WER                                                                                  | CER  | WER                                                                 | CER  |
| 1       | 36.85                                  | 9.43  | 32.21                                                                                | 5.63 | 27.64                                                               | 4.81 |
| 2       | 20.01                                  | 8.63  | 23.71                                                                                | 3.99 | 10.90                                                               | 1.82 |
| 3       | 21.36                                  | 6.74  | 22.92                                                                                | 3.88 | 11.05                                                               | 2.08 |
| 4       | 37.95                                  | 8.11  | 31.09                                                                                | 5.40 | 27.08                                                               | 5.12 |
| 5       | 37.37                                  | 8.55  | 32.47                                                                                | 5.91 | 27.69                                                               | 5.36 |
| 6       | 34.65                                  | 7.79  | 32.25                                                                                | 5.64 | 25.69                                                               | 4.83 |
| 7       | 35.60                                  | 11.50 | 31.58                                                                                | 5.87 | 26.69                                                               | 5.13 |
| 8       | 38.33                                  | 10.71 | 30.96                                                                                | 5.60 | 26.10                                                               | 5.02 |
| 9       | 38.97                                  | 7.57  | 34.08                                                                                | 6.11 | 28.65                                                               | 5.37 |
| 10      | 39.76                                  | 9.77  | 30.22                                                                                | 5.49 | 27.46                                                               | 5.43 |
| Average | 34.09                                  | 8.88  | 30.15                                                                                | 5.35 | 23.99                                                               | 4.49 |

#### 5.4.3. Effectiveness of Three-Phase Progressive Fine-Tuning

The superior performance of the three-phase approach confirms our hypothesis that gradual unfreezing prevents knowledge loss, adapts to diacritical characteristics effectively, and yields improvements across all folds. For instance, Fold 10 showed a reduction from 39.76% to 27.46% WER when we compared direct training to three-phase fine-tuning, which represents a 30.94% relative improvement.

The CER improvements are particularly significant for diacritical Arabic recognition, as character-level accuracy directly reflects the system's ability to distinguish between diacritical variants. The reductions in CER across all folds indicate that the three-phase approach enhances the sensitivity of the model to acoustic differences associated with diacritical marks.

#### 5.4.4. Impact of Data Augmentation

Following the best results reported in [5], we applied parallel speed and volume data augmentation techniques. Table 5 shows the WER and CER for the original and augmented datasets.

**Table 5.** Performance Comparison Between Original and Augmented Datasets Using Three-Phase Fine-Tuning.

| Fold    | Original Dataset |      | Augmented Dataset |      |
|---------|------------------|------|-------------------|------|
|         | WER              | CER  | WER               | CER  |
| 1       | 26.64            | 4.81 | 25.06             | 4.31 |
| 2       | 10.90            | 1.82 | 8.24              | 1.66 |
| 3       | 11.05            | 2.08 | 8.72              | 2.29 |
| 4       | 27.08            | 5.12 | 23.39             | 4.32 |
| 5       | 27.69            | 5.36 | 25.71             | 4.93 |
| 6       | 25.69            | 4.83 | 25.15             | 5.03 |
| 7       | 26.69            | 5.13 | 26.13             | 4.69 |
| 8       | 27.10            | 5.02 | 25.58             | 4.67 |
| 9       | 28.65            | 5.37 | 26.52             | 4.90 |
| 10      | 27.46            | 5.43 | 25.58             | 4.93 |
| Average | 23.99            | 4.49 | 22.01             | 4.17 |

All folds showed consistent improvements in the data augmentation experiments. The augmented dataset achieved an average WER of 22.01% compared to 23.99% for the original dataset representing an additional 8.25% relative improvement. Moreover, the CER decreased from 4.49% to 4.17%, corresponding to a 7.13% relative improvement. The augmented dataset outperforms the original dataset and achieves around 2% and 0.3% reduction in WER and CER, respectively. These results indicate that data augmentation provides complementary benefits to the three-phase fine-tuning strategy.

