Journal of Computer Science

Original Research Paper

Convolution Neural Networks of Dynamically Sized Filters with Modified Stochastic Gradient Descent Optimizer for Sound Classification

Manu Pratap Singh and Pratibha Rashmi

Department of Computer Science, Dr. Bhimrao Ambedkar University, Khandari Campus, Agra, India

Article history
Received: 18-02-2023
Revised: 16-04-2023
Accepted: 27-09-2023

Corresponding Author:
Pratibha Rashmi
Department of Computer Science, Dr. Bhimrao Ambedkar University, Khandari Campus, Agra, India
Email: pratibha.rashmi@gmail.com

Abstract: Deep Neural Networks (DNNs), specifically Convolution Neural Networks (CNNs) are found well suited to address the problem of sound classification due to their ability to capture the pattern of time and frequency domain. Mostly the convolutional neural networks are trained and tested with time-frequency patches of sound samples in the form of 2D pattern vectors. Generally, existing pre-trained convolutional neural network models use static-sized filters in all the convolution layers. In this present work, we consider the three different types of convolutional neural network architectures with different variable-size filters. The training set pattern vectors of time and frequency dimensions are constructed with the input samples of the spectrogram. In our proposed architectures, the size of kernels and the number of kernels are considered with a scale of variable length instead of fixed-size filters and static channels. The paper further presents the reformulation of a minibatch stochastic gradient descent optimizer with adaptive learning rate parameters according to the proposed architectures. The experimental results are obtained on the existing dataset of sound samples. The simulated results show the better performance of the proposed convolutional neural network architectures over existing pre-trained networks on the same dataset.

Keywords: Deep Neural Network, Convolution Neural Networks, Sound Recognition, VGGNet, Pattern Classification, Stochastic Gradient Descent Optimizer

Introduction

In recent years, deep neural networks have been successfully applied to many real-world problems of various domains (Chu *et al*., 2009; Radhakrishnan *et al*., 2005; Mydlarz *et al*., 2017). More emphasis has been given to the problem domain of image recognition. It includes the task of classification or labeling of object recognition from the input images (Nair and Hinton, 2009). Convolutional Neural Networks (CNNs) are successfully applied in the domain of automatic speech recognition (Graves *et al*., 2013). Convolutional neural networks are used for various audio-processing tasks (Choi *et al*., 2016). Sound identification and sound tagging have many applications in security systems mainly for crime detection, alarm controlling, and password control (Radhakrishnan *et al*., 2005). The Internet of Things (IoT) devices are embedding automatic sound recognition systems for controlling various devices (Wang *et al*., 2014; Mydlarz *et al*., 2017). The machine may become adaptive to understand the sounds and able to recognize the sounds with an individual being. The deep neural network exhibited its strength and capabilities to accomplish such types of complex pattern recognition tasks. A deep convolution neural network considers the supervised environment to adjust its behavior for the given classification tasks. Thus, CNNs are considered the most suited machine learning technique to perform the sound classification task due to their ability to capture the pattern vectors from the time and frequency domains. The spectrogram is used in some cases to construct the input patterns with time and frequency pattern patches (Bogdanov *et al*., 2013). In early work, hand-drawn feature extraction methods were used to construct the pattern training set of the sound frequency samples. These feature extraction methods used filters and different transformations to remove unwanted noise from the sound inputs. In this attempt, the Mel Frequency Cepstral Coefficient (MFCC) modeled with the Gaussian mixture method and support vector machines were used in many applications (Salamon *et al*., 2014). The hand-crafted feature extraction technique leads to inferior performance due to the different pitches of the sound signals. Therefore, more discriminative features were developed but all of them were hand-crafted and derived from low-level descriptors such as MFCC (Phan *et al*., 2015), filter bank (Geiger and Helwani, 2015), or time-frequency descriptors (Chu *et al*., 2009). It is analyzed that all these models discard the temporal order of the frame level features and due to this, the considerable information loss. Thus, the earlier methods based on a handcrafted approach optimized the feature extraction process and the classification process separately rather than learning end-to-end. Further, the artificial neural networks were used for the classification of automatic sound recognition systems but again the training set for the fully connected feed-forward neural network was constructed with a handcrafted feature extraction process (Srisuk *et al*., 2018). The instinctive nature of convolution neural networks to jointly learn feature representation and appropriate classification leads the way for better performance for the automatic sound recognition system (Kons *et al*., 2013). Lots of research has been reported on automatic sound recognition using various models of deep neural networks, especially with convolutional neural networks. The spectrogram-based features of frequency and time vector were used generally for training. Lots of attempts were made to obtain better accuracy and good generalization for sound signals, but still, there is a challenge to efficiently improve the accuracy and generalization for the sound classification even though it contains noise. Thus, to improve the efficiency in sound classification there is a requirement for evolving the optimal architecture of CNNs with effective learning methods. In this present work, we consider the three different types of convolutional neural network architectures with different variable-size filters. The time and frequency dimensions are used to construct the training set pattern vectors from the input samples of the spectrogram. Thus, the 2D input samples of sound signals are used as input to the proposed convolutional networks. Our proposed architectures of convolutional neural networks are inspired by VGGNet Simonyan and Zisserman (2014) because it replaces large convolutional kernels with a stack of small kernels without pooling between these layers. Thus, it helps in the reduction of network parameters. In our proposed architectures of convolutional networks, the size of kernels and number of kernels are considered with a scale of variable length in ascending and descending order. In the proposed approach, a kernel of variable size is used to distribute the features extracted from the 2D input data to convolutional filters of variable size arranged in parallel. The novelty of the proposed approach is that it considers the convolution layers of filters and the number of filters as per the variable length scale. Thus, the size of kernels and the number of filters in a convolution layer are selected dynamically with variable length scale instead of fixed size filters and static channels. The proposed convolutional networks are trained with a reformulated mini-batch stochastic gradient descent optimizer with adaptive learning rate parameters. The experimental results show the better performance of the proposed convolutional neural networks over existing pre-trained convolutional neural networks on the same dataset. The effect of changes in the variable length scale of filters and size of filters are analyzed and the role of regularization and optimization are also considered in the performance analysis for the classification. The accuracy in the classification for sound samples of proposed models is considered and the suitable optimized design of the convolutional neural network is identified, which yields state-of-the-art performance for the classification of given sound data.

The major contribution of the authors in this present research paper can be considered as:

- A novel approach has been used for the construction of convolutional neural network models. The proposed approach considered the variable size filters of the receptive field instead of fixed size filters in all convolution layers. Besides this, the number of channels for each convolution layer is also considered variable length scale
- Mini-batch stochastic gradient descent optimizer is re-formulated for the proposed architectures of CNNs
- The proposed architectures of CNNs exhibited more flexibility for deciding the size of filters of receptive fields with respect to other existing pre-trained models
- Performance analysis of proposed architectures was presented on different parameters of classification accuracy

In recent work, the dynamic convolution is proposed to increase the complexity of the without increasing the network depth or width (Chen *et al*., 2020). In this approach, a single convolution kernel per layer, dynamic convolution aggregates multiple parallel convolution kernels dynamically based upon their attentions, which are input dependent. It has been found that by simply using dynamic convolution architecture accuracy of ImageNet classification is increased by 2.9%. In another approach, the

