Kazakh-Russian Sign Language Recognition using MediaPipe and Recurrent Neural Networks

Shyngyskhan Abilkassov
School of Engineering and Digital Sciences
Nazarbayev University
Astana, Kazakhstan

Asset Malik
School of Engineering and Digital Sciences
Nazarbayev University
Astana, Kazakhstan

Madi Nurmanov
School of Engineering and Digital Sciences
Nazarbayev University
Astana, Kazakhstan

Anara Sandygulova
School of Engineering and Digital Sciences
Nazarbayev University
Astana, Kazakhstan

***Abstract*—Sign language is an essential tool for communication for the hearing impaired or deaf people. This part of population is overlooked, while being a significant fraction of it. The development of technologies for interpretation of sign language could enormously benefit those people. The purpose of this work, therefore, is to develop of a system, that could act as a mediator tool for communication between deaf and hearing people. In this report the implementation of Kazakh-Russian sign language recognition algorithms based on Google's MediaPipe and Recurrent Neural Networks is described. MediaPipe is a framework which allows feature extraction from input graphical source in the form of data key-points, the sequence of which constructs basic elements of sign language. Afterwards, output is synthesized as a textual information based on the result of processing by Recurrent Neural Network. The data set of Kazakh-Russian Sign Language was kindly provided by the Human-Robot Interaction laboratory at Nazarbayev University. The performance of the resulting system is established to be at 88.74% of accuracy.**

***Index Terms*—Sign language recognition, Mediapipe, RNN, LSTM, Human-Robot Interaction.**

I. Introduction

Even in a contemporary world society widely integrated with high-end technologies and innovations, some groups of people are still struggling. Some of them regard themselves as members of a cultural minority who use sign language. This is hugely affecting their social, economic, and emotional development. The problems of deaf community are misunderstood and often underrepresented. Relatively low incidence (5%) which is about 466 million people is a primary reason for that. People around the world that are facing the issues caused by Hard of Hearing and 1 billion people in the age of 12-35 are at risk. The World Health organization estimates [1] that by 2050 the number of people with impaired hearing is expected to increase up to 1 billion as shown in Fig. 1.

The consequences of muteness are social isolation, psychological pressure, economical instability and educational gap in comparison with hearing population. Current solutions for the existing problem are ineffective or not popular amongst the deaf community, because devices are impractical, expensive or difficult to maintain [1].

Nevertheless, there is a possibility of usage of sign language recognition system for people with complete hearing loss, or for whom current solutions are unsatisfactory.

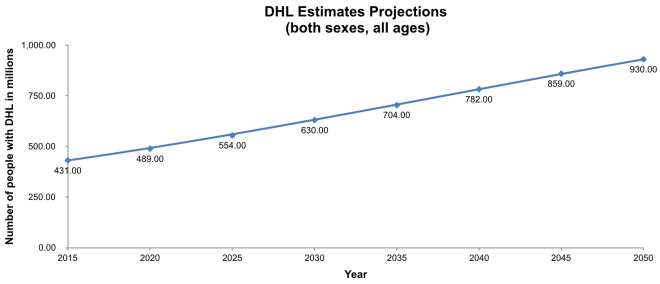

Figure 1. WHO estimates on Disabling Hearing Loss by 2050

The solution proposed in this report is a user centered human-computer interaction system. It should be of semi-autonomous nature, in the sense that it will exploit pretrained machine learning models for on display sign language interpretation.

Current state of the art research that focuses on the machine learning application in the discussed field is not a novelty. Most of the modern classification and recognition techniques for visual events are done using Convolutional Neural Network and are great for identifying features like finger position and direction [2]. However, the use of CNNs for sign lanugage recognition has its own drawbacks. Very deep convolutional neural networks are required to classify sign language directly from images, and it also requires powerful computation units and lot of training time. Another approach of finding features without training the CNN model would be use of OpenPose [3] library. Although the OpenPose is a powerful tool, it is very computationally expensive and can run on a limited number of powerful devices. The solution this report suggest is rather different, in the sense that it uses Google's MediaPipe: Framework for perceptional pipelines [4]. It uses pretrained model for targeted features' extraction, which is, for instance, is based on 30 thousand manually annotated images of human palms. Such approach gives a high precision of 95% in hand detection applications, which can be essential in terms of sign language recognition. Moreover, MediaPipe supports more platforms that OpenPose does and the performance in terms of FPS in dramatically different, as MediaPipe can be used even in mobile applications and on embedded devices. This in its

order, allows to integrate sign language recognition algorithms into robot's computation units and develop a mobile user-oriented interfaces.

