electronics

IMPACT FACTOR 2.6

CITESCORE 6.1

Article

Implementation of a Whisper Architecture-Based Turkish Automatic Speech Recognition (ASR) System and Evaluation of the Effect of Fine-Tuning with a Low-Rank Adaptation (LoRA) Adapter on Its Performance

Hüseyin Polat, Alp Kaan Turan, Cemal Koçak and Hasan Basri Ulaş

Special Issue

Computational Intelligence and Machine Learning: Models and Applications

Edited by

Dr. Grzegorz Dudek

MDPI

https://doi.org/10.3390/electronics13214227

electronics

MDPI

Article

Implementation of a Whisper Architecture-Based Turkish Automatic Speech Recognition (ASR) System and Evaluation of the Effect of Fine-Tuning with a Low-Rank Adaptation (LoRA) Adapter on Its Performance

Hüseyin Polat [1,*], Alp Kaan Turan [1], Cemal Koçak [1] and Hasan Basri Ulaş [2]

[1] Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara 06560, Turkey; akaan.turan@gazi.edu.tr (A.K.T.); ccckocak@gazi.edu.tr (C.K.)
[2] Department of Manufacturing Engineering, Faculty of Technology, Gazi University, Ankara 06560, Turkey; bulas@gazi.edu.tr
* Correspondence: polath@gazi.edu.tr

Abstract: This paper focuses on the implementation of the Whisper architecture to create an automatic speech recognition (ASR) system optimized for the Turkish language, which is considered a low-resource language in terms of speech recognition technologies. Whisper is a transformer-based model known for its high performance across numerous languages. However, its performance in Turkish, a language with unique linguistic features and limited labeled data, has yet to be fully explored. To address this, we conducted a series of experiments using five different Turkish speech datasets to assess the model's baseline performance. Initial evaluations revealed a range of word error rates (WERs) between 4.3% and 14.2%, reflecting the challenges posed by Turkish. To improve these results, we applied the low-rank adaptation (LoRA) technique, which is designed to fine-tune large-scale models efficiently by introducing a reduced set of trainable parameters. After fine-tuning, significant performance improvements were observed, with WER reductions of up to 52.38%. This study demonstrates that fine-tuned Whisper models can be successfully adapted for Turkish, resulting in a robust and accurate end-to-end ASR system. This research highlights the applicability of Whisper in low-resource languages and provides insights into the challenges of and strategies for improving speech recognition performance in Turkish.

Keywords: automatic speech recognition; artificial intelligence; deep learning; representation learning; self-supervised learning; Whisper model

Citation: Polat, H.; Turan, A.K.; Koçak, C.; Ulaş, H.B. Implementation of a Whisper Architecture-Based Turkish Automatic Speech Recognition (ASR) System and Evaluation of the Effect of Fine-Tuning with a Low-Rank Adaptation (LoRA) Adapter on Its Performance. *Electronics* **2024**, *13*, 4227. https://doi.org/10.3390/electronics13214227

Academic Editor: Grzegorz Dudek

Received: 2 October 2024
Revised: 24 October 2024
Accepted: 25 October 2024
Published: 28 October 2024

1. Introduction

Automatic speech recognition (ASR) systems have become an integral part of many modern technologies, enabling voice-activated assistants, transcription services, and real-time communication across various platforms. The development of ASR systems has historically been focused on high-resource languages such as English, which benefit from large, labeled datasets and sophisticated linguistic models. However, low-resource languages, including Turkish, continue to face challenges in terms of speech recognition accuracy due to the scarcity of labeled data and unique linguistic features such as agglutinative morphology and vowel harmony [1].

The Whisper architecture, developed by OpenAI, represents a significant advancement in ASR technology. As a transformer-based model, Whisper is designed to handle multiple languages, accents, and noisy environments with high accuracy. However, despite its broad language support, its performance in low-resource languages like Turkish has not been fully optimized, as most of the model's training data are skewed toward high-resource languages such as English [2]. The need for the fine-tuning of such models in low-resource languages is evident to improve their performance in real-world applications [3].

Turkish presents specific challenges for ASR systems due to its agglutinative structure, where suffixes are attached to root words to form complex words. This results in a vast number of possible word forms, making it difficult for ASR systems to accurately segment and recognize words. Additionally, Turkish exhibits notable dialectal diversity and phonological complexity, further complicating the development of a robust ASR system [4]. The scarcity of high-quality labeled datasets exacerbates these issues, leading to higher word error rates (WERs) compared to other languages.

In this study, we aim to address the gap in the literature by adapting the Whisper ASR system to Turkish using the low-rank adaptation (LoRA) method, a parameter-efficient fine-tuning technique that allows large-scale models to be adapted to specific tasks without the need for retraining all the model parameters [5]. LoRA significantly reduces the computational cost and memory requirements of fine-tuning large models, making it ideal for adapting ASR systems to low-resource languages like Turkish. By fine-tuning Whisper with LoRA, we aim to enhance its performance on Turkish speech datasets and provide a more accurate and robust ASR solution for Turkish.

Previous research on ASR systems for Turkish has primarily focused on traditional machine learning approaches and language models tailored to Turkish [6]. However, these approaches often lack the scalability and adaptability offered by modern transformer-based models like Whisper. Our work bridges this gap by leveraging the strengths of Whisper while addressing the specific challenges posed by the Turkish language. The key contribution of this study is the evaluation of Whisper's performance across five distinct Turkish speech datasets, both before and after fine-tuning with LoRA. This comprehensive evaluation provides insights into how well transformer-based models can be adapted to low-resource languages and highlights the potential for further improvements in ASR systems for Turkish.

Literature Review

The system model for speech analysis and synthesis was proposed by Dudley et al. at Bell Laboratories [7,8] in 1939, which is considered the beginning of ASR systems [9]. The first experimental work was the system for isolated digit recognition for a single speaker developed by Davis et al. of Bell Laboratories in 1952 [10]. Between 1950 and 1960, studies aimed to create pattern recognition systems for phoneme, single letter, or syllable discrimination [11,12]. In the period between 1960 and 1970, three hardware-based systems were developed in Japan [13–15]. Additional prominent works include IBM's Shoebox software [16], Martin's work at RCA Laboratories [17], and Vintsyuk's study using dynamic programming methods [18]. Between 1970 and 1980, the Viterbi algorithm was used in ASR systems [19], along with statistics-based approaches such as Itakura's LPC-based study [20]. Noteworthy works include the VIP-100 software, the Hearsay and HWIM software developed within the scope of the DARPA SUR program, as well as the Harpy software developed by Carnegie-Mellon University [21].

Between the 1980s and 1990s, statistical models and artificial neural network (ANN) studies emerged. The period between 1990 and 2000 saw developments such as the AT&T Voice Recognition Call Processing (VRCP) solution and the Hidden Markov Model Tool Kit (HMM Tool Kit). After 2000, ANN-based systems continued to develop, and significant advancements like the "Voice Search" feature in Android and Siri integration with iOS were introduced. Additionally, the Effective Affordable Reusable Speech-to-Text (EARS) and Global Autonomous Language Exploitation (GALE) datasets were created within the DARPA program for ASR systems [22–28].

After 2010, the use of artificial intelligence (AI) in automatic speech recognition (ASR) systems has increased, with significant developments such as connectionist temporal classification (CTC), recurrent neural network (RNN), long short-term memory (LSTM), two-way LSTM using the listen, attend, speech (LAS) mechanism, convolution mechanism, residual network-based studies, the transducer mechanism, Wav2vec model, Wav2vec 2.0, the ASR system using a convolutional neural network (CNN) and the Conformer

model, and the Whisper ASR system developed by OpenAI. Major companies such as IBM, Microsoft, Google, and Amazon also released their advanced ASR systems during this period [29–36].

The existing literature on Turkish speech recognition primarily focuses on the distinctive characteristics of the language and their implications for automatic speech recognition (ASR) systems. Turkish poses several challenges in terms of speech recognition due to its agglutinative structure, rich morphology, complex phonology, and significant dialectal diversity. To address issues related to this diversity, ASR systems often rely on robust language models. In addition, the use of a lexicon helps improve recognition accuracy by providing a structured mapping of words, which is particularly useful in managing the vast variety of word forms in Turkish. Despite these techniques, the scarcity of high-quality speech and text data remains a major obstacle to further advancement in the field [37–39].

The early works on Turkish ASR focused on developing speaker-dependent systems and acoustic models. In recent years, there has been significant progress in Turkish ASR systems, with the introduction of deep learning models such as LSTM, gated recurrent unit (GRU), and Transformer, and the augmentation of Turkish speech and text data [40–51].

This research explores the implementation of a Whisper-driven automatic speech recognition system for the Turkish language and evaluates the effects of refining the model using LoRA approach. Whisper is a relatively new technology, so there are a limited number of studies on it. Various studies have compared Whisper with other architectures, such as Wav2Vec 2.0, and have utilized different datasets for testing. Additionally, some researchers have employed innovative methods, such as image-converted EEG data and direct fine-tuning, to improve Whisper's ASR performance [52–56].

This study expands upon the existing body of research on Turkish ASR by addressing some of the critical limitations that have hindered previous efforts in the field. Earlier works on Turkish ASR typically focused on traditional machine learning models, such as hidden Markov models (HMMs) or Gaussian mixture models (GMMs), and relied on relatively small or less diverse datasets. These models, while effective in some contexts, struggled with the unique linguistic challenges of Turkish, such as its agglutinative morphology, rich inflectional structure, and the presence of numerous dialects.

In contrast, this study leverages the Whisper ASR model, which is based on a transformer architecture trained on a large, multilingual corpus. This allows the model to capture long-range dependencies in speech and handle the complexities of Turkish more effectively than traditional models. The key differentiating factor of our work is the use of the LoRA method for fine-tuning Whisper. LoRA enables the model to be fine-tuned efficiently with fewer trainable parameters, addressing the computational challenges associated with fine-tuning large-scale models for low-resource languages like Turkish.

Moreover, unlike many previous studies that focused on a single dataset or task, this study evaluates the Whisper model on multiple Turkish speech datasets, representing a more comprehensive assessment of its performance across different contexts. This study also introduces corrections to the datasets, improving the quality of the training data and reducing the impact of noise and other errors.

2. Materials and Methods

ASR systems convert spoken language into text. Unlike humans, who use past experiences and grammar, ASR systems process speech as an audio signal [57–59]. While traditional ASR systems have separate components, modern end-to-end architectures perform these processes in a single step. With recent advancements in deep learning, pre-trained transformer-based large-scale models have gained popularity for fine-tuning specific ASR tasks. Transformer models, originally developed for natural language processing, have been effectively repurposed for speech recognition. Their exceptional ability to capture long-range dependencies in sequential data is critical for accurately comprehending the contextual nuances of speech. This adaptation has significantly improved the

performance and accuracy of ASR systems, leading to more reliable and efficient speech recognition technology [60].

