### Convolutional Neural Network Optimization for Beehive Classification Using Bioacoustic Signals

Harshit <sup>1\*</sup>, Rahul Jana<sup>2, 3</sup> and Ritesh Kumar<sup>2\*</sup>

- <sup>1</sup>National Institute of Technology, Thanesar, Kurukshetra, 136119, Haryana, India.
- <sup>2</sup> CSIR-Central Scientific Instruments Organization, Chandigarh, 160030, India.
- <sup>3</sup> TIF Agriculture and Water Technology Development Hub(AWaDH), Indian Institute of Technology, Bara Phool, 140001, Punjab, India.

\*Corresponding author(s). E-mail(s): harshitwork4032@gmail.com; riteshkr@csio.res.in;

 $Contributing \ authors: {\bf i.rahuljana@gmail.com};$ 

#### Abstract

The behavior of honeybees is an important ecological phenomenon not only in terms of honey and beeswax production but also due to the proliferation of flora and fauna around it. The best way to study this significant phenomenon is by non-invasive monitoring of beehives using the sounds produced by various body movements that give out audio signals which can be exploited for various predictions related to the objectives mentioned above. This study investigates the application of Convolutional Neural Networks to classify and monitor different hive states with the help of joint time and frequency image representations such as Spectrogram, Mel-Spectrogram, Smoothed-Spectrogram, and Cochleagram. Our findings indicate that the Cochleagram outperformed all the other representations, achieving an accuracy of 98.31% on unseen data. Furthermore, we employed various strategies including pruning, quantization, and knowledge distillation to optimize the network and prevent any potential issues with model size. With these optimizations, the network size was lowered by 91.8% and the inference time was accelerated by 66%, increasing its suitability for real-time applications. Thus our study emphasizes the significance of using optimization approaches to minimize model size, avoid deployment problems, and expedite inference for real-time application as well as the selection of an appropriate time-frequency representation for optimal performance.

#### 1 Introduction

Bees are essential to our ecosystem, playing a crucial role in pollination [1] and honey production. In an effort to enhance honey production and ensure the sustainability of the population, artificial or man-made beehives have become increasingly important. For better production and regulating the bee population, queen bee plays a crucial role [2]. The queen bee's ability to lay eggs and regulate the hive's social structure directly impacts the health and productivity of the colony [3]. Proper monitoring of these dynamics is crucial for maintaining a healthy hive, especially in artificial environments. Following the importance of the queen bee in maintaining the productive hive we explored audio signals which can be used to classify and monitor the behavior of the hive by analyzing the sound produced by the bees within the hive [4]. Studies have shown that different hive states produce distinct acoustic signatures, which can be effectively classified using advanced machine learning techniques such as Convolutional Neural Networks (CNNs) [5]. This approach not only helps in early detection of hive issues but also aids in optimizing hive management practices [6].

Convolutional neural networks have been showing remarkable results in understanding image classification problems [7] and now being widely applied to other domains, including signal classification [8]. Since CNNs utilize images of fixed dimensions as input, it has become really essential to wisely choose the techniques to represent the audio signal into an image. For this, we experimented four different time–frequency representation coupled with the CNN including Spectrogram, Mel–Spectrogram, Smoothed–Spectrogram and Cochleagram. As shown in the paper by Haran at el., cochleagram time–frequency image representation fits the best with CNN [9], and we witnessed the same that the cochleagram time–frequency image representation outperformed all the other representations. And after hypertuning, network improvement and data augmentation by transforming the raw audio files (e.g. pitch shift, time stretch, and change speed) [10] we achieved an accuracy of 98.31% on unseen data.

Although the performance of the Cochlea–CNN is outstanding, but the size and inference time can cause deployment issues, exploitation of resources and may cause lag in the real–time operation because of larger size and slow inference respectively as compared to traditional statistical machine learning approach. In our research we explored different size reduction and inference acceleration techniques including pruning, knowledge distillation and quantization and implemented them subsequently while preserving the performance of the Cholea–CNN. After optimization, we significantly reduced the size of the model by 91.8% from 21.82 mb to 1.79 mb and accelerated the inference by 66%. Our research shows the role of choosing the appropriate time-frequency representation to achieve high performance and the potential of optimization techniques for reducing the size of the model to avoid any deployment issues, resources exploitation and to accelerate inference to avoid lag in the real–time process.

The remainder of this paper is organized as follows. Section 2 lists a review of the related work in bioacoustic signal classification and beehive monitoring and classification. Section 3 details the proposed methodology for benchmarking and optimization, including data preprocessing, feature extraction, classification approach, and model optimization techniques for reducing neural network size and inference time. Sections 4 & 5 shows the experimental results, and an in-depth performance analysis and discussion. Finally, Section 6 concludes the paper and outlines potential directions for future research.

#### 2 Related Work

Several studies have explored sound-based monitoring on beehives. For instance, a proposed method for identifying queen bees using audio analysis and machine learning was developed [11]. In their study, audio data was sourced from a public dataset provided by the Nu-Hive project [12]. In their work, SVM and CNN were the experimented models and SVM performed better than CNN with an average test AUC score of approximately 0.94. The author performed experiments using SVMs, incorporating various feature sets such as 120 frequency bands from 85 Mel spectra, 20 MFCCs, and 20 bands from the Hilbert–Huang Transform (HHT). They also exploited MFCCs and Mel-spectrograms coupled with CNN for feature extraction, and the resulting features were then input into the SVM classifier, achieving an AUC of approximately 0.82.

Another study explored the effectiveness of various algorithms in machine learning, including Support Vector Machines (SVM), Random Forest (RF), and Extreme Gradient Boosting (XGBoost), as well as two CNN architectures—shallow CNN and VGG-13—for feature extraction in identifying bee and non-bee sounds [4]. In their study, they used audio data from beehives provided by the Open Source Beehives (OSBH) initiative, which is included in a publicly available dataset [11]. Their findings showed that VGG-13, coupled with Mel Frequency Cepstral Coefficients (MFCCs) as input, achieved an accuracy of 91.93%. Moreover, VGG-13 proved most effective at identifying non-bee sounds, with an F1-score of 0.79.

Similarly, the research utilized CNN coupled with MFCC wave spectrogram images for identifying the beehive sound [3]; this approach achieved 85.04% accuracy.

Soares et al. [13] explored an MFCC-based descriptor using machine learning and deep learning methods to identify queen bee presence by analyzing the spectral and temporal characteristics of hive acoustics. The author utilized publicly available datasets related to queen bee presence, specifically incorporating audio recordings from the Open Source Beehive project and the NU-Hive dataset [12], with annotations provided by Nolasco and Benetos [14]. A total of 508 samples, each averaging 10 minutes in duration, were processed into 15-second segments. These segments were converted into Mel-spectrograms on a logarithmic scale for CNN input, using a sampling rate of 22,050 Hz, window size of 4096, hop length of 2030, and the Hann window function [12].