This paper presents Kazakh-Russian Sign Language recognition method that involves the usage of MediaPipe library as feature extractor and LSTM RNN as a classification model to convert local video or a video stream directly into a text end-to-end.

II. Related work

This part of the report discusses current trends in sign language recognition and the research development of machine learning application in the field.

A. Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective

The work [5] by Danielle Bragg et al. authored by a team of researchers from Microsoft and partner institutions, this paper gives a brief summary of the technology level applied towards the sign language recognition and translation as of 2019. It serves to orient readers both within and outside of computer science to the field, highlights opportunities for interdisciplinary collaborations, and helps the research community prioritize which problems to tackle next. For instance, deaf community involvement is considered crucial for the purpose of user oriented system development, as well as a security measure for the right of ownership of sign language by deaf community. The report also calls for multidisciplinary approach for the solution of the problem, as the existing methods are not real world oriented, they don't deal with the domain's constraints and consequently the resulting systems are not well structured.

B. MediaPipe: A framework for building perception pipelines

This paper [4] is a first introduction of a MediaPipe open-source software developed by Google. The major contribution of this work is the ease of use, balanced resource consumption, and robustness towards edge cases of machine learning algorithms in the MediaPipe pipeline. This allows for developers to prototype cross-platform computer vision applications and focus on the algorithm or model development. Moreover, it contributes to the reproducibility in the machine learning research field. MediaPipe can be used to build a perception pipeline as a a graph of separate modules combining model inference, data transformation and augmentation, and media processing. Any visual or sensory data type enters the graph and the required landmark stream exits it as shown in Fig. 2.

C. MediaPipe Hands: On-device Real-time Hand Tracking

This paper [6] represent approach of Google researchers in the VR/AR application for real-time hand tracking and skeletonization as shown in Fig. 3. Authors propose the usage of software based solutions for hand detection instead of hardware, since depth information, which is the main tool of hardware approach, is not essential. It can be retrieved from the RGB image as 2.5D representation of the hand. This is a

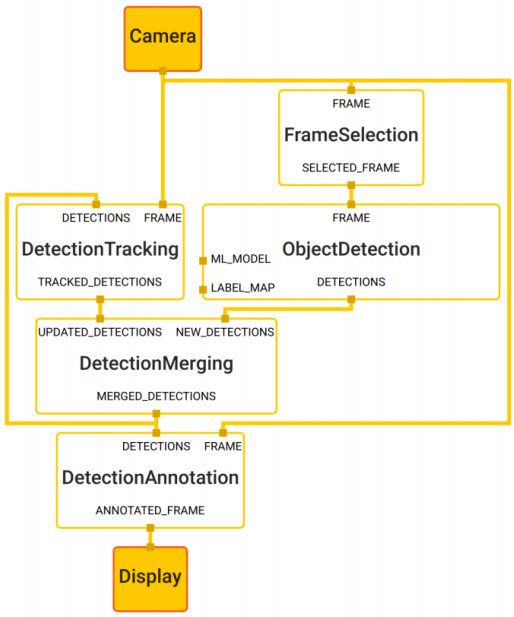

Figure 2. Architecture for object detection using MediaPipe by Lugaresi et al.

unique feature of the proposed method, as well as it is more robust in performance in comparison with other architectures.