2.1. Transformers

RNNs and their variations, such as LSTM and GRU, have succeeded significantly in tasks like machine translation, language modeling, sequence transfer, and sequence modeling. The success of RNNs in these tasks is due to their ability to process sequential data. However, their sequential nature limits parallel processing, as each step depends on the result of the previous step, making it challenging to handle long sequences effectively. Over time, RNNs tend to forget important information, especially in long sequences, which leads to difficulties in capturing long-term dependencies. The sequence-to-sequence (Seq2Seq) [61] architecture was proposed to address these challenges. This architecture consists of an encoder and a decoder, typically composed of RNN units. The encoder takes the input sequence and compresses it into a fixed-length context vector, which is passed to the decoder. The decoder then generates the output sequence from this context vector. Seq2Seq improves the learning process by efficiently handling sequence data. However, when dealing with long sequences, it experiences performance degradation because it compresses the entire input into a fixed-length vector, which increases the risk of losing crucial information.

To overcome this issue, the attention mechanism was introduced. The attention mechanism allows the model to focus on specific parts of the input sequence dynamically, rather than compressing all the input information into a single fixed-length vector. This enables the model to consider different input parts at each step of the output generation, improving the performance, particularly for long sequences. By highlighting important points in the data, attention mechanisms make it easier for the model to retain critical information. However, even with attention, challenges like poor long-distance dependency retention, high computational costs, and limited parallel processing still persist [62].

To address these limitations and overcome the challenges of RNN-based models, the transformer architecture was proposed by Vaswani et al. in 2017 [32]. Unlike RNNs, transformers remove the need for sequential processing and rely entirely on the attention mechanism. This architecture allows for parallel processing and significantly reduces the risk of losing information in long sequences. One of the most significant advantages of the transformer is its ability to support parallel processing, as it removes the need for step-by-step sequential computations. These speed up the training process, especially when working with large datasets. Additionally, the self-attention mechanism excels in tasks requiring long-term dependencies by capturing relationships between distant elements in a sequence. This feature enables it to overcome one of the key weaknesses of RNNs and Seq2Seq models, which often struggle with such dependencies. The transformer is also highly scalable, making it a foundational architecture for large language models such as bidirectional encoder representations from transformers (BERT) and generative pre-trained transformer (GPT).

The basic structure of the transformer architecture is illustrated in Figure 1. The transformer consists of an encoder–decoder structure designed to process sequences of data (like sentences).

The encoder processes the input data. It consists of N layers. Each layer has two sub-layers: multi-head self-attention mechanism and position-wise fully connected feed-forward network Each of these sub-layers is followed by add and layer normalization steps, which stabilize and normalize the input. Positional encoding is added to the input embeddings to provide the model with information about the position of tokens in a sequence. This is necessary because, unlike RNNs or LSTMs, the transformer model does not inherently understand the order of tokens.

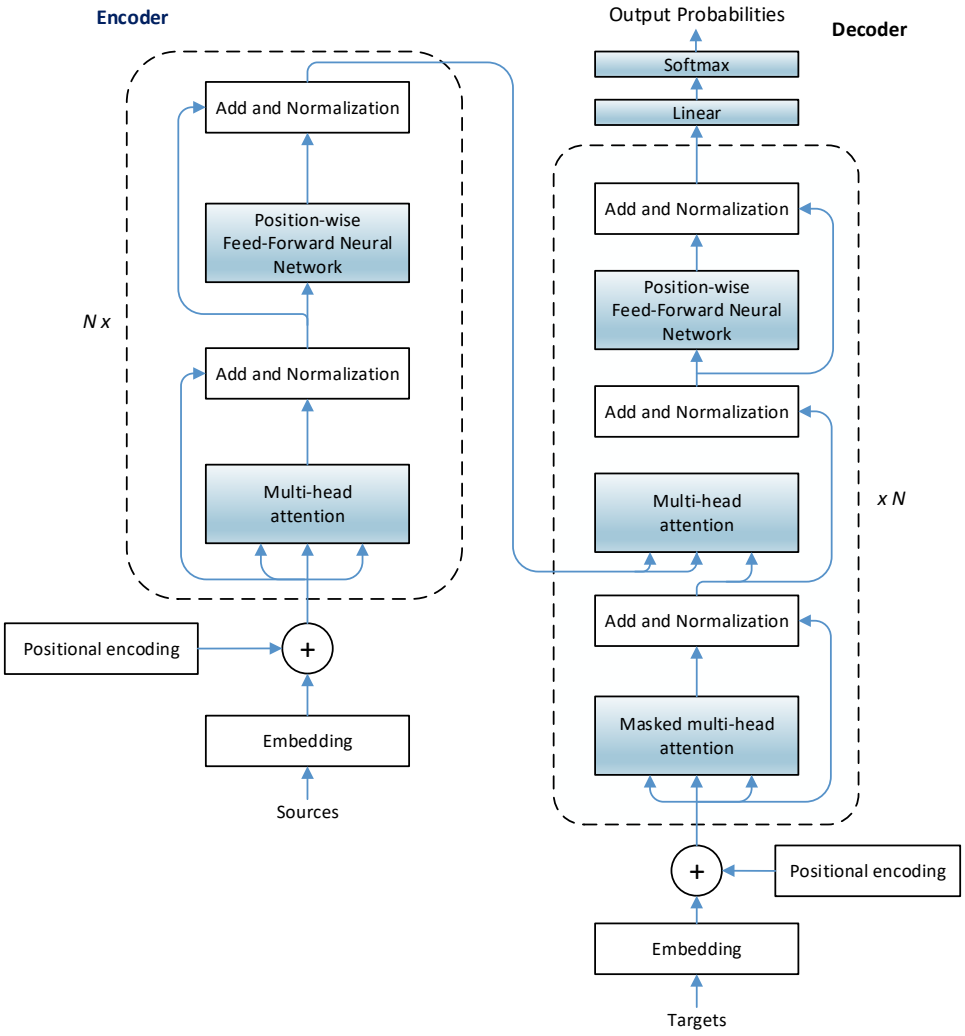

Figure 1. Basic transformer architecture.

The decoder generates the output sequence, conditioned on the input sequence encoded by the encoder. The decoder also has N layers, similar to the encoder but with a few differences.

In addition to the multi-head self-attention mechanism and feed-forward network, the decoder includes a masked multi-head attention layer. This ensures that predictions for a given position only depend on the outputs before that position, preserving the autoregressive nature of the language generation. Similar to the encoder, each layer in the decoder is followed by add and layer normalization steps.

2.2. Components of the Transformer

2.2.1. Multi-Headed Attention

Instead of relying on a single attention function, the transformer employs multi-head attention. In this approach, multiple attention heads operate in parallel, allowing the model to capture different aspects of the information simultaneously. Each attention head computes a weighted sum of the input vectors, focusing on different positions of the input. This mechanism helps the model understand the relationships between words in the sequence, regardless of their distance.

As can be seen in Figure 2, the multi-head attention (MHA) is formed by the combination of parallel scaled dot-product attention mechanisms. By multiplying the transformed

input X by separate weight matrices, the query, key and value are obtained, which are fed to each attention unit and given in Equations (1)–(3).

$$Q = XW_Q^T \tag{1}$$

$$K = XW_K^T \tag{2}$$

$$V = XW_V^T \tag{3}$$

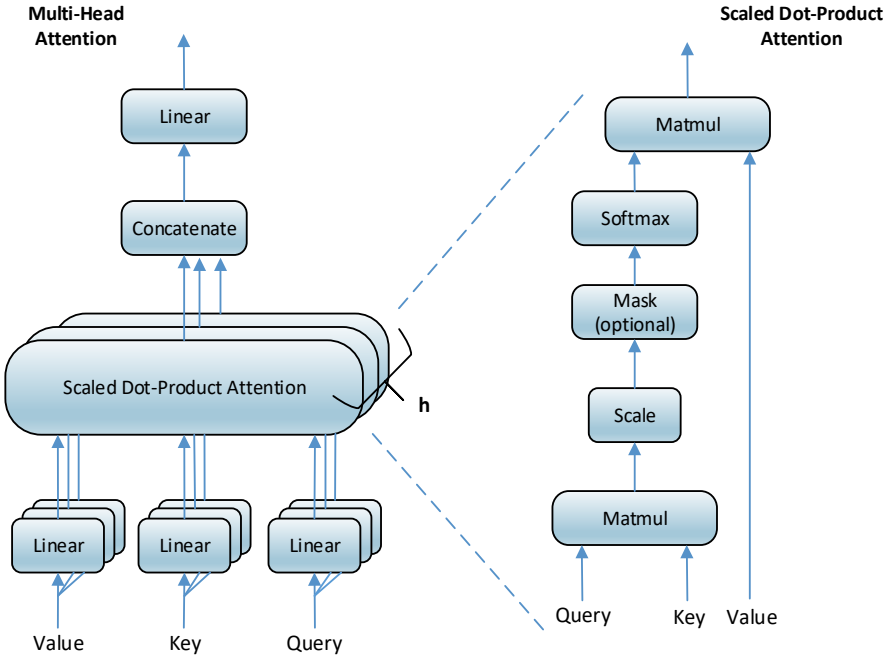

Figure 2. Multi-headed attention.

In Equation (4), the query and key matrix product is scaled by the square root of the dimension and the weight obtained by the softmax function is multiplied by the value matrix.

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = \text{softmax}\left(\frac{\text{QK}^{\text{T}}}{\sqrt{\text{d}_\text{k}}}\right)\text{V} \tag{4}$$

The result from each attention unit (Equation (5)) is combined, as shown in Equation (6) and then transmitted to the next layer.

$$\text{Head}_\text{i} = \text{Attention}\left(\text{QW}_\text{i}^\text{Q}, \text{KW}_\text{i}^\text{K}, \text{VW}_\text{i}^\text{V}\right) \tag{5}$$

$$\text{MHA}(\text{Q}, \text{K}, \text{V}) = \text{Concat}(\text{Attention}_{1,}\text{Attention}_{2,} \ldots) \tag{6}$$

2.2.2. Positional Encoding

As the array length increases, the index of the input becomes very large. In this state, the indices are not suitable for use in the transformer. Although the normalization process provides a solution up to a point, differences in array size cause problems again.

With positional encoding, the input array is transformed into a position matrix using the sine and cosine functions. In positional encoding, the order of the input output size of the model, denoted as d_{model} $\left(0 \leq i \leq \frac{d_{\text{model}}}{2}\right)$ and user-defined criterion n, is as outlined in Equations (7) and (8).

$$PE_{(pos,2i)} = sin\left(pos/n^{2i/d_{\text{model}}}\right) \tag{7}$$

$$PE_{(pos,2i+1)} = cos\left(pos/n^{2i/d_{\text{model}}}\right) \quad (8)$$

To illustrate, the positional encoding matrix of the initial six words in a text input with $d = 4$ and $n = 10000$ is presented in Table 1.

