Journal of Data Analysis and Information Processing, 2022, 10, 198-219
https://www.scirp.org/journal/jdaip
ISSN Online: 2327-7203
ISSN Print: 2327-7211

Autonomous Surveillance of Infants' Needs Using CNN Model for Audio Cry Classification

Geofrey Owino*, Anthony Waititu, Anthony Wanjoya, John Okwiri

Department of Statistics and Actuarial Sciences, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya
Email: *jeffowino97@gmail.com, agwaititu@gmail.com, awanjoya@gmail.com, johnmitch254@gmail.com

How to cite this paper: Owino, G., Waititu, A., Wanjoya, A. and Okwiri, J. (2022) Autonomous Surveillance of Infants' Needs Using CNN Model for Audio Cry Classification. *Journal of Data Analysis and Information Processing*, **10**, 198-219.
https://doi.org/10.4236/jdaip.2022.104013

Received: October 10, 2022
Accepted: November 27, 2022
Published: November 30, 2022

Abstract

Infants portray suggestive unique cries while sick, having belly pain, discomfort, tiredness, attention and desire for a change of diapers among other needs. There exists limited knowledge in accessing the infants' needs as they only relay information through suggestive cries. Many teenagers tend to give birth at an early age, thereby exposing them to be the key monitors of their own babies. They tend not to have sufficient skills in monitoring the infant's dire needs, more so during the early stages of infant development. Artificial intelligence has shown promising efficient predictive analytics from supervised, and unsupervised to reinforcement learning models. This study, therefore, seeks to develop an android app that could be used to discriminate the infant audio cries by leveraging the strength of convolution neural networks as a classifier model. Audio analytics from many kinds of literature is an untapped area by researchers as it's attributed to messy and huge data generation. This study, therefore, strongly leverages convolution neural networks, a deep learning model that is capable of handling more than one-dimensional datasets. To achieve this, the audio data in form of a wave was converted to images through Mel spectrum frequencies which were classified using the computer vision CNN model. The Librosa library was used to convert the audio to Mel spectrum which was then presented as pixels serving as the input for classifying the audio classes such as sick, burping, tired, and hungry. The study goal was to incorporate the model as an android tool that can be utilized at the domestic level and hospital facilities for surveillance of the infant's health and social needs status all time round.

Keywords

Convolutional Neural Network (CNN), Mel Frequency Cepstral Coefficients (MFCCs), Rectified Linear Unit (ReLU), Activation Function, Audio Analytics, Deep Neural Network (DNN)

1. Introduction

Every year, millions of babies are born across the globe. The challenge of taking care of these babies has been highlighted to be tougher than the actual process of birthing. The study has emphasized this difficulty, especially for first-time parents. When babies are born, their only means of communication is through crying. Failure to understand the meaning of infant cries has proved to be the biggest challenge in taking care of infants.

According to the report released by Ely and Driscoll, 47% of infants lose their lives due to misunderstandings between caregivers and infants [1]. Globally, there are approximately 6700 newborn deaths every day. This has been attributed to the minimal chances of infant survival due to misinterpretation of their needs. According to the literature, there exists an untapped gap in communication between infants and caregivers. The language that infants speak primarily relies on suggestive cries. The misunderstanding of the infants' needs has escalated more complex misfortunes for the infants. Take for example a situation when an infant needs a diaper change, and the caregivers insist on giving them more food. In fact, the misunderstanding of the infants' needs not only leads to a higher rise of infant mortalities, but also encroaches on the rights and well-being of the infants.

Previous research looked to use DNNs in modeling audio classification. When DNNs are used, the spatial topology of the input which is an important attribute is disregarded. When these spatial correlations are not utilized, they inherently don't model the input topology causing the network to be dense hence increasing computation and complexity [2]. Moreover, previous studies specifically selected audio recordings under a conducive balance of external sounds and with a short recording duration. This is never the case under normal circumstances. Therefore, these studies outlined inconsistencies with the algorithms.

This study, therefore, potentially comes in to solve these uncertainties in misunderstanding the infants' needs by leveraging the power of convolution neural networks to classify the suggestive cries. Moreover, this study also embraces the sustainable development goal and vision 2030 for Kenya in particular through the eradication of factors that compromise the infant's health.

Machine learning algorithms have been used to solve health problems in the past decade, but very few scientists have presented their solutions in the area of audio analytics. Perhaps the main reason for modelling audio analytics is the black-box nature of the deep learning algorithms and the complexities of extracting features from audio data. The computer vision based on deep learning is really intense based on computational power and deployment. This study, therefore, leverages the skills of statistics, mathematics and computer science to solve the menace of the limited understanding of infant language through the deployment of a simple, accessible android infant auditory monitoring application.

2. Background

Research on infant cries started as early as the 1960s when the four types of cries—hunger, pain, pleasure, and birth were auditorily identified by a research group known as Wasz-Hockert [3]. Researchers determined that trained adult listeners could auditorily differentiate the types of infant cries. However, training of human perception for infant cry takes a longer time and is more difficult than training a machine learning model. According to Mukhopadhyay's study, 33.09% is the highest classification accuracy that can be achieved when a group of people is trained to recognize some cry sound.

Studies regarding infant crying in the past decades have narrowed down to two main perspectives, that is, classification or interpretation of cries and detection or recognition of cries through audio signals. Some studies tried to identify and interpret cries associated with certain statuses based on acoustic signals. In a study conducted by Ntalampiras, he experimented with various machine learning algorithms, that is, Random Forest, Support Vector Machine, Reservoir Network, and Multi-layer Perceptron with the aim of using audio-based features to classify pathological states of cry [4]. The audio-based features included Mel frequency cepstral coefficients and temporal modulation features. In his report, the Random Forest classifier had the best rate performance of 84.5%. In addition to that, Al-Azzawi aiming to detect infant cry based on various physiological statuses, proposed an automatic system that used fuzzy transform to extract features and then feed them into a Multi-Layer Perceptron (MLP) Artificial Neural Network (ANN).

Even so, in the past years, researchers have applied different methods with the aim of achieving robust auditory classification to overcome the primary environmental and demographic obstacles. Techniques such as the HMM, matrix factorization, Hough transform and-vector restricted DNNs and CNNs have been used within the domain of classifying audio. Researchers have also experimented with classifying audio by representing the audio in form of spectrograms [5].

Recent works revolving around the use of CNN for audio classification have shown improved efficiency. CNNs with spectrograms for audio classification have accounted for spatial differences in their input by using locally connected sparse structures to improve accuracy. It can model frequency and time components between adjacent audio samples. Thus, CNN's have outperformed DNNs in modeling audio classification systems.

Summary of Research Gaps from Previous Studies

In theory, GMM-HMMs can complete a gamut of precision to model probabilistic distribution. Consequently, to enhance speed and accuracy the focus has been on constraining GMMs. As a result, a study shows that with the adoption of computing systems with high processing speeds, DNNs have proven to be better than GMMs in modeling audio recognition systems [6]. However, literature shows that DNNs have significant drawbacks. First, the spatial topology of the input which is an important attribute is disregarded by DNNs. Audio is normally constituted of a strong correlations structure with respect to its frequency domain. When DNNs are used, these spatial correlations are not utilized since they inherently don't model the input topology. Furthermore, DNNs are affected by translational variances in audio signals. Although translational invariance can be achieved with an adequate number of parameters in the DNN architecture, the network would be dense. As such, complexity and computation would dramatically increase.

Most of the studies specifically selected audio recordings under a conducive balance of external sounds and with a short recording duration [7]. Under normal circumstances, this is never the case. External noise such as car engines, music, and parents talking, can present a combination of cries. The placement of the microphone position may also vary depending on the parent's personal preference or even the layout of the baby's room. As such, these studies outlined inconsistencies with the algorithms described in the above studies and hence may not be optimal for the continuous monitoring of the baby in real-life applications. Moreover, many studies have suggested numerous models that have never been put on production.

