Characterising the Role of Pre-Processing Parameters in Audio-based Embedded Machine Learning

Wiebke Toussaint*
Delft University of Technology
Delft, Netherlands

Akhil Mathur
Nokia Bell Labs
Cambridge, UK

Aaron Yi Ding
Delft University of Technology
Delft, Netherlands

Fahim Kawsar
Nokia Bell Labs
Cambridge, UK

ABSTRACT

When deploying machine learning (ML) models on embedded and IoT devices, performance encompasses more than an accuracy metric: inference latency, energy consumption, and model fairness are necessary to ensure reliable performance under heterogeneous and resource-constrained operating conditions. To this end, prior research has studied model-centric approaches, such as tuning the hyperparameters of the model during training and later applying model compression techniques to tailor the model to the resource needs of an embedded device. In this paper, we take a data-centric view of embedded ML and study the role that pre-processing parameters in the data pipeline can play in balancing the various performance metrics of an embedded ML system. Through an in-depth case study with audio-based keyword spotting (KWS) models, we show that pre-processing parameter tuning is a remarkable tool that model developers can adopt to trade-off between a model's accuracy, fairness, and system efficiency, as well as to make an embedded ML model resilient to unseen deployment conditions.

CCS CONCEPTS

• **Hardware → Emerging tools and methodologies**; • **Computing methodologies → Neural networks**; **Speech recognition**; • **Social and professional topics →** *User characteristics*.

KEYWORDS

embedded machine learning, audio keyword spotting, pre-processing parameters, fairness

ACM Reference Format:
Wiebke Toussaint, Akhil Mathur, Aaron Yi Ding, and Fahim Kawsar. 2021. Characterising the Role of Pre-Processing Parameters in Audio-based Embedded Machine Learning. In *The 3rd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things (AIChallengeIoT 21), November 15–17, 2021, Coimbra, Portugal.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3485730.3493448

1 INTRODUCTION

With the widespread deployment of IoT devices, porting and deploying machine learning (ML) models to embedded devices and running them directly on-device is becoming a priority research area [17, 23]. In a typical embedded ML lifecycle involving deep neural networks (DNNs), model training is done offline and involves experimenting with various model architectures and training hyperparameters (e.g., learning rate, batch size, optimizers) with the goal of achieving the best prediction accuracy on a test set. Thereafter, the selection of a model for deployment is done by considering the trade-offs between prediction accuracy and system-focused metrics such as inference latency and energy consumption. To this end, model developers adopt a few common approaches: i) a model which can provide an acceptable prediction performance with as few parameters as possible (e.g., less deep neural architectures) is preferred for deployment, or ii) techniques such as pruning [25] or quantization [7] are applied to compress the model and tailor it to the resource requirements of the embedded device.

This model-centric view to embedded ML has seen extensive research in the past few years [5]. In contrast, we take a data-centric view [1] to embedded ML in this paper. We study the impact of pre-processing parameters on input features in the embedded ML pipeline, and their subsequent influence on the model's prediction performance and system efficiency. While seemingly innocuous, pre-processing parameters directly influence the dimensionality and data distribution of the data input to the ML model, which subsequently can have implications for the model's inference latency and prediction performance.

We focus our investigation on an audio keyword spotting (KWS) task. Audio KWS is one of the most popular embedded ML workloads currently, and an essential component of modern voice assistants in mobile phones and wearable devices such as earbuds [4, 10, 11, 13]. In the data pipeline of a KWS task (detailed in §2.1 and illustrated in Figure 1), pre-processing parameters influence how the raw speech waveform is segmented into frames, which type of numerical operations are done on the segmented frames (e.g., extraction of Mel spectrograms or cepstral coefficients), and what dimensionality the input features to the model have. To date, little research has been done to understand the role of these important parameters in the context of training and deploying embedded ML models.

While prediction accuracy and system efficiency are the two primary metrics for evaluating models in the embedded ML community, we additionally advocate for evaluating *model fairness*. Many IoT

[*] This work was done while the author was an intern at Nokia Bell Labs Cambridge.

