A Lightweight Downscaled Approach to Automatic Speech Recognition for Small Indigenous Languages

George Vlad Stan
g.v.stan@proton.me
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

André Baart
andre@andrebaart.nl
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

Francis Dittoh
fdittoh@uds.edu.gh
University for Development Studies
Tamale, Ghana

Hans Akkermans
hansakkermans@akmc.nl
University for Development Studies
Tamale, Ghana

Anna Bon
a.bon@vu.nl
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

ABSTRACT

Development of fully featured Automatic Speech Recognition (ASR) systems for a complete language vocabulary generally requires large data repositories, massive computing power, and a stable digital network infrastructure. These conditions are not met in the case of many indigenous languages. Based on our research for over a decade in West Africa, we present a lightweight and downscaled approach to AI-based ASR and describe a set of associated experiments. The aim is to produce a variety of limited-vocabulary ASRs as a basis for the development of practically useful (mobile and radio) voice-based information services that fit needs, preferences and knowledge of local rural communities.

CCS CONCEPTS

• **Social and professional topics** → **Cultural characteristics**; • **Computing methodologies** → **Machine learning algorithms**.

KEYWORDS

under-resourced/indigenous languages, low resource environments, machine learning, voice-based technologies, neural networks, automatic speech recognition

ACM Reference Format:
George Vlad Stan, André Baart, Francis Dittoh, Hans Akkermans, and Anna Bon. 2022. A Lightweight Downscaled Approach to Automatic Speech Recognition for Small Indigenous Languages. In *14th ACM Web Science Conference 2022 (WebSci '22), June 26–29, 2022, Barcelona, Spain.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3501247.3539017

1 DESIGNING FOR LOW-RESOURCE ENVIRONMENTS

Development of fully featured ASR systems for a complete language vocabulary generally requires large data repositories, massive computing power, and a stable digital infrastructure for the collection and computational processing of speech data. For "big" world languages such data are commonly available on the Web, for example in the form of audio books or spoken Wikipedia articles. Large ASR providers such as Google and Apple additionally generate proprietary data sets, for example by hiring speech actors.

However, such a high-investment, high-resource approach is not feasible for many "small" indigenous languages, as our research for over a decade in West Africa, in particular in collaboration with rural organizations and communities in the Sahelian drylands, has shown. This is not just due to the fact that technology requirements as mentioned above are not met, and will not be for a long time to come.

Additional issues to deal with stem from the fact that literacy rates in rural West Africa are low. In Mali and Burkina Faso less than 30% of the rural population has reading and writing skills. Therefore, oral communication is preferred instead of written text, especially for native speakers of under-resourced languages, such as for example Bambara and Bomu in Mali, Mooré in Burkina Faso, and Dagbani and Frafra in Northern Ghana.

These above-mentioned factors, in combination with the rapid uptake of smartphones and the internet, reinforce the importance of ASR systems for the inclusion of low-literate speakers of low resource languages in the digital world.

West Africa's drylands, where many indigenous languages are spoken, can be characterized as low-resource environments. The benefits of voice-based information services in this context have been demonstrated in various studies [3, 8, 9]: voice-based services can facilitate local economic activities, related to agriculture, weather and market information, livestock rearing or trade. Generally, our experience is that there is a wide useful range of community-oriented voice services conceivable, but adaptation to the specific local requirements and contextual conditions is a key success factor for any digital voice service that aims to serve users in these low-resource environments.

A concrete example of an adaptive voice-based service that was designed and deployed specifically for users in rural Mali, in order to support and enhance local citizen journalism, is Foroba Blon. This mobile voice-based service allows village reporters to leave a spoken message or announcement that they want to have broadcast on the local community radio. When they dial a specific phone number, they are presented with a spoken welcome phrase followed by a voice menu asking them to select between two options: (i) leave a

WebSci '22, June 26–29, 2022, Barcelona, Spain

ACM ISBN 978-1-4503-9191-7/22/06…$15.00
https://doi.org/10.1145/3501247.3539017

voice message or (ii) listen to the pre-recorded message again. If they select and leave a message they are prompted again with four options: (i) to listen to their recorded message, (ii) to update the message, (iii) to listen to the menu again or (iv) to hangup. The call flow in the indigenous language Bambara is presented in Figure 1.

In the current system, user input was limited to the pressing of digits on the phone keypad (DTMF). This is a natural consequence of the unavailability of ASR for the language (in this case Bambara). However, a spoken dialogue would be a more natural way of communication for the users of this application. We note that for many voice-based mobile information systems, a limited vocabulary of possible spoken user input can be sufficient. An ASR system focused on a small vocabulary is also much less complex to develop and consequently more affordable and practical. Although clearly short of a full-fledged ASR, many practically useful voice services can still be developed in line with the interests of local communities and populations.

