Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University

Altynbek Amirzhanov

Automatic Speech Recognition for the Kazakh Language

THESIS

Presented in Partial Fulfillment for the
Degree of Master of Science in Computer Science
(degree code: 7M061002)
Department of Computer Sciences
Faculty of Engineering and Natural Sciences

Supervisor: **Ph.D. Darkhan Kuanyshbay**

Kaskelen, 2022

Suleyman Demirel University

Faculty of Engineering and Natural Sciences

Department of Computer Science

Dean of Faculty

Associate Professor, PhD Zhamanov A.

«31» 05 2022

Topic of the thesis:

Automatic Speech Recognition for the Kazakh Language

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science", SDU, 2020-2022

Head of Department Associate professor PhD Cemil Turan

Academic Supervisor PhD Darkhan Kuanyshbay

Master's student Altynbek Amirzhanov

Kaskelen, 2022

Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University

Altynbek Amirzhanov

Automatic Speech Recognition for the Kazakh Language

THESIS

Presented in Partial Fulfillment for the
Degree of Master of Science in Computer Science
(degree code: 7M061002)
Department of Computer Sciences
Faculty of Engineering and Natural Sciences

Supervisor: **Ph.D. Darkhan Kuanyshbay**

Kaskelen, 2022

Abstract

In the presented study, the problem of automatic speech recognition (ASR) for the Kazakh language was discussed. It starts with general information about the field such as the history of ASR, invented algorithms, models. Then the state-of-the-art architectures in the world will be studied. The following is a specific topic of implemented automatic speech recognition systems for the Kazakh language. Studying the works related to the topic of the thesis, an end-to-end model based on the Connection Temporal Classifier (CTC) algorithm was chosen to conduct experiments with various types of optimizers. The results showed that the proposed method works better with the Momentum optimizer than others such as Adam and Adagrad.

Аңдатпа

Ұсынылған зерттеуде қазақ тілі үшін автоматты түрде сөйлеуді тану (ASR) мәселесі талқыланды. Ол ASR тарихы, ойлап табылған алгоритмдер, модельдер сияқты сала туралы жалпы мәліметтерден басталады. Содан кейін әлемдегі ең заманауи архитектуралар зерттеледі. Сосын қазақ тіліне арналған сөзді автоматты түрде тану жүйесіне жасалған нақты зерттеулер беріледі. Диссертация тақырыбына қатысты жұмыстарды зерделей отырып, әр түрлі типтегі оптимизаторлармен тәжірибелер жүргізу үшін Connection Temporal Classifier (CTC) алгоритміне негізделген end-to-end моделі таңдалды. Нәтижелер ұсынылған әдістің Momentum оңтайландырғышымен Адам және Адаград сияқты басқа оңтайландырғыштарға қарағанда жақсы жұмыс істейтінін көрсетті.

Аннотация

В представленном исследовании обсуждалась проблема автоматического распознавания речи (ACP) для казахского языка. Он начинается с общей информации об области, такой как история ASR, изобретенные алгоритмы, модели. Затем будут изучены самые современные архитектуры в мире. Ниже приводится конкретная тема внедренных систем автоматического распознавания речи для казахского языка. Изучая работы, относящиеся к теме диссертации, для проведения экспериментов с различными типами оптимизаторов была выбрана сквозная модель на основе алгоритма Connection Temporal Classifier (CTC). Результаты показали, что предложенный метод лучше работает с оптимизатором Momentum, чем с другими, такими как Adam и Adagrad.

Acknowledgements

Thanks to my academic supervisor for the constant support and useful discussions. Thanks to the coordinator of the Computer Science program, Mr. Sadyk, who reminded us of the important dates for the submission of documents. Thanks to everyone who helped me become a good researcher and a good person after all.

To my besties

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Aims and Objectives 2

1.3 Thesis Outline 3

2 Background **4**

2.1 History of ASR 4

2.2 ASR models 5

2.2.1 Hidden Markov Models 6

2.2.2 Connection Temporal Classification 7

2.3 Deep Neural Networks 8

2.3.1 Convolutional Neural Networks 9

2.3.2 Self Attention 10

2.4 Language Modelings 11

2.4.1 N-Gram Language Model 12

2.4.2 Transformer Language Modeling 13

3 Related Works **14**

4 About Dataset **25**

4.1 Collection and Cleaning of text 26

4.2 Narration and Checking the text 26

4.3 Specifications of database 27

5 End-to-end ASR systems **30**

5.1 A model based on CTC 32

5.1.1 The main idea of the CTC 32

5.1.2 The estimation of path probability . . . 33
5.1.3 Aggregation of path . . . 34
5.1.4 Related works . . . 35
5.1.5 A large data training . . . 36
5.2 An end-to-end RNN-transducer model . . . 37
5.2.1 The main idea of the RNN-transducer . . . 37
5.2.2 Related works . . . 40
5.3 A model based on Attention . . . 40
5.3.1 The information and delay redundancy . . . 41
5.3.2 The network structure . . . 43
5.3.3 Related works . . . 43
5.3.4 Continuity problem . . . 44
5.3.5 Monotonicity problem . . . 45
5.3.6 The issue of extracting key information . . . 45
5.3.7 The delay . . . 47
5.3.8 The decoder . . . 49
5.3.9 Summary . . . 50

6 Experiments & Results **52**
6.1 Experiments . . . 52
6.2 Results . . . 53
6.2.1 Adagrad optimizer . . . 53
6.2.2 Momentum optimizer . . . 54
6.2.3 Adam optimizer . . . 54

7 Conclusion & Future Work **55**

References **56**

1. Introduction

1.1 Motivation

Human interaction happens in a variety of ways such as mimics of facial expressions, gestures of hand or other body parts, text messages, waves of speech sounds, etc. A special one of them among above mentioned is a speech as it might facilitate the communication between people and make it faster to deliver the message you want.

Speech is a useful statement with a definite meaning that is composed of numerous words, each having various letters and voices. This voice may move as waves across air, empty space, and inanimate objects; a wave that overlaps between them or begins as little rings of the sound source. This condition is distinguished by force, which gradually expands the circles until they vanish entirely when dispersed across large distances. Speakers who speak the same language produce logical communication, suggesting that the sender and receiver share the same set of keys for interpreting the message.

The researchers took advantage of this phenomenon and developed it into a critical branch of human-machine communication, in which sound has assisted in the user's use of the computer and the production of genuine dialogue between them. Automatic voice recognition has benefited the development of artificial intelligence, which promises to give very flexible ways of managing machines by allowing users to speak and share information without the need of standard input/output modules such as the keyboard. Voice-based input/output systems are useful in a number of scenarios, including as the care of disabled people, the operation of cars, particularly while driving, distress calls, and so on.

To perform well, all of these distinct domains that employ automated speech recognition require large amounts of data. The number one challenge that has to

be solved is a data problem, because deep neural networks perform much better than medium neural networks because to recent advances in graphical processing unit (GPU) performance.

It should be noted that automatic speech recognition systems are almost never employed in Kazakhstan, necessitating extensive research and analysis by developers. The absence of speech data is the fundamental cause for the lack of inquiry and study in the field of voice recognition for the Kazakh language. High-quality ASR systems require a vast quantity of data, as has been found in common languages such as English, Spanish, and Chinese. Popular speech corpora, such as TIMIT and Switchboard, include massive volumes of transcribed audio recordings of many types of presentations, including phone talks, conversational speeches, and clean microphone speeches. On the internet, there are virtually no acceptable speech corpora in Kazakh. The ones that are available are usually not free to use and are insufficient for developing powerful and efficient ASR models. A decent speech corpus for an ASR system demands time, a well-structured environment, and a reliable monitoring mechanism. However, in order to completely overcome the issue of data scarcity, the neural network structure and methods must be addressed as well.

1.2 Aims and Objectives

There are a few goals to reach in the scope of this work:

- to learn the history of the automatic speech recognition;
- to explore the algorithms invented in the chosen area;
- to investigate the current state of the area in the world;
- to make research about the implemented models/architectures for the Kazakh language;
- to conduct experiments with an open-source dataset and publicly available automatic speech recognition model code.

It is believed that performing above mentioned aims will help to dive deep into the chosen area and deep learning field.

1.3 Thesis Outline

The Introduction chapter will give an overview of the area of automatic speech recognition and its current state in the Kazakh language. In Chapter 2 a history and existing models of ASR will be presented. Chapter 3 will introduce related works. There will be written detailed information about the used speech corpus in the Chapter 4. Chapter 5 will be about end-to-end ASR models and 6 will be about experiments conducted and results of them. And in Conclusion & Future Work chapter all the findings will be concluded and future work will be discussed.

2. Background

The basics of this thesis, as well as related work, will be discussed in this chapter.

As a result, Section 2.1 begins with a brief history of how ASR has evolved in the past. ASR models will be discussed in length in Section 2.2: Hidden markov models and the connection temporal classification algorithm, which enabled today's end-to-end models of ASR. In the section 2.3, neural networks will be gone through. The next section 2.4 discusses contemporary general language modelings used to enhance the ASR modelings presented in 2.2. Related works will be discussed in the next chapter.

2.1 History of ASR

Audrey, a "digit recognizer" established by Bell Laboratories researchers in 1952, was one of the first initiatives. Audrey was able to distinguish spoken numerical digits by searching for auditory fingerprints known as formants, which are the distilled essences of sounds.

IBM created Shoebox in the 1960s, a system that could detect digits and arithmetic commands such as "plus" and "total." Shoebox could even hand the math problem on to an adding machine, which would calculate and display the answer.

Meanwhile, Japanese researchers developed technology that could distinguish speech constituents such as vowels, and other systems could analyse speech structure to determine where a word may stop. By studying phonemes, or discrete sounds in a language, a team from University College in England was able to distinguish four vowels and nine consonants.

The US Department of Defense's ARPA (now known as DARPA) supported a five-year effort named Speech Understanding Research in the early 1970s. As a

result, numerous new ASR systems were developed, the most successful of which was Carnegie Mellon University's Harpy, which by 1976 could recognize just over 1000 words.

The popularization of Hidden Markov Models (HMMs) in the mid-1980s was a watershed moment. This method constituted a substantial transition "from simple pattern recognition approaches based on templates and a spectral distance measure to a statistical strategy for voice processing," resulting in a significant improvement in accuracy.

For better or worse, the 1990s saw the introduction of automatic speech recognition in the form we know today. Dragon Dictate was released in 1990 for a whopping $9,000, boasting an 80,000-word lexicon and natural language processing capabilities. Dragon could only recognize 30–40 words per minute at first while people normally speak four times quicker than that.

Others thought neural networks were the key to developing a new sort of ASR. With the introduction of big data, faster computers, and GPU processing, a new ASR method, end-to-end deep learning ASR, was developed. As additional data is given into the neural networks, this new ASR approach might "learn" and be "trained" to become more accurate. There will be no need for developers to re-code each portion of the trigram serial model in order to add additional languages, parse accents, remove noise, or add new words. Another significant benefit of adopting an end-to-end deep learning ASR is that it allows you to achieve high accuracy, speed, and scalability without sacrificing cost.

2.2 ASR models

Automatic Speech Recognition (ASR) is the process of converting spoken language into proper transcription using a computer. In order to use the recorded speech as an input signal, it must first be pre-processed. This is important because the nature and manner of "raw" audio records of speech might vary, making algorithms not easy to identify. After the audio recordings have been transformed into a common format, such as acoustic vectors, they can be utilized to train ASR models with appropriate transcriptions.

The first attempts at automatic speech recognition focused on detecting the numbers 1 through 9. ASR models are far more advanced today, thanks in part

to big data and deep learning. They've become an indispensible aspect of modern living, and voice assistants are possible to find in almost any pouch on majority of smart systems. Simple support jobs and home control to speech-to-speech translation are all possible applications. Despite this, such technologies are still not ideal and have issues, mostly with extremely complex terminology, which will contribute to the field's progress.

2.2.1 Hidden Markov Models

After initial attempts, such as "Audrey," met with limited success, the invention of Gaussian mixture models (GMM) and hidden Markov models (HMM) marked a watershed moment in ASR [23, 27]. Each class (e.g., phonemes) in a GMM/HMM model has an HMM with different hidden states s. Each class has several hidden states to represent differences in speech. If this system is given an auditory entry x, it will attempt to predict the most likely series of concealed variables s, in this instance phonemes.

A HMM/GMM system now tries to calculate the conditions s with the greatest probability given an acoustic input x:

$$\arg\max_{s} P(s \mid x) \tag{2.1}$$

The Bayes rule can be used to rewrite this probability:

$$\arg\max_{s} \frac{P(x \mid s)P(s)}{P(x)} \tag{2.2}$$

In this case, the chance $P(x)$, the likelihood that this remark occurs, may be overlooked in the expansion because it is significantly independent of the condition s. A language model can determine the chance $P(s)$, the likelihood of the phoneme(s). A multidimensional GMM can now be used to train the last remaining probability $P(x \mid s)$.

For decades, these HMM/GMM systems was the gold norm in ASR systems, and they were continuously upgraded by brand-new techniques to enhance model parameter discovery, such as maximum likelihood linear regression [26].

Regrettably, the size of the training datasets has a strong influence on the effectiveness of the GMM/HMM models. The reason for this is because while the

GMM model performs direct distribution of input features, it requires a representation that incorporates all essential information. 100 thousand hours of to trained data could be necessary for a HMM/GMM system to reach an precision of over 95% [33].

