# Exploring the Relative Value of Collaborative Optimisation Pathways (Student Abstract)

**Sudarshan Sreeram***

Department of Computing, Imperial College London
180 Queen's Gate, South Kensington, London
SW7 2AZ, United Kingdom
sudarshan.sreeram19@imperial.ac.uk

## Abstract

Compression techniques in machine learning (ML) independently improve a model's inference efficiency by reducing its memory footprint while aiming to maintain its quality. This paper lays groundwork in questioning the merit of a compression pipeline involving all techniques as opposed to skipping a few by considering a case study on a keyword spotting model: DS-CNN-S. In addition, it documents improvements to the model's training and dataset infrastructure. For this model, preliminary findings suggest that a full-scale pipeline isn't required to achieve a competent memory footprint and accuracy, but a more comprehensive study is required.

## Introduction

ML empowers solutions to wide-ranging real-world problems. Traditionally, cloud-based servers power the lifecycle of production ML models, from training to inference. However, server-side scaling for model deployment isn't financially nor environmentally sustainable (Menghani 2021). Furthermore, centralised processing of user data doesn't necessarily provide privacy guarantees. These drawbacks, among others, have motivated a shift in consumer-facing solutions towards enabling efficient, privacy-preserving on-device inference and, more generally, pushing practical ML applications to the Edge (e.g. TinyML). For instance, iOS 15 (2021) enabled Siri to function offline on iPhones with a competent Neural Engine (hardware accelerator). In addition, popular tasks in machine vision and language have seen a rise in mobile-friendly model offerings: MobileNet, MobileViT and MobileBERT, to name a few.

### Model Compression

Edge devices, often commodity microcontrollers (MCUs), have much less computational resources than mobile phones and add further constraints to a model's memory footprint, resource usage and performance (latency and quality). Pruning, weight clustering and quantisation are compression techniques that independently aid in reducing this footprint while improving inference latency and aiming to maintain quality (accuracy), thus preparing a model for deployment in such resource-constrained settings (Han, Mao, and Dally 2015). Collaborative optimisation is a compression pipeline that achieves an accumulated optimisation effect by chaining techniques while preserving their contributions. By nature, this process presents a sheer number of optimisation paths for deployment (incl. fine-tuning) that could prove daunting for those seeking quick deployment solutions, especially for sizeable models with significant training overhead.

### Methodology

This paper questions the merit of a full-scale pipeline involving pruning, clustering and quantisation aware training (PCQAT) as opposed to skipping either or both pruning and clustering, which would reduce re-training costs and could yield better metrics. In specific, it considers DS-CNN-S, a lightweight keyword spotting model trained on Google SpeechCommands v0.0.2 (Zhang et al. 2017), as an initial case study. The TensorFlow Model Optimisation Toolkit (TF-MOT) facilitates the compression pipeline.

## Keyword Spotting

Always-on keyword spotting is a crucial part of the human-computer interface for virtual assistants (e.g. "Hey Siri" or "Ok Google"). The DS-CNN-S model, which uses depthwise-separable convolutions to achieve a compact architecture, classifies audio across 12 labels. The dataset preprocessing stage for this model involves extracting Mel-Frequency Cepstral Coefficients (MFCCs) from labelled audio clips (Fig. 1). The TensorFlow V2-port of this model achieves an accuracy of 93.87%, incurring a training time of 25.18 minutes; the GPU-enabled training spanned 75 epochs, each taking about 23 seconds to process.

### Improved Infrastructure

A secondary contribution of this paper is a complete rewrite of the DS-CNN-S model architecture, dataset processing and training infrastructure. The use of GPU-supported operations to compute MFCCs, dropout layers to improve generalisation, label weights to offset severe dataset imbalance, and parallelised pre-fetching and caching mechanisms result in nearly 3x reduction in training time (8.5 minutes for 75 epochs) with a minor boost in accuracy to 95.05% after fine-tuning. With early stopping measures, the effective training time drops to just three minutes!

*The author conducted this work as an intern at Arm Limited

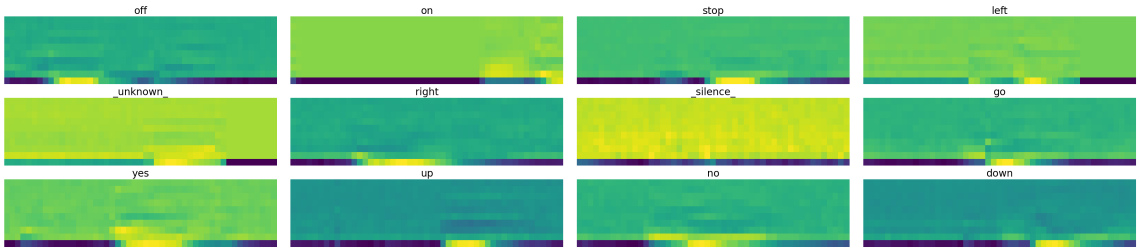


Figure 1: MFCCs (rotated 49 x 10 images) of 12 sample audio clips from the dataset, each representing a unique label. The padded one-second clips (16 KHz) were processed to construct a power spectral density matrix (with 513 frequency banks and 49 windows). The linear scale spectrogram is then transformed to the mel-scale, where the first 10 mel bins were extracted.

