Speech Command Recognition: Text-to-Speech and Speech Corpus Scraping Are All You Need

Askat Kuzdeuov
Institute of Smart Systems and AI
Nazarbayev University
Astana, Kazakhstan
askat.kuzdeuov@nu.edu.kz

Shakhizat Nurgaliyev
Institute of Smart Systems and AI
Nazarbayev University
Astana, Kazakhstan
shakhizat.nurgaliyev@nu.edu.kz

Diana Turmakhan
Institute of Smart Systems and AI
Nazarbayev University
Astana, Kazakhstan
diana.turmakhan@nu.edu.kz

Nurkhan Laiyk
Institute of Smart Systems and AI
Nazarbayev University
Astana, Kazakhstan
nurkhan.laiyk@nu.edu.kz

Huseyin Atakan Varol
Institute of Smart Systems and AI
Nazarbayev University
Astana, Kazakhstan
ahvarol@nu.edu.kz

2023 3rd International Conference on Robotics, Automation and Artificial Intelligence (RAAI) | 979-8-3503-0756-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/RAAI59955.2023.10601292

***Abstract*—Speech Command Recognition (SCR) is rapidly gaining prominence due to its diverse applications, such as virtual assistants, smart homes, hands-free navigation, and voice-controlled industrial machinery. In this paper, we present a data-centric approach to creating SCR systems for low-resource languages, particularly focusing on the Kazakh language. By leveraging synthetic data generated by Text-to-Speech (TTS) and data extracted from a large-scale speech corpus, we successfully created the Kazakh language equivalent of the Google Speech Commands dataset. Moreover, we also compiled the Kazakh Speech Commands dataset with data collected from 119 participants. This dataset was used to benchmark the performance of the Keyword-MLP model trained using our synthetic dataset. The results showed that the model achieves 89.79% accuracy for the real-world data demonstrating the efficacy of our approach. Our work can serve as a recipe for creating customized speech command datasets, including for low-resource languages, obviating the need for laborious and costly human data collection.**

***Index Terms*—Speech commands recognition, text-to-speech, Kazakh Speech Corpus, voice commands, data-centric AI**

I. Introduction

Speech Command Recognition (SCR) aims to develop computational systems for interpreting and executing human voice instructions [1]. This technology facilitates the direct interaction between humans and machines by transcending traditional input modalities such as keyboards and touchscreens, enabling a more intuitive, efficient, and accessible mode of communication. SCR has become increasingly relevant in recent years due to its widespread adoption in virtual assistants like Amazon's Alexa and Apple's Siri, smart home devices, hands-free navigation systems, and voice-based interaction with industrial devices.

While both automatic speech recognition (ASR) and SCR focus on interpreting human voice inputs, there are key differences. ASR converts continuous spoken language into its corresponding textual representation [2] and is typically designed to handle various languages, accents, and speaking styles. On the other hand, SCR is a more specialized subset of speech recognition that focuses on identifying and executing specific voice commands within a limited vocabulary and a predefined context, allowing for more efficient processing, reduced complexity, and quicker response times. In terms of hardware requirements, ASR systems usually demand more powerful computational resources, while SCR systems can often operate on less resource-intensive hardware, e.g., edge devices and embedded systems.

The development of SCR systems requires the use of diverse and representative speech command datasets for model training and evaluation. The Google Speech Commands Dataset (GSCD) [3], available in two versions, V1 and V2, offers an extensive collection of English keywords. V1 contains over 64,000 recordings of 30 distinct keywords, while V2 expands the dataset to over 100,000 recordings and 35 keywords also introducing more variations in background noise. This diverse sample of human speech commands has made the GSCD a frequently used benchmark for evaluating SCR performance.