2.2.2 Connection Temporal Classification

Many alternative concealed conditions s are required in HMM/GMM systems, as discussed in Section 2.2.1, because the pronunciation of individual letters/phonemes or even words might change greatly amongst talkers.

This is particularly imperative in a temporary setting since it might be difficult to determine how long an input audio series takes to reach which destination. [17] introduced the Connectionist Temporal Classifier (CTC) algorithm to solve this problem. This method attempts to match the entry series $X = [x_1, x_2, ..., x_T]$, such as sound recordings, with the outcome series $Y = [y_1, y_2, ..., y_T]$, such as letters, while keeping temporary pliability. Especially , the series do not need to be of the equal duration, and there is no need for straight assigning. As a result, assigning X_2 to Y_2 does not rule out the possibility of assigning X_3 to Y_2.

As with the HMM algorithm, the best candidate of the Y pattern with the X pattern is sought:

$$\arg\max_{s} P(y \mid x) \tag{2.3}$$

An issue occurs, however, because numerous allocations of Y are feasible. If the phrase "Hello" is anticipated, the probable interpretation is: $[h, h, e, e, l, l, o, o]$. To overcome that, repetitive characters must be eliminated, resulting in the term $Helo$. To remedy that, a novel null symbol ϵ is developed to distinguish between repeatitive characters. A fresh prediction would result in the next outcome: $[h, h, e, e, l, l, \epsilon, l, o, o]$.

It may be represented by selecting utmost probable symbols a for every iteration interval t, yielding the formula:

$$\arg\max_{A} \prod_{t=1}^{T} P_t(a_t \mid x) \tag{2.4}$$

Whenever each repeated characters and null symbols in A are deleted, this results in an accurate transcription of $Y =$ "$Hello$".

2.3 Deep Neural Networks

HMM/GMM systems formed the foundation of majority ASR structures for decades. It got different by continuously rising computer capacity, particularly in the graphical realm, which enabled a huge number of computations to be performed in parallel. When computer capacity was utilised for deep neural (DNN) networks in the domain of machine learning, it became significant for the ASR domain [21].

A DNN is a network consisting of many neurocytes. Such neurocytes are frequently put in order in layers, with each layer fully linked to the one before it. Because the entry pattern is implemented in the first level of such systems, it is also known as the entry layer. The final level is sometimes referred to as the outcome layer since it is where the categories are given. As a result, if letters are to be anticipated, the outcome layer must have the same length as the amount of probable categories, i.e. the dictionary. The levels in between being referred to as hidden layers since they are where the real training / learning occurs. Figure 2.1 is a simplified depiction of such a DNN.

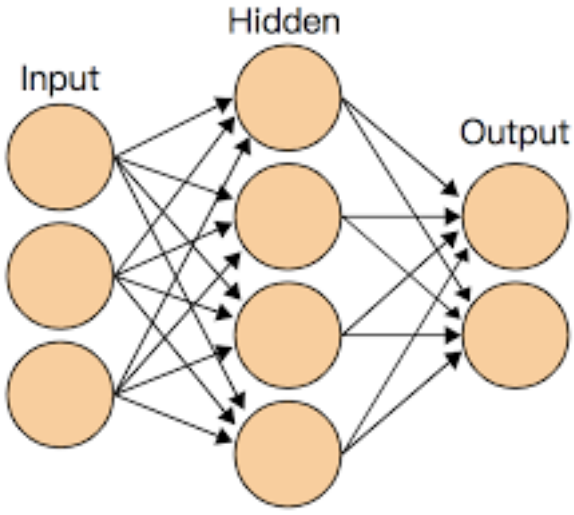

Figure 2.1: A basic DNN depiction

The error δ in the output layer is used to train this network. The outcome of that outlet neurocytes is normally within zero and one whenever that systems are employed for categorization. As a result, the right outlet neurons forecast a 1 and all other neurons are inactive (0) for a valid categorization. Because targeting neurons are seldom properly anticipated, particularly whenever the network is

initiated arbitrarily, an incorrect forecast Y' for an implanted input pattern X occurs. As a result, training may be carried out with corresponding the faulty speculation Y' within the accurate forecast Y, yielding a fault δ:

$$\delta = |Y \ - \ Y'| \tag{2.5}$$

The weights of the network's neurons can be adjusted using this error. Back-propagation is the term for this procedure [37]. The network's prediction improves over time by modifying the weights based on the error, given that the entry series X include a foreseeable design that is planned out by Y.

Initially, that DNN systems were employed to store lots of consistent representations of sound information, that were subsequently transmitted to traditional HMM/GMM systems [20]. GMMs were eventually totally supplanted with DNNs. Neural networks' function and structure have gradually changed above the last ten years, issuing a wide range of network architectures for various app areas. ASR models today are made out of HMMs or/and numerous mixtures of dissimilar neural networks.

The majority of essential neural network variants for ASR systems, particularly those pertinent to this discipline, will be discussed in the following subsections.

2.3.1 Convolutional Neural Networks

Convolutional neural networks, or CNNs, are a further refinement of neural networks [25]. Because of its capacity to discern trends in high-dimensional arrays, CNNs have acquired popularity in the field of image processing, particularly in picture identification.

CNNs vary from traditional neural networks in that they execute a convolution operation in the hidden layers. Unlike traditional hidden layers, this convolution function transforms an input into an output. In the case of picture identification, two-dimensional filterings, also known as kernels, are employed for twist.

Because that kernels are typically considerably not larger than the entries, they have to be moved over the full entry picture. A scalar multiplication is generated betwixt the entries and the kernels in this case. This technique generates a feature map as output, as shown in Figure 2.2:

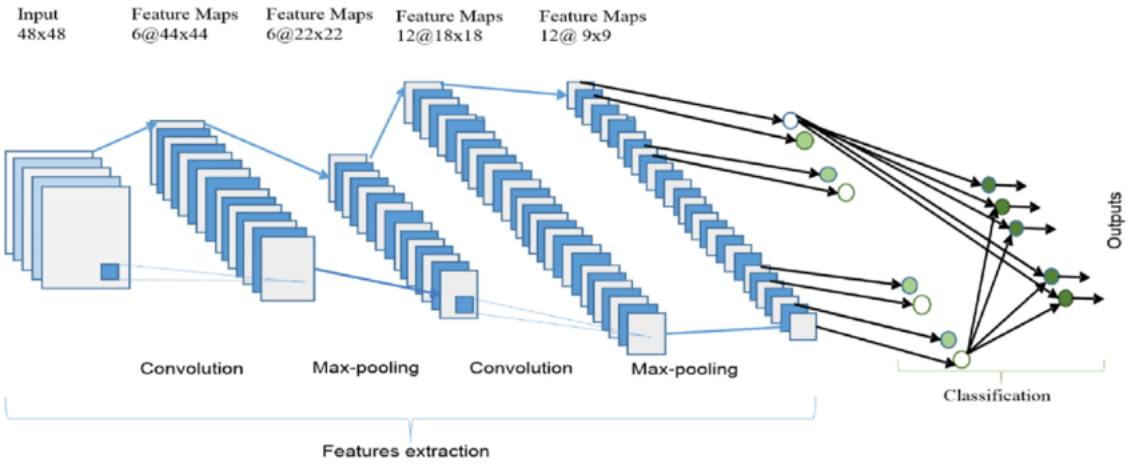

Figure 2.2: Typical illustration of a CNN

The feature maps created by training these kernels can distinguish edges, for example. As shown in Figure 2.2, sub-sampling levels are put among these convolutions to improve adaptability. To perform this, these sub-sampling or max-pooling layers use kernels to diminish spatial resolution. Sub-sampling kernels, unlike convolutional kernels, are not taught. Instead, these kernels use max-pooling, which sends just the greatest value, or average-pooling, which sends an average value.

Convolutional layers, like traditional hidden levels, is able to be layered before the required degree of abstracting is attained in the last pecularity chart, as well known as unused area. This lofty latent-space abstracting, which shown in Figure 2.2, may be utilized to do categorization in a standard neural network, as explained in Section 2.3.

Although CNNs are designed for image recognition, if duration is viewed as the next dimensionality, they may also be taught on one dimensional matrices. As a result, Time Delay Neural Networks (TDNNs) were created [43]. As an example, an audio sequence may be split into a temporal series of evenly dimensioned designs that is able to be utilized as entry into a CNN. Designs in sound data, such as phonemes, might as well be discovered using this approach.

2.3.2 Self Attention

The architecture of Transformer is one of neural networks which is designed to function as an converter [42]. The model converts an input sequence into an outlet series, therefore that models are as well known as models of sequential manner.

The Self Attention processes are introduced in Transformer Models. These methods of self-attention assist the network in learning relationships between the presented sequences. Using deep features and positioning data, the input sequence is converted into vector representations. A scalar product may then be used to calculate the "attention" of these vectors.

The Transformer model employs several attention heads to learn the widest possible depiction of these interactions. This multi-head attention technique allows for the extraction of relationships from various representations and situations. As discussed in Section 2.3, these abstractions may now be categorized using a standard neural network. These layers, like CNNs, may be layered to obtain an even greater degree of abstraction.

The self-attention mechanism is especially well-suited for natural language processing applications when implemented as a sequence-to-sequence Transformer model. This encoder-to-decoder construction may be taught to encrypt a text from one language and decrypt it to the other one, yielding a translation architecture. However, so complicated usage scenarios are also feasible, similar as question-and-answer systems in which encodes a query preceded by a sequenceo of text and the decoder is taught to extract the proper response to the query from the sequence of text.

One more whip hand of converter Transformer models is that they facilitate self supervised studying in Natural Language Processing (NLP). The BERT Transformer Model [12] established the approach of masked-language-modeling (MLM), which uses the same phrases as inlet and outlet. Nonetheless, phrases are arbitrarily disguised in the oulet series that the model must predict. As a consequence, the model learns language patterns and word context.

2.4 Language Modelings

Because of algorithms like CTC 2.2.2, most modern ASR models no simply reliant on explicit language models to forecast. As a result, these models may be trained end to end, from sound inlet to written outlet. It owns the huge benefit of allowing acoustical models of ASR to receive speaking training with no having to grasp the grammar or context. However, this independence has a drawback in that any projected letter may not be tested for accuracy in scope. For instance, if an empty

synbol is incorrectly forecasted, the phrase "Helo" is a valid forecast which may not be rectified by the acoustical architecture.

Language models step in here, estimating the probability for specific words and word combinations. As a result, these models can not only correct basic mistakes like exchanged or missed characters, but they are also able to find and repair situational issues in a phrase conjunction. It's more feasible to forecast the coming most likely phrases. The most prevalent language modelings are detailed in the following subsections.

2.4.1 N-Gram Language Model

N-Gram models are the most often used language models [6]. Sequences are divided into discrete pieces in an N-Gram model. The size of the pieces is completely customizable. It may be phonemes, characters, or even entire phrases. The One Gram architecture, often known as the unigram model, is the most basic N-Gram model. This model just saves the probability of individual pieces. When the fragment size is determined by the number of words, each recognized phrases' likelihoods are recorded only.:

$$P(w) \tag{2.6}$$

Because it recognizes and corrects misspelled words, this simple technique is already quite successful. For example, depending on the extent of the vocabulary, the word "Helo" has no or only a very low likelihood. As a consequence, it is rectified to the recognizable word "Hello". However, this approach has a significant drawback in that it eliminates terms that were previously unknown. Furthermore, because the phrase "the," as "they," is element of the 1-gram architecture, context-dependent inaccuracies like "the went swimming" have not yet been identified in this situation.

Larger N gram architectures, like the two-gram modeling, as well known as bigram, or the three-gram design, as well known as tri-gram, can help with this job. That architectures hold the probabilities for the pieces along with the probability for single pieces that come after them. The two previous words would also be evaluated in a trigram word language model:

$$P(w_n \mid w_{n-1}, w_{n-2}) \tag{2.7}$$

As shows exemplification above, the language modeling might most likely rectify "the went swimming" to "they went swimming," as this conjunction has a greater likelihood than the initial. Even though the level of the N Gram architecture is theoretically extensible indefinitely, technological constraints are frequently achieved in this instance because the amount of potential conjunctions grows enormously within every extent, making computer assets a limitizing issue.

2.4.2 Transformer Language Modeling

Transformer models, as discussed in Section 2.3.2, are ideally fitted for NLP applications. These models can acquire language patterns and even word context thanks to the capacity of self-supervised learning.

As a result, models of Transformer are as well appropriate in order to develop a language modeling. They, as the learning technique of MLM, may find misinterpreted or miscategorized phrases. Unknown words, on the other hand, represent an issue if these transformer models are trained on whole words. The model depends on learnt vector representations due to the first embedding layer in the encoder and decoder. As a result, unseen words may obtain character-based vector representations and are not appropriately recorded by the system. This owns a detrimental influence on zero shot studying as well. Nevertheless, because sequential architectures may be learnt on various phrase representations, this issue can be avoided.

Transformer models have a significant advantage over typical n-gram language models in that they forecast using the full input sequence rather than just the n final words. As a result, when the Transformer model predicts the following word, it has access to the whole input context.