Other examples of opportunities for ASR include the processing of WhatsApp voice messages. This enables for example the development of voice-based chatbots in local languages, leveraging WhatsApp voice-messages for interaction with the user. [12]

This paper presents such a lightweight and downscaled approach to voice services in indigenous languages, and describes a set of successful experiments with it. To generate a small-corpus ASR to enhance voice information systems in indigenous languages, speech data are collected by crowd-sourcing through a simple application. The computational process is optimized for relatively small amounts of speech data and is based on well-established methods. The resulting ASR works for limited topic-focused vocabularies, but provides sufficient functionality as required by locally relevant voice-based information systems.

2 RESEARCH APPROACH

To design a light-weight method to developing and deploying small vocabulary ASR systems, our workflow must meet several requirements: (i) it must provide an easy means of data-collection that meets the requirements of low resource environments (ii) it must be capable of processing and analyzing speech data of low quality (iii) it must be easily integrated into existing voice-applications that have been designed by local users in low resource environments (iv) the ASR development tool must be made available as Open Source to encourage local developers to build local voice-applications.

Since many voice applications require only simple human-computer interaction, in which questions can be answered by with a few single words, and speaking is more natural for users than DTMF (e.g. "pressing 1 for yes" and "2 for no"), we started creating a system that can recognise the words "yes" and "no".

Utterances for "yes" and "no" are collected using a crowd-sourcing application. The envisaged system should be able to recognise a variety of different voice types, irrespective of the person's gender, age, or dialect. In order to achieve this, it is important to ensure a diversity of the population that contributes the utterances.

The workflow consists of (i) data collection of speech utterances with native speakers of the indigenous language (ii) processing of the data to generate the ASR model (iii) implementing the ASR model in a voice service development system (iv) test and deploy the ASR to build voice applications for local communities. This study has only been completed for steps i and ii. Steps iii and iv are still research in progress.

Firstly, a (i) crowd-sourcing application has to be designed to collect utterances from low-resource languages and tested, and secondly (ii) a model to easily generate the ASR, in the availability of a small vocabulary and a relatively small corpus. For (i) a mobile and desktop web application is built. For (ii) a Machine Learning model is used, adapted and tested. The results have been tested with local users.

The next step will be to integrate the resulting ASR system into a voice software development system. This is a platform that supports easy creation of voice-based information services, which can be used for example, to provide information to farmers in a community. This is not yet realised, but we plan to integrate the system with the KasaDaka platform, which is targeted at voice services in low-resource contexts.[1]

3 RELATED WORK

The number of studies on computational processing of speech data for small, indigenous languages is rapidly gaining momentum. From various studies we learn different aspects that can be useful for our present study.

In 2019 and 2020 Fantaye et al. [6, 7] investigated which Deep Neural Network acoustic modeling units worked best for developing an ASR system for the Ethiopian languages Chana and Amharic. Similarly, for India, Toshniwal et al. trained a single ASR model on nine different Indian languages [14]. Remarkably, these three studies disposed over much larger amounts of data available than we our current project will have.

Already in 2010, Ajisafe et al. did a collaborative work on the development of ASR for Pidgin English in Nigeria and Rwanda. The authors converted audio-data to Mel Spectrograms. They had high quantities of speech data available for training the model, and their model attempted to recognize entire sentences.

In 2011, for collection of speech data in the African context of small languages, an open-source tool called Woefzela was developed and tested with South African languages by de Vries et al. [4]. This application has the limitation of only functioning on the Android operating system and requiring hardware with high performance, due to the real-time quality control. Especially its offline functionality can be considered an important feature for a data collection application, given the lack of internet in the low-resource areas where the target populations of indigenous languages often live.

To cater for the lack of large amounts of speech data to train machine learning models on, Van Heerden et al. [16] performed several experiments in 2010 with data pooling (similar to transfer learning) from related languages in order to improve speech recognition performance. They concluded that as long as the languages are closely related, two hours of speech in a related language is equivalent to one hour of data from the target language, yet this benefit decreases rapidly if the languages become more distant/different. In our project, we will also explore the role that this type of so-called "transfer (machine) learning" can play on speech recognition accuracy improvement.