Deep neural network approaches have significantly advanced the field of SCR. For instance, researchers developed a convolutional recurrent network with attention, achieving 94.1% and 94.5% accuracy on the 20-commands recognition task for the GSCD V1 and V2, respectively [4]. Subsequently, Berg et al. introduced the Keyword Transformer (KWT), a fully self-attentional architecture that further improved the state-of-the-art performance on the same datasets [5]. Similarly, another group of researchers created the Audio Spectrogram Transformer (AST), a convolution-free, purely attention-based model that demonstrated state-of-the-art results on various benchmarks, including the GSCD V2 dataset [6]. Morshed et al. explored the use of gated multilayer perceptrons as an attention-free alternative to the KWT and achieved comparable performance while being more parameter-efficient [7]. Efficient end-to-end networks and extensive audio augmentations were utilized to achieve state-of-the-art results across various

2023 3rd International Conference on Robotics, Automation and Artificial Intelligence (RAAI)

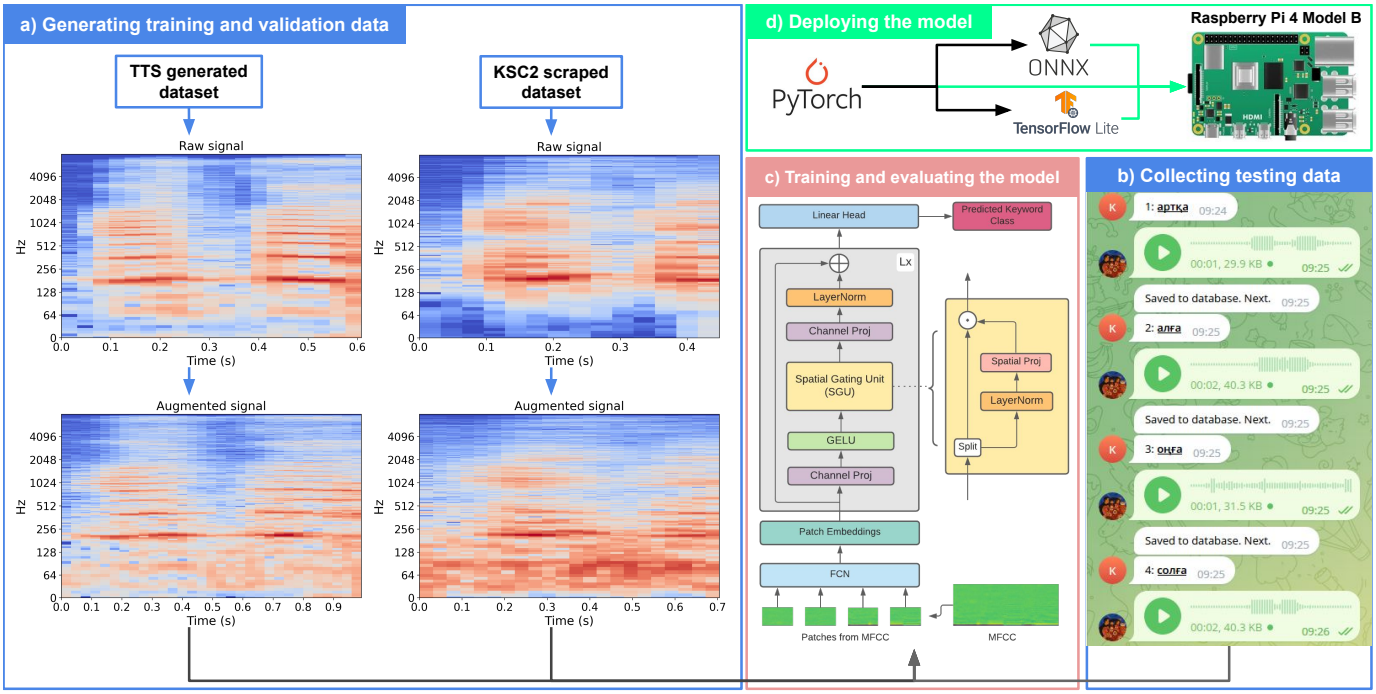

Figure 1. Speech Command Recognition Pipeline: a) Synthetic speech command dataset generation using TTS and general-purpose corpus scraping, b) Data collection for the Kazakh Speech Commands Benchmark dataset, c) Deep learning model training and evaluation, and d) Model deployment on a single-board computer leveraging ONNX and TensorFlow Lite runtime optimizations.

sound classification tasks, including SCR in [8]. Recently, Niizumi et al. proposed the Masked Modeling Duo (M2D) method, a self-supervised learning approach that achieved high performance on a number of benchmarks, e.g., GCSD V2 [9].

