COUGH DETECTION ON THE EDGE: HEAR MODEL COMPRESSION USING KNOWLEDGE DISTILLATION

Alex Phang Kian Jing
PGDAT Student
Unitec Institute of Technology
Auckland, New Zealand
phanga01@myunitec.ac.nz

ABSTRACT

This project explores the compression and edge deployment of Google's HeAR acoustic model for real-time cough detection on resource-constrained devices. A black-box knowledge distillation framework was used to transfer the HeAR model's representational capability into a lightweight MobileNetV3Small student network. A logistic regression classifier trained on HeAR embeddings was reimplemented as a one-layer Keras model to enable TensorFlow Lite conversion, forming a complete on-device pipeline with log-mel preprocessing and the distilled student model. Evaluation results showed that the compressed models achieved performance comparable to the baseline HeAR–logistic regression combination, while the final deployment model maintained strong accuracy and sensitivity despite operating on short audio segments. Although hardware deployment on Android emulators and Raspberry Pi encountered compatibility limitations, the findings demonstrate that large acoustic models can be effectively reduced for privacy-preserving, real-time health monitoring on edge devices.

Keywords: Health Acoustic Representations (HeAR), Black-Box Knowledge Distillation, MobileNetV3Small

1 Introduction

The recent rapid development of machine learning and embedded computing has enabled the possibility of developing applications for real-time health monitoring. One of the most widely researched acoustic biomarkers is cough detection, which is relevant to respiratory infections, chronic lung conditions, and disease surveillance in the population [1, 2, 3]. Traditional diagnostic solutions are based on clinical consultations or expensive specialized hardware, which are not accessible and convenient for everyone [4]. As the improvement of edge devices becomes more powerful, machine learning systems are gaining interest in deploying lightweight models on resource-constrained devices, where the models are capable of detecting and classifying health-related acoustic events without cloud connectivity. This completely shifts the processing from the cloud to the edge [5]. As a result, all the sensitive health information can be processed locally rather than being transmitted externally.

The Google HeAR (Health Acoustic Representations) research paper [6] demonstrates that it provides a strong foundation for modeling health-related acoustic events. HeAR is pretrained on more than 313 million two-second audio recordings and provides a powerful framework for detecting cough-related acoustics. Its architecture utilizes a masked autoencoder (MAE) and a ViT-L (Vision Transformer Large) encoder to generalize across various types of health-related acoustic recordings. The benchmark of cough datasets, such as CoughVid, CIDRZ, and Coswara, is used to evaluate the performance of HeAR. The HeAR model performed the best in the task of cough detection with the assistance of downstream classifiers. Nevertheless, the original HeAR model is computationally heavy due to its ViT-L encoder, which makes it unreasonable to run on resource-constrained devices. It poses technical challenges in maintaining the strength of detecting health acoustic tasks while making the HeAR model small enough for practical real-time deployment.

To overcome these limitations, the project applied knowledge distillation frameworks to reduce the size of the HeAR model and adapted it to resource-constrained device deployments [7]. However, the internal layers and logits of the HeAR model are not disclosed, so traditional distillation strategies cannot be used in the project [8]. Instead, the project

applied black-box knowledge distillation strategies [9], where the MobileNetV3Small served as the lightweight student model and directly learn knowledge from the HeAR embedding output. Meanwhile, a logistic regression classifier was trained on the HeAR model to classify the HeAR embeddings as cough or non-cough [10]. To ensure the compatibility of TensorFlow Lite, the logistic regression was rebuilt as an equivalent one-layer Keras network [11]. In addition, the log-mel spectrogram function was implemented to fully replicate the input preprocessing used in HeAR. This enabled the MobileNetV3Small student model to operate the same way as its teacher model. The final deployment system integrated the log-mel spectrogram module, the distilled MobileNetV3Small student model, and the one-layer Keras logistic regression model. Then, the final deployment system was converted into TensorFlow Lite to execute on resource-constrained devices.

This report presents a complete end-to-end workflow that includes dataset preprocessing, embedding feature extraction, black-box knowledge distillation, logistic regression classifier deployment, model conversion, and deployment testing. Each stage model is evaluated by using standard binary classification metrics and a confusion matrix to compare the performance with the baseline HeAR model [12, 13]. The results demonstrate that the final deployment model achieved comparable performance and strong reliability to the baseline HeAR model. These findings highlight the feasibility of compressing large acoustic models for real-time cough detection on resource-constrained devices.

2 Literature Review

2.1 HeAR (Health Acoustic Representation) Model

The Google research team proposed the HeAR model to analyze non-semantic health acoustic signals, such as coughs, breathing, and spirometry events [6]. The HeAR model is a self-supervised model that was trained on a total of 313.3 million two-second clips from the YouTube Non-Semantic (YT-NS) dataset. The extension dataset comprises various health-related acoustic events from a wide variety of participants. This large-scale pretraining enables downstream models to generalize effectively to unseen recordings. The HeAR model has demonstrated strong performance across multiple health acoustic events, which indicates that HeAR is a robust foundation model.

The architecture of the HeAR model is a masked autoencoder (MAE) framework, where it masks 75% of the spectrogram patches [14]. Then, it uses a ViT-L (Vision Transformer Large) encoder to reconstruct masked spectrogram patches based on the remaining visible spectrogram patches [15]. This design enables the encoder to learn the characteristics of health acoustic events and generalize well. The research paper demonstrated that the HeAR model achieved the highest score in cough events compared to TRILL, FRILL, and BigSSL models. The cough datasets, such as CIDRZ, CoughVid [16], and Coswara [17], are used to evaluate the performance on cough-related events. These datasets consist of metadata like sex, age, smoking status, and COVID-19 status that play a key role in detecting cough under different real-world conditions. These results highlight the potential of HeAR for scalable, device-diagnostic health monitoring applications.

2.2 HeAR Embeddings with Logistic Regression for Pediatric Asthma Detection

The recent study [10] has used the HeAR (Health Acoustic Representations) model as the foundation for pediatric asthma detection. The respiratory sound samples of the SPRSound dataset were converted into HeAR embeddings and then subjected to the classical machine learning classifiers, such as the Support Vector Machine (SVM), Random Forest, Gradient Boosting, and Logistic Regression. The encoding of each embedding of temporal-spectral features of wheeze, stridor, crackle, or normal breathing sounds allowed the machine learning classifier to train the threshold lines between normal and asthma-predictive events. According to the research [10], the logistic regression classifier achieved the highest overall accuracy and minimal false-negative rates. It is an important feature in the clinic's screening activities, and the omission of positive cases may have severe medical repercussions.

The logistic regression model was also very interpretable and computationally efficient as compared to more bulky neural structures. Although the multilayer perceptron (MLP) demonstrated a marginally better accuracy and AUC, the logistic regression demonstrated a better balance between performance, simplicity, and resource efficiency. Its confusion matrix had the least number of false negatives as compared to all models, which indicates that it is sensitive to wheezing and abnormal airway sounds. This renders it very useful in practice in the diagnostic application of pediatric recordings, which can be brief and acoustically faint. Moreover, logistic regression is not heavy, and it is less vulnerable to overfitting, which makes it applicable to edge devices and low-resource healthcare settings. These results support

the possibility of using classical classifiers with fundamental audio models, including HeAR, to develop scalable, interpretable, and energy-efficient respiratory screening systems.