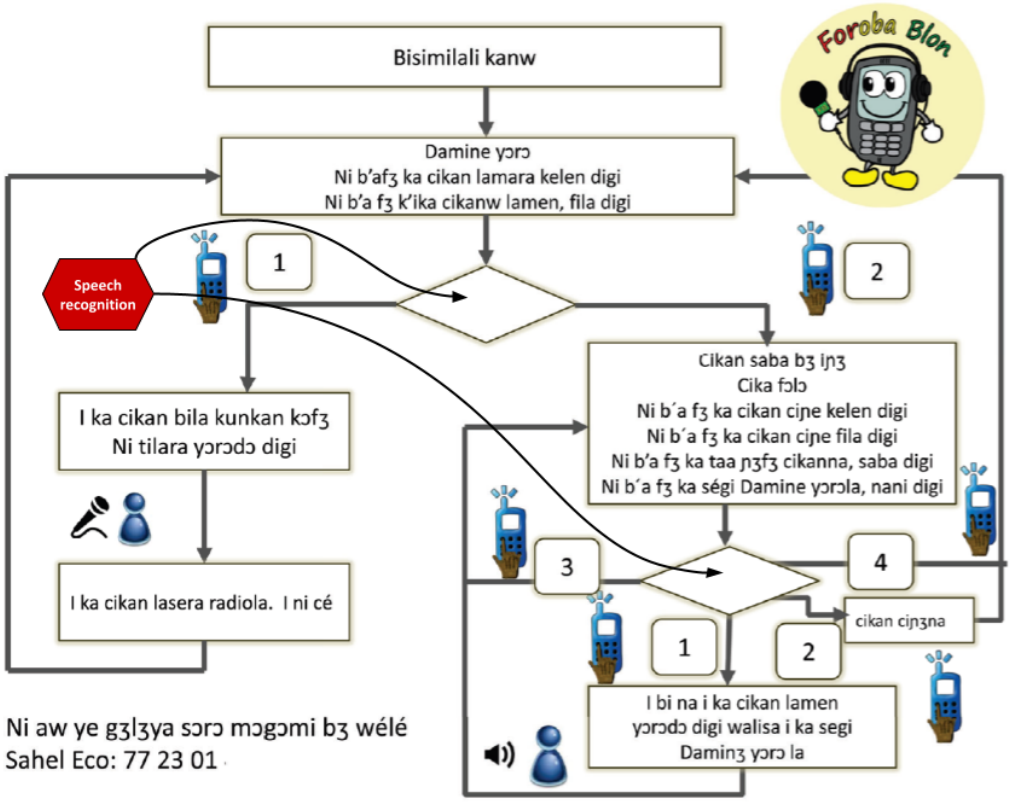

Figure 1: An example of a call flow for an existing voice application in the Bambara language (Mali). The red oval shows the need to implement the ASR for "yes" or "no" in the local language. Diagram adapted after [2], p 94.

Another related project is Common Voice[1], an initiative created by the Mozilla Foundation, which aims to crowd-source speech data to build both a Text-to-Speech (TTS) system as well as an Automatic Speech Recognition (ASR) system for different languages, avoiding the reliance on big data companies. A number of languages we are targeting in our study, including Twi (for Ghana) and Bambara (for Mali) are not yet supported by this Common Voice project.

Regarding methods for the design of Text-to-Speech systems for small, under-resourced languages, Justyna Kleczar [11] presented a general purpose Text-to-Speech TTS system, for which she used the language Twi from Ghana as the exemplar, which is also one of the target languages for our project. Various limitations and observations mentioned in her paper, with respect to difficulties with speech data collection, local availability of a digital infrastructure in low resource environments, computational processing of small corpus of speech, testing and deploying in the local context, also apply to the project presented in this paper.

A recent publication by van der Westhuizen et al. [15] describes a system that recognizes 18 keywords in the Luganda language. The system is able to recognize these keywords during radio programs. While intended for a different purpose, the principle of a limited domain ASR is similar to this study.

Based on this literature and resources on machine learning, we selected a state-of-the-art method for classifying audio data that involves converting the speech sound files to Mel Spectrograms and using them to train a Convolutional Neural Network adapted to the specific problem, a process which could also function with limited amounts of data.

4 COLLECTING UTTERANCES WITH A CROWD-SOURCING APP

Taking into account the hardware limitations we learned about in the literature review, we decided to start developing an application for collecting audio recordings of the words "yes" and "no" in a number of languages which can be easily expanded and adapted to future needs. The resulting recordings will later be used as input for the training of the ASR models.

4.1 Requirements

In order to ensure the application is fit for the limited resource environment of the devices it will likely be used on, we decided to develop our data collection application as a web application,

[1]See: https://commonvoice.mozilla.org/nl

using standard JavaScript, HTML and CSS and avoiding the use of software frameworks which would add significant overhead to the application. The result is a small web application which requires only 490 kB of space on the devices it's running on. In order to record and play back the audio clips, the application makes use of the MediaRecorder interface of the MediaStream Recording HTML5 API, which is supported by all major desktop and mobile browsers, including Google Chrome, Firefox and Safari. The audio clips collected are compressed using the Opus codec and stored in an Ogg container. In our tryouts, the average size of such a file containing 5 seconds of audio data was just 15 kB, which means it can easily be uploaded even on a very limited bandwidth internet connection (on a 0.1 mbps 2G connection it can be uploaded in less than 2 seconds). The resulting ogg files are then sent to an Amazon S3 bucket. This service can be replaced at any time with a private server or a different cloud storage service. The source code for the application is available in a repository on GitHub[2]. The web application was then published[3] using GitHub Pages.