This study, therefore, tends to offer solution to the following existing problems:

1) Bridging the communication gap between infants and caregivers.

2) Reduction of infant mortality rate through minimization of infant health misdiagnosis.

3) Provision of simple, accessible and affordable infant care tool as compared with traditional way of consulting pediatrician.

3. Methodology

Sound is a function of amplitude and time with parameters such as frequency, bandwidth and decibel.

Secondary audio dataset having wave extension format was obtained from Coswara database and was combined with the secondary datasets provided by the UNICEF. The pediatrician experts collected the audio dataset and accurately classified the files as either hungry, tired, sick or burping. All the secondary audio files were recorded from strictly infants of the age 0 to 19 months of age as shown in Figure 1 below.

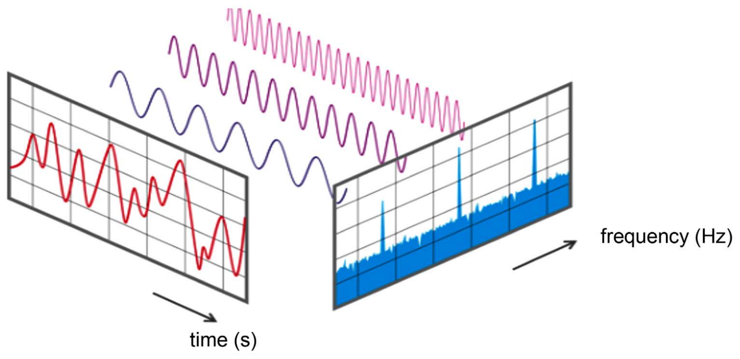


Figure 1. Representation of the audio record (Image source KNuggets Blog).

3.1. Target Population

The study targeted only infants aged between zero to 19 months with an exclusion criterion of any infant beyond 3 years of age.

3.2. Analytical and Data Preprocessing Tools

Python programming is a high-level open-source general purpose programming language with myriad of libraries which has promised efficient machine learning models until their stage of production. Numerical array (NumPy) library was used to manipulate the matrices during image processing.

Sckitlearn is a library in python that specifically was built for preprocessing of the input data. It was used in this study to randomly split the datasets into testing, training and validations tests.

3.3. Study Variables

The study variables were the recorded infants cry audio files from the four main classes that the study targeted. The main aim of this study was to come up with a classifier deep learning that could be used to discriminate the audio. The main classes were:

- Burping.
- Hungry.
- Tired.
- Sick.

Below (Figure 2) is a pictorial view of some of the main classes.

Figure 2. Image of infant audio classes, source (Child Maltreat, 2016, 21, 327-342).

3.4. Data Preprocessing

Data preprocessing plays a fundamental role in modelling as such, garbage in garbage out could prevail thus posing a threat to the consumption of the model.

The Audio data was read into the python environment using the free Graphic Processing Unit (GPU) that's provided by Google collaboratory environment. This accelerated the processing power and run time as compared with the local memory environments. However, since the model was expected to discriminate the infant audio from the real-world environment where there will be acoustics noise, the study intentionally included background noise as another class. The study took this into considerations as the expected background noise could assume any sound disturbances like spatting of rains, hooting of vehicles and more.

3.4.1. Resizing the Audio into the Same Length

All audio records were then converted into 6 seconds length by either extending the duration through padding or by truncating longer records. Augmentation of the time shift was done to have the records ready for Mel spectrum generations.

3.4.2. Generation of Mel Spectrum

The augmented data was then transformed to images, a Mel spectrum frequency that was modelled by CNN model. The Mel spectrum was achieved from Fourier transformation of the audio sounds frequency. The more granular form of MFCC was, therefore, achieved by the cosine transformation of the signals. This was very necessary as the Mel spectrum captures the essential features of the audio, goes beyond what human may not have perceived from different audio class based on the distinction points. This was the key first step in transformation of the audio to consumable format that was the raw input for CNN model. Some of the audio files were expected to be mono (one audio channel) while others were stereo having two audio channels. All the audios were, therefore, converted to stereo channels (Figure 3).

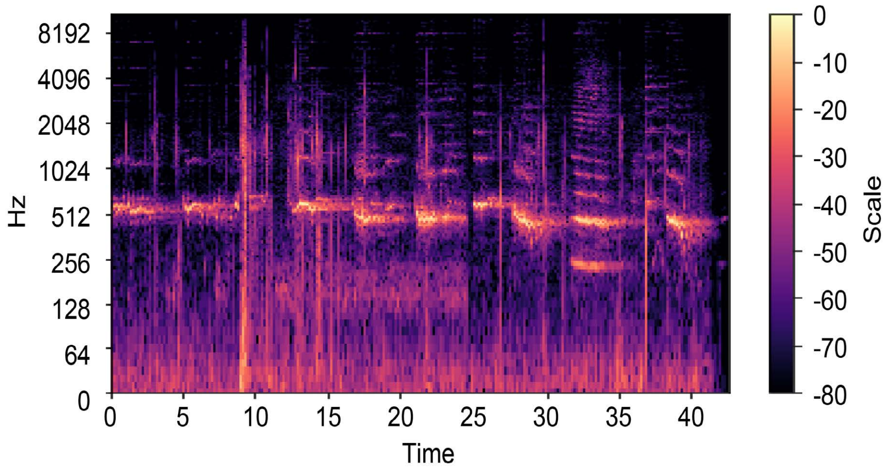


Figure 3. Mel spectrum image of a transformed audio signal.

3.4.3. Mel Scale Augmentation: Time and Frequency Masking

The Mel spectrum was augmented once more based on the time and frequency masking. The transformed audio records were ready to be modelled by the deep learning models as shown in Figure 4.

From the Mel spectrum, the audio records were fully transformed to images ready for computer vision algorithms. The realized input datasets inform of spectrum was in a 3 dimension of frequency, time and channel.

The files were divided into training, testing and validation test. To better have the most accurate model and asses the aspects of overfitting, a new dataset that the model has never seen was used to run the model again. The validation set was used to validate the model.

3.5. Deep Learning Models

Deep learning is a branch of machine learning models that tends to mimic the logical functionalities of the human brain. The models tend to find the unknown associations and patterns by employing logical structures. Deep learning models uses a series of multiple hidden layers as opposed to traditional generic neural network that only contains a few hidden layer [8]. The models are trained severally and maps the input data with the prior knowledge achieved through trainings using train datasets thus delivering accurate outputs. The ideological concepts that's pinning this novel innovation is similar with the biological neural networks which operates by comparing the new information with the already known patterns to arrive at the unknown information. Deep learning models are trained using very large, labeled datasets and initiate the feature learning.

The main drive to shift into the deep learning models was the limitations of the machine learning models that tends to plateau in performance during training with larger datasets and thereafter, the diminishing returns abruptly kicks in the gradients. This, therefore, interferes with the accuracy and precision

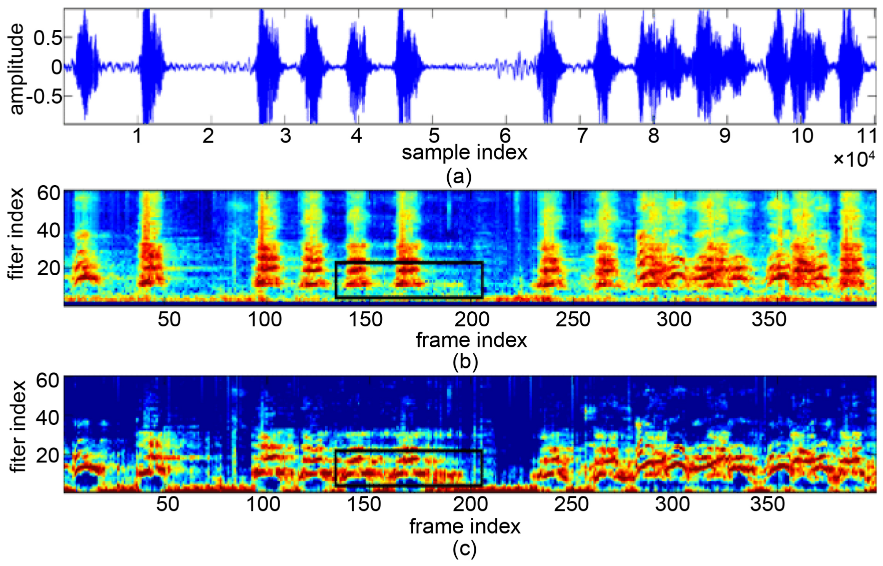


Figure 4. Full framework of signal wave conversion to Mel spectrum scale.

of the results. On the other hand, deep learning models also has stronger power for horizontal scaling and do not suffer from drift on productions, a case that many researchers have reported based on the machine learning models (**Figure 5**).