2.3 Black-Box Knowledge Distillation

Black-box knowledge distillation [9], also called zero-shot knowledge distillation [18], is a generalization of the concept of knowledge distillation to scenarios where the teacher model is inaccessible, and only its predictions of outputs are available. In this case, the student model does not learn from the intermediate feature representations of the teacher model but is directly based on the teacher's output. The research paper [19] introduced the Multi-Level Logit Distillation (MLLD) framework, which constructs additional supervision signals for the student model in a black-box environment. The MLLD framework first executes prediction augmentation that involves scaling the outputs with temperature to make the predictions softer and reveal the latent interactions between classes [20, 21]. With higher temperatures, the softened distributions will give out dark knowledge, including small overlaps or dependencies between categories commonly obscured by certain confident predictions. By doing so, the student model will be able to learn richer decision boundaries using the predictions of the teacher and generalize better across classification tasks.

Following prediction augmentation, the student model is then trained through multi-level alignment mechanisms, including instance-level, batch-level, and class-level alignments [19]. At the instance-level alignment, the student model is forced to mimic the teacher's predictions, so that it can minimize divergence across prediction augmentation. This encourages the student to learn the prediction behavior even when the teacher's predictions are softened or adjusted. The batch-level alignment aligns the pairwise similarity structure among samples by matching the similarity matrices computed from teacher and student outputs. On the other hand, the class-level alignment captures how different classes relate to each other by analyzing the covariance between their predicted probabilities. It teaches the student to learn the relationships between classes from the teacher's softened predictions. These alignments compute their own distinct loss components. Combined, the loss components compensate for the absence of logits or internal representations.

Nevertheless, not all these techniques are applied to the HeAR model, as it cannot be used in embeddings. For example, classification-oriented distillation techniques cannot be applied directly because the HeAR model does not provide probability distributions of classes but high-dimensional embeddings. As a result, the prediction augmentation and class-level alignment methods cannot be used. This is because they are unable to discover correlation structures when there are no logits and common categories. Instead, distillation strategies must focus on the representation space. The gap between teacher and student representations can be minimized with the help of such methods as instance-level alignment. In addition, the batch-level alignment preserves pairwise similarity relationships among samples. These approaches maintain the quality of acoustic representation learned by HeAR and allow for the effective transfer of knowledge to a smaller model.

2.4 Knowledge Distillation for Speech Detection Using MobileNetV3-Small

The latest trends in bioacoustic monitoring have focused on the need for computationally effective models to perform real-time voice detection in environmental audio [22]. The EcoVAD deep learning model demonstrated high accuracy in detecting human speech in a complex sound environment [23, 24]. Nevertheless, there is a limitation to the deployment of edge devices on the EcoVAD model due to its high computational resource requirements. To overcome this limitation, the research paper [24] proposed knowledge distillation techniques to reduce the large EcoVAD teacher model to a lightweight MobileNetV3Small student model. The MobileNetV3-Small was chosen due to its combination of depthwise separable convolution DWC, low number of parameters, and compatibility with low-power processors [25]. The student network was trained to learn from the output of the teacher, while maintaining a smaller model size, FLOPs, and shorter inference time by more than 70 percent.

The study [24] demonstrated that the MobileNetV3Small student model achieved almost the same results as the large EcoVAD teacher model on edge devices, such as the Raspberry Pi 4 and AudioMoth devices. The F1-score value of the student model is over 0.96 on benchmark speech-detection datasets. Relational knowledge distillation showed the best trade-off between compression and accuracy among the tested techniques. The results proved that the student architecture preserves the characteristics of acoustic recognition even after parameter reduction. The study also emphasized that transferring intermediate-layer relations and output logits can highly improve generalization and robustness in dynamic environmental recordings, instead of relying solely on output logits.

The MobileNetV3Small distillation architecture provides useful suggestions on the project, where efficiency and scalability are also key considerations. The HeAR model generates complex acoustic representations, which are similar

to EcoVAD. Thus, it requires high computational resources, which are not applicable to small devices. Therefore, the project applied knowledge distillation techniques to smaller networks by adapting MobileNetV3Small as the student model. It provides a balance between representational capacity and computational efficiency, which makes it suitable for real-time respiratory sound monitoring on embedded hardware. This approach enables the potential for health acoustic events and operates reliably without compromising accuracy or interpretability.

2.5 One-Layer Keras Implementation of Logistic Regression

Logistic regression can be designed as a single-layer neural network, where a solitary neuron with a sigmoid activation function directly models the probability of a binary outcome. This has been well established in prior work. The research paper [26] demonstrates that the implementation of the logistic regression in TensorFlow/Keras utilized a 1D layer with a node and a sigmoid function. The structure of the model is mathematically equivalent to a standard and traditional logistic regression model. The book [27] further explains the methods and concepts of generalized linear and logistic regression models. It stated that the logistic regression model can be interpreted as a degenerate neural network that does not have any hidden layers, where the regression coefficients correspond to the weights of the dense layer, and the regression intercept represents the bias term. This finding makes logistic regression a special case of neural networks, which offers a theoretical foundation for how it can be reconstructed within the Keras framework. This concept is further explored in the blog post [28], which illustrates how Keras utilized a one-layer Dense model to recreate the linear and logistic regression models. In the simulation, the kernel parameters of the Dense layer are equal to the coefficients vector, and the output bias term replicates the intercept learned by the original regression model. If all these weights are set correctly, the Keras model will achieve the same prediction accuracy as the scikit-learn logistic regression. The justification aligns with the project scope that remains a lightweight model and is compatible with the TensorFlow deployment pipeline. The key reason for using the Keras model is that scikit-learn classifiers cannot be converted into TensorFlow Lite, which makes them incompatible in the final deployment pipeline.

2.6 One-Layer Keras Implementation of Logistic Regression

Binary classification evaluation metrics can be calculated based on the count in the confusion matrix, which contains true positives, false positives, true negatives, and false negatives. With these values, the accuracy metric can provide an overall measure of correctness in the evaluation process. However, the research paper [29] emphasizes that the standalone accuracy metric is unreliable in the imbalanced datasets because the score is dominated by the majority class only. Therefore, precision and recall should be included, as they focus on the model's behavior with respect to the positive class. Precision measures the correctness of the predicted positive class, which involves true positives and false positives. In contrast, recall measures how well the model can predict the actual positive class, which includes true positives and false negatives. These metrics provide more information about the behavior of the actual and predicted positive class, even if they are in the minority class. The paper [29] also demonstrates that the F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall into a single value to demonstrate the performance of the model, especially for imbalanced datasets.

Besides these metrics, the area under the ROC curve (ROC-AUC) can be adopted in the project because it is a stable and reliable metric for evaluating binary classification results. The paper [30] discusses how the area under the ROC curve is one of the most stable evaluation metrics for both imbalanced and balanced datasets. ROC-AUC examines the relationship between the true positive rate and the false positive rate without any influence from the threshold and class distribution. Consequently, ROC-AUC exhibits a small variance across various dataset distributions, noise, and model thresholds. To measure how well the models perform across the entire project, several key metrics are used, including accuracy, precision, recall, F1-score, and ROC-AUC. Furthermore, the confusion matrix is utilized to provide a view of the performance in every classification.

3 System Design and Implementation

3.1 Methodology

The project was designed to reduce the size of the HeAR model and deploy it on edge devices. The project implementation consisted of ten steps, which are illustrated in *Figure 1*. These steps covered the entire project pipeline, including

loading the model and dataset, preprocessing the CoughVid dataset, pre-trained HeAR model process, data splitting, training and evaluating the logistic regression classifier, black-box knowledge distillation, a 1D layer Keras logistic regression, model testing, combining the pipeline for deployment, and deployment testing.