This work is licensed under a Creative Commons Attribution-NonCommercial International 4.0 License.
SenSys '21, November 15–17, 2021, Coimbra, Portugal
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9097-2/21/11.
https://doi.org/10.1145/3485730.3493448

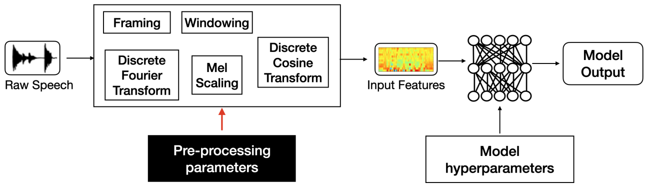


Figure 1: Audio processing pipeline during training and inference.

devices are consumer-centric, which makes fairness a particularly important metric to ensure that applications are unbiased and work reliably for all user groups. The prediction outputs of embedded KWS models, for example, typically trigger other applications. If used in critical applications, like voice-activated emergency response for elderly care, it is important that an embedded KWS model is not only resource-efficient and accurate, but also fair towards user groups with different speech characteristics. Evidence in other applications has shown that ML models can discriminate against certain user groups if fairness is not taken into consideration during model development [15, 16, 19]. Fairness is thus increasingly considered an important property of machine learning systems [14]. Moreover, regulatory shifts are bound to require ML models to be fair and non-discriminatory in some jurisdictions in future [2], and are a further motivation for evaluating fairness. Hence, in addition to characterizing the role of pre-processing parameters on a model's prediction accuracy, we also evaluate their impact on model fairness.

This paper makes the following contributions:

- We evaluate the impact of pre-processing parameters on the prediction accuracy of embedded audio KWS models, and consider the resulting effect on system efficiency.
- We introduce a metric to compute fairness of an embedded KWS model and present the first-ever evaluation of the impact of pre-processing parameters on model fairness.
- We investigate the role of pre-processing parameters when deployment conditions vary significantly from assumptions made during training, and show that choosing the right parameters can improve a model's robustness in such challenging deployment conditions.

2 BACKGROUND

This section discusses background and related work on audio KWS and model fairness.

2.1 Overview of audio KWS

The processing pipeline of an audio KWS task is depicted in Figure 1 and works as follows [6, 20]: raw input speech segments are split into overlapping, short time duration frames using a sliding window approach. *Frame length* and *frame step* are the two important parameters that respectively define the duration of each frame and the step size by which the sliding window is moved. *Frame step* is usually set to a certain fraction of the *frame length*. Next, a *windowing function* such as the Hamming window is applied to each frame to reduce spectral leakage. Thereafter, the frames are processed using Discrete Fourier-Transform to obtain a frequency-domain power spectrum. To mimic the non-linear hearing perception of the human ear, triangular filters on the Mel scale are applied to the power spectrum to obtain filter bank features. A log operation on the filter bank features is applied to obtain *log Mel spectrograms*. Optionally, Discrete Cosine Transform (DCT) is applied to de-correlate the log Mel spectrograms and yield a compressed representation of the filter banks in the form of cepstral coefficients, also known as *MFCCs*. The features (either the log Mel spectrograms or MFCCs) from all frames are concatenated and often mean-normalized to form a two-dimensional (2D) representation of the speech signal, which is then fed to a deep neural network for training and inference.

Embedded ML trade-offs The choice of pre-processing parameters can have a profound impact on the performance of an audio KWS model in an embedded system. Frame length and frame step together determine the temporal dimension of the 2D features that are fed to a DNN, and the number of log Mel spectrogram or MFCC features determine the length of features in each time segment. Together, these features influence the dimensions of the input data to the model, which in turn impacts the number of computations during inference. Table 1 shows the impact of various parameters on the system efficiency of an audio KWS pipeline. For instance, longer frame length and frame steps are better for system performance, as they lead to fewer frames and subsequently, smaller input feature size for the ML model. Similarly, by using fewer Mel bins for log Mel spectrograms or fewer MFCC features, we can also reduce the input feature size, which in turn enhances the inference speed on an embedded device.

Parameter	Impact on System Efficiency	Values used in this study
frame length (ms)	longer is better	20, 25, 30, 40
frame step (% of frame length)	longer is better	0.4, 0.5, 0.6
window type	none	Hamming, Hann
feature type	depends on size	log Mel spectrogram, MFCC
# Mel bins	fewer is better	20, 26, 32, 40, 60, 80
# MFCC coefficients	fewer is better	None, 10, 11, 12, 13, 14

Table 1: Pre-processing parameters, their potential impact on system-level metrics at inference and parameter options investigated in this study.

Even though these parameter choices could serve as an important tool during model development to design efficient audio KWS pipelines, little is known about their impact on model performance. As an illustration, Table 2 lists several prior works which have focused on embedded KWS. Most studies adopt a fixed set of standardized feature extraction parameters borrowed from classical audio processing literature (e.g. 25ms frame length, 40% frame step) and do not evaluate the impact of varying these parameters on model performance. In this paper we aim to characterize the impact of these parameters, in order to uncover how they can be tuned to balance fairness, inference accuracy and resource efficiency to suit the goals and requirements of an embedded ML application.

