Implementation of Speech Recognition on Edge Devices

Shaik Shabana,
OTC,SSG
Intel Corporation
Bangalore,India
shaik.shabana@intel.com

Sujith Thomas,
OTC,SSG,
Intel Corporation,
Bangalore,India
sujith.thomas@intel.com

***Abstract*— In Today's world, deep learning and artificial intelligence are changing the way we approach and resolve life's problems. With the advancements in neural networks, the scope for deep learning based solutions is widening. Edge computing is taking the world by storm penetrating into segments never thought off. Implementing speech recognition on edge devices using deep learning techniques gives a wider foray to businesses to accelerate the processing power of hardware and leverage the user experience. This paper explains the challenges, techniques and provides an insight in the implementation of speech recognition models on edge devices. Edge devices with android as the operating system is taken as platform to analyze the problems and to quantify the benefits of such an implementation. This paper also showcases the process to integrate and enable speech model for on-device inferencing on android devices. Proposed method is unique in its way because of two reasons. Firstly, it opens the doorways of on-device inferencing with the help of neural network API's (NNAPI's) which can eliminate more resources, costs, time needed for cloud based inferencing, promote secure control of data, and leverage user experience in no time. Secondly it demonstrates the performance impact of the android runtime HAL on NNAPI's. There is very little research on deploying speech recognition using the android neural network runtime on edge devices to accelerate the processing and quantifying the performance impact. This research is aimed at developers who can deploy and enable speech recognition with ease and at low cost.**

Keywords*— *Deep Learning, Speech Recognition, Android, On Device Inferencing, NNAPI, Neural Network, Edge Devices

I. INTRODUCTION

With the progress of technology in the field of sensors, networks, storage, camera, process blocks etc. edge computing is coming a full circle with farfetched benefits of deploying such modules in everyday lives that is making a man's job more connected and easier. And the advent of deep learning and neural networks is making these modules perceive and perform more human like. With android as operating platform in the edge devices, the computing deployment and performance is reaping beyond limits.

Android has always been a popular option for edge devices because it is an open sourced Linux based software which provides a front end interface for the users which is easy to use. Android stack is mature, easy to maintain, and supports different hardware architectures.

With the advent of neural network API's in Android 8.1 and above, the traction for deep learning based solutions has gained pace. The android neural networks API (NNAPI) is an android C API designed for running computationally intensive operations for machine learning on mobile devices. NNAPI is designed to provide a base layer of functionality for higher-level machine learning frameworks (Tensorflow Lite, Caffe2, or others) that build and train neural networks. Neural network algorithms can be used to solve problems of image classification, predicting user behavior, speech recognition, predicting responses to user's query etc.[1]

The rise of artificial intelligence is grounded in the success of deep learning. Three major drivers that have caused the major breakthrough of deep neural networks: availability of huge training data, powerful computational infrastructure, advances in academia. This rising trend of artificial intelligence in day to day lives has propelled the growth of dedicated hardware/accelerators which can be embedded in the System on Chip(SoC) like google pixel's visual core, apple's neural engine in A11 bionic chip, intel's movidius stick etc. The additional chip in the SoC is complimentary to the already existing CPU and GPU with an aim for doing the AI tasks faster

Speech recognition is the ability of a program/software to capture the words spoken and convert them into machine readable format like text [2].The quality of speech recognition systems is assessed by the two factors:

1. Accuracy (error rate in converting spoken words to digital data)
2. Speed (inference time) how quickly can the words be recognized/inference

The biggest advantage of speech recognition lies in its ease of use. It needs the user to just talk and not do any action like tap, point, press etc. This simplistic approach is the only reason speech recognition is widely used in varied applications ranging from home automation, robotics, hands free computing, automated customer service, machine translation etc. Speech recognition is becoming a huge success in the field of deep learning because of its high accuracy rate and faster inference times. High accuracy rate is contributed by the growth of publicly available large datasets which can be used for training machine learning models. Inferencing is becoming faster due to the allocation of dedicated hardware (processor) and the advancements in the accelerating software which is making the speech recognition process seam so realistic.

Inferencing on edge devices can be either on-device or cloud based. On-device inferencing relates to the fact that the model is trained and stored on a device and inferencing will occur using the model on the same device. However in cloud based inferencing, the model is trained & stored on a cloud server, and any input that is to be inferenced is sent to the cloud where the inferencing occurs and the output is later sent on the device.

Presented at WinTechCon-2018, organized by IEEE CAS Bangalore Chapter, IEEE Bangalore Section, and IEEE WiE Council

There has been a lot of research on speech recognition techniques and methods to improve its accuracy. One such research revolves around analyzing on cloud and on device inferencing [3]. There has been research papers on MFCC [4], support Vector machine [5], HMM models [6] to build and improve speech recognition systems.