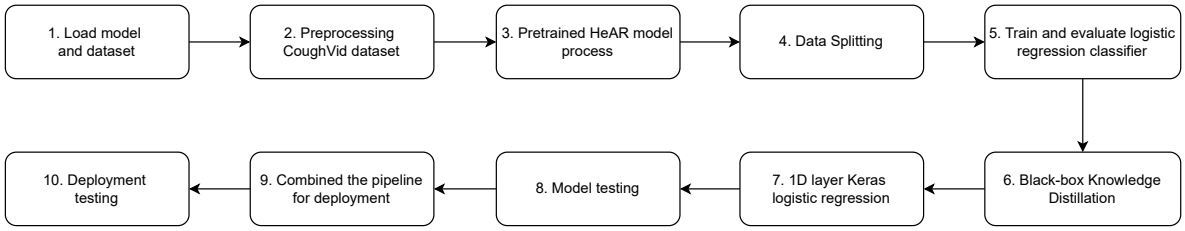

Figure 1: HeAR Model Optimization and Deployment Process Flow

The initial step began with initializing the necessary resources, such as loading the HeAR model and the CoughVid dataset, as demonstrated in *Figure 2*. The pre-trained HeAR model is available on the Hugging Face model hub, which enables researchers to facilitate reuse. This model has been trained on approximately 313.3 million two-second clips, which provides a strong and robust fundamental model for health acoustic events. In order to access and load the HeAR model into memory, an authentication token from Hugging Face is required. This step ensures secure access and mitigates potential unauthorized usage or data breaches.

The CoughVid dataset was selected because it served as one of the primary evaluation datasets of the HeAR model. The size of the CoughVid zipped file is 951.4 MB, and it contains more than 20,000 recordings, including `OGG` and `WEBM` files. The recordings include a wide variety of ages, genders, and geographic locations, which provide sufficient data to train and evaluate the results. Both the model and the dataset were successfully loaded and stored locally for subsequent preprocessing.

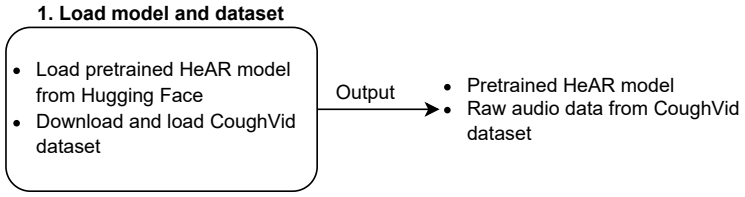

Figure 2: Load Data and Model Process

Due to the architecture of the HeAR model, it only allows input of a two-second audio clip with a sample rate of 16kHz. Therefore, the following step was to preprocess the CoughVid dataset to meet those requirements, as shown in *Figure 3*. Each raw audio was resampled to 16kHz and segmented into two-second frames with a 50% overlap. For instance, a five-second audio clip is divided into four segments, including 0-2 seconds, 1-3 seconds, 2-4 seconds, and 3-5 seconds. The overlapping segmentation ensures each cough sound is fully captured within at least one segment, preventing misses at the edges of fixed time intervals and reducing potential data loss.

For the cough label, the CoughVid dataset provides a `JSON` file for each audio clip that consists of cough probability. There are approximately 2,000 recordings that contain additional information on diagnosing medical abnormalities, but for simplicity, this project did not utilize this information. This project only used cough probability data since it focuses on detecting cough events. If the cough probability value exceeded 0.5, it was labeled as 1, otherwise labeled as 0. Each cough label was saved as a NumPy file inside the `Audio_Label` folder, and the filename corresponds to the audio file's basename.

After the previous two steps were completed, the next step was to process the pre-trained HeAR model, as demonstrated in *Figure 4*. Each two-second segmented audio from step 2 was passed through the pre-trained HeAR model and produced a 512-dimensional embedding with float32 precision. For example, a five-second audio clip divided into four segments yields four corresponding 512-dimensional embeddings. These embeddings were stored as a NumPy file inside the `Audio_Segment` directory, whose filename follows the audio file's basename.

However, since each audio recording has a different duration, the number of resulting segments varies. The classifier model for training and evaluation in step 5 requires a fixed-length input, where the embeddings for each audio clip

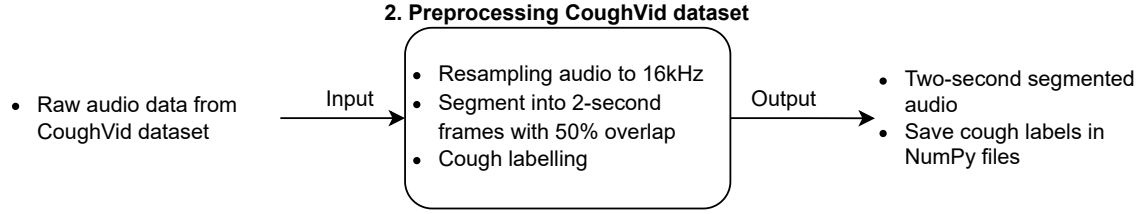

Figure 3: Preprocessing CoughVid Dataset Process

are aggregated using mean and maximum pooling operations. The reason for using mean pooling is to capture the average values of the embeddings across time. On the other hand, the maximum pooling emphasizes the most prominent acoustic characteristics among the embeddings. With these two pooling operation embeddings, it will form a constant size of 1024-dimensional embedding for each audio clip. These standardized embeddings were saved as a NumPy file in the `Audio_MeanMax` folder.

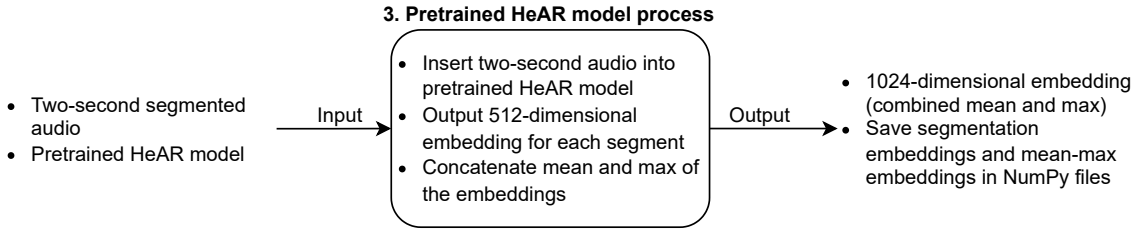

Figure 4: Pretrained HeAR Model process

As shown in *Figure 5*, the data splitting process was designed to provide sufficient data for other steps to perform their tasks. All the NumPy files that were saved in the `Audio_Label`, `Audio_Segment`, and `Audio_MeanMax` folders were first loaded into memory. The primary purpose of saving information into NumPy files is to reduce memory consumption, prevent system overload, and prevent values from changing or being lost during the pipeline. A huge number of high-dimensional embeddings would exceed the hardware memory limitation.

To balance data utilization across tasks, the data was divided into four subsets, including 60% for knowledge distillation, 25% for classifier training, 10% for classifier evaluation, and 5% for deployment. The main reason for allocating a large portion of data to knowledge distillation is because a scratch student model requires a substantial amount of data to train its network. Each subset was saved in its corresponding folder that includes labels, segments, and mean-max directories. This ensures consistency and traceability throughout the workflow.