2.2 Fairness in ML

Fairness issues constitute a discriminatory action, typically against an individual or a group of people with one or more protected attributes. Protected attributes can be location and context dependent, and are often legally protected. EU non-discrimination law, for example, prohibits both direct and indirect discrimination based on race

Paper	frame length	frame step (% of frame length)	feature type	feature size
Tucker et al. [20]	25ms	0.4	log mel	20
He et al. [9]	25ms	0.4	log mel	80
Chen et al. [6]	25ms	0.4	log mel	40
He et al. [10]	25ms	0.4	log mel	80
Alvarez et al. [4]	30ms	0.33	log mel	40
Higuchi et al. [11]	25ms	0.4	MFCC	13
Zhang et al. [24]	40ms	0.5	MFCC	40

Table 2: Prior works on audio KWS focus on model optimization for embedded devices and do not evaluate the impact of pre-processing parameters.

and ethnicity, gender, religion and belief, age, disability, or sexual orientation [21]. Fairness in ML has become an important area of study over the past decade [14]. In the speech recognition domain, it is well known that automated processing techniques are sensitive to demographic attributes of speakers [8]. Commercial products have been found to exhibit discriminatory behaviour, for example automated caption systems that have a higher word error rate for speakers of colour [18], racial disparities that exist in speech-to-text systems [12], and speaker verification models that produce worse predictions for some nationalities, and for female speakers of most nationalities [19]. However, little work has been done to investigate the fairness of ML systems deployed on embedded devices. With commercial deployments of ML-enabled devices having reached global scale, investigating and evaluating their fairness is a matter of necessity.

3 EXPERIMENT SETUP

In this section, we introduce the dataset and evaluation metrics, along with various parameters studied in this paper.

3.1 Pre-processing parameters

Table 1 lists the parameters and their values that we considered during training, as well as the expected impact that parameter values will have on system efficiency at inference. As our objective is to study the trade-offs of parameter choices for model performance and fairness during inference, we focus on feature extraction and model architecture parameters. Other hyperparameters such as learning rate, number of training iterations etc. which do not have a direct impact on inference efficiency are not studied in this paper.

We experiment with two types of convolutional neural architectures originally proposed in [17] and later implemented in the TensorFlow framework [3], namely CNN and low-latency CNN (llCNN). CNN consists of two convolutional layers followed by one Dense hidden layer, while llCNN consists of one convolution layer followed by two Dense hidden layers. The authors showed that llCNN, by virtue of having less convolution operations, is more optimized for embedded KWS. We trained each of the architectures with combinations of all the pre-processing parameters listed in Table 1. As discussed earlier, longer frame lengths and frame steps, and fewer Mel bins and MFCC coefficients reduce the dimensions of the input features to the neural network, and hence they lead to higher inference efficiency. In Section 4 we present a detailed characterization of the impact that these pre-processing parameters have on inference accuracy and fairness.

3.2 Dataset

We use the Speech Commands [22] dataset for this study. The dataset consists of 104,541 spoken keywords from 35 keyword classes such as *Yes, No, One, Two, Three* recorded at a 16KHz sample rate. To investigate fairness, we focus on speakers' sex, the distinction between biological and physical characteristics of male and female speakers, as a key attribute for which the model should be fair. In other words, we want to evaluate whether an embedded KWS model trained with a certain parameter configuration provides similar performance for male and female speakers. To this end, we label all the audio samples in the dataset with the speaker's sex using a crowd-sourced data labeling campaign. The original train, validation and test sets of the dataset were preserved, but split by sex between male and female speakers. 30% of the training, 32% of the validation and 29% of the test data are female speakers. During training, the train set was equally weighted for male and female speakers. Validation and test sets were not equally weighted.

3.3 Metrics

Quantifying embedded KWS accuracy. We evaluate model performance with five accuracy metrics: Cohen's kappa coefficient, precision, recall, weighted F1 score and the Matthews Correlation Coefficient (MCC). The trends we observed are consistent across metrics and hence we present results only for the MCC metric. A higher MCC metric implies better prediction performance.

Quantifying embedded KWS fairness. We define an embedded KWS model as fair for a subgroup (i.e., male/female) if the subgroup's individual accuracy score equals the model's average accuracy score across all subgroups. If a model performs better or worse than average for a subgroup, we consider it to be favouring or prejudiced against that subgroup. Both favouritism and prejudice reduce model fairness. A suitable fairness metric should capture these definitions and penalise favouritism and prejudice equally. Additionally, a fairness metric should be able to score models as being more or less fair, and should consider positive and negative prediction outcomes. Given these requirements, we operationalize fairness as follows. For a subgroup i ($i = 1 \cdots N$), we define its relative performance over the model's average performance by τ_i

$$\tau_i = \frac{MCC_i}{\frac{1}{N}\sum_{j=1}^{N} MCC_j} \tag{1}$$

where MCC_i denotes the MCC score for the i^{th} subgroup.