4.2 Description of the utterance contribution process

The data collection web application currently supports 18 different languages including Twi, Frafra, Bambara, Mooré, and Bomu. Languages can be added and removed as needed using a text editor. After selecting the language, the user needs to first give permission for the application to use the microphone, after which they are able to see a visualization of the sound wave generated by their voice. They first record the word for "yes", and afterwards the word for "no". The application automatically stops recording 5 seconds after tapping the record button, in order to prevent unnecessarily large files from being uploaded to the server. Before they submit the recording files to us, users are able to play back their recording to make sure they are satisfied with the quality of the recording. If they are not, they can start over with recording both words, until the quality is satisfactory. The final step is to tap the Send button, which uploads both audio files. The web application has a responsive user interface, which means it can be displayed correctly on any type or size of screen. In Figure 2 can see a screen shot of the home page of the web application with the Twi language selected from the drop down menu.

4.3 Collection of speech data

The application can be used to collect speech data at any time by simply sharing the URL with persons who speak the desired language. During the Open International Webinar Artificial Intelligence in & for the Global South[4], which took place between the 2nd and 4th June 2021 as well as the follow-up course at Vrije Universiteit Amsterdam, a large amount of speech data was collected from the participants, for different languages. English had the most respondents and was chosen as the language that will be used as the proof-of-concept for our automated speech recognition system. A total of 104 speech utterances were recorded for "yes" and "no".

5 PREPARATION AND ANALYSIS OF COLLECTED DATA

The next step, after collecting sufficient data for a specific language, in our case English, was to analyse the speech recordings collected. In figure 4.1 the wave forms of nine data points along with their respective labels are shown; despite containing the pronunciation of the same word, the wave forms vary a great deal from one another.

5.1 Data quality and cleaning

Each file in our data set was individually assessed to determine whether the quality of the recording was sufficient for use in the machine learning stage of our project. The results of our analysis showed that 20 files out of 208 were not readily usable for training a machine learning model or were completely unusable. 8 of the unusable files contained only noise or no sound data at all. The remaining 12 files contained the right words pronounced by different individuals, but repeated multiple times within the same file. Our decision was to split these multiple utterances into separate files, increasing the amount of data we had available. Our final data set contained 248 audio files, 124 with the word "yes" and 124 with the word "no".

5.2 Data Preparation

Our next step involved preparing the data for use in a machine learning model.

5.2.1 Standardization. The data set of utterances we had collected included audio files of different lengths and of different encodings[5]. The machine learning framework expects input data of the same length (in seconds) and of the same format. Because of this we decided to standardise all our data to be 1 second in length and to use a 16 kHz 16-bit single channel PCM wave file format. We also normalized all our audio data to a standard amplitude.

5.2.2 Data augmentation. Data augmentation is a process in which certain techniques are used to increase the amount of data by adding slightly modified copies of already existing data. In our case this process involved taking the existing sound files and modifying their pitch or adding artificial noise, or both. For each of our audio files we created two new versions of the file, one with lower pitch and one with higher pitch. We then took the three resulting files and added artificial noise to them, resulting in a total of six files. This meant that using data augmentation, we now had six times more data available, which amounted to 1488 audio files. For the two new versions of data with changed pitch, we also added an additional type of background noise, similar to the noise in a poor GSM connection, so our final total number of samples was 1984.

[2]https://github.com/vladpke/rare-language-recorder
[3]https://vocesrares.nl/
[4]See: https://perspectives-on-ict4d.org/

[5]Not all browsers produced files of the same audio encoding.