3. Related Works

The research [32] developed an end-to-end model with transfer learning targeted at identifying Kazakh and Azerbaijani languages and resolving issues with voice resources. Previous research has shown that satisfactory results may be reached for an end-to-end model without incorporating language models.

Transfer learning is a technique for transferring models learned on one dataset to another. This strategy was anticipated to result in the following enhancements. To begin with, using the technique derived from the Kazakh representation model would save training time as compared to starting from scratch. Second, compared to models for the Azerbaijani language, an end-to-end model trained via transfer learning requires less data for equal assessment. As a result, researchers anticipated to utilize less GPU memory because they didn't need to provide gradients for all layers.

The study's goal was to use the transfer learning approach to construct an automated voice recognition system for agglutinative languages like Kazakh and Azerbaijani.

The following objectives were defined in order to attain the study's goal:

- to gather and produce speech corpora with transcriptions for the Kazakh and Azerbaijani languages for system training, and to create a corpus that contains all forms of speech – planned and spontaneous – that must be considered;

- to use the transfer learning approach to train a voice recognition system with limited data resources, and to assess the success of the training by comparing the results achieved with the transfer model and the basic models.

Transfer learning was used in an end-to-end model to boost efficiency and swiftly solve challenges associated with restricted resources. Although multilingual data is used to teach transfer learning, acoustic similarities from common layers are more beneficial. Feature extraction from multilingual voice data was

regarded to be an effective strategy for embedding generic acoustic information into end-to-end models in this study. Several separate RNNs with shared hidden layers were trained using two language resources in the initial stage of the experiment. The input with two language data for transfer learning is depicted in the Figure 3.1 below; the automated recognition system's design has common hidden layers with the output.

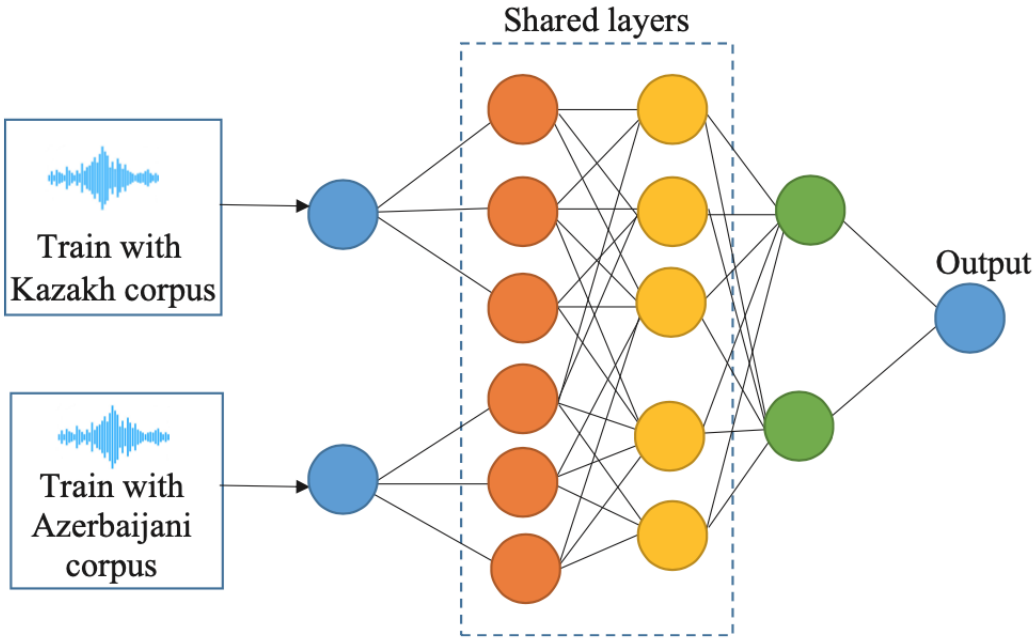

Figure 3.1: RNN with shared hidden layers

To minimize overfitting and find the best common features, the maxout activation function was combined with dropout training. The maximum was then averaged or calculated using a simple max pooling technique. To retrieve low-dimensional features from the RNN, all parameters under the final hidden layer were shifted, and a new layer SoftMax with random parameters was introduced, followed by an adjustment of the whole target RNN. Such training adaption allows for maximal nonlinearity for later processing without compromising the structure of the neural network during training.

A joint end-to-end model based on two architectures, CTC and attention, was constructed on the basis of earlier works, with a single encoder and a combination decoder. To increase system performance after modeling, the monotonic constraint was transferred from the CTC to a decoder using an attention mechanism.

A network was considered as an intermediary procedure for their joint model to extract the characteristics from the incoming signal. As a result, retrieved char-

acteristics were already high-level, and mapping these original data to phonemes was unnecessary. A model based on shallow bidirectional LSTMs was developed in this study. The Figure 3.2 below shows the implemented model.

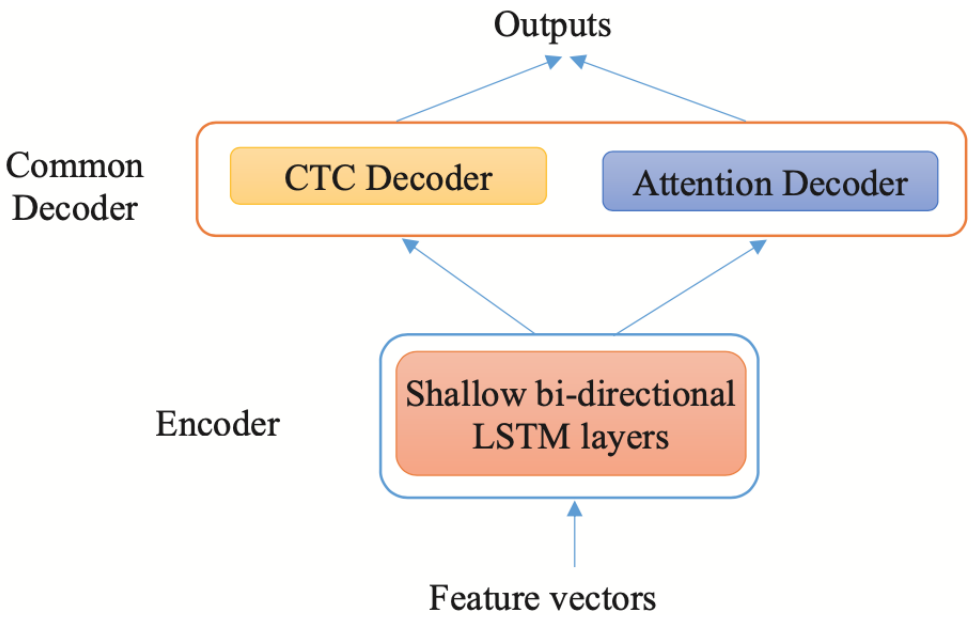

Figure 3.2: The implemented model

The developed end-to-end models were trained using 32 phonemes from Azerbaijani and 28 phonemes from Kazakh. Only 60 phonemes were chosen for testing. There were 470 hours of data for training. The percentages utilized for test and development (dev) data were 20% and 80%, respectively.

The attention-based model was trained separately after the CTC model was developed in the first stage of the experiment. A directed six-layer BLSTM with 256 cells in each layer makes up the basic CTC model. The encoder for the attention-based model is a three-layer directional BLSTM with 256 cells in each layer. The 120 cells in the attention layer are depending on location. The decoder is a 256-cell single-layer LSTM. For the encoder, attention, and decoder, the dropout rates for inputs were –0.2, –0.5, and –0.1, respectively. The Adam algorithm was used to optimize the models. CTC had a decoding weight of 0.3. At the decoding stage, the beam research width was 15. The learning process could not identify acoustically similar words from the Kazakh and Azerbaijani languages until the 45^{th} epoch. This corpus contains 71.649 comparable acoustic words.

The model with the highest accuracy was chosen as the final model after 45 training epochs to solve the aforesaid issues.

For the other hidden layers, all RNNs were trained with a dropout rate of 0.2. The initial learning rate stayed constant at 0.3 for the first 26 epochs before tripling.

The phoneme error rate (PER) is a regularly used statistic for evaluating the phoneme recognition system, and it is determined using the Levenshtein distance, where phonemes are employed instead of words. The suggested model has a PER of 14.23 percent after assessing the symbol error rate.

A new strategy based on transfer learning was presented for end-to-end voice recognition. The NMF algorithm was used to extract features in the first step. The NMF method was used to train the joint CTC attention models based on the retrieved characteristics in the second step. At two levels, multilingual learning and multitasking were used to implement transfer learning. The model outperforms all end-to-end models and achieves great performance when compared to a current speech recognition system, according to the findings of the experiments. Although the suggested transfer learning method increases the performance of end-to-end speech recognition models, it is not without flaws. The findings revealed that the voice recognition was of good grade. The transfer learning strategy was found to lower the PER indicator by 14.23% when compared to the basic models in studies using two corpora.

The work [31] proposed a model based on Recurrent Neural Transducer (RNN-T) for Kazakh speech recognition.

The RNN-T model is made up of an encoder, a prediction network, and a joint network; the RNN-T model has a similar structure to an encoder-decoder with an attention mechanism, if the decoder may show as a connection between the prediction network components and the joint network. In a conventional speech recognition system, RNN is an encoder that turns the input acoustic data into a high-level intermediate representation and performs the same function as AM. As a result, the RNN network's output is conditioned by the sequence of preceding acoustic data, as in the CTC model. On including an element of the RNN prediction network that is explicitly stated by the expected history of past non-empty model targets, RNN-T eliminates the conditional independence requirement in CTC.

To train the RNN-T model, a speech corpus was used, which comprises over 300 hours of speech and was assembled in the Institute of Information and Compu-

tational Technologies MES RK's laboratory "Computer Engineering of Intelligent Systems." This corpus includes recordings of Kazakh speakers of various genders and ages, as well as telephone calls with transcriptions and recordings from news sites and art publications.

The audio files were all in.wav format. Because telephone conversations are recorded on two channels, it was decided to combine all of the recordings into a single channel. The data was converted to digital form using the PCM technique. The sample rate is 44.1 kHz, with a bit depth of 16 bits.

The PyTorch toolkit was used to create an end-to-end voice recognition system based on the RNN-T model. Experiments were carried out on AMD Ryzen 9 GPUs with GeForce RTX3090 on equipment given by the Institute of Information and Computational Technologies, on which this study was undertaken as a research practice. To enable for quicker data stream during training and recognition, the datasets were kept on 1000 GB of SSD memory.

The network was trained using 90% of the corpus data, and the model was validated using 10% of the data. The system was also tested using the ISSAI Kazakh Speech Corpus speech basis.

A BLSTM with 5 layers and 1024 units was used in the encoder, augmented by CNN layers, and an LSTM with 2 layers and 1024 units was utilized as a prediction network, each with a dropout.

Algorithms were chosen and parameter values were set to speed up training and increase the model's quality, including regularization coefficient, batch size, gradient descent optimization approach, and others.

The Kazakh letter lexicon has 42 characters, and the output length of proposed networks was adjusted to 44 due to the insertion of extra tokens for alignment.

It should be noticed that the RNN-T model decoded faster than the CTC model. During testing, the RNN-T-trained system swiftly converted the sentences into a series of words. Of course, the increase in training data was important.

The findings proved that the RNN-T model for the Kazakh language can run flawlessly without the need of an extra external language model and outperformed other end-to-end models.

The current situation of end-to-end speech recognition systems, namely the RNN-T model for stream speech recognition, was discussed in the paper. This model's architecture was created utilizing neural networks such as LSTM and

BLSTM. The work's findings revealed that the implemented model can achieve good performance without using language models for the Kazakh language, and that it outperformed other end-to-end models that were trained on a smaller volume of speech data, with the best outcomes in recognizing Kazakh speech in terms of character recognition rate (CER) of 10.6 percent. Furthermore, the RNN-T model-based system may be deployed on mobile devices since it uses less memory.

The aim of the research [34] was to investigate the efficacy of a multilingual end-to-end (E2E) automated speech recognition (ASR) system applied to three languages that are in Kazakhstan: Kazakh, Russian, and English. The purpose of the project was to become fluent in the aforementioned languages of Kazakhstani people. In this context, the first investigation of Kazakh, Russian, and English languages to be concurrently recognized by an E2E single joint ASR model was launched.

Three languages were used in the studies, each with its own collection of graphemes. In addition, each language has its own distinct training set, with X being an acoustic feature input sequence and Y denoting a target sequence.

The multilingual models' training dataset was created by merging all three datasets without any re-weighting or re-balancing, and the grapheme set for the combined dataset was created in the same way.

Experiments began with the training of randomly-initialized monolingual ASR models for each language using the associated training data $\{X_i, Y_i\}$ and grapheme set G_i, where i is in (kz, ru, en). These models were utilized as a baseline and are encoder-decoder networks based on the Transformer architecture.

Following that, a joint model was trained using the multilingual datasets $\{X_{all}, Y_{all}\}$, and the combined grapheme set G_{all}. The joint model was based on the Transformer architecture as well, but it was a single model with common parameters across all three languages. The fact that the training dataset was made up of different languages was not explicitly stated in this model. The amount of parameters and structure of the multilingual and monolingual models were set to be similar for fair comparison.