They evaluated four CNN architectures that were pre-trained on the ImageNet dataset [15] to extract deep features: ResNet50 [16], VGG16 [17], VGG19 [17], and Inception\_v3 [18]. In their work, handcrafted features were chosen to distinguish

between the presence and absence of the queen bee, and recursive feature elimination with cross-validation (RFECV) [19] was applied to refine the feature set. The algorithm identified 36 significant features, which included 31 MFCC coefficients, two contrast measures, zero-crossing rate (ZCR), polynomial coefficients, and spectral flatness. For classification, the author explored the Random Forest (RF) and Support Vector Machine (SVM) classifiers and fed the extracted features to them. The best classification performance using deep features was achieved with the ResNet50 model combined with Random Forest (RF), reaching an accuracy of 0.99, while the combination of MFCC features with RF also yielded an accuracy of 0.99. However, their work only focuses on binary classification — either the queen bee is present or not.

However, to our knowledge, no one has yet worked or experimented with cochleagram time-frequency representation for beehive classification and multiclass beehive state classification. In our research, we explored the potential of cochleagram time-frequency representation, outperforming all other time-frequency representations including spectrogram, mel-spectrogram, and smoothed-spectrogram, showing the potential of CNNs in classifying audio signals. In addition, the dataset was divided into four classes: Queen Not Present, Queen Present and Newly Accepted, Queen Present and Rejected, Queen Present or Original Queen.

#### 3 Methodology

This section outlines the detailed methodology implemented to develop and optimize a CNN for classifying beehive states from bioacoustic data is shown. We started with the conversion of raw bioacoustic signals into spectrograms, Mel-spectrograms, smoothed-spectrograms, and cochleagrams time—frequency representations. These representations are used as fixed dimension input to Convolutional Neural Networks were employed to extract features from the time-frequency representation of bioacoustic data. We then benchmarked the effectiveness of each image representation was assessed to determine the most suitable one with the CNNs. Subsequently, we describe the classification model and evaluation metrics which are used to validate and evaluate the model. For Further enhancing the efficiency, we applied three model optimization techniques, pruning, knowledge distillation, and quantization. These strategies significantly reduce model/network size and inference time while preserving the model's accuracy and effectiveness. The following subsections detail each step, from feature extraction to network compression.

#### 3.1 Data Collection

The bees audio data we have used was collected from kaggle.com [20]. The dataset, is the largest of its kind which consists of beehive audio data paired with multi-dimensional sensor data [5]. It was gathered utilizing a specially designed IoT device equipped with an ESP32 Wi-Fi module, an INMP441 microphone, and a BME280 sensor to measure temperature and humidity. The recordings were obtained from European honeybee colonies located in California and is divided into 60-second segments, resulting in a total of 7,100 samples. It was divided into four classes Queen

Not Present, Queen Present and Newly Accepted, Queen Present and Rejected, Queen Present or Original Queen [3].

#### 3.2 Data Preprocessing

It has become really important to process raw data since we were using CNN for classification, and it consumes images of fixed input to perform further computations [21]. In our research, we experimented with four different time-frequency image representations of raw audio data files including spectrogram, mel-spectrogram, smoothed-spectrogram, and cochleagram as shown in "Fig. 1a, 1b, 1c and 1d" respectively. We adopted LibROSA library [22] to generate spectrogram, Mel-Spectrogram, and Smoothed-spectrogram and the Cochleagram representations were generated using gtgram module present in the gammatone library [23]

#### 3.2.1 Spectrogram

Spectrograms have been produced through the application of the Short-Time Fourier Transform (STFT) [24]. It first breaks the audio signals in overlapping frames, and then DFT was applied to each frame to convert them to frequency domain [24, 25]. Formula for segmenting the signal into overlapping frames and DFT for each frame is shown in equation 1, where x[n] represents the discrete time sample of the signal, n is the length of each frame, m is the step size between frames ( usually m < n) and w[n] is a window function applied on each frame to minimize spectral leakage. [24, 26]

$$x_m[n] = x[n + mM] \cdot w[n] \tag{1}$$

DFT can be computed using the equation 2.

$$X(k,r) = \sum_{n=0}^{N-1} x(n)w(n)e^{-2\pi i \frac{kn}{N}}, \quad k = 0, \dots, N-1$$
 (2)

Where, X(k,r) is the k-th harmonic for the r-th frame [27]. The spectrogram values are then obtained by taking the logarithm of the magnitude of the DFT values:

$$S(k,r) = \log|X(k,r)|^2 \tag{3}$$

The number of points for FFT and the hope length of the STFT was 1024 and 512 respectively. These parameters determine the time-frequency resolution of the spectrogram and are commonly used in signal processing tasks [28].

#### 3.2.2 Mel-Spectrogram

he Mel-spectrogram represents a modified form of a spectrogram where the frequency axis is transformed according to the Mel scale. It was obtained after computing STFT using equation 3, then the magnitude spectrogram was converted to a power spectrogram using equation 4 before passing it through the Mel-Filterbank [22, 29].

$$P(k,r) = \left| S(k,r) \right|^2 \tag{4}$$

For a given power spectrogram P(m,k), the Mel-Spectrogram M(m,n) is obtained using the equation 5.

$$M(k,r) = \sum_{n=0}^{N-1} H_r(n) \cdot P(k,n)$$
 (5)

where  $H_r(n)$  signifies the weight of the r-th Mel filter for the n-th frequency bin, and N denotes the total count of frequency bins in the spectrogram [30]. The count of points for FFT and the hope length was the same as in the spectrogram.

#### 3.2.3 Smoothed-Spectrogram

It follows the same process that was used to generate spectrograms with additional smoothing by taking the moving average along the time and frequency axis [31]. Equation 6, 7 is used for time and frequency smoothing respectively.

$$S_{\text{smooth}}(k,r) = \frac{1}{T} \sum_{t=-T/2}^{T/2} S(k+t,r)$$
 (6)

where T is the window size over which you average the neighboring time frames.

$$S_{\text{smooth}}(k,r) = \frac{1}{F} \sum_{f=-F/2}^{F/2} S(k,r+f)$$
 (7)

with F indicating the the window size over which you average the neighboring frequency bins.

#### 3.2.4 Cochleagram

The gammatone filter's impulse response, employed in creating the cochleagram image representation, using the equation 8,

$$h(t) = At^{j-1}e^{-2\pi Bt}\cos(2\pi f_c t + \phi)$$
(8)

In this context, A denotes the amplitude, j indicates the filter's order, B defines its bandwidth, and  $f_c$  refers to the center frequency of the filter,  $\phi$  corresponding to the phase component, and t represents time [31]. The maximum frequency for the gammatone filters (in Hz) is set to be 20, and the window time, hop time, and the count of channels are set to 0.025, 0.010, and 32, respectively.

The raw audio is downsampled to 16khz before converting the image representation of the audios, the original raw audio's sampled rate was 22050 hz but in our experimentation, the representations of downsampled audios performed better than the time–frequency representation of the original raw audio of sample rate of 22050 hz. Since sound produced by majority of the bees stays below 1000 Hz [5, 24], so downsampling the audio didn't cost us any loss in information and additionally it reduced the memory dependency.