Then, the fairness of the model to a subgroup i is defined as:

$$fairness_i = ln\,(\tau_i) \tag{2}$$

The $fairness_i$ metric is 0 when a model is fair to subgroup i, negative when it performs worse than average (i.e. when it is prejudiced against a subgroup) and positive when it performs better than average (i.e. when it favours a subgroup). Furthermore, the magnitude of the metric is equal for an accuracy ratio and it's inverse, as $ln(x) = -ln(\frac{1}{x})$. This has intuitive appeal that supports the interpretability of the metric, as the magnitude of the $fairness_i$ metric is equal for subgroups that perform half as good and twice as good as average.

Finally, we obtain the fairness of the model over all subgroups as:

$$fairness_{model} = \sum_{i=1}^{N} |fairness_i| \tag{3}$$

The $fairness_{model}$ metric defined above is valid under the conditions that all subgroups are equally important within the application context and that fairness is independent of the group size. A lower value of $fairness_{model}$ is preferred as it signifies that all subgroups have similar accuracy as the model's average accuracy.

4 ANALYSIS OF PARAMETER IMPORTANCE

In this section, we aim to understand and contrast the impact of the various pre-processing parameters on model accuracy, fairness, and system metrics.

4.1 Findings

We begin our analysis by first studying the impact of the neural architecture on model performance. Thereafter, we study the role of pre-processing parameters separately for each architecture.

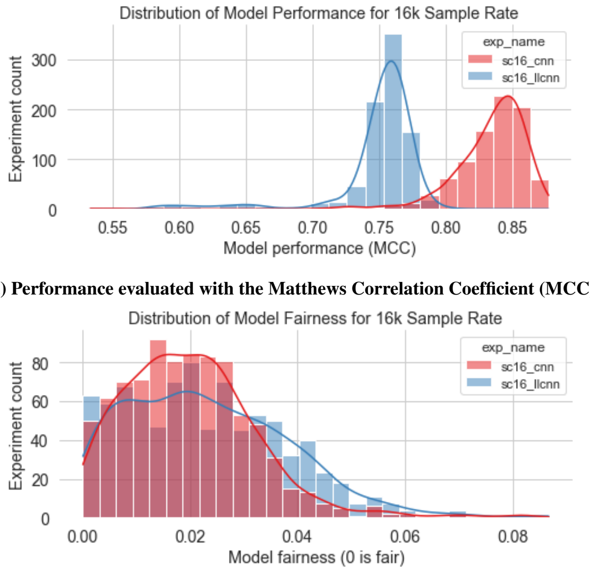
Impact of model architecture. The distributions of MCC and model fairness scores for CNN and llCNN architectures trained with 16k audio input are shown in Figure 2. Unsurprisingly, models trained with the larger CNN architecture perform better than those trained with the optimized llCNN architecture. For model fairness, the distribution of scores is less pronounced. Importantly both architectures have models with scores close to 0, implying that both architectures produce some models that are fair. Using a univariate linear regression test across all parameters, we confirmed that model architecture has the greatest effect on predictive performance and a significant impact on model fairness, with CNN models being better and fairer ($F_{MCC} = 3125.624$, $p < 0.01$ and $F_{Fair} = 22.755$, $p < 0.01$ for $F_{crit}(1, 3454) = 6.642$ at $alpha = 0.01$). Given the dominating effect of the model architecture, we analyse pre-processing parameter importance separately for CNN and llCNN models.

Architecture	16k CNN			16k llCNN		
Parameters	F_{MCC}	p	**better if**	F_{MCC}	p	**better if**
# Mel bins	384.032*	5.2e-71	fewer	44.344*	4.9e-11	more
MFCCs	0.242	6.2e-1	-	101.267*	1.3e-22	not none
feature type	2.041	1.5e-1	-	392.356*	2.9e-72	MFCC
frame length	4.668	3.1e-2	-	8.705*	3.3e-3	not 0.04
frame step	16.065*	6.6e-5	shorter	0.094	7.6e-1	-
window type	3.927	4.8e-2	-	0.726	3.9e-1	-

Table 3: Parameter importance for the *MCC performance* metric for CNN and llCNN architectures trained and evaluated on *audios* sampled at 16KHz.

Architecture	16k CNN			16k llCNN		
Parameters	F_{Fair}	p	**fairer if**	F_{Fair}	p	**fairer if**
# Mel bins	46.449*	1.8e-11	fewer	13.090*	3.1e-4	fewer
MFCCs	25.534*	5.3e-7	-	0.252	6.2e-1	-
feature type	43.179*	8.6e-11	log Mel	12.018*	5.5e-4	MFCC
frame length	20.003*	8.8e-6	longer	3.386	6.6e-2	-
frame step	2.648	1.0e-1	-	2.773	9.6e-2	-
window type	9.199*	2.5e-3	Hamming	0.180	6.7e-1	-

Table 4: Parameter importance for *model fairness* for CNN and llCNN architectures trained and evaluated on *audios* sampled at 16KHz.