Data-centric approaches have emerged as a promising direction to improve SCR performance. Huang et al. demonstrated the effectiveness of multi-speaker text-to-speech (TTS) as a data augmentation technique for keyword spotting, and by optimizing the synthesized speech to better represent the target speaker's speaking style, they achieved significant improvements [10]. Researchers explored the use of synthesized speech data in training spoken term detection models and showed that models trained on synthetic speech exhibited comparable accuracy to models trained on a much larger amount of real recordings [11]. Werchniak et al. conducted an initial investigation into the use of TTS audio as a "helper" tool for training keyword spotters in low-resource settings and found that carefully mixing TTS audio with human speech during training led to substantial error reductions [12]. Novel techniques for curating wake word engine (WWE) datasets by leveraging "found data" from public sources and synthetic data generated through TTS and voice conversion were presented in [13], achieving comparable performance to WWEs trained with data collected from humans.

The advancement of SCR in languages other than English has been hindered by the lack of comprehensive datasets. However, due to the increasing need for SCR systems, several works have emerged in various languages. Hung et al. developed a Vietnamese speech command dataset consisting of 15 commands and reported a recognition accuracy of 98.19% using recurrent neural networks [14]. An Italian Forestry Speech Commands dataset and baseline SCR results were presented in [15]. Anindya et al. developed a deep neural network-based Indonesian SCR system for six commands, obtaining 99.43% accuracy on testing data and 89.57% in actual conditions [16].

In this work, we integrate TTS-based synthetic data generation, which has been frequently employed for SCR, with other data sources to enhance SCR performance in low-resource languages. General-purpose speech corpora for ASR are available for many languages, with projects such as CommonVoice leading the way as a crowdsourced global effort. By leveraging ASR systems with timestamping capabilities [17], these general-purpose corpora can be scraped for keywords, providing a valuable data source for developing SCR systems. Our primary contribution is the creation of a Kazakh SCR system using data obtained from both TTS and general-purpose speech corpus scraping (see Fig. 1). This approach eliminates the time-consuming and labor-intensive speech command dataset collection and can serve as a general recipe for the development of SCR systems in other low-resource languages. Additionally, we introduce the Kazakh Speech Commands Dataset i.e., Kazakh versions of the 35 commands in the GSCD obtained from 119 subjects. This dataset aims to serve as a benchmark for evaluating SCR models in languages other than English, further promoting research and development in this critical area. We have made the source code, dataset, and pre-trained models publicly available at

https://github.com/IS2AI/Kazakh-Speech-Commands-Dataset.

The remainder of this paper is organized as follows: Section II outlines the methodology for generating a synthetic dataset using TTS models, extracting keywords from a large-scale speech corpus, and collecting a real dataset. Section III offers insights into model training, evaluation, and implementation on an edge device. Following that, Section IV discusses the experimental results, and finally, Section V presents a conclusion of our research.

II. Methodology

To create a set of commands for the Kazakh Speech Commands dataset, we translated 35 English commands in GSCD V2-35 to Kazakh. However, two commands in GSCD V2-35, "marvin" and "sheila", were replaced with "read" and "write" commands due to the unavailability of direct translations for these terms in the Kazakh language. The translated commands and their phonetic transcriptions are given in Table I. The GSCD V2-35 includes 105,829 samples of 35 commands, involving 2,618 unique speakers in the data collection process. Each utterance is stored as a WAV format file with a maximum duration of one second, and the sample data is encoded using linear 16-bit single-channel PCM values at a 16 kHz sampling rate. Our objective is to generate a dataset of equivalent size for the Kazakh language by generating synthetic data using TTS models and extracting data from speech corpora. To validate the efficacy of our approach, we also collected real-world data from 119 participants for testing purposes.

The pipeline of our approach is shown in Fig. 1. Initially, we generated training and validation data by using TTS models and scraping a comprehensive speech corpus. To expand the dataset, we applied various augmentations to the raw audio signals. Subsequently, to demonstrate the effectiveness of our approach, we collected real-world data via a Telegram bot for testing purposes. Then, we trained and evaluated one of the state-of-the-art SCR models on our dataset. Ultimately, we incorporated the model into an edge device with limited computational capacity, thereby illustrating potential applicability in industrial settings. The details of each stage are provided in the subsequent sections.