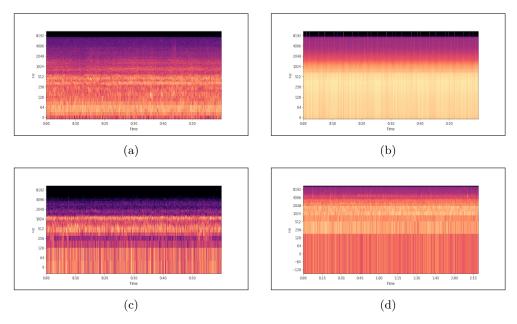


Fig. 1: Comparison of time–frequency image representations of a beehive bioacoustic signal using different transformation techniques. (a) Spectrogram, (b) Smoothed Spectrogram, (c) Mel–Spectrogram, and (d) Cochleagram. These representations are used as input to the CNN model for multiclass hive state classification.

## 3.3 Model architecture & Benchmarking time-frequency Image Representation

#### 3.3.1 Feature Extraction

The CNN architectures and hyperparameter configurations are detailed in Fig. 2 and Table 1 respectively. An input image representation with dimensions  $64 \times 64 \times 3$  was employed. This network features eight convolutional layers (Conv. i where  $i \in \mathbb{N} \mid 1 \leq n \leq 8$ ), each followed by ReLU activation, max pooling and batch normalization on every odd number layer as shown in the Fig. 2.

Table 1: Hyperparameter settings used for training the proposed CNN model on time–frequency image representations of beehive audio signals.

| HyperParameters                  | Values  |
|----------------------------------|---------|
| Optimization algorithm           | Rmsprop |
| Max epochs                       | 250     |
| learning rate reduction factor   | 0.5     |
| learning rate reduction interval | 6       |

Fig. 2: Architecture of the proposed CNN model used for feature extraction and classification. The network processes time–frequency image representations of beehive audio signals through stacked convolutional and pooling layers, followed by dense layers for multiclass hive state prediction.

#### 3.3.2 Classification

After extraction of the features from the time—frequency representation we fed those features into the shallow artificial neural network (ANN) model. The ANN consists of 2 layers containing 36 and 4 hidden units respectively. The final output was passed through a softmax function, the softmax function is described in equation 9,

$$\operatorname{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$
(9)

Where  $z_i$  represents the i-th logit, N is the total number of logits and e is the exponential function. It guarantees that the output probabilities range from 0 to 1 and collectively sum to 1, allowing them to be interpreted as valid probability values for classification tasks [32, 33].

#### 3.3.3 Benchmarking time-frequency representation

We trained and evaluated the cnn model with each representation, and the cochleagram representation outperformed all the other representations with respect to accuracy, as presented in Table 2.

**Table 2:** Classification accuracy (%) of the proposed CNN model using different time–frequency image representations of beehive audio signals.

| Representation       | Testing Accuracy |
|----------------------|------------------|
| Spectrogram          | 31.61%           |
| Smoothed-Spectrogram | 24.11%           |
| Mel-Spectrogram      | 66.66%           |
| Cochleagram          | <b>74.97</b> %   |

The cochlea-CNN model performed best among the other time-frequency representations and achieved an accuracy of 74.97%. After that network tuning, modification and data augmentation by transforming the raw audio files (e.g. pitch shift, time stretch, and change speed) [10] the cochlea-CNN model attained a notable accuracy of 98.31% on previously unseen test data. The Cochleagram Image Representation of all four beehive states is shown in "Fig 3".

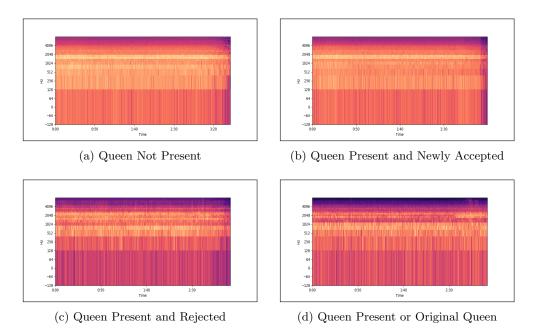


Fig. 3: Cochleagram-based time-frequency image representations: (a) Queen Not Present, (b) Queen Present and Newly Accepted, (c) Queen Present and Rejected, (d) Queen Present or Original Queen. These visualizations highlight distinct acoustic patterns used for hive state classification.

#### 3.4 Training and Validation

We selected the accuracy, F1-score, along with the confusion matrix to assess the classifier's performance. The following techniques are described in equation 10, 11 and Fig. 4 respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

Accuracy = 
$$\frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = \frac{TP}{TP + 0.5(FP + FN)}$$
(10)

where TP, FP, TN and FN represent true positives, false positives, true negatives, and false negatives, respectively. These methods offer an in-depth assessment of

#### Actual Values

|             | Positive(1) | Negative(0)    |
|-------------|-------------|----------------|
| Positive(1) | TP          | FP             |
| Negative(0) | FN          | TN             |
|             |             | Positive(1) TP |

Fig. 4: Confusion Matrix Diagram

a classification model. Accuracy offers an overall indication of how well the model performs, where F1–score focuses on the minority classes and the confusion matrix that breaks down model outputs into true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) [34, 35]. We split the data set with an 85:15 split between training and validation sets, respectively and the count of data samples was 2368. Fig. 5a and 5b illustrate the training and validation loss and accuracy across epochs, respectively.

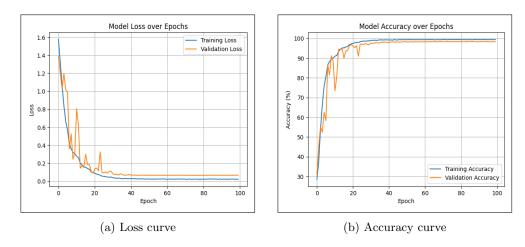


Fig. 5: Training trends of the CNN model using cochleagram inputs: (a)
Training and validation loss, (b) Training and validation accuracy.

#### 3.5 Neural Network Optimization

For size reduction and inference acceleration we implemented pruning, knowledge distillation and quantization subsequently. And reduced the size of the model by 91.8% and inference by 66.4%, while preserving the performance of the model.

#### 3.5.1 Pruning

During the pruning process we performed two types of pruning: random neuron and layer pruning. Random Neuron pruning is the process of removing the neurons from the layer of the network [36]. And through analysis we've seen that those conv layers without batch normalization didn't participate in the learning and classification task. So removing those layers didn't cost any performance and significantly reduced the size of the model [37, 38].

#### 3.5.2 Knowledge Distillation

Knowledge Distillation is also called model distillation, in this process the knowledge is transferred from the bigger model to a smaller and less complex model. This idea was initially introduced by Geoffrey Hinton, et al. [39], and the objective is to guide the student model to emulate the performance of the teacher model, focusing on achieving computational efficiency and potentially reducing model size. During training, the student model learns from both, the actual labels and the softened outputs generated by the teacher model. The distillation loss encourages the student model to reduce the difference between its predictions and the actual truth labels, while also aiming to match the distribution of predictions generated by the teacher model [39], as illustrated in "Algorithm 1".

Both models use a combination of convolutional layers followed by ReLU activations, batch normalization, and max pooling operations in the feature extraction part, followed by dropout, flatten, and dense layers combined with batch normalization in the classification part. The architecture of both the models are shown in "Fig. A1", the Hidden Layers may not be reduced in the after knowledge–distillation but the number of units/neurons is decreased which decreased the total count of parameters by 40.5% and the model size reduced from 4.29 MB to 2.56 MB which is nearly 10 times smaller than the earliest model with the almost same accuracy.