Deep learning models have gained a lot of popularities in the recent years based on their strengths in handling big datasets, revealing complex unknown relationship between the datasets. The models work well with the unstructured dataset.

3.5.1. Convolution Neural Network Deep Learning Model

Convolution Neural network in particular has been widely used in computer vision and problems that perhaps involves very large complex datasets that do not promise any clear statistical distributions. This study employs Convolution neural network to achieve an efficient auditory discrimination. Convolution neural networks are typical biological motivated architecture and have potentially stood out as one of the best deep learning models for computer visions. This study, therefore, leveraged on the potential feasibly state of art performance of Convolution neural networks efficient results.

3.5.2. Theoretical Architecture of CNN Model

The hidden convolution layers' parameters are made up of a series of learnable filters called kernels. CNN filters have varying dimensions which maps the number of channels in the input layer [9]. The RGB images and gray scale images are 3 and 1 respectively. The model uses pooling layers after convolution layers with the aim of reducing dimensions a process called down sampling.

The hyper parameters of the pooling layers may be referred to as filter size and strides. The CNN model utilizes the maximum pooling layers and average pooling layers as a way of achieving the dimension reductions or down sampling where the maximal and average value are used respectively [10]. Maximum pooling has been used in many applications of the CNN as it denotes a corresponding feature detection by large numbers.

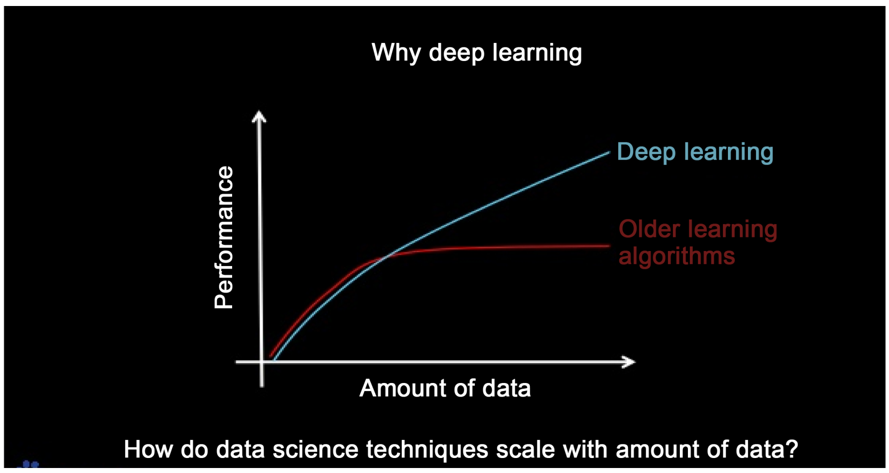


Figure 5. Image source Andrew NG why deep learning.

3.6. Architecture of CNN

1) *Convolution Layer*

The convolution layer is main fundamental part of CNN that is responsible for computational executions and loads. This layer performs a dot product of the two matrices that are learnable parameters which is also known as kernels. The CNN model is always spatially smaller than the image, but it is more in-depth feature that always aids in understanding the RGB images [11]. The kernel slides on height and width of the images thus producing an image that represents the receptive region thus realizing a two-dimensional image called activation.

2) *Pooling Layer*

The pooling layer tentatively represent the out of the network at different stages by either picking a maximum value or averaging the points thus arriving to the maximum pooling or average pooling. This really helps in attaining the down sampling through minimal spatial operations. The Pooling, therefore, is essential in enhancing a faster computational processing time and efficiency.

Several pooling techniques exist such as Average pooling, rectangular neighborhood and weighted pooling-based centrality of the image pixels. This study, therefore, employed the maximum pooling in particular to achieve the down sampling. The maximum poling, therefore, took care of drift when the model was deployed into production. At every stage, the model represents the extremal values derived from the primary pixel depth slice.

For the activation size of $W \times W \times D$, a pooling kernel of size F and S strides all the time, then the output size volume will be arrived by the following formula.

$$W_{out} = \left(W - F/S\right) + 1 \tag{3.1}$$

Pooling, therefore, provides translation invariance thus, an object will be detected regardless of the position it's located.

3) *Fully Connected Layers*

All neurons have got full connectivity between the preceding and succeeding layers. This, therefore, allows matrix computations together with bias effect. The main goal for the fully connected layers is to enhance mapping between the input and output layers.

4) *Non-Linearity Layers* (*Activation Functions*)

The convolution tends to operate linearly but the images may not be adequately mapped linearly. The nonlinear layers, therefore, come in and relaxe the assumption of linearity thus fitting the nonlinear image datasets well. This study used ReLU activation function. The Rectified Linear Unit is one of the most popular activation functions due to its considerable flexibility. The ReLU amplifies the convergences fasters as compared with sigmoid and Tanh functions.

The rectified Linear Unit (ReLU) is, therefore, a truncation performed individually for every element in the input. This layer does not change the size of the input. Its input comprises $m_1^{(l-1)}$ feature maps of size $m_2^{(l-1)} * m_3^{(l-1)}$ and the absolute value of each feature map is computed as:

$$Y_1^{(l)} = \left| Y_1^{(l)} \right| \tag{3.2}$$

It is computed pointwise such that the output consists of $m_1^{(l)} = m_1^{(l-1)}$ feature maps unchanged in size. It's represented as.

$$y_{i,j,d} = \max\left\{0, x_{i,j,d}^{l}\right\} \tag{3.3}$$

with $0 \le i < H^l = H^{l+1}$, $0 \le j < W^l = W^{l+1}$ and $0 \le d < D^l = D^{l+1}$. There is no parameter inside this layer hence no parameter learning. From equation above, it is obvious that:

$$\frac{\mathrm{d}y_{i,j,d}}{\mathrm{d}x_{i,j,d}^{l}} = \left[x_{i,j,d}^{l} > 0\right] \tag{3.4}$$

where is the indicator function, being 1 if its argument is true, and 0 otherwise. Hence, we have:

$$\left[\frac{\mathrm{d}z}{\mathrm{d}x^l}\right]_{i,j,d} = \left[\frac{\mathrm{d}z}{\mathrm{d}y}\right]_{i,j,d} \tag{3.5}$$

if $x'_{i,j,d} > 0$ and 0 otherwise. y is an alias of x^{l+1}.

The purpose of ReLU is to increase the non-linearity of the CNN. Since the semantic information in an image is obviously a highly nonlinear mapping of pixel values in the input, we want the mapping from CNN input to its output also be highly nonlinear. The ReLU function, although simple, is a nonlinear function.

3.7. Mathematical Structure of the CNN Model

A CNN usually takes an order 3 tensor as its input, but higher order tensor inputs can also be handled by CNN in a similar way [12]. The input then sequentially goes through a series of processing. One processing step is usually called a layer, which could be a convolution layer, a pooling layer, a normalization layer, a fully connected layer, a loss layer, etc.

$$x^1 \to \boxed{w^1} \to x^2 \to \cdots \to x^{L-1} \to \boxed{w^{L-1}} \to x^L \to w^L \to z \tag{3.6}$$

The input in this case is x^1 usually an image. It goes through the processing in the first layer, which is the first box. We denote the parameters involved in the first layer's processing collectively as a tensor w^1. The output of the first layer is x^2 which also acts as the input to the second layer processing.