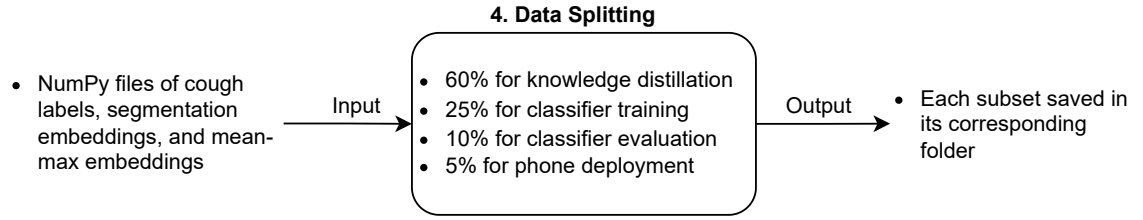

Figure 5: Data Splitting Process

Figure 6 illustrates the workflow of the logistic regression classifier used for training and evaluation. From the directories of classifier training and evaluation, the mean-max embeddings and the cough label data are loaded into the system, which acts as the inputs for the classifier.

In this project, the logistic regression model is selected as the classifier. The Scikit-learn library in Python provides a logistic regression model with default hyperparameters that fit the project. During the training process, the model

utilized the classifier training dataset to learn the coefficients and intercepts. It separated the embeddings of cough and non-cough.

Once trained, the logistic regression model used the classifier evaluation dataset to predict the label of the cough audio clips. To evaluate the performance of the logistic regression classifier, key metrics, including accuracy, precision, recall, F1 Score, and ROC AUC, are measured. Furthermore, a confusion matrix is generated to visualize the distribution of cough and non-cough predictions.

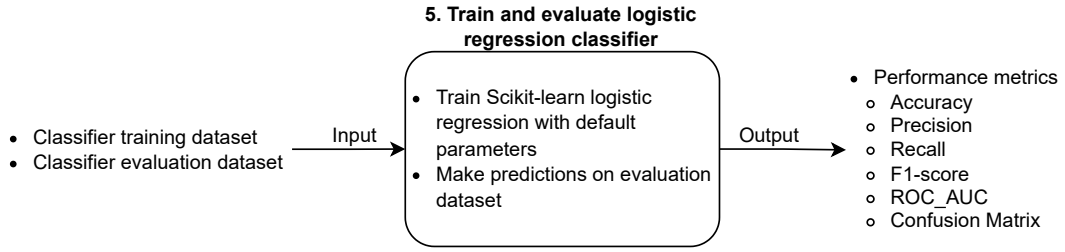

Figure 6: Logistic Regression Classifier Process

Figure 7 presents the process of black-box knowledge distillation, which enables a smaller student model to learn from the pre-trained HeAR model. This approach is used because the internal layers of the HeAR model from the Hugging Face model hub are not publicly available. In a black-box setup, the student model learns from the output of the teacher model without knowing its internal structure or the true labels of the training data.

In this project, the HeAR model served as the teacher, and MobileNetV3Small was used as the student model due to its compact size and suitability for edge deployment. The knowledge distillation dataset consists of the output embeddings generated by the HeAR model. These embeddings represent the features extracted from the CoughVid audio samples. Since MobileNetV3Small cannot process raw waveform inputs, an additional function was used to convert each audio clip into a log-mel spectrogram before sending it to the student model.

The distillation process ran for 10 epochs, during which the student model learned to reproduce the output features of the HeAR model using a loss function to minimize the difference between them. The distillation relied on instance-level alignment to match as closely as possible to the teacher's embeddings, and batch-level alignment to preserve the pairwise similarity structure across 50 samples within each batch. After training, the student model carries the main characteristics of the HeAR model but in a smaller and more efficient form, making it suitable for use on mobile or embedded devices.

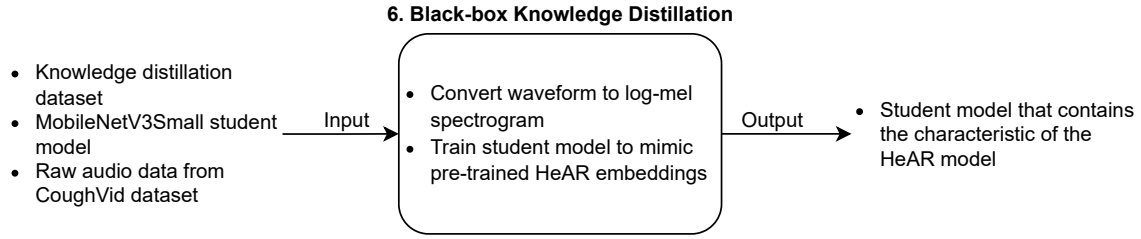

Figure 7: Black-box Knowledge Distillation Process

The Scikit-learn logistic regression model cannot be deployed on mobile devices because its implementation is not compatible with mobile frameworks and requires a Python runtime. To overcome this limitation, a simple 1D layer Keras model is created to replace the Scikit-learn logistic regression, as illustrated in *Figure 8*. The Keras model was initialized as an empty model, with its input layer defined as 1024 dimensions to match the mean–max embeddings generated earlier.

The internal layer of this 1D Keras model is then configured to replicate the coefficient and intercept values learned from the Scikit-learn logistic regression. This ensures that the Keras version performs identically to the original classifier while being lightweight and deployable. The model produced a single probability value representing the prediction of a cough event for each input embedding.

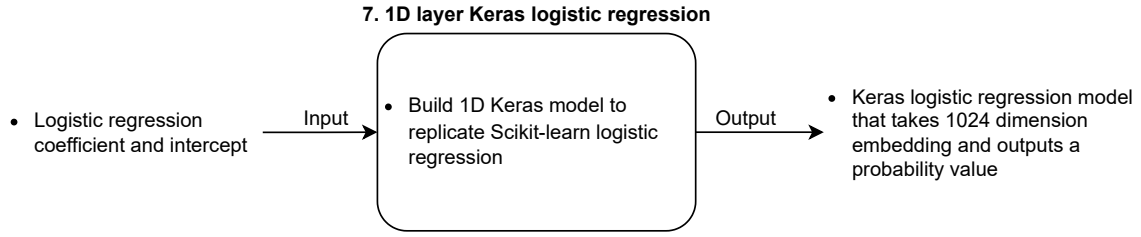

Figure 8: 1D Layer Keras Logistic Regression Process

Figure 9 illustrates the testing process for both the MobileNetV3Small student model and the Keras logistic regression model. The main goal of this step is to evaluate the models individually before combining them into a single TFLite deployment model. The classifier evaluation dataset is used to measure performance and confirm that both models function correctly prior to integration.

The testing process began by evaluating the MobileNetV3Small model together with the Scikit-learn logistic regression classifier to verify how well the student model replicates the behavior of the HeAR model. The same performance metrics from step 5 are used, including accuracy, precision, recall, F1-score, ROC-AUC, and the confusion matrix. If the results show a significant drop in performance compared to the earlier evaluation, the black-box knowledge distillation parameters from step 6 are adjusted, and the training process is repeated to improve accuracy.

If the performance only drops slightly, the MobileNetV3Small student model is then tested with the 1D layer Keras logistic regression model, using the same evaluation metrics for comparison. However, if a significant performance decrease is observed at this stage, the parameters of the Keras logistic regression model will need to be revised, and the testing process will need to be rerun.