#### 3.5.3 Quantization

In Quantization we reduced the precision of a model's weights and activations by converting them from floating-point precision to lower-bit integer formats, such as 16-bit or 8-bit integers [40]. This process significantly reduced the overall model size while accelerating the inference process by reducing the computational load. We encountered some problems during the quantization. As we had to fuse multiple sequential modules (eg: [Conv2d, BatchNorm, ReLU]) into one [41], but when we tried to fuse the [Conv2d, BatchNorm] layers the accuracy of the model/network decreased drastically. So we tried to calibrate the model after fusing for several epochs, but still the results were the same. We also trained the fused model but the training loss wasn't converging and the effectiveness of the model remained the same. Since the size of the Distilled network (Student Model) was already small relative to the initial model, so instead of quantizing the whole model/network we only quantized the Classification layer of the model (Dense layer) as nearly  $\frac{1}{3}$  of the total networks is consist of classification layer. The total count of parameters in the student layer were 651,404 and the

number of parameters in the classification layer were 262,604, the classification network architecture as outlined in Table 3. The count of parameters in the classification layer were calculated using the procedure written below.

Table 3: Layer-wise configuration of the fully connected classification head of the CNN model. The architecture transforms 4D feature maps into class scores through dropout regularization and two sequential linear layers with batch normalization.

| Layer                | Description   |
|----------------------|---|
| Dropout 1            | Dropout with probability 0.5  |
| Flatten              | Converts 4D tensor to 2D  |
| Linear + BatchNorm 1 | Input: $256 \times 4 \times 4 \rightarrow \text{Output: } 64 \text{ units}$ |
| Dropout 2            | Dropout with probability 0.5  |
| Linear + BatchNorm 2 | Input: $64 \rightarrow \text{Output: 4 (number of classes)}$                |

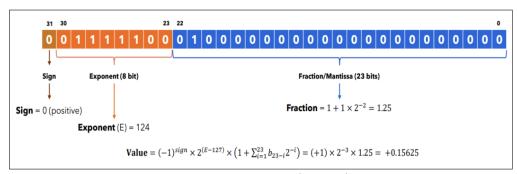


Fig. 6: IEEE 754 single-precision floating-point (Float32) representation format. It consists of a 1-bit sign, 8-bit exponent, and 23-bit mantissa (fraction). This standard format serves as the baseline for model weights and activations before applying lower-bit quantization techniques in neural network optimization [42].

The total count of parameters in the given block is 262,604. Since all the parameters of the Distilled network were in float 32, the size in megabytes of the 262,604 parameters  $\approx 1.002$  MB floating point numbers representation is shown in Fig. 6. And after quantization, the parameters of the classifier layer are converted to int8 which reduces the size of the classifier layer 4 times.

So after quantizing the distilled network's classification layer the size compressed to 1.79 mb. This can be calculated by subtracting the storage requirement of the classifier layer from that of the student model, and then adding 25% of the classifier layer's size.

Size of the classification layer  $(S_{CL}) \approx 1.002$  MB Size of the Student Model  $(S_{SM}) = 2.5$  MB Size of the Quantized model =  $S_{SM}$  – $S_{SL}$ + 25100 x  $S_{SL}$ Size of the Quantized model = 2.5 – 1.002 + 0.2505 = 1.75 MB

1.75 mb is the calculated storage requirement of the model and 1.79 mb is the observed size of the model which is very close.

We followed a post-training quantization procedure since we were not quantizing the whole model. First, we determined the range of weights and activations in the classification layer using a calibration dataset [43]. Then, we computed the scale factor S and zero-offset Z using the equation 12 & 13,

$$S = \frac{x_{\text{max}} - x_{\text{min}}}{q_{\text{max}} - q_{\text{min}}} \tag{12}$$

$$Z = \text{round}\left(q_{\min} - \frac{x_{\min}}{S}\right) \tag{13}$$

where  $x_{\min}$  and  $x_{\min}$  is the minimum and the maximum values of the original weights/activations. Using these values, each floating-point weight/ activation x was quantized as in equation 14,

$$q = \text{round}\left(\frac{S}{x} + Z\right) \tag{14}$$

During inference, the quantized weights and activations were dequantized back into floating-point approximations using the equation 15,

$$x_{\text{approx}} = S \cdot (q - Z) \tag{15}$$

This allowed the classification layer to perform computations with reduced precision, minimizing the model's memory dependencies and computational demands, while retaining the majority of the network's original performance [43, 44]. By quantizing only the classification layer, we preserved the model's overall accuracy while achieving a significant reduction in inference time and resource consumption [44, 45].

#### 4 Results

This section outlines the conclusions drawn from the research work and the detailed results of the proposed models are shown and analyzed. The results are organized in way to show the comparative analysis of different time-series representation, their effectiveness with CNN and the impact of models optimization techniques. Each subsection highlights the keys insights, supported by the evaluation metrics employing evaluation measures like accuracy, F1-score, and confusion matrix, to understand the comprehensive understanding of the proposed model performance. Model performance and optimization results and detail analysis is shown section 4.1 and 4.2 respectively.

#### 4.1 Model Performance

Initially the use of cochleagram image representation resulted in superior classification performance in comparison to the other time-frequency representations achieving as

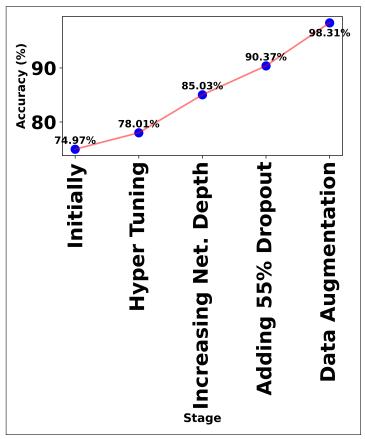


Fig. 7: Accuracy progression across key model development stages for the CNN using cochleagram inputs. Each stage—initial setup, hyperparameter tuning, increased network depth, dropout regularization, and data augmentation—demonstrates incremental improvements, culminating in a peak accuracy of 98.31%.

accuracy of 74.97% as shown in Table 2. And after model hyper parameter tuning the accuracy of the model increase by 4.05% and achieved the accuracy of 78.01%. Then we increased the depth of the neural network by increasing the count of convolution layer followed by batchnorm layer and max pooling layer which improved the model's overall accuracy by 9% and achieved the accuracy of 85.03%. After that we regularize the model by adding 55% dropout in the classification layer which improve the model's accuracy by 6.28% and achieved the accuracy of 90.37%. And at the end we performed data augmentation on the raw audio data which substantially enhanced the model's performance by 8.8% and pushed the model performance to 98.31% test accuracy. The progress graph is shown in "Fig. 7", illustrates the progressive improvement in accuracy through various stages of model refinement. Each stage significantly boosted the accuracy from an initial 74.97% to a remarkable 98.31% after implementing data augmentation.