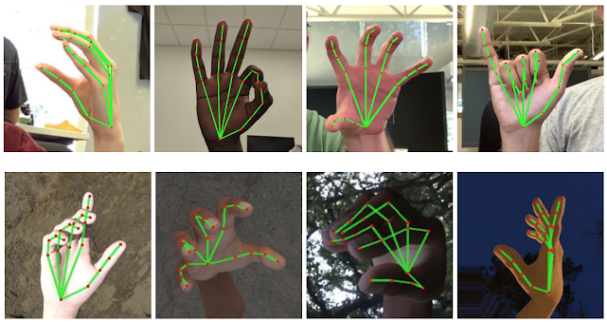

Figure 3. Examples of Zhang et al. datasets. (Top): Annotated real- world images. (Bottom): Rendered synthetic hand images with ground truth annotation

To detect the bounding box for the hand, authors use BlazePalm detector - a single shot detector, as the articulation of the fingers is initially not required. Afterwards, Feature Pyramid Networks and minimization of focal loss are applied, which makes the detection contrast invariant and more precise based on the visual features. The architecture is represented by Fig. 4

The handlandmark is then processed using regression algorithm. Model is trained based on both synthetic and real-world images, while the datapoints were annotated manually on more than 16 thousand frames and using commercial model on 100 thousand synthetic hand images with varying light, texture and angle with the respect to camera. The resulting system is then

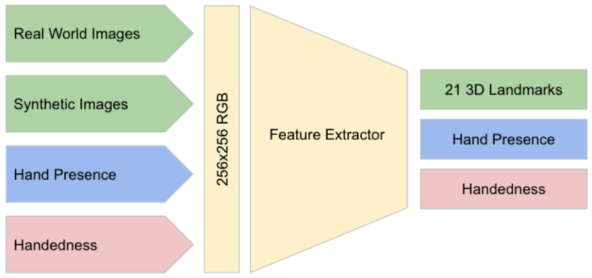

Figure 4. Architecture for hand landmark extraction by Zhang et al.

modeled using MediaPipe graphs as shown in Fig. 5, and then it can be used as the ready solution for rapid prototyping.

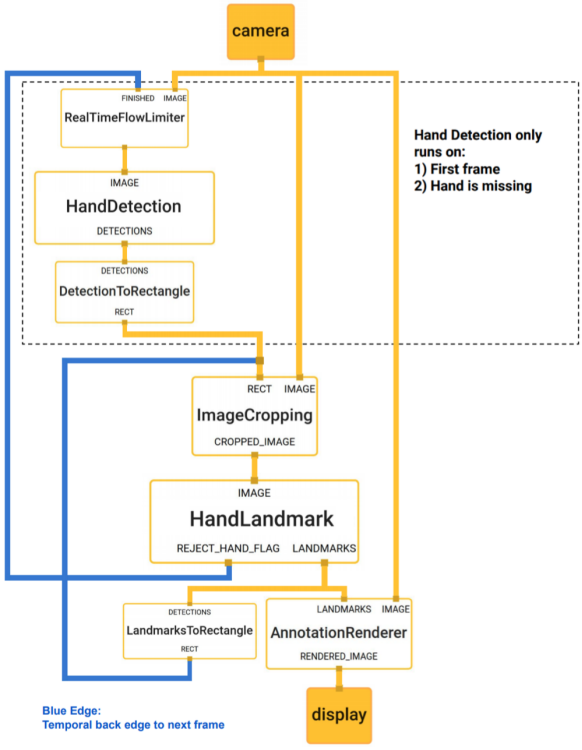

Figure 5. MediaPipe graph for hand landmark extraction by Zhang et al.

III. Materials and Methods

A. Used Databases

A lot of data is required to train the model that is able robustly identify signs the person have shown in the video. This means that it is required to gather videos of different people showing various signs multiple times in different lighting conditions, camera position, and signing speed. To do so, we gained access to three different sign language datasets each with their unique features and content.

1) Isolated signs: Initially, the Isolated Signs dataset was used for the training process. The dataset consists of 20 isolated sign classes where each one represents the word that is an independent part of speech, a pronoun. Depending on the context, each word is an ancillary part of speech, a union or a speech particle. Some of the words contained in the dataset also contain a question mark at their end and are used in the sentence to represent an interrogative sentence. Moreover, videos in this dataset are much shorter in length comparing to other datasets. The classes and corresponding number of videos are represented in Table I.