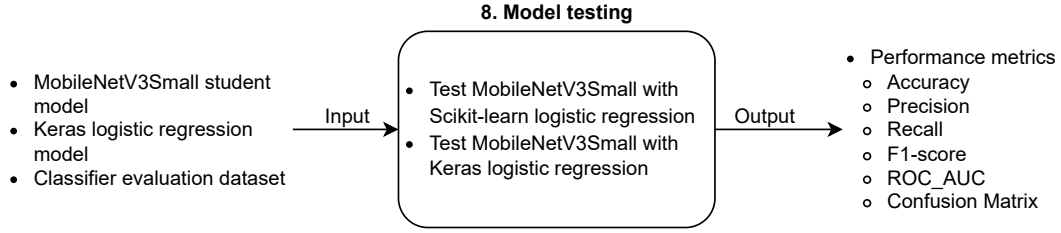

Figure 9: Project Model Testing Process

Once the performance of the student model and the Keras logistic regression model is verified, both models are integrated into a pipeline, which is shown in *Figure 10*. The pipeline began by converting waveforms into log-mel spectrograms. Then, it passes to the MobileNetV3Small student model to extract the characteristics of the log-mel spectrograms and generate embeddings based on them. Lastly, the embeddings are fed into the Keras logistic regression model, which outputs probability values for each audio clip. Therefore, the final model takes a two-second audio clip and outputs a probability value indicating whether it contains a cough.

However, a small performance decrease is expected because of differences in the model mechanisms. The main distinction lies in the data flow between the models. In step 5, the mean–max embeddings from each audio clip are concatenated and then passed into the logistic regression model. While in this pipeline setup, each two-second segment is passed individually through the student model and the Keras logistic regression model, which outputs a probability for each segment. The maximum probability value among the segments is selected. If the value is greater than 0.5, the clip is classified as cough, otherwise as non-cough.

As demonstrated in *Figure 11*, the final deployment model was tested in edge devices. The phone deployment dataset is used to evaluate how well the final model performs in an edge environment. Each audio clip is resampled to 16 kHz and split into two-second segments with 50% overlap to capture all possible cough events. These segments are passed through the final combined model, which outputs a probability value for cough detection. Each audio clip takes the highest probability value among the segments. If the probability value is more than 0.5, it is labeled as a

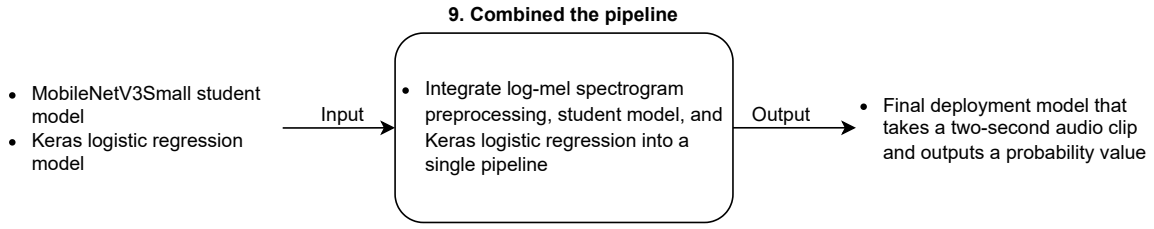

Figure 10: Pipeline Combination Process

cough, otherwise non-cough. The performance of the deployment test was evaluated by key metrics, such as accuracy, precision, recall, F1-score, ROC AUC, and confusion matrix.

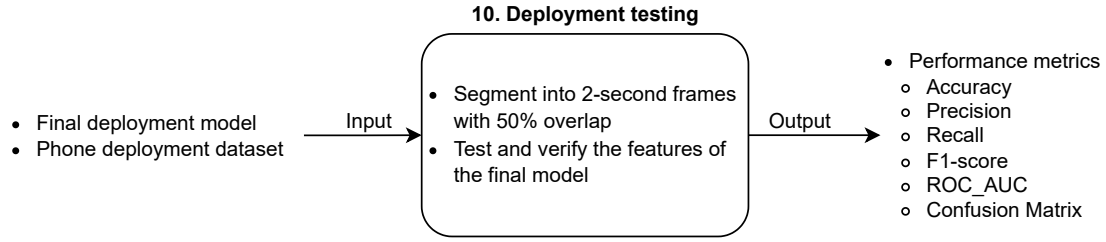

Figure 11: Deployment Testing Process

To ensure reproducibility of this project, all implementation files, experiment configurations, and supporting resources were organized and made publicly accessible through a dedicated GitHub repository (https://github.com/AlexPhang-M/CoughDetection_TinyML_KnowledgeDistillation_google-health). The repository contains the full end-to-end workflow as shown in *Figure 1*, including scripts for data preprocessing, embedding extraction, data splitting, the MobileNetV3Small student model implementation on the knowledge distillation framework, logistic regression classifier conversion, and TensorFlow Lite deployment model pipeline. Additional resources, such as model evaluation, confusion matrices, performance visualizations, and the final TFLite deployment files, are also included. This repository serves as a reference implementation for anyone who is interested in extending the future work of TinyML-based acoustic health modeling.

3.2 Model Deployment Using TensorFlow Lite

TensorFlow Lite (TFLite) was selected as the deployment framework because it provides an optimized inference engine specifically designed for low-power and resource-constrained devices. Unlike Scikit-learn, which requires a full Python runtime, TFLite enables efficient on-device execution through a lightweight interpreter that supports quantized and memory-optimized models. To ensure deployability, the logistic regression classifier was reconstructed as a one-layer Keras model so that it could be directly converted into the TFLite format alongside the MobileNetV3Small student model. The final deployment pipeline processes two-second log-mel spectrogram segments, extracts embeddings using the distilled student model, and performs classification using the Keras logistic regression layer. Running the entire pipeline locally enhances privacy, as audio remains on the device and is never uploaded to external servers. This approach provides fast inference while meeting the computational constraints of edge devices such as mobile phones and Raspberry Pi systems.

3.3 Hardware

The project was carried out on a 64-bit PC running Windows 10 Education. The system used a QEMU virtual CPU operating at 2.8 GHz, with two cores and two logical processors to handle tasks in parallel. It had 32 GB of RAM and

36.7 GB of virtual memory, which allows heavy processes to run smoothly without performance issues. An NVIDIA L40S-6Q graphics card was used to accelerate the machine learning computations by processing in parallel. This can significantly reduce the computation time compared to relying solely on the CPU.

3.4 Software

The project was implemented using Python version 3.10.11 and TensorFlow version 2.10.0 to execute the methodology. The Android Phone Simulator was considered for phone deployment, but it was not used. This is because the system GPU was not compatible with the emulator's required graphics acceleration, which caused the unsuccessful simulation of the Android environment.

To simulate a resource-constrained device, the final deployment model was tested on Raspberry Pi Zero/1 and Raspberry Pi 2/3. The testing on Raspberry Pi Zero/1 failed due to the version of TensorFlow only supporting up to 1.14, which did not support the range function in the TFLite model. On the other hand, the installation on Raspberry Pi 2/3 also failed because the NIC socket in the simulator was not connected, which prevented the console from establishing a connection. Furthermore, the Raspberry Pi Imager tool was unable to download any system image on any Raspberry Pi version.

3.5 Ethics

The project did not use any live audio from participants or patients, which greatly reduces potential privacy and consent concerns. There was also a risk that incorrect predictions could lead to unnecessary worry or harm. To prevent these issues, the project relied only on publicly available and anonymized datasets that contain no personal or identifiable information and are intended for research use. Both the original and compressed HeAR models were used strictly as research prototypes and are not intended for diagnostic or clinical purposes.