This processing proceeds till all layers in the CNN has been finished, which outputs x^L.One additional layer, however, is added for backward error propagation, a method that learns good parameter values in the CNN. Let's suppose the problem at hand is an image classification problem with C classes. A commonly used strategy is to output x^L as a C dimensional vector, whose i^{th} entry encodes the prediction. To make x^L a probability mass function, we can set the processing in the $(L-1)^{\text{th}}$ layer as a soft max transformation of x^{L-1}. In other applications, the output x^L may have other forms and interpretations.

The last layer is a loss layer. Let us suppose it is the corresponding target value for the input x^1, then a cost or loss function can be used to measure the discrepancy between the CNN prediction x^L and the target *t*. A simple loss function could be:

$$\frac{1}{2}\left\|t-x^L\right\|^2 \tag{3.7}$$

Although more complex loss functions can be used this squared loss function is used in a regression problem. A cross entropy loss function is used or a classification problem.

3.7.1. Convolution

Assume an image to be defined by a function:

$$I:\{1,\cdots,n_1\}*\{1,\cdots,n_2\}\to W\subseteq\mathbb{R},(i,j)\mapsto I_{i,j} \tag{3.8}$$

Such that image I can be represented by an array of size $n_1 * n_2$. Given that filter

$$K\in\mathbb{R}^{2h_1+1*2h_2+1} \tag{3.9}$$

the discrete convolution of the image I with filter K is given as:

$$(I*K)_{r,s}:=\sum_{u=-h_1}^{h_1}\sum_{v=-h_2}^{h_2}K_{u,v}I_{r+u,s+v} \tag{3.10}$$

where the filter K is given by:

$$K=\begin{pmatrix} K_{-h_1,-h_2} & \cdots & K_{-h_1,h_2} \\ \vdots & K_{0,0} & \vdots \\ K_{h_1,-h_2} & \cdots & K_{h_1,h_2} \end{pmatrix} \tag{3.11}$$

3.7.2. Convolution Layer

Let layer l be a convolution layer then the input of the layer comprises $m_1^{(l-1)}$ feature maps from the previous layer each of size $m_2^{(l-1)} * m_3^{(l-1)}$. When $l = 1$ the input is a single image I consisting of one or more channels thus a CNN accepts raw images as input. The output of layer l consists of m_1^l feature maps of size $m_2^l * m_3^l$. The i^{th} feature map in layer l denoted by Y_i^l is computed as:

$$Y_i^{(l)}=B_i^{(l)}+\sum_{j=1}^{m_1^{(l-1)}}K_{i,j}^{(l)}*Y_j^{(l-1)} \tag{3.12}$$

where $B_i^{(l)}$ is a bias matrix and $K_{i,j}^{(l)}$ is the filter of size $2h_1^{(l)}+1*2h_2^{(l)}+1$ connecting the j^{th} feature map in layer $(l-1)$ with the i^{th} feature map in layer l. When applying the discrete convolution only in the valid region of the input feature maps the output feature maps are of size:

$$m_2^{(l)}=m_2^{(l-1)}-2h_1^{(l)} \tag{3.13}$$

and

$$m_3^{(l)}=m_3^{(l-1)}-2h_2^{(l)} \tag{3.14}$$

The filters used for computing a fixed feature map $Y_i^{(l)}$ are the same. *i.e.*:

$$K_{i,j}^{(l)} = K_{i,k}^{(l)} \tag{3.15}$$

for $j \neq k$. This can be extended to multilayer perceptron as each feature map $Y_i^{(l)}$ in layer *l* consists of $m_2^{(l)} * m_3^{(l)}$ units arranged in a two-dimensional array. The unit at position (*r*, *s*) computes the output as:

$$\begin{aligned}\left(Y_i^{(l)}\right)_{r,s} &= \left(B_i^{(l)}\right)_{r,s} + \sum_{j=1}^{m_1^{(l-1)}} \left(K_{i,j}^{(l)} * Y_j^{(l-1)}\right)_{r,s} \\ &= \left(B_i^{(l)}\right)_{r,s} + \sum_{j=1}^{m_1^{(l-1)}} \sum_{u=-h_1^{(l)}}^{h_1^{(l)}} \sum_{v=-h_2^{((l)}}^{h_2^{(l)}} \left(K_{i,j}^{(l)}\right)_{u,v} * \left(Y_j^{(l-1)}\right)_{r+s,u+v}\end{aligned} \tag{3.16}$$

The trainable weights of the network can be found in the filters $K_{i,j}^{(l)}$ and the bias matrices $B_i^{(l)}$. Now the size of the output is:

$$m_2^{(l)} = \frac{m_2^{(l-1)} - 2h_1^{(l)}}{s_1^{(l)} + 1} \tag{3.17}$$

and

$$m_3^{(l)} = \frac{m_3^{(l-1)} - 2h_2^{(l)}}{s_2^{(l)} + 1} \tag{3.18}$$

where $s_1^{(l)}$ and $s_2^{(l)}$ are the subsampling skipping factors.

3.7.3. Non-Linearity Layer

Let layer l be non-linearity layer then its input is given by $m_1^{(l)}$ feature maps and its output comprises again $m_1^{(l)} = m_1^{(l-1)}$ feature maps each of size $m_2^{(l-1)} * m_3^{(l-1)}$ such that $m_2^{(l)} = m_2^{(l-1)}$ and $m_3^{(l)} = m_3^{(l-1)}$ and it's given by:

$$Y_i^{(l)} = f\left(Y_i^{(l-1)}\right) \tag{3.19}$$

where f is the activation function used in layer l and operates point wise.

3.7.4. Feature Pooling and Subsampling Layer

Reducing resolution can be accomplished in various ways. It can also be combined with pooling and done in a separate layer. Let l be the pooling layer its output comprises of $m_1^{(l)} = m_1^{(l-1)}$ feature maps of reduced size. Pooling operates by placing windows at non-overlapping positions in each feature map and keeping one value per window such that the feature maps are subsampled. There are two types of pooling, average pooling and max pooling the latter takes the maximum value of each window while the former takes the average.

3.7.5. Fully Connected Layer

Let layer *l* be a fully connected layer, if layer (*l*–1) is a fully connected layer as well we may apply:

$$z_i^{(l)} = \sum_{k=1}^{m^{(l-1)}} w_{i,k}^{(l)} y_k^{(l-1)} \tag{3.20}$$

Otherwise, layer l expects $m_1^{(l-1)}$ feature maps of size $m_2^{(l-1)} * m_3^{(l-1)}$ as input and the *i*th unit on the layer l computes:

$$Y_i^{(l)} = f\left(z_i^{(l)}\right) \tag{3.21}$$

with:

$$z_i^{(l)} = \sum_{j=1}^{m_1^{(l-1)}} \sum_{r=1}^{m_2^{(l-1)}} \sum_{s=1}^{m_3^{(l-1)}} w_{i,j,r,s}^{(l)} \left(Y_j^{(l-1)}\right)_{r,s} \tag{3.22}$$

where $w_{i,j,r,s}^{(l)}$ denotes the weight connecting the unit at position (r,s) in the jth feature map of layer $(l-1)$ and the ith unit in layer l. The fully connected layer is mainly used for classification purposes.

3.7.6. Error Backpropagation

This is an algorithm used to evaluate the gradient of the error function in each iteration step. For every layer, we compute two sets of gradients: the partial derivatives of z with respect to the layer parameter w^i and the layers input x^i.

