KALDI GOES ANDROID

C. Gaida[1,2]*, R. Petrick*[2]*, D. Suendermann-Oeft*[1,3]

[1] DHBW, Stuttgart, Germany [2] Linguwerk, Dresden, Germany
[3] EMR.AI, San Francisco, USA

{christian.gaida,rico.petrick}@linguwerk.de david@emr.ai

ABSTRACT

We describe the development of an application running a derivative of the Kaldi Gaussian Mixture Model (GMM) decoder physically on a mobile Android device. The application is written in C/C++ using the Qt development framework providing abstraction of the audio interface, the creation of a frontend and a multi-threaded architecture. It is shown that this open-source large-vocabulary speech recognition system successfully runs on Android as real-time decoder of live streamed audio on ordinary smart devices.

***Index Terms*—** Open-Source, Large Vocabulary, Continuous Speech Recognition, Kaldi, Android

1. INTRODUCTION

Large Vocabulary Continuous Speech Recognition (LVCSR) on mobile devices is almost exceptionless accomplished by client-server network solutions, e. g. Google Speech [1], Apple Siri [2] or Nuance Dragon Dictate [3]. Audio capture, at times feature extraction to compress data on the client side, and visualization of the recognition results are performed on the mobile device, but the decoding is processed on an external server. A reason for such a solution is the limitation of resources of mobile devices with respect to memory and computational power.

However, decoding on an external server has certain drawbacks:

- It requires internet connectivity.
- Applications use servers provided by foreign companies. This affects data privacy, e. g. personal or confidential data in the medical domain or in the field of government or law.
- The modification or customization of the decoder by the user is difficult or impossible.
- The response time of the system depends on the quality of the internet connection and work load of the server. Applications like control or dictation require feedback in real-time, otherwise the usability is compromised.

The increasing computational power of tablets and smart phones makes it possible to accomplish the speech recognition task locally on the device. The presented work describes the port of a Kaldi-based, open-source large-vocabulary decoder on a mobile device running Android. The only existing related work is, to the best of our knowledge, the development of PocketSphinx [4], a decoder initially designated for embedded platforms and subsequently ported to Android [5].

The choice of the Kaldi toolkit [6] was motivated by the results of comparisons [7, 8, 9, 10, 11] of open-source speech recognition toolkits. It was consistently shown that Kaldi offers the best performance of the compared toolkits, even in the case of using Gaussian Mixture Models (GMM), instead of the even better performing Deep Neural Networks (DNN) for acoustic modelling. This work is based on GMM related examples of the Kaldi toolkit.

The paper is organized as follows: in Section 2 the development of the Android application is described. Section 3 gives an overview of the features of the final application. In Section 4 we discuss observations, conclusions and suggest future work.

The present work was carried out in the scope of the project OASIS (*O*pen-source *A*utomatic *S*peech recognition *I*n *S*mart devices) [12], sponsored by the Baden-Wuerttemberg Ministry of Science and Art.

2. DEVELOPMENT

The development was done on a Linux Ubuntu 14.04.1 LTS 64-bit system. The software development environment includes

- Qt[1], Version 5.3.1, Community Edition,
- Android Native Development Kit[2] (NDK), Version 10, 64-bit systems, 32-bit target,
- Android Software Development Kit (SDK) as part of the Android Studio Bundle[3], Version 135.

[1] Available from http://qt-project.org/.

[2] Available from https://developer.android.com/tools/sdk/ndk/index.html.

[3] Available from http://developer.android.com/sdk/

The goal was the creation of a Kaldi-based solution completely from source. Qt is a software development kit for applications written in C/C++, which aimes at platform independent development. Associated with the Android NDK and SDK it provides tool chains to cross-compile to ARM[4] architecture and tools to deploy applications on Android. Qt was also chosen for the development of an interface for live audio recording and creation of a modular, multithreaded application. ARM architecture has special features e. g. low power consumption and is therefore primarily used in the domain of embedded devices, smart phones and tablets.

The entry point of the project was the source code of the implementation of the *online-gmm-decode-faster* example. Dependencies with respect to *portaudio* were removed, because this low-level audio I/O application programming interface (API) accesses the audio interface of the installed operating system directly. Therefore it supports desktop systems but contradicts the platform abstraction of Android.

The language model implementation of Kaldi depends on the OpenFst Library[5]. This library is written in C, has no obstructive dependencies, and could therefore be simply included and cross-compiled.

The *matrix* calculus of Kaldi is based on the Basic Linear Algebra Subprograms[6] (BLAS) and the Linear Algebra Package[7] (LAPACK). The structures and routines of these libraries are optimized for efficient computation of vector and matrix operations. Higher-level libraries like Automatically Tuned Linear Algebra Software[8] (ATLAS) or OpenBLAS[9] are alternatives, but also based on BLAS and LAPACK.

The port of the math dependencies was the most challenging part of the project. The BLAS and LAPACK sources are written in Fortran, directly cross-compiling would have required the build of a Fortran compiler for Android. The additional hardware dependent optimization of the higher-level libraries with respect to the system, on which the library is build, made a port very complicated or impossible. Finally, the low-level BLAS and LAPACK sources were parsed to C with *f2c*, mapping dependencies were identified and the source included. To prevent problems caused by unused code, the Kaldi source and included LAPACK source were partially reduced.