The classification report of the best performing model can be observed in Table 4, which indicates the balanced performance of the model across all classes that are Queen Not Present, Queen Present and Newly Accepted, Queen Present and Rejected, Queen Present or Original Queen. The class "Queen not present" achieved a precision value of 0.98, a recall value of 0.99 along with an F1-score of 0.99. The "Queen present and newly accepted" class recorded a marginally reduced recall recall of 0.97 but maintained high precision (0.99) and F1-score (0.98). The "Queen present and rejected" class yielded consistent scores of 0.97 across all three metrics. Notably, the "Queen present or original queen" class achieved nearly perfect classification with a recall of 1.00 and F1-score of 0.99.

To further assess the model performance and class-wise behavior, we analyzed the confusion matrix is depicted in "Fig. 8" on unseen data. The model demonstrated particularly high accuracy in distinguishing between the classes, correctly identifying 98.96% of the "queen not present" samples, 98.25% of "queen present and newly accepted," 97.24% of "queen present and rejected," and 98.97% of the "queen present or original queen" samples. Most of the misclassifications were limited to acoustically similar categories, such as confusion between the "newly accepted" and "rejected" classes. This shows the model's capacity to generalize effectively, while capturing the subtle differences in beehive acoustic patterns.

Table 4: Class-wise precision, recall, and F1-score of the final CNN model trained on cochleagram representations for multiclass beehive state classification. The model demonstrates consistently high performance across all four hive states.

| Class                    | Precision | Recall | F1-Score | Support |
|--------------------------|-----------|--------|----------|---------|
| Queen not present        | 0.98      | 0.99   | 0.99     | 549     |
| $Q_P$ and newly accepted | 0.99      | 0.97   | 0.98     | 627     |
| $Q_P$ and rejected       | 0.97      | 0.97   | 0.97     | 583     |
| $Q_P$ or original queen  | 0.99      | 1.00   | 0.99     | 609     |
| Accuracy                 |           |        | 0.98     | 2368    |
| Macro avg                | 0.98      | 0.98   | 0.98     | 2368    |
| Weighted avg             | 0.98      | 0.98   | 0.98     | 2368    |

#### 4.2 Model Optimization

Subsequent to enhancing its performance, the model required 21.82 MB of storage space. Then we simplified the network by optimizing it using optimization techniques such as pruning, knowledge distillation and Quantization. At first we converted the RGB cochleagram time-series representation to grayscale representation and change the cochleagram-cnn input from 3 dimensional input to single dimensional input, which led to a marginal size reduction but yielded a 0.1% improvement in model accuracy after calibration, indicating that grayscale input had minimal impact on performance while maintaining similar size and inference time.

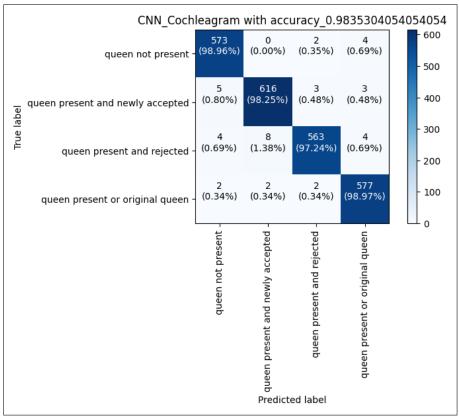


Fig. 8: Confusion matrix of the best-performing CNN model trained on cochleagram representations for beehive state classification. The model achieves an overall accuracy of 98.35%, with high precision and minimal misclassification across all four hive state categories

Subsequent pruning including neuron and layer pruning reduce the size of the model 48.73% and 61.63% respectively and reduce the overall size from 21.81 mb to 4.29 mb which is 80.33% reduction in the size and 41.67% reduction in inference time from 5.88 sec to 3.43 sec, while maintaining a reasonable accuracy of 97.00%.

Further gains were observed through knowledge distillation and quantization, where model was compressed to just  $2.50~\mathrm{MB}$  with a modest accuracy of 97.09% and a significantly faster inference time of  $2.13~\mathrm{seconds}$ . And quantization reduced the size to  $1.79~\mathrm{MB}$  and inference time to  $1.91~\mathrm{seconds}$ , exhibiting a mere 1.3% reduction in accuracy when benchmarked against the original

We successfully compressed the model 91.8% reduced it's size from 21.82 mb to 1.79 mb and accelerated the inference by 66.4% without losing much of the overall competency of the model. The Table 5 summarizes the model's overall efficacy on multiple stages of optimization considering accuracy, size, inference time and number of parameters.

Table 5: Comparison of the baseline CNN model and its optimized variants in terms of classification accuracy, model size, inference time, and number of parameters. Optimizations include grayscale preprocessing, pruning, and distillation. For the quantized model, only the final classification layer was quantized to reduce memory usage while maintaining performance.

| Model                   | Accuracy (%) | Size (MB) | Time (s) | Parameters |
|-------------------------|--------------|-----------|----------|------------|
| CNN_Tensorflow          | 98.31        | 21.82     | 5.69     | 5,719,690  |
| $Grayscale\_Tensorflow$ | 98.41        | 21.81     | 5.88     | 5,717,962  |
| Neuron Pruned           | 96.75        | 11.18     | 7.20     | 2,858,224  |
| Layer Pruned            | 97.00        | 4.29      | 3.43     | 1,094,952  |
| Distilled Model         | 97.09        | 2.50      | 2.13     | 651,404    |
| Quantized Model         | 97.00        | 1.79      | 1.91     | 651,404    |

Figure 9 shows how different optimization stages affect model size, parameter count, inference time, and classification accuracy. In Figure 9a, the baseline CNN\_TF model is 21.82 MB. This size stays the same after grayscale preprocessing but drops significantly to 11.18 MB after neuron pruning and to 4.29 MB after layer pruning. Knowledge distillation then reduces the size to 2.50 MB, and partial quantization compresses it further to 1.79 MB. This represents a total size reduction of 91.8% compared to the baseline.

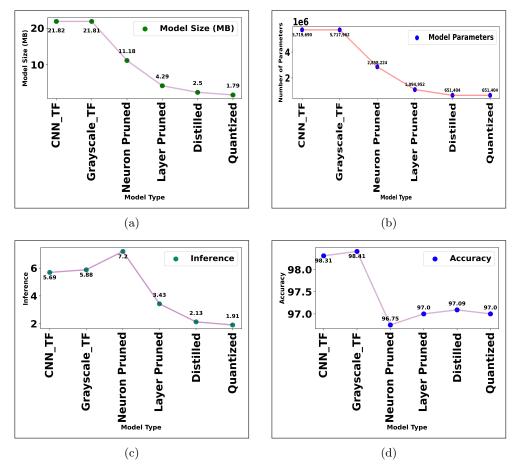
Figure 9b shows a similar trend in parameter count. The baseline starts with 5.72 million parameters, which decreases to 2.85 million after neuron pruning and to 1.49 million after layer pruning. Distillation creates a smaller architecture with 651,000 parameters, but there is no further reduction from quantization since only the classification layer weights are quantized.

In terms of inference time (Figure 9c), the baseline model takes 5.69 seconds. This increases slightly to 5.88 seconds after grayscale conversion and peaks at 7.20 seconds after neuron pruning due to irregular sparsity overhead. Layer pruning improves this by cutting inference time to 3.43 seconds. Distillation and quantization further reduce the inference time to 2.13 seconds and 1.91 seconds, respectively, achieving a 66.4% reduction in inference time compared to the baseline model.