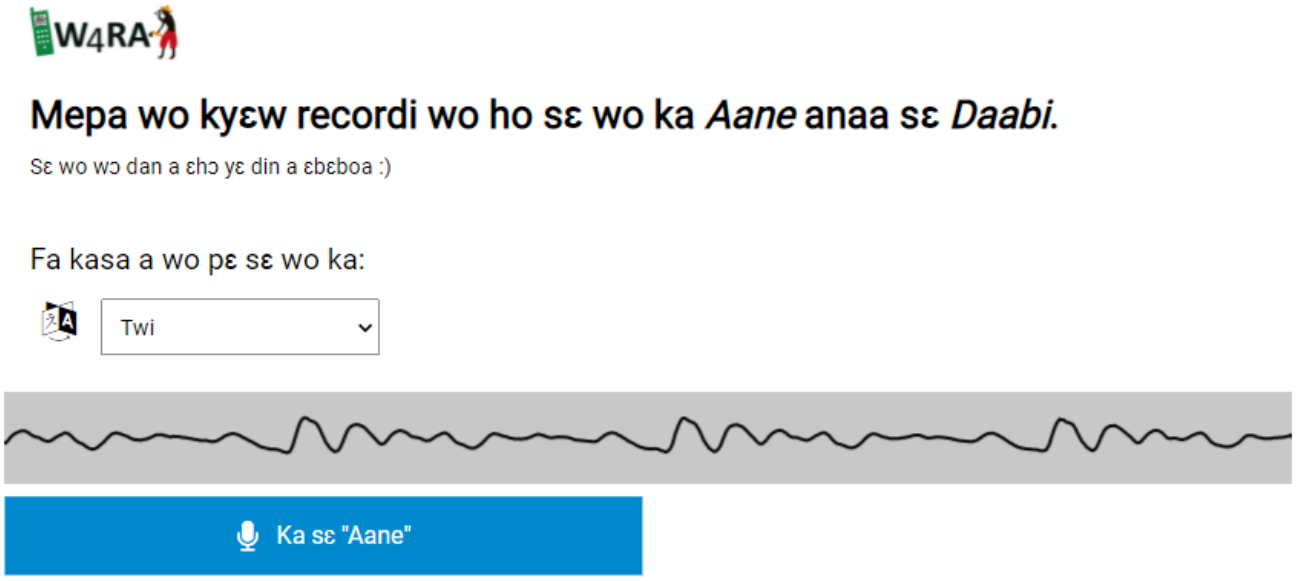

Figure 2: The interface of the *vocesrares.nl* web application; it supports data collection in various low-resource languages; the Ghanaian language Twi is displayed here.

The data augmentation process not only increases the amount of data, but also helps increase the accuracy of our speech recognition by preventing the machine learning model from over-fitting during training and by making the model's predictions compatible with different voices and noisy environments.

5.2.3 Conversion to Mel Spectrograms. In order to prepare our audio dataset for the training stage of our convolutional neural network, we must first transform the sound signal into a Mel spectrogram, a two-dimensional visual representation of the spectrum of frequencies of a sound signal as it varies with time. Mel here refers to the use of the Mel scale, which is a non-linear frequency scale in which sounds of equal distance from each other on the graph also sound as being equal in distance from one another to humans. For example, in the hertz (Hz) scale, the difference between 500 and 1000 Hz is obvious to a person, whereas the difference between 8000 and 8500 Hz is not noticeable. In order to transform each audio signal into a spectrogram, we had to first compute the short-time Fourier transform for each file, which is a mathematical function that gets a signal in the time domain as input, and outputs its decomposition into frequencies. Since our files are all 1 second in length, we used a short-time Fourier transform with window-size of 255 and a hop-size of 128. These parameters can be adjusted and determine the resolution of the resulting spectrogram. Next we computed the sound magnitudes at each frequency window, in decibels (dB) and we converted the linear hertz scale to the Mel scale mentioned above. We now had our spectrograms ready to be used in the training of the machine learning model. In figure 3 a number of spectrograms are shown for random files in our dataset, along with their corresponding labels, "yes" or "no".

6 MACHINE LEARNING MODEL

6.1 Training the model

In this section we will talk about the construction of the machine learning model using the data we prepared in the previous chapter for training, validation and testing. We first randomized the data after which split it into three datasets using the percentage ratio 70:20:10. That means 70% of the data was reserved for the training dataset, 20% of the data for the validation dataset, and 10% of the data for the testing dataset. There are a number of Python libraries that can be used to create a convolutional neural network machine learning model. The two libraries that we tried working with were Keras, an open-source library which is now part of the TensorFlow library [5], and fast.ai [10], which is also an open-source deep learning library built by a non-profit research group.

Our final yes/no model is created using the Keras library, which is well-known and has extensive documentation and support. For our project, the library can be swapped for another, if deemed necessary. Using the Sequential model from the Keras library, we were able to build our convolutional neural network. This type of neural network contains convolutional layers, based on the mathematical operation of convolution; they are sets of filters, in the shape of 2D matrices, convolved with the input image during learning, enhancing distinguishing features in it and aiding greatly in computer image classification.

- a Resizing layer to downsample the input, enabling the model to train faster
- a Normalization layer to normalize each pixel in the image based on its mean and standard deviation

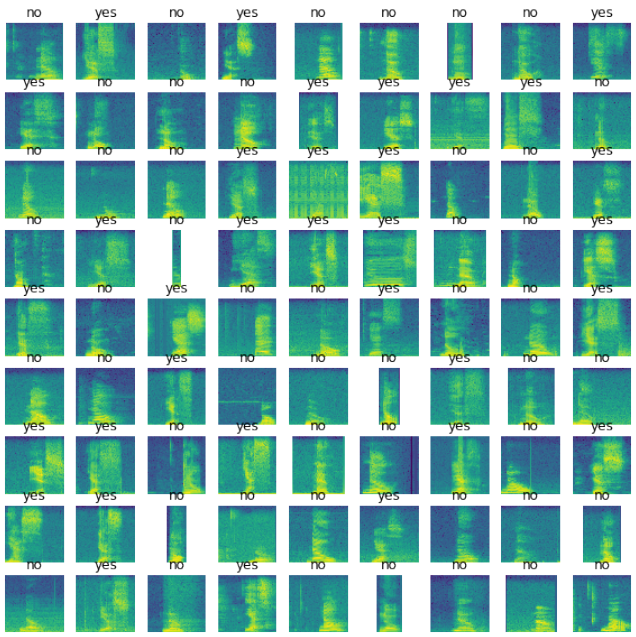

Figure 3: Visualisation of Mel Spectrograms of files, randomly selected from our dataset.