Dynamic Convolutional Neural Network (DCNN) is used for the reconstruction of high-resolution images from single low-resolution images (Bhujel and Pant, 2017). In this approach, the dynamic convolutional neural network directly learns an end-to-end mapping between low-resolution and high-resolution images. It has been found that the performance of the network is measured by PSNR, WPSNR, SSIM, and MSSSIM better than the pre-trained networks. Further, a Dynamic and Progressive Filter Pruning (DPFPS) scheme is proposed which is directly learning the structured sparsity network from Scratch (Ruan *et al*., 2021). It has been found that the proposed network increases the performance of the Convolutional Neural Networks for images. A stacked CNN and Recurrent Neural Network (RNN) model for sound event classification using weakly labeled data is proposed by Adavanne and Virtanen (2017). The model was evaluated on the UrbanSound8k dataset and outperformed several baseline models. Earlier, in convolution neural networks a general approach was followed to deal with classification problems for sound signals. In this approach, the audio signals are first converted into 2-dimensional pattern vectors, and then after, these are presented to pre-trained convolutional neural network architecture of image recognition (Deng *et al*., 2014). Generally, in most of the cases pre-trained CNN architectures are used for automatic sound recognition in which the spectrogram pattern of sound samples is considered as the training set (Cotton and Ellis, 2011). The problem that encountered here is the non-availability of the large quantities of training data to learn a non-linear function from input to output that is generalized well and yields high classification accuracy on unseen data. A method based on a 2D CNN with five layers is proposed (Salamon and Bello, 2017). In this method, new training samples are generated using the data augmentation method. In another method, 2D CNNs with random weights are proposed (Pons and Serra, 2019) for extracting features from sound spectrograms, and raw audio samples are used for sound classification. In this attempt, several experiments were conducted to find the best architecture for this method and the best result was obtained with VGG 2D CNN model with SVM classifier. In the Boddapati *et al*. (2017) spectrogram, the Mel-Frequency Cepstral Coefficient (MFCC) and Cross Recurrence Plot (CRP) are used with AlexNet and GoogleNet for the classification of sounds. A new technique of learning is proposed (Abdoli *et al*., 2019) named as Between Class (BC) learning for the training of neural networks. In this model the input is considered as the mixture of two audio samples and the network is trained to predict the mixing ratio of the samples. It performed well on various datasets of sounds compared to convolutional learning techniques (Piczak, 2015a). An end-to-end learning approach is proposed for speech recognition based on multi-scale convolutional that learns the representation directly from audio waveforms (Zhu *et al*., 2016). In this approach, three 1D convolutional layers with different filter sizes have been used for feature extraction and these features are further concatenated by a pooling layer to ensure a consistent sampling frequency for the rest of the network. Another end-to-end approach named SincNet is proposed for speaker identification and verification (Ravanelli and Bengio, 2018). An important investigation is considered for speech recognition using end-to-end multi-channel 1D CNN. It is found that the timing difference between channels is an indicator of the location of the input in space (Hoshen *et al*., 2015). Recently, several new deep convolutional models have been proposed for sound classification (Dai *et al*., 2017). These models consist of batch normalization, residual learning, and downsampling in the initial layers of CNN. In another approach dilated convolutional is used for feature extraction in audio clips to improve classification accuracy (Zhang *et al*., 2017). The dilated convolution is different from conventional CNN in that it does not use max-pooling layers and achieves good performance for sound classification. In another approach, one network learns directly from the audio waveform and the other one learns high-level representation from log-Mel features. These models are trained independently and the prediction of two models is combined using the dempster-Shafer method (Kim *et al*., 2018). Further, another hybrid model is proposed which also combines the prediction of two CNNs using the dempster-Shafer method (Salamon *et al*., 2014). In this approach, features such as Log-Mel spectrogram, MFCC, Chroma, Spectral contrast, and Tonnetz (CST) are extracted from the audio signals. The log-Mel, spectral contrast, and Tonnetz are stacked and considered as one feature set. Similarly, MFCC and CST features are stacked and considered as another feature set. These two feature sets are used for the training of two identical four-layer CNNs and the Dempster-Shafer method is used for the prediction from the combined CNNs. Thus, lots of research has reported on automatic sound recognition using various models of deep neural networks, especially with convolutional neural networks. The spectrogram-based features of frequency and time vector were used generally for training.

Materials and Methods

The existing dataset of sound samples i.e., UrbanSound8k is used for the construction of sample patterns for training and testing with the log scale Mel spectrogram method. In the simulation, we considered the 8732 time-frequency patches of the spectrogram as the input samples. Among these 6985 samples were used for training purposes 1747 samples were used for testing and a total (of 10) classes or labels were used for the classification of input samples. The class distribution of the UrbanSound8k dataset is shown in Table 1.

Table 1: Class distribution of UrbanSound8k dataset

Class name	Air conditioner [AI]	Car horn [CA]	Children playing [CH]	Dog bark [DO]	Drilling [DR]	Engine idling [EN]	Gun shot [GU]	Jackhammer [JA]	Siren [SI]	Street music [ST]
Label	1	2	3	4	5	6	7	8	9	10
Size	1000	429	1000	1000	1000	1000	374	1000	929	1000

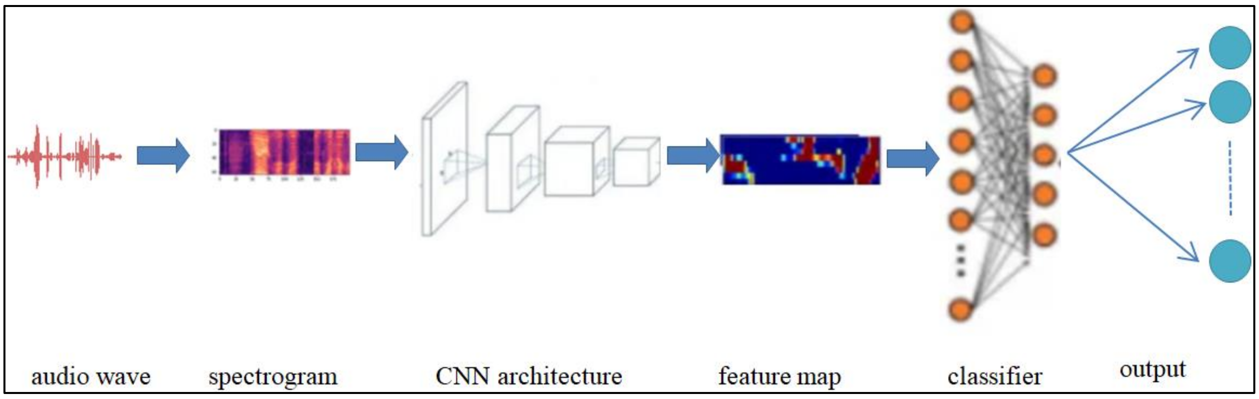


Fig. 1: Steps for UrbanSound classification

These input samples were presented to the convolutional neural network architectures. In our proposed method three convolutional neural network architectures were considered with different variable size filters of the receptive field. The size of the kernels and the number of channels are considered with a scale of variable length for the first two architectures. In the third architecture, the kernel of maximum variable size as per our scale i.e., 2^n is used to distribute the feature map extracted from the 2D input samples of sound signals to a block of the three convolution filters of scale 2^n arranged in parallel. In all three architectures, a single max pool is used after the last convolution layer followed by the two fully connected layers (dense network).

Deep Convolutional Neural Networks

Deep Neural Network architectures proposed in this present work are inspired by the Visual Geometry Group Network (VGGNet) (Simonyan and Zisserman, 2014). In the VGGNet, the depth of the network is a critical component to achieving better recognition or classification accuracy in Convolutional Neural Networks (CNNs). In our proposed model the two Convolutional Neural Network (CNN) architectures are comprised of variable length scales for a number of filters and for the size of filters of the convolution layers interleaved with one pooling operation followed by two fully connected layers. The first dense layer uses the regularization function whereas the output layer or classification layer uses the SoftMax activation function. The third proposed architecture consists of one convolution layer of maximum size variable scaled filters followed by a stack of convolution layers with the dynamical size of filters. These layers are further concatenated and passed through one pooling operation. The output of the pooling layer passes through the two fully connected layers as discussed earlier for the previous two proposed architectures. The steps for urban sound classification can be shown in Fig 1. The input to these networks consists of time-frequency patches of log-scaled mel-spectrogram representation of the audio signal. Generally, we use Essentia (Bogdanov *et al*., 2013) to extract a log-scale mel-spectrogram with 128 components covering the audible frequency. Thus, we considered the size of input TF patch X-128 frames i.e. $X \in R^{128\times128}$.

Therefore, the two-dimensional input X applies to the first convolution layer (H_1) which consists of m channels (filters) of size $n \times n$, where $n = 1, 2, \ldots \ldots \ldots .N$ Let $[X]_{i,j}$ and $[H_1]_{i,j}$ denote the value (TF patch) at location (i, j) in the 2D representation of the audio signal. Hence, this input data is processed through several trainable convolution layers for an appropriate representation of the input. Since the neurons in a layer are connected only to a small region of the previous layer, each of the hidden units receives input from each of the input pixels through the parameter weight tensor W. Let U contain biases, so that we can express the layer output as (Luo *et al*., 2016; Bologna and Pellegrini, 1998):

$$[H]_{i,j} = F\left[[U]_{i,j} + \textstyle\sum_k \sum_j [W]_{i,j,k,l}[X]_{k,l}\right] \quad (1)$$

Or:

$$[H]_{i,j} = F\left[[U]_{i,j} + \textstyle\sum_a \sum_b [V]_{i,j,a,b}[X]_{i+a,j+b}\right] \quad (2)$$

Such that $k = i + a$ and $l = j + b$

Here V represents the convolution filter or kernel of the convolution layer and F is a non-linear output function. Therefore, the deep convolution network is designed to learn the set of parameters V of convolution layers and of the dense (fully connected) layers to map the input to the predicted output T. Generally, with the hierarchy of layers and to use the Eq. 2, we can express the predicted output T in the terms of unknown parameters (W) for a fully connected network with non-linear output function F as:

$$T_1 = \Xi[F(X|V)]$$

or:

$$T_1 = \Xi[F_L(\dots\dots\dots F_2(F_1[V + V_1 \otimes X_1]|V_2)|V_L)] \quad (3)$$

and:

$$T = F(T_1|W) = F_o(F_H(T_1 * W_H + b_H) * W_o + b_o) \quad (4)$$

where, $\otimes$ represents the convolution operation or tensor product $*$ represents the dot product of the vectors, Ξ represents the max pool operator, T_1 is the final feature map obtained from the max pool layer inserted after the last convolution layer and b is a bias vector used by the layers of a fully connected network, L is the number of convolution (hidden) layers of the network, X_1 is the 2-dimensional input matrix of N features maps and V is a collection of the two-dimensional filters. The output of the final convolution layer (after max-pooling) is flattered and used as input for the first layer of the dense network. In the case of multiclass classification, the number of neurons in the output layer is considered according to the number of classes. Hence, for the output layer of the classification layer, the SoftMax activation function is used. The network is trained for the input samples and the parameters of the network are optimized using mini-batch stochastic gradient learning and regularization methods to minimize the error or cross-entropy (Kukačka *et al*., 2017).