Table 1. Positional encoding matrix example.

i=	0	0	One	One
	$\mathbf{sin}\left(\frac{\mathbf{pos}}{\mathbf{n^{2i/d}}}\right)$	$\mathbf{cos}\left(\frac{\mathbf{pos}}{\mathbf{n^{2i/d}}}\right)$	$\mathbf{sin}\left(\frac{\mathbf{pos}}{\mathbf{n^{2i/d}}}\right)$	$\mathbf{cos}\left(\frac{\mathbf{pos}}{\mathbf{n^{2i/d}}}\right)$
x0	0	1	0	1
x1	0.841471	0.540302	0.009999	0.999950
x2	0.909297	−0.416147	0.019999	0.999800
x3	0.141120	−0.989992	0.029996	0.999550
x4	−0.756802	−0.653644	0.039989	0.999200
x5	−0.958924	0.283662	0.049979	0.998750

2.2.3. Feed-Forward Neural Network

The feed-forward neural network (FNN) is fed with the output of the multi-head attention mentioned in the previous part. The computation of the FNN is performed as in Equation (9). The bold part represents the ReLU activation function. The output of the previous layer is weighted and fed into the ReLU activation function. It is then multiplied by a second weight matrix and a constant is added and used as input for the analyzer.

$$\text{FNN}(x) = \mathbf{max}(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)W_2 + b_2 \quad (9)$$

The same mechanisms involved in the encoder are also involved in the decoder. The difference is that in the positional encoding process in the encoder, the input x_1, x_2, x_3,... x_n in the parser when receiving y_0 (start), y_1, y_2, y_3, ... y_{n-1} is used. The first MHA in the analyzer is masked to prevent overlearning. The last output in the analyzer is the SoftMax function, which assigns a probability value between 0 and 1 to each element in the result array and selects the outputs with the highest probability.

2.2.4. Embedding Layer

Transforms input tokens (source or target) into a high-dimensional space, converting each token into a vector.

2.2.5. Masked Multi-Head Attention

This is applied in the decoder to prevent it from attending to future tokens that have not been generated yet. It is essential for tasks such as language generation, where the model must predict the next token in a sequence by relying solely on the previous tokens.

2.2.6. Add and Layer Normalization

After each multi-head attention or feed-forward layer, the model applies residual connections, where the original input is added to the output. This is followed by layer normalization, which helps stabilize the training and reduce overfitting, contributing to the overall efficiency and robustness of the model.

2.2.7. SoftMax Layer

After the decoder produces the output probabilities, they are passed through a SoftMax layer to generate a probability distribution over the possible output tokens.

2.3. Low-Rank Adaptation

The models used in large language model (LLM), NLP, ASR systems usually contain a lot of parameters and have large sizes. For example, Whisper large-v2 has about 1.5 billion parameters, while in GPT-3 this number increases to 175 billion. For such models, training

all the parameters in the fine-tuning process is very costly in terms of processing power and time.

Low-rank adaptation (LoRA) is a technique developed to efficiently fine-tune large pre-trained models, including language and diffusion models. It achieves this by drastically reducing the number of trainable parameters, making the fine-tuning process more efficient and resource-friendly. LoRA keeps the weights of the pre-trained model fixed. Instead of updating all the parameters, it introduces trainable low-rank matrices into each layer of the model. This method relies on the low-rank decomposition of weight matrices. This involves breaking down a large matrix into the product of two smaller matrices, which reduces the number of parameters that need to be trained [6].

LoRA significantly reduces the computational and memory overhead associated with fine-tuning large models. This makes it feasible to adapt large models to specific tasks without the need for extensive computational resources. Compared to direct fine-tuning of the model, this method can reduce the number of trained parameters to about a thousandth and the memory requirement to about a third. Despite the reduction in trainable parameters, LoRA often matches or exceeds the performance of full fine-tuning. It has been shown to perform well on models like a robustly optimized BERT pretraining approach (RoBERTa), decoding-enhanced BERT with disentangled attention (DeBERTa), GPT-2, and GPT-3. Unlike some other fine-tuning methods, LoRA does not introduce additional inference latency, making it suitable for real-time applications. The operation of the method is shown in Figure 3.

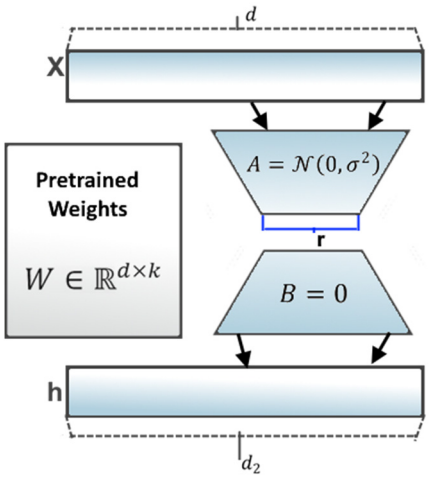

Figure 3. How the low-rank adaptation (LoRA) method works.

In standard fine-tuning, the pre-trained weight matrices of a neural network are updated directly. For the sake of simplicity, we will denote a weight matrix in the network by W, where $W \in \mathrm{R}^{\mathrm{d} \times \mathrm{k}}$, with d representing the input dimension and k representing the output dimension [6]. During fine-tuning, the weight matrix W is updated as follows:

$$W' = W + \Delta W \tag{10}$$

In this context, W' represents the updated weight matrix, W denotes the original pre-trained weight matrix, and ΔW is the full-rank matrix of learned weight updates, with a dimensionality of $d \times k$. The number of parameters in ΔW is $d \times k$.

The objective of LoRA is to reduce the number of trainable parameters by constraining ΔW to being a low-rank matrix. Instead of learning the complete matrix ΔW, LoRA postulates that ΔW can be decomposed into the product of two lower-dimensional matrices: $\Delta W = AB$, where:

$A \in \mathrm{R}^{\mathrm{d} \times \mathrm{r}}$ is a matrix with a rank r much smaller than min (d, k).

$B \in \mathrm{R}^{\mathrm{r} \times \mathrm{k}}$ is another matrix with the same rank r.

Thus, the update rule becomes:

$$W' = W + AB \tag{11}$$

This decomposition indicates that, instead of learning $d \times k$ parameters, it is sufficient to learn $d \times r$ parameters for matrix A and $r \times k$ parameters for matrix B, resulting in a total of $r(d + k)$ parameters. If r is considerably smaller than $min(d, k)$, this signifies a notable reduction in the number of trainable parameters.

We can calculate the reduction in parameters more explicitly. For the full-rank update ΔW, there are:

$$\text{Full-rank parameters} = d \times k \tag{12}$$

For the low-rank update $\Delta W = AB$, the parameter count is:

$$\text{Low-rank parameters} = r(d + k) \tag{13}$$

The ratio of the number of low-rank parameters to the number of full-rank parameters is as follows:

$$\text{Parameter reduction ratio} = r(d + k)/d \times k \tag{14}$$

At the beginning of the training, A is assigned a random Gaussian value and B is assigned 0. Therefore, $\Delta W = BA$ has a value of zero in the first stage. It is then scaled by $\frac{\alpha}{r}$ using a hyperparameter (α).

2.4. Whisper Model Architecture

Whisper [4,5], developed by OpenAI, is an open-source ASR (automatic speech recognition) system that is capable of transcribing and translating spoken language. Whisper is based on an encoder–decoder transformer architecture. The capacity of transformer models to track the interrelationships between words and sentences enables the consideration of long-range dependencies, a capability that is not afforded by other models. In other words, transformers are capable of recalling previously uttered words and sentences, which enables them to contextualize new utterances and thereby enhance the accuracy of their transcription. Whisper is distinguished from other ASR models by its end-to-end deep learning model, extensive language support, training on a large and diverse dataset, and high performance.

Traditional ASR models, often based on HMM-GMM, use probabilistic and statistical methods to model speech signals. Still, they have limitations in understanding complex language models. Other ASR models that use RNNs, LSTMs, or GRUs process sequential data to capture temporal dependencies in audio. However, due to their sequential nature, these models tend to be slower when processing longer speech segments. As a transformer-based model, Whisper processes data in parallel, enhancing the speed and efficiency. Its attention mechanism effectively captures long-term speech dependencies and relationships between words, which can be challenging and time-consuming for other models to learn.

Previously, ASR technology combined deep learning with HMM-GMM structures and other audio–text processing techniques. The major drawback of these approaches was their inability to be trained end-to-end; each model component needed separate training. The advent of end-to-end models, such as Whisper, eliminated this limitation by adopting a unified modeling approach that discards the traditional distinction between acoustic and language models.

Whisper officially supports around 100 languages. Whisper was trained on a large dataset of approximately 680,000 h, of which around 117,000 h are multilingual. This constitutes an order of magnitude more data than used to train Wav2Vec 2.0, namely 60,000 h of unlabeled audio. Of the data utilized for Whisper's training, 65% (or 438,000 h) was dedicated to English speech recognition, 17% (or 117,000 h) to multilingual speech recognition, and the remaining 18% (or 126,000 h) to English translation. This diverse dataset enhances Whisper's robustness and generalizability across different languages,

accents, and speech types. Whisper can transcribe 99 different languages and is capable of not just transcription but also translating conversations and timestamping speech. The model can be optimized for various tasks and is available in five different sizes, ranging from 39 million to 1.55 billion parameters, allowing developers to strike a balance between accuracy and processing speed [4,5].

Whisper's architecture is composed of two main components: the encoder and the decoder (Figure 4). The raw audio is divided into 30 s segments and transformed into a log-Mel spectrogram, which generates perceptually relevant frequency representations. Whisper has been designed to work on audio samples of up to 30 s in duration. However, the use of a chunking algorithm allows it to be used to transcribe audio samples of any length. This is made possible through the transformer's pipeline method.