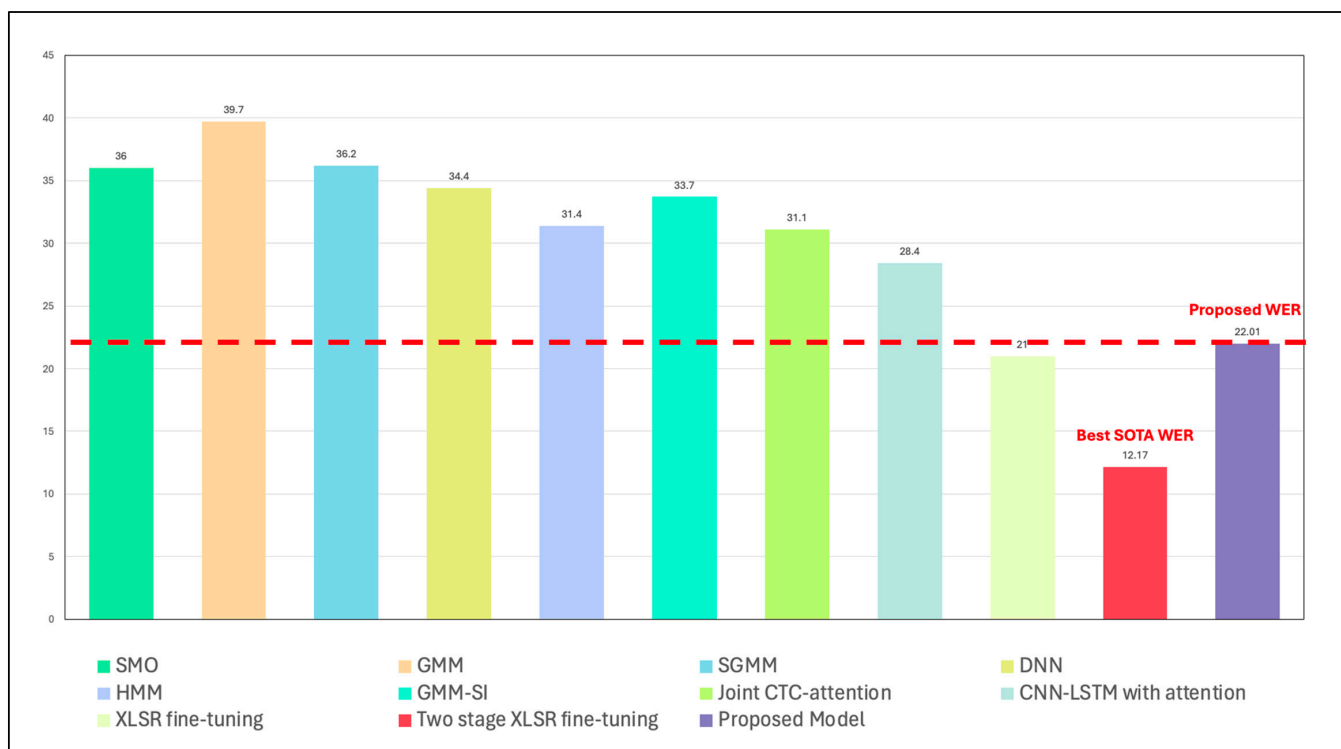
#### 5.5. Comparison with State-of-the-Art Models

To evaluate the effectiveness of our proposed approach, we compared our best-performing model (three-phase fine-tuning with data augmentation) against existing state-of-the-art diacritical Arabic ASR systems. Table 6 presents the comprehensive comparison results for the SASSC dataset, with WER values obtained from the respective publications. The results of a comparison with other diacritical Arabic ASR systems for the SASSC corpus are shown in Figure 3.

Our proposed method achieved competitive performance with a WER of 22.01%, while also providing distinct methodological advantages. The comparison reveals important insights about different training paradigms and their impact on performance.

**Table 6.** Comparison of State-of-the-Art Diacritical Arabic ASR Systems for the SASSC Dataset.

| Paper                | Traditional o2. | Model                           | WER   |
|----------------------|-----------------|---------------------------------|-------|
| Azim et al. [41,57]  | Traditional     | SMO                             | 36.0  |
| Ali et al. [41,58]   | Traditional     | GMM                             | 39.7  |
| Ali et al. [41,58]   | Traditional     | SGMM                            | 36.2  |
| Ali et al. [41,58]   | Traditional     | DNN                             | 34.4  |
| Azim et al. [41,59]  | Traditional     | HMM                             | 31.4  |
| Alsayadi et al. [41] | Traditional     | GMM-SI                          | 33.7  |
| Alsayadi et al. [41] | E2E             | Joint CTC-attention             | 31.1  |
| Alsayadi et al. [41] | E2E             | CNN-LSTM with attention         | 28.4  |
| Alaqel et al. [5]    | E2E             | XLSR finetuning                 | 21.0  |
| Alaqel et al. [5]    | E2E             | Two stage XLSR finetuning       | 12.17 |
| Our Model            | E2E             | Transformer Encoder + CTC + RPE | 22.01 |

**Figure 3.** Comparison of Word Error Rates (WER) for Arabic ASR models on the SASSC dataset. The dashed red line indicates the proposed model (WER = 22.01%); models above it perform worse (higher WER), while those below perform better.

### 5.5.1. Cross-Fold Evaluation Metrics

To provide a broader evaluation beyond WER, the model's performance was also analyzed across all 10 folds using precision, recall, and F1-score metrics. These measures assess how consistently the model distinguishes diacritical and non-diacritical characters throughout the dataset.

