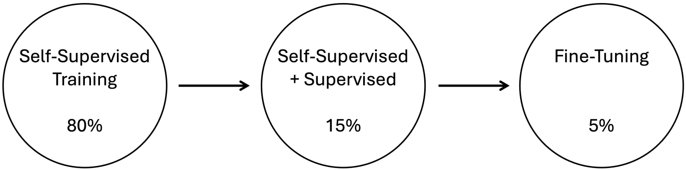
On-Device Constrained Self-Supervised Learning for Keyword Spotting via Quantization Aware Pre-Training and Fine-Tuning

Gene-Ping Yang, Yue Gu, Sashank Macha,
Qingming Tang, Yuzong Liu

University of Edinburgh, Amazon

Introduction

- **Large** self-supervised models are primary building blocks for speech foundation models.
  - Google USM (BestRQ): 0.6B / 2B
  - SeemlessM4T (w2v-BERT): 1.2B / 2.3B
  - AudioPaLM (w2v-BERT): 8B

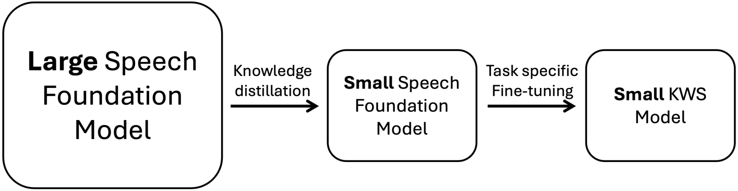


[1] Google USM: Scaling Automatic Speech Recognition Beyond 100 Languages

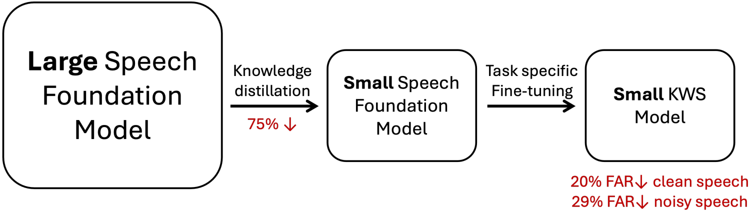
Limitation for on-device application

- Due to substantial memory footprint, deploying these large models on edge devices is impractical.
- The need for a continuous internet connection and potential network latency are limitations when accessing these models through cloud APIs.

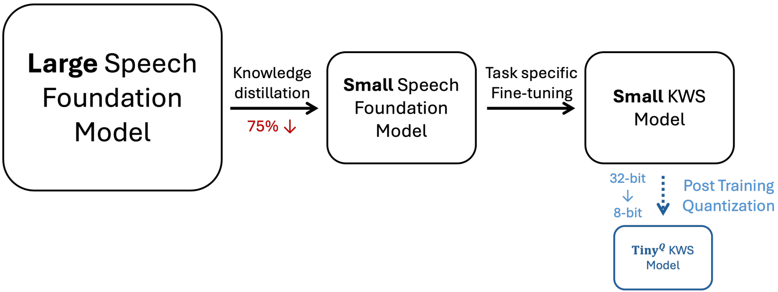
Setting


Large Speech Foundation Model
Knowledge distillation
Small Speech Foundation Model
Task specific Fine-tuning
Small KWS Model

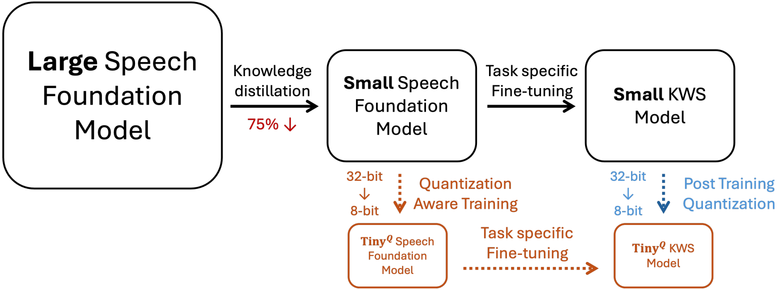
Setting


Large Speech Foundation Model
Knowledge distillation
75% ↓
Small Speech Foundation Model
Task specific Fine-tuning
Small KWS Model
20% FAR↓ clean speech
29% FAR↓ noisy speech

Setting


Large Speech Foundation Model
Knowledge distillation
75% ↓
Small Speech Foundation Model
Task specific Fine-tuning
Small KWS Model
32-bit
↓
8-bit
Post Training Quantization
Tiny^Q KWS Model

Setting


Large Speech Foundation Model
Knowledge distillation
75% ↓
Small Speech Foundation Model
Task specific Fine-tuning
Small KWS Model
32-bit ↓ 8-bit
Quantization Aware Training
32-bit ↓ 8-bit
Post Training Quantization
Tiny^Q Speech Foundation Model
Task specific Fine-tuning
Tiny^Q KWS Model

Motivation

- Our goal is to develop a tiny speech foundation model, which can be a better initialization for various downstream tasks requiring restricted memory footprint.
  - Achieved through knowledge distillation by reducing model **width** and **depth**[1].
  - Reduce the **bit size** of model weights and activations.