A. Synthetic Data Generation Using TTS

In order to create a synthetic speech commands dataset, we leveraged the TTS models presented in [18], consisting of five speakers' vocalizations, with two males and three females. To increase the number of utterances for each command, we applied a variety of augmentation techniques. In the beginning, we adjusted the textual configuration of the commands, prompting the TTS models to yield divergent pronunciations for identical words. For each command, we generated one textual augmentation. For instance, the command "артқа" was augmented as "артқа**аа**" with the additional letters at the end. Then, the original and augmented text commands were converted into speech commands using the five TTS models, resulting in 10 samples per command.

Algorithm 1 KSC2 Speech Command Scraping Algorithm

```
# loop over all N utterances in KSC2
for i = 0 to N do
    # read the utterance and transcription
    utterance = utterances[i]
    transcription = transcriptions[i]
    # split the transcription into words
    words = split(transcription)
    # loop over the words
    for j = 0 to M do
        # if the word is one of the commands
        if word[j] in commands then
            # pass the utterance to the Vosk speech recognition
            # model to recognize words with their timestamps - ts
            [rec_words, ts] = Vosk(utterance)
            # trim the word in the utterance if
            # it was recognized by the model
            if word[j] in rec_words then
                command = trim(utterance, ts)
                # save the trimmed audio in a WAV format
                # with 16 kHz sampling rate
                save(command, wav, 16kHz)
            end if
        end if
    end for
end for
```

To increase the dataset size further, we applied audio augmentation techniques such as time stretching, pitch shifting, and adding background noise. As background noise, we utilized environmental noise and white Gaussian noise to simulate real-world environments. We employed the ESC-50 dataset for the background noise component, which consists of a collection of 2,000 environmental audio samples, categorized into 50 semantically distinct classes [19]. As an example, the log spectrograms of raw audio signals before and after augmentations are shown in Fig. 1a. It can be seen that the raw audio signals were stretched and merged with randomly selected background noise signals. In total, we generated 2,430 synthetic utterances per command, resulting in 85,050 utterances in total. This amount is close to the combined training and validation set of GSCD V2-35. The generated speech commands were saved in a WAV format with a maximum duration of one second and a 16 KHz sampling rate.

B. Scraping the Kazakh Speech Corpus 2 Dataset

The Kazakh Speech Corpus 2 (KSC2) represents the premier open-source speech corpus for the Kazakh language [20]. KSC2 contains data collected from diverse sources such as TV shows, radio broadcasts, senate sessions, and podcasts. KSC2 encompasses roughly 1,128 hours of high-quality transcribed content, corresponding to over 520 thousand utterances with 16 kHz frequency. We scraped the KSC2 dataset to identify transcriptions containing at least one command of interest. In order to automatically trim the identified commands from the utterances, we utilized the Vosk speech recognition model [17]. This model has several advantages, such as rapid inference on a central processing unit (CPU), provision of