In our proposed work, the variable length scale filters and channels are considered for the convolution layers instead of static size filters. In this approach, we select the size of filters and number of channels in a dynamic way according to the variable length scale for each convolution layer. Let n represent the variable scale according to which the size of filters is considered with variable scale k for considering the number of channels or filters of the receptive field. Hence, the size of filters of the convolution layer (L) is considered according to $2^n \times 2^n$ where n = 1, 2, 3 … N. The numbers of filters are considered as:

$$c = 2k \text{ where } k = 2^n + n \quad (5)$$

The output from the first convolution layer for a number of channels i.e., c_1 can be expressed as:

$$H_1^{c_1} = f_1^{c_1}((W^1 \otimes X) + U_1^{c_1} \quad (6)$$

Similarly, for the second layer and third layer, we have:

$$H_2^{c_2} = f_2^{c_2}((W^2 \otimes Z_1^{C_1}) + U_2^{c_2}) \quad (7)$$

$$H_3^{c_3} = f_3^{c_3}((W^3 \otimes Z_2^{C_2}) + U_3^{c_3}) \quad (8)$$

Thus, in general, for the Lth Layer, we have:

$$H_L^{c_L} = f_L^{c_L}((W^L \otimes Z_{L-1}^{C_{L-1}}) + U_L^{c_L}) \quad (9)$$

Let d_0 and d_1 be the dimension of the input vector and m_0 and m_1 be the dimension of the first convolution filter. Thus, the shape of X_0 and V will be (N, d_0, d_1) and (M, N, m_0, m_1) respectively.

Now, we can define the shape of W and H in a general way. The shape of W^1 will be (c_1, N, m_0, m_1) and the shape of $H_1^{c_1}$ will be $(c_1, d_0 - m_0 + 1, d_1 - m_1 + 1)$. Similarly, the shape of W^2 will be (c_2, c_1, m_2, m_3) and shape of $H_2^{c_2}$ will be $(c_2, (d_0 - m_0 + 1 - m_2 + 1), (d_1 - m_1 + 1 - m_3 + 1)$ or $(c_2, (d_0 - (m_0 + m_2) + 2), (d_1 - (m_1 + m_3) + 2)$.

Similarly, the shape of W^3 will be (c_3, c_2, m_4, m_5) and the shape of $H_3^{c_3}$ will be:

$$(c_3, (d_0 - (m_0 + m_2 + m_4) + 3), \\ (d_1 - (m_1 + m_3 + m_5) + 3)) \quad (10)$$

Hence, in this way, we can compute the size of W^L and $H_L^{c_L}$. Thus, the shape of W^L will be $(c_L, c_{L-1}, m_L, m_{L+1})$ and the shape of $H_L^{c_L}$ will be:

$$\begin{pmatrix} c_{L-1}, (d_0 - (m_0 + m_2 + m_4 + \dots\dots m_{L-1}) + L), \\ (d_1 - (m_1 + m_3 + m_5 + \cdots m_L) + L) \end{pmatrix} \quad (11)$$

Thus, in our proposed architectures the variable length scale for the number of filters as specified in equation 5 is considered. The first proposed architecture consists of three convolution layers and two dense layers (one hidden and one classification layer). Hence, $n = 1, 2, 3,$ and the number of filters in each layer will be of size 8×8, 4×4, and 2×2 respectively with the number of filters $c = 2k$ i.e., 22, 12, and 6 respectively. A detailed description of this architecture is as follows:

L_1 layer :	22 Filters with a receptive field of 8×8, so that W^1 has the shape $(22, 1, 8, 8)$ and the shape of H_1 is $(22, 121, 121)$. It uses the Rectified Linear Unit (ReLU) activation function i.e., $f_1(y) = \max(y, 0)$
L_2 layer:	12 Filters with a receptive field of 4×4, so that W^2 has the shape (12, 22, 4, 4) and the shape of H_2 is (12, 118, 118). It also uses the rectified linear unit activation function
L_3 layer :	6 Filters with a receptive field of 2×2, so that W^3 has the shape (6, 12, 2, 2) and the shape of H_3 is (6, 117, 117). This is also followed by a Rectified Linear Unit (ReLU) activation unit function

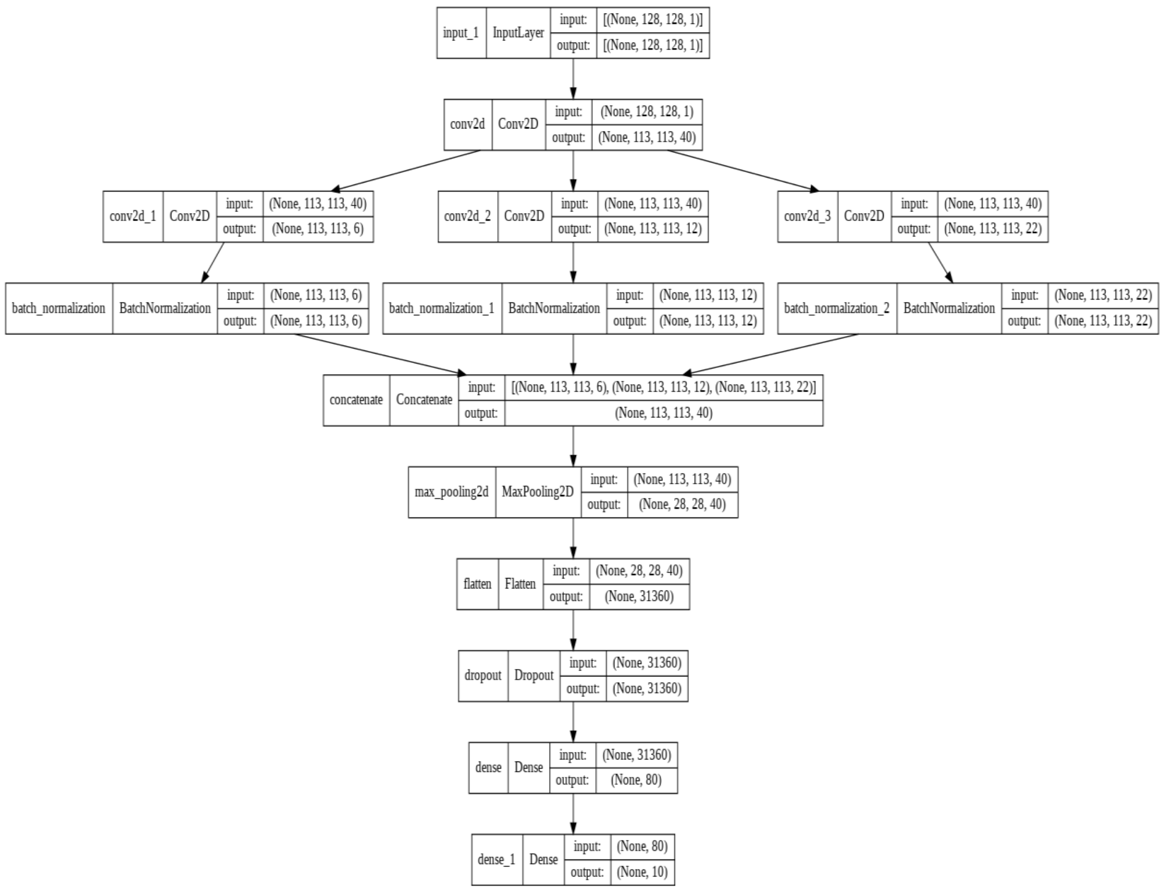

input_1 InputLayer input: [(None, 128, 128, 1)] output: [(None, 128, 128, 1)]
conv2d Conv2D input: (None, 128, 128, 1) output: (None, 113, 113, 40)
conv2d_1 Conv2D input: (None, 113, 113, 40) output: (None, 113, 113, 6)
conv2d_2 Conv2D input: (None, 113, 113, 40) output: (None, 113, 113, 12)
conv2d_3 Conv2D input: (None, 113, 113, 40) output: (None, 113, 113, 22)
batch_normalization BatchNormalization input: (None, 113, 113, 6) output: (None, 113, 113, 6)
batch_normalization_1 BatchNormalization input: (None, 113, 113, 12) output: (None, 113, 113, 12)
batch_normalization_2 BatchNormalization input: (None, 113, 113, 22) output: (None, 113, 113, 22)
concatenate Concatenate input: [(None, 113, 113, 6), (None, 113, 113, 12), (None, 113, 113, 22)] output: (None, 113, 113, 40)
max_pooling2d MaxPooling2D input: (None, 113, 113, 40) output: (None, 28, 28, 40)
flatten Flatten input: (None, 28, 28, 40) output: (None, 31360)
dropout Dropout input: (None, 31360) output: (None, 31360)
dense Dense input: (None, 31360) output: (None, 80)
dense_1 Dense input: (None, 80) output: (None, 10)