→ Quantized self-superivsed models via quantization aware self-supervised training (QAT).

[1] On-Device Constrained Self-Supervised Speech Representation Learning for Keyword Spotting via Knowledge Distillation, Yang et al., Interspeech 2023

Fixed Point Quantization

- 32-bit floating point $\rightarrow$ 8-bit fixed point integer (INT8)
- Value set for 8-bit consists of 2^8 values:
  - $\{0, 1, 2, ..., 254, 255\}$
  - linear transformation:
    $\{-1, -\frac{255}{256}, -\frac{254}{256}, ..., -\frac{1}{256}, 0, \frac{1}{256}, ..., \frac{254}{256}, \frac{255}{256}\}$

[1] Fixed-point quantization aware training for on-device keyword-spotting, Macha et al., ICASSP 2023

Fixed Point Quantization

- Pros
  - INT8 multiplications consumes 18.5x less energy and half the memory compared to FP32
  - INT8 model size is 4x less
- Cons
  - **Training** with INT8 is significantly slow (on FPGA)
  - Direct post training quantization (PTQ) from FP32 to INT8 cause severe information loss

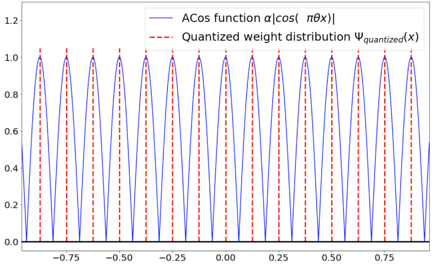
[1] Fixed-point quantization aware training for on-device keyword-spotting, Macha et al., ICASSP 2023

Quantization Aware Training (QAT)

- Forward Quantization
  - Forward pass: simulate 8-bit operation
  - Gradient: 32-bit
- Soft Quantization
  - Forward pass: 32-bit
  - Additional loss represents the discrepancy between the entries in a 32-bit weight matrix and their counterparts when quantized according to a specified fixed-point (FXP) scheme.

Soft Quantization - ACR

- Absolute Cosine Regularization (ACR)[1]
  - $L_{\text{ACR}} = -\sum_i |cos(\pi f w_i)|$
  - The peaks resemble the FXP value set
  - If a model weight aligns with one of the peak, the gradient is 0



[1] Quantization Aware Training with Absolute-Cosine Regularization for Automatic Speech Recognition

Input and Activation Quantization

- Min-max scaling[1] (under linear operation)
- Quantize A into full INT8 range [0, 255]

$$A_{\text{INT8}} = \left\lfloor (A - \underbrace{A_{min})}_{\text{shift}} \underbrace{\frac{255}{A_{max} - A_{min}}}_{\text{scale}} \right\rceil$$

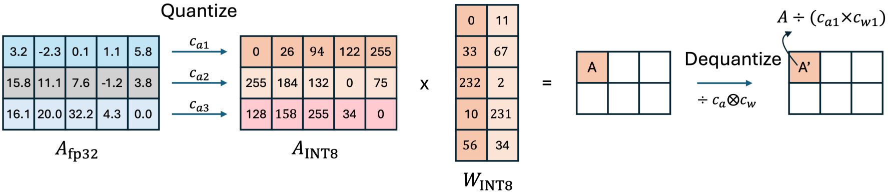
- De-Quantize into original dynamic range (256 values)

$$A_{\text{DeINT8}} = \left\lfloor (A - A_{min}) \frac{255}{A_{max} - A_{min}} \right\rceil \times \frac{A_{max} - A_{min}}{255} + A_{min}$$

[1] LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, NeurIPS 2022

Input and Activation Quantization

- Dynamic scaling quantization (Vector-wise quantization[1,2])
  - View matrix multiplication as independent inner products
  - Assign different scaling constant c_a to each row of A and c_w each column of W

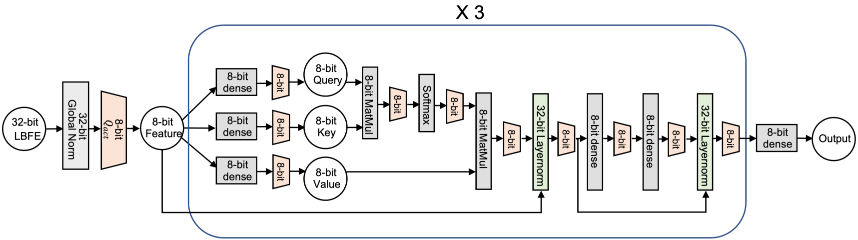


[1] LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, NeurIPS 2022
[2] 8-bit Optimizers via Block-wise Quantization, Dettmers et al., ICLR 2022

Overview

- Problem Setup
  - Train models under FP32 while being aware of INT8 scheme (Quantization Aware Training, QAT)
- Scientific Questions
  - How does QAT affect the expressiveness of self-supervised models?
  - Which QAT technique and scheme is best for self-supervised stage and fine-tuning stage?