Table I
Isolated Signs Dataset Summary

Isolated sign dataset							
Class	#total	#train	#val	Class	#total	#train	#val
1: gde	260	234	26	11: kto	257	231	26
2: gdeq	260	234	26	12: ktoq	258	232	26
3: kak	254	229	25	13: kuda	260	234	26
4: kakoi	262	236	26	14: kudaq	259	233	26
5: kakoiq	260	234	26	15: skolko	259	233	26
6: kakq	258	232	26	16: skolkoq	260	234	26
7: kogda	260	234	26	17: what	260	234	26
8: kogdaq	262	236	26	18: whatq	260	234	26
9: kotoriy	320	288	32	19: zachem	260	234	26
10: kotoriyq	238	214	24	20: zachemq	275	248	28
Total					5242	4718	524

2) Spread The Sign: Spread The Sign [7]is a useful website used to store and teach sign language for everyone that is interested in those. It has a lot of words in different languages including Russian. One video per every class trained was picked in order to test our models.

3) Kazakh Russian Sign Language: Kazakh Russian Sign Language (KRSL) dataset created and provided by the HRI-lab at Nazarbayev University [8]. KRSL is the biggest among all three, containing about 40,000 videos in total that weight around 200 GB as shown in Table II. Those gigabytes include 175 phrases, with 50 person per phrase and 5 videos per person. We have downloaded only 123 out of 175 phrases to local PC II, which took us more than three days and 150GB of storage. The act of moving this data from PC to external HDD took around 6 hours. This is truly massive data-set. Pre-processing this data before training is not an option but a requirement. We suggest use of modern file-systems to store the data as any simple action like changing the directory on a same HDD may cause unintended duplication of the files. If it is possible there is also an option to use RAID array to increase HDD read-write speeds. Usage of solid state drives with large cache is another option, as solid state drives with small cache can cause bottleneck of the cache and throttle down the write speeds to low flash speeds that are commonly slower than modern HDDs.

4) Short list: To test if the architecture on works on KRSL, we have chosen 10 small phrase and trained the model on them. Carefully picked 10 words were about 1GB folder each with sum of 10.21GB

B. MediaPipe feature extraction

MediaPipe is a very useful tool that was have used as an alternative for OpenPose library. It is capable of extracting

Table II
GOOGLE DRIVE FOLDER WITH KRSL DATA

name	size	name	size	name	size	name	size	name	size
0	913M	25	1.1G	50	1.5G	77	1.4G	153	1.2G
1	735M	26	1.7G	51	1.0G	78	1.4G	154	1.4G
2	1.2G	27	1.2G	52	1.2G	79	1.7G	155	1.3G
3	1.2G	28	1.2G	53	1.1G	80	1.3G	156	1.4G
4	1.2G	29	1.4G	54	781M	81	1.3G	157	1.2G
5	1.1G	30	1.2G	55	1.0G	82	1.5G	158	1.2G
6	1.2G	31	1.2G	56	1.1G	83	1.7G	159	971M
7	1.2G	32	1.2G	58	1.1G	84	1.1G	160	953M
8	1.4G	33	1.2G	59	906M	85	940M	161	859M
9	1.1G	34	1.3G	60	862M	86	1.0G	162	987M
10	1.1G	35	1.4G	61	1021M	87	861M	163	1.1G
11	1000M	36	1.3G	62	948M	88	927M	164	1.2G
12	1.2G	37	2.3G	64	1.1G	89	1.0G	165	2.2G
13	1.0G	38	1.2G	65	1.1G	90	1.1G	166	1.2G
14	1.3G	39	1.6G	66	1.4G	91	1.4G	167	1.2G
15	934M	40	1021M	67	1.3G	92	1.4G	168	1.3G
16	1.1G	41	1.3G	68	1.4G	93	1.3G	169	1.1G
17	965M	42	1.3G	69	1.5G	94	1.8G	170	1.1G
18	1.2G	43	1.0G	70	1.4G	95	1.8G	171	1.3G
19	1.1G	44	1013M	71	1.3G	96	1.2G	172	1.2G
20	1.3G	45	1.0G	72	1.4G	97	1.0G	173	1.1G
21	1.1G	46	1.1G	73	1.3G	98	1.0G	174	1.2G
22	1.1G	47	1.0G	74	1.6G	99	1.0G	175	1.7G
23	1.0G	48	1.3G	75	1.2G	151	1.5G		
24	1.4G	49	987M	76	1.4G	152	1.1G		
Total	150GB								