Lastly, Figure 9d shows the accuracy trade-offs. The baseline accuracy is 98.31%, which slightly improves to 98.41% after grayscale conversion but falls to 96.75% after neuron pruning. Layer pruning brings performance back to 97.0%, with distillation achieving 97.09% and quantization maintaining 97.0%. This shows that significant compression and runtime improvements can be reached with only a 1.3% absolute accuracy loss compared to the uncompressed baseline model.

The confusion matrix of the final optimized model is shown in "Fig. 10", revealing the model's stable and balanced classification results across all categories.

These results highlight the superior trade-off achieved through optimizing using techniques like pruning, knowledge distillation and quantization, offering substantial reductions in memory and latency without significant loss in classification performance.



**Fig. 9**: (a)Optimization Stages vs Network's Size (in MB) (b)Optimization Stages vs Number of Parameters (c) Optimization Stages vs inference time (in seconds) (d) Optimization Stages vs Accuracy

#### 5 Discussion

In this research the effectiveness of CNNs in classifying complex bioacoustic signals to monitor beehive states has been shown, and also the importance of time-series representation of an audio signal. By evaluating the performance of four varied time–frequency image representation methods, including Spectrogram and Mel-Spectrogram, Smoothed-Spectrogram, and Cochleagram with CNN, we identified that the Cochleagram as the most informative and suitable representation for CNN-based classification scenarios. The final model produced an impressive accuracy of 98.31% on unseen test dataset, after subsequent refinements including network architectural enhancements and data augmentation on raw audio data.

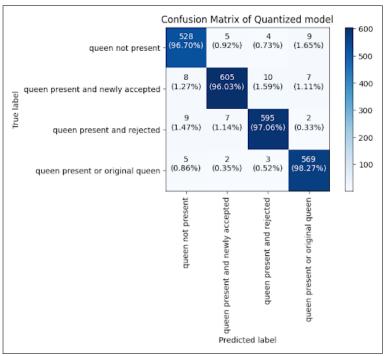


Fig. 10: Confusion matrix of the final optimized CNN model for beehive state classification. The model was compressed using pruning, knowledge distillation, and quantization of the classification layer. It maintains high predictive accuracy across all four classes with minimal misclassification.

Our findings align with the previous researches, that suggest the importance of time-series image representation of for beehive classification using CNN. However none of them explored or demonstrated the superior performance of the Cochleagram time-series representation in beehive classification task. Previous studies were largely focused on binary classification (e.g., queen present vs. not present) and relied on representations such as MFCCs or Mel-Spectrograms.

In contrast, our research shows the feasibility and advantages of cochleagram time-series image representation which improved the CNN performance in multiclass-classification. These four class (Queen Not Present, Queen Present and Newly Accepted, Queen Present and Rejected, Queen Present or Original Queen) are both practical and beneficial in beehive monitoring. This more detailed classification provides deeper insights of the hive and has the potential to significantly improve real-time beehive monitoring.

This paper also addresses one of the critical barrier to real-time beehive monitoring i.e model size and inference time. Although Deep Convolutional Neural Network models often shows high classification accuracy, their practical application in field settings is limited due to computational and memory constraints which also makes the hardware expensive. Though the optimization techniques such as pruning, knowledge

distillation and quantization, we reduce the CNN model size by 91.8% and reduce the inference time by 66.4% with only marginal degradation in performance (final accuracy of 97.00%), which reduce the dependency of high end hardware and also reduces the cost of the overall system. This highlights a favorable trade-off between computational efficiency and predictive power.

Notably, the evaluation metrics such as confusion matrices and classification report which provides the class-wise evaluation, shows that the model's balanced performance across all four beehive states, with majority of misclassifications occurring between acoustically similar states. This indicates the model's capability to capture nuanced acoustic variations in hive behavior, an essential aspect for accurate hive health monitoring.

An identified shortcoming in this study is the utilization of a single publicly available dataset collected from a specific geographic region. This may impact the generalizability of the model to hives under different environmental conditions or bee species. Future work can address this by incorporating diverse datasets and exploring domain adaptation techniques. Moreover, real-time deployment and field testing are needed to evaluate robustness under operational constraints such as background noise, hardware limitations, and temporal drifts in data.

In summary, this research not only advances the state of audio-based beehive monitoring but also sets a practical foundation for scalable, real-time applications through lightweight CNN architectures.

#### 6 Conclusion

This research focuses on a holistic method of classifying beehives states based on Convolutional Neural Networks (CNNs) applied to bioacoustic signals. Through a systematic study of four time–frequency image representations, we found that the Cochleagram time-series representation is the most suitable input format to facilitate CNN performance using bioacoustic data. This representation enabled a successful classification model for beehive multi state classification, while in previous research only the binary state classification was applied.

The proposed model outperformed prior approaches with an accuracy of 98.31% on the testing samples with a strong generalization across all the four classes or states. In addition, we addressed the major obstacles encountered during deployment of models and real-time monitoring —size and inference latency—by applying optimization techniques involving pruning, knowledge distillation, and quantization. With these optimizations, the size and inference time of the model was reduced by 91.8% and 66.4% respectively, while maintaining high performance. The optimized model exhibited only a minor accuracy drop of 1.33%, achieving 97.00% accuracy on unseen data, thus allowing the model to be deployed to real-time and resource-limited scenarios.

This research enhances not only extending past the contemporary state-of-the-art approaches in hive monitoring, but also provides a deployable, scalable and cost-effective solution for both researchers and beekeepers. The methodology described here can be applied to other environmental or agricultural audio classification tasks.

Future work should verify the model's efficacy on more diverse datasets and under real-time/real-world conditions aiming to improve the robustness and generalization of the model. Incorporating external factors such as temperature, humidity, and weather data, which could further enhance the classification performance and practical utility.

#### 7 Acknowledgements

The author(s) acknowledge and are thankful to Council of Scientific and Industrial Research–Central Scientific Instruments Organisation (CSIR–CSIO), Sector 30, Chandigarh, India for provind technical support for conducting the study under the project GAP00492.

#### **Declarations**

#### • Data availability

The dataset used in this study is publicly available at: https://www.kaggle.com/datasets/annajyang/beehive-sounds.

#### • Materials availability

The computational experiments were conducted on a local machine equipped with an Intel Core i9 (12th Gen) processor, 32 GB RAM, and an octa-core CPU. Python 3.9 was used for implementation, along with open-source libraries such as Tensor-Flow, Keras, scikit-learn, librosa for audio processing, and Matplotlib, Seaborn, Google Sheets for visualizations. No specialized hardware or proprietary software was used.

#### • Source code availability

The source code available at https://github.com/CSIO-FPIL/beehive-classification.

#### • Author contribution

Harshit: Conceptualization, Methodology, Experimentation, Writing – original draft, Visualization. Rahul Jana: Discussion, Writing – review & editing. Dr. Ritesh Kumar: Supervision.