Table I
KAZAKH SPEECH COMMANDS DATASET: LIST OF COMMANDS, DATASET STATISTICS, AND THE RESULTS OF THE BEST MODEL

Kazakh Speech Commands			**Dataset Statistics**				**Classification Results (%) of the Best Model**			
Google's commands	Kazakh commands	Phonetic scripts	Synthetic TTS	Extracted KSC2	Processed KSC2	Collected Real	Precision score	Recall score	F1 score	Most confused command:score
backward	артқа	ɑrtˈqɑ	2,430	110	2,430	113	93.97	96.46	95.20	forward: 0.9
forward	алға	ɑlˈɣɑ	2,430	944	2,430	112	87.50	87.50	87.50	six: 3.6
right	оңға	wɔŋˈɣɑ	2,430	58	2,430	106	83.51	76.42	79.80	**read: 16.0**
left	солға	sɔlˈɣɑ	2,430	32	2,430	104	87.04	90.38	88.68	right: 4.8
down	төмен	tөˈmʲen	2,430	1,391	2,430	102	89.72	94.12	91.87	dog: 2.9
up	жоғары	ʒɔɣɑˈrɤ	2,430	4,949	2,430	104	96.77	86.54	91.37	read: 4.8
go	жүр	ʒʏr	2,430	1,883	2,430	101	83.49	90.10	86.67	learn: 3.0
stop	тоқта	tɔqˈtɑ	2,430	47	2,430	107	94.17	90.65	92.38	read: 6.5
on	қос	qɔs	2,430	967	2,430	101	78.18	85.15	81.52	**bird: 12.9**
off	өшір	wөˈʃɪr	2,430	2	338	105	100.0	91.43	95.52	**go: 7.6**
yes	иә	iˈjæ	2,430	1,294	2,430	110	81.89	94.55	87.76	go: 0.9
no	жоқ	ʒɔq	2,430	15,374	2,430	107	100.0	90.65	95.10	left: 2.8
learn	үйрен	ʏjˈrʲen	2,430	15	1,815	108	87.50	90.74	89.09	yes: 5.6
follow	орында	wɔrʏnˈdɑ	2,430	494	2,430	104	90.83	95.19	92.96	right: 2.9
zero	нөл	nөl	2,430	2,146	2,430	105	91.75	84.76	88.12	one: 5.7
one	бір	bɪr	2,430	50,922	2,430	107	91.95	74.77	82.47	zero: 5.6
two	екі	ʲeˈkɪ	2,430	43,426	2,430	99	89.36	84.85	87.05	yes: 5.1
three	үш	ʏʃ	2,430	17,811	2,430	107	100.0	90.65	95.10	bird: 5.6
four	төрт	tөrt	2,430	13,332	2,430	97	94.87	76.29	84.57	down: 3.1
five	бес	bʲes	2,430	19,544	2,430	104	98.94	89.42	93.94	write: 1.9
six	алты	ɑlˈtɤ	2,430	10,529	2,430	101	93.14	94.06	93.60	forward: 2.0
seven	жеті	ʒʲeˈtɪ	2,430	10,350	2,430	103	100.0	89.32	94.36	two: 3.9
eight	сегіз	sʲeˈgɪz	2,430	12,147	2,430	103	98.02	96.12	97.06	two: 1.9
nine	тоғыз	tɔˈɣɤz	2,430	16,170	2,430	100	97.83	90.00	93.75	on: 4.0
bed	төсек	tөˈsʲek	2,430	226	2,430	97	96.84	94.85	95.83	eight: 2.1
bird	құс	qʊs	2,430	567	2,430	96	76.42	84.38	80.20	**on: 15.6**
cat	мысық	mɤˈsɤq	2,430	90	2,430	97	93.14	97.94	95.48	wow: 2.1
dog	ит	it	2,430	423	2,430	102	64.97	100.0	78.76	-
happy	бақытты	bɑqɤtˈtɤ	2,430	775	2,430	101	96.88	92.08	94.42	read: 4.0
house	үй	ʏj	2,430	3770	2,430	107	100.0	56.07	71.86	**dog: 35.5**
marvin → read	оқы	wɔˈqɤ	2,430	35	2,430	105	67.57	95.24	79.05	backward: 1.9
sheila → write	жаз	ʒɑz	2,430	342	2,430	105	90.43	99.05	94.55	up: 1.0
tree	ағаш	ɑˈɣɑʃ	2,430	566	2,430	104	98.04	96.15	97.09	backward: 1.0
visual	көрнекі	kөrnʲeˈkɪ	2,430	22	2,430	100	90.91	100.0	95.24	-
wow	мәссаған	mæssɑˈɣɑn	2,430	12	1,452	99	95.10	97.98	96.52	left: 1.0

starting and ending timestamps for recognized words, offline functionality, and compatibility with the Kazakh language.

A pseudocode of the KSC2 scraping algorithm is presented in Algorithm 1. Each transcription in the KSC2 was examined to ascertain if it contained any command of interest. If at least one of the commands was detected, the corresponding utterance was processed using the Vosk speech recognition model to recognize the command and the respective timestamps within the utterance. Subsequently, the detected commands were trimmed using the timestamps and saved in a WAV format with a 16 kHz sampling rate. The statistics of the extracted commands are available in Table I. In total, 228,765 samples were extracted for 35 commands. However, the dataset exhibited a considerable degree of imbalance among the commands. In order to make its the size same as the synthetic TTS dataset, commands with more than 2,430 samples were randomly downsampled, while the others were undersampled utilizing the augmentation techniques employed for the synthetic TTS generation. As a result, the processed KSC2 dataset contains 81,365 samples. The number of samples per command is given in Table I. It is crucial to note that only two samples were found for the "off" command in the KSC2 dataset. Thus, after applying all augmentations, a total of 338 samples were obtained for this command. Therefore, we over-sampled this command to balance it with other commands.