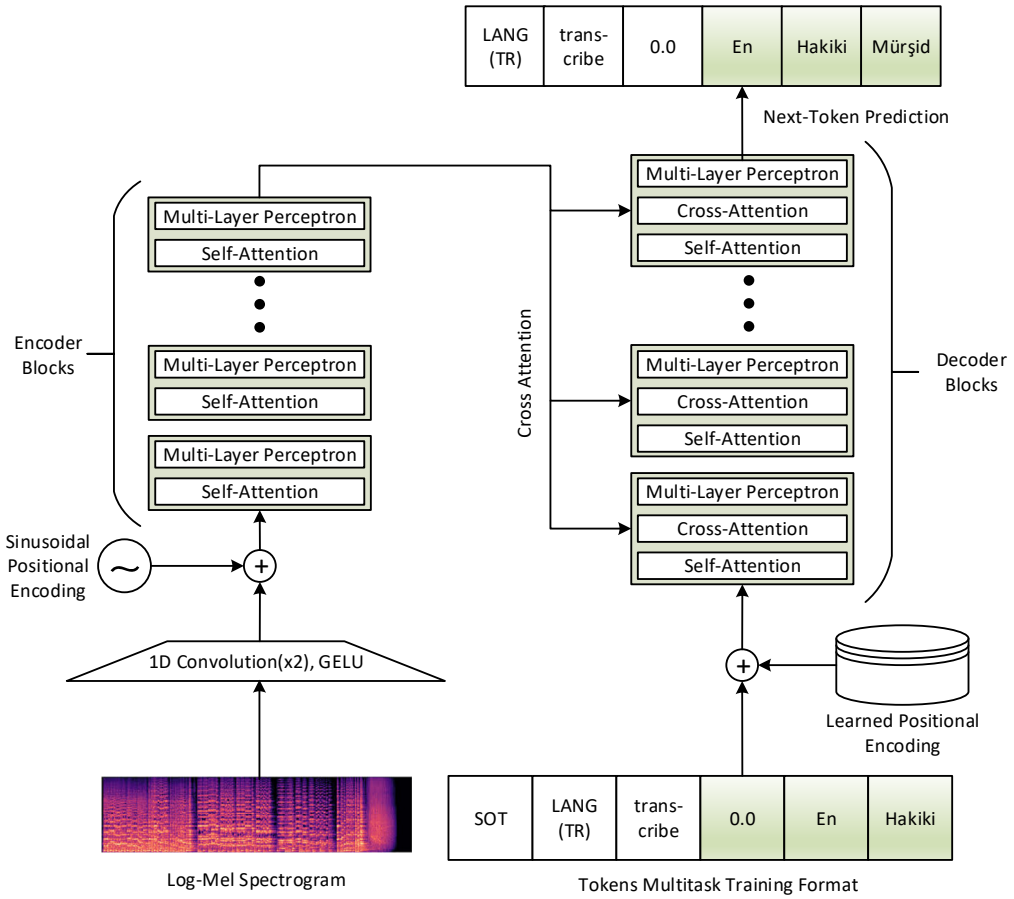

Figure 4. Whisper model architecture.

The spectrogram is processed through 2x1D convolutional layers with GELU (Gaussian error linear unit) activations to match the transformer's width [63]. Positional coding assigns temporal locations to the outputs of these convolutional layers, helping the model track sequential information from the audio data. This processed input is then fed into the encoder stack in the transformer [4,5].

The encoder consists of multiple blocks, each containing self-attention and multi-layer perceptron (MLP) layers. The self-attention mechanism analyzes each time segment of the audio concerning all the others, capturing short- and long-term speech dependencies. For instance, a word at the start of a sentence may be semantically related to one at the end; self-attention allows for such long-term correlations. After establishing these relationships through self-attention, the multi-layer perceptron layer enables deeper and more advanced processing of these connections, allowing the model to extract nuanced features and discern a broader range of linguistic patterns. Multiple encoder blocks are required to grasp the

complexities of language and expedite understanding. The encoder operates only once per 30 s segment to produce a latent representation from the spectrogram [4,5].

This latent representation is passed to the decoder, where each block contains cross-attention, self-attention, and multi-layer perceptron layers. Cross-attention facilitates the transfer of the latent representation from the encoder to the decoder, linking the audio data to the text generation process. This allows the decoder to make accurate predictions based on the audio signal. Self-attention within the decoder considers previously generated words to maintain grammatical consistency and meaning in the text output. The multi-layer perceptron in the decoder enriches the information provided by self-attention and cross-attention, ensuring that the text generated by the model is both grammatically coherent and accurate in capturing the subtleties of the audio signal. The decoder predicts the text step by step using the latent representation; first predicting the most likely word, then generating the next word based on the previous prediction, and continuing until the entire speech sequence is transcribed [4,5].

Based on the number of layers, the width of the feature representation, and the number of attention heads, the Whisper model is categorized into five versions: tiny, base, small, medium, and large. The specifics of each version are summarized in Table 2. Furthermore, Large-v2 had 2.5 times more training iterations than the large version, while Large-v3 utilized data collection, processing, and pseudo-labeling with Large-v2 to augment the training data to 5 million hours. Both the large-v2 and large-v3 models surpassed the large model, with the large-v3 model exhibiting even stronger capabilities than the large-v2, particularly in terms of model training [4,5].

Table 2. Whisper model types and configuration parameters.

Model	Layers	Width	Attention Heads	Parameters (Million)
tiny	4	384	6	39
base	6	512	8	74
small	12	768	12	244
medium	24	1024	16	769
large	32	1280	20	1550

This study identified the following factors as influencing the selection of Whisper over other potential ASR models.

The Whisper model is a transformer-based ASR system that has demonstrated robust performance across a range of languages, including low-resource languages such as Turkish. The robust performance of Whisper in noisy environments and its capacity to handle a diverse range of speech conditions, including accents and spontaneous speech, render it particularly well-suited for real-world applications in Turkish speech recognition.

Whisper operates in an end-to-end manner, whereby the model is trained to map audio inputs directly to text outputs, obviating the need for separate acoustic and language models. This unified approach markedly enhances the system's capacity to generalize across disparate languages and accents. Additionally, Whisper's transformer architecture enables parallel processing, rendering it more efficient than RNNs or LSTM units, which process input data sequentially and are slower when handling long speech sequences.

Furthermore, Whisper is capable of fine-tuning through techniques such as LoRA, which enables the model to be efficiently adapted to specific tasks or languages without the necessity of retraining all the parameters. This represents a significant advantage over other models, which may require more extensive resources to achieve the same level of fine-tuning. In light of the challenges posed by Turkish, including its agglutinative morphology and dialectal variations, the ability of Whisper to be fine-tuned with fewer trainable parameters represents a significant advantage and was a key factor in its selection for this study.

In conclusion, Whisper was selected for this study due to its advanced transformer-based architecture, multilingual training, adaptability to low-resource languages, and efficient fine-tuning capabilities. These features render it a more suitable model for developing a robust ASR system for Turkish compared to other traditional or even deep learning-based ASR models.

2.5. Turkish Speech Datasets

In this section, the five Turkish speech datasets, the Middle East Technical University (METU) Microphone Speech Dataset (METU MS), Turkish Broadcast News Speech and Text Dataset (TNST), FLEURS Dataset, Mozilla Common Voice Dataset (Mozilla CV) and Turkish Automatic Speech Recognition Test (TASRT), used in the experimental studies are introduced by listing their characteristics, such as the size, audio file type, metadata fields, etc. The METU MS, TNST, FLEURS, Mozilla CV and TASRT datasets contain 6618, 82331, 3127, 52,477 and 286 records, respectively.

METU MS dataset's [64] data availability status: data available in a publicly accessible repository. The original data presented in the study are openly available at https://catalog.ldc.upenn.edu/LDC2006S33 (accessed on 15 September 2024).

The METU MS dataset consists of approximately 5.6 h of audio and corresponding textual transcripts from 120 speakers of various age groups (19–50 years) and genders (60 males, 60 females). Each speaker vocalized 40 randomly selected sentences out of a total of 2462 sentences. Audio data, WAV with a sample frequency of 16 kHz, are stored in TXT format files in the form of text.

TNST dataset's [65] data availability status: data available in a publicly accessible repository. The original data presented in the study are openly available at https://catalog.ldc.upenn.edu/LDC2012S06 (accessed on 15 September 2024).

The TNST is a dataset of speeches collected from news bulletins of various Turkish-language news channels. It contains 194 h of audio and text data consisting of approximately 13,000,000 words. The audio data are in the WAV format, with a sampling frequency of 16 kHz, and the text is in TXT format files.

Mozilla CV dataset's [66] data availability status: data available in a publicly accessible repository. The original data presented in the study are openly available at https://commonvoice.mozilla.org/en/datasets (accessed on 15 September 2024).

The Mozilla CV is an open-source voice dataset created by the Mozilla company. This dataset consists of speech recordings of more than 500 languages, donated by volunteers from around the world. The most recent version released at the time of writing, Common Voice Corpus 15.0, contains 115 h of audio files from 1511 individuals, 111 h of which have been verified. The audio files are in MP3 format, with a sampling frequency of 48 kHz. The speech text is stored in TSV format files, with attributes such as the accent, gender, age, region, etc.

FLEURS dataset's [67] data availability status: data available in a publicly accessible repository. The original data presented in the study are openly available at https://huggingface.co/datasets/google/fleurs (accessed on 15 September 2024).

The FLEURS is a dataset of approximately 12 h of speech in 102 different languages, one for each language. The audio data are stored in WAV files, with a sampling frequency of 16 kHz. The speech text is stored in TSV format files, with attributes such as the text, speaker gender, etc. The recordings are divided into three parts: dev, train and test. The Turkish dataset consists of a total of 3607 records, with 743, 2526, and 338 records in the train, test, and dev sections, respectively.

TASRT dataset's [68] data availability status: data available on request due to restrictions (commercial use). The data presented in this study are available from the corresponding author upon request.

The TASRT dataset is compiled by Oyucu [68]. The TASRT dataset contains approximately 186 min of speech data from 286 speakers (143 women and 143 men) in 20 different categories. The audio data are stored in WAV files, with a sampling frequency of 16 kHz.

The dataset contains transcript files with TXT extension corresponding to each audio file and named with category, speaker gender and order.

2.6. Automatic Speech Recognition Performance Evaluation Criteria

The three possible types of errors that can be encountered in a text generated with ASR systems are insertion, deletion and substitution. Insertion is the insertion of an extra symbol/word that is not in the text. Deletion is the absence of a symbol/word that should be in a certain position in the text. Substitution is when a symbol/word that should be in a certain position in the text is replaced by another symbol/word.

For a text consisting of N_W words, the number of words added is I_W, deleted is D_W and substituted is S_W Then, the word error rate (WER) is calculated as given in Equation (15) and the accuracy, also referred to as the word recognition rate (WRR), is calculated as given in Equation (16).

$$\mathrm{WER} = \frac{\mathrm{I_W} + \mathrm{D_W} + S_W}{\mathrm{N}_W} \tag{15}$$

$$\mathrm{WRR} = 1 - \mathrm{WER} \tag{16}$$

The WER is generally used as the ASR evaluation criterion. Another widely used measure is the character error rate (CER). The CER value is calculated similarly to the WER value. N_C indicates the number of symbols, the added symbol number is I_C, the deleted symbol number is D_C and *the* substitution is S_C. The CER and character recognition rate (CRR) are found as given in Equations (17) and (18).

$$\mathrm{CER} = \frac{\mathrm{E_C} + \mathrm{S_C} + \mathrm{D_C}}{\mathrm{N_C}} \tag{17}$$

$$\mathrm{CRR} = 1 - \mathrm{CER} \tag{18}$$

The WER and CER are often confused with each other due to their similarities in calculations. While the WER is measured by the number of word errors; in terms of the CER, the number of symbol errors is taken into account. If we consider the following expression:

Bir işi yapmak için neden yarını bekliyorsun bugün de dünün bir yarını değil midir

Let us assume that the output produced by the ASR system for a sentence consisting of 14 words and 82 symbols with spaces is as follows.

Biri işi yapmak işin neden yarın_bekliyorsun bugün de dünün bir yarını değil_____

Symbol insertion was performed in the first word, a symbol change was made in the fourth word, and symbol deletion was performed in the sixth word. The last word has been completely deleted. There have been 7 deletions, 1 substitution, 1 insertion, giving total of 9 symbol errors; There are a total of 4 word errors, 3 changes and 1 deletion.