- two Convolutional layers with 32 and 64 output filters respectively, both using the Rectified Linear Unit (ReLU) activation function and a 3x3 kernel
- a two dimensional Max-pooling layer, which down samples the input in order to highlight the most present feature in an image or output matrix
- a Dropout layer which randomly sets input units to 0 with a desired frequency, which in our case is 0.25; this layer reduces over-fitting
- a Flatten layer, which converts the data into a 1-dimensional array for inputting it to the next layer
- a Dense layer, which is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer; the Dense layer we used has an output dimensionality of 128 and uses the ReLU activation function
- an additional Dropout layer with a 0.5 probability setting
- finally, a second Dense neural network layer with an output dimensionality of 2, for the number of labels we have in our data, "yes" and "no".

The model was then compiled using the Adam optimization algorithm.

6.2 Testing the model

Our next step is to test the resulting machine learning model with the testing dataset as well as with real life speech inputs. For each combination of parameters and quantity of data attempted we followed these steps 10 times and computed the average accuracy: re-randomised the data, trained a new model and collected the resulting accuracy. In table 1, the resulting accuracy is shown depending on the total quantity of data used.

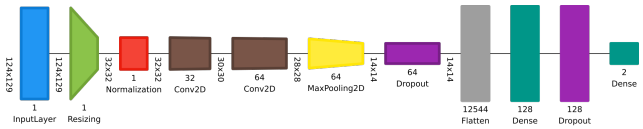

Figure 4: Diagram visualizing the entire configuration of our CNN model.

The confusion matrix in 5 shows a comparison between the word predicted by the model and the actual label the sound file from the testing dataset. When the model predicted the word "no", it was correct 100 times and wrong 6 times, while when the model predicted "yes" it was correct 95 times and wrong 7 times. The accuracy of the model needs to be extensively tested further in real life. In our (limited) real-life tests, the accuracy was very high.

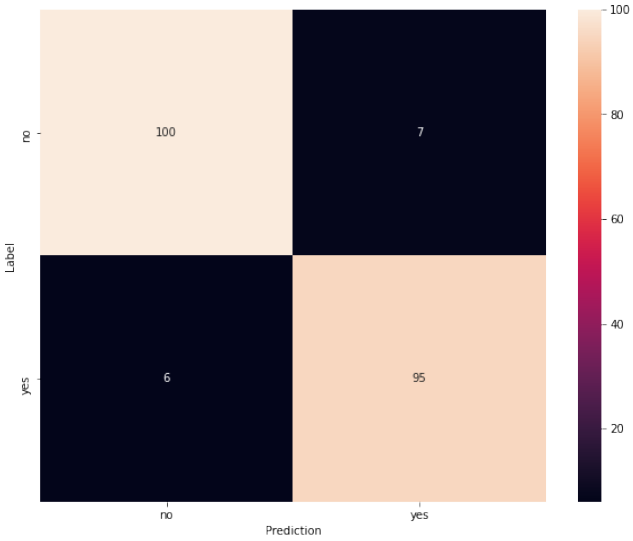

Figure 5: Confusion matrix.

6.3 Model deployment in real world cases

In order to showcase the functionality of our "yes and no" speech recognition model, we are planning to implement the model in a voice service which uses voice control for interaction. The integration will work by recording the audio of the caller and rapidly sending it to the model for assessment, after which it returns its prediction. The technical implementation of this process can also be used in other contexts, such as smartphone applications or websites.

An example of a more complex voice service (in comparison to the above-mentioned Foroba Blon citizen journalism system) is the "Mali Milk" application. The goal of this application is to connect milk-producing farmers with milk cooperatives, enabling them to quickly get their (rapidly-decaying) milk delivered to the processing factory. Figure 6 displays the call flow for this application, the red element points to the instances where the ASR can be implemented.