C. Kazakh Speech Commands Benchmark Dataset

In order to validate the efficacy of our approach, we collected a real-world speech commands dataset for testing purposes via a Telegram bot (see Fig. 1b). In the beginning, the bot requests a user to input gender, age, and region. Following this, the bot sequentially presents commands to the user for reading. The user can rerecord incorrectly recorded utterances. The recordings are saved in a WAV format with a sampling rate of 16 kHz. In total, 119 participants (62 males, 57 females) from different regions of Kazakhstan took part in data collection. The collected dataset underwent a manual evaluation by two moderators to remove any subpar samples, including incomplete or incorrect readings, as well as quiet or silent recordings. The statistics of the collected dataset are provided in Table I. The dataset exhibits a balanced distribution and contains 3,623 samples in total.

Table II
ACCURACY (%) RESULTS OF THE MODELS

Training and validation dataset	Training From Scratch			Transfer Learning		
	Training set	Validation set	Testing set (real)	Training set	Validation set	Testing set (real)
Synthetic TTS	90.83	86.82	70.30	93.88	90.40	82.50
Processed KSC2	85.55	83.51	68.53	88.46	84.74	75.38
Synthetic TTS + Processed KSC2	87.39	87.63	85.62	88.53	88.64	**89.79**

III. EXPERIMENTS

A. Dataset Splitting

The synthetic TTS and processed KSC2 datasets were split into training and validation sets. In generating the synthetic TTS dataset, we used the voices of five speakers (3 females, and 2 males). Consequently, the data generated by two males and two females were designated as training data, while the remaining female voice was assigned to the validation data. As a result, the synthetic training and validation sets consist of 75,600 and 9,450 samples, respectively. The processed KSC2 dataset was partitioned in a manner that ensured augmented samples in the training and validation sets were derived from distinct, non-overlapping samples. The training and validation sets contain 74,891 and 6,474 samples, correspondingly. The collected real-world dataset (3,623 samples) was employed only for model testing.

B. Model Selection and Training

In this study, the Keyword-MLP model [21] was utilized for SCR. This model was chosen due to its high accuracy (97.56%) on the test set of GSCD V2-35, which is comparable to state-of-the-art models, such as KWT [22] and AST [6]. Most importantly, the Keyword-MLP model offers enhanced parameter efficiency, with 424,811 parameters, making it suitable for deployment on edge devices with limited computing resources. We employed two distinct model training strategies. The first approach entailed training from scratch with random initial weights, while the second involved transfer learning. We trained the Keyword-MLP model on the GSCD V2-35 dataset using the default hyperparameters. The model achieved 94.58% on the training set, 97.39% on the validation set, and 97.5% on the test set of the GSCD V2-35 dataset. The results were close to the ones presented in the original paper [21]. In both training strategies, the model was trained on synthetic TTS, processed KSC2, and a combination of these two datasets. In all cases, the model was tested on the collected real-world dataset. The models were trained on a single NVIDIA A100 GPU for 100 epochs with a batch size of 256. The augmentation rates were tuned for each training strategy.

C. Model Deployment on an Edge Device

We examined the performance of various machine learning frameworks, such as ONNX Runtime and TensorFlow Lite, in comparison to the original PyTorch model. ONNX is a package developed by Microsoft, which is designed to optimize inference and provide a cross-platform engine capable of running on both CPU and GPU. On the other hand, TensorFlow Lite is an open-source deep learning framework developed by Google, which enables efficient on-device inference, minimizing performance loss and memory cost without significant accuracy degradation. In order to obtain ONNX and TensorFlow Lite models, the original PyTorch model (.pth) underwent a series of conversions: initially to the ONNX format (.onnx), then to the TensorFlow format (.pb), and finally to the TensorFlow Lite format (.tflite). The models were deployed on a Raspberry Pi OS 64-bit with Python 3.9, TensorFlow 2.12, ONNX Runtime 1.14.1, and Torch 1.12.1. The models were compared with each other in terms of size and inference time.