In this situation,$\mathrm{WER} = \frac{4}{14} \cong 0.286, \mathrm{CER} = \frac{9}{82} \cong 0.089$

Apart from these, although not common, there are also the sentence error rate (SER), phone error rate (PER), utterance error rate (UER), and frame error rate (FER). Criteria such as these can also be used to measure ASR performance. Low error rates indicate high accuracy and thus high model success.

3. Testing the Performance of Whisper Models on Turkish Speech Datasets and Fine-Tuning with LoRA

This paper presents the implementation of an end-to-end Turkish speech recognition system, utilizing the Whisper ASR model. Whisper is a robust ASR architecture; however, its performance requires further enhancement for low-resource languages such as Turkish. In this study, the performance of five scaled pre-trained models of the Whisper architecture (small, basic, small, medium, and large) was evaluated concerning Turkish speech recognition using five distinct Turkish speech datasets (METU MS, TNST, Mozilla CV, FLEURS, and TASRT). In addition to the comparison of the Whisper-large-v2 and Whisper-large-v3 models in terms of performance, a comparison was also conducted between Google USM

and the Whisper-large-v3 model using five distinct Turkish datasets. Subsequently, the Whisper models were fine-tuned with the LoRA method, and the performance of the Whisper models in terms of Turkish speech recognition was evaluated in comparison. The word error rate was employed as the performance metric.

In this study, experiments were conducted using a workstation equipped with a 32-core Intel(R) Xeon(R) Gold 6426Y 2.50 GHz 2-processor and an Nvidia A5000 GPU. The experiments were carried out using the *Python v3.12* programming language and the *Miniconda v24.5.0* environment. *PyTorch v2.4.0*, a machine learning and deep learning library developed by Facebook, was utilized for the tests with the datasets. Audio file processing was performed using the *pydub v0.25.1* library. The Whisper ASR model library, developed by OpenAI, along with the fast-whisper, Whisper online, and Whisper Live applications, was employed. To address the issue of Whisper ASR converting numbers into text, the *num2words v0.5.2* library was utilized with modifications. The *Numpy v2.1.0* and *Pandas v2.2.2* libraries were employed for data processing and calculations, while the *jiwer v3.0.4* libraries were used for calculating the WER and CER. Additionally, standard libraries such as *datetime* for duration determination, *io*, *sys*, and *tarfile* for file operations, *re* for text processing and regular expressions, and *argparse* for managing application parameters were also used.

During this study, we conducted around 950 h of work. This included approximately 150 h of examining and editing datasets, as well as 800 h dedicated to experimenting, Whisper fine-tuning, and developing a simultaneous ASR.

3.1. Preparation of Datasets

The METU MS, TNST, Mozilla CV, FLEURS and TASRT Turkish datasets were used in the experimental study. These datasets consist of WAV files with a 16 kHz sampling frequency and the Mozilla CV dataset consists of MP3 files with a 48 kHz sampling frequency. The MP3 files of the Mozilla CV dataset were converted to a single channel with a sampling frequency of 16 kHz.

Before starting the experiments, we analyzed the Turkish speech datasets. It became clear that the METU MS, TNST, and TASRT datasets needed some corrections. The issues and necessary modifications identified in the datasets are outlined below.

Some texts in the METU MS dataset do not comply with the Turkish Language Association's spelling rules or are not correctly vocalized. In this case, even though the ASR system produced the correct text, the CER and WER values were higher than they should have been due to spelling errors in the dataset. Common spelling errors include the spelling of dates without spaces, the contractions of expressions such as "bir şey", "her şey" and "birçok", and the contractions of suffixes and conjunctions such as "-mi", "-mi", "-de", "-da", "-ki". Even though it is rare, another type of error found in the dataset is the writing of Turkish characters such as ç, ğ, i, ö, ş, and ü without a dot.

Similarly, there are many problematic records in the TNST dataset. Adjacent words ae were written together. Additionally, many foreign words are included with their Turkish pronunciations. These issues caused an increase in the WER and CER values.

The texts in the datasets were reorganized by listening to the audio files in cases of ambiguity. Foreign names in the TNST dataset, which were written side by side with their Turkish pronunciations, were reduced to one in the text so that only their commonly used equivalents were included. Apart from these, other problems encountered with both the text files and the functioning of the existing model are listed below:

1. In some files, errors occurred in the output produced because the recording was terminated before the end of the speech. Especially in recordings with more than one speaker's speech, it was observed that if there were gaps, the model did not process the speech of subsequent speakers. Therefore, voice activation detection (VAD) was used to reduce the gaps that cause errors.
2. Confusion about whether conjunctions such as "with", "ise", "de", "da" are written adjacent to the word or separately is one of the most common errors. In the text,

suffixes or conjunctions, which are difficult to identify even for the listener, are sometimes misspelled or incorrectly transformed by the model. The text was corrected where it should have been separate or adjacent.
3. When the end of one word and the start of the next word have similar sounds, the model sometimes merges them. This causes it to skip the start of the following word.
4. The model abbreviates expressions such as doctor, professor, etc., kilometers and converts them into text as Dr., Prof., etc., km.

It was important to determine how the corrections made to the METU MS, TNST, and TASRT datasets affected the performance of the pre-trained Whisper models. To this end, the original and the corrected versions of the datasets were subjected to testing with pre-trained Whisper models, and the outcomes were evaluated. Afterward, LoRA was used to fine-tune the pre-trained Whisper models with the corrected datasets. The performances of these fine-tuned models were then evaluated.

3.2. Testing the Performance of Whisper Models on Five Turkish Speech Datasets

Table 3 presents the WER, CER and the duration values in hours and minutes for the pre-tests conducted on the Whisper models at five different scales (tiny, base, small, medium, large) before any modifications were made to the Turkish speech datasets. In the test, all of the samples in the METU MS, TASRT and FLEURS datasets with a low number of records and 30% of the samples in the TNST and Mozilla CV datasets with a high number of records were used.

Table 3. Pre-testing the performance of the Whisper models on five Turkish speech datasets.

Dataset	Metric	Tiny	Base	Small	Medium	Large-v2
METU MS (100%)	WER	0.36	0.24	0.16	0.15	0.10
	CER	0.10	0.07	0.06	0.04	0.03
	Duration	00:20	00:21	00:25	00:36	00:47
TNST (30%)	WER	0.53	0.36	0.22	0.15	0.13
	CER	0.18	0.12	0.08	0.06	0.05
	Duration	01:30	01:33	01:57	02:44	03:35
FLEURS (100%)	WER	0.47	0.31	0.17	0.11	0.08
	CER	0.15	0.09	0.04	0.03	0.02
	Duration	00:16	00:18	00:22	00:31	00:43
Mozilla CV (30%)	WER	0.49	0.37	0.24	0.19	0.16
	CER	0.22	0.16	0.12	0.11	0.09
	Duration	00:52	00:56	01:05	01:28	01:50
TASRT (100%)	WER	0.41	0.29	0.19	0.14	0.13
	CER	0.14	0.09	0.06	0.05	0.04
	Duration	00:03	00:03	00:04	00:07	00:10

In the METU MS dataset, while the WER is 0.36 in the tiny model, it drops to 0.10 in the large-v2 model. In other words, the errors decreased as the model size increased. While the CER is 0.10 in the tiny model, it decreases to 0.03 in the large-v2 model. The TNST dataset has higher error rates compared to the METU MS. Especially, the tiny model has the highest error rate, with a WER of 0.53. However, again with the large-v2 model, the WER drops to 0.13. The FLEURS dataset performs better than the TNST. With the large-v2 model, the WER drops to 0.08 and the CER to 0.02, indicating very low error rates. The Mozilla CV dataset also exhibits high error rates, similar to the TNST. Even in the large-v2 model, the WER drops to 0.16, but this is higher than in the METU MS and FLEURS. The TASRT dataset shows a balanced performance across all the models. In the large-v2 model, the WER is 0.13 and the CER is 0.04. Compared to the other datasets, the average error rates are lower.

3.3. Testing Whisper Models After Data Correction

The results obtained from the performance tests performed after rearranging the METU MS, TNST and TASRT datasets are shown in Table 4.

Table 4. Test performance evaluation of the Whisper models after dataset correction.

Dataset	Metric	Tiny	Base	Small	Medium	Large-v2
METU MS (100%)	WER	0.33	0.21	0.12	0.11	0.06
	CER	0.09	0.06	0.04	0.04	0.02
	Duration	00:20	00:21	00:24	00:35	00:47
TNST (30%)	WER	0.52	0.35	0.20	0.14	0.10
	CER	0.17	0.10	0.06	0.05	0.04
	Duration	01:13	01:18	01:37	02:15	02:57
TASRT (100%)	WER	0.38	0.25	0.15	0.09	0.08
	CER	0.13	0.08	0.05	0.03	0.03
	Duration	00:03	00:03	00:04	00:07	00:10

In the experiments performed using the Whisper-large-v2 model after rearranging the datasets, in the METU MS dataset, there is a notable decline in the WER and CER ratios with an increase in the model size. The Large-v2 model exhibits the lowest WER of 0.06 and the lowest CER of 0.02. This suggests that the dataset was transcribed with a high degree of accuracy. In comparison to the other datasets, the TNST dataset exhibits higher WER and CER ratios. This may be indicative of the dataset being more challenging to transcribe or comprising elements such as different languages and accents. The lowest WER was 0.10 in the "large-v2" model, while the lowest CER was 0.04. In the TASRT dataset, the WER and CER rates decrease with the model size. Notably, the values of the WER (0.08) and CER (0.03) in the "large-v2" model demonstrate an improvement trend comparable to that observed in the other datasets.

3.4. Performance Comparison of Whisper-Large-v2 and Whisper-Large-v3 Models

On 6 November 2023, the Whisper-large-v3 model was introduced by OpenAI. This model has the same number of layers, attention headers and parameters as the Whisper-large model. However, it differs from the other models by using 128 Mel frequency bands instead of 80 as input and by adding Cantonese to the group of defined languages. The Whisper-large-v3 model was obtained by training the Whisper-large-v2 model with one million hours of weakly labeled and four million hours of pseudo-labeled data. Table 5 shows the comparison of the Whisper-large-v2 and Whisper-large-v3 models for the datasets studied.

The Whisper-large-v3 model yielded substantial enhancements in the WER and CER values across all the datasets. The most substantial improvements were observed in the METU MS dataset, which may suggest that the model is more effectively tailored to the data in this particular dataset or that this dataset derives greater benefit from the enhancements introduced in the v3 model. The improvements in the WER ranged from 8.77% to 29.08%, while those in the CER ranged from 6.29% to 44.27%. No notable alterations were observed in the processing times, which remained largely consistent or exhibited only minimal fluctuations. A 3.60% increase in the processing time was observed on the Mozilla CV dataset, but this is equivalent to approximately four seconds and does not have a significant practical impact.

The Whisper-large-v3 model demonstrates a notable enhancement in accuracy relative to its predecessor, the large-v2 model, while exhibiting a minimal increase in the processing time. This renders the v3 model a more appealing option for users seeking to attain enhanced performance in transcription tasks.