3.6 Project Timeline

This project was started after the completion of the project proposal. The development period had a total duration of approximately 12 weeks, from 8 September 2025 to 28 November 2025, to complete the model script and the final report. *Table 1* provides an overview of the key activities conducted each week throughout the project period.

4 Validation

The CoughVid dataset contains a total of 20,072 audio recordings that contain both cough and non-cough sounds. From step 4 of the methodology, 12,043 recordings are allocated for the knowledge distillation process to train the student HeAR model. Additionally, 5,018 recordings are used to train the logistic regression model, and 2,007 recordings are reserved for evaluating its performance. The remaining 1,004 recordings are retained for testing the final model in a resource-constrained phone simulation environment.

To validate the models' performance, evaluation metrics including accuracy, precision, recall, F1-score, and ROC AUC, are applied across all stages. This can highly reflect how well the models generalize new and unseen audio clips.

For clarity, the **Baseline Model** refers to the combination of the pre-trained HeAR model and the Scikit-learn logistic regression. The **Student Model (SKL-LR)** refers to the MobileNetV3Small student model paired with the Scikit-learn logistic regression. In contrast, the **Student Model (Keras-LR)** refers to the MobileNetV3Small student model paired with the 1D layer Keras logistic regression model. The **Final Deployment Model** refers to the final combined pipeline of the log-mel spectrogram input, the MobileNetV3Small student, and the 1D layer Keras logistic regression model. *Table 2* presents the performance comparison of these models.

The student model (SKL-LR) performed slightly better than the baseline model. The reason is that the MobileNetV3Small student model was only trained on cough recordings, whereas the HeAR model is designed to classify a wide range of acoustic events, which include cough events. If the student model were trained together with other acoustic events, its performance would likely fall below that of the baseline model.

Table 1: Project Timeline

Week	Start Date	End Date	Activities
1	8 Sep 2025	14 Sep 2025	• Prepared the Unitec environment mentioned in the Hardware section. • Loaded the pre-trained HeAR model from Hugging Face. • Imported and explored the CoughVid dataset.
2	15 Sep 2025	21 Sep 2025	• Ran the CoughVid dataset through the HeAR model, but the system crashed due to large embedding size. • Extracted and saved embeddings into NumPy files. • Identified that the PQK framework could not be applied because the HeAR model's intermediate layers are inaccessible.
3	22 Sep 2025	28 Sep 2025	• Split the dataset into four subsets. • Trained and evaluated a Scikit-learn logistic regression classifier. • Adopted a black-box knowledge distillation framework. • Searched for a suitable student model for acoustic tasks.
4	29 Sep 2025	5 Oct 2025	• Selected MobileNetV3Small as the student model. • Implemented the black-box knowledge distillation framework. • Researched Android Studio emulators for deployment.
5	6 Oct 2025	12 Oct 2025	• Evaluated the performance of the student model. • Attempted TFLite export with Scikit-learn logistic regression + student model. • Investigated solutions for TFLite export errors.
6	13 Oct 2025	19 Oct 2025	• Re-implemented logistic regression as a 1D Keras layer. • Evaluated the Keras logistic regression with HeAR and student embeddings.
7	20 Oct 2025	26 Oct 2025	• Combined all components and exported the final deployment model to TFLite. • Evaluated the final deployment model. • Analysed reasons for the performance drop.
8–9	27 Oct 2025	9 Nov 2025	• Attempted to run the deployment model on Raspberry Pi Zero/1 and Pi 2/3. • Tested Raspberry Pi imaging tools but encountered installation issues. • Ran the final deployment model in the training environment.
10–12	10 Nov 2025	25 Nov 2025	• Analysed evaluation results and prepared the final report. • Submitted a draft to the supervisor, Dr. Tim Hunt, and incorporated feedback.

Table 2: Performance Metrics of All Models

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Baseline Model	0.87	0.86	0.85	0.85	0.92
Student Model (SKL-LR)	0.88	0.87	0.86	0.87	0.94
Student Model (Keras-LR)	0.88	0.87	0.86	0.87	0.94
Final Deployment Model	0.86	0.85	0.83	0.84	0.91

The student model (Keras-LR) performed the same as the student model (SKL-LR), which shows that converting the logistic regression classifier into a Keras model did not affect its performance or behavior. This result also proved that the coefficients and intercept were transferred correctly from the Scikit-learn logistic regression to the Keras logistic regression.

The final deployment model showed a slight drop in performance compared to the other models. This is expected because the model processed each two-second segment individually and then selected the maximum probability value as the final output of the audio clip. Nevertheless, the previous three models used the mean-max embeddings of the entire audio clip. Although this approach slightly reduced accuracy, it demonstrated how the system operates in a real deployment scenario, which analyzes audio in shorter time windows for faster response.

Table 3 presents a comparison of the confusion matrices for all models. The confusion matrices provide a detailed view of how each model classified cough and non-cough events, compared to the metrics in *Table 2*. Since the project aims to detect cough occurrences, missing a real cough is more critical than classifying a non-cough as a cough. The count of missed cough events (False Non-Cough) was relatively low across all models, which demonstrated that the system is reliable. The student models showed slight improvement in true positive detection (lower False Non-Cough and higher True Cough) compared to the baseline model, which reflects better discrimination between cough and non-cough sounds.

Table 3: Confusion Matrices of All Models

Model	**True Non-Cough**	**False Non-Cough**	**True Cough**	**False Cough**
Baseline Model	536	121	1210	140
Student Model (SKL-LR)	540	97	1234	136
Student Model (Keras-LR)	540	97	1234	136
Final Deployment Model	501	113	1218	175

Table 4 presents the performance metrics of the final deployment model, which was tested using a new and unseen deployment dataset. The final deployment model achieved strong performance with an accuracy of 0.87, a precision of 0.86, a recall of 0.83, an F1-score of 0.84, and an ROC AUC of 0.90. *Figure 12* shows that the model correctly classified 608 cough events (True Cough) and 227 non-cough events (True Non-Cough). However, the model missed 45 cough events (False Non-Cough), and misclassified 85 non-cough events as coughs (False Cough). Overall, the results indicated that the final deployment model is reliable in detecting cough events while maintaining its high sensitivity.

Table 4: Performance Metrics of the Final Deployment Model on the Deployment Dataset

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**ROC AUC**
Final Deployment Model on Deployment Dataset	0.87	0.86	0.83	0.84	0.90

The results indicate that the model achieved steady performance throughout all stages of the methodology. Although the system occasionally missed short or weak cough events and misclassified other events as coughs, it still maintained a highly precise and accurate approximately 0.86. The performance metrics of the final deployment model demonstrate that it is ready for deployment on resource-constrained devices and is suitable for real-world applications.

5 Discussion

The project aims to reduce the size of the HeAR model while maintaining its performance and deploy it on resource-constrained devices. The results of the project indicate that the small final deployment model maintained similar performance to the HeAR model, while preserving the capability to detect cough tasks. This finding demonstrates the potential of model compression can be used in practical health acoustic applications.

As illustrated in *Table 2*, the student model (SKL-LR) achieved a slightly higher performance than the baseline model. The main reason is that the student model was only trained on cough data, while the baseline model is trained to classify multiple health acoustic events. If the student model is trained with the same number of health events as the baseline model, its performance may be lower than the baseline model. Nevertheless, the student model still demonstrated that knowledge distillation techniques have high potential in detecting health-related acoustic events.