## Experiments

The case study considers four compression pipelines based on quantisation aware training (QAT) as it benefits model accuracy. The results are documented in Table 1. In all cases, re-training for each contributing technique lasted 15 epochs. Furthermore, the model-specific hyperparameters were maintained to match that of the baseline. The time per inference metric was computed using performance estimates from Ethos™-U Vela 3.4.0 for Arm® Ethos™-U55 micro NPU with 256 MACs. The model size reflects that of the Gzip-compressed TFLite model.

| Model | Model Size (KB) | Evaluation Accuracy | Time Per Inference ($\mu s$) |
|---|---|---|---|
| Baseline | 35.76 | 95.05 | N/A |
| QAT | $30.42 \pm 0.011$ | $\mathbf{94.94 \pm 0.13}$ | 118.95 |
| PQAT | $25.18 \pm 0.020$ | $94.51 \pm 0.06$ | 113.42 |
| CQAT | $26.52 \pm 0.030$ | $94.79 \pm 0.05$ | $116.47 \pm 0.07$ |
| PCQAT | $\mathbf{24.66 \pm 0.010}$ | $94.50 \pm 0.07$ | $\mathbf{113.20 \pm 0.04}$ |

Table 1: Summary of performance metrics for different optimisation strategies, including that of the baseline model.

**QAT** This barebones strategy, which downsamples the precision of weights from FP32 to INT8, reduced the model footprint by 15% with minimal to no loss in accuracy. The re-training time was minimal at only 3.1 minutes.

**PQAT** The model was pruned at 50%, 75%, and 87.5% sparsities. The last two cases see catastrophic degradation in accuracy: 87.03% and 50.43% (respectively). Table 1 documents the first case. Here, the model footprint was reduced by 30%, with a re-training time of 5 minutes.

**CQAT** The model was clustered with 8, 16 and 32 clusters, with all three performing better than the original FP32 baseline (93.87%): 94.34%, 94.79% and 94.68% (respectively). The case with 16 clusters is reported in Table 1. The model footprint was reduced by 26%, and the re-training time is 5 minutes, which drops with more clusters, a pattern that draws insight on compressing larger, architecturally-similar models.

**PCQAT** The model is pruned at 50% sparsity and clustered at 8, 16, and 32 clusters: 94.06%, 94.36% and 94.5%. Table 1 documents the case with 32 clusters. The compression benefit is only slightly better than that of PQAT while the re-training time is considerably worse at 8 minutes.

## Conclusion and Future Work

All pipelines produced a model with accuracy within 0.5% of the baseline. While PCQAT achieves the best model size and inference frequency, the 1.6x increase in re-training time relative to both CQAT and PQAT prompts further investigation at scale. For the DS-CNN-S model, a full-scale compression pipeline isn't necessary to achieve a competent memory footprint and accuracy. Further fine-tuning, such as per-channel clustering or m-by-n structured pruning, could reveal hidden trends. DS-CNN-S is a *small-dense* model, and applying collaborative optimisation on DS-CNN-L, which is 10x as large, to produce a *large-sparse* model could improve accuracy (Zhu and Gupta 2017). The lightweight nature of this model enabled a quick exploration of the optimisation space. Larger models (e.g. MobileNet-V2) would require smarter strategies such as selective clustering or pruning. Generalisation, however, warrants further comprehensive investigation to uncover trends among a wide range of architecturally-similar models.

## Acknowledgments

The author conducted the work presented in this paper while interning at Arm Limited (Cambridge, UK) as an ML Software Engineer and is grateful to Dr Elena Zhelezina and Dr Anton Kachatkou for their continued support, constructive comments, and insightful discussions.

## References

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv:1510.00149.

Menghani, G. 2021. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. arXiv:2106.08962.

Zhang, Y.; Suda, N.; Lai, L.; and Chandra, V. 2017. Hello Edge: Keyword Spotting on Microcontrollers. arXiv:1711.07128.

Zhu, M.; and Gupta, S. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv:1710.01878.