Table 5. Performance comparison of the Whisper-large-v2 and Whisper-large-v3 models.

Dataset	Metric	Large-v2	Large-v3	Difference (%)
METU MS (100%)	WER	0.061	0.043	−29.08
	CER	0.018	0.010	−44.27
	Duration	00:47	00:47	0.00
TNST (30%)	WER	0.103	0.085	−17.30
	CER	0.037	0.028	−24.97
	Duration	02:57	02:58	0.86
FLEURS (100%)	WER	0.083	0.075	−10.11
	CER	0.021	0.020	−7.41
	Duration	00:43	00:43	−0.96
Mozilla CV (30%)	WER	0.156	0.142	−8.77
	CER	0.090	0.084	−6.29
	Duration	01:50	01:54	3.60
TASRT (100%)	WER	0.081	0.069	−13.84
	CER	0.032	0.027	−15.35
	Duration	00:10	00:10	−0.94

3.5. Performance Comparison of Whisper and Google USM Models

Whisper and Google USM are two current speech recognition systems. OpenAI has shared the Whisper ASR system under an open-source distribution license. Google promised to openly share its model for Google USM, but this has not happened over time. In contrast, the model can be accessed through paid application programming interfaces (APIs). The optimization points of Google USM, which runs behind APIs, and the hardware on which it runs are not clear. Despite these uncertainties about the model, a performance comparison between the two systems can be performed indirectly. A performance comparison between the two ASR systems was performed on the WER and CER using the METU MS, TNST, FLEURS, Mozilla CV and TASRT datasets. A certain number of random speech files were selected from each dataset. The selected files were given as input to the Whisper large-v3 model and the API (called Chirp) running Google USM behind it. The resulting output texts were compared with real speech texts to find the WER and CER values. The number of samples for the datasets used in the comparison, the criterion values obtained and the percentage differences between the values of the two models are given in Table 6.

Table 6. Performance comparison of the Whisper and Google USM models.

Dataset	Metric	Whisper Large-v3	Google USM	Difference(%)
METU MS (100%)	WER	0.043	0.049	13.95
	CER	0.01	0.01	0
TNST (30%)	WER	0.084	0.087	3.57
	CER	0.028	0.025	−10.71
FLEURS (100%)	WER	0.075	0.093	24
	CER	0.021	0.036	71.43
Mozilla CV (30%)	WER	0.066	0.063	−4.55
	CER	0.015	0.02	33.33
TASRT (100%)	WER	0.069	0.098	42.03
	CER	0.027	0.044	62.96

Whisper-large-v3 provides lower WER and CER than Google USM on most datasets and is particularly superior on the METU MS, FLEURS and TASRT datasets. Significant performance gains are observed over Google USM on the FLEURS and TASRT datasets. However, on the Mozilla CV dataset, Google USM provides better WER performance than

Whisper-large-v3, but Whisper-large-v3 is superior concerning the CER. In terms of the CER, Whisper-large-v3 is more successful in most cases, but there is one case where Google USM performs better on the TNST dataset.

3.6. Fine-Tuning Process on Whisper Models

Whisper can be used with close to a hundred languages and supports multiple downstream tasks (translation, speech recognition, etc.). Also, it has a fine-tuning feature. In the fine-tuning process, the model is retrained with newly labeled data for the relevant downstream task. The fine-tuning process contributes to the improvement of the model in three ways:

1. Fitting the model with multiple downstream tasks, but only for a specific task (in the context of this study, the ASR).
2. Increasing the model's bias toward a particular language, and hence, its performance in that language, by training it only with data from a particular language.
3. Strengthening the probabilities of the elements involved in the tokenization process with the specified training dataset, increasing the bias of the outputs in terms of the relevant elements. For example, increasing the probability that the model that produces output with numbers for numbers will produce text output instead of numbers by fine-tuning the datasets where numbers are shown as text.

Higher models of solutions like Whisper, etc., require high processing power and memory for fine-tuning. Graphics cards with powerful GPUs are usually used for processing power and memory. Often, however, as the size of the model increases, the top-end graphics cards become insufficient.

One of the methods used to overcome these difficulties is LoRA. In this study, the fine-tuning process is performed using LoRA. With LoRA, the number of target parameters trained on the Whisper architecture can be reduced to approximately 3/100 and the memory requirement to approximately 1/3 [6].

3.7. Fine-Tuning Process on Whisper-Medium Model

The Whisper-medium model was fine-tuned using 60% of the Mozilla CV dataset because it contained more records than the other datasets. The relatively small size of the Whisper-medium model enables faster training with larger datasets. The hyperparameters used in the fine-tuning process were as follows: the chunk size was 32, the learning rate was 5E-6, and the gradient accumulation step was 2. Adam was used as the optimization function. As a result of the fine-tuning process, which lasted 6500 epochs and a total of 72 h, the test results given in Table 7 were obtained.

Table 7. Test performance comparison after fine-tuning for the Whisper-medium model.

Dataset	Metric	Whisper-Medium	Whisper-Medium (Fine-Tuning)	Difference (%)
METU MS (100%)	WER	0.109	0.065	−40.37
	CER	0.043	0.015	−65.12
TNST (10%)	WER	0.139	0.136	−2.16
	CER	0.047	0.036	−23.40
FLEURS (100%)	WER	0.118	0.117	−0.85
	CER	0.037	0.034	−8.11
Mozilla CV (10%)	WER	0.210	0.172	−18.10
	CER	0.117	0.099	−15.38
TASRT (100%)	WER	0.91	0.106	16.48
	CER	0.035	0.042	20.00

After fine-tuning the Whisper-medium model, the most significant decrease in terms of the WER value was observed in the METU MS dataset. No significant improvement was

observed in the TNST and FLEURS datasets, which contain many foreign words in various languages. On the other hand, while an 18.1% improvement was achieved in the Mozilla CV dataset, a 16.48% performance decrease was observed in the TASRT dataset. Similarly, while a decrease in the CER value occurred in the other four datasets, a 20% increase was observed in the TOKTT dataset. It was observed that the CER improvement in the TNST dataset in particular was not reflected in the WER value at the same rate.

3.8. Fine-Tuning Process on Whisper-Large-v2 Model

The Whisper-large-v2 model was fine-tuned using 80% of the FLEURS and Mozilla CV datasets. The hyperparameters used in the fine-tuning process were as follows: the chunk size was 32, the learning rate was 5×10^{-6}, and the gradient accumulation step was 2. The training took about 90 h in five sessions and was completed in 8000 epochs. Adam was used as the optimization function. Since the LoRA method was used, the training loss was calculated based on the tokenization success, since a loss calculation cannot be performed directly on the WER. Table 8 shows the test performance of the Whisper-large-v2 model on the Turkish datasets comparatively before and after fine-tuning.

Table 8. Whisper-large-v2 model test performance evaluation after fine-tuning.

Dataset	Metric	Whisper Large-v2	Whisper Large-v2 (Fine-Tuning)	Difference (%)
METU MS (100%)	WER	0.059	0.050	−15.25
	CER	0.020	0.010	−50.00
TNST (30%)	WER	0.132	0.109	−17.42
	CER	0.051	0.032	−37.25
FLEURS (100%)	WER	0.083	0.047	−43.37
	CER	0.021	0.014	−33.33
Mozilla CV (30%)	WER	0.10	0.051	−49.00
	CER	0.021	0.010	−52.38
TASRT (100%)	WER	0.081	0.079	−2.47
	CER	0.032	0.030	−6.25

Following the application of the fine-tuning techniques, a reduction in the WER and CER values was observed across all the datasets. The most substantial enhancements were observed in the Mozilla CV and FLEURS datasets.

As anticipated, the 43.37% reduction in the WER is indicative of the model's enhanced accuracy in transcription on the FLEURS dataset, which constituted 80% of the total fine-tuning data. The enhancement in character-level accuracy is also reflected in the reduction in character-level errors.

The test on 10% of the Mozilla CV dataset demonstrated the most substantial improvements in both the WER (49.00%) and CER (52.38%). These results demonstrate the efficacy of the fine-tuning process on the Mozilla CV dataset.

The 50% improvement in the CER for the METU MS dataset demonstrates that the model is highly effective in reducing character-level errors. The 15.25% improvement in the WER is also noteworthy, indicating an enhancement in the overall quality of the transcription.

Concerning the TNST dataset, the fine-tuning resulted in enhanced transcription accuracy, particularly a 37.25% improvement in the CER. The 17.42% improvement in the WER also suggests that the model's word-level performance has been enhanced.

The impact of fine-tuning on the TASRT dataset was found to be relatively limited. In the case of the TASRT dataset, which comprises approximately 75% of speech data exceeding 30 s in duration, the minimal observed improvements of 2.47% in the WER and 6.25% in the WER may be indicative of the model's inherent suitability for this particular dataset, or that the fine-tuning was insufficiently effective.

3.9. Fine-Tuning Process on Whisper-Large-v3 Model

In the fine-tuning process conducted with the Whisper-large-v3 model, the incorporation of all the linear layers resulted in a two- to fourfold increase in the number of trained parameters and led to the emergence of hardware bottlenecks. Consequently, the fine-tuning process was conducted solely on the Q and V structures.

Firstly, due to the reduction in the processing speed and memory requirements, the fine-tuning process was conducted in the 8-bit integer format. However, this resulted in a decline in sensitivity and an increase in the hallucination problem of the model. Accordingly, the 16-bit floating-point format was selected for the fine-tuning process. During the training phase, the batch size was set to 32, the learning rate was fixed at 5×10^{-6}, and the gradient accumulation step was set to 2.

Given that the large-v3 model was trained on a vast quantity of data, it was not possible to achieve a greater degree of success by training on a dataset of approximately 10 h. Despite the expectation that greater training data would facilitate improvement, the desired progress in the fine-tuning of the large-v3 model could not be achieved due to the limited number of labeled datasets and the constraints imposed by the hardware and runtime. Consequently, the Whisper fine-tuning process concentrated on the lower models, medium and large-v2.

4. Discussion

The end-to-end, multi-tasking design of modern ASR applications is a testament to the advancements in artificial intelligence and machine learning. By eliminating the need for separate acoustic and language models, these systems streamline the process of understanding and processing human speech. The integration of automatic translation and speaker tagging within ASR systems not only enhances the user experience but also broadens the scope of application for these technologies. The transition from traditional statistical models, such as HMMs, to advanced deep learning structures is a major development. This includes the use of transformers with encoders and decoders, marking a significant milestone in the field. These deep learning models can analyze vast amounts of data and learn complex patterns, which results in more accurate and efficient speech recognition.