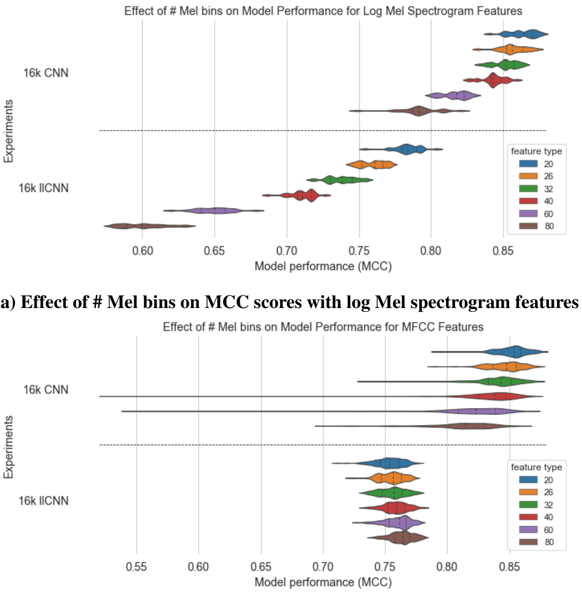


(a) Performance evaluated with the Matthews Correlation Coefficient (MCC)

(b) Model fairness

Figure 2: Distribution of scores for CNN and low latency CNN (llCNN) architectures trained with 16k audio data.

Impact of feature type and count. The F-scores and p-values for parameters of the CNN and llCNN models are shown in Table 3 for the MCC performance metric and in Table 4 for model fairness. Starred values reject the null hypothesis for $F_{crit}(1, 1726) = 6.650$ at $alpha = 0.01$ and their parameters impact the metric at a 1% significance level. *better if* and *fairer if* parameter values are specified for important parameters based on mean values and are indicative of overall trends, not the values of the best parameter configuration.



(a) Effect of # Mel bins on MCC scores with log Mel spectrogram features

(b) Effect of # Mel bins on MCC scores with MFCC features

Figure 3: MCC scores for CNN and llCNN architectures trained with 16k data.

For CNN architectures we find that the number of Mel bins has the greatest impact on model performance, and a significant impact on fairness. Models perform better and are fairer if they have a lower number of Mel bins, and performance degrades rapidly with larger numbers of Mel bins. CNN models are fairer when using log Mel spectrogram features. llCNN architectures on the other hand perform better and fairer when using MFCC features. This finding is emphasised by the absence of MFCC coefficients (i.e. MFCCs parameter value is *none*) strongly impacting model performance. However, provided that MFCC features are used, the actual number of coefficients has no significant impact on model performance, and does not impact fairness. This can be explained by the first cepstral coefficients containing most of the signal information.

The effect of the number of Mel bins for log Mel spectrogram and MFCC feature types is shown in Figure 3. With log Mel spectrogram features, fewer Mel bins provide significantly higher MCC scores (Figure 3a). However, when MFCC features are used in the case of llCNN model, the impact of the number of mel bins is less pronounced (Figure 3b).

Impact of temporal filters and windowing. The model performance of CNN architectures is sensitive to the frame step. Frame steps that are shorter relative to the frame length result in models with higher MCC scores. CNN model fairness is sensitive to frame length and the window type, but not to frame step. Longer frame lengths and the Hamming window result in fairer CNN models. Overall, CNN models are more sensitive to feature type and count, than to other pre-processing parameters. The MCC performance scores of llCNN architectures are lowest when the frame length is 0.04s, the longest frame length that we investigated. Both model fairness and MCC performance are otherwise not significantly impacted by frame length, frame step or window type. This presents an opportunity to optimise these parameters for system performance in embedded applications.

4.2 Implications for embedded ML design

We summarise the impact of pre-processing parameters on system performance, model performance and model fairness in Table 5. Feature type significantly impacts model accuracy and fairness. Selecting log Mel spectrograms as features for CNN architectures, and MFCCs as features for llCNN architectures results in better accuracy and fairness. As feature type by itself does not have any effect on system efficiency, model developers can choose the features best suited for the neural architecture.