Fig. 2: Third convolution neural network architecture

Max-pool layer: L_3 layer is followed by $(4, 4)$ stride max-pooling. Therefore, the size of H_3 after the max pool will be $(6, 29, 29)$:

L_4 layer : The number of units for this layer is considered with the variable-length scale i.e. $v = 2 \times max(c)$. Thus, for this architecture, there will be 44 hidden units, and the connection weight W_h has the shape $(5046, 44)$ followed by the ReLu activation function

L_5 layer : In this layer, the 10 output units are used, and the weight W_o has the shape (44, 10) followed by a SoftMax activation function

The second proposed architecture consists of three convolution layers and two dense layers (one hidden and one classification layer). Again, n = 1, 2, 3 and the number of filters in each layer are of size 2×2, 4×4, and 8×8 respectively. A detailed description of this architecture can be discussed as:

L_1 layer: 22 Filters with a receptive field of 2×2, so that W^1 has the shape $(22, 1, 2, 2)$ and the shape of H_1 is $(22, 127, 127)$. It uses the Rectified Linear Unit (ReLU) activation function i.e., $f_1(y) = \max(y, 0)$

L_2 layer: 12 filters with a receptive field of 4×4, so that W^2 has the shape (12, 22, 4, 4) and the shape of H_2 is (12, 124, 124). It also uses the rectified linear unit activation function

L_3 layer: 6 filters with a receptive field of 8×8, so that W^3 has the shape (6, 12, 8, 8) and the shape of H_3 is (6, 123, 123). This is also followed by a Rectified Linear Unit (ReLU) activation unit function

Maxpool layer: In this architecture, the L_3 layer is followed by (4, 4) stride max-pooling. Therefore, the size of H_3 after the max pool will be $(6, 30, 30)$:

L_4 layer: The number of units for this layer is considered with the variable length scale i.e., $v = 2 \times max(c)$. Thus, for this architecture, there will be 44 hidden units and connection weight W_h has the shape (5400, 44) followed by the ReLu activation function

L_5 layer : In this layer, the 10 output units are used, and the weight W_o has the shape (44,10) followed by the SoftMax activation function

In the proposed third convolutional neural network architecture, we considered n = 4 and as per equation 5, the first convolution layer $\{L_0\}$ consists of 40 filters of size 16×16. The output of the L_0 layer is further distributed parallelly in the second convolution layer which consists of three convolution blocks i.e., $\{L_1, L_2$, and $L_3\}$ as shown in Fig. 2.

A detailed description of this architecture is as follows:

L_0 layer: 40 filters with a receptive field of 16×16, so that W^0 has the shape of $(40, 1, 16, 16)$ and the shape of H_0 is $(40, 113, 113)$. It uses the Rectified Linear Unit (ReLU) activation function i.e. $f_1(y) = \max(y, 0)$

The output of $the\ L_0$ layer is distributed into the three convolution blocks L_1, L_2, and L_3 in parallel. Hence, the parameters of L_1, L_2 and L_3 are as follows:

L_1 block: 22 Filters with a receptive field of 8×8, so that W^1 has the shape of $(22, 40, 8, 8)$ and the shape of H_1 is $(22, 113, 113)$. This is also followed by the Rectified Linear Unit (ReLU) activation function
L_2 block: 12 Filters with a receptive field of 4×4, so that W^2 has the shape of $(12, 40, 4, 4)$ and the shape of H_2 is $(12, 113, 113)$. It also uses the rectified linear unit activation function
L_3 block: 6 Filters with a receptive field of 2×2, so that W^3 has the shape of $(6, 40, 2, 2)$ and the shape of H_3 is $(6, 113, 113)$. This is also followed by a Rectified Linear Unit (ReLU) activation unit function

The output of L_1, L_2, and L_3 blocks are concatenated to produce a feature map as:

$$y = H_c[H_1, H_2, H_3 \tag{12}$$

Here, H_c represents the composite function for the concatenation. The shape of W_c is $(40, 40, 8, 8)$ and the shape of y is $(40, 113, 113)$. Now the y is passed through 4×4 stride max pooling.

Maxpool layer: The concatenated output y is passed through (4, 4) stride max pooling. Therefore, the size of y after the max pool will be (40, 28, 28):

L_4 layer : The number of units for this layer is decided with the variable-length scale i.e., $u = 2 \times max(c)$. Thus, for this architecture, there are 80 hidden units, and the connection weight W_h has the shape (31360, 80), followed by the ReLU activation function

L_5 layer: In this layer, the ten output units are used, so that the weight W_o has the shape (80,10) followed by the SoftMax activation function

Thus, for the first and second architectures, the layer L_4 is containing 44 hidden units followed by a ReLU activation function. In the third architecture layer, L_4 contains 80 hidden units. The L_5 layer contains 10 output units i.e., W_o has the shape $(44, 10)$ for the first and second architecture whereas $(80, 10)$ for the third architecture followed by a SoftMax activation function. It is quite clear, that dynamically sized filters of the receptive field according to Eq. 5 allow the network to learn small, localized features that can be fused at subsequent layers to gather evidence in support of a larger time-frequency patch that is indicative of the presence or absence of different sound classes even when the Spectro-temporal is masking with interfacing sources.

Learning

The proposed three architectures of CNNs are using mini-batch stochastic gradient descent learning (Bottou, 2010) to train the networks for the given log-el-spectrogram representation of the sample patterns. These sample patterns are considered in the form of TF-Patch and each batch consists of 128 TF-patches randomly selected from the training data without any repetition. The mini-batch Stochastic Gradient Descent (SGD) uses the independent identically distributed samples or batches as the sample sets to update the unknown parameters in each iteration. SGD has the advantages over the earlier gradient-based approaches due to the reason that SGD is using one sample randomly to update the gradient per iteration, rather than directly calculating the extract value of the gradient. It reduces the variance of the gradient and makes the convergence more stable (Darken *et al*., 1992). Therefore, the SGD has a better chance of finding the global optimal solution for complex problems (Nemirovski *et al*., 2009). In the proposed approach the models optimize the error function $l(W)$. Here l is an error function and W_o is the parameter to be optimised. The error function can be expressed as:

$$l(W_o) = \frac{1}{2M} \sum_{j=1}^{N} (d^j - f_o(y_j))^2 \tag{13}$$

where:

$$y_i = \sum_{j=1}^{N} W_o^i H_i + b_j$$

Here, M is the number of training samples, N is the number of units in the output layer, d is the predicated class and y is the activation of the unit.

In the stochastic gradient descent approach, change along the error function is obtained in the descent direction with respect to unknown parameter W_o in the weight space as:

$$\frac{\partial l(W_o)}{\partial W_o} \propto -\frac{1}{M} \sum_{j=1}^{N} \left(d^j - f_o(y_j)\right) . H_i \tag{14}$$

The weight update in (t +1) iteration can be expressed as:

$$W_o(t+1) = W_o(t) + \eta_o \frac{1}{M} \sum_{j=1}^{N} \left(d^j - f_o(y_j)\right) . H_i \quad (15)$$

Here, to reduce the over-fitting, the batch normalization is applied after the activation function of each convolution layer, whereas the dropout and regularization method is applied to the weights of both two layers i.e., L4 and L5 with a probability of 0.5 and penalty factor of 0.001 respectively (Srivastava *et al.*, 2014). However, the performance of mini-batch SGD can be further improved with the use of various optimization and regularization processes (Bühlmann and Van De Geer, 2011). One of the important factors is the choice of a proper learning rate. A too-small learning rate results in a slower convergence rate while a too-large learning rate can hinder convergence making error function fluctuate at the minimum. Besides the learning rate due to the existence of fluctuations in the SGD, the objective function is trapped in infinite numbers of local minimum. Therefore, the Nesterov Accelerated Gradient Descent method (Botev *et al.*, 2017) can be used to make improvements in the performance of SGD:

$$\widehat{W} = W_t + v^{old}.\alpha = v^{old}.\alpha + \eta\left(-\frac{\partial l(\widehat{W})}{\partial W}\right) \quad (16)$$

and, $W_{t+1} = W_t + v$

Here, α is a momentum factor and v^{old} represents the previous updates. Another issue in SGD is related to the choice of learning rate parameter. The most straightforward improvement in NAGD can be observed with the Adagrad method (Lydia and Francis, 2019). It adjusts the learning rate dynamically based on the reused gradients as:

$$g_t = \frac{\partial l(W_t)}{\partial W} V_t = \sqrt{\sum_{i=1}^{t} (g_i)^2 + \epsilon}$$

And:

$$W_{t+1} = W_t - \eta \frac{g_t}{V_t} \quad (17)$$

Here, g_t is the gradient of parameter W at iteration t. V_t represents the reused gradient of parameter W at iteration t and W_t is the value of parameter W at iteration t. Thus, the learning process for the proposed architectures is formulated with improvement as specified in Eqs. 16-17 for mini-batch SGD. Now the mini-batch SGD is reformulated for the proposed architectures of CNNs. Let us consider the first and second architectures for the formulation of the learning rule. In both architectures, three convolution layers are used followed by the two dense layers, and batch normalization is applied after each convolution layer. The 2-order state-dependent connection can be defined by the fact that the output of the L_1layer is just only related to the input of the L_2 layer and so on. The weight filters of different sizes are associated with each convolution layer $\{L_1, L_2, L_3, \ldots\ldots\}$. The input pattern ($X$) is two-dimensional and the output of L_1, L_2, and L_3 are also two-dimensional, whereas input to the dense layer L_4 and L_5 are one-dimensional pattern vectors. Thus, the forward propagation of L_1, L_2 and L_3 layers can be defined as:

$$H_1 = f_1\left[\left(X \otimes W_1^{c_1} + b\right)\right] \quad (18)$$

where, c_1 is the number of channels of respective filters used in the H_1 layer and b represents the offset value.