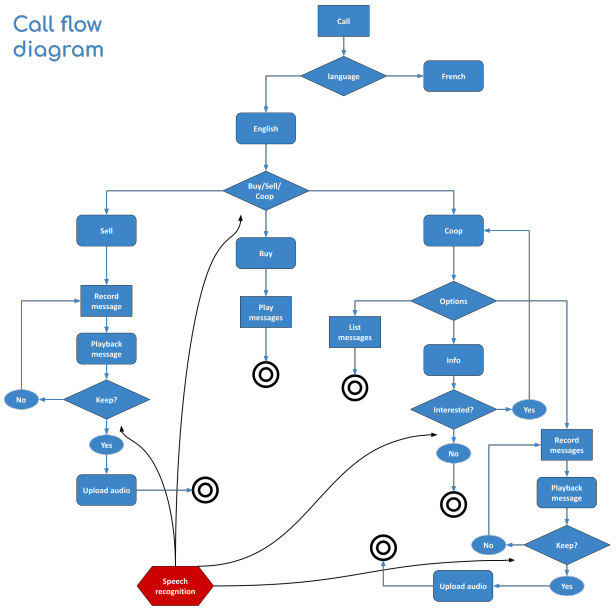

Figure 6: An example of a call flow for a voice application for livestock rearers in Mali. The red oval shows the need to implement the ASR for e.g. "yes" or "no" . Diagram is adapted after [13], p 16.

Table 1: Accuracy increases with data size.

1984 samples	97 - 98 % accuracy
496 samples	90% accuracy
248 samples	80% accuracy

7 DISCUSSION AND FUTURE WORK

We have identified many areas of improvements and expansion of the proposed system, which contribute to increasing the practical usefulness of the system. We will introduce several of these below.

7.0.1 Expanding the supported corpus and improving diversity. The presented ASR system will be expanded to support a larger corpus, including the numbers from 0 to 9, as well as to resource other indigenous languages which have no ASR support. We performed experiments with number recognition in addition to the words "yes" and "no" and the results were promising. The multiple biases introduced by our data can be mitigated. A good range of voices needs to be supported which does not favor a specific gender, age or dialect. For example, if the data collected comes predominantly from males between the ages of 30 and 50, the model will have a disproportionately better accuracy for individuals in that group. Similarly, if the data contains only speakers of a certain dialect of the target language, the model will perform better with speakers of that dialect.

7.0.2 Improve quality of user-contributed utterances. During our data collection phase, we noticed that some of the recordings were done either too far or too close to the microphone, which resulted in audio data that could not be recognised even by human listeners. It is therefore important that in the future, during data collection, we encourage users to listen back to their recordings before they submit them, and re-record themselves if necessary. If they are unable to record a clear enough sample, likely due to a faulty microphone, we should encourage them to not submit their recordings and instead try a different device. As the amount of data for our model training increases, so does the difficulty in detecting flaws in data samples. An automated system should be developed which can flag audio files which might not fulfil quality standards.

7.0.3 Oral user interface for contributing utterances. Our data collection application currently only interacts with users via written text. Considering the predominantly oral communication present in many of the targeted communities, a version of our application should be created that works with vocal instructions.

7.0.4 Improving model accuracy and inclusion of more languages. More work can be done in finding better parameters for the Convolutional Neural Network model used, as well as in experimenting with other machine learning models which could yield better performance. Since the currently built model (for English) managed to achieve a high accuracy, it should now be trained with low-resource languages, Twi for Ghana and Bambara for Mali. At the time of writing, we did not yet have access to sufficient data to train models for these languages. Data for these languages will be collected in the field or through the internet in the near future. The proposed methodology is likely to work for any language. A limitation, however, is that the words that are to be recognized should not be similar sounding[6]. Finally, the models for each of the languages we support should be able to improve over time, using transfer learning with new speech data, as well as data from very similar languages.

7.0.5 Additional real-world testing. After this process, the ASR should be extensively tested in real-world conditions, such as over phone connections which often suffer from low audio quality. Attention should be given to the integration of the ASR in the system, in particular there should be fallbacks for when the ASR does not function in a satisfactory manner for the user. The effectiveness of the ASR in improving usability should be assessed in more detail.

8 CONCLUSION

This research has presented a lightweight, downscaled approach and model to resource indigenous languages for people in less privileged communities. The workflow consists of three modules: one for data collection, one for machine learning model training and one for showcasing the accuracy of the resulting model.

With this approach we have demonstrated that it is feasible to build a lightweight, Open Source Automatic Speech Recognition system, with a limited vocabulary focused on specific service areas of local interest, using a very low amount of data.

With just 248 data samples per word, we have been able to achieve an accuracy above 90% for recognising the words "yes" and "no". Such a limited-vocabulary ASR can be used to enable the hundreds of millions of low-literate, low resource language

[6]Similar sounding words could probably still be recognizable, but would likely require a higher amount of training data.

speaking populations, to interact with technology in an appropriate manner.

Although arguably the limited vocabulary limits the possibilities of this specific ASR, the vocabulary can be adapted to a specific domain or use-case, using the same methodology. For speakers of the thousands of low-resource languages for which previously no ASR existed, the step from zero to a limited domain ASR constitutes a significant improvement.