Table 7 reports the average performance consistency of the proposed model across all folds. The results show an overall mean F1-score of 92.99%, indicating strong generalization and stability.

**Table 7.** Precision, Recall, and F1-Score across 10 folds on the SASSC dataset.

| Fold | Precision | Recall | F1    |
|------|-----------|--------|-------|
| 1    | 92.99     | 93.48  | 93.24 |
| 2    | 96.34     | 97.66  | 97.00 |
| 3    | 95.13     | 96.18  | 95.65 |
| 4    | 92.09     | 92.92  | 92.50 |
| 5    | 91.66     | 92.29  | 91.97 |
| 6    | 91.61     | 92.11  | 91.86 |
| 7    | 91.07     | 92.14  | 91.60 |
| 8    | 92.13     | 92.82  | 92.47 |
| 9    | 91.80     | 92.05  | 91.93 |
| 10   | 91.07     | 92.35  | 91.70 |
| Mean | 92.59     | 93.40  | 92.99 |
| Std  | 1.68      | 1.84   | 1.75  |

### 5.5.2. Performance Analysis in Context

Our model achieved a WER of 22.01%, demonstrating strong performance compared to other diacritical Arabic ASR models. While the model does not achieve the lowest WER compared to XLSR models (12.17% WER and 21% WER), this comparison requires careful contextualization regarding computational resources and training data requirements.

The XLSR models use much more computational power and data. XLSR was pre-trained on 53 different languages using massive datasets with more than 300 M parameters, while our lightweight model has only 14 M parameters and was trained on a small MSA dataset first, then fine-tuned on diacritical Arabic. Despite using much fewer resources, our model (22.01% WER) performs almost as well as the single-stage XLSR model (21.0% WER). This small difference is impressive given that our model is much smaller and uses far less training data.

The XLSR models require massive computational resources and extensive training time. XLSR was pre-trained on 53 different languages using large-scale datasets with more than 300 M parameters over weeks or months on high-end GPU clusters. In contrast, our lightweight model with only 14 M parameters required around 15 h of total training time on a single Google Colab T4 GPU (12 h for MSA pre-training and 3 h for diacritical fine-tuning). Despite this dramatic difference in computational resources and training time, our model (22.01% WER) performs almost as well as the single-stage XLSR model (21.0% WER).

This small performance gap is remarkable, taking into consideration the resource efficiency. Our model achieves competitive results with over 100 times less training time and significantly lower computational requirements, making it more practical for researchers and developers with limited access to GPU clusters. The ability to train a competitive diacritical Arabic ASR model in under a day on freely available hardware represents a substantial advancement in accessibility.

Our model outperforms traditional methods, surpassing GMM systems (33.7–39.7% WER), HMM approaches (31.4% WER), and DNN models (34.4% WER) by large margins. Compared to other modern E2E systems, it also performs well against CNN-LSTM models (28.4% WER) and joint CTC-attention systems (31.1% WER) while offering superior training efficiency and deployment feasibility.

### 5.6. Computational Efficiency and Practical Implications

Our results establish a new benchmark for diacritical Arabic ASR systems that do not rely on extensive pretraining models. The achieved WER of 22.01% represents the best performance obtained with a transfer learning approach from MSA to diacritical Arabic. This makes our method particularly relevant for scenarios where computational resources for pre-training are limited. These findings demonstrate that effective diacritical

Arabic speech recognition can be achieved through architectural innovations and progressive training strategies, offering a practical alternative to expensive pretraining methods while maintaining competitive accuracy. Our method ensures transparency and reproducibility through a structured two-stage training process, delivering competitive results without requiring extensive computational resources.

The gap between our method (22.01%) and traditional approaches (28.4–39.7% WER) demonstrates the effectiveness of our architecture and training strategy. Our approach provides an effective balance of performance and efficiency for diacritical Arabic speech recognition systems, while acknowledging the superior WER performance of extensive pretrained models such as XLSR.

#### Inference Efficiency Comparison

To assess the deployment practicality of the proposed lightweight Transformer, inference latency was measured on a Google Colab T4 GPU (16 GB VRAM, batch size = 1) using 5 s audio inputs from the SASSC dataset. The comparison focused on real-time capability rather than training efficiency.

The proposed Lightweight Transformer ( $\approx 14$  M parameters) achieved an average inference latency of  $\approx 33.85$  ms per clip, confirming real-time performance with a Real-Time Factor (RTF) of 0.007 ( $\approx 148 \times$  faster than real time). The XLSR-Large baseline ( $\approx 300$  M parameters) processed the same input in  $\approx 14.05$  ms (RTF = 0.0147,  $\approx 71 \times$  faster than real time). A detailed comparison of latency, parameter count, and real-time performance is summarized in Table 8.