#### References

- [1] Khalifa, S.A.M., Elshafiey, E.H., Shetaia, A.A., El-Wahed, A.A.A., Algethami, A.F., Musharraf, S.G., AlAjmi, M.F., Zhao, C., Masry, S.H.D., Abdel-Daim, M.M., Halabi, M.F., Kai, G., Al Naggar, Y., Bishr, M., Diab, M.A.M., El-Seedi, H.R.: Overview of bee pollination and its economic value for crop production. Insects 12(8), 688 (2021) https://doi.org/10.3390/insects12080688 . PMID: 34442255; PMCID: PMC8396518
- [2] Shaara, F., Adgaba, N., Al-Ghamdi, A.: Current knowledge about behaviors of honey bee queens with highlighting of the importance future studies. The Journal of Basic and Applied Zoology 82 (2021) https://doi.org/10.1186/s41936-021-00234-x

- [3] Winston, M.L.: The Biology of the Honey Bee. Harvard University Press, Cambridge, MA (1991)
- [4] Cejrowski, T., Szymanski, J., Mora, H., Gil, D.: Detection of the Bee Queen Presence Using Sound Analysis, pp. 297–306 (2018). https://doi.org/10.1007/ 978-3-319-75420-8\_28
- [5] Abdollahi, M., Giovenazzo, P., Falk, T.H.: Automated beehive acoustics monitoring: A comprehensive review of the literature and recommendations for future work. Applied Sciences 12(8), 3920 (2022) https://doi.org/10.3390/app12083920
- [6] Zacepins, A., Kviesis, A., Stalidzans, E., Liepniece, M., Meitalovs, J.: Remote detection of the swarming of honey bee colonies by single-point temperature monitoring. Biosystems Engineering 148, 76–80 (2016) https://doi.org/10.1016/ j.biosystemseng.2016.05.012
- [7] Liu, X., Deng, Z., Yang, Y.: Recent progress in semantic image segmentation. Artificial Intelligence Review **52**, 1089–1106 (2019) https://doi.org/10.1007/s10462-018-9641-3
- [8] Cetin, R., Gecgel, S., Kurt, G.K., Baskaya, F.: Convolutional neural network-based signal classification in real time. IEEE Embedded Systems Letters 13(4), 186–189 (2021) https://doi.org/10.1109/LES.2021.3049731
- [9] Haran, R.V., Xiong, H., Berkovsky, S.: Benchmarking audio signal representation techniques for classification with convolutional neural networks. Sensors (Basel) 21(10), 3434 (2021) https://doi.org/10.3390/s21103434. PMID: 34069189; PMCID: PMC8156023
- [10] Ferreira-Paiva, L., Alfaro Espinoza, E., Martins Almeida, V., Felix, L., Neves, R.: A survey of data augmentation for audio classification, vol. 3 (2022). https://doi.org/10.20906/CBA2022/3469
- [11] Nolasco, I., Terenzi, A., Cecchi, S., Orcioni, S., Bear, H.L., Benetos, E.: Audio-based identification of beehive states. In: 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, pp. 8256–8260 (2019). https://doi.org/10.1109/ICASSP.2019.8682981
- [12] Labonno, S., Mostafa, S., Akhter, S.: Identify The Beehive Sound Using Deep Learning. Preprint at https://arxiv.org/abs/2209.01374 (2022). https://doi.org/10.48550/arXiv.2209.01374
- [13] Soares, B., Luz, J., Macêdo, V., Silva, R., Araújo, F., Magalhães, D.: Mfcc-based descriptor for bee queen presence detection. Expert Systems with Applications 201, 117104 (2022) https://doi.org/10.1016/j.eswa.2022.117104
- [14] Nolasco, I., Benetos, E.: To bee or not to bee: Investigating machine learning

- approaches for beehive sound recognition. In: 2018 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), pp. 133–137 (2018)
- [15] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115(3), 211–252 (2015)
- [16] He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. Preprint at https://arxiv.org/abs/1512.03385 (2015). https://doi.org/10.48550/arXiv.1512.03385
- [17] Zhang, X., Zou, J., He, K., Sun, J.: Accelerating very deep convolutional networks for classification and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 38(10), 1943–1955 (2015) https://doi.org/10.1109/TPAMI. 2015.2502579
- [18] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (2016). https://doi.org/10.1109/CVPR.2016.308
- [19] Koul, N., Manvi, S.: A scheme for feature selection from gene expression data using recursive feature elimination with cross validation and unsupervised deep belief network classifier, pp. 31–36 (2019). https://doi.org/10.1109/ICCCT2.2019.8824943
- [20] Yang, A.: Smart Bee Colony Monitor: Clips of Beehive Sounds. Kaggle (2022). https://doi.org/10.34740/KAGGLE/DSV/4451415 . https://www.kaggle.com/dsv/4451415
- [21] Ghosh, S., Das, N., Nasipuri, M.: Reshaping inputs for convolutional neural network: Some common and uncommon methods. Pattern Recognition 93, 79–94 (2019) https://doi.org/10.1016/j.patcog.2019.04.009
- [22] McFee, B., Raffel, C., Liang, D., Ellis, D.P.W., McVicar, M., Battenberg, E., Nieto, O.: librosa: Audio and music signal analysis in python. In: Huff, K., Bergstra, J. (eds.) Proceedings of the 14th Python in Science Conference (SciPy), pp. 18–25 (2015). SciPy. https://conference.scipy.org/proceedings/scipy2015
- [23] Heeris, J.: Gammatone Filterbank Toolkit: Utilities for Analysing Sound Using Perceptual Models of Human Hearing. https://github.com/detly/gammatone. Accessed: 2025-05-30 (2013)
- [24] Oppenheim, A.V., Schafer, R.W.: Discrete-Time Signal Processing, 2nd edn. Pearson Education Signal Processing Series, p. 864. Pearson Education India, India (1999)

- [25] Smith, J.O.: Mathematics of the Discrete Fourier Transform (DFT). W3K Publishing, http://www.w3k.org/books/ (2007)
- [26] Müller, M.: Fundamentals of Music Processing, p. 487 (2015). https://doi.org/ 10.1007/978-3-319-21945-5
- [27] Harris, F.J.: On the use of windows for harmonic analysis with the discrete fourier transform. Proceedings of the IEEE **66**(1), 51–83 (1978) https://doi.org/10.1109/PROC.1978.10837
- [28] Oppenheim, A.V., Schafer, R.W.: Discrete-Time Signal Processing, 3rd edn. Pearson, Upper Saddle River, NJ (2010)
- [29] Fayek, H.M.: Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between (2016). https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html
- [30] O'Shaughnessy, D.: Speech Communications: Human And Machine (ieee). Universities Press (India) Pvt. Limited. https://books.google.co.in/books?id=UWmD8aY\_448C
- [31] Patterson, R.D., Robinson, K., Holdsworth, J., McKeown, D., Zhang, C., Allerhand, M.: Complex sounds and auditory images. In: Cazals, Y., Horner, K., Demany, L. (eds.) Auditory Physiology and Perception, pp. 429–446. Pergamon, Oxford, UK (1992)
- [32] Bishop, C.: Pattern Recognition and Machine Learning, vol. 16, pp. 140–155 (2006). https://doi.org/10.1117/1.2819119
- [33] Mohamed, A.-r.R., Dahl, G.E., Hinton, G.: Acoustic modeling using deep belief networks. IEEE Transactions on Audio, Speech, and Language Processing **20**(1), 14–22 (2012)
- [34] Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing & Management 45(4), 427–437 (2009)
- [35] Powers, D.M.W.: Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. Journal of Machine Learning Technologies 2(1), 37–63 (2020)
- [36] Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. In: Advances in Neural Information Processing Systems, vol. 28 (2015). https://arxiv.org/abs/1506.02626
- [37] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proceedings of