Table III
SHORT LIST KRSL DATASET SUMMARY

#orig	#in DB	Class	#total	#train	#val
0	1	1: Здравствуйте	250	225	25
1	2	2: Привет	250	225	25
43	43	3: Доброе утро	250	225	25
44	44	4: Добрый день	250	225	25
45	45	5: Добрый вечер	250	225	25
46	46	6: Спокойной ночи	250	225	25
54	54	7: До свидания	250	225	25
160	159	8: Извините меня	250	225	25
161	160	9: Простите	250	225	25
162	161	10: Мне жаль	250	225	25
Total			2500	2250	250

feature points of person's hands from a video, locating them and classifying the spatial orientation of each finger separately. Use of MediaPipe freed accelerated the work, since it provided a ready framework and training separate convolutional neural network was not required. It helped to focus directly on development of algorithm of recognizing the signs.

1) Feeding video from file: Initial MediaPipe graph demo architecture takes images taken from camera and draws landmark points directly on image. However, the training process requires data to be generated, classified, and shuffled in order to be fed into the training pipeline. This requires some updates to the internal MediaPipe structure. Therefore, updated internal calculators that output landmark data into txt files was used [9]. The landmark data from video input was generated afterwards. The sample image with landmarks drawn by MediaPipe is represented in Fig. 6.

2) Running on GPU: While running our first data set we immediately figured out that each class takes a lot of time to process and processing for a KRSL dataset could take up to 16 days. Even though it was running on high end Ryzen CPU with six cores and 12 threads all clocked at 4Ghz frequency each. Best option would be to use of graphic accelerated

Figure 6. Hand feature landmarks shown on a video from the KRSL dataset

processing directly on GPU, with even more benefit from CUDA accelerated GPUs.

3) Use of DGX: Professor and research assistant provided the access to the DGX server. But it was impractical to use it due to inability to use apt package manager that is required for MediaPipe installation process. Even though it had 16 core, most of them were already at 100% utilization. This is, perhaps, due to solving the tasks from another professors and research groups, as the access is shared. Also downloading the 200GB dataset to external PC without GUI from google drive was a problem and we eventually decided that it wouldn't worth it due to lack of GPU acceleration on code and already high demand of DGX's CPU.

4) Generated data: After running MediaPipe feature extraction on datasets, the list of folders containing txt files was generated. As it can be observed, each txt corresponds to a particular video and stores landmarks obtained from them. At each time instance, maximum number of 84 data points can be obtained, where 21 are number of landmark points as shown on Fig. 7, each with a corresponding x and y coordinates, for two hands. Alongside the txt MediaPipe also generates video outputs that are exactly same in content as the input but with overlayed hand position marker and features for debugging and demonstration purposes. It is unknown whether disabling this action can increase the performance of the algorithm, but was ignored due to low priority of investigation. The output videos are also typically lower in weight, with almost 85% reduction in size for KRSL input.

C. RNN on LSTM cells

As the data used is a sequence of landmarks, the Recurrent Neural Network (RNN) model was used to train a recognition algorithm. Recurrent networks were first mentioned in the work of Rumelhart et al. [10] and is the best suitable architecture as the sign language is a sequence of hand motions usually called glosses and the recognition algorithm must store and use the information in sequence. RNN is basically a network of cells that have a feedback loops and can store any information. However, the most prominent type of RNNs that proved its effectiveness are Long short-term memory (LSTM) models,