**Table 8.** Inference efficiency comparison of the proposed lightweight Transformer and the XLSR-Large baseline on the SASSC dataset. Latency is reported in milliseconds (ms) per audio clip, RTF denotes the Real-Time Factor, and Speed expresses the multiple of real-time performance ( $\times$ RT).

| Model          | Parameters | Latency | RTF    | Speed                   |
|----------------|------------|---------|--------|-------------------------|
| Proposed Model | 14         | 33.85   | 0.0071 | $\approx 148 \times$ RT |
| XLSR-Large     | 300        | 14.05   | 0.0147 | $\approx 71 \times$ RT  |

Although XLSR yields slightly lower latency, the proposed model delivers comparable inference responsiveness with  $20\times$  fewer parameters, enabling practical deployment on resource-limited systems while maintaining competitive WER performance.

#### 5.7. Per-Diacritic Performance Analysis

A detailed analysis of CER was performed for each diacritic using the final model (three-phase fine-tuning with data augmentation) across all ten folds to gain a deeper understanding of the model's behavior. The quantitative results are summarized in Tables 9 and 10.

**Table 9.** Per-diacritic CER across ten folds, with the final column showing the mean CER.

| Diacritic    | Folds |      |       |       |       |       |       |       |       |       | Mean CER |
|--------------|-------|------|-------|-------|-------|-------|-------|-------|-------|-------|----------|
|              | 1     | 2    | 3     | 4     | 5     | 6     | 7     | 8     | 9     | 10    |          |
| Fatha        | 5.03  | 2.67 | 3.55  | 5.91  | 6.82  | 5.63  | 5.81  | 6.43  | 6.16  | 5.89  | 5.39     |
| Kasra        | 7.24  | 2.47 | 4.07  | 8.72  | 10.56 | 8.81  | 8.58  | 9.32  | 7.97  | 8.29  | 7.57     |
| Damma        | 6.30  | 2.60 | 3.22  | 6.76  | 8.43  | 10.56 | 11.57 | 9.53  | 7.00  | 9.73  | 7.60     |
| Sukun        | 12.41 | 3.26 | 5.89  | 14.26 | 13.82 | 15.04 | 15.62 | 14.46 | 18.21 | 16.18 | 12.92    |
| Shadda       | 14.03 | 3.05 | 3.10  | 12.40 | 16.59 | 20.40 | 16.45 | 12.32 | 19.85 | 14.07 | 13.23    |
| Tanween Fath | 12.87 | 7.04 | 3.37  | 10.11 | 20.62 | 18.75 | 12.40 | 7.95  | 24.72 | 16.67 | 13.45    |
| Tanween Kasr | 39.13 | 2.82 | 18.07 | 21.69 | 26.15 | 17.95 | 14.08 | 23.88 | 24.68 | 16.92 | 18.99    |
| Tanween Damm | 30.77 | 7.55 | 4.55  | 6.82  | 11.43 | 19.61 | 30.00 | 26.00 | 35.29 | 17.86 | 20.54    |

**Table 10.** Average Precision, Recall, and F1-Score per Arabic diacritic. Results (mean  $\pm$  std) are averaged over ten folds. Count indicates the number of occurrences in the SASSC dataset.

| Diacritic    | Precision (Mean $\pm$ Std) | Recall (Mean $\pm$ Std) | F1 Score (Mean $\pm$ Std) | Count  |
|--------------|----------------------------|-------------------------|---------------------------|--------|
| Fatha        | 97.14 $\pm$ 0.68           | 98.75 $\pm$ 0.45        | 97.94 $\pm$ 0.44          | 40,385 |
| Kasra        | 96.94 $\pm$ 0.82           | 97.74 $\pm$ 0.94        | 97.33 $\pm$ 0.74          | 21,883 |
| Damma        | 96.77 $\pm$ 1.18           | 97.84 $\pm$ 1.02        | 97.30 $\pm$ 0.95          | 9874   |
| Sukun        | 95.63 $\pm$ 1.26           | 95.73 $\pm$ 1.72        | 95.67 $\pm$ 1.39          | 19,768 |
| Shadda       | 95.17 $\pm$ 2.01           | 95.83 $\pm$ 2.63        | 95.48 $\pm$ 1.72          | 5802   |
| Tanween Fath | 94.55 $\pm$ 1.85           | 96.05 $\pm$ 2.83        | 95.27 $\pm$ 1.88          | 937    |
| Tanween Kasr | 93.30 $\pm$ 3.13           | 93.09 $\pm$ 3.59        | 93.16 $\pm$ 2.85          | 729    |
| Tanween Damm | 93.59 $\pm$ 4.01           | 93.80 $\pm$ 4.73        | 93.60 $\pm$ 3.32          | 428    |

The analysis reveals that the model demonstrates high precision and recall across all diacritics, with mean F1-scores exceeding 93%. The lowest CERs were obtained for the short vowels Fatha (5.39%), Kasra (7.60%), and Damma (7.57%), which correspond to frequent and acoustically distinct phonemes in Arabic. Their clear spectral formant patterns enable the Transformer encoder to capture vowel-related cues effectively.