The term $\frac{\partial z}{\partial w^i}$ can be used to update the current (i–th) layer's parameters. The term $\frac{\partial z}{\partial x^i}$ can be used to update parameters backwards, to the $(i-1)$th layer. Thus, we can continue the back-propagation process, and use $\frac{\partial z}{\partial x^i}$ to propagate the errors backward to the ith layer.

4. Results and Discussion

The main goal of this study was to achieve a robust classifier model that could discriminate the infant's audio cries thus responding to their needs accordingly. Before modelling of the audio datasets, data preparation was conducted in order to promise precise and accurate results.

4.1. Data Preparation

The fundamental stage that perhaps ensures that this study objective was met was data preparation stage. The trimmed audio dataset from different classes were loaded as folder on the Google GPUs. All the audio datasets were in form of a wave format as shown in Figure 6.

There were a total of 14,456 audio cry records from all the four classes of infant needs that this study research on. Different cries classes were having different number of cry records as shown in Table 1 below.

This study relied much on Librosa, a python library developed by Liang [13]. The module allows the audio to read and play in the python environment. The audio datasets were then transformed to frequency domain by leveraging the first Fourier transform function. Fourier transform converted the audio data to a function of amplitude and time in seconds. Interestingly, different audio classes portrayed different results after transformation.

This study used a more accurate granular form of audio representation called Mel spectrum images. The Mel spectrum images provide a fundamental basis for the features that the deep learning models uses as the distinction point. A more

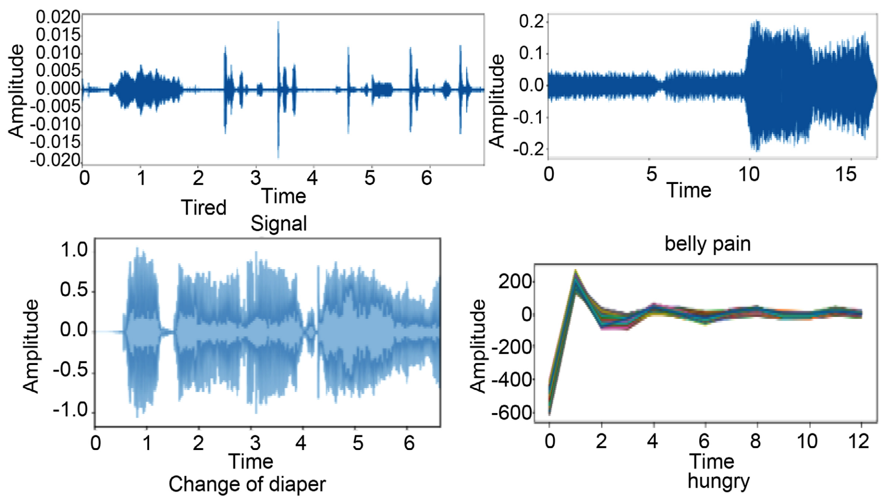


Figure 6. Datasets in wave format.

Table 1. Number of audio files per cry class.

Class	Number of audio files
Burping	2601
Discomfort	3863
Tired	3985
Hungry	4007
Total	14,456

robust index called MFCC stands for Mel frequency cepstral coefficients that converted the audio cries to time domain and to frequency domain provided even a plainer basis to explicitly identify the features of the infant audio cries. The MFCC ideally mimics the biological functionalities of the human ears. The MFCC index, therefore, provided a well granular data point for this study as compared with the time domain representation of the infant audio cries. As the study focused on the classification problem, there was need for accurate feeding of data points into the CNN model. The low intensity in the MFCC portrays the corresponding lower amplitude (Figure 7).

The Mel spectrum images shows difference in the intensity of the colors at different time points thus depicting distinguishing features the infant's audio classes.

4.2. Data Modeling

4.2.1. Mel Spectrum Coefficient

The Fourier transform of the wave file was transformed to Mel spectrum coefficient as shown below. This two-dimensional input dataset that was being fed into the CNN model with the corresponding labels (Figure 8).

4.2.2. Convolutional Neural Network Model Training

The input dataset was divided into the proportion of 80% to 20% training and

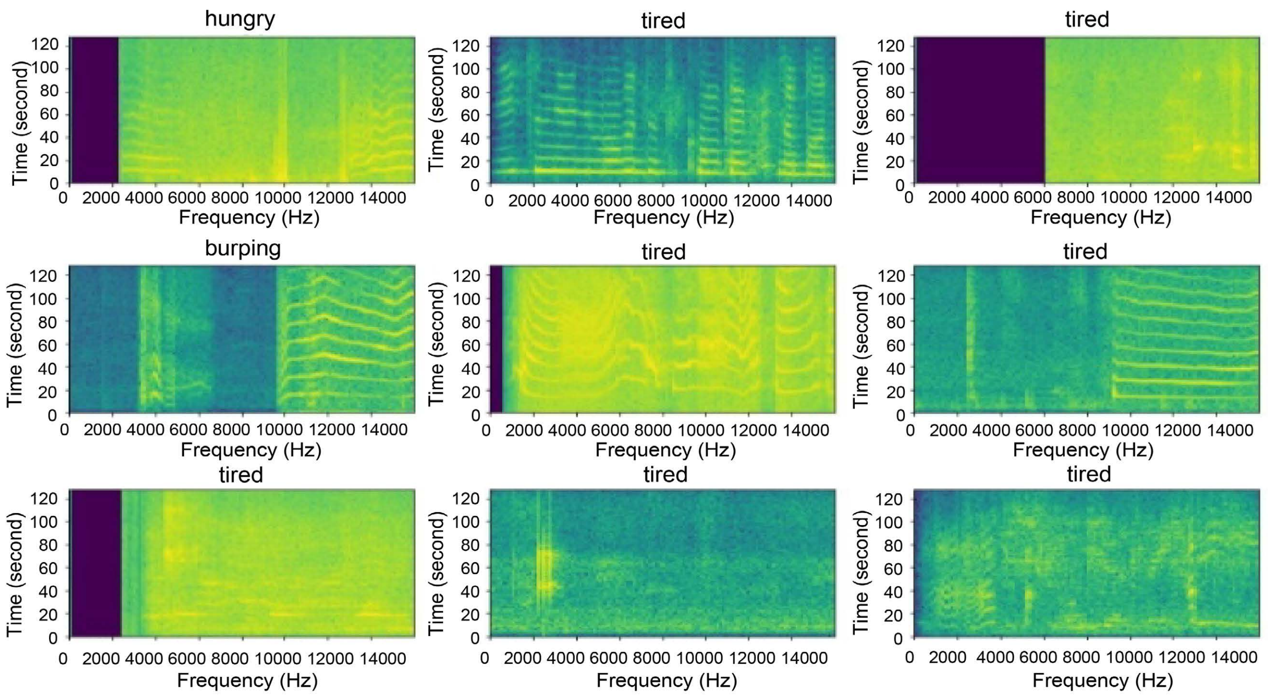
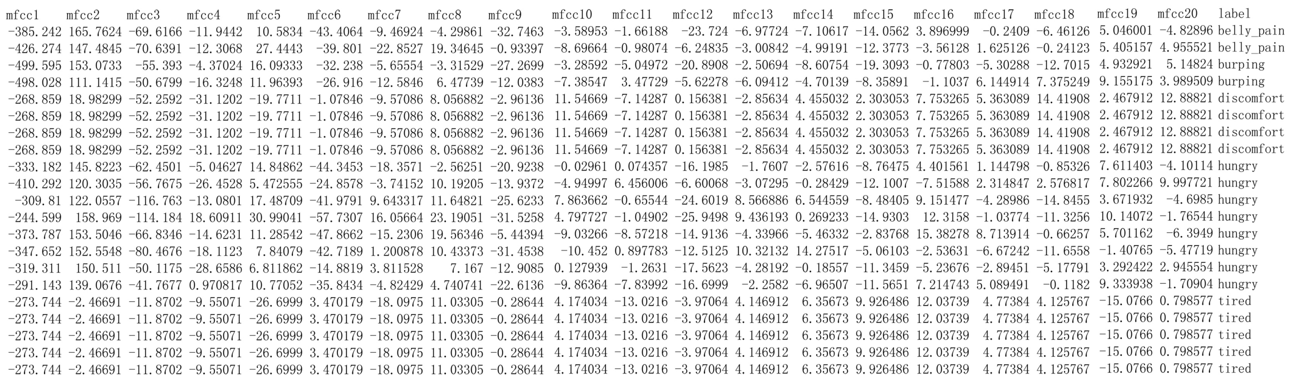