Or:

$$H_1(i,j) = f_1[BN\ [\textstyle\sum_{c_1} \sum_m \sum_n [X_{c_1}(i+m, j+n)\ W_{c_1}^1(x,y)] + b]] \quad (19)$$

Here, $H(i,j)$ corresponds to the pixel on the feature map, c_1 is the number of channels of the feature map, and m and n are the size of the convolution kernel. $BN(\,)$ is the batch normalization and $f(\,)$ is the activation function. Similarly, for $the\ L_2$ layer, we have:

$$H_2(i,j) = f_2[BN[\textstyle\sum_{c_2} \sum_{m_1} \sum_{n_1} [H_1(i+m_1, j+n_1)\ W_{c_2}^2(x,y)] + b]] \quad (20)$$

For $the\ L_3$ layer, we have the:

$$H_3(i,j) = f_3[BN[\textstyle\sum_{c_3} \sum_{m_2} \sum_{n_2} [H_2(i+m_2, j+n_2)\ W_{c_3}^3(x,y)] + b]] \quad (21)$$

Now, we consider the same for layers L_4 and L_5 as:

$$H_4 = f_H\left(\textstyle\sum_{i=1}^{c_3 \times m_2 \times n_2} W_{h_i}.H_3^i + b_h\right) \quad (22)$$

And:

$$S_j = f_o\left(\textstyle\sum_{h=1}^{R} W_o.H_4^h + b_o\right) \quad (23)$$

where, R is the number of units in the L_4 layer and $f_o(\,)$ is the SoftMax function. The backpropagated error is computed for these architectures and the weight update for each layer output as:

L_5 or output layer:

$$W_{oh}(t+1) = W_{oh}(t) \quad (24)$$

$$\eta\left(\frac{\partial l}{\partial W_{oh}(t)}\right)\Bigg/\sqrt{\sum_{i=1}^{t}\left(\frac{\partial l(i)}{\partial W_{oh}(i)}\right)^2+\epsilon}$$

or:

$$W_{oh}(t+1) = W_{oh}(t) - \eta\delta_1(t)/\Delta_1(t) \quad (25)$$

L_4 Layer: $W_{h_i}(t+1) = W_{h_i}(t) - \eta\delta_2(t)/\Delta_2(t)$ (26)

L_3 Layer: $W^3_{c_3}(m_2,n_2)(t+1) = W^3_{c_3}(m_2,n_2)(t) - \eta\delta_3(t)/\Delta_3(t)$ (27)

L_2 Layer: $W^2_{c_2}(m_1,n_1)(t+1) = W^2_{c_2}(m_1,n_1)(t) - \eta\delta_4(t)/\Delta_4(t)$ (28)

L_1 Layer: $W^1_{c_1}(m,n)(t+1) = W^1_{c_1}(m,n)(t) - \eta\delta_5(t)/\Delta_5(t)$ (29)

Here, $\delta(t)\ and\ \Delta(t)$ are presenting the first derivative of the error and reused gradient respectively.

Hence, the learning will take place here with the update of weight vectors to minimize the back-propagated error. Thus, the back-propagating process with weight update for the first two architectures can be represented for each layer as:

Output layer: $\delta_o = \frac{\partial L}{\partial S_j}$

L_5 Layer: $\delta_1 = \delta_0 * W_{oh}.\frac{\partial S_j}{\partial H_4}$ (30)

L_4 Layer: $\delta_2 = \delta_1 * W_{h_i} \otimes \frac{\partial H_4}{\partial H_3}$ (31)

L_3 Layer: $\delta_3 = \delta_2 * W^3_{c_3} \otimes \frac{\partial H_3}{\partial H_2}$ (32)

L_2 Layer: $\delta_2 = \delta_1 * W^2_{c_2} \otimes \frac{\partial H_2}{\partial H_1}$ (33)

L_1 Layer: $\delta_1 = \delta_2 * W^1_{c_2} \otimes X$ (34)

Now, we consider the third architecture for formulating the learning process. In the third architecture, one convolution layer $\{L_0\}$ is used for the input layer followed by the three convolution blocks $\{L_1, L_2,$ and $L_3\}$ in parallel with variable scale filters of receptive field. The feature maps of these three convolution layers are concatenated followed by a max-pooling layer. The feature map from the max-pool layer is flattened and presented to the first dense layer $\{L_4\}$ followed by the dense output layer $\{L_5\}$. The 2-order state-dependent connection can be defined by the fact that the output of the $\{L_0\}$ layer is related to the input of the $\{L_{1,} L_{2,} and\ L_3\}$ layers, and the output of these layers is further concatenated. The input pattern X is two-dimensional and the output of $\{L_{0,} L_{1,} L_2,$ and $L_3\}$ are also two dimensional whereas the input to the dense layer $\{L_4$ and $L_5\}$ are one-dimensional pattern vectors. Thus, the forward propagation for $L_{0,} L_{1,} L_2$ and L_3 can be defined as:

$$H_0 = f_1(X \otimes W_0^{c_1} + b) \quad (35)$$

or:

$$H_0(i,j) = f_1\left(BN\left(\textstyle\sum_{c_1}\sum_m\sum_n\left[X_{c_1}(i+m,j+n)W^0_{c_0}(x,y)\right]+b\right)\right) \quad (36)$$

Here, c_0 is the number of channels of receptive filters used in the L_0 layer and b represents the offset value.

Similarly, for $the\ L_1$ block we have:

$$H_1(i,j) = f_2\left(BN\left(\textstyle\sum_{c_2}\sum_{m_1}\sum_{n_1}\left[H_0(i+m_1,j+n_1)W^1_{c_1}(x,y)\right]+b\right)\right) \quad (37)$$

for $the\ L_2$ block, we have the:

$$H_2(i,j) = f_3\left(BN\left(\textstyle\sum_{c_2}\sum_{m_1}\sum_{n_1}\left[H_0(i+m_1,j+n_1)W^2_{c_2}(x,y)\right]+b\right)\right) \quad (38)$$

and for $the\ L_3$ block, we have:

$$H_3(i,j) = f_4\left(BN\left(\textstyle\sum_{c_3}\sum_{m_2}\sum_{n_2}\left[H_0(i+m_2,j+n_2)W^3_{c_3}(x,y)\right]+b\right)\right) \quad (39)$$

Since, for a 2-order state-dependent connection, the final output (H_4) from the concatenation of feature maps produced by the previous blocks $\{L_{1,} L_{2,} L_3\}$ can be represented as:

$$H_4 = h([\,H_1, H_2, H_3]) \quad (40)$$

Here, $h(\,)$ represents the concatenation function.

Now, we obtain the output for layers L_4 and L_5 as:

$$H_5 = f_H\left(\textstyle\sum_{i=1}^{u} W_{h_i} * H_4^i + b_h\right) \quad (41)$$

Here:

$$u = [(c_1,c_2,c_3)*(m_1,m_2,m_3)*(n_1,n_2,n_3)] \quad (42)$$

and $S_j = f_o\left(\sum_{h=1}^{R} W_{oh}.H_5^h + b_o\right)$

Here, R is the number of units in the L_5 layer and $f_o(\,)$is the softmax function.