Three datasets corresponding to the Kazakh, Russian, and English languages were utilized to undertake multilingual voice recognition studies. Figure 3.3 depicted the dataset's specs. The training set sizes of languages were made identical

to reduce the performance loss caused by the problems unique to imbalanced data (in terms of duration). Prior to training and evaluation, all audio recordings were resampled to 16 kHz and 16-bit format.

Languages		**Corpora**	**Duration**	**Utterances**	**Words**
Kazakh	train	KSC [19]	318.4 hr	147.2k	1.6M
	valid		7.1 hr	3.3k	35.3k
	test		7.1 hr	3.3k	35.9k
Russian	train	OpenSTT-CS334	327.1 hr	223.0k	2.3M
	valid		7.1 hr	4.8k	48.3k
	test-B (books)	OpenSTT [30]	3.6 hr	3.7k	28.1k
	test-Y (YouTube)		3.4 hr	3.9k	31.2k
English	train	CV-330	330.0 hr	208.9k	2.2M
	valid	CV [6]	7.4 hr	4.3k	43.9k
	test		7.4 hr	4.6k	44.3k
	test-SF	SpeakingFaces [3]	7.7 hr	6.8k	37.7k
Total	train	-	975.6 hr	579.3k	6.0M
	valid		21.6 hr	12.4k	127.5k
	test		29.1 hr	22.5k	177.3k

Figure 3.3: Statistics for the Kazakh, Russian, and English languages in the dataset. Utterance and word counts are measured in thousands (k) or millions (M), respectively, while durations are measured in hours (hr). The total statistics are calculated by merging the training, validation, and test sets from all languages.

The recently released open-source Kazakh Speech Corpus (KSC) was utilized for Kazakh. The KSC comprised around 332 hours of transcribed audio crowd-sourced over the Internet, in which volunteers from various areas and age groups were requested to read phrases delivered through a web browser. Around 153,000 recordings were approved from over 1,600 different devices in total. Native Kazakh speakers personally vetted all submitted recordings. All texts in the KSC were written in the Cyrillic script, and audio recordings were saved in the WAV format. The typical split of non-overlapping speakers was employed for the training, validation, and test sets.

A carefully cleaned subset of the Russian Open Speech To Text (OpenSTT) dataset was utilized for Russian. The OpenSTT collection had about 20,000 hours of transcribed audio data from various domains (e.g., radio, lectures, phone

conversations, and so on). However, the offered transcriptions were untrustworthy since they were made automatically through the use of ASR systems, YouTube subtitles (both user-provided and auto-generated), and so on. Only the three validation sets from the books, YouTube, and phone calls domains received clean transcriptions.

Fluent Russian speakers were employed and a randomly selected 334-hour subset of the OpenSTT was painstakingly re-transcribed to acquire more trustworthy training data. Only recordings from the books and YouTube domains were chosen. The clean subset was given the name OpenSTT-CS334 and the 334-hour original version was given the name OpenSTT-ORG334. A 7-hour subset of the OpenSTT-CS334 was randomly selected for the validation set, leaving the remaining 327 hours for training. The official OpenSTT validation sets from the books (test-B) and YouTube (test-Y) domains were utilized to match the specified training data for the test set.

A 330-hour subset of Mozilla's Common Voice (CV) project was employed for English, which was dubbed the CV-330. The CV was a multilingual dataset created for research and development in voice technology. It was built in the same way as the KSC, with volunteers being recruited to read and check texts. The CV-330 is made up of certified recordings that have gotten the most up-votes. A 7-hour subset of the standard validation and test sets was randomly taken for assessment reasons and supplied in the CV. It's worth noting that throughout the training, validation, and test sets, the speakers and texts did not overlap.

A separate assessment set (test-SF) was employed, which consisted of Kazakh-accented English recordings derived from the SpeakingFaces dataset. SpeakingFaces was a multimodal dataset made accessible to the public that included thermal, visual, and audio data streams. The audio was recorded at a distance of around one meter using a built-in microphone (44.1 kHz) on a webcam (Logitech C920 Pro HD). The dataset included almost 13,000 audio recordings of 142 speakers of various ethnicities speaking imperative phrases. Recordings of Kazakhs were chosen, resulting in a total of 75 speakers, each delivering roughly 90 orders. The assessment set created was gender balanced (38 females and 37 males), with a speaker age of 26 years (ranging from 20 to 47). This dataset was thought to be the first open-source Kazakh-accented English data, and it was better suitable for evaluating our E2E ASR models' English speech recognition capacity.

The V100 GPUs operating on an Nvidia DGX-2 server were used to train all E2E ASR systems on the training sets; hyper-parameters were tweaked on the validation sets; and the final systems were assessed on the test sets. The input auditory characteristics were represented as 80-dimensional log Mel filter bank features with pitch computed every 10 ms over a 25-ms window for all systems, while the output units were represented by character-level graphemes.

The ESPnet toolbox was used to train the E2E ASR systems, and the Wall Street Journal (WSJ) formula was used. The Transformer network, which consisted of 12 encoder and 6 decoder blocks, served as the foundation for the E2E design. Under the multi-task learning framework, it was jointly trained with the Connectionist Temporal Classification (CTC) objective function. During the training and decoding stages, the interpolation weight for the CTC goal was set to 0.3 and 0.4, respectively. The number of heads in the self-attention layer for the Transformer module was set to 8, each having 512-dimension hidden states, and the feed-forward network dimensions were set at 2,048. In addition, before the encoder blocks, a VGG-like convolution module was utilized to pre-process the input audio information. The Noam optimizer was used to train all models for 120 epochs with an initial learning rate of 10 and 25k warm-up steps. The dropout rate was set at 0.1, as was the label smoothing. A typical 3-way speed perturbation with factors of 0.9, 1.0, and 1.1, as well as spectral augmentation, were employed for data augmentation. The results were based on an average model built from the past 10 checkpoints.

Character-level LMs were developed using the transcripts of the training sets to assess the influence of language models (LM) on recognition performance. The LMs were constructed as a two-layer long short-term memory (LSTM) network with 650 memory cells per layer. For monolingual and multilingual E2E ASRs, monolingual and multilingual LSTM LMs were created. The combined training set was used to train the multilingual LSTM LM. Shallow fusion was used to use the LSTM LMs during the decoding stage. The beam size was set to 60 and the LSTM LM interpolation weight at 0.6 for decoding. Also, it was mentioned that the authors' GitHub repository contains the settings for the remaining hyper-parameters.

Figure 3.4 depicts the experiment outcomes for the three languages. The WERs were weighted by the quantity of data in the validation and test sets to

generate an average WER across all languages.

Model	Kazakh		Russian			English			Average	
	valid	test	valid	test-B	test-Y	valid	test	test-SF	valid	test
mono	21.5	18.8	15.2	17.2	33.7	29.8	34.6	62.0	22.0	34.3
+LM	15.9	13.9	11.5	14.5	28.8	24.7	29.1	57.7	17.3	29.7
+LM+SP	15.3	12.7	9.8	13.4	25.5	23.1	26.7	53.9	15.9	27.3
+LM+SP+SA	**9.4**	8.0	**7.5**	**11.8**	**21.9**	**16.3**	**18.9**	41.6	**11.1**	20.9
multi	20.4	16.3	13.7	16.5	31.5	28.0	32.2	56.0	20.5	31.4
+LM	15.4	12.6	11.2	14.7	28.0	23.7	27.5	51.4	16.7	27.6
+LM+SP	14.4	11.8	10.2	13.8	25.8	22.5	26.4	48.3	15.6	26.0
+LM+SP+SA	9.7	**7.9**	8.2	12.5	23.3	17.1	19.9	39.5	11.7	21.1
multi-igs	20.7	16.6	13.8	16.5	31.3	27.7	32.4	56.4	20.5	31.6
+LM	15.9	12.8	11.3	14.6	27.7	23.4	27.4	51.6	16.7	27.6
+LM+SP	14.9	12.1	10.4	13.6	25.8	22.4	26.1	49.9	15.8	26.3
+LM+SP+SA	9.7	**7.9**	8.3	12.5	23.2	**16.3**	**18.9**	**38.3**	11.5	**20.5**

Figure 3.4: The WER (%) findings for monolingual (mono), multilingual (multi), and independent grapheme set (multi-igs) models. The effects of language model (LM), speed perturbation (SP), and spectrum augmentation (SA) are also presented. Weighting the WERs based on the quantity of data in the validation and test sets yields the average WER.

The findings of monolingual models demonstrated that using LMs and data augmentation consistently increases WER performance for all languages, with a 13.4% average WER improvement attained on the test sets (from 34.3% to 20.9%). On the test set, the best WER result for Kazakh was 8.0%. On the test-B and test-Y sets, the best WER values for Russian were 11.8% and 21.9%, respectively. It is worth noting that recognizing YouTube recordings (i.e., test-Y) was more difficult than recognizing audiobooks (i.e., test-B) since the former comprises recordings with spontaneous speech. On the test and test-SF sets, the top WER values for English were 18.9% and 41.6%, respectively. Because the English training set recordings were read by native English speakers, the low WER performance on the latter was most likely owing to a domain mismatch between the training and test-SF sets. Furthermore, these sets were gathered in various ways. For the monolingual models, the best average WER performance on the test sets was 20.9%.

The combination of LMs and data augmentation was similarly beneficial for multilingual models, with a 10.3% average WER improvement on test sets (from

31.4% to 21.1%). The overall trend of WER performance was comparable to that of monolingual models. The best WER for Kazakh, for example, was 7.9%, which was extremely near to the monolingual baseline. Russian achieved somewhat lower WERs than monolingual baselines, with the test-Y set being more difficult than the test-B set. Similarly, for English, minor WER degradations were seen, and performance on the test-SF set was worse than on the test set. However, it is worth noting that the multilingual models demonstrated a 2.1% WER improvement on the test-SF set when compared to the monolingual baseline (39.5% vs. 41.6%). This improvement is thought to be mostly attributable to knowledge transfer from the KSC and OpenSTT-CS334 datasets. The best average WER result for the multilingual models on the test sets was 21.1%.

The experimental findings indicated that multilingual models with separate grapheme sets (i.e., multi-igs) performed similarly to monolingual models for Kazakh and English, but somewhat worse for Russian. Notably, it performed better on the test-SF than the monolingual baseline (38.3 percent versus 41.6 percent). Overall, the WER results of the two grapheme set creation approaches were equal, with the independent grapheme set marginally outperforming the other. The best average WER score for the multi-igs models on the test sets was 20.5 percent, which was the lowest of all the E2E ASR models.

4. About Dataset

This chapter will introduce a dataset used in the experiments part of the research work. It is a Kazakh Speech Corpus (KSC) developed by the Institute of Smart Systems and Artificial Intelligence (ISSAI) at Nazarbayev University (NU).

The paper [24] proposes an open-source Kazakh speech corpus (KSC) designed to aid in the development of Kazakh speech and language processing systems. Kazakh is an agglutinative language with vowel harmony that belongs to the Turkic language family. During the Soviet period, the Kazakh language was dominated by Russian, resulting in a reduction in Kazakh language usage. It was proclaimed Kazakhstan's official language in the 1990s, and several projects were started to expand the number of Kazakh speakers. It is now spoken by about 10 million Kazakhs and over 3 million persons in other nations. The authors' goal in creating the KSC was to expedite the Kazakh language's penetration of Internet of things (IoT) technology and to foster research in Kazakh speech processing applications.

Despite the fact that various Kazakh speech corpora have been given in earlier studies, there is no widely approved common corpus. The majority of them are either publicly unavailable or have insufficient data to build credible models. These databases, in particular, are insufficient for developing modern end-to-end models, which require a large amount of data. As a result, various research organizations often perform their studies on data generated internally, preventing repeatability and comparison of different methodologies.

The KSC, which contains around 332 hours of transcribed audio, was built to resolve the aforementioned restrictions. It was crowdsourced through the Internet, with volunteers invited to interpret texts displayed in a web browser. Over 153,000 utterances were approved from over 1,600 distinct device IDs in total. The recordings were examined manually first, and then partially verified

automatically using a speech recognition system when a significant quantity of data was acquired. The Kazakh Speech Corpus (KSC) is the biggest open-source speech corpus in Kazakh and is free for research and commercial usage under the Creative Commons Attribution 4.0 International License.

4.1 Collection and Cleaning of text

Initially, Kazakh textual material was gathered from a variety of sources, including electronic books, legislation, and websites such as Wikipedia, news portals, and blogs. A dedicated web crawler was created for each page to increase the quality of the retrieved content. The retrieved texts were manually screened to exclude information that was improper, such as sensitive political concerns, user privacy, violence, and so on. Additionally, texts containing Russian language were completely filtered out. Because there are numerous borrowed Russian terms in Kazakh and it is normal practice among Kazakh speakers to code-switch between Kazakh and Russian, texts containing mixed Kazakh-Russian utterances were maintained. The materials were then divided into sentences, and sentences with more than 25 words were deleted. Finally, redundant sentences were eliminated. The entire amount of retrieved phrases was estimated to be roughly 2.3 million.