Figure 7. Cry class Mel spectrum images.



mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	mfcc6	mfcc7	mfcc8	mfcc9	mfcc10	mfcc11	mfcc12	mfcc13	mfcc14	mfcc15	mfcc16	mfcc17	mfcc18	mfcc19	mfcc20	label
-385.242	165.7624	-69.6166	-11.9442	10.5834	-43.4064	-9.46924	-4.29861	-32.7463	-3.58953	-1.66188	-23.724	-6.97724	-7.10617	-14.0562	3.896999	-0.2409	-6.46126	5.046001	-4.82896	belly_pain
-426.274	147.4845	-70.6391	-12.3068	27.4443	-39.801	-22.8527	19.34645	-0.93397	-8.69664	-0.98074	-6.24835	-3.00842	-4.99191	-12.3773	-3.56128	1.625126	-0.24123	5.405157	4.955521	belly_pain
-499.595	153.0733	-55.393	-4.37024	16.09333	-32.238	-5.65554	-3.31529	-27.2699	-3.28592	-5.04972	-20.8908	-2.50694	-8.60754	-19.3093	-0.77803	-5.30288	-12.7015	4.932921	5.14824	burping
-498.028	111.1415	-50.6799	-16.3248	11.96393	-26.916	-12.5846	6.47739	-12.0383	-7.38547	3.47729	-5.62278	-6.09412	-4.70139	-8.35891	-1.1037	6.144914	7.375249	9.155175	3.989509	burping
-268.859	18.98299	-52.2592	-31.1202	-19.7711	-1.07846	-9.57086	8.056882	-2.96136	11.54669	-7.14287	0.156381	-2.85634	4.455032	2.303053	7.753265	5.363089	14.41908	2.467912	12.88821	discomfort
-268.859	18.98299	-52.2592	-31.1202	-19.7711	-1.07846	-9.57086	8.056882	-2.96136	11.54669	-7.14287	0.156381	-2.85634	4.455032	2.303053	7.753265	5.363089	14.41908	2.467912	12.88821	discomfort
-268.859	18.98299	-52.2592	-31.1202	-19.7711	-1.07846	-9.57086	8.056882	-2.96136	11.54669	-7.14287	0.156381	-2.85634	4.455032	2.303053	7.753265	5.363089	14.41908	2.467912	12.88821	discomfort
-268.859	18.98299	-52.2592	-31.1202	-19.7711	-1.07846	-9.57086	8.056882	-2.96136	11.54669	-7.14287	0.156381	-2.85634	4.455032	2.303053	7.753265	5.363089	14.41908	2.467912	12.88821	discomfort
-333.182	145.8223	-62.4501	-5.04627	14.84862	-44.3453	-18.3571	-2.56251	-20.9238	-0.02961	0.074357	-16.1985	-1.7607	-2.57616	-8.76475	4.401561	1.144798	-0.85326	7.611403	-4.10114	hungry
-410.292	120.3035	-56.7675	-26.4528	5.472555	-24.8578	-3.74152	10.19205	-13.9372	-4.94997	6.456006	-6.60068	-3.07295	-0.28429	-12.1007	-7.51588	2.314847	2.576817	7.802266	9.997721	hungry
-309.81	122.0557	-116.763	-13.0801	17.48709	-41.9791	9.643317	11.64821	-25.6233	7.863662	-0.65544	-24.6019	8.566886	6.544559	-8.48405	9.151477	-4.28986	-14.8455	3.671932	-4.6985	hungry
-244.599	158.969	-114.184	18.60911	30.99041	-57.7307	16.05664	23.19051	-31.5258	4.797727	-1.04902	-25.9498	9.436193	0.269233	-14.9303	12.3158	-1.03774	-11.3256	10.14072	-1.76544	hungry
-373.787	153.5046	-66.8346	-14.6231	11.28542	-47.8662	-15.2306	19.56346	-5.44394	-9.03266	-8.57218	-14.9136	-4.33966	-5.46332	-2.83768	15.38278	8.713914	-0.66257	5.701162	-6.3949	hungry
-347.652	152.5548	-80.4676	-18.1123	7.84079	-42.7189	1.200878	10.43373	-31.4538	-10.452	0.897783	-12.5125	10.32132	14.27517	-5.06103	-2.53631	-6.67242	-11.6558	-1.40765	-5.47719	hungry
-319.311	150.511	-50.1175	-28.6586	6.811862	-14.8819	3.811528	7.167	-12.9085	0.127939	-1.2631	-17.5623	-4.28192	-0.18557	-11.3459	-5.23676	-2.89451	-5.17791	3.292422	2.945554	hungry
-291.143	139.0676	-41.7677	0.970817	10.77052	-35.8434	-4.82429	4.740741	-22.6136	-9.86364	-7.83992	-16.6999	-2.2582	-6.96507	-11.5651	7.214743	5.089491	-0.1182	9.333938	-1.70904	hungry
-273.744	-2.46691	-11.8702	-9.55071	-26.6999	3.470179	-18.0975	11.03305	-0.28644	4.174034	-13.0216	-3.97064	4.146912	6.35673	9.926486	12.03739	4.77384	4.125767	-15.0766	0.798577	tired
-273.744	-2.46691	-11.8702	-9.55071	-26.6999	3.470179	-18.0975	11.03305	-0.28644	4.174034	-13.0216	-3.97064	4.146912	6.35673	9.926486	12.03739	4.77384	4.125767	-15.0766	0.798577	tired
-273.744	-2.46691	-11.8702	-9.55071	-26.6999	3.470179	-18.0975	11.03305	-0.28644	4.174034	-13.0216	-3.97064	4.146912	6.35673	9.926486	12.03739	4.77384	4.125767	-15.0766	0.798577	tired
-273.744	-2.46691	-11.8702	-9.55071	-26.6999	3.470179	-18.0975	11.03305	-0.28644	4.174034	-13.0216	-3.97064	4.146912	6.35673	9.926486	12.03739	4.77384	4.125767	-15.0766	0.798577	tired
-273.744	-2.46691	-11.8702	-9.55071	-26.6999	3.470179	-18.0975	11.03305	-0.28644	4.174034	-13.0216	-3.97064	4.146912	6.35673	9.926486	12.03739	4.77384	4.125767	-15.0766	0.798577	tired

Figure 8. Mel spectrum coefficients.

testing respectively. The input data was divided in the proportion of 0.8 to 0.2 in every audio class folder. The resultant proportional files were then shuffled robustly so as to ensure efficient modelling in generalizability of the model inference. This was achieved by the python tensor flow libraries. The testing dataset was then shredded accordingly to completely minimize the mixture so that the model could be challenged with total new data that it has never seen.