Moderate error rates were observed for Sukun (12.92%) and Shadda (13.23%). These marks are temporally or contextually complex—Sukun indicates the absence of a vowel, while Shadda represents gemination (consonant doubling). Both depend on subtle timing and intensity variations that are harder to learn consistently across speakers and recording conditions.

The highest CERs occurred in the Tanween forms—Tanween Fath (13.45%), Tanween Kasr (20.54%), and Tanween Damm (18.99%)—which exhibit strong data sparsity and weak acoustic salience. Their realization often relies on morphological context rather than clear acoustic cues, explaining the larger fold-to-fold variability (standard deviation  $\approx$  9–10%).

Overall, these findings indicate that the proposed model captures frequent vowelized diacritics with high consistency, while rarer and context-dependent Tanween marks remain challenging. This suggests that further improvement could be achieved through synthetic data balancing or morphologically aware augmentation strategies. The results confirm that the final Transformer effectively models the Arabic phonetic–diacritical structure, maintaining both linguistic accuracy and computational efficiency.

## 6. Conclusions and Limitations

This work presents a lightweight encoder-only Transformer architecture with CTC and RPE for diacritical Arabic ASR. We demonstrate that competitive performance can be achieved without extensive multilingual pre-training through systematic seven-phase ablation studies and three-phase progressive fine-tuning.

Our approach achieved a 22.01% WER on the SASSC dataset. This represents significant improvements compared to traditional methods (28.4–39.7% WER) and requires far fewer computational resources than pretrained ASR models ( $\sim$ 14 M vs.  $\sim$ 300 M parameters for XLSR). The two-stage training strategy leverages the linguistic relationship between MSA and diacritical Arabic, effectively addressing the challenge of limited diacritical training data through strategic transfer learning.

The comprehensive experimental validation shows that post-normalization configuration, Swish activation, and  $6\times$  feedforward expansion are the best architectural choices for this task. The integration of RPE and CTC proves particularly effective in capturing temporal dependencies that are crucial for distinguishing subtle diacritical variations in Arabic speech.

Overall, our findings demonstrate that it is possible to build accurate and resource-efficient diacritical Arabic ASR systems without relying on massive multilingual pretraining. By combining architectural innovations with a progressive fine-tuning strategy, we

provide a practical framework that balances accuracy and efficiency. This work not only establishes a new benchmark for diacritical Arabic ASR but also lays the foundation for extending lightweight ASR solutions to other low-resource languages and real-world applications where computational constraints are critical.

Although the proposed lightweight transformer with relative positional encoding demonstrates competitive accuracy and efficiency, several limitations remain. First, the training and evaluation were conducted on a single diacritical Arabic dataset (SASSC), which may limit generalization to broader dialectal or spontaneous speech. Second, although the proposed model substantially reduces parameter count and training cost, real-time inference on embedded or mobile devices was not fully explored. Third, this study focused exclusively on diacritical Arabic, without addressing local Arabic dialects, which may exhibit different phonetic and linguistic characteristics that affect recognition performance. Future work will focus on expanding dataset diversity, integrating more efficient decoding schemes, and testing deployment performance on edge hardware.

**Author Contributions:** Conceptualization, H.A. and K.E.H.; Methodology, H.A.; Software, H.A.; Validation, H.A.; Formal analysis, H.A.; Investigation, H.A.; Resources, H.A.; Data curation, H.A.; Writing—original draft, H.A.; Writing—review and editing, H.A. and K.E.H.; Supervision, K.E.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** The authors extend their appreciation to King Saud University, Saudi Arabia for funding this work through Ongoing Research Funding Program, (ORF-2025-953), King Saud University, Riyadh, Saudi Arabia.

**Data Availability Statement:** The data presented in this study are openly available in [SASSC] [[https://www.isca-archive.org/ssw\\_2013/almosallam13\\_ssw.pdf](https://www.isca-archive.org/ssw_2013/almosallam13_ssw.pdf)] (accessed on 15 October 2025) and Common Voice 6.1 dataset at [<https://commonvoice.mozilla.org/en/datasets>] (accessed on 15 October 2025).