- the IEEE International Conference on Computer Vision (ICCV) (2017). https://arxiv.org/abs/1708.06519
- [38] He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017). https://arxiv.org/abs/1707.06168
- [39] Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
- [40] Nagel, M., Amjad, R., Baalen, M., Fournarakis, M., Bondarenko, Y., Blankevoort, T.: A white paper on neural network quantization. arXiv preprint arXiv:2106.08295 (2021)
- [41] Hubara, I., Nahshan, Y., Hanani, R., Banner, R., Soudry, D.: Improving post training neural quantization: Layer-wise calibration and integer programming. arXiv preprint arXiv:2006.10518 (2020)
- [42] Modi, I.: From Binary to Computation: How CPUs and GPUs Understand Numbers. https://medium.com/@ishan.modi24/from-binary-to-computation-how-cpus-and-gpus-understand-numbers-516a99f026fc. Medium article (2022)
- [43] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2704–2713 (2018). https://arxiv.org/abs/1712.05877
- [44] Krishnamoorthi, R.: Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342 (2018)
- [45] Wu, J., Li, J., Lin, Y., Li, F.: Integer quantization for deep learning inference: Principles and empirical evaluation. arXiv preprint arXiv:2004.09602 (2020)

# Appendix A Detailed Architectures of Teacher and Student Models with Distillation Pseudocode

This appendix provides additional visual and algorithmic details related to the knowledge distillation optimization process discussed in Section 3.5.2.

| Layer                           | Teacher Model  | Student Model   |
|---------------------------------|--|---|
| Input Layer                     | 64 × 64 × 1  | 64 × 64 × 1   |
| Feature<br>Extraction<br>Layers | ConvBlock 1: Conv2d: 1 input channel, 48 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 48 features Max Pooling: 2x2 kernel, stride of 2 ReLU Activation ConvBlock 2: Conv2d: 48 input channels, 96 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 96 features Max Pooling: 2x2 kernel, stride of 2 ReLU Activation ConvBlock 3: Conv2d: 96 input channels, 192 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 192 features Max Pooling: 2x2 kernel, stride of 2 ReLU Activation ConvBlock 4: Conv2d: 192 input channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 192 input channels, 3x4 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 384 features Max Pooling: 2x2 kernel, stride of 2 ReLU Activation | ConvBlock 1: Conv2d: 1 input channel, 32 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 32 features Max Pooling: 2x2 kernel, stride of 2 ReLU Activation ConvBlock 2: Conv2d: 32 input channels, 64 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 64 features Max Pooling: 2x2 kernel, stride of 2 ReLU Activation ConvBlock 3: Conv2d: 64 input channels, 128 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 128 features Max Pooling: 2x2 kernel, stride of 2 ReLU Activation ConvBlock 4: Conv2d: 128 input channels, 256 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 128 input channels, 256 output channels, 3x3 kernel, padding of 1 ReLU Activation BatchNorm2d: 256 features Max Pooling: 2x2 kernel, stride of 2 ReLU Activation |
| Classification<br>Layers        | Dropout: Probability of 0.55 Flatten Layer LinearBatchNorm 1: Linear: 384 * 4 * 4 input features, 36 output features BatchNorm1d: 36 features Dropout: Probability of 0.55 LinearBatchNorm 2: Linear: 36 input features, 4 output features BatchNorm1d: 4 features   | Dropout: Probability of 0.5 Flatten<br>Layer StudentLinearBatchNorm 1:<br>Linear: 256 * 4 * 4 input features,<br>64 output features BatchNorm1d: 64<br>features Dropout: Probability of 0.5<br>StudentLinearBatchNorm 2: Linear:<br>64 input features, 4 output features<br>BatchNorm1d: 4 features   |

Fig. A1: Network Architecture of Teacher and Student Model

#### Algorithm 1 Knowledge Distillation Training Procedure

```
Inputs:
 T_Model: Teacher model (pre-trained)
                                                       S_Model: Student model (untrained)
 D_{-}train: Training dataset
                                                       \eta: Number of training epochs
 \Gamma: Temperature for softening probabilities
                                                       \alpha: Weight for distillation loss
 \beta: Weight for ground truth loss
                                                       optimizer: Optimization algorithm
                                                       (e.g., SGD, Adam)
 distill_loss: Cross-entropy loss between
                                                       gt\_loss: Cross-entropy loss between
 student and teacher soft targets
                                                       student predictions and true labels
 1: function FORWARDPASS(Model, X)
        logits \leftarrow Model(X)
        return logits
 3:
 4: end function
    function SoftmaxWithTemperature(logits, \Gamma)
        \begin{array}{l} \text{exp\_logits} \leftarrow \exp(\frac{logits}{\Gamma}) \\ \text{probs} \leftarrow \frac{exp\_logits}{\sum exp\_logits} \end{array}
 6:
 7:
        return probs
 8:
    end function
    function CrossEntropyLoss(pred_probs, true_probs)
10:
11:
        loss \leftarrow -\sum (true\_probs \cdot log(pred\_probs))
        return loss
12:
    end function
13:
    \mathbf{function}\ \mathrm{DistillationLoss}(S\_\mathrm{probs},\ T\_\mathrm{probs})
14:
        distill\_loss \leftarrow CrossEntropyLoss(S\_probs, T\_probs)
15:
        return distill_loss
16:
17: end function
    function CalculateTotalLoss(S_logits, y, T_probs, \alpha, \beta, \Gamma)
18:
        S-probs_distill \leftarrow SoftmaxWithTemperature(S_logits, \Gamma)
19:
        distill\_loss \leftarrow Distill\_ationLoss(S\_probs\_distill, T\_probs)
20:
21:
        S_{probs\_gt} \leftarrow SoftmaxWithTemperature(S_{logits}, 1)
        gt\_loss \leftarrow CrossEntropyLoss(S\_probs\_gt, y)
22:
        total_loss \leftarrow \alpha \cdot \text{distill_loss} + \beta \cdot \text{gt_loss}
23
        return total_loss
24:
    end function
25:
    for epoch = 1 to \eta do
26:
        for each batch (X, y) in D-train do
27:
             with no gradient:
28:
             T\_logits \leftarrow ForwardPass(T\_Model, X)
29:
             T-probs \leftarrow SoftmaxWithTemperature(T_logits, \Gamma)
30:
             S\_logits \leftarrow ForwardPass(S\_Model, X)
31:
             total_loss \leftarrow CalculateTotalLoss(S_logits, y, T_probs, \alpha, \beta, \Gamma)
32:
             optimizer.zero_grad()
33:
             total_loss.backward()
34:
             optimizer.step()
35:
        end for
37: end for
```