The final deployment model has a minimal drop in performance because its mechanism differs from those of the other models. The earlier models utilized the mean and maximum values of all embeddings from each audio clip as an input

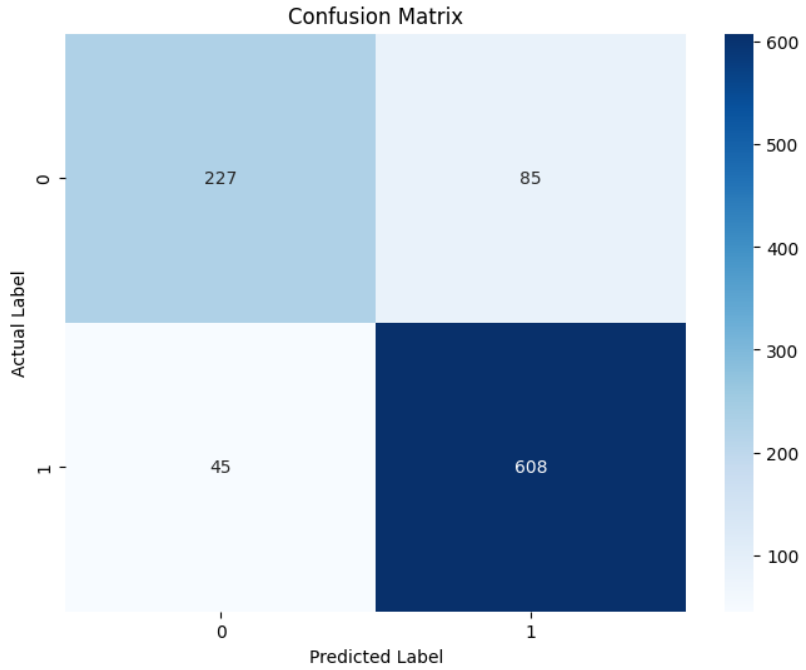

Figure 12: Confusion matrix of the final deployment model on the deployment dataset. True Cough = 608, False Non-Cough = 45, True Non-Cough = 227, False Cough = 85.

to the logistic regression classifier. In contrast, the final deployment model inputs each two-second segment individually and uses the maximum probability value among the segments of each audio clip as the final prediction. Even though there is a minor loss in this approach, it reflects how the model should operate in a real-world environment.

The key factor in the earlier model mechanism is the labelling of the CoughVid dataset. The dataset does not specify which portion of each recording contains cough sounds. This led to the use of mean and max pooling operations to summarize all embeddings into a single embedding, which served as the input to the logistic regression classifier. Then, the classifier learns the general characteristics of cough embeddings to predict cough events. In a real-time detection scenario, the system will not process aggregated audio recordings, but it should analyze every two-second clip individually and determine the probability of cough events.

The project encountered several challenges in the deployment of the final model. The initial plan was to use the Android Studio Emulator software to simulate a real phone environment, but it failed because the platform does not support the NVIDIA L40S-6Q graphics card. The next approach was to use the Raspberry Pi Imager tool to install the Raspberry Pi environment, but this attempt was unsuccessful.

To overcome this, the project tried to manually simulate the Raspberry Pi environment by installing the kernel, CPU, and image drive files individually. The Raspberry Pi Zero/1 was successfully booted up and installed the required Linux and Python libraries. The audio recordings and the TFlite final model were placed on the Raspberry Pi Zero/1, but the script ran unsuccessfully because the latest supported TensorFlow 1.14 is not compatible with the range function used in the TFlite model. The project then attempted to set up the Raspberry Pi 2/3, which supports the TensorFlow 2.10.0 version used during training. However, during boot up, an error message appeared indicating that the NIC port was not attached, which caused the installation process to fail again.

6 Conclusion and Future Work

The project aimed to compress the HeAR model into a lightweight model that can be deployed in resource-constrained devices. This work was motivated by the interest in accessible on-device monitoring systems that can operate efficiently in any environment on Earth [4]. For this reason, the project aimed to enable audio processing directly on embedded hardware, such as mobile phones or Raspberry Pi devices. To validate the performance of the lightweight model, the CoughVid dataset is selected to identify cough events occurring on the audio recordings.

The project successfully implemented a complete deployment pipeline, including a log-mel spectrogram function, a MobileNetV3Small student model trained using the black-box knowledge distillation framework, and a 1D layer Keras logistic classifier. Throughout the entire project, the models achieved a comparable performance to the original HeAR model. The final deployment model was saved in TensorFlow Lite file type and demonstrated strong accuracy, precision, recall, and ROC-AUC values. The results from the confusion matrix confirmed that real-time acoustic health monitoring can be performed efficiently on small devices without the support of cloud services.

The outcomes of this project carried several important implications. The first implication is that the successful compression of the HeAR model demonstrated that high computational models can be adapted into lightweight architectures. This approach showed that large models have the potential to be deployed and run on limited computational resources. Since all processing is performed locally on the device, the user's sensitive information can be safely stored on the device without being transmitted externally. This project also contributes to an understanding of model compression without utilizing teacher logits or internal layers, which can be executed in the black-box knowledge distillation framework.

The project also facilitated a deeper understanding of TinyML (Tiny Machine Learning) techniques for compressing large models. Prior to testing the pre-trained HeAR model, the project explored the PQK framework, which is pruning, quantization, and knowledge distillation, as core strategies for reducing model size. Although the project ultimately used the black-box knowledge distillation framework, this experience strengthened my understanding of how TinyML methods can be applied in practice and highlighted their relevance for deploying machine learning models on resource-constrained devices.

Looking back at the beginning of the project, several technical "showstoppers" were encountered. The pre-trained HeAR model from Hugging Face and the emulator deployment software were not tested and verified before writing the project proposal. Due to this, the project consumed valuable time resolving and seeking solutions due to the lack of knowledge. Therefore, regardless of the project, it should conduct quick preliminary trials to identify any feasibility issues and confirm that the core ideas and techniques will work in practice.

Future work could focus on deploying the final deployment model into both physical and simulated edge devices to evaluate the model's performance in a practical setting. Additionally, it should aim to expand other health-related acoustic events in the deployment pipeline that was introduced in this project. These explorations definitely provide valuable insights into health monitoring applications in real-life settings. Overall, the project demonstrated a pathway for using machine learning techniques towards on-device health acoustic applications.

7 Acknowledgement

I would like to thank my supervisor, Dr Tim Hunt, for his consistent guidance and feedback throughout the entire project. I appreciate that he always leads the project in the correct direction and provides clarity when needed. I am grateful to have had this opportunity to complete this research under his supervision and to have benefited from his valuable insights into each aspect of the project.

I would also like to thank my partner, Tan Jia Wei, for her unwavering support throughout this project. Her ongoing support, patience, and understanding were what kept me motivated throughout the challenging stages of experimentation and writing. She was a tremendous source of mental and emotional strength, empowering me to handle the workload effectively and persevere with this research.