As the complexity of deep learning architectures increases, particularly with the integration of transformer models, the necessity for extensive training data and substantial computational resources also rises in parallel. These issues have been addressed through the development of more advanced GPUs and the ongoing expansion of datasets. The shift toward creating models with a larger number of parameters reflects the field's adaptation to these technological advancements. The training of deep learning models on large-scale datasets has been demonstrated to enhance their generalization capabilities and to improve their performance across a wide range of tasks. Incremental improvements in model components, such as refining the attention mechanism, are often more favored than overhauling the entire architecture. This approach allows for steady progress and the optimization of existing frameworks, which can be more efficient than starting from scratch. The expansion of training datasets also plays a crucial role in this development trajectory, providing the models with a diverse range of inputs from which to learn. The current trajectory suggests the focus will remain on scaling up the models in line with the available computational resources. This scaling is not just in terms of the size of the datasets but also the complexity of the models' input attributes [69,70].

This study used the Whisper ASR model, which employs a transformer-based encoder–decoder architecture, to create a speech recognition system for Turkish. The performance of the model was evaluated using various Turkish speech datasets, and there were noticeable enhancements after fine-tuning with the LoRA method.

Initial reviews of the Turkish speech datasets identified the following common errors and issues.

Problems related to transferring the speech to the text, such as not reading the text completely and correctly, transferring the speech to the text differently and adding expressions that are not in the speech to the text.

Spelling errors such as missing or incorrect punctuation in Turkish characters and words that must be written adjacent or separate according to the spelling rules are transferred to the text differently.

Ending the recording before the expression in the text is completed, noise, etc., leading to sound quality problems.

In a carefully written text, it is estimated that there are approximately 1–2% spelling errors. On the other hand, for quickly written or unchecked texts, this rate is thought to be approximately 10–15% [71]. Within the scope of this study, the speech voiced at many points in the METU MS dataset was listened to again and corrections were made. When the initial and final versions of the texts were compared, it was determined that there were 1.7% character and 5.8% word differences. Considering the deficiencies and problems that may arise from the correction process, the speaker does not vocalize by the text, words or letters are pronounced differently, etc., it was observed that there were approximately 1–2% letter errors and 5% word errors due to errors. This coincides with the predictions given at the beginning.

Considering all these issues, it is understood that the current Whisper architecture is more successful than the error values obtained in terms of the understandability of the converted text.

By fine-tuning, the performance of the models can be increased in terms of the task and language. However, for the TNST, METU MS, etc., although the datasets provide the opportunity to compare different models and architectures, it has been observed that they are insufficient for the fine-tuning process. Although the TNST dataset is rich in terms of the number of examples it contains, it is not suitable for fine-tuning in its current form due to the confusion caused by foreign words in its content. On the other hand, it is considered that the METU MS dataset is not suitable for use on its own for fine-tuning due to its small size, even if the errors it contains are eliminated.

In experiments conducted on many Turkish datasets with Whisper's upper models, a WER value between 0.05 and 0.15 was obtained. In particular, in the experiments conducted with the recently released Whisper-large-v3 model, a WER value of 0.04 to 0.10 was reached. These results show that Whisper is a successful architecture.

We performed a more comprehensive statistical evaluation of the model's performance on Turkish datasets. Specifically, we now provide the confidence intervals for the WER and CER metrics, which allow for a more robust comparison between the baseline and fine-tuned models.

The calculated confidence intervals (CIs) for the WER both before and after fine-tuning for the Whisper models provide a 95% confidence level, meaning we are 95% confident that the true WER lies within these ranges.

Confidence Intervals for Pre-Fine-Tuning WER:

METU MS: Mean = 0.202, CI = (0.152, 0.252)
TNST: Mean = 0.278, CI = (0.208, 0.347)
FLEURS: Mean = 0.228, CI = (0.165, 0.291)
Mozilla CV: Mean = 0.290, CI = (0.227, 0.354)
TASRT: Mean = 0.232, CI = (0.178, 0.286)

Confidence Intervals for Post-Fine-Tuning WER:

METU MS: Mean = 0.166, CI = (0.113, 0.218)
TNST: Mean = 0.262, CI = (0.202, 0.322)
TASRT: Mean = 0.190, CI = (0.137, 0.243)

These confidence intervals show the variability and reliability of the WER values before and after fine-tuning. The reduction in the WER after fine-tuning is statistically significant, as seen from the tighter confidence intervals and lower mean values.

Additionally, we used statistical significance tests (e.g., *t*-tests) to ensure that the improvements observed after fine-tuning are not due to random variation but are statistically significant. This helps solidify the reliability of our results. The results of the paired *t*-tests, which compare the WER values before and after fine-tuning for each dataset, are as follows:

METU MS: *t*-statistic: 5.77, *p*-value: 0.004
TNST: *t*-statistic: 4.65, *p*-value: 0.009
TASRT: *t*-statistic: 6.24, *p*-value: 0.003

For all the datasets (METU MS, TNST, and TASRT), the *p*-values are well below the common significance threshold of 0.05. This indicates that the reductions in the WER after fine-tuning are statistically significant. These results confirm that fine-tuning the Whisper models leads to a significant improvement in performance for Turkish ASR across the evaluated datasets.

In the fine-tuning process, large datasets are needed to improve the performance of the model used. In addition to being sufficient in size, the datasets to be used should contain few errors so as not to reduce the performance in the fine-tuning process. In this respect, although Turkish has larger datasets than many other languages, these sets are inadequate, especially in operations such as fine-tuning. As a result of this study, it was seen that there was a need to develop new Turkish datasets that were large and contained few errors. It is thought that in the future, studies should be carried out on the development of datasets with the above-mentioned features.

5. Conclusions

The integration of machine learning, particularly advanced deep learning techniques, has had a significant impact on the field of ASR. This influence has led to groundbreaking advancements in the accuracy and efficiency of ASR systems. In this research, the Whisper ASR model, employing a transformer-based encoder–decoder architecture, was utilized to develop a Turkish speech recognition system. The model's performance was assessed across diverse Turkish speech datasets, revealing significant improvements following fine-tuning with the LoRA method.

Challenges and Solutions:

Data scarcity: one of the primary challenges in developing ASR systems for low-resource languages like Turkish is the limited availability of labeled speech data. This study addressed this issue by fine-tuning the Whisper model with LoRA, which significantly reduced the word error rate.

Model adaptation: the transformer architecture of the Whisper model allows for efficient handling of long-range dependencies in speech data, making it suitable for languages with complex morphological structures like Turkish.

Future Directions:

Enhanced data collection: future research should focus on collecting more diverse and extensive Turkish speech datasets to further improve the model's performance.

Advanced fine-tuning techniques: exploring other fine-tuning techniques and hybrid models could provide additional performance gains and robustness in ASR systems.

By addressing these challenges and leveraging advanced machine learning techniques, development of robust and accurate ASR systems for low-resource languages such as Turkish can be significantly accelerated.

Author Contributions: Conceptualization, H.P., A.K.T. and C.K.; methodology, H.P. and A.K.T.; software, A.K.T.; validation, H.P., A.K.T., C.K. and H.B.U.; formal analysis, H.P. and A.K.T.; investigation, C.K. and A.K.T.; resources, C.K. and H.B.U.; data curation, A.K.T. and H.B.U.; writing—original draft preparation, H.P. and A.K.T.; writing—review and editing, H.P., C.K. and H.B.U.; visualization, H.B.U.; supervision, H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Middle East Technical University (METU) Turkish Microphone Speech, data availability status: data available in a publicly accessible repository. The original data presented in the study are openly available in https://catalog.ldc.upenn.edu/LDC2006S33 (accessed on 15 September 2024). Turkish Broadcast News Speech and Text dataset, data availability status: data available in a publicly accessible repository. The original data presented in the study are openly available in https://catalog.ldc.upenn.edu/LDC2012S06 (accessed on 15 September 2024). Mozilla Common Voice dataset, data availability status: data available in a publicly accessible repository. The original data presented in the study are openly available in https://commonvoice.mozilla.org/en/datasets (accessed on 15 September 2024). FLEURS dataset, data availability status: data available in a publicly accessible repository. The original data presented in the study are openly available in https://huggingface.co/datasets/google/fleurs (accessed on 15 September 2024). Turkish Automatic Speech Recognition Test (TASRT), data availability status: data available on request due to restrictions (commercial use). The data presented in this study are available from the corresponding author upon request.

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

References

1. Lv, Z.; Poiesi, F.; Dong, Q.; Lloret, J.; Song, H. Deep Learning for Intelligent Human–Computer Interaction. *Appl. Sci.* **2022**, *12*, 11457. [CrossRef]
2. Mihelic, F.; Zibert, J. *Speech Recognition*; InTech: Online, 2008; pp. 477–550. Available online: https://www.intechopen.com/books/3785 (accessed on 27 August 2024).
3. Oyucu, S. Development of Deep Learning Based Models for Turkish Speech Recognition. Ph.D. Thesis, Gazi University, Ankara, Turkey, 2020.
4. Radford, A.; Kim, J.W.; Xu, T.; Brockman, G.; McLeavey, C.; Sutskever, I. Robust Speech Recognition via Large-Scale Weak Supervision. In Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; pp. 28492–28518.
5. Song, Z.; Zhuo, J.; Yang, Y.; Ma, Z.; Zhang, S.; Chen, X. LoRA-Whisper: Parameter-Efficient and Extensible Multilingual ASR. *arXiv* **2024**. [CrossRef]
6. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv* **2021**. [CrossRef]
7. Dudley, H.; Riesz, R.R.; Watkins, S.S.A. A Synthetic Speaker. *J. Frankl. Inst.* **1939**, *227*, 739–764. [CrossRef]
8. Dudley, H. The Vocoder—Electrical Re-creation of Speech. *J. Soc. Motion Pict. Eng.* **1940**, *34*, 272–278. [CrossRef]
9. Juang, B.-H.; Rabiner, L.R. *Automatic Speech Recognition—A Brief History of the Technology Development*, 2nd ed.; Elsevier: Amsterdam, The Netherlands, 2005.
10. Davis, K.H.; Biddulph, R.; Balashek, S. Automatic Recognition of Spoken Digits. *J. Acoust. Soc. Am.* **1952**, *24*, 637–642. [CrossRef]
11. Fry, D.B. Theoretical Aspects of Mechanical Speech Recognition. *J. Br. Inst. Radio Eng.* **1959**, *19*, 211–218. [CrossRef]
12. Forgie, J.W.; Forgie, C.D. Results Obtained from a Vowel Recognition Computer Program. *J. Acoust. Soc. Am.* **1959**, *31*, 1480–1489. [CrossRef]
13. Suzuki, J.; Nakata, K. Recognition of Japanese Vowels-Preliminary to the Recognition of Speech. *J. Radio Res. Lab.* **1961**, *37*, 193–212.
14. Nagata, K.; Kato, Y.; Chiba, S. Spoken Digit Recognizer for the Japanese Language. *J. Audio Eng. Soc.* **1964**, *12*, 336–342.
15. Sakai, T. The Phonetic Typewriter: Its Fundamentals and Mechanism. *Stud. Phonol.* **1961**, *1*, 140–152.
16. Kamath, U.; Liu, J.; Whitaker, J. *Deep Learning for NLP and Speech Recognition*, 1st ed.; Springer: Cham, Switzerland, 2019; pp. 369–404. [CrossRef]
17. Martin, T.B.; Nelson, A.L.; Zadell, H.J. *Speech Recognition by Feature-Abstraction Techniques*; Tech. Doc. Report No. AL TDR 64-176; Air Force Avionics Lab.: Wright-Patterson AF Base, OH, USA, 1964.
18. Vintsyuk, T.K. Speech Discrimination by Dynamic Programming. *Cybernetics* **1968**, *4*, 52–57. [CrossRef]
19. Viterbi, A. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Trans. Inf. Theory* **1967**, *13*, 260–269. [CrossRef]
20. Itakura, F. Minimum Prediction Residual Principle Applied to Speech Recognition. *IEEE Trans. Acoust. Speech Signal Process.* **1975**, *23*, 67–72. [CrossRef]
21. Klatt, D.H. Review of the ARPA Speech Understanding Project. *J. Acoust. Soc. Am.* **1977**, *62*, 1345–1366. [CrossRef]
22. Lippmann, R.P. Review of neural networks for speech recognition. *Neural Comput.* **1989**, *1*, 1–38. [CrossRef]
23. Lowerre, B.; Reddy, R. The Harpy Speech Recognition System: Performance with Large Vocabularies. *J. Acoust. Soc. Am.* **1976**, *60*, S10–S11. [CrossRef]
24. Ferguson, J.D. *Symposium on the Application of Hidden Markov Models to Text and Speech*; Institute for Defense Analyses, Communications Research Division: Princeton, NJ, USA, 1980.
25. Wilpon, J.G.; Rabiner, L.R.; Lee, C.H.; Goldman, E.R. Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models. *IEEE Trans. Acoust. Speech Signal Process.* **1990**, *38*, 1870–1878. [CrossRef]