Diagram of our QAT Transformer



- Self-supervised Learning with QAT: $L_{SSL\_QAT} = L_{SSL} + \alpha L_{ACR}$
- Downstream fine-tuning with QAT: $L_{DS\_QAT} = L_{DS} + \alpha L_{ACR}$

Experiment setup

- Train set: 16.6k hours of de-identified audio recording
- Test set: 85 hours of clean and noisy condition
- Self-supervised Methods: Autoregressive Predictive Coding[1]
- Downstream: Keyword Spotting

[1] Autoregressive predictive coding: A comprehensive study, Yang et al., JSTSP 2022

Model Architecture

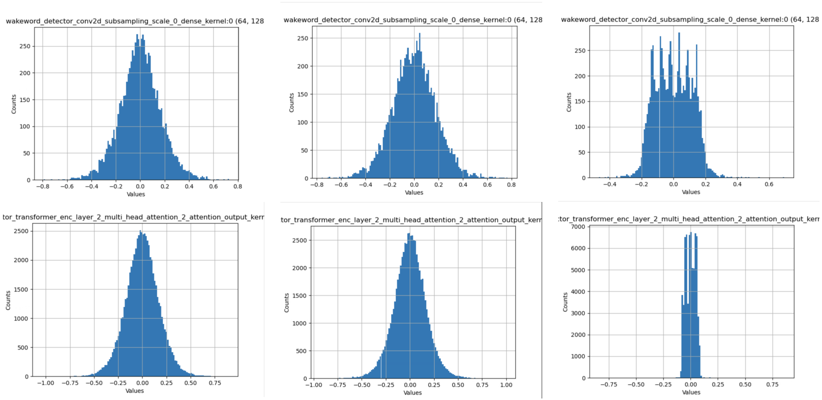
- 3-layer transformer, with 4 attention heads and 128 hidden dimension
- 400K parameters, where 99.8% of the parameters will be quantized (both weights and biases)

S3RL QAT + KWS QAT

	S3RL	KWS	Final Precision	Relative FAR	
				Normal	Playback
(1)	FP	FP	w32a32	1.0	1.0
(3)	FP	FP	$w8a8_{Dyn}$	1.39	1.32
(5)	ACR+Dyn	ACR+Dyn	$w8a8_{Dyn}$	1.86	1.75
(9)	Dyn	Dyn	$w8a8_{Dyn}$	1.02	1.01

[1] S3RL = **S**elf-**S**upervised **S**peech **R**epresentation **L**earning

What happen after ACR quantization?



FP_{32} Non-ACR + Dyn ACR + Dyn

Quantization Utilization

- Efficiency: percentage of the quantized value set being used

Setup	Training	Zeros ↓	Efficiency ↑	Compression ↓
KWS	FP	4.7%	41.8%	23.8%
S3RL + KWS	FP	4.3%	51.2%	23.9%

Figure: Average percentage over all quantized model weights

Quantization Utilization

- Efficiency: percentage of the quantized value set being used

Setup	Training	Zeros ↓	Efficiency ↑	Compression ↓
KWS	FP	4.7%	41.8%	23.8%
	Dyn	3.7%	48.4%	24.1%
S3RL + KWS	FP	4.3%	51.2%	23.9%
	Dyn	**3.7%**	**53.5%**	**24.1%**

Figure: Average percentage over all quantized model weights

Quantization Utilization

- Efficiency: percentage of the quantized value set being used

Setup	Training	Zeros ↓	Efficiency ↑	Compression ↓
KWS	FP	4.7%	41.8%	23.8%
	Dyn	3.7%	48.4%	24.1%
	ACR + Dyn	11.0%	20.0%	22.2%
S3RL + KWS	FP	4.3%	51.2%	23.9%
	Dyn	**3.7%**	**53.5%**	**24.1%**
	ACR + Dyn	11.0%	20.5%	22.3%

Figure: Average percentage over all quantized model weights

Summary

- We proposed QAT with no restriction on model weights and dynamic quantization on activations, achieving superior performance among various QAT methods.
- ACR is excessively restrictive for model weights, primarily due to the normal distribution pattern of the weights, pushing model weights toward 0.
- A combination of dynamic quantization on activations without ACR yields the best results, with performance comparable to the 32-bit model in an 8-bit setting.
- Self-supervised pre-training improves the effectiveness of using quantized values, as opposed to models without pre-training.