	System impact	16k CNN		16k llCNN	
Parameters		**better if**	**fairer if**	**better if**	**fairer if**
feature type	depends on size	-	log Mel	MFCC	MFCC
# Mel bins	fewer is better	fewer	fewer	N/A	N/A
MFCCs	fewer is better	-	none	not none	-
frame length	longer is better	-	longer	not 0.04	-
frame step	longer is better	shorter	-	-	-
window type	none	-	Hamming	-	-

Table 5: Summary of system, predictive performance and fairness impact of pre-processing parameters on CNN and llCNN models trained with 16k data.

Focusing on CNN architectures with log Mel spectrogram features, our findings show that models with *fewer Mel bins* have better predictive performance, are fairer, and are likely to have better system performance (energy consumption and latency). Frame step has an opposing effect on CNN architectures, with shorter frame steps resulting in models with better predictive performance, while longer frame steps are better for system performance. This finding reveals an important design choice for model developers: depending on the application and system requirements, they can modify the frame step to trade-off between predictive performance and system cost.

For llCNN architectures with MFCC features, the number of MFCC coefficients does not have a significant impact on predictive performance and fairness, and can thus be optimised for system performance by selecting as few coefficients as possible. Similarly, frame step and frame length (except 40ms) also do not have a significant impact on predictive performance and fairness, and can thus be optimized for system efficiency by selecting longer frame length and step sizes.

In Table 6 we show the parameter values, prediction accuracy, model fairness scores, and inference latency for models selected based on *highest* MCC score, best model fairness score and optimal system heuristics. For optimal system heuristics we chose parameter values that reduce the system impact, provided that they do not oppose MCC performance or model fairness. The inference latency was computed by executing each model on a Raspberry Pi 4. The trade-offs between the best models for each metric are clear and allow a model developer to tune the pre-processing pipeline to application requirements. For the CNN model, we can reduce inference latency by 7.2% at the cost of 3.7% reduction in absolute prediction accuracy. Similarly, by tuning the pre-processing parameters for the llCNN model, we can improve model fairness by 81% with a 4.2% drop in prediction accuracy.

architecture	feature type	frame length	frame step	#Mel bins	MFCCs	inference latency	MCC score	model fairness
CNN (MCC)	log Mel	20s	0.4	20	None	180ms	0.877	1.2e-2
CNN (fairness)	log Mel	30s	0.6	26	None	176ms	0.849	1.8e-4
CNN (system)	log Mel	40s	0.6	20	None	167ms	0.840	3.5e-3
llCNN (MCC)	MFCC	20s	0.5	20	14	173ms	0.804	6.6e-4
llCNN (fairness)	MFCC	20s	0.6	20	14	169ms	0.762	1.2e-4
llCNN (system)	MFCC	30s	0.6	40	10	160ms	0.773	2.3e-2

Table 6: MCC, model fairness scores, and inference latency for three 16k CNN and llCNN models. For model fairness, lower scores are better.

5 PARAMETER IMPORTANCE UNDER VARYING DEPLOYMENT CONDITIONS

In embedded applications the audio data that devices can collect is constrained by hardware capabilities such as microphone quality, and hardware constraints, such as power consumption. Collecting data at a lower sample rate consumes less power than collecting it at a higher sample rate, which is beneficial in applications where battery life is a concern. We thus consider parameter importance in two scenarios that are likely to arise when deploying ML models to heterogeneous devices. In the first scenario we consider the effect of lowering the training and evaluation sample rate from 16kHz to 8kHz, a realistic possibility in many embedded applications. For this experiment we down-sampled the Speech Commands dataset to 8kHz, both during training and inference. In the second scenario we consider what happens when a black-box model is deployed in an application that collects data at a different sample rate to what the model was trained at, that is, a 16kHz model receives an 8kHz input and vice versa.

5.1 Impact of a lower sample rate

Table 7 shows a summary of our findings on parameter importance for 8kHz audios. In contrast to the results for 16kHz audios, we observe that the choice of features used for 8kHz audios offer a clear tradeoff between model accuracy and fairness. While MFCC features are better for prediction accuracy, log mel spectrograms result in fairer models for both CNN and llCNN architectures. As such, depending on the application requirements (e.g., whether to prioritize accuracy or fairness), a developer can choose the optimal feature. We also observe that using fewer mel bins can satisfy the goal of system efficiency and fairness for both architectures. Similarly, by using fewer MFCC coefficients, we can achieve better system efficiency and prediction accuracy. Table 7 also illustrates how the temporal parameters such as frame length and frame step could be tuned to balance system efficiency and model performance.

	System impact	8k CNN		8k llCNN	
Parameters		**better if**	**fairer if**	**better if**	**fairer if**
feature type	depends on size	MFCC	log Mel	MFCC	log Mel
# Mel bins	fewer is better	N/A	fewer	N/A	-
MFCCs	fewer is better	not none	N/A	not none	N/A
frame length	longer is better	-	longer	-	shorter
frame step	longer is better	shorter	-	-	-
window type	none	-	-	-	-

Table 7: Summary of system, predictive performance and fairness impact of pre-processing parameters on CNN and llCNN models trained with 8k data.