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

|     |                                       |
|-----|---------------------------------------|
| ASR | Automatic Speech Recognition          |
| RPE | Relative positional encoding          |
| CTC | Connectionist temporal classification |
| MSA | Modern standard Arabic                |
| E2E | End-to-end                            |
| HMM | Hidden Markov Models                  |
| APE | Absolute positional encoding          |
| MHA | Multi head attention                  |
| WER | Word error rate                       |
| CER | Character error rate                  |

## References

1. Hackett, C.; Lipka, M. The demographic factors that make Islam the world's fastest-growing major religious group. *Scr. Instituti. Donneriani. Abo.* **2018**, *28*, 11–14. [[CrossRef](#)]
2. Algihab, W.; Alawwad, N.; Aldawish, A.; AlHumoud, S. Arabic Speech Recognition with Deep Learning: A Review. In *Social Computing and Social Media. Design, Human Behavior and Analytics*; Meiselwitz, G., Ed.; Springer International Publishing: Cham, Switzerland, 2019; pp. 15–31.
3. Zerari, N.; Abdelhamid, S.; Bouzgou, H.; Raymond, C. Bidirectional deep architecture for Arabic speech recognition. *Open Comput. Sci.* **2019**, *9*, 92–102. [[CrossRef](#)]

4. El Choubassi, M.M.; El Khoury, H.E.; Alagha, C.E.J.; Skaf, J.A.; Al-Alaoui, M.A. Arabic speech recognition using recurrent neural networks. In Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (IEEE Cat. No. 03EX795), Piscataway, NJ, USA, 17 December 2003; pp. 543–547.
5. Alaqel, H.; El Hindi, K. Improving Diacritical Arabic Speech Recognition: Transformer-Based Models with Transfer Learning and Hybrid Data Augmentation. *Information* **2025**, *16*, 161. [\[CrossRef\]](#)
6. Hamed, O.; Zesch, T. A Survey and Comparative Study of Arabic Diacritization Tools. *J. Lang. Technol. Comput. Linguist.* **2017**, *32*, 27–47. [\[CrossRef\]](#)
7. Ismail, H.; Abbache, M.; Zohra, B.F. An Approach for Arabic Diacritization. In *Natural Language Processing and Information Systems*; Métais, E., Meziane, S., Vadera, S., Solanki, M., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 337–344.
8. Abushariah, M. TAMEEM V1.0: Speakers and text independent Arabic automatic continuous speech recognizer. *Int. J. Speech Technol.* **2017**, *20*, 261–280. [\[CrossRef\]](#)
9. Al-Sulaiti, L.; Atwell, E.S. The Design of a Corpus of Contemporary Arabic. *Int. J. Corpus Linguist.* **2006**, *11*, 135–171. [\[CrossRef\]](#)
10. Alansary, S.; Nagi, M.; Adly, N. Building an International Corpus of Arabic (ICA): Progress of Compilation Stage. In Proceedings of the 7th International Conference on Language Engineering, Cairo, Egypt, 16–20 July 2007; pp. 5–6.
11. Farghaly, A.; Shaalan, K. Arabic Natural Language Processing: Challenges and Solutions. *ACM Trans. Asian Lang. Inf. Process.* **2009**, *8*, 1–22. [\[CrossRef\]](#)
12. Habash, N. Arabic Computational Linguistics. In *The Cambridge Handbook of Arabic Linguistics*; Cambridge University Press: Cambridge, UK, 2021; pp. 427–445.
13. Hori, T.; Cho, J.; Watanabe, S. End-to-end speech recognition with word-based RNN language models. In Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece, 18–21 December 2018; pp. 389–396.
14. Graves, A.; Mohamed, A.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649.
15. Battenberg, E.; Chen, J.; Child, R.; Coates, A.; Li, Y.G.Y.; Liu, H.; Satheesh, S.; Sriram, A.; Zhu, Z. Exploring Neural Transducers for End-to-End Speech Recognition. In Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Okinawa, Japan, 16–20 December 2017; IEEE: New York, NY, USA, 2017; pp. 206–213.
16. He, Y.; Sainath, T.N.; Prabhavalkar, R.; McGraw, I.; Alvarez, R.; Zhao, D.; Rybach, D.; Kannan, A.; Wu, Y.; Pang, R.; et al. Streaming End-to-End Speech Recognition for Mobile Devices. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; IEEE: New York, NY, USA, 2019; pp. 6381–6385.
17. Li, J.; Zhao, R.; Hu, H.; Gong, Y. Improving RNN Transducer Modeling for End-to-End Speech Recognition. In Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Singapore, 14–18 December 2019; IEEE: New York, NY, USA, 2019; pp. 114–121.
18. Li, J.; Zhao, R.; Meng, Z.; Liu, Y.; Wei, W.; Parthasarathy, S.; Mazalov, V.; Wang, Z.; He, L.; Zhao, S.; et al. Developing RNN-T Models Surpassing High-Performance Hybrid Models with Customization Capability. In Proceedings of the Interspeech, Shanghai, China, 25–29 October 2020; ISCA: Baixas, France, 2020; pp. 3590–3594.
19. Prabhavalkar, R.; Rao, K.; Sainath, T.N.; Li, B.; Johnson, L.; Jaitly, N. A Comparison of Sequence-to-Sequence Models for Speech Recognition. In Proceedings of the Interspeech, Stockholm, Sweden, 20–24 August 2017; ISCA: Baixas, France, 2017; pp. 939–943.
20. Punjabi, S.; Arisikere, H.; Raeesy, Z.; Chandak, C.; Bhave, N.; Bansal, A.; Müller, M.; Murillo, S.; Rastrow, A.; Stolcke, A.; et al. Joint ASR and Language Identification Using RNN-T: An Efficient Approach to Dynamic Language Switching. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; IEEE: New York, NY, USA, 2021; pp. 7218–7222.
21. Sainath, T.N.; He, Y.; Li, B.; Narayanan, A.; Pang, R.; Bruguier, A.; Chang, S.-Y.; Li, W.; Alvarez, R.; Chen, Z.; et al. A Streaming On-Device End-to-End Model Surpassing Server-Side Conventional Model Quality and Latency. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; IEEE: New York, NY, USA, 2020; pp. 6059–6063.
22. Saon, G.; Tüske, Z.; Bolanos, D.; Kingsbury, B. Advancing RNN Transducer Technology for Speech Recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; IEEE: New York, NY, USA, 2021; pp. 5654–5658.
23. Zhang, X.; Zhang, F.; Liu, C.; Schubert, K.; Chan, J.; Prakash, P.; Liu, J.; Yeh, C.-F.; Peng, F.; Saraf, Y.; et al. Benchmarking LF-MMI, CTC and RNN-T Criteria for Streaming ASR. In Proceedings of the IEEE Spoken Language Technology Workshop (SLT), Shenzhen, China, 19–22 January 2021; IEEE: New York, NY, USA, 2021; pp. 46–51.
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
25. Conneau, A.; Baevski, A.; Collobert, R.; Mohamed, A.; Auli, M. Unsupervised cross-lingual representation learning for speech recognition. *arXiv* **2020**, arXiv:2006.13979. [\[CrossRef\]](#)