In this ongoing research project we have also demonstrated that Artificial Intelligence technologies do not need to exclusively serve people in industrialized countries and big companies. There is a high potential for democratizing AI, crowd-sourcing its training data and leveraging its power for all communities around the world, no matter their socioeconomic status.

ACKNOWLEDGMENTS

The authors thank Adama Tessougué, Amadou Tangara for the help in translating the application in various languages and provisioning of utterances and testing. We thank Enrico Rotundo and Alberto Caroli, data scientists, for their advise on machine learning models and Mark Hoogendoorn, Professor of Artificial Intelligence at Vrije Universiteit Amsterdam and chair of the Quantitative Data Analytics group for great support and good advise. We extend our gratitude to everyone that contributed voice samples for this research.

REFERENCES

[1] André Baart, Anna Bon, Victor de Boer, Francis Dittoh, Wendelien Tuijp, and Hans Akkermans. 2018. Affordable Voice Services to Bridge the Digital Divide: Presenting the Kasadaka Platform. In *International Conference on Web Information Systems and Technologies*. Springer, 195–220.

[2] Anna Bon. 2020. *Intervention Or Collaboration?: Redesigning Information and Communication Technologies for Development*. Pangea, Amsterdam. 1–362 pages. https://doi.org/10.26481/dis.20201215ab

[3] Victor de Boer, Pieter De Leenheer, Anna Bon, et al. 2012. Radiomarché: Distributed voice-and web-interfaced market information systems under rural conditions. In *Advanced Information Systems Engineering*. Springer, 518–532.

[4] Nic J De Vries, Jaco Badenhorst, Marelie H Davel, Etienne Barnard, and Alta De Waal. 2011. Woefzela-an open-source platform for ASR data collection in the developing world. Conference paper.

[5] Francois Chollet et al. 2021. Keras: the Python deep learning library. Retrieved December 31, 2021 from http://keras.io

[6] Tessfu Geteye Fantaye, Junqing Yu, and Tulu Tilahun Hailu. 2019. Investigation of Various Hybrid Acoustic Modeling Units via a Multitask Learning and Deep Neural Network Technique for LVCSR of the Low-Resource Language, Amharic. *IEEE Access* 7 (2019), 105593–105608.

[7] Tessfu Geteye Fantaye, Junqing Yu, and Tulu Tilahun Hailu. 2020. Investigation of automatic speech recognition systems via the multilingual deep neural network modeling methods for a very low-resource language, Chaha. *Journal of Signal and Information Processing* 11, 1 (2020), 1–21.

[8] Nana Baah Gyan. 2016. *The Web, Speech Technologies and Rural Development in West Africa, An ICT4D Approach*. Vrije Universiteit Amsterdam. 1–136 pages.

[9] Nana Baah Gyan, Victor de Boer, Anna Bon, Chris van Aart, Hans Akkermans, Stephane Boyera, Max Froumentin, Aman Grewal, and Mary Allen. 2013. Voice-based Web access in rural Africa. In *Proceedings of the 5th Annual ACM Web Science Conference*. ACM, 122–131.

[10] Jeremy Howard and Rachel Thomas. 2021. fast.ai: Making Neural Networks Uncool Again. Retrieved December 31, 2021 from http://fast.ai

[11] Justyna Kleczar. 2017. *General purpose methodology and tooling for Text-to-Speech support in voice services for under-resourced languages*. Ph.D. Dissertation. Master thesis. Vrije Universiteit Amsterdam.

[12] Thomas Reitmaier, Electra Wallington, Dani Kalarikalayil Raju, Ondrej Klejch, Jennifer Pearson, Matt Jones, Peter Bell, and Simon Robinson. 2022. Opportunities and Challenges of Automatic Speech Recognition Systems for Low-Resource Language Speakers. In *CHI Conference on Human Factors in Computing Systems*. 1–17.

[13] Aske Robenhagen and Bart Aubers. 2016. The Mali Milk Service 3.0 – Voice-based platform for enabling farmer networking and connections with buyers. In *Proceedings of the 4th Workshop on Downscaling the Semantic Web (Downscale2016)*. https://w4ra.org/wp-content/uploads/2016/07/Downscale2016_paper_2.pdf

[14] Shubham Toshniwal, Tara N Sainath, Ron J Weiss, Bo Li, Pedro Moreno, Eugene Weinstein, and Kanishka Rao. 2018. Multilingual speech recognition with a single end-to-end model. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 4904–4908.

[15] Ewald van der Westhuizen, Herman Kamper, Raghav Menon, John Quinn, and Thomas Niesler. 2022. Feature learning for efficient ASR-free keyword spotting in low-resource languages. *Computer Speech & Language* 71 (2022), 101275. https://doi.org/10.1016/j.csl.2021.101275

[16] Charl Van Heerden, Neil Kleynhans, Etienne Barnard, and Marelie Davel. 2010. Pooling ASR data for closely related languages. (2010).