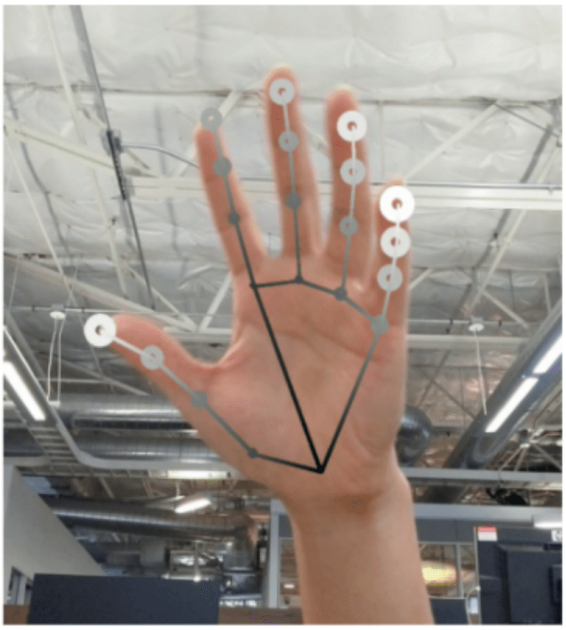

Figure 7. Hand feature landmarks shown on a video stream

which were initially introduced by Hochreiter and Schmidhuber [11] and updated by multiple researchers worldwide. LSTMs are designed in such a way to overcome the main disadvantage of classical RNNs and are able to avoid the long-term dependency problem and therefore are able to store information for long periods of time. All RNNs are a chain of repeating modules of neural network with a simple structure like a single tanh activation function. However, LSTMs have a number of internal gates in single neural network layer which allows them to preserve information for long periods.

1) Architecture 1: Initially a simple model with three LSTM layers with 32 parameters was used to test and see if it actually works. The model was trained on for 100 epochs and it was found that the network actually work, and a maximum validation accuracy of 52% was achieved.

2) Architecture 2: The final architecture consists of seven LSTM layers each consisting of 64 nodes. The first six have a return sequences mode turned on which allows to pass the network output of the full sequence of hidden states to the last layer. The model architecture summary is represented in Table IV. Total trainable parameter number is 237,588.

IV. Results

The training was made on Isolated Signs dataset which consists of 20 classes, each corresponding to a particular word. These words are shown in Table I. Some of the words had a pair with words that have a question mark in the end. Therefore, some classes were of high similarity with only minor gloss difference. The input to the model is a vector consisting of 84 data points. Each hand has 21 landmarks, with point having x and y coordinates, made for two hands. As there were some videos where only one hand visible, the left data points were filled with zeros. The training data was split into two parts: training and validation. Validation data used was 10% of the whole data. The final training was made for 250 epochs which took only approximately 2 hours of GPU training on a laptop. The final accuracy achieved was 88.74%. Afterwards, training also was done using the short list data from the KRSL dataset shown in Table III, and training accuracy peaked at around 70% accuracy with 100 of epoch for training. Nevertheless it has shown that the RNN can be trained on a more complex data as multiple words and collocations, but more data and time for training is required.

Table IV
Final LSTM architecture summary

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 70, 64)	38144
lstm_1 (LSTM)	(None, 70, 64)	33024
lstm_2 (LSTM)	(None, 70, 64)	33024
lstm_3 (LSTM)	(None, 70, 64)	33024
lstm_4 (LSTM)	(None, 70, 64)	33024
lstm_5 (LSTM)	(None, 70, 64)	33024
lstm_6 (LSTM)	(None, 64)	33024
dense (Dense)	(None, 20)	1300

V. Discussion and Conclusion

The resulting system is a prototype of Kazakh-Russian Sign Language recognition pipeline. It uses a video as an input and produces text output as a result of several processing techniques application. Accuracy of the solution, which fluctuates around 88% is a good result. However it can cause some undesirable artifacts or create sources of miscommunication. Usage of larger datasets in combination with further tuning of the parameters of neural networks can contribute in the overall improvement of the system. Another essential application of larger dataset processing is expansion of the dictionary that the system operates and interprets. Currently, the short list is consisting only of 10 phrases, which is a small part of KRSL dataset. Including the total of 173 phrases from KRSL and combining them with other sources can influence the performance of the system and possible application. Afterwards, the question of testing the system in the real-world conditions rises, as it is required for more rigorous accuracy estimation. However, it all will be possible in less limited time constraint and with the introduction of more computational power. COVID-19 pandemics resulted in the work from home situation, where equipment is not suitable for efficient processing of large amount of data. GPU computations would drastically improve the training process, but at the moment of this report submission they were not available.

VI. Future work

As an implication for future, it is proposed to finish the processing of the KRSL dataset. Also, depth information might

be useful and additional research must be done on the topic. It was not done due to limitation in computation power and time constraints. Proposed future solution is usage of GPU based machine learning computations, since MediaPipe supports it and shows substantial performance increase in comparison with CPU only applications. It may require usage of Nvidia GPU with CUDA library support, as it is more efficient with MediaPipe in comparison with analogues. Considering the proposed MediaPipe + RNN model, it is fairly robust, but is primarily used to identify separate words and may fail to identify phrases. More testing and expansion of applied dataset is required. As the language may have separate ways to say same word, it is also proposed to create some support for many-to-many relation of the glosses to words or even phrases.