From the literatures thus far, the number of epochs and batch size have been proposed to be an integer which should be chosen randomly, sequentially with keen monitoring of the learning rates. The CNN model was trained using 100 epochs with a batch size of 125. The accuracies of the model were realized to be significantly increasing with the number of epochs. However, the accuracies attained its plateau curve from 23. As seen in Figure 9, the realization depicted from this point onwards, increase in the number of the epochs was leading to a

```
Epoch   1/100
13/13   [==============================]   -   1s   4ms/step   -   loss:   1.0504   -   accuracy:   0.7704
Epoch   2/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.3097   -   accuracy:   0.9611
Epoch   3/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.1681   -   accuracy:   0.9611
Epoch   4/100
13/13   [==============================]   -   0s   3ms/step   -   loss:   0.1493   -   accuracy:   0.9611
Epoch   5/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.1416   -   accuracy:   0.9611
Epoch   6/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.1353   -   accuracy:   0.9617
Epoch   7/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.1321   -   accuracy:   0.9617
Epoch   8/100
13/13   [==============================]   -   0s   3ms/step   -   loss:   0.1220   -   accuracy:   0.9611
Epoch   9/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.1139   -   accuracy:   0.9636
Epoch   10/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.1104   -   accuracy:   0.9662
Epoch   11/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.1062   -   accuracy:   0.9643
Epoch   12/100
13/13   [==============================]   -   0s   4ms/step   -   loss:   0.1079   -   accuracy:   0.9694
```

Figure 9. Number of epochs versus training accuracies.

very negligible increase in accuracy of the model.

The successive training time was also realized to be decreasing with the number of epochs. At first, the model took 2 seconds to train in the first epoch while this duration was significantly reducing with increase in the number of epochs.

The CNN model through its soft max activation rendered the final classified class by voting from the predicted probabilities.

The highest probability attributed to a class was picked to be the most accurate predictions.

4.2.3. Accuracy versus Loss

The loss was significantly reducing with increase in the number of epochs. On the other hand, the accuracy was realized to be increasing with successive decrease in the loss. This potentially shows that; the loss function was being minimized in the successive epochs as the model penalizes well thus depicting constant linear learning rate.

4.2.4. Graphical Representation of the Learning Rate

According to the findings in the graph below, the accuracies in the successive epochs were realized to be inversely proportional to the losses. Both the training error was exponentially decreasing with increase in the number of training epochs. These potentially affirm the strength of CNN model to be the robust model for audio data. The choice of the 100 epochs also agreed with the learning rate as the accuracy stabilizes from epoch 80. The error also stabilizes from epoch 60 but steeply on the elbow at epoch 20 (**Figure 10**). The advantage that the model enjoyed in implementing this parameter is the exhaustive training of the model at optimal learning point.

4.2.5. Results from Other Models

The study also explored other machine learning models like Naive Bayesian, support vector machine, cat boost classifier model to ascertain the classifier model that fitted the audio dataset well. Decision trees had the lowest accuracy score of 70.65%. The convolutional neural networks performed relatively well with an accuracy of 92.73%. In the studies conducted by Nanni *et al.* (2015) the MLP promised a very higher accuracy but suffered from overfitting [14]. The validation stage eliminated the MLP model as the algorithm could not classify well the datasets it has never seen. The models were also evaluated in terms of computational time in minutes.

The Convolution Neural Networks (CNN) outperformed other generic models at the accuracy level of evaluations. This study was also keen to evaluate the model based on the training, testing accuracy and computational time. There were some machine learning models that really performed well based on training accuracy but really dropped in modelling the new datasets that were fed(testing). The convolutional neural networks did relatively well in generalizing the testing data thus having a higher accuracy. This promised the stability of the model thus affirming the operational stability of the model on the production environment. The convolution neural networks really possess the strength of down sampling with translational invariance. The dimension reduction has been scientifically embraced to reduce the computational time for the model. This brings a smooth significance in efficient operations of the model on the production environment. From the model accuracy (**Table 2**) below, it was realized that the mobile ResNet transfer model performed pretty well but relatively failed on testing data as compared with the CNN model.

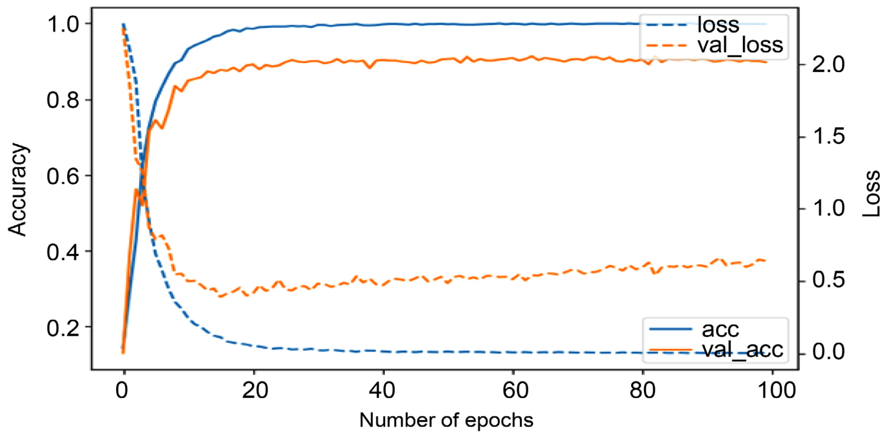


Figure 10. Successive epochs versus training.

Table 2. Model comparison.

Model	Training Acc.	Testing Acc.	Epoch	Computational time (minutes)
Decision Tree	0.70652	0.5671	100	73.02
Cat Boost Classifier	0.83696	0.6834	100	72.23
SVM	0.80102	0.6906	100	72.21
Naive Bayes	0.73029	0.6308	100	76.04
SGD	0.79348	0.7192	100	75.01
CNN	0.94739	0.9175	100	32.49
RNN	0.82609	0.8717	100	79.36
MLP	0.5783	0.6756	100	78.32

4.3. Model Evaluation

The study used different model evaluation metrics such as:

4.3.1. Model Precision Score

This metric was used to ascertain the degree of positively predicted class that was indeed correct. Although this metric is affected by the class imbalance, the study seeks to compare the index with other model evaluation metrics.

Precision Score = TP/(FP + T) = 0.72

4.3.2. Recall Score

This metric was used to evaluate how well the CNN model classified the positives out of all the possible positives present. This study in particular, used the recall to track all the classes that were predicted to be true out of the possible records that belonged to those particular infant needs. This metric is given by.

Recall Score = TP/(FN + TP) = 0.81

This study relied on this model evaluation metric as it is not affected by the data imbalance thus promising stable evaluation of the model performance even on the rare events. Burping in particular as one of the infant's cry needs was relatively rare in this study. There was a great improvement in the recall score as compared with precision index.

4.3.3. Model F1 Score

The study relied on this score as it combined the recall and precision together thus solving the problem in class imbalance. The model F1 score is a function of recall and precision. The score was realized to be.

F1 Score = 2 × Precision Score × Recall Score/(Precision Score + Recall Score)

$= 2 \times 0.72 \times 0.81/(0.72 + 0.81) = 0.76$

The study relied on the F1 score as the strongest way to evaluate the classification strength of the CNN model. The F1 score which is the function of the harmonic mean of the precision and recall potentially motivated the best way to handle class imbalance in evaluated the classifier model.

4.3.4. Confusion Matrix

The confusion matrix in Figure 11 shows the probabilities of classifying infant audio classes. The hungry and sick classes were accurately classified correctly with a probability of 0.89. The burping class was least classified correctly with probability of 0.29. This was because of the data imbalance as the burping audio files were less as compared with other records. Generally, the convolution neural networks performed well in classifying the audio datasets with only one class of burping poorly classified. Hidayati *et al.* (2009) confirmed that a reactively poorly imbalance class in classification problem is quite insignificant and thus renders the classifier models to be evaluated in the next production stage [15].

4.4. Deployment of the Model

After carefully evaluating the model, the chosen CNN model was deployed into production. The performance of the model at this stage was also evaluated based on the model's ability to handle the covariance shift. In production, the model deployment was designed to be taking the input on real time basis rather than batch processing. The model was packaged and containerized in a docker image that provided universal operations in almost near all the operating systems' any progressive changes and updates, the continuous development and continuous integration (CI/CD) was designed so as to take care of changes that could be pushed in the model without necessarily bring the entire system to downtime.

Application Functionality

The tool is expected to be working efficiently for about 7 meters radius. This means that any sound far apart may not be well discriminated by the application. The application was made very simple and accessible to every user who may not be tech-savvy. The application was also designed to be very light for the convenience of installation into smartphones without consuming much space.