5.2 Sample rate divergence during deployment

Table 8 and Figure 4 illustrate our findings on parameter importance when the model encounters sample rate divergence during deployment. The results reveal some very surprising findings. Firstly, we observe that in the presence of sample rate divergence, MFCCs are no longer the right features to choose for llCNN models (in contrast to our findings in Table 5). In fact, the performance degradation by using MFCC features in llCNN models is so severe that the best MCC scores drop from 0.804 to 0.25. Instead, a model developer should opt for mel spectrogram features if they expect the possibility of sample rate divergence at deployment, because even though mel spectrograms have lower accuracy than MFCCs for the llCNN models, they are more robust to sample rate divergence.

	System impact	16k input | 8k llCNN		8k input | 16k llCNN	
Parameters		**better if**	**fairer if**	**better if**	**fairer if**
feature type	depends on size	log Mel	log Mel	log Mel	log Mel
# Mel bins	fewer is better	-	more	-	-
MFCCs	fewer is better	none	none	none	none
frame length	longer is better	-	-	-	-
frame step	longer is better	-	-	-	-
window type	none	-	-	-	-

Table 8: Summary of system, predictive performance and fairness impact of pre-processing parameters on llCNN models when the input sample rate is different to the training sample rate: llCNN models trained with 16k data were evaluated on 8k input, and vice versa.

6 DISCUSSION AND CONCLUSIONS

The aim of this paper was to characterize the impact that configuration parameters can have on different performance metrics of a ML model, such as prediction accuracy, fairness or system efficiency. Based on our findings, following are the takeaways and implications for future work on embedded ML.

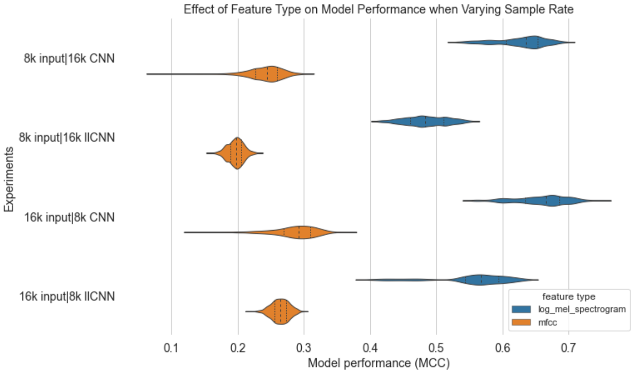


Figure 4: Effect of feature type on the distribution of MCC scores of CNN and low latency CNN architectures trained and evaluated at different sample rates.

Pre-processing parameters matter. Our results show that data pre-processing parameters have a statistically significant effect on model accuracy, fairness and inference efficiency in audio KWS tasks. We advocate that tuning these parameters should be considered as an important part of the embedded ML training pipeline, something which we have not seen in prior works (see Table 2). Depending on the objectives and resource availability of an embedded ML application, model developers can modify these parameters to balance model accuracy, fairness and inference efficiency.

Choosing the right parameter can help in unknown deployment conditions. Embedded ML models need to deal with the heterogeneity in sensor hardware at inference time. For instance, a KWS model trained on 16kHz audios may – at inference time – encounter microphones which cannot sample audio at 16kHz. Our findings show that choosing the right pre-processing parameters can alleviate the negative impact of such unseen deployment conditions on model performance. For example, we found that the use of mel spectrograms as input features instead of MFCCss can prevent severe degradation in model performance when the samples rates diverge between training and inference stages.

Towards data-centric ML. Our work directly relates to the emerging trend in the ML community on approaching ML from a data-centric perspective as opposed to only model-centric perspective [1]. Specifically, in the case of embedded ML, a number of works have taken a model-centric approach by applying model pruning [25], weight quantization [7] to ensure that ML models can satisfy the resource requirements of embedded devices. Our findings highlight that there is a clear merit in taking a data-centric view to this problem and by configuring the parameters of the data pipeline (e.g., frame step, number of mel bins), we can balance various performance metrics of an embedded ML system. As a future work, we plan to explore how both data- and model-centric embedded ML approaches can work together.

At the workshop. We hope to discuss with other workshop attendees how this work can be extended to other sensing modalities (e.g., vision, accelerometer data). Moreover, we would like to get feedback on how the data-centric approach to embedded ML presented in this paper can be combined with model-centric optimization approaches from the literature, such as layer pruning and weight quantization.

ACKNOWLEDGMENTS

We thank Ekdeep Singh Lubana and Alessandro Montanari for their useful suggestions on the work. This work is partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101021808.