IV. RESULTS & DISCUSSION

The model, trained solely on the synthetic TTS dataset, achieved 90.83% accuracy on the synthetic training set, 86.82% on the synthetic validation set, and 70.3% on the real-world test set (see Table II). Employing transfer learning further enhanced the accuracy for each set, with the model reaching 93.88% on the synthetic training set, 90.4% on the synthetic validation set, and 82.5% on the real-world test set. Although we can generate a large-scale synthetic dataset, the underlying source is still five speakers' voices. Therefore, the models tend to overfit quickly. To address this issue, we applied spectral augmentation methods such as time masking and frequency masking during the training steps.

When the model was trained on the processed KSC2 dataset, it achieved 85.55% and 83.51% on the training and validation sets of processed KSC2, and 69.53% accuracy on the real-world test set. Incorporating transfer learning further enhanced these results, with the model obtaining 88.46% accuracy on the training set, 84.74% on the validation set, and 75.38% on the real-world test set. In both cases, the model's accuracy was lower compared to when it was trained only on the synthetic dataset. The reason is that the extracted speech commands exhibit shorter duration due to their pronunciation within sentences. Specifically, the extracted KSC2 commands possess an average duration of 0.27 s, while the synthetic TTS and real-world commands have means of 0.75 s and 1.17 s, respectively. Furthermore, the articulation of commands may exhibit variations when they are uttered in isolation compared to their enunciation within the context of sentences.

When the model was trained using a combination of the synthetic TTS and processed KSC2 datasets, it attained 87.39% accuracy on the combined training set, 87.63% on the combined validation set, and 85.62% on the real-world test set.

Employing transfer learning further enhanced the outcomes, with the model achieving 88.53% accuracy on the training set, 88.64% on the validation set, and 89.79% on the real-world test set. These results indicate that the synthetic TTS dataset and processed KSC2 dataset complement each other, leading to a more accurate model.

The classification results for the best-performing model are presented in Table I, highlighting the five most frequently confused commands in bold. For instance, the model often mistakes the command "bird" (құс/qʊs) for the command "on" (қос/qɔs) in 15.6% of cases and vice versa in 12.9% of cases. The main factors for this confusion are the short length and similarity of the commands, as they differ by a single phoneme. Additionally, the model confuses "house" (үй/ʏj) with "dog" (ит/it) in 35.5% of instances. The primary cause for this confusion is that both commands are quite brief, consisting of only two phonemes, and the pronunciation of "house" (үй/ʏj) in the Kazakh language can be challenging even for native speakers. Consequently, many people pronounce this word differently, leading to the model's confusion. Furthermore, the model often misclassifies "right" (оңға/wɔŋˈɣɑ) as "read" (оқы/wɔˈqʏ) due to the similar sounds of these words (i.e., homophones). In a similar vein, the model confuses "off" (өшір/wөˈʃɪr) with "go" (жүр/ʒʏr) in 7.6% of cases.

This study also assessed the inference time of the SCR model across various formats, including Pytorch, ONNX, and TensorFlow Lite, on a Raspberry Pi 4. The original PyTorch model had the largest file size at 5.2 MB, followed by TensorFlow Lite at 2.6 MB, and ONNX with the smallest size at 1.7 MB. Ten one-second utterances were processed using each model, and the inference time was documented for every command. The original PyTorch model exhibited an average inference time of 25.2 ms with a 1.1 ms standard deviation. The ONNX model showed no improvement with a slower 33.9 ms average and a 14.1 ms standard deviation. The TensorFlow Lite model also did not improve the PyTorch model's inference time by achieving an average of 32.5 ms with a standard deviation of 1.5 ms. These results suggest that the model can run in real time, making it suitable for various industrial applications.