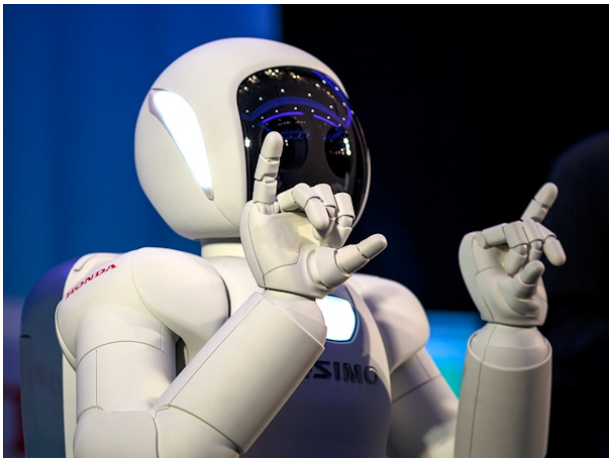

Figure 8. Sign language by Honda's ASIMO robot

Moreover, one of the ideas for future work is to change the format of the data output from MediaPipe. The initial used approach for landmark storage works best for pose estimation algorithms [12], where the number of landmarks is constant, but for the hand landmark storage as the number of landmark varies (from 0 to 84). Although the training went smooth and the network is even able to classify images from completely different environment, it is best to change the format of data output from MediaPipe. Initially, the list of landmarks is stored in the list linearly, and the data about which point belongs to which hand (if any detected) is lost during that process. However, the MediaPipe can obtain that information and it would be better to store that information and if none hand detected, fill the rest with null data during that process, rather than pre-processing for RNN training. It seems reasonable that it could potentially increase the final classification accuracy. Finally, the main goal of the future work is to develop an end-to-end robot mediator that could be able to become a bridge between signers and non-signers as shown in Fig. 8. Moreover, as the work uses the MediaPipe software for the whole pipeline, its features allows to integrate this software into mobile devices. This in fact would allow to close the gap between the people and let the deaf people communicate freely in any situation.

ACKNOWLEDGMENT

Our team expresses deep gratitude for the members of Human - Robot Interaction laboratory at Nazarbayev University for provision of Kazakh - Russian sign language dataset, Isolated Signs dataset, and compiled version of videos from Spread The Sign website. We also would like to mention professor Anara Sandygulova and Medet Mukushev for their support and vision throughout the execution of this work.

REFERENCES

[1] A. E. I. Vaughan, *Deafness and hearing loss*, 2020 (accessed October 17, 2020). [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[2] K. Bantupalli and Y. Xie, "American sign language recognition using deep learning and computer vision," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 4896–4899.

[3] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1812.08008*, 2018.

[4] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee *et al.*, "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[5] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudrealt, A. Braffort, N. Caselli, M. Huenerfauth, H. Kacorri, T. Verhoef, C. Vogler, and M. R. Morris, "Sign language recognition, generation, and translation: An interdisciplinary perspective," 2019.

[6] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[7] M. Hilzensauer and K. Krammer, "A multilingual dictionary for sign languages:" spreadthesign," *Alpen-Adria-Universität Klagenfurt*, 2015.

[8] A. Sandygulova, *Human-Robot Interaction research @ Nazarbayev University*, Accessed November 29, 2020. [Online]. Available: https://www.hrilab-nu.com/home

[9] J. Kim, *RNN for Human Activity Recognition - 2D Pose Input*, Accessed November 10, 2020. [Online]. Available: https://github.com/rabBit64/Sign-language-recognition-with-RNN-and-Mediapipe/tree/master/modified_mediapipe

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] S. Eiffert, *RNN for Human Activity Recognition - 2D Pose Input*, Accessed November 30, 2020. [Online]. Available: https://github.com/stuarteiffert/RNN-for-Human-Activity-Recognition-using-2D-Pose-Input/blob/master/LSTM.ipynb