There are a lot of android applications that deploy speech recognition which rely on cloud based translators or speech API's that has been the legacy in traditional speech recognition systems. They do not exploit the advances made by machine learning techniques and the concept of dedicated AI hardware acceleration unveiled by android in its latest versions. There is very limited research on deploying speech recognition on android for edge devices and using the android neural network runtime to accelerate the processing and quantifying the performance impact. This research is aimed at developers who can deploy and enable speech recognition with ease and at low cost.

The paper is organized as follows: section 2 explains the android neural network API's and section 3 deals with the system methodology and important criteria's for modelling the system. Section 4 describes the experimental results. Section 5 explains the future scope of work and we conclude in the section 6.

II. ANDROID NEURAL NETWORK API RUNTIME

NNAPI is a novel feature introduced in android SDK 8.1 and above keeping in mind the paradigms deep learning is bringing to technology and mankind. NNAPI is the foundational layer for machine learning frameworks and libraries that allows the developers to train and deploy the machine learning models on android devices. Based on the application's requirement and hardware specifications, android neural network runtime efficiently distributes the workload across available on device processors, including neural network hardware, GPU's and DSP's. [1].In the absence of any dedicated AI chips, API falls back on to CPU to execute requests. The below diagram explains the changes in android framework.

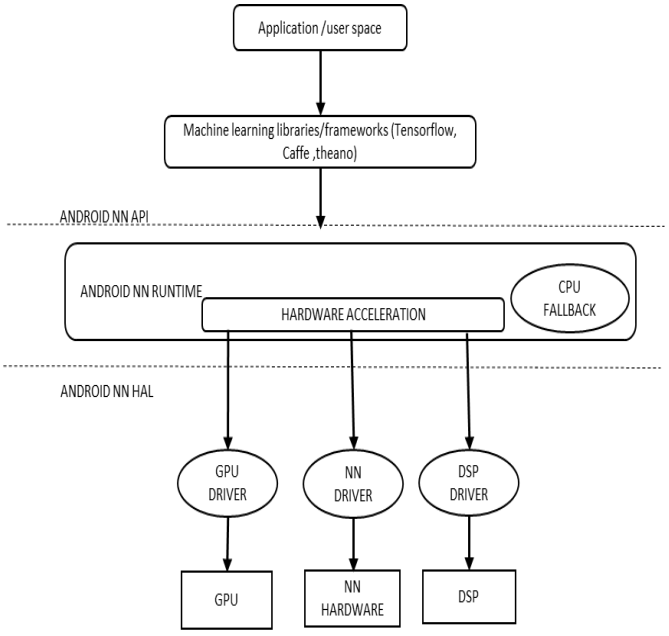

Fig 1: Android neural network runtime framework (Source: https://developer.android.com/ndk/guides/neuralnetworks/)

NNAPI is android's effort to bring human like intelligence to machines. With the advent of NNAPIs, the horizon of on-device inferencing has widened. By supporting on-device inferencing, NNAPI's not only keeps the phone's sensitive data on the device, but also reduces the latency and can be used even in the absence of network coverage. It reduces the cost as well since there is no necessity of any cloud for inferencing. NNAPI's make use of the highly computational AI hardware for inferencing, hence improving the inference speeds. But there are some trade-offs as well. Since the computation is occurring on the device, system utilization is high causing the batteries to drain faster. The size of the model is also a constraint as it may bloat up the application size.

III. SYSTEM METHODOLOGY

This section proposes a method to implement the speech recognition model on android with on-device inferencing capability. The below diagram shows the pictorial representation of this implementation.

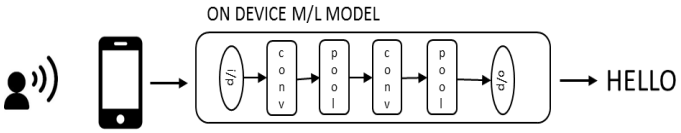

Fig 2: Pictorial representation of the system model

The following are the important criteria's and techniques that should be taken into account before the actual implementation.

1. Deep learning framework: There are a lot of open sourced popular deep learning frameworks like Tensorflow, caffe, torch, theano etc. which can be used to train and build machine learning models. We chose to use the Tensorflow framework because of its easy to use approach and advancements with reference to NNAPI's on android operating system. Using TensorflowLite models on android gives an additional advantage of using the hardware accelerated runtime inferencing capabilities, which means an additional processor can be used extensively for deep learning and computer vision computational needs without loading the CPU, GPU or DSP.