4.2 Narration and Checking the text

A web-based audio recording technology capable of operating on personal computers and smartphones was created to narrate the extracted phrases. The platform selected a sentence at random from the pool of extracted texts and delivered it to the reader. It also showed the recording status and information like elapsed time and total number of read sentences. It also contained "pause" and "next" buttons for controlling the recording process. The readers were free to leave at any moment. Readers were recruited through publicizing the initiative on social media, in the press, and in WhatsApp and Telegram open messaging communities. Readers above the age of 18 were included so that they may legally consent to participate in data collecting. The audios were originally recorded at 48 kHz and 16 bits, but were downsampled to 16 kHz and 16 bits for web release. Except for the geographical coordinates, IP address, and device type, readers' personal

information were not saved in accordance with the experimental protocol.

To ensure the quality of the recordings, many native Kazakh transcribers were employed. Transcribers logged onto a specific transcription testing platform and were given an audio segment as well as the accompanying sentence read by a reader. The aim was to determine if the reader had read the text correctly and to record any deviations or other audio occurrences using a set of transcription instructions. A linguist was employed as an extra quality control measure and was tasked to monitor the transcribers and randomly examine the jobs they did. The linguist also held "walk through mistakes" sessions with the transcribers to harmonize the transcriptions.

Transcribers were told to eliminate utterances with noticeable mispronunciations or loud sounds, to turn numbers into words, and to cut extended silences at the beginning and conclusion of audio segments. They were also told to put partial repeats and hesitations in parenthesis, such as '(he) hello' and '(ah)', and to use an unique '[noise]' symbol to reflect nonverbal sounds made by readers, such as sneezing and coughing. Noises in the background were not identified.

When the number of acceptable utterances reached 100 hours, an ASR system was constructed to verify the recordings automatically. The system only accepted recordings that exactly matched the associated text prompts – that is, 0% character error rate (CER), leaving the remainder utterances to human transcribers.

4.3 Specifications of database

Category	Train	Valid	Test	Total
Duration (hours)	318.4	7.1	7.1	332.6
# Utterances	147,236	3,283	3,334	153,853
# Words	1.61M	35.2k	35.8k	1.68M
# Unique Words	157,191	13,525	13,959	160,041
# Device IDs	1,554	29	29	1,612
# Speakers	-	29	29	-

Figure 4.1: The specifications of KSC database.

Figure 4.1 depicts the KSC database specs. The data was divided into three non-overlapping sets of speakers: training, validation, and test. While the training set recordings came from anonymous individuals, the validation and test sets came

from identified speakers to verify that they did not overlap with the training set, represented diverse age groups and areas, and were gender balanced (see Figure 4.2). A total of about 153,000 utterances were approved, resulting in 332 hours of recorded speech data. It should be noted that the device IDs could not be utilized to indicate the number of speakers in the training set since many speakers may have used the same device or the same speaker could have used various devices. As a result, Figure 4.1 does not reflect the complete number of speakers in the training set. The entire database building procedure took around four months, and the database size is approximately 38GB.

Category		Valid	Test
Gender (%)	Female	51.7	51.7
	Male	48.3	48.3
Age (%)	18-27	37.9	34.5
	28-37	34.5	31.0
	38-47	10.4	13.8
	48 and above	17.2	20.7
Region (%)	East	13.8	13.8
	West	20.7	17.2
	North	13.8	20.7
	South	37.9	41.4
	Center	13.8	6.9
Device (%)	Phone	62.1	79.3
	Computer	37.9	20.7
Headphone (%)	Yes	20.7	17.2
	No	79.3	82.8

Figure 4.2: The details of speakers in validation and test sets.

One of the most important aspects of the proposed KSC database is that it was collected in a variety of environments (e.g., home, office, café, transportation, and street), with varying background noises, using mobile devices (e.g., phones and tablets) and personal computers, with and without headphone sets, to mimic realistic use-case scenarios. As a result, the given database allows for the creation and testing of ASR systems intended for use in real-world speech-enabled applications such as voice commands, voice search, message dictation, and so on.

The KSC database was made up of different files containing audio recordings, transcripts, and metadata. The audio and transcription filenames were identical, except that the audio recordings were saved as WAV files and the transcriptions were saved as TXT files with UTF-8 encoding. The metadata included informa-

tion on the data splitting (training, validation, and test) as well as the speaker's characteristics (gender, age, region, device, and headphones) for the validation and test sets.

5. End-to-end ASR systems

Due to the problems of HMM-based ASR systems described in the preceding sections, as well as the popularity of the deep learning area, many researches and studies have been aimed toward end-to-end huge vocabulary ongoing speech recognition. Figure 5.1 depicts the construction of an end-to-end system based on directly translating the entry audio series into a phoneme or word series.

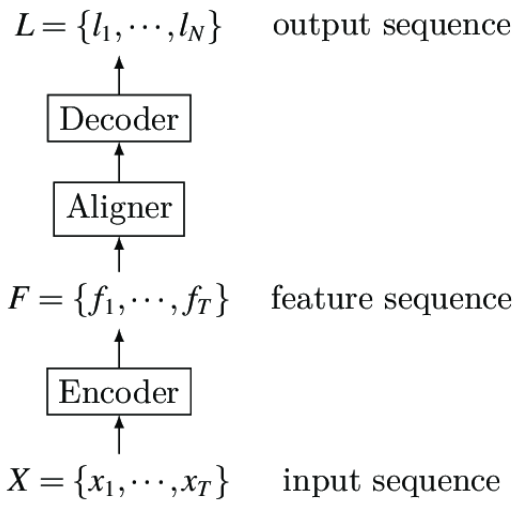

Figure 5.1: The structure of end-to-end based model.

The following components are almost included in all end-to-end models: an encoder (which maps an input sequence into a sequence of functions), an equalizer (which ensures consistency between the sequence of features and the language), and a decoder (which decodes the final result).

However, because the end-to-end system has a comprehensive design, its structure may not always be the same; it is sometimes difficult to discern which component is accountable for what.

The end-to-end model differs from typical HMM-based models in that it employs deep neural networks to directly turn an input audio sequence into a sequence of words. Furthermore, no post-processing is required for the output.

End-to-end models for continuous speech recognition with a large vocabulary contain the following qualities, in contrast to the HMM-based models outlined above:

- For collaborative learning, several modules can be integrated into a single network. The primary benefit of merging modules is that it eliminates the need to build many modules to describe intermediate states. Co-learning instructs the end-to-end model to optimize the feature most closely connected to the ultimate score, allowing for a global outcome to be sought;

- It matches the acoustic pattern to the text pattern quickly, avoiding the need for a second processing step to achieve the right transcription. Separate models represent acoustic and pronunciation aspects in HMM-based models.

When comparing the advantages of end-to-end based models to HMM based models, it becomes clear that end-to-end based models make it easier to build the architecture for speech recognition.

However, data alignment issues are ubiquitous in sequence modeling challenges, including speech recognition. The challenge of mapping a label in a sequence of labels to audio data cannot be avoided by end-to-end models or HMM-based models. Using a probability distribution, each audio frame matches every potential condition without necessitating a forced match.

The end-to-end models are divided into three categories based on how well they achieve soft alignment:

- CTC (Connectionist Temporal Classifier): CTC calculates all feasible alignments first, then combines these hard alignments to create the soft alignment. When calculating these alignments, CTC assumes that each label is independent of the others;

- Recurrent Neural Network (RNN) Transducer: To compute the soft equalization, the RNN transducer assesses and adds all hard equalizations. The RNN transducer, unlike the CTC converter, does not presume label independence throughout hard equalization. It takes a totally different attitude to route and probability computations than CTC;

- Attention-based model: this method eliminates the requirement to predict all hard alignments. It uses the so-called Attention mechanism to determine the soft alignment between input and output.

5.1 A model based on CTC

The performance of hybrid models based on HMM-DNN is now cutting-edge, and DNN's impact on the whole system is minimal. It is frequently used to simulate the HMM's posterior probability and to describe local information. HMM is still in charge of the feature in the time domain. Attempts to substitute the HMM with a CNN or RNN to simulate time-domain characteristics always fail due to data alignment issues. Because alignment is critical, the RNN or CNN loss function is interpreted at every position in the sequence, making it trainable.

CTC was proposed for the first time in [17]. Its advancement gave an opportunity to take the complete application of DNN in ASR as well as the design of an end-to-end model. CTC is only a fundamental loss function. In spite of that, because it addresses the alignment problem while measuring the loss, it has a significant influence on total training. CTC essentially solves two major roadblocks:

- A problem with data alignment. There is no need to manually split and align the data any more. DNN may be used to represent time-domain characteristics by solving this challenge, which increases DNN's value in the entire system;

- Direct output of the transcription. Traditional approaches frequently produce a mixture of phonemes and other factors, necessitating a second step to reach the final goal transcription. CTC ignores the requirement for these parts and outputs the direct target, greatly simplifying the overall design.

By addressing the aforementioned concerns, CTC may map input sequences to output sequences and execute a well-structured end-to-end system with the assistance of a single network.

5.1.1 The main idea of the CTC

The essential concept underlying CTC is that it comprises of two key processes: route probability estimation and path aggregation. Furthermore, these two operations generate a new blank sign ("-," which denotes the absence of a label).

5.1.2 The estimation of path probability

Let's explore the path probability estimation for CTC. An encoder converts an input sequence $X = \{x_1, ..., x_T\}$ with a T length into a feature sequence $F = \{f_1, ..., f_T\}$ with the same T length. The feature vector ft at time t has a dimension of one plus the vocabulary size ($f_t \in R^{|v|+1}$).

CTC converts the feature sequence $F = \{f_1..., f_T\}$ into a probability distribution $Y = \{y_1, ..., y_T\}, y_t = \{y_t^1, ..., y_t^{|v|+1}\}$, where y_t^a represents the likelihood of an output at time t is a. The probability of a blank symbol at time t is shown by $y_t^{|v|+1}$.

V'^T describes the set of all sequences of length T that were described in vocabulary V' if $V' = V \cup \{b\}$. Taking into mind the description of y_t^k, we can state that the probability distribution of a certain sequence is estimated as follows for a given input sequence X:

$$p(\pi \mid X) = \prod_{t=1}^{T} y_t^{\pi_t}, \forall \pi \in V'^T \tag{5.1}$$

where π_t is the tag for the sequence π in t. V'^T is referred to as a route, and it is denoted by the symbol π.

The input pattern will be matched with a specific route of the same length T when the computation is completed. An equation will be used to estimate the conditional probability of a certain path. In layman's words, this technique ensures that there is a unique labeling pi_t for each frame x_t. This procedure of translating an entry to an outcome is also known as a hard-alignment operation.

We can see that the above-mentioned estimate equation is based on a considerable assumption of independence. It signifies that the output labels are fully independent of one another. Any label chosen at any moment has no effect on the general distribution at any other time. However, the value of y_t^k and the information about the future or past of the speech remain interdependent throughout the whole encoding process. To put it another way, CTC bases its assumptions on the language model rather than the acoustical model. As a result, the CTC encoder is entirely an acoustic model and lacks the capacity to represent language.

5.1.3 Aggregation of path

We see that the longitude of the final vector is the equal as the longitude of the entry pattern during route probability calculation, which is often not the case. The transcription is typically considerably shorter than the spoken sequence. As a result, long and short pathways must be matched in order to form a short sequence.

The aggregation procedure calculates the label sequence probability in addition to obtaining the label sequence from these pathways. $B^{-1}(L)$ represents the set of all pathways that correspond to the label sequence L. As a result, given an input sequence X, the label sequence probability will be determined as follows:

$$p(L \mid X) = \sum_{\pi \in B^{-1}(L)} p(\pi \mid X) \tag{5.2}$$

Equation 5.1 depicts the estimate of $p(\pi \mid X)$.

The calculation of $p(L \mid X)$ may clearly be separated. Once the label series probability has been determined, the system may be trained employing back-propagation.

However, a stumbling block appears when it comes to estimating the label sequence probability. It's difficult to say how many pathways from V'^{T} are encompassed in $B^{-1}(L)$. As a result, the forward-backward technique is used to calculate the label sequence probability. Despite the fact that the mapping of input to pathways is a difficult alignment procedure, CTC does not force an input and output to be aligned by a certain path due to the path aggregation process. As a result, it is possible to claim that CTC is the one that does the soft alignment, as opposed to models of HMM.

The development of the CTC approach simplifies the process of building and training speech recognition systems with a large vocabulary. It eliminates the need for the creation of additional dictionaries, ignores the data alignment requirement, and allows any neural network topology with the desired number of layers to be utilized for map function between output and input.

5.1.4 Related works

The removal of handheld data segmentation is one of the key advantages of adopting CTC. As a result, neural networks like CNNs and RNNs are crucial in this process. Many models based on different types of networks have been developed by researchers, with encouraging outcomes in E2E ASR systems.

Here, in [14], among others, the first CTC-related attempts to train models on LVCSR end-to-end tasks was suggested. A three-layer network was used to construct it. The first layer consisted of 78 direct link units, the second layer of 120 LSTM units, and the third layer of a 27-neuron LSTM. Training was the responsibility of the CTC level. According to the findings, boosting the network depth and the amount of blocks in the hidden layers can enhance performance dramatically. This was corroborated by the findings of [28].

In light of the preceding studies, work in [17] concluded that a three-layer LSTM is too tiny for an experiment and does not produce the intended results. This research employs the CTC algorithm to train a five-layer bidirectional LSTM (BiLSTM) with 500 neurons per layer. This model's total performance came close to becoming state-of-the-art. Many works on 5 layer based networks have appeared as a result of this study [18, 30, 38].