Now, we compute the backpropagated error for this architecture and perform the weight update for each layer as:

Output L_5: $W_{oh}(t+1) = W_{oh}(t) - \eta\delta_1(t)/\Delta_1(t)$ (43)

Here $\delta_1(t)$ and $\Delta(t)$ have been already specified from Eq. 24:

L_4 Layer: $W_{h_i}(t+1) = W_{h_i}(t) - \eta\delta_2(t)/\Delta_2(t)$ (44)

L_3 Block: $W_{c_3}^3(m_2,n_2)(t+1) = W_{c_3}^3(m_2,n_2)(t) - \eta\delta_3(t)/\Delta_3(t)$ (45)

L_2 Block: $W_{c_2}^2(m_1,n_1)(t+1) = W_{c_2}^2(m_1,n_1)(t) - \eta\delta_4(t)/\Delta_4(t)$ (46)

L_1 Block: $W_{c_1}^1(m,n)(t+1) = W_{c_1}^1(m,n)(t) - \eta\delta_5(t)/\Delta_5(t)$ (47)

L_0 Layer: $W_{c_0}^0(x,y)(t+1) = W_{c_1}^o(x,y)(t) - \eta\delta_6(t).\Delta_6(t)$ (48)

The weight update for these layers to minimize the backpropagated error can be expressed as:

$$\text{Output layer: } \delta_o = \frac{\partial L}{\partial s_j}$$

L_5 Layer: $\delta_1 = \delta_o * W_{oh}\cdot\frac{\partial s_j}{\partial H_5}$ (49)

L_4 Layer: $\delta_2 = \delta_1 * W_{h_i} \otimes \frac{\partial H_5}{\partial H_4}$ (50)

L_3 Block: $\delta_3 = \delta_2 * W_{c_3}^3 \otimes \frac{\partial H_3}{\partial H_0}$ (51)

L_2 Block: $\delta_2 = \delta_2 * W_{c_2}^2 \otimes \frac{\partial H_2}{\partial H_0}$ (52)

L_1 Block: $\delta_1 = \delta_2 * W_{c_1}^1 \otimes \frac{\partial H_1}{\partial H_0}$ (53)

L_0 Layer: $\delta_o = \delta_1 * W_0^{c_1} \otimes X + \delta_2 * W_0^{c_1} \otimes X + \delta_3 * W_0^{c_1} \otimes X$ (54)

Thus, the back-propagated error from the output layer to all convolution layers is expressed and the weight update is also formulated for each layer. In the third architecture, the backpropagation needs to calculate the influence of the block layers, and the same error information is used for all the blocks i.e., $L_{3,}$ $L_{2,}$ and L_1. Thus, it is more conducive to the calculation of gradient information and the overall convergence speed of the network in comparison to the other two architectures. In the third architecture, equation 40 reflects the concatenation output. It merges the multi-channel parallel outputs into a single channel. In all three architectures, only one max-pooling layer is used in the end to avoid the loss of important features from the TF patches. The output of the last pooling layer for all features map is flattened and used as input to a fully connected layer.

Implementation Details and Simulation Design

In this present work, the audio signals of environmental sounds are considered for classification. The existing dataset UrbanSound8k of sound samples is used to provide the training of the proposed three different architectures of convolutional neural networks. Four methods i.e., Log-Mel Scale Spectrogram (LM), Mel Frequency Cepstral Coefficient (MFCC), Gammatone Frequency Cepstral Coefficients (GFCC) and Spectrogram. are used to extract the features from sound samples Though, MFCC is the most widely used feature extraction scheme for speech recognition and audio classification due to its better adaptability of network when noise is taken into consideration but most of the audio data we considered from already available datasets were clean samples so that, the spectrogram method is used for the feature extraction and to represent the audio data into the time-frequency patches. In the process of feature extraction, audio data pre-processing is performed with sampling, quantization, pre-emphasis processing, and windowing to convert the Analog audio signal into a sequence of audio frames. Further, a log-scale Mel-spectrogram is used to represent the pre-processed audio data into the time-frequency patches. Thus, two-dimensional feature vectors in the form of TF patches are used as input to the proposed convolutional neural network architectures as shown in Fig. 3.

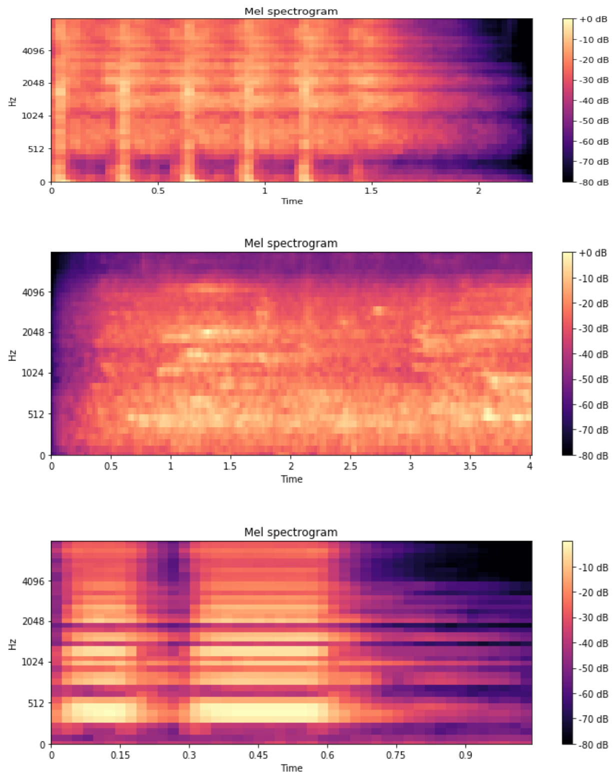


Fig. 3: Spectrogram for Siren, children playing, and horn voice sample

Table 2: Number of parameters for the first CNN architecture

Layer (type)	Output shape	Param #
Conv2d (Conv2d)	(None, 121, 121, 22)	1430
Activation (activation)	(None, 121, 121, 22)	0
Batch normalization	(None, 121, 121, 22)	88
Conv2d_1 (Conv2D)	(None, 118, 118, 12)	4236
Activation_1 (activation)	(None, 118, 118, 12)	0
Batch_normalization_1	(None, 118, 118, 12)	48
Conv2d_2 (Conv2D)	(None, 117, 117, 6)	294
Activation_2 (activation)	(None, 117, 117, 6)	0
Batch_normalization_2	(None, 117, 117, 6)	24
Max_pooling2d	(None, 29, 29, 6)	0
Flatten (flatten)	(None, 5046)	0
Dropout (Dropout)	(None, 5046)	0
Dense (dense)	(None, 44)	222068
Activation_3 (activation)	(None, 44)	0
Dense_1 (dense)	(None, 10)	450
Activation_4 (activation)	(None, 10)	0

Total params: 228, 638
Trainable params: 228, 558
Non-trainable params: 80

Three convolutional neural network architectures are proposed with different variable-size filters of the receptive field. The size of the kernels and the number of channels are considered with a scale of variable length for the first two architectures. In the third architecture, the kernel of maximum variable size as per our scale i.e., 2^n is used to distribute the feature map extracted from the 2D input samples of sound signals to a block of the three convolution filters of scale 2^n arranged in parallel. In all three architectures, a single max pool is used after the last convolution layer followed by the two fully connected layers (Dense network). The number of units in the first dense layer is set according to the maximum number of channels as per the variable-length scale parameter i.e., $c = 2k$ where $k = 2^n + n$, the value of n is considered as $n = 1, 2$ and 3 for the first two architectures and $n = 1, 2, 3$ and 4 for the third architecture. The number of units in the second dense layer is set according to the number of classes in which the sample audio will be classified. The ten (10) distinct classes are considered to classify the environmental sound samples. In the first architecture, we select $n = 1, 2$ and 3, so that, the three convolution layer filters with the receptive field of 8×8, 4×4, and 2×2. The numbers of channels are 22, 12, and 6 respectively in the three convolution layers. The last convolution layer is followed by 4×4 stride max-pooling over the obtained feature maps. The batch normalization has been applied after each convolution layer. There are two dense layers are used after the max-pool layer. Dropout is applied to the input of both the dense layers with 0.5 probabilities with L1-regularization to the weights of these two layers with a penalty factor of 0.001. The number of parameters for the first architecture can be shown in Table 2.

Now, in the second proposed convolutional neural network architecture, we again select the scale $n = 1, 2, 3$ and use the filters in reverse order i.e., the three convolution layer filters with receptive field of size 2×2, 4×4 and 8×8 and the number of channels are $22, 12$ and 6 respectively in these convolution layers. Again, the last convolution layer i.e., L_3 is followed by the $(4, 4)$ stride max-pooling over the obtained features map. The batch-normalization and regularization as used in the first architecture are considered in the same way also with the same probability and penalty factor. Thus, the number of parameters for the second architecture can be shown in Table 3.

In the third proposed CNN architecture, the variable length scale for the size of filters and number of channels is selected according to $n = 1, 2, 3$ and 4. Thus, primary convolution layer L_0 is considered with 40 channels $c = 2k$, $k = 2^n + n$ and $n = 4$ with receptive field of 16×16. The output feature map of the L_0 layer is distributed further in the convolution block which contains three convolution blocks $\{L_1, L_2,$ and $L_3\}$ in parallel with convolution filters with the receptive field of 2×2, 4× and 8×8 with 22, 12, and 6 number of channels respectively. The feature maps of blocks L_1, L_2, and L_3 are concatenated and passed through the (4, 4) stride max-pooling layer. The batch normalization and regularization are used after each convolution layer. Again, there are two dense layers are used after the max pool layer. Dropout is applied to the input of both the dense layers with 0.5 probability with L1 regularization to the weights of these layers with a penalty factor of 0.001. The number of parameters for the third architecture can be shown in Table 4.