2. Selection of model: Since the inferencing is going to be on-device, the model chosen should not be of very huge size. The time to load the model also depends on the size of the model. The model should be comprehensive of an extensive dataset so that the accuracy is good. The trade-off between the size of the model and the dataset should be well-handled depending on the application where the model is deployed. Several Recurrent Neural Networks (RNN) and Convolution Neural Network (CNN) models can be trained for speech recognition. We chose the Wavenet model for speech to text conversion because the pre-trained

model with checkpoints is publicly available [7]. This pre-trained model is based on the open sourced VCTK corpus dataset which is a collection of audio files of speakers spoken in English in different accents.

3. Wavenet: Wavenet are a deep generative model of raw audio waveforms. They are used to generate speech and they seem more natural than the existing text to speech systems. They are popularly used for speech synthesis. Basically wavenets are Convolutional Neural Networks (CNN) that take an audio signal as input and synthesize the output sample by sample. The layers are arranged in the form of dilated stacks so that there are a number of input layers. Higher the number of input layers, higher is the receptive field of this network, which is the input for generating the next sample [8] [9].

   Wavenet can be used for speech recognition systems where the audio raw input is converted to text. Recurrent neural networks such as LSTM have been used in speech recognition applications because they allow for building models in long range contexts. With the layer of dilated convolutions that the Wavenet exhibits, the receptive field grows longer and hence they can be used in a simpler way than LSTM. For the needs of speech recognition, a mean pooling layer after the dilated convolution layer is added for down-sampling [10].

4. Input Preprocessing technique: pre-processing of raw input (audio files) is an essential component of Automatic Speech Recognition (ASR) systems. We use MFCC techniques to extract features that is the part of the audio signal which is good to identify the linguistic content from the audio waveforms [11]. The MFCC inputs are fed into the model.

5. Training the model: Training the model is an important step in perfecting the accuracy of the neural network that is deployed. Model can be trained on a highly computational CPU, GPU or using dedicated AI hardware. The batch size and number of epochs should be appropriately decided depending on the accuracy of the train data and test data.

IV. EXPERIMENTAL RESULTS

The methodology is exhibited in the form of an android application which uses Wavenet model to inference the input given from the microphone. Microphone can be used as an input source in which the user can utter any sentence for a duration of 5sec and then press the inference button. The sampling rate is set to 16 KHz. Deep learning model used is the Wavenet model. Chosen model is stored in the Tensorflow format (.pb format).

To maintain the consistency of input on all test devices, we have recorded some sentences (instead of using the microphone) and stored them as wav files which will be read into a short buffer before inferencing. This also helps to avoid the background noises from the environment which may vary person to person and prevent the loss of accuracy. Inference time is a key criteria which defines the time taken to inference what has been spoken by the user. Accuracy is also as much important as inference time.

The below table is a result of the application being tested on different test devices and the time taken to inference certain specific sentences.

Table1: Performance analysis of speech recognition module on popular market devices

Inference Time	SDK	"HOW ARE YOU"	"I AM A BOY"
Device 1	8.1	1.43s	1.55s
Device 2	8.1	0.93s	0.98s
Device 3	8.1	1.46s	1.58s

The test devices used for analysis are high end popular smartphones, the names of which cannot be revealed due to privacy issues. It is also evident that the inference times are close to 1-1.5 seconds, which can be further reduced optimized when the model is converted to TensorflowLite format. This makes the speech recognition module very realistic with minimal delay between the words uttered by the user and time for the module to infer.

With the introduction of android runtime HAL in android SDK 8.1 and above, if the NNAPI's are not enabled, as in the case of this application's model in Tensorflow format, the execution requests falls back to the optimized Tensorflow libraries on CPU. If the model is trained in the TensorflowLite format, then the flow of checks is as below:

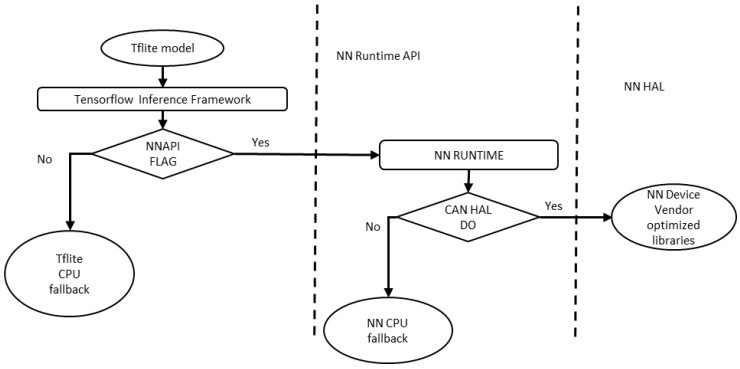

Fig 3: Flowchart describing the flow of checks for a TensorflowLite (Tflite) model in android SDK 8.1 and above

If the NNAPI Flag is enabled, then the execution goes to the NNAPI runtime, where the framework checks for NN hardware. If NN hardware is present, execution goes to the NN HAL where the code runs on the vendor specific optimized deep/machine learning libraries. Else it falls back to the NN runtime framework and executes requests in this layer on CPU.