The CTC training-based ASR models have evolved in terms of overall structures and development as a result of the major growth of deep learning.

In terms of network architecture, [41] advocated combining CNN and RNN for voice recognition. The network is made up of 4 CNN layers, 2 DNN levels, 2 layers of RNN, and a layer of CTC. [45] built a system that ignores RNN and alternately employs CTC and CNN layers to overcome the problem of training difficulties while using RNN. The model is composed of 10 layers of CNN and three levels of complete connection. Convolution was conducted in both the frequency and temporal domains. The results demonstrated that the more complicated and deep networks that were used, the higher the model's performance became. Furthermore, they determined that well-designed CNN layers can represent temporal domain properties.

In terms of the sophisticated network structure, [1] attempted to train a 9-layer network with seven levels of RNN. In some instances, this model may be able to outperform humans. They employed a network based on CTC with seven BiLSTM layers, each having thousand neurons, in [40]. 125 thousand hours of audio data

from YouTube were used in the training. Similarly, [47] had encouraging outcomes on his own dataset using a network based on CTC with nine BiLSTM layers, each with 1024 blocks.

However, just because establishing a complicated and profound network to develop end-to-end ASR systems it doesn't guarantee that will function every time. Because their dataset does not permit training for big networks, several recent studies [3, 2] employed a smaller network (5 layers). For example, for 2000 hours of data, [3] utilized just 5 layers of network, whereas [40] needed 7 levels for 125000 hours of data. It's easy to detect the difference.

5.1.5 A large data training

A number of studies show that profound and complicated networks need a large amount of dataset, and voice recognition is no exclusion. Large-scale data is extremely beneficial to well-functioning ASR systems.

[18] employed datasets containing clean speech and their proper data of seven and five thousand hours, respectively, and additional sources of information such as Switchboard, Fisher, and others to correctly train the network with 5 layers. They got a total of 100000 hours of voice data for training by combining it with other noisy speech data. Researchers have attained a very high performance on their noisy speech using these massive datasets, outperforming several significant firms like as Google, Apple, and others. They employed twelve and nine thousand hours of English and Chinese data of speech, respectively, to train the network of nine layer in their subsequent studies [1]. They outperformed humans in terms of results.

According to them, [40] conducte a training of a BiLSTM of seven layers using dataset of 125 thousand hours and 100 thousand Google-based transcripts. In another experiment, [35] employed traffic dataset of 125 thousand hours Google voice-search.

The model's performance is significantly influenced by a large amount of data. Using that much data, however, necessitates a major increase in training methods.

[18] developed a technique with two aspects to speed up the training process on huge datasets: parallelism of model and data. The technique employed parallelism of processing unit of graphics (GPU) in data parallelism, where specimens were processed in parallel utilizing GPU. Parallelism among GPUs was achieved too,

with every processing of GPU its proper mini-batches and then combining the findings. Parallelism of model boosts RNN calculation speed by dividing jobs into time sequences and runs, accordingly.

They built effective techniques for training huge datasets more fundamentally in the works [18, 35]: to mix gradients of different GPUs, accessible letter transferring interface program was developed, on GPU, an efficient CTC calculation was shown, and a different storage management strategy was produced. In total, the model was able to train 4-20 times quicker.

5.2 An end-to-end RNN-transducer model

There are various disadvantages to utilizing CTC reducing its efficiency:

- Because systems based on CTC are predicated on the notion that each block of an output are not dependent of others, they cannot simulate the dependency between output sequences. As a result, CTC lacks the capacity to learn language models. As a result, a CTC-trained speech recognition system is an acoustic model;

- CTC maps sequences from input to output that are not longer than the provided entry.

In circumstances where the outcome is durable than the entry, CTC is absolutely ineffective.

Modeling output element dependencies is critical and has a significant influence on voice recognition systems. A RNN transducer-based model was given in the work [15]. It creates a new system that focuses on resolving the CTC issues mentioned above. It can translate any length input sequence to any length output sequence. In addition, the interdependencies between output variables may be represented.

5.2.1 The main idea of the RNN-transducer

In terms of structure, the RNN-transducer-based model is quite similar to the CTC-based model. They use the common miss appointment, for example; they fix the issues of handmade division across output and input sequences; they both employ a "black symbol" component; and they together calculate the chances of all pathways and aggregate routes to form the tag series. However, the methods of

path formation and route likelihood prediction are fundamentally different. The benefits of the Recurrent neural transducer arrive into act here.

The model based on Recurrent neural transducer has 3 key parts: a network of transcription ($F(x)$), a network of prediction ($P(y, g)$), and a combined network ($J(f, g)$). Figure 5.2 depicts the building of the RNN-transducer model.

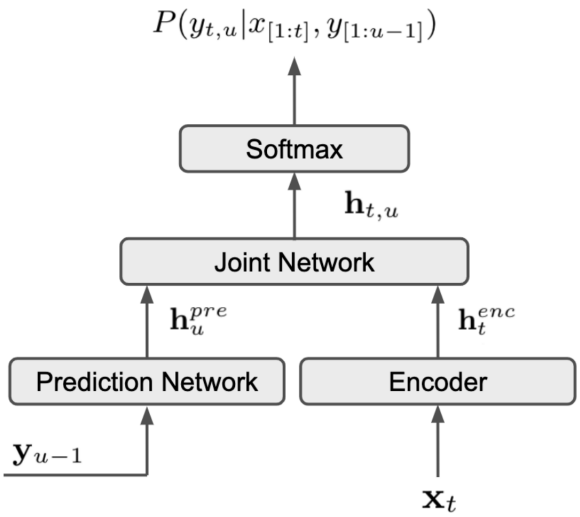

Figure 5.2: The structure of RNN-T model.

The role of each of these three components is as follows: - ($F(x)$) Transcription Network. This component serves as an encoder for an acoustic model. The function $F(x)$ converts an input sequence $X = \{x_1, ..., x_T\}$ to features $F = \{f_1, ..., f_T\}$. As a consequence, the transcribing method is applied as an acoustical model for the value x_T, yielding an outcome with a measurement rate of $|v| + 1$.

- Predicting network: This part serves as a language translator.The dependencies within auditory input are modelled by a transcription network, whereas the relationships within output sequences are modelled by a prediction network. Support the hidden state h_u and an output g_u at any place of the label $P(l)$. The following is the procedure for calculating these components:

$$h_u = H(W_{ih}l_{u-1} + W_{hh}g_{u-1} + b_h) \tag{5.3}$$

$$g_u = W_{ho}h_u + b_o \tag{5.4}$$

We can see from the combined network that the estimation of l_{u-1} is directly dependent on g_{u-1}. In basic terms, the foregoing equations state that the l_u is

defined by the $l_{[1:u-1]}$ sequence;

- $(J(f,g))$ Joint network: it solves the problem of input and output sequence alignment. The network uses the outcome of the network of transcription and the network of prediction to determine the allocation of label for $t \in [1,T], u \in [1,N]$:

$$e(k,t,u) = exp(f_t^k \ + \ g_u^k) \tag{5.5}$$

$$p(k \in V'|t,u) = \frac{e(k,t,u)}{\sum_{k' \in V'} e(k',t,u)} \tag{5.6}$$

It is obvious that $p(k \in V'|t,u)$ is an f_t and g_u based function. Knowing that f_t is derived from x_t and g_u is obtained from the sequence $\{l1,...,l_{u-1}\}$, the combined network calculates the label distribution $P(l_u|\{l_1,...,l_{u-1}\},x_t)$ at point u.

Various methods of path creation and path aggregating by Recurrent neural transducer in the base of $p(k \in V'|t,u)$ exist, and it owns an approach to assess the likelihood of a certain series [15]. Because of the existence of the route aggregation process, the Recurrent neural transducer is built as a gentle orientation technique that does not need no intentional information aligning.

The RNN-transducer decoding method is as follows: for each input x_t inside the system, it will return the label before it encounters the empty sign "-". After facing this character, it keeps to read the following element, x_{t+1}. This method is continued unless each of the entry has been read. Because the entry x_t can yield many labels, the RNN-transducer can easily overcome scenarios where the output sequence length is longer than the input sequences.

We may make the following claims based on the preceding principles that lie behind RNN-transducer:

- Because the input can issue in any size label series, the Recurrent neural transducer can translate any length entry series into any length outcome sequence;

- The RNN structure is used to build the prediction network. This means that each state is determined by the status and output of the preceding state. This allows the RNN-transducer to learn the dependencies within the output sequence, effectively modeling the language;

- To calculate the label distribution probability, the joint network employs both an auditory model and a linguistic model. As a result, by jointly training the acoustic and linguistic models, the RNN-transducer can simulate the relationships

between input and output sequence.

5.2.2 Related works

The work in [16] made tweaks to the RNN-performance transducer's in order to improve it further. They went from a simple addition to a comprehensive connectivity layer for the shared network. They raised the network's complexity and pre-trained the prediction and transcribing networks. As a consequence, they were able to reach a 17 percent PER on TIMIT data, which was a significant improvement. The experiment's comparisons revealed that training the RNN-transducer from zero is difficult. Model performance may be improved by using a pre-trained RNN-transducer.

The study in [36] delves deeper into the topic and improves the RNN-transducer in a variety of ways. They built a network of transcription model and a network of prediction model with twelve layers and with two layers of LSTM, respectively. However, it uses CTC to pre-train to produce word(s), typefaces, and phoneme(s) at the 12th, 10th, and 5th layers. Furthermore, it outputs the sub-word unit rather than the whole word. As a consequence, using Google Voice Search data, they got an 8 percent WER.

While RNN-transducer provides significant advantages over CTC, these advantages can also lead to certain restrictions. The development of unnecessary routes is one of these difficulties. Because the RNN-transducer can create many output sequences from a single input sequence, this problem is likely to arise. For example, one audio frame may create all outputs while other frames produce empty labels. In the case of RNN-transducers, these irrational process routes are obviously an exception. The recurrent neural aligner was proposed in articles [39, 13] to overcome this issue. This concept is based on the constraint of RNN-transducers having just one output for each input. However, it did not resolve the underlying problem.

5.3 A model based on Attention

On the machine translation problem, the attention-based paradigm was initially presented in [5]. Their goal was to tackle the difficulty of the encoder-decoder approach on engine decoding, which requires encrypting the text into a specific

size vector, limiting its encryption capabilities. Inside the study, the encrypter encrypts the entry into a series of vectors, and the decrypter gives not the same masses to every vector in the series using an attention mechanism at each output. The future output is dictated by the weighted summation and historical output. This approach solves the problem of encrypting all data into a restricted-dimensioned vector instead of for lengthy words to get a good encrypting impact.

Current study had a significant impact on the area of voice recognition in the next ways:

- Speech recognition may also be thought of as a sequential procedure, in which the entry is matched with the outcome. This is very analogous to the challenges with translation software;

- Data segmentation is not required for the attention converter technique. It can easily replicate fuzzy congruence across inlet and outlet patterns using attention, which is a huge step forward in speech recognition;

- Because the encrypting technique is not restricted to adjusted length vectors, it has the potential to benefit long-length sequences. It enables the processing of voice recognition with varying durations.

Due to the benefits listed above, attention-based end-to-end models have garnered a lot of appeal and interest.

Encoder, aligner, and decoder are the three components of an attention-based paradigm. The aligner component makes advantage of the attention mechanism. Each component may be improved independently.

The acoustic model, like the CTC and RNN-transducer-based models, is implemented via an encoder. The challenges that an attention-based model may encounter are the same as those encountered by CTC and RNN-transducers. The attention-based encoder, on the other hand, may run into certain difficulties.

5.3.1 The information and delay redundancy

Latency is an important challenge that the combination of an encrypter and an attention machinery may encounter. Attention is paid to the whole output of an encoder, implying that it must be waiting before an encoder has completed the entire operation before advancing. On the contrary, CTC and RNN-transducer-based models do not have this problem. The longer the encoding process takes, the more delay there is in the model. Furthermore, if the encoder does not reduce

the length of the sequence, the outcome will be significantly longer than the goals (in most cases, the input voice sequences are lengthier than the transliteration.). As a consequence, 2 problems might arise: firstly, quite enough computation for attention because the output series can be lengthy, resulting in additional waiting time; and secondly, the produced series generated by an encrypter without any random selection can create lots of unneeded data to an attention. The length of the series problem, which might arise in voice recognizing, is not taken into consideration in the engine translation complexity described in [46]. Furthermore, research in [4, 10] for an autonomous speech recognizing scheme that was built on an attention machinery did not solve this issue.

The purpose of [5] was to articulate and confront the issue for the first time. They have produced this work as a follow-up to the prior one [10]. To make the model quicker, they employed the pooling approach on numerous RNN layers. The pooling approach is used in the temporal dimension, resulting in a shorter input sequence. Between the first and second layers, an encoder's time step is cut in half. Sub-sampling or other approaches are used to achieve this decrease. The encoder has a four-layer bidirectional gated recurrent unit (GRU) structure [86], with the last two layers reading the result of the preceding layer per time step. In comparison to an initial input sequence, this resulted in a considerable 14 input sequence reduction. Sub sampling is the term for this method.