It is quite clear from the simulation design of all the proposed architectures that the variable length scale is used for the number of channels and the filters of the receptive field $(2^n, 2^n)$. The maximum variable scale length i.e., $n = 3$ is used for the number of channels and filters of the receptive field i.e., (8, 8), (4, 4), and (2, 2) for the first two architectures, and the maximum variable scale length i.e., n = 4 is used for the number of channels and filters of receptive field i.e., (16, 16), (8, 8), (4, 4) and (2, 2) for the third architecture.

Table 3: Number of parameters for the second CNN architecture

Layer (type)	Output shape	Param #
Conv2d (Conv2D)	(None, 127, 127, 22)	110
Activation (activation)	(None, 127, 127, 22)	0
Batch normalization	(None, 127, 127, 22)	88
Conv2d_1 (Conv2D)	(None, 124, 124, 12)	4236
Activation_1 (activation)	(None, 124, 124, 12)	0
Batch_normalization_1	(None, 124, 124, 12)	48
Conv2d_2 (Conv2D)	(None, 123, 123, 6)	294
Activation_2	(None, 123, 123, 6)	0
Batch_normalization_2	(None, 123, 123, 6)	24
Max_pooling2d	(None, 30, 30, 6)	0
Flatten (flatten)	(None, 5400)	0
Dropout (dropout)	(None, 5400)	0
Dense (dense)	(None, 44)	237644
Activation_3 (activation)	(None, 44)	0
Dense_1 (dense)	(None, 10)	450
Activation_4 (activation)	(None, 10)	0

Total params: 242, 894
Trainable params: 242, 814
Non-trainable params: 80

Table 4: Number of parameters for the third CNN architecture

Layer (type)	Output shape	Param #
Input_1 (input Layer)	[(None, 128, 128, 1)]	0
Conv2d (Conv2D)	(None, 113, 113, 40)	10280
conv2d_1 (Conv2D)	(None, 113, 113, 6)	966
Conv2d_2 (Conv2D)	(None, 113, 113, 12)	7692
Conv2d_3 (Conv2D)	(None, 113, 113, 22)	56342
Batch normalization	(None, 113, 113, 6)	24
Batch_normalization_1	(None, 113, 113, 12)	48
Batch_normalization_2	(None, 113, 113, 22)	88
Concatenate (concatenate)	(None, 113, 113, 40)	0
Max_pooling2d (Maxpooling2d)	(None, 28, 28, 40)	0
Flatten (flatten)	(None, 31360)	0
Dropout (dropout)	(None, 31360)	0
Dense (dense)	(None, 80)	2508880
Dense_1 (dense)	(None, 10)	810

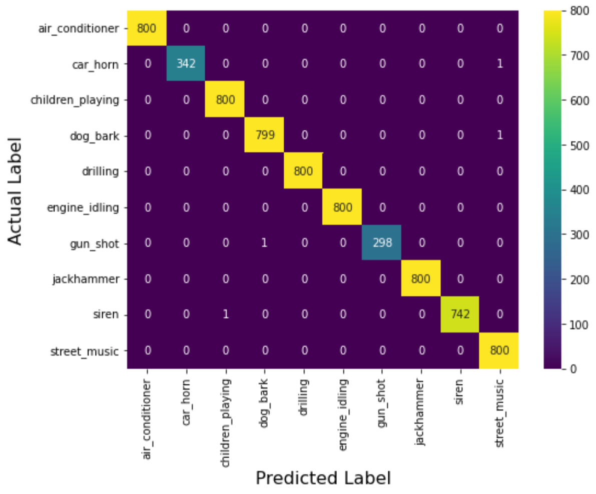
Total params: 2, 585, 130
Trainable params: 2, 585, 050
Non-trainable params: 80

Mini-batch Nesterov Accelerated Gradient Descent with AdaGrad method is reformulated for the proposed architectures to minimize the mean squared error. During the training mini batches are constructed for the given sound data. Each batch consists of 128 TF patches randomly selected from the training samples without any repetition. Each 3-sec TF-patch is taken from a random position in time from the full log-Mel Spectrogram representation of each training sample. All three proposed models are trained for a maximum of 200 epochs and a checkpoint is used. After each epoch, the models are trained on random mini-batches until 1/10 of all training data is exhausted. A validation set is used to identify the parameters setting achieving the highest classification accuracy, where prediction is performed by slicing the test sample into overlapping TF- patches, making a prediction for each TF patch and finally choosing the sample label prediction of the class with the highest near output activation over all frames. Simulation results for the proposed architecture are obtained in a Python programming environment (Rolon-Mérette *et al.*, 2016).

Results and Discussion

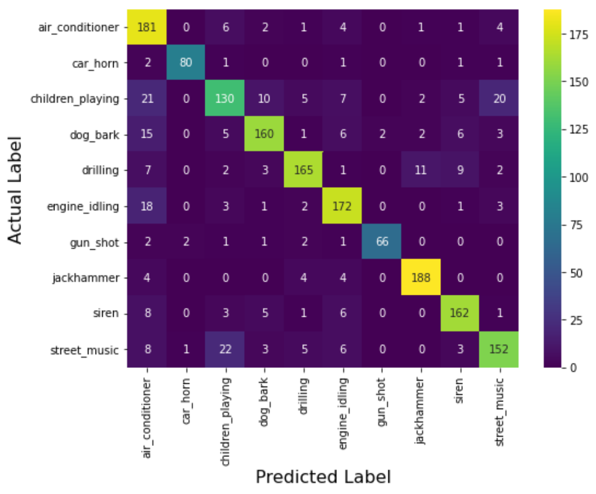
Three different types of convolutional neural network architectures are considered with different variable-size filters and channels. The pattern vectors of time and frequency dimensions are considered for the training and testing. In the proposed three architectures of convolutional neural networks, the size of the filters and the number of filters are considered dynamically with variable length scale instead of fixed or static sizes. In the first proposed architecture, three convolution layers followed by the max-pooling layer with a stride of (4, 4) are used. The flattened feature map from the max pool is presented as input to the first dense layer. The output of this dense layer is fed forwarded to the output layer for classification.

The first proposed convolutional neural network is trained with Nadam Optimizer with the loss function of mean square error. The simulated results are presented in Table 1 with a test loss of 0.0287 and test classification accuracy of 0.833. The simulation results are presenting 99.62% maximum accuracy and 85.25% maximum validation accuracy for the proposed architecture. The confusion matrix of this architecture for testing and training data is presented in Figs. 4a-b. The model accuracy and model loss are presented in Figs. 5-6. The Fig. 5 represents the model accuracy for training and validation. It can be observed that there is a continuous curve for training but a fluctuation in validation.



	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
air_conditioner	800	0	0	0	0	0	0	0	0	0
car_horn	0	342	0	0	0	0	0	0	0	1
children_playing	0	0	800	0	0	0	0	0	0	0
dog_bark	0	0	0	799	0	0	0	0	0	1
drilling	0	0	0	0	800	0	0	0	0	0
engine_idling	0	0	0	0	0	800	0	0	0	0
gun_shot	0	0	0	1	0	0	298	0	0	0
jackhammer	0	0	0	0	0	0	0	800	0	0
siren	0	0	1	0	0	0	0	0	742	0
street_music	0	0	0	0	0	0	0	0	0	800

(a)



	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
air_conditioner	181	0	6	2	1	4	0	1	1	4
car_horn	2	80	1	0	0	1	0	0	1	1
children_playing	21	0	130	10	5	7	0	2	5	20
dog_bark	15	0	5	160	1	6	2	2	6	3
drilling	7	0	2	3	165	1	0	11	9	2
engine_idling	18	0	3	1	2	172	0	0	1	3
gun_shot	2	2	1	1	2	1	66	0	0	0
jackhammer	4	0	0	0	4	4	0	188	0	0
siren	8	0	3	5	1	6	0	0	162	1
street_music	8	1	22	3	5	6	0	0	3	152

(b)

Fig. 4: (a) Confusion matrix of first CNN architecture for training; (b) Confusion matrix for first CNN architecture for testing