V. Conclusion

In this research, we introduced a methodology for creating a speech commands dataset using synthetic data generated by TTS and data gathered from a general speech corpus. This method removes the necessity for time-consuming human data collection and facilitates the generation of various speech commands without additional data acquisition. Experimental results indicated that our approach can effectively be employed to develop accurate SCR models. Our real-world dataset can be used as a benchmark for assessing models trained on synthetic and scraped data, as well as for evaluating different audio augmentation strategies. Furthermore, this work offers a comprehensive pipeline for developing a speech recognition model, encompassing data collection, model training and evaluation, and deployment on edge devices. To promote further research in this field, we have made our source code, pre-trained models, and dataset publicly available.

References

[1] U. Kamath, J. Liu, and J. Whitaker, *Deep Learning for NLP and Speech Recognition*, 1st ed. Springer Publishing Company, 2019.

[2] M. Malik, M. K. Malik, K. Mehmood *et al.*, "Automatic speech recognition: A survey," *Multimedia Tools and Applications*, vol. 80, no. 6, pp. 9411–9457, Mar 2021.

[3] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *ArXiv e-prints*, Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.03209

[4] D. C. de Andrade, S. Leo, M. L. D. S. Viana *et al.*, "A neural attention model for speech command recognition." *CoRR*, 2018.

[5] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword Transformer: A Self-Attention Model for Keyword Spotting," in *Proc. Interspeech 2021*, 2021, pp. 4249–4253.

[6] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," in *Proc. Interspeech 2021*, 2021, pp. 571–575.

[7] M. M. Morshed and A. O. Ahsan, "Attention-free keyword spotting," 2021.

[8] A. Gazneli, G. Zimerman, T. Ridnik *et al.*, "End-to-end audio strikes back: Boosting augmentations towards an efficient audio classification network," 2022.

[9] D. Niizumi, D. Takeuchi, Y. Ohishi *et al.*, "Masked modeling duo: Learning representations by encouraging both networks to model the input," 2023.

[10] H. Huang and B. Qian, "Speaking style compensation on synthetic audio for robust keyword spotting," in *Proc. of the International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2022, pp. 448–452.

[11] J. Lin, K. Kilgour, D. Roblek *et al.*, "Training keyword spotters with limited and synthesized speech data," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7474–7478.

[12] A. Werchniak, R. B. Chicote, Y. Mishchenko *et al.*, "Exploring the application of synthetic audio in training keyword spotters," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7993–7996.

[13] B. Ramanan, L. Drabeck, T. Woo *et al.*, "Eliminating data collection bottleneck for wake word engine training using found and synthetic data," in *Proc. of the IEEE International Conference on Big Data (Big Data)*, 2019, pp. 2447–2456.

[14] P. D. Hung, T. M. Giang, L. H. Nam *et al.*, "Vietnamese speech command recognition using recurrent neural networks," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 7, 2019. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2019.0100728

[15] L. Ortenzi, G. Colle, C. Costa *et al.*, "Italian speech commands for forestry applications," in *Proc. of the IEEE International Workshop on Metrology for Agriculture and Forestry*, 2021, pp. 401–405.

[16] C. Anindya, D. Purwanto, and D. I. Ricoida, "Development of indonesian speech recognition with deep neural network for robotic command," in *Proc. of the International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2019, pp. 434–438.

[17] N. V. Shmyrev, "Vosk Speech Recognition Toolkit," 2023. [Online]. Available: https://github.com/alphacep/vosk-api

[18] S. Mussakhojayeva, Y. Khassanov, and H. A. Varol, "KazakhTTS2: Extending the open-source Kazakh TTS corpus with more data, speakers, and topics," in *International Conference on Language Resources and Evaluation (LREC)*, 2022.

[19] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proc. of the Annual ACM Conference on Multimedia*. ACM Press, 2015, pp. 1015–1018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390

[20] S. Mussakhojayeva, Y. Khassanov, and H. A. Varol, "KSC2: An Industrial-Scale Open-Source Kazakh Speech Corpus," in *Proc. Interspeech 2022*, 2022, pp. 1367–1371.

[21] M. M. Morshed and A. O. Ahsan, "Attention-free keyword spotting," 2022.

[22] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword Transformer: A Self-Attention Model for Keyword Spotting," in *Proc. Interspeech 2021*, 2021, pp. 4249–4253.