26. Hsu, W.-N.; Bolte, B.; Tsai, Y.-H.H.; Lakhota, K.; Salakhutdinov, R.; Mohamed, A. HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2021**, *29*, 3451–3460. [\[CrossRef\]](#)
27. Kheddar, H.; Himeur, Y.; Al-Maadeed, S.; Amira, A.; Bensaali, F. Deep Transfer Learning for Automatic Speech Recognition: Towards Better Generalization. *Appl. Sci.* **2023**, *13*, 6630. [\[CrossRef\]](#)
28. Thomas, B.; Kessler, S.; Karout, S. Efficient Adapter Transfer of Self-Supervised Speech Models for Automatic Speech Recognition. In Proceedings of the ICASSP 2022—IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 7102–7106.
29. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv* **2019**, arXiv:1901.02860.
30. Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-attention with relative position representations. *arXiv* **2018**, arXiv:1803.02155. [\[CrossRef\]](#)
31. Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 369–375.
32. Abed, S.; Alshayej, M.; Sultan, S. Diacritics effect on Arabic speech recognition. *Arab. J. Sci. Eng.* **2019**, *44*, 9043–9056. [\[CrossRef\]](#)
33. Al-Anzi, F.S.; AbuZeina, D. The effect of diacritization on Arabic speech recognition. In Proceedings of the 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Aqaba, Jordan, 11–13 October 2017; pp. 1–5.
34. Al-Anzi, F.S.; AbuZeina, D. The impact of phonological rules on Arabic speech recognition. *Int. J. Speech Technol.* **2017**, *20*, 715–723. [\[CrossRef\]](#)
35. Pantazoglou, F.K.; Papadakis, N.K.; Kladis, G.P. Implementation of the Generic Greek Model for CMU Sphinx Speech Recognition Toolkit. In Proceedings of the eRA-12 International Scientific Conference at: Thivon 250 & P. Ralli Av., PUAS Conference Center, Athens, Greece, October 2017.
36. AbuZeina, D.; Al-Khatib, W.; Elshafei, M.; Al-Muhtaseb, H. Toward enhanced Arabic speech recognition using part-of-speech tagging. *Int. J. Speech Technol.* **2011**, *14*, 419–426. [\[CrossRef\]](#)
37. Boumehdi, A.; Yousfi, A. Comparison of new approaches of semi-syllable units for speech recognition of any Arabic word. *J. Phys. Conf. Ser.* **2022**, 2337, 12002. [\[CrossRef\]](#)
38. Ding, I.J.; Hsu, Y.M. An HMM-like dynamic time warping scheme for automatic speech recognition. *Math. Probl. Eng.* **2014**, *2014*, 898729. [\[CrossRef\]](#)
39. Zheng, F.; Zhang, G.; Song, Z. Comparison of Different Implementations of MFCC. *J. Comput. Sci. Technol.* **2001**, *16*, 582–589. [\[CrossRef\]](#)
40. Rabiner, L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE* **1989**, *77*, 257–286. [\[CrossRef\]](#)
41. Alsayadi, H.A.; Abdelhamid, A.A.; Hegazy, I.; Fayed, Z.T. Arabic speech recognition using end-to-end deep learning. *IET Signal Process.* **2021**, *15*, 521–534. [\[CrossRef\]](#)
42. Haeb-Umbach, R.; Ney, H. Linear Discriminant Analysis for Improved Large Vocabulary Continuous Speech Recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), San Francisco, CA, USA, 23–26 March 1992; Volume 92, pp. 13–16.
43. Gales, M.J.F. Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition. *Comput. Speech Lang.* **1998**, *12*, 75–98. [\[CrossRef\]](#)
44. Leggetter, C.J.; Woodland, P.C. Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models. *Comput. Speech Lang.* **1995**, *9*, 171–185. [\[CrossRef\]](#)
45. Huang, X.D.; Jack, M.A. Semi-Continuous Hidden Markov Models for Speech Signals. *Comput. Speech Lang.* **1989**, *3*, 239–251. [\[CrossRef\]](#)
46. Davis, S.; Mermelstein, P. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans. Acoust. Speech Signal Process.* **1980**, *28*, 357–366. [\[CrossRef\]](#)
47. Passricha, V.; Aggarwal, R.K. A Hybrid of Deep CNN and Bidirectional LSTM for Automatic Speech Recognition. *J. Intell. Syst.* **2019**, *29*, 1261–1274. [\[CrossRef\]](#)
48. Watanabe, S.; Hori, T.; Kim, S.; Hershey, J.R.; Hayashi, T. Hybrid CTC/Attention Architecture for End-to-End Speech Recognition. *J. Sel. Top. Signal Process.* **2017**, *11*, 1240–1253. [\[CrossRef\]](#)
49. Ali, A.M.; Mubarak, H.; Vogel, S. Advances in dialectal Arabic speech recognition: A study using Twitter to improve Egyptian ASR. In Proceedings of the IWSLT, Lake Tahoe, CA, USA, 4–5 December 2014.
50. AbdAlmisreb, A.; Abidin, A.F.; Tahir, N.M. Maxout based deep neural networks for Arabic phonemes recognition. In Proceedings of the 2015 IEEE 11th International Colloquium on Signal Processing & Its Applications (CSPA), Kuala Lumpur, Malaysia, 6–8 March 2015; pp. 192–197.
51. Ramachandran, P.; Zoph, B.; Le, Q.V. Searching for activation functions. *arXiv* **2017**, arXiv:1710.05941. [\[CrossRef\]](#)