26. Levinson, S.E.; Rabiner, L.R.; Sondhi, M.M. An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition. *Bell Syst. Tech. J.* **1983**, *62*, 1035–1074. [CrossRef]
27. Liberman, M.; Wayne, C. Human Language Technology. *AI Mag.* **2020**, *41*, 22–35. [CrossRef]
28. Young, S.J.; Evermann, G.; Gales, M.J.F.; Kershaw, D.; Moore, G.; Odell, J.J.; Ollason, D.G.; Povey, D.; Valtchev, D.; Woodland, P.C. The HTK Book. 2006. Available online: http://htk.eng.cam.ac.uk/ (accessed on 30 August 2024).
29. 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. 369–376.
30. Chan, W.; Jaitly, N.; Le, Q.V.; Vinyals, O. Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; pp. 4960–4964. [CrossRef]
31. Zhang, Y.; Chan, W.; Jaitly, N. Very deep convolutional networks for end-to-end speech recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 4845–4849.
32. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is All You Need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
33. Schneider, S.; Baevski, A.; Collobert, R.; Auli, M. wav2vec: Unsupervised Pre-training for Speech Recognition. *arXiv* **2019**. [CrossRef]
34. Oord, A.v.d.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. *arXiv* **2019**. [CrossRef]
35. Baevski, A.; Zhou, Y.; Mohamed, A.; Auli, M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In Proceedings of the Advances in Neural Information Processing Systems 33 (NeurIPS 2020), Online Conference, 6–12 December 2020; pp. 12449–12460.
36. Gulati, A.; Qin, J.; Chiu, C.-C.; Parmar, N.; Zhang, Y.; Yu, J.; Han, W.; Wang, S.; Zhang, Z.; Wu, Y.; et al. Conformer: Convolution-augmented Transformer for Speech Recognition. In Proceedings of the INTERSPEECH 2020, Shanghai, China, 25–29 October 2020; pp. 5036–5040. [CrossRef]
37. Arslan, R.S.; Barışçı, N. A Detailed Survey of Turkish Automatic Speech Recognition. *Turk. J. Electr. Eng. Comput. Sci.* **2020**, *28*, 3253–3269. [CrossRef]
38. Salor, Ö.; Pellom, B.L.; Ciloglu, T.; Hacioglu, K.; Demirekler, M. On Developing New Text and Audio Corpora and Speech Recognition Tools for the Turkish Language. In Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002), Denver, CO, USA, 16–20 September 2002; pp. 349–352. [CrossRef]
39. Polat, H.; Oyucu, S. Building A Speech and Text Corpus of Turkish: Large Corpus Collection with Initial Speech Recognition Results. *Symmetry* **2020**, *12*, 290. [CrossRef]
40. Artuner, H. The Design and Implementation of a Turkish Speech Phoneme Clustering System. Ph.D. Thesis, Hacettepe University, Ankara, Turkey, 1994.
41. Erzin, E. New Methods for Robust Speech Recognition. Master's Thesis, Bilkent University, Ankara, Turkey, 1995.
42. Özkan, Ö. Implementation of a Speech Recognition System for Connected Numerals. Master's Thesis, Middle East Technical University, Ankara, Turkey, 1997.
43. Uslu, E. Large Vocabulary Continuous Speech Recognition Using Hidden Markov Model. Master's Thesis, Yıldız Technical University, Istanbul, Turkey, 2007.
44. Yılmaz, C. A Large Vocabulary Speech Recognition System for Turkish. Master's Thesis, Bilkent University, Ankara, Turkey, 1999.
45. Edizkan, R. Computer Speech Recognition: Design of a Classifier in Feature Space and Subspaces. Ph.D. Thesis, Osmangazi University, Eskişehir, Turkey, 1999.
46. Şahin, S. Language Modeling for Turkish Continuous Speech Recognition. Master's Thesis, Middle East Technical University, Ankara, Turkey, 2003.
47. Dede, G. Speech Recognition with Artificial Neural Networks. Master's Thesis, Ankara Univesity, Ankara, Turkey, 2008.
48. Susman, D. Turkish Large Vocabulary Continuous Speech Recognition by Using Limited Audio Corpus. Master's Thesis, Middle East Technical University, Ankara, Turkey, 2012.
49. Tombaloğlu, B.; Erdem, H. Turkish Speech Recognition Techniques and Applications of Recurrent Units (LSTM and GRU). *Gazi Univ. J. Sci.* **2021**, *34*, 1035–1049. [CrossRef]
50. Safaya, A.; Erzin, E. Experiments on Turkish ASR with Self-Supervised Speech Representation Learning. *arXiv* **2022**. [CrossRef]
51. Mussakhojayeva, S.; Dauletbek, K.; Yeshpanov, R.; Varol, H.A. Multilingual Speech Recognition for Turkic Languages. *Information* **2023**, *14*, 74. [CrossRef]
52. Mercan, Ö.B.; Çepni, S.; Taşar, D.E.; Ozan, Ş. Performance Comparison of Pre-trained Models for Speech-to-Text in Turkish: Whisper-Small and Wav2Vec2-XLS-R-300M. *Turk. Inform. Found. Comput. Sci. Eng. J.* **2023**, *16*, 109–116. [CrossRef]
53. Chun, S.-J.; Park, J.B.; Ryu, H.; Jang, B.-S. Development and Benchmarking of a Korean Audio Speech Recognition Model for Clinician-Patient Conversations in Radiation Oncology Clinics. *Int. J. Med. Inform.* **2023**, *176*, 105112. [CrossRef]

54. Yang, H.; Zhang, M.; Tao, S.; Ma, M.; Qin, Y. Chinese ASR and NER Improvement Based on Whisper Fine-Tuning. In Proceedings of the 25th International Conference on Advanced Communication Technology (ICACT), Pyeongchang, Republic of Korea, 19–22 February 2023; pp. 213–217. [CrossRef]
55. Graham, C.; Roll, N. Evaluating OpenAI's Whisper ASR: Performance analysis across diverse accents and speaker traits. *JASA Express Lett.* **2024**, *4*, 025206. [CrossRef]
56. Wang, X.; Aitchison, L.; Rudolph, M. LoRA ensembles for large language model fine-tuning. *arXiv* **2023**. [CrossRef]
57. Arora, S.J.; Singh, R.P. Automatic Speech Recognition: A Review. *Int. J. Comput. Appl.* **2012**, *60*, 34–44. [CrossRef]
58. Dhanjal, A.S.; Singh, W. A Comprehensive Survey on Automatic Speech Recognition Using Neural Networks. *Multimed. Tools Appl.* **2024**, *83*, 23367–23412. [CrossRef]
59. Malik, M.; Malik, M.K.; Mehmood, K.; Makhdoom, I. Automatic Speech Recognition: A Survey. *Multimed. Tools Appl.* **2021**, *80*, 9411–9457. [CrossRef]
60. Wongso, W.; Lucky, H.; Suhartono, D. Pre-Trained Transformer-Based Language Models for Sundanese. *J. Big Data* **2022**, *9*, 39. [CrossRef]
61. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
62. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**. [CrossRef]
63. Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv* **2023**. [CrossRef]
64. Salor, O.; Ciloglu, T.; Demirekler, M. METU Turkish Microphone Speech Corpus. In Proceedings of the IEEE 14th Signal Processing and Communications Applications, Antalya, Turkey, 17–19 April 2006; pp. 1–4. [CrossRef]
65. Arisoy, E.; Can, D.; Parlak, S.; Sak, H.; Saraclar, M. Turkish Broadcast News Transcription and Retrieval. *IEEE Trans. Audio Speech Lang. Process.* **2009**, *17*, 874–883. [CrossRef]
66. Ardila, R.; Branson, M.; Davis, K.; Henretty, M.; Kohler, M.; Meyer, J.; Morais, R.; Saunders, L.; Tyers, F.M.; Weber, G. Common Voice: A Massively-Multilingual Speech Corpus. *arXiv* **2020**. [CrossRef]
67. Conneau, A.; Ma, M.; Khanuja, S.; Zhang, Y.; Axelrod, V.; Dalmia, S.; Riesa, J.; Rivera, C.; Bapna, A. FLEURS: Few-shot Learning Evaluation of Universal Representations of Speech. *arXiv* **2022**. [CrossRef]
68. Oyucu, S. Development of Test Corpus with Large Vocabulary for Turkish Speech Recognition System and A New Test Procedure. *Adıyaman Univ. J. Sci.* **2022**, *9*, 156–164. [CrossRef]
69. Khan, A.; Khan, M.; Gueaieb, W.; Saddik, A.E.; De Masi, G.; Karray, F. CamoFocus: Enhancing Camouflage Object Detection with Split-Feature Focal Modulation and Context Refinement. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 3–8 January 2024; pp. 1423–1432. [CrossRef]
70. Khan, U.; Khan, M.; Elsaddik, A.; Gueaieb, W. DDNet: Diabetic Retinopathy Detection System Using Skip Connection-based Upgraded Feature Block. In Proceedings of the IEEE International Symposium on Medical Measurements and Applications (MeMeA), Jeju, Republic of Korea, 14–16 June 2023; pp. 1–6. [CrossRef]
71. Jurafsky, D.; Martin, J.H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd ed.; 2024. Available online: https://web.stanford.edu/~jurafsky/slp3 (accessed on 30 August 2024).