The application will be launched upon which the audio record feature will pop out. Once the audio is recorded, the records will be moved to the model for

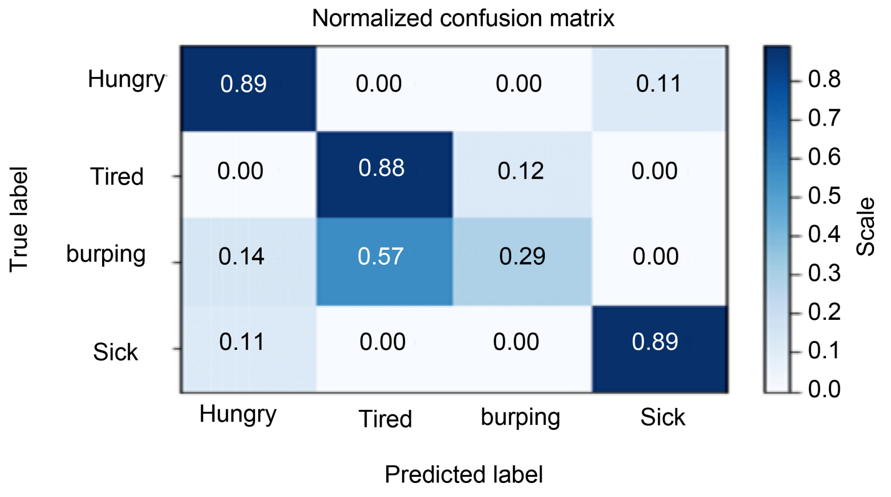


Figure 11. Normalized confusion matrix.

recognition and get to the server. A response shall be rendered back with the correct class of the classifications.

The final classification result was presented to the users on a simple bar graph showing the corresponding probabilities of the classified infant audio cry needs as shown in Figure 12 below.

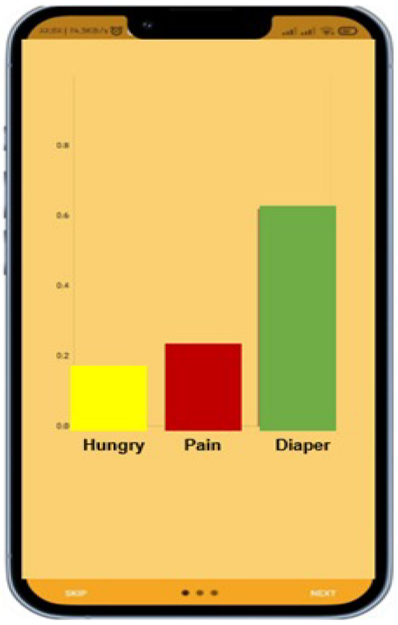


Figure 12. Snapshot classification results from the Application.

5. Conclusions and Recommendation

This study, therefore, confirms that the convolution neural networks indeed fitted the infant audio dataset well as compared with another model. The study also motivates the stage-wise evaluation of the deep learning model during the training stage and also upon production. On the realization of the efficient model that was least affected by the covariance shift, light and stable, the study came to conclude that the CNN model was very stable on deployment and faster on the inferential stage. The inherent down sampling through convolving the image pixels to arrive at receptor fields promoted the dimension reduction that facilitated the light weight of the CNN model. This was also supported by maximum pooling, a layer that was not affected by feature translation thus confirming the translational invariance of the activation functions in the CNN model.

The realization of the model also motivates the deliberate consideration of the acoustic background effects of the audio during the training of the models. This study found that most of the models being trained by the scientist are expected to work in a vacuum or controlled environment where there are no external influences. This study, therefore, through its introduction of background noise promised a stable classification operation even in noisy background noise. For this reason, the deployed model was not affected by the distribution drift from input audio datasets.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Ely, D.M. and Driscoll, A.K. (2021) Infant Mortality in the United States, 2019: Data from the Period Linked Birth/Infant Death File. *National Vital Statistics Reports*, **70**, 1-17. https://doi.org/10.15620/cdc:111053

[2] Ji, C., Mudiyanselage, T.B., Gao, Y. and Pan, Y. (2021) A Review of Infant Cry Analysis and Classification. *EURASIP Journal on Audio, Speech, and Music Processing*, **2021**, Article No. 8. https://doi.org/10.1186/s13636-021-00197-5

[3] Wasz-Höckert, O., Partanen, T.J., Vuorenkoski, V., Michelsson, K. and Valanne, E. (1964) The Identification of Some Specific Meanings in Infant Vocalization. *Experientia*, **20**, 154. https://doi.org/10.1007/BF02150709

[4] Ntalampiras, S. (2015) Audio Pattern Recognition of Baby Crying Sound Events. *AES: Journal of the Audio Engineering Society*, **63**, 358-369. https://doi.org/10.17743/jaes.2015.0025

[5] Sainath, T.N., Kingsbury, B., Saon, G., Soltau, H., Mohamed, A., Dahl, G. and Ramabhadran, B. (2015) Deep Convolutional Neural Networks for Large-Scale Speech Tasks. *Neural Networks*, **64**, 39-48. https://doi.org/10.1016/j.neunet.2014.08.005

[6] Chen, G., Parada, C. and Heigold, G. (2014) Small-Footprint Keyword Spotting Using Deep Neural Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, 4-9 May 2014, 4087-4091. https://doi.org/10.1109/ICASSP.2014.6854370

[7] Limantoro, W.S., Fatichah, C. and Yuhana, U.L. (2017) Application Development for Recognizing Type of Infant's Cry Sound. *Proceedings of* 2016 *International Conference on Information and Communication Technology and Systems*, Surabaya, 12 October 2016, 157-161. https://doi.org/10.1109/ICTS.2016.7910291

[8] Tran, K.T., Griffin, L.D., Chetty, K. and Vishwakarma, S. (2020) Transfer Learning from Audio Deep Learning Models for Micro-Doppler Activity Recognition. 2020 *IEEE International Radar Conference, RADAR*, Washington DC, 28-30 April 2020, 584-589. https://doi.org/10.1109/RADAR42522.2020.9114643

[9] Sinha, H., Awasthi, V. and Ajmera, P.K. (2020) Audio Classification Using Braided Convolutional Neural Networks. *IET Signal Processing*, **14**, 448-454. https://doi.org/10.1049/iet-spr.2019.0381

[10] Williams, T. and Li, R. (2018) An Ensemble of Convolutional Neural Networks Using Wavelets for Image Classification. *Journal of Software Engineering and Applications*, **11**, 69-88. https://doi.org/10.4236/jsea.2018.112004

[11] Lee, J., Park, J., Kim, K.L. and Nam, J. (2018) SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification. *Applied Sciences* (*Switzerland*), **8**, Article 150. https://doi.org/10.3390/app8010150

[12] Kuo, C.C.J. (2016) Understanding Convolutional Neural Networks with a Mathematical Model. *Journal of Visual Communication and Image Representation*, **41**, 406-413. https://doi.org/10.1016/j.jvcir.2016.11.003

[13] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G. and Vesely, K. (2011) The Kaldi Speech Recognition Toolkit. IEEE Signal Processing Society, Piscataway.

[14] Nanni, L., Costa, Y. and Brahnam, S. (2015) Set of Texture Descriptors for Music Genre Classification. 22*nd International Conference in Central Europe on Computer Graphics*, *Visualization and Computer Vision*, 145-152.

[15] Hidayati, R., Purnama, I.K.E. and Purnomo, M.H. (2009) The Extraction of Acoustic Features of Infant Cry for Emotion Detection Based on Pitch and Formants. *International Conference on Instrumentation*, *Communication*, *Information Technology*, *and Biomedical Engineering* 2009, Bandung, 23-25 November 2009, 1-5. https://doi.org/10.1109/ICICI-BME.2009.5417242