Listen, attend, and spell (LAS) presented another way in [8] for the same rationale of reducing the length of an encoder's sequence. Listener is the encoding component, which is made up of four BiLSTM layers. To decrease the number of RNN steps, each layer uses the preceding levels' two time steps as input. RNN steps can be reduced to 1/8 proportion in this method. This RNN construction is referred to as a pyramid structure. Following that, the pyramid structure has been used to an encoder in the majority of tests and works. A few of the writings shortened an input to the ratio of 1 over 4, while others lowered it to the ratio of 1 over 8.

Furthermore, there is a method for shortening the entry series well until turning on the encoder (when used immediately to the entry) [29, 44]. Employing these reduction methodologies, the duration of the encoder response is shortened, preventing delay and data repetition in the attention machinery.

5.3.2 The network structure

The attention-based encoder-decoder model is growing increasingly sophisticated in order to improve encoding capabilities, analogous to the evolution of CTC and RNN-transducer based models. The depth of a network indicates the most evident intricacy. An encoder's original structure consisted of three layers, which gradually increased to four, five, and six layers [9, 44]. The model's performance improves as the network grows more sophisticated in terms of depth. The study in [46] built a 15-layer encoder network using batch normalization, convolutional LSTM, and residual network. As a consequence, they were able to achieve a 10.5 percent WER (Word Error Rate) on the Wall Street Journal dataset with no external language model.

5.3.3 Related works

The attention mechanism is separated into three components in terms of functionality: location-based, context-based, and hybrid. Previous equations outline the technique of estimating their attention weights. They differ in terms of how information is processed and used, as well as in terms of personality:

- Context based calculates the weight for each position by using the previous concealed state and an input feature. The disadvantage of this strategy is that it does not make use of positional data. It will return the same result for features in feature sequence with various properties. The problem is known as the similarity fragment problem;

- The prior weight is utilized as location information in the location-based component to estimate the current weight. Because it does not use it, it is useless for input features;

- The hybrid component makes use of context and location-based attention. To do the computation, it considers the prior weight, previous concealed state, and input features:

$$a_u = \begin{cases} Attend(s_{u-1}, f) \\ Attend(s_{u-1}, a_{u-1}) \\ Attend(s_{u-1}, a_{u-1}, f) \end{cases} \tag{5.7}$$

In terms of the definition and functioning of the attention strategy, there are a number of challenges in speech recognition that may be handled using these distinct ways.

5.3.4 Continuity problem

When doing a machine translation operation, it is common for two nearby words in the output sequence to be placed in totally different positions in the initial sequence, and for two completely distinct words to appear in the original sequence close to each other. This circumstance, however, does not exist in a voice recognition task. Two nearby words in the transcription should match to two neighboring portions in the audio, according to the sound production. This phenomenon is known as speech recognition continuity. A location-based or hybrid attention strategy must be built to tackle the problem of continuity.

Some improvements have been made to alleviate the continuity problem based on [4, 11]. The attention mechanism is strengthened by the following expressions:

$$e_{u,t} = a(s_{u-1}, f_t) \tag{5.8}$$

$$\hat{e}_{u,t} = d(t - E_{\alpha_{u-1}}[t])exp(e_{u,t}) = d(t - \sum_{t=1}^{T} \alpha_{u-1,t}t)exp(e_{u,t}) \tag{5.9}$$

$$a_{u,t} = \frac{\hat{e}_{u,t}}{\sum_{t=1}^{T} \hat{e}_{u,t}} \tag{5.10}$$

$$c_u = \sum_{t=1}^{T} a_{u,t} f_t \tag{5.11}$$

As we can see, $a()$ and $d()$ are the functions that should be taught during training. It is evident that Equations 37 and 38 use s_{u-1}, f, and α_{u-1} in their formulations, resulting in a hybrid attention mechanism. In contrast to context-based, the formula 38 employs $d(t - E_{\alpha_{u-1}}[t])$, which allows for the punishment of attention hefts based on a bias among the prior and present location. They tackle the continuity issue in this way.

5.3.5 Monotonicity problem

The problem of monotonicity is sometimes associated with the problem of continuation, which states that sound is unidirectional in time. It is quite difficult to map a component of audio to another labeling once we have identified which element of the audio belongs to which category. In other sense, if an audio segment was connected with a certain cue through attention, that audio segment should be disregarded.

One label, in particular, may appear many times in various parts of a transcription. These labels should correspond to various audio pieces. However, it is feasible that they will attend on the same audio piece during calculating time. This problem can be solved by utilizing attention's location information.

[11] presented a method including additional loss terms in training:

$$p_u = max\{0, \sum_{t=1}^{T} CDF(u,t) - CDF(u-1,t)\} \tag{5.12}$$

where $CDF(u,t) = \sum_{j=1}^{t} a_{u,j}$ is the accumulated label allocation function at juncture u and time walk t, and $a_{u,j}$ provides the peculiarity mass at time j to create label location u. The sooner the attention mass, the more often it is calculated in p_u.

5.3.6 The issue of extracting key information

Previously, [4] observed that attention-based algorithms are only designed to detect voice data of around the same duration. When an audio file is noticeably longer than the training data, the model's performance suffers dramatically. The authors of [10] believe that the causes for this problem are as follows: firstly, a flaw of position data utilization in attention mechanisms; secondly, an attention machinist's inability to take out important essential data from peculiarities.

The initial problem, [10], built the next attention using location data:

$$s_i = Recurrency(s_{i-1}, c_{i-1}, l_{i-1}) \tag{5.13}$$

$$h_i = F \ * \ \alpha_{i-1} \tag{5.14}$$

$$e_{i,j} = \omega^T tanh(W s_i + V f_i + U h_{i,j} + b) \tag{5.15}$$

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{j=1}^{t} exp(e_{i,j})} \tag{5.16}$$

$$c_i = \sum_{t=1}^{T} \alpha_{i,j} f_i \tag{5.17}$$

We may deduce from the foregoing equations that $\alpha_i = Attend(s_{i-1}, \alpha_{i-1}, f)$ is the hybrid mechanism. Convolution, as opposed to basic hybrid attention, is used to perform an operation on location information.

To solve the second issue, [10] said that the next qualities of the softmax function, which is used to compute the masses, cause the attention machinery to fail to concentrate on important and helpful data about the function:

- Every weighted element approximated with softmax is larger than 0, implying that each characteristic is retained in the scope established with an attention. As a result, it gives undesirable bustling data;

- Softmax concentrates likelihood on big factors, causing attention to focus on one aspect while leaving additional aspects with greater impact that are not completely recalled.

Furthermore, because attention weights must be generated for each feature in the sequence at each step, this results in a large amount of work. The following techniques are recommended to address this issue:

- Increasing larger masses and reducing lesser masses by setting a soft-max to scaling meaning more than one. This strategy obscures the problem of noise performance. Nevertheless, this causes the likelihood function to concentrate on singularity items with a greater likelihood.

- After calculating them, it stores the largest number and standardizes the mass of a soft-max. This strategy increases the attention computation quantity;

- The weights of the attention are approximated only for data in a window that has been determined and using a window sliding mechanism that has been previously established. This is known as the window method.

Based on the methodology described above, the authors of [5] achieved more improvements. That time frame approach, however, was employed not just during

the reasoning step, but as well during the learning phase. Furthermore, the frame technique varies from [10]. The quality of the preceding frame is used to pick a window in [11]. This makes the model train difficult in the beginning since it might lead to a local minimum. This reliance is reduced by the fix in [5]. It handles window initialization based on data speaking speed and blank voice length, resulting in improved and more effective performance.

5.3.7 The delay

There are two causes for the delay. The first occurs when the result sequence is lengthy, which may cause the decoder to be sluggish during the computing phase. Subsampling has reduced the efficacy of this problem. The second is connected to the activity of attention on the complete outcome of an encrypting. It comes out that attention is not able to begin to operate before the entire outcome of an encrypting phase has been formed. Researchers made several attempts to remedy this issue.

To address the issue of latency, the creators of [7] created an online voice recognition system. Despite being built on a one-way RNN encoding, this module uses the windowing approach proposed by [5]. The advantage of utilizing this method is that no attention is required to expect for the encrypting to finish. As soon as that gets sufficient data to close the frame, he may start a process. As a consequence, the issue of attention latency may be managed.

The window is initially determined at some u point using the following equations:

$$m_u = median(\alpha_{u-1}) \tag{5.18}$$

$$W_u = \{f_{m_p-p}, ..., f_{m_u+q}\} \tag{5.19}$$

In the attention operation, the model only considers f_i that appears only within the window W_u:

$$s_u = RNNDecoder(l_{u-1}, s_{u-1}, c_{u-1}) \tag{5.20}$$

$$e_{u,j} = v^T tanh(\psi(s_u) + \phi(f_i) + b) \tag{5.21}$$

$$\alpha_{u,j} = \frac{exp(e_{u,j})}{\sum_{j=m_u-p}^{m_u+q} exp(e_{u,j})} \tag{5.22}$$

$$c_u = \sum_{j=m_u-p}^{m_u+q} a_{u,j} f_i \tag{5.23}$$

Adopting the window approach, [22] concluded that using a set window size is not appropriate for a real-world situation. As a result, they presented a novel window approach based on Gaussian prediction. Instead of being predetermined, the sliding step and window size are approximated based on the status of the system. This method is adaptable to any type of variation in pace.

The Gaussian window calculation is as follows:

$$s'_u = Recurrent(s_{u-1}, l_{u-1}) \tag{5.24}$$

$$\delta p_u = S_w sigmoid(V_p^T tanh(W_p s'_u)) \tag{5.25}$$

$$p_u = \delta p_u + p_{u-1} \tag{5.26}$$

$$\sigma_u = D_w sigmoid(V_\sigma^T tanh(W_\sigma s'_u)) \tag{5.27}$$

$$e_{u,j} = \begin{cases} exp(-\frac{(j-p_u)^2}{2\sigma_u^2}), j \in [1, floor(p_u + k\sigma_u)] \\ 0, others \end{cases} \tag{5.28}$$

$$a_{u,j} = \frac{e_{u,j}}{\sum_{j'=1}^{T} e_{u,j'}} \tag{5.29}$$

$$c_u = \sum_{j=1}^{T} a_{u,j} f_i \tag{5.30}$$

where p_u is the Gaussian distribution's mean and σ_u is its variance. They reflect the location and dimension at the u location. That might appear that the approximation of $a_{u,j}$ occurs in the entire output of an encrypting. Outside the window, nonetheless, $e_{u,j}$ equals zero, which substantially reduces the latency and

performs a dynamical frame.

5.3.8 The decoder

The decoding element, which is frequently built on RNNs, receives attention-generated contextual data and deciphers it into a decryption. The decoder may function in two ways structurally. Utilizing prior condition, first passes the attention machinery to the encoder's output to gather contextual data, and afterwards determines the present invisible condition of the decoder. The second assesses the decoder's present condition before calculating contextual data about the resultant series utilizing attention approach. The last choice is much popular than the previous one [7, 46, 5].

A number of research aimed to improve the decoder performance of attention-based speech recognition systems.

The authors of [29] revealed that the decoder has only one layer and is not organized for language characteristics. As a result, they constructed the decoder as follows:

$$p_i = f(l_{i-1}, p_{i-1}) \tag{5.31}$$

$$s_i = f(p_i, s_{i-1}, c_i) \tag{5.32}$$

$$p(l_i|l - i - 1, ..., l_1, c_i) = g(s_i, c_i) \tag{5.33}$$

In [29], they added a layer based on the above equation to a basic decoder structure. It operates on historical output, implying that it functions entirely as a language model. This enables the decoder to operate on longer timeframes.

The authors of [19] developed a multi-head decoder approach. Different attentions are employed in this technique to obtain the context information of the encoding result and decoding. The results of these attentions are combined into a single final product.

In terms of human perception of the world, attention technology has been widely researched and investigated. For every problem that arises, new solutions and analyses emerge. However, several studies conducted by a variety of researchers reveal that attention-based models outperform CTC and RNN-

transducer-based models.

5.3.9 Summary

With respect to several aspects, in the below is a juxtaposition and derivation of 3 popular end-to-end model types:

- Regarding latency, CTC's method of training is able to predict the likelihood of decrypter outcome with no having to wait for the encoding result, resulting in a very fast delay period. The Recurrent neural transducer, in contrast, is comparable to the CTC. Nevertheless, since the RNN converter employs decoder result from the preceding walk in the forward-inverse logic except the encoder output, the latency time may be larger than in CTC. They both enable streaming recognition, which means the output is generated in real time as the user speaks. Because it must expect before the full output of the encrypter is formed until it may perform a decoding and alignment, the attention-based approach has a considerable delay time. This is not an instance of streaming recognition;

- CTC has an $O(T)$ computing complexity for predicting the probability of length N based on features with length T, because each time step requires a softmax operation. The complexity of the RNN-transducer is due to the need to conduct a softmax on each input and output portion $O(TN)$. Traditional attention mechanisms, such RNN-transducers, need an operation on both the entry and outcome parts, that adds to the $O(NT)$ complexity. However, because the attention mechanism uses a windowing approach, its complicatedness is decreased because it only reuires to operate on entries within a frame;

- Capability of modeling language : Because CTC assumes that entry and outcome are not dependent, it does not require a model of language. Attention based models accomplish this capability through the use of a decoder, whereas RNN-transducers get this capability through the use of a joint network;

- Training capability: CTC is essentially a basic loss function, which means there are no weights to train with. CTC's sole purpose is to train an acoustic model. As a result, CTC may easily get ideal results. Many functions and weights must be taught in models based on RNN-transducer and attention technologies. It is substantially more difficult to train acoustic and linguistic models together than it is to train CTC models alone. Furthermore, a combined network based on an RNN-transducer generates a lot of undesired data alignment between the input

and output. As a result, training attention-based systems is significantly more difficult. In order to get a promising performance, it usually requires pre-trained transcription and prediction networks;

- The precision of the RNN-transducer and attention mechanism is far superior to that of a simple CTC. The performance of an attention-based model, on the other hand, is generally better than that of an RNN-transducer-based model.