52. Torrey, L.; Shavlik, J. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*; IGI Global: Hershey, PA, USA, 2010; pp. 242–264.
53. Baeviski, A.; Zhou, Y.; Mohamed, A.; Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 12449–12460.
54. Ardila, R.; Branson, M.; Davis, K.; Kohler, M.; Meyer, J.; Henretty, M.; Morais, R.; Saunders, L.; Tyers, F.M.; Weber, G. Common Voice: A Massively-Multilingual Speech Corpus. In Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), Marseille, France, 13–15 May 2020.
55. Almosallam, I.; Alkhalifa, A.; Alghamdi, M.; Alkanhal, M.; Alkhairy, A. SASSC: A Standard Arabic Single Speaker Corpus. In Proceedings of the Eighth ISCA Workshop on Speech Synthesis, Barcelona, Spain, 31 August 2013.
56. Graves, A.; Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1764–1772.
57. Azim, M.A.; Badr, N.L.; Tolba, M.F. An Enhanced Arabic Phonemes Classification Approach. In Proceedings of the 10th International Conference on Informatics and Systems, Giza, Egypt, 9–11 May 2016; pp. 210–214.
58. Ali, A.; Zhang, Y.; Cardinal, P.; Dahak, N.; Vogel, S.; Glass, J. A Complete Kaldi Recipe for Building Arabic Speech Recognition Systems. In Proceedings of the 2014 IEEE Spoken Language Technology Workshop (SLT), South Lake Tahoe, NV, USA, 7–10 December 2014; IEEE: New York, NY, USA, 2014; pp. 525–529.
59. Azim, M.; Abdelhamid, A.; Badr, N.; Tolba, M. Large Vocabulary Arabic Continuous Speech Recognition Using Tied States Acoustic Models. *Asian J. Inf. Technol.* **2019**, *18*, 49–56. [[CrossRef](#)]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.