A live audio interface was created using the Qt framework, recording raw audio from microphone and writing samples to a separate buffer. The decoder example implementation was running and polling audio data in a loop, which caused blocking behaviour and suppressed audio recording in the decoder's thread. This was resolved by development of a decoder module, which is completely seperated from the audio source. It encapsulates only the decoding and reads data from the separate buffer. Timer triggered processing was used to reduce CPU time usage. Development was finished by creating a multi-threaded application running audio input and decoder in seperate threads to make use of multi-core processor architecture. A graphical user interface (Figure 1) was created for testing and demonstrating purposes.

3. APPLICATION FEATURES

The application was successfully run on a two year old Samsung Galaxy Tab2 10.1 and a more powerful Samsung Galaxy S5 Smartphone. A language model (an ARPA 3-gram model) with a vocabulary of 6,452 words and an acoustic model were trained on the Verbmobil 1[10] [13] (VM1) corpus with 285k training tokens and approximately 30 hours of speech.

The runtime features of the Samsung Galaxy Tab2 10.1 are follows:

- Acoustic model size: 3.2 MB,
- Language model size: 95.0 MB,
- Runtime binary size: 32.2 MB,
- RAM usage: up to 200 MB (installed 1 GB),
- CPU usage: up to 78.2 % of the total power of two 1.0 GHz cores.

The decoder was executed with standard parameters. Internally the decoder module is not multithreaded, so it uses a single core. The response time of the system was 0.4-0.8 s in the case of completely or partially known phrases. The processing time increases in the case of word sequences not occuring in the language model training, or low quality of the recorded audio from microphone, e. g. noise or background voices. E. g. the delay between the ending of the utterance of the unknown phrase *"Hallo, mein Name ist Herr Mueller. Ich fliege mit dem Flugzeug zu einer Tagung nach Berlin."* and displaying the complete result in the text box was 1.1 s in a slightly noisy environment. Speed up of processing without changing the Kaldi implementation could be achived by reduction of the search space by decreasing the search beam width.

4. CONCLUSIONS

We described the development of an Android application used for continuous speech recognition with real-time response based on the Kaldi toolkit. This work includes two major achievements:

installing/studio.html.

[4]Advanced Reduced Instruction Set Computer (RISC) Machines

[5]Available from http://www.openfst.org/twiki/bin/view/FST/.

[6]Available from http://www.netlib.org/blas/.

[7]Available from http://www.netlib.org/lapack/.

[8]Available from http://math-atlas.sourceforge.net/.

[9]Available from http://www.openblas.net/.

[10]Available from Bavarian Archive for Speech Signals (BAS), http://www.phonetik.uni-muenchen.de/Bas/BasVM1eng.html.