6. Experiments & Results

6.1 Experiments

It was attempted to train two models that were comparable. Google colaboratory was used to train these models. The structure of the first model is as follows:

1) Use a vocabulary size of 50 for the embedding layer;

2) A 50-neuron LSTM layer with L2 regularization;

3) 0.5 value dropout regularization approach;

4) A 50-neuron LSTM layer with L2 regularization;

5) 0.5 value dropout regularization approach;

6) For feature extraction, a dense layer with 100 neurons and a relu activation function is used;

7) A single vector predicts the next word in the output layer.

Instead of LSTM, the second model uses a BiLSTM model:

1) Use a vocabulary size of 50 for the embedding layer;

2) L2 regularized BiLSTM layer with 50 neurons;

3) 0.5 value dropout regularization approach;

4) L2 regularized BiLSTM layer with 50 neurons;

5) 0.5 value dropout regularization approach;

6) For feature extraction, a dense layer with 100 neurons and a relu activation function is used;

7) A single vector predicts the next word in the output layer.

When the performance of two distinct models was compared, the second model with BiLSTM layer was picked since it produced highly promising results when compared to the model with LSTM layer in Figure 6.1. The loss was dramatically increased by the LSTM model at some point. This atypical behavior can lead to issues such as overfitting, which is not found in the BiLSTM model.

Name	Loss	Accuracy	Epochs
LSTM	1.70	59.43	100
BiLSTM	1.32	73.71	100

Figure 6.1: The performance comparison of LSTM and BiLSTM

Tensorflow, a Python package, was utilized to train the CTC model. Actually, there are several frameworks for developing ASR systems, like as Kaldi and Sphinx. The benefit of Tensorflow is that it allows us to leverage the computer's GPU to parallelize jobs into tensors.

The Neural Network's structure is based on an RNN (Recurrent Neural Network) with 100 hidden layers. For collecting all information in the sequence, an LSTM (Long-Short Term Memory) kind of RNN was utilized. Three algorithms (Adam, Momentum, and Adagrad) were evaluated for the optimization component, as previously described. Each offers benefits over the other two, however for speech data, the LER (label error rate) and CTC loss were used to assess these methods.

The dataset has undergone data augmentation using the temporal warping approach before being fed into the neural network. The model will be trained over 100 epochs. The CTC model's output is a series of characters that includes blanks (encoding part). The decoding is done using a dynamic programming approach called greedy search in this example. Greedy search was chosen over beam search because it decodes quicker, despite the fact that beam search performs better. At prevent the problem of overfitting, the learning rate was set to 0.005.

6.2 Results

The following results were obtained after training the model three times using various optimization strategies.

6.2.1 Adagrad optimizer

Following then, the loss value begins to fluctuate dramatically between 500 and 50. (approximately). At the same time, LER begins to hesitate between 1 and

0.9 (roughly), which prevents the model from learning.

6.2.2 Momentum optimizer

Momentum performs far superior to Adagrad. For training and validation sets, LER and CTC loss are constantly lowering. After the gradient steps reach 900 epochs, the CTC loss approaches one, while the LER remains zero. Because of the large difference between the training and validation sets, this may be prone to overfitting. This difficulty may be avoided in the future by using alternative optimization approaches such as dropout and inputting even more data. Because the learning rate is equal to 0.005, the declining process slows somewhat. The learning process is far superior to Adagrad optimizer.

6.2.3 Adam optimizer

CTC loss decreases at the start of the gradient steps, and the optimizer, like in Adagrad, begins to pause between two values with a large difference. LER, on the other hand, drops to 1 after a few repetitions and then remains constant for a lengthy gradient step. After roughly 750 iterations, LER begins to hesitate between 1 and 0.5 (about), which is not a good performance.

Name	Training cost	Validation cost	Training LER	Validation LER
Adagrad	166.774	242.164	0.970	0.980
Momentum	2.405	71.289	0.000	0.500
Adam	166.774	242.164	0.970	0.980

Figure 6.2: The results of each optimizer.

7. Conclusion & Future Work

This work clearly demonstrates the superiority of Momentum optimizer over Adam and Adagrad for CTC algorithm for voice recognition. The experiment demonstrated that the model with the Momentum optimizer learns quicker, lowering the CTC loss and LER after each gradient step, but the Adagrad and Adam optimizers performed extremely badly, exhibiting a hesitation of mistakes from large to tiny. Aside from that, this work demonstrates the sophisticated method for voice detection known as Connectionist Temporal Class in action. It also outlines the obvious advantages of this approach over the previous technique, which is based on HMMs, which include simplicity and efficacy.

It is intended to include further regularization techniques in the future to make our CTC method perform much quicker and more accurately. To overcome issues like as overfitting and ballooning gradients, it is intended to use methods such as dropout and residual blocks, as well as gather additional data.

References

[1] Dario Amodei et al. "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 173–182.

[2] Kartik Audhkhasi et al. "Building Competitive Direct Acoustics-to-Word Models for English Conversational Speech Recognition". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 4759–4763.

[3] Kartik Audhkhasi et al. "Direct Acoustics-to-Word Models for English Conversational Speech Recognition". In: Aug. 2017, pp. 959–963. DOI: 10.21437/Interspeech.2017-546.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *ArXiv* 1409 (Sept. 2014).

[5] Dzmitry Bahdanau et al. "End-to-end attention-based large vocabulary speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 4945–4949. DOI: 10.1109/ICASSP.2016.7472618.

[6] Peter F. Brown et al. "Class-Based n-gram Models of Natural Language". In: *Computational Linguistics* 18.4 (1992), pp. 467–480. URL: https://aclanthology.org/J92-4003.

[7] William Chan and Ian Lane. "On Online Attention-Based Speech Recognition and Joint Mandarin Character-Pinyin Training". In: *Proc. Interspeech 2016*. 2016, pp. 3404–3408. DOI: 10.21437/Interspeech.2016-334.

[8] William Chan et al. “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 4960–4964. DOI: 10.1109/ICASSP.2016.7472621.

[9] Chung-Cheng Chiu et al. “State-of-the-art Speech Recognition With Sequence-to-Sequence Models”. In: (Dec. 2017).

[10] Jan Chorowski et al. “Attention-Based Models for Speech Recognition”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 577–585.

[11] Jan Chorowski et al. “End-to-end continuous speech recognition using attention-based recurrent nn: First results”. English (US). In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[12] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[13] Linhao Dong et al. “Extending Recurrent Neural Aligner for Streaming End-to-End Speech Recognition in Mandarin”. In: Sept. 2018, pp. 816–820. DOI: 10.21437/Interspeech.2018-1086.

[14] Florian Eyben et al. “From speech to letters - using a novel neural network architecture for grapheme based ASR”. In: *2009 IEEE Workshop on Automatic Speech Recognition Understanding*. 2009, pp. 376–380. DOI: 10.1109/ASRU.2009.5373257.

[15] Alex Graves. “Sequence Transduction with Recurrent Neural Networks”. In: *ArXiv* abs/1211.3711 (2012).

[16] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE Inter-*

national Conference on Acoustics, Speech and Signal Processing (2013), pp. 6645–6649.

[17] Alex Graves et al. "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 369–376. ISBN: 1595933832. DOI: 10.1145/1143844.1143891. URL: https://doi.org/10.1145/1143844.1143891.

[18] Awni Hannun et al. "DeepSpeech: Scaling up end-to-end speech recognition". In: (Dec. 2014).

[19] Tomoki Hayashi et al. "Multi-Head Decoder for End-to-End Speech Recognition". In: *ArXiv* abs/1804.08050 (2018).

[20] H. Hermansky, D.P.W. Ellis, and S. Sharma. "Tandem connectionist feature extraction for conventional HMM systems". In: *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*. Vol. 3. 2000, pp. 1635–1638. DOI: 10.1109/ICASSP.2000.862024.

[21] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Comput.* 18.7 (July 2006), pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. URL: https://doi.org/10.1162/neco.2006.18.7.1527.

[22] Junfeng Hou, Shiliang Zhang, and Lirong Dai. "Gaussian Prediction Based Attention for Online End-to-End Speech Recognition". In: *INTERSPEECH*. 2017.

[23] F. Jelinek. "Continuous speech recognition by statistical methods". In: *Proceedings of the IEEE* 64.4 (1976), pp. 532–556. DOI: 10.1109/PROC.1976.10159.

[24] Yerbolat Khassanov et al. "A Crowdsourced Open-Source Kazakh Speech Corpus and Initial Speech Recognition Baseline". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguis-

tics, Apr. 2021, pp. 697–706. DOI: 10.18653/v1/2021.eacl-main.58. URL: https://aclanthology.org/2021.eacl-main.58.

[25] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[26] C.J. Leggetter and P.C. Woodland. "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models". In: *Computer Speech & Language* 9.2 (1995), pp. 171–185. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.1995.0010. URL: https://www.sciencedirect.com/science/article/pii/S0885230885700101.

[27] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition". In: *The Bell System Technical Journal* 62.4 (1983), pp. 1035–1074. DOI: 10.1002/j.1538-7305.1983.tb03114.x.

[28] Jie Li et al. "Towards end-to-end speech recognition for Chinese Mandarin using long short-term memory recurrent neural networks". In: *16th Annual Conference of the International Speech Communication Association*. Sept. 2015, pp. 3615–3619. DOI: 10.21437/Interspeech.2015-717.

[29] Liang Lu, Xingxing Zhang, and Steve Renais. "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 5060–5064. DOI: 10.1109/ICASSP.2016.7472641.

[30] Andrew L. Maas et al. "First-Pass Large Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs". In: *ArXiv* abs/1408.2873 (2014).

[31] Orken Mamyrbayev et al. "End-to-End Model Based on RNN-T for Kazakh Speech Recognition". In: *2021 3rd International Conference on Computer Communication and the Internet (ICCCI)*. 2021, pp. 163–167. DOI: 10.1109/ICCCI51764.2021.9486811.

[32] Orken Mamyrbayev et al. “Identifying the influence of transfer learning method in developing an end-to-end automatic speech recognition system with a low data level”. In: *Eastern-European Journal of Enterprise Technologies* 1.9(115) (Feb. 2022), pp. 84–92. DOI: 10.15587/1729-4061.2022.252801. URL: http://journals.uran.ua/eejet/article/view/252801.

[33] Roger K. Moore. “A comparison of the data requirements of automatic speech recognition systems and human listeners”. In: *INTERSPEECH*. 2003.

[34] Saida Mussakhojayeva, Yerbolat Khassanov, and Huseyin Atakan Varol. “A Study of Multilingual End-to-End Speech Recognition for Kazakh, Russian, and English”. In: *Speech and Computer*. Ed. by Alexey Karpov and Rodmonga Potapova. Cham: Springer International Publishing, 2021, pp. 448–459. ISBN: 978-3-030-87802-3.

[35] Rohit Prabhavalkar et al. “A Comparison of Sequence-to-Sequence Models for Speech Recognition”. In: *Proc. Interspeech 2017*. 2017, pp. 939–943. DOI: 10.21437/Interspeech.2017-233.

[36] Kanishka Rao, Hasim Sak, and Rohit Prabhavalkar. “Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer”. In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)* (2017), pp. 193–199.

[37] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[38] Hasim Sak et al. “Fast and accurate recurrent neural network acoustic models for speech recognition”. In: *INTERSPEECH*. 2015.

[39] Haşim Sak et al. “Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping”. In: Aug. 2017, pp. 1298–1302. DOI: 10.21437/Interspeech.2017-1705.

[40] Hagen Soltau, Hank Liao, and Haşim Sak. “Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition”. In: *Interspeech 2017*. ISCA, Aug. 2017. DOI: 10.21437/interspeech.2017-1566. URL: https://doi.org/10.21437%5C%2Finterspeech.2017-1566.

[41] Will Song. "End-to-End Deep Neural Network for Automatic Speech Recognition". In: 2015.

[42] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[43] A. Waibel et al. "Phoneme recognition using time-delay neural networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (1989), pp. 328–339. DOI: 10.1109/29.21701.

[44] Chao Weng et al. "Improving Attention Based Sequence-to-Sequence Models for End-to-End English Conversational Speech Recognition". In: *INTERSPEECH*. 2018.

[45] Y. Zhang et al. "Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks". In: *INTERSPEECH*. 2016.

[46] Yu Zhang, William Chan, and Navdeep Jaitly. "Very deep convolutional networks for end-to-end speech recognition". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 4845–4849. DOI: 10.1109/ICASSP.2017.7953077.

[47] Geoffrey Zweig et al. "Advances in all-neural speech recognition". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 4805–4809. DOI: 10.1109/ICASSP.2017.7953069.