The below set of results indicate the performance difference android is bringing to deep learning applications.

Table2: Performance analysis of Speech recognition modules on different Android versions

Inference Time	"HOW ARE YOU"	"I AM A BOY"
Device 3 (SDK 7.0)	2.1s	2.3s
Device 3 (SDK 8.1)	1.46s	1.58s

From the above table, it is evident that the performance of deep learning modules, here for example the speech module, is greatly improved from android versions 8.1 and above. It is because of the advent of the Android neural network runtime API which was propelled due to the huge demand for deep learning powered applications on mobile devices.

This application is used to showcase the true strength of speech recognition and how it can be enhanced with android as an operating system. It can be used in different use-cases for varied needs. For example, with minor changes to the proposed methodology, this model can be deployed to identify panic events in automotive to prevent accidents. To elaborate the use-case, model should be trained with words like "SAVE","HELP", "THIEF" etc. A person when in trouble first screams or cries. If the model detects such events along with the sentiment with good accuracy, the proposed model is going to be a preventive solution for kidnapping and trafficking cases in automotives.

V. FUTURE SCOPE OF WORK

The Tensorflow speech model can be converted into TensorflowLite format which allows us to enable NNAPI on android 8.1 and above. This will significantly reduce the inference times because on enabling NNAPI's, the framework will be able to leverage the computational power of the dedicated AI hardware for deep learning needs. The model can be trained on a broader dataset instead of using one corpus which will improve the accuracy of the model.

VI. CONCLUSION

A method to develop an android application demonstrating the speech recognition module is proposed. This paper explains the importance of android neural network APIs, quantifies the performance metrics and how they can be leveraged to enhance the inference timings of deep learning model. The techniques proposed for implementation are easy to use and can be deployed by any developer to enable speech recognition module on edge devices.

AUTHORS

Shaik Shabana is a software engineer, working at Intel Corporation. She has a master's degree in power electronics from visweshvaraya technological university and bachelor's degree in electrical engineering from the PES Institute of technology, Bangalore. She has 3 years of industry experience in android power & performance, deep learning tensorflow and caffe2 frameworks, application development, android tools, automation etc. She has worked on benchmarking of different hardware like CPU, GPU, DSP, Movidius, for deep learning use-case and have knowledge on industry wide famous benchmarks/models.

Sujith Thomas is an engineering manager, working at Intel Corporation. He has 17 years of industry experience in the area of power & performance, power & thermal management, and machine learning & artificial intelligence. He was involved in concept, design and execution of various Intel android based products segments including phones, tablets and car infotainment systems. He has a Bachelor's degree in Computer Science and Engineering from R.E.C. Calicut.

REFERENCES

[1] https://developer.android.com/ndk/guides/neuralnetworks/

[2] http://www.streetdirectory.com/travel_guide/139545/technology/key_differences_between_speech_recognition_and_voice_recognition.html

[3] Tian Guo,"Cloud-based or On-device:An Empirical Study of Mobile Deep Inference" in 2018 IEEE International Conference on Cloud Engineering (IC2E).Available: https://arxiv.org/pdf/1707.04610.pdf

[4] Wei Han, Cheong-Fat Chan, Chiu-Sing Choy Kong-Pang Pun," An efficient MFCC extraction method in speech recognition" in 2006 IEEE International Symposium on Circuits and Systems.

[5] M. Tharaniya soundhari, S. Brilly Sangeetha," Intelligent interface based speech recognition for home automation using android application" in 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)

[6] Ibrahim Patel,Dr.Y Srinivas Rao," Speech Recognition Using Hidden Markov Model With MFCC-Subband Technique" in the proceedings of 2010 International Conference on Recent Trends in Information, Telecommunication and Computing.

[7] https://github.com/buriburisuri/speech-to-text-wavenet

[8] http://deepsound.io/wavenet_first_try.html

[9] https://deepmind.com/blog/wavenet-generative-model-raw-audio/

[10] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu," WaveNet: A Generative Model for Raw Audio".Available: https://arxiv.org/abs/1609.03499

[11] http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/