REFERENCES

[1] 2021. MLOps: From Model-centric to Data-centric AI. https://www.deeplearning.ai/wp-content/uploads/2021/06/MLOps-From-Model-centric-to-Data-centric-AI.pdf. Accessed: October 27, 2021.

[2] 2021. Regulation of the European Parliament and Council. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52021PC0206&from=EN. Accessed: October 27, 2021.

[3] 2021. TensorFlow Audio Recognition. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/speech_commands/models.py. Accessed: October 27, 2021.

[4] Raziel Alvarez and Hyun Jin Park. 2019. End-to-end streaming keyword spotting. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 6336–6340.

[5] Anthony Berthelier, Thierry Chateau, Stefan Duffner, Christophe Garcia, and Christophe Blanc. 2021. Deep model compression and architecture optimization for embedded systems: A survey. *Journal of Signal Processing Systems* 93, 8 (2021), 863–878.

[6] Guoguo Chen, Carolina Parada, and Georg Heigold. 2014. Small-Footprint Keyword Spotting Using Deep Neural Networks. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. IEEE.

[7] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[8] John H.L. Hansen and Taufiq Hasan. 2015. Speaker recognition by machines and humans: A tutorial review. *IEEE Signal Processing Magazine* 32, 6 (2015), 74–99. https://doi.org/10.1109/MSP.2015.2462851

[9] Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw. 2017. Streaming Small-Footprint Keyword Spotting Using Sequence-to-Sequence Models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

[10] Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, DIng Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo Yiin Chang, Kanishka Rao, and Alexander Gruenstein. 2019. Streaming End-to-end Speech Recognition for Mobile Devices. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. https://doi.org/10.1109/ICASSP.2019.8682336 arXiv:1811.06621

[11] Takuya Higuchi, Mohammad Ghasemzadeh, Kisun You, and Chandra Dhir. 2020. Stacked 1D convolutional networks for end-to-end small footprint voice trigger detection. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. https://doi.org/10.21437/Interspeech.2020-2763 arXiv:2008.03405

[12] Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R. Rickford, Dan Jurafsky, and Sharad Goel. 2020. Racial disparities in automated speech recognition. *PNAS* 117, 14 (2020), 7684–7689. https://doi.org/10.1073/pnas.1915768117/-/DCSupplemental.y

[13] Akhil Mathur, Anton Isopoussu, Fahim Kawsar, Nadia Berthouze, and Nicholas D Lane. 2019. Mic2mic: using cycle-consistent generative adversarial networks to overcome microphone variability in speech systems. In *Proceedings of the 18th international conference on information processing in sensor networks*. 169–180.

[14] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A survey on bias and fairness in machine learning. *arXiv* (2019).

[15] Luca Oneto and Silvia Chiappa. 2020. Fairness in machine learning. *Recent Trends in Learning From Data* (2020), 155–196.

[16] Alvin Rajkomar, Michaela Hardt, Michael D Howell, Greg Corrado, and Marshall H Chin. 2018. Ensuring fairness in machine learning to advance health equity. *Annals of internal medicine* 169, 12 (2018), 866–872.

[17] Tara Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. (2015).

[18] Rachael Tatman and Conner Kasten. 2017. Effects of talker dialect, gender & race on accuracy of bing speech and youtube automatic captions. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 2017-Augus (2017), 934–938. https://doi.org/10.21437/Interspeech.2017-1746

[19] Wiebke Toussaint and Aaron Yi Ding. 2021. SVEva Fair: A Framework for Evaluating Fairness in Speaker Verification. *arXiv* (2021).

[20] George Tucker, Minhua Wu, Ming Sun, Sankaran Panchapagesan, Gengshen Fu, and Shiv Vitaladevuni. 2016. Model compression applied to small-footprint keyword spotting. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 08-12-Sept (2016), 1878–1882. https://doi.org/10.21437/Interspeech.2016-1393

[21] Sandra Wachter, Brent Mittelstadt, and Chris Russell. [n. d.]. Bias Preservation in Machine Learning : The Legality of Fairness Metrics Under EU Non-Discrimination Law. *West Virginia Law Review, Forthcoming* ([n. d.]), 1–51. https://ssrn.com/abstract=3792772

[22] Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209* (2018).

[23] Pete Warden and Daniel Situnayake. 2019. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media.

[24] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. 2017. Hello edge: Keyword spotting on microcontrollers. *arXiv* (2017), 1–14. arXiv:1711.07128

[25] Zhongpeng Zhang. 2021. Model Pruning Techniques for Boosting the Inference Efficiency on Embedded Systems. In *2021 2nd International Conference on Computing and Data Science (CDS)*. IEEE, 119–124.