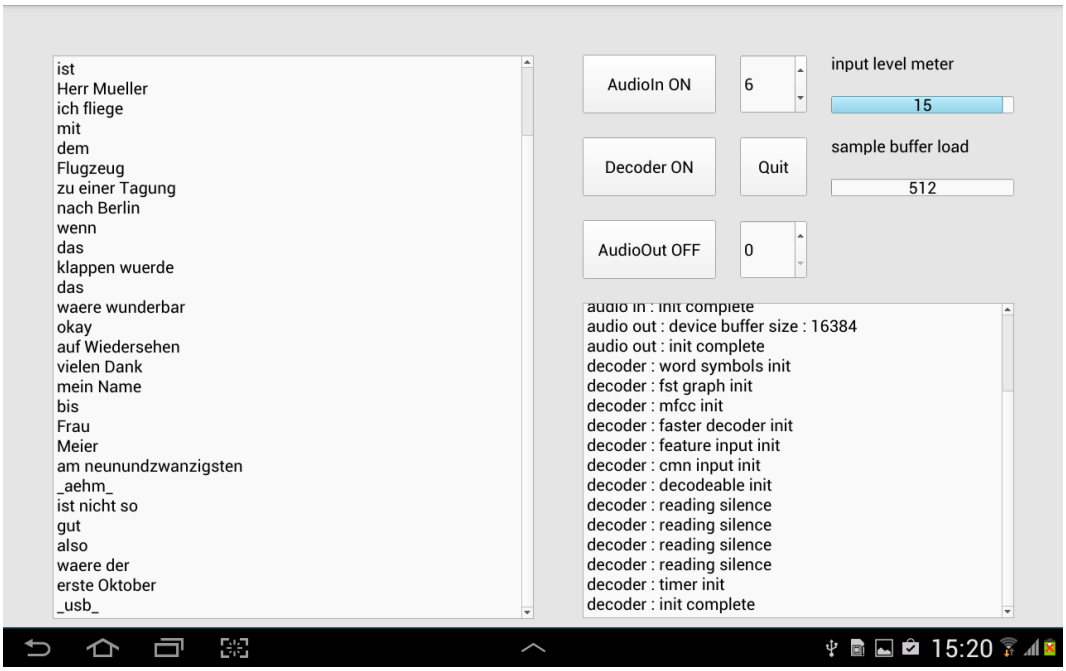

Fig. 1. Screenshot of the graphical user interface created for testing and demonstration purposes. The left text box shows the hypotheses, the right text box is used for debugging. Buttons and spin boxes are used to control live streaming audio from microphone, decoder and audio output. The horizontal bars show logarithmic microphone level and load of the buffer. The shown examples are decoding results of the spoken phrases *"Hallo, mein Name ist Herr Mueller. Ich fliege mit dem Flugzeug zu einer Tagung nach Berlin. Wenn das klappen wuerde, das waere wunderbar. Okay, auf Wiedersehen und vielen Dank."* and *"Mein Name ist Frau Meier, am neunundzwanzigsten (hesitation) ist nicht so gut, also waere der erste Oktober besser."*

- running the open-source large-vocabulary decoder locally on an Android device and
- processing live streamed audio with this decoder.

The use of Kaldi without extensive changes has some drawbacks with respect to the limited resources on mobile devices. The model sizes, especially of the language model, can be very large and, due to the restricted size of RAM, language models trained on large corpora e. g. the Wall Street Journal[11] (WSJ) corpus [14] cannot be used. The necessary high computational power reduces battery time contrary to the idea of free mobility.

The described approach of porting the Kaldi decoder offers many possibilities of further development and improvement. The quality of the audio data from microphone input is device dependent and has a markable influence of the recognition speed and accuracy. Especially the compact construction of smart devices, the microphone integrated into the casing along with computational hardware, touch screen and speakers, make the audio input prone to transferred vibrations and noise. A speech signal improvement with e. g. noise reduction is planned along with a voice activity detection, combined with application logic to run the decoder only when necessary to reduce CPU usage and save battery power. We will also look into using DNNs for acoustic modelling as natively supported by Kaldi.

A video of the application [15] is shown on Youtube. The source code will be subsequently made available to the community in a repository.

[11] Orderable from Linguistic Data Consortium (LDC), `https://catalog.ldc.upenn.edu/LDC94S13A`.

5. REFERENCES

[1] J. Adorf, "Web Speech API," Tech. Rep., KTH Royal Institute of Technology, Stockholm, Sweden, 2013.

[2] Apple Inc., *About Siri*, 2014, http://support.apple.com/kb/ht4992.

[3] Nuance Communications, Inc., *Dragon Dictation*, 2014, http://www.nuancemobilelife.com/support/USA/engusa/dragon-dictation.

[4] D. Huggins-Daines, M. Kumar, A. Chan, A. Black, M. Ravishankar, and A. Rudnicky, "Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices," in *Proc. of the ICASSP*, Toulouse, France, 2006.

[5] Carnegie Mellon University, *pocketsphinx on Android*, 2014, http://cmusphinx.sourceforge.net/wiki/tutorialandroid.

[6] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proc. of the ASRU*, Hawaii, USA, 2011.

[7] C. Gaida, P. Lange, R. Petrick, P. Proba, A. Malatawy, and D. Suendermann-Oeft, "Comparing Open-Source Speech Recognition Toolkits," Tech. Rep., Baden-Wuerttemberg Cooperative State University, Stuttgart, Germany, to appear.

[8] K. Vertanen, "Baseline WSJ Acoustic Models for HTK and Sphinx: Training Recipes and Recognition Experiments," Tech. Rep., University of Cambridge, Cambridge, UK, 2006.

[9] X. Yao, P. Bhutada, K. Georgila, K. Sagae, R. Artstein, and D. Traum, "Practical evaluation of speech recognizers for virtual human dialogue systems," in *Proc. of the LREC*, Malta, 2010.

[10] F. Morbini, K. Audhkhasi, R. Artstein, M. Van Segbroeck, K. Sagae, P. Georgiou, D. Traum, and S. Narayanan, "A reranking approach for recognition and classification of speech input in conversational dialogue systems," in *Proc. of the SLT*, Miami, Florida, USA, 2012.

[11] F. Morbini, K. Audhkhasi, K. Sagae, R. Artstein, D. Can, P. Georgiou, S. Narayanan, A. Leuski, and D. Traum, "Which ASR should I choose for my dialogue system?," in *Proc. of the SIGDIAL*, Metz, France, 2013.

[12] Baden-Wuerttemberg Cooperative State University, Stuttgart, Germany, *OASIS—Open-Source Automatic Speech Recognition In Smart Devices*, 2014, http://www.dhbw-stuttgart.de/themen/kooperative-forschung/fakultaet-technik/oasis.html.

[13] F. Schiel, *Verbmobil I - VM1*, Bavarian Archive for Speech Signals, Munich, Germany, 2012, http://www.phonetik.uni-muenchen.de/Bas/BasVM1eng.html.

[14] Linguistic Data Consortium, Philadelphia, USA, *CSR-II (WSJ1) Complete*, 1994, http://catalog.ldc.upenn.edu/LDC94S13A.

[15] C. Gaida, "Kaldi on Android demo," http://youtu.be/7etFg-vsSZI, 2014.