References

[1] S. Ghrabli, M. Elgendi, and C. Menon, "Identifying unique spectral fingerprints in cough sounds for diagnosing respiratory ailments," *Scientific Reports*, vol. 14, no. 593, 2024. [Online]. Available: https://doi.org/10.1038/s41598-023-50371-2

[2] S. Hegde, S. Sreeram, I. L. Alter, C. Shor, T. A. Valdez, K. D. Meister, and A. Rameau, "Cough sounds in screening and diagnostics: a scoping review," *The Laryngoscope*, vol. 134, no. 3, pp. 1023–1031, 2024. [Online]. Available: https://doi.org/10.1002/lary.31042

[3] C. Brown, J. Chauhan, A. Grammenos, J. Han, A. Hasthanasombat, D. Spathis, T. Xia, P. Cicuta, and C. Mascolo, "Exploring automatic diagnosis of COVID-19 from crowdsourced respiratory sound data," in *Proc. 26th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining* (KDD '20), Virtual Event, CA, USA, Aug. 23-27, 2020, pp. 3474-3484. [Online]. Available: https://doi.org/10.1145/3394486.3412865

[4] A. Imran, I. Posokhova, H. N. Qureshi, U. Masood, M. S. Riaz, K. Ali, C. N. John, I. Hussain, and M. Nabeel, "AI4COVID-19: AI enabled preliminary diagnosis for COVID-19 from cough samples via an app," *Informatics in Medicine Unlocked*, vol. 20, p. 100378, 2020. [Online]. Available: https://doi.org/10.1016/j.imu.2020.100378

[5] N. D. Lane, P. Georgiev, and L. Qendro, "DeepEar: Robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *Proc. 2015 ACM Int. Joint Conf. Pervasive and Ubiquitous Computing (UbiComp '15)*, Osaka, Japan, Sep 2015, pp. 283-294. [Online]. Available: https://doi.org/10.1145/2750858.2804262

[6] S. Baur, et al., "HeAR – Health Acoustic Representations," *arXiv preprint* arXiv:2403.02522, Mar. 2024. [Online]. Available: https://arxiv.org/abs/2403.02522

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint* arXiv:1503.02531, Mar. 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1503.02531

[8] Google, "HeAR – Health Acoustic Representations," Hugging Face model card, 2024. [Online]. Available: https://huggingface.co/google/hear

[9] Z. Wang, "Zero-Shot Knowledge Distillation from a Decision-Based Black-Box Model," in *Proc. 38th Int. Conf. Machine Learning (ICML 2021),* PMLR 139, Virtual Event, Jul. 2021, pp. 10675–10685. [Online]. Available: https://arxiv.org/abs/2106.03310

[10] A. Ehtesham, S. Kumar, A. Singh, and T. T. Khoei, "Pediatric asthma detection with Google's HeAR model: An AI-driven respiratory sound classifier," arXiv preprint arXiv:2504.20124, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2504.20124

[11] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 3rd ed. Sebastopol, CA: O'Reilly Media, Oct. 2022.

[12] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PLoS ONE*, vol. 10, no. 3, e0118432, Mar. 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0118432

[13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked Autoencoders Are Scalable Vision Learners," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR '22)*, New Orleans, LA, USA, Jun. 2022, pp. 16000-16009. [Online]. Available: https://doi.org/10.1109/CVPR52688.2022.01553

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image Is Worth 16×16 Words: Transformers for Image Recognition at Scale," *arXiv preprint* arXiv:2010.11929, Oct. 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[15] L. Orlandic, T. Teijeiro, and D. Atienza, "The COUGHVID crowdsourcing dataset, a corpus for the study of large-scale cough analysis algorithms," *Scientific Data*, vol. 8, no. 1, p. 156, Jun. 2021. [Online]. Available: https://doi.org/10.1038/s41597-021-00937-4

[16] D. Bhattacharya, N. K. Sharma, D. Dutta, S. R. Chetupalli, P. Mote, S. Ganapathy, C. Chandrakiran, S. Nori, K. K. Suhail, S. Gonuguntla, and M. Alagesan, "Coswara: A respiratory sounds and symptoms dataset for remote screening of SARS-CoV-2 infection," *Scientific Data*, vol. 10, no. 1, p. 397, Jun. 2023. [Online]. Available: https://doi.org/10.1038/s41597-023-02266-0

[17] Q. Zhang, J. Zhang, J. Yuan, H. Huang, Y. Zhang, B. Zhang, G. Lv, S. Lin, N. Wang, X. Liu, M. Tang, Y. Wang, H. Ma, L. Liu, S. Yuan, H. Zhou, J. Zhao, Y. Li, Y. Yin, L. Zhao, G. Wang, and Y. Lian, "SPRSound: Open-Source SJTU Paediatric Respiratory Sound Database," *IEEE Trans. Biomed. Circuits Syst.*, vol. 16, no. 5, pp. 867–881, Oct. 2022. [Online]. Available: https://doi.org/10.1109/TBCAS.2022.3204910

[18] Z.Wang, "Zero-Shot Knowledge Distillation from a Decision-Based Black-Box Model," in *Proc. 38th Int. Conf. Machine Learning (ICML '21)*, PMLR vol.139, 2021, pp.10675–10685. [Online]. Available: https://proceedings.mlr.press/v139/wang21a.html

[19] Y.Jin, J.Wang, and D.Lin, "Multi-Level Logit Distillation," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR '23)*, June 2023, pp.24276–24285. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2023/html/Jin_Multi-Level_Logit_Distillation_CVPR_2023_paper.html

[20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint* arXiv:1503.02531, Mar. 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1503.02531

[21] C.Guo, G.Pleiss, Y.Sun, and K.Q.Weinberger, "On Calibration of Modern Neural Networks," in *Proc. 34th Int. Conf. Machine Learning (ICML '17)*, PMLR vol.70, 2017, pp.1321–1330. [Online]. Available: https://proceedings.mlr.press/v70/guo17a.html

[22] D. Stowell, "Computational bioacoustics with deep learning: a review and roadmap," *PeerJ*, vol. 10, art. e13152, Mar. 2022. [Online]. Available: https://doi.org10.7717/peerj.13152

[23] D. Priebe, B. Ghani, and D. Stowell, "Efficient Speech Detection in Environmental Audio Using Acoustic Recognition and Knowledge Distillation," *Sensors*, vol. 24, no. 7, p. 2046, 2024. [Online]. Available: https://doi.org/10.3390/s24072046

[24] Cretois, B., C. M. Rosten, and S. S. Sethi, "Voice activity detection in eco-acoustic data enables privacy protection and is a proxy for human disturbance," *Methods in Ecology and Evolution*, vol. 13, no. 12, pp. 2865–2874, 2022. [Online]. Available: https://dor.org/10.1111/2041-210X.14005

[25] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," in *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision (ICCV '19)*, Seoul, Korea, Oct.–Nov. 2019, pp. 1314–1324. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00140

[26] S. Hedar, "Applying Machine Learning Methods to Predict the Outcome of Shots in Football," Master's Thesis, Uppsala University, 2020. [Online]. Available: https://www.diva-portal.org/smash/get/diva2%3A1448482/FULLTEXT01.pdf

[27] G. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.

[28] E. P. Csirmaz, "Linear and Logistic Regressions as Degenerate Neural Networks in Keras," *TDS Archive* (Medium), Apr. 17, 2022. [Online]. Available: https://medium.com/data-science/linear-and-logistic-regressions-as-degenerate-neural-networks-in-keras-c9b309cbd80a

[29] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS One*, vol. 10, no. 3, p. e0118432, Mar. 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0118432

[30] J. Li, "Area under the ROC Curve has the most consistent evaluation for binary classification," *PLoS ONE*, vol. 19, no. 12, p. e0316019, Dec. 2024. [Online]. Available: https://doi.org/10.1371/journal.pone.0316019