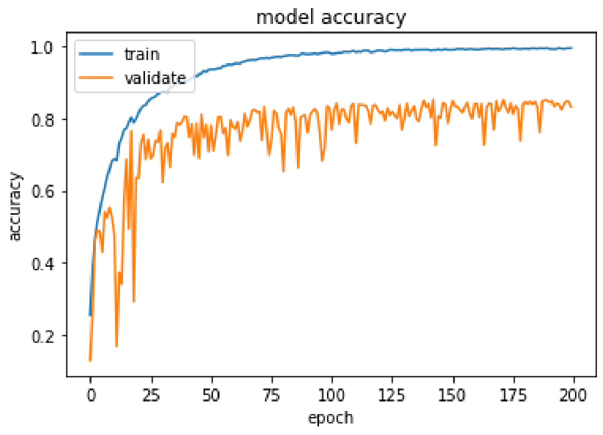


Fig. 5: Accuracy for First CNN evaluated on training and testing data

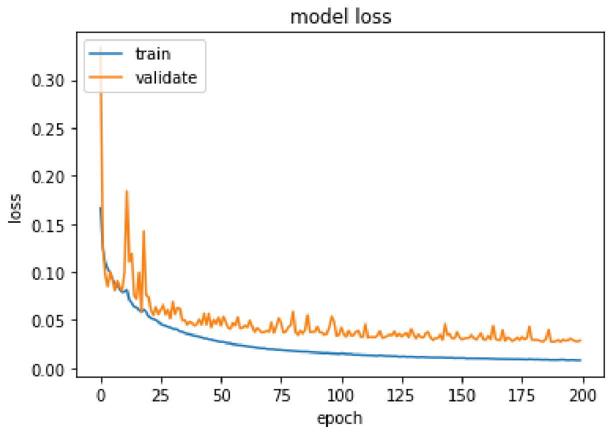
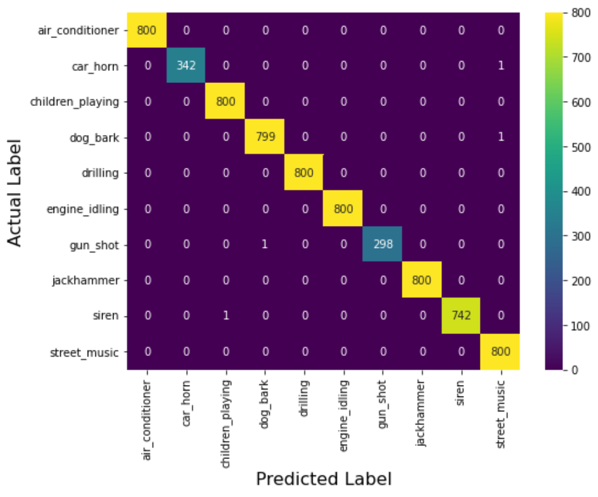


Fig. 6: Loss for First CNN evaluated on training data and testing data



	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music
air_conditioner	800	0	0	0	0	0	0	0	0	0
car_horn	0	342	0	0	0	0	0	0	0	1
children_playing	0	0	800	0	0	0	0	0	0	0
dog_bark	0	0	0	799	0	0	0	0	0	1
drilling	0	0	0	0	800	0	0	0	0	0
engine_idling	0	0	0	0	0	800	0	0	0	0
gun_shot	0	0	0	1	0	0	298	0	0	0
jackhammer	0	0	0	0	0	0	0	800	0	0
siren	0	0	1	0	0	0	0	0	742	0
street_music	0	0	0	0	0	0	0	0	0	800

(a)

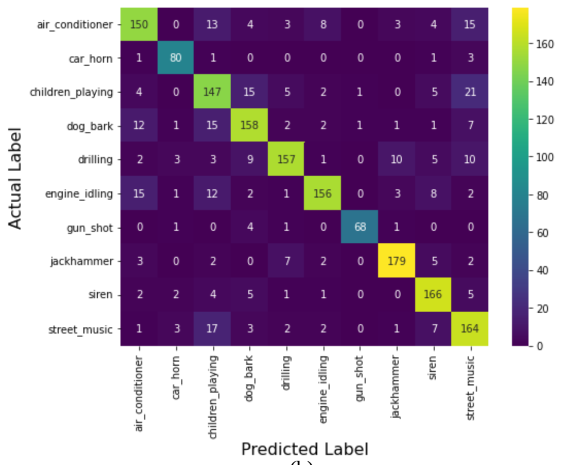

(b)

Fig. 7: (a) Confusion Matrix of second CNN architecture for training; (b): Confusion Matrix of second CNN architecture for testing

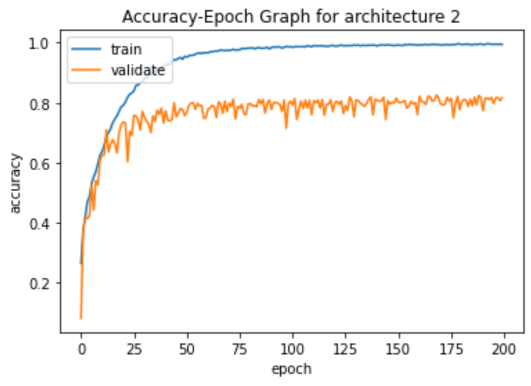


Fig. 8: Accuracy for second CNN evaluated on training and testing data

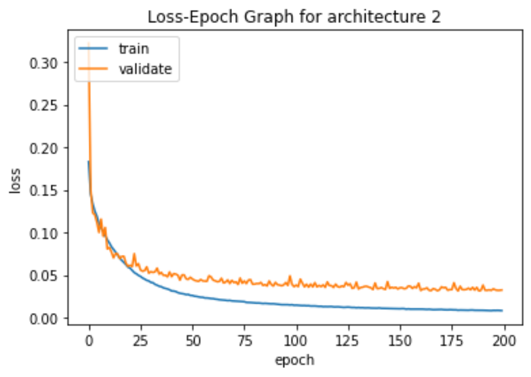
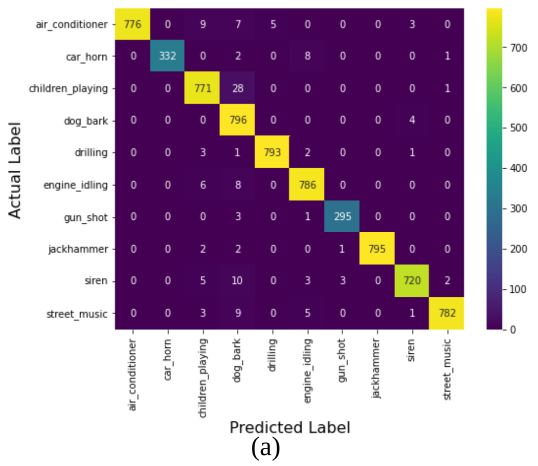


Fig. 9: Loss for second CNN evaluated on training and testing data

The second proposed convolutional neural network is trained with Nadam optimizer with the loss function of mean square error. The simulated results are presented in Table 2 with a test loss of 0.032 and test classification accuracy of 0.815. The simulation results present 99.64% maximum accuracy and a maximum validation accuracy of 82.54% for this architecture. The confusion matrix of this architecture for training and testing data is presented in Figs. 7a-b. The model accuracy and model loss are presented in Figs. 8-9. The Fig. 8 represents the model accuracy for training and validation. It can be observed that there is also a continuous curve for training but fluctuation in validation.

The third proposed convolutional neural network is implemented and trained with Nadam Optimizer with the loss function of mean square error. The simulated results are presented in Table 3 with a test loss of 0.042 and test classification accuracy of 0.736. The simulation results are presenting a 97.59% maximum accuracy and a maximum validation accuracy of 75.95% for this architecture. The confusion matrix of this architecture for training and testing samples is presented in Figs. 10a-b. The model accuracy and model loss are presented in Figs. 11-12. Figure 11 represents the model accuracy for training and validation. It can be observed that there is a continuous curve for training but a fluctuation in validation.

The precision, recall, F1-score, and average accuracy are computed for the proposed architectures and presented in Table 5.


(a)

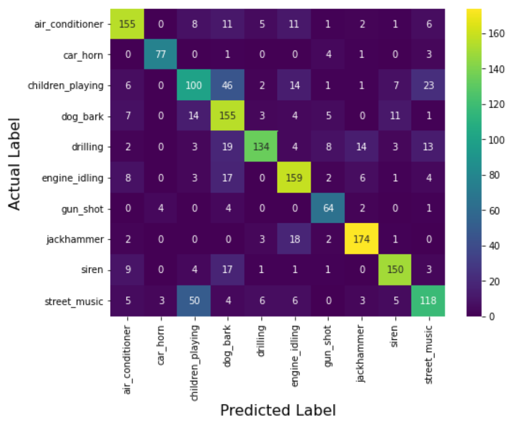

(b)

Fig. 10: (a) Confusion matrix of third CNN architecture for training; (b) Confusion matrix of third CNN architecture for testing

Table 5: Performance of proposed architectures and existing pre-trained CNN architectures

Models		AI	CA	CH	DO	DR	EN	GU	JA	SI	ST	Macro-average
Arch1	Precision	0.68	0.96	0.75	0.86	0.89	0.83	0.97	0.92	0.86	0.82	0.85
CNN	Recall	0.91	0.93	0.65	0.80	0.82	0.86	0.88	0.94	0.87	0.76	0.84
	F1-score	0.78	0.95	0.70	0.83	0.85	0.84	0.92	0.93	0.87	0.79	0.85
Arch2	Precision	0.79	0.88	0.69	0.79	0.88	0.90	0.97	0.90	0.82	0.72	0.83
CNN	Recall	0.75	0.93	0.73	0.79	0.79	0.78	0.91	0.90	0.89	0.82	0.83
	F1-score	0.77	0.90	0.71	0.79	0.83	0.83	0.94	0.90	0.86	0.76	0.83
Arch3	Precision	0.80	0.92	0.55	0.57	0.87	0.73	0.73	0.86	0.84	0.69	0.75
CNN	Recall	0.78	0.90	0.50	0.78	0.67	0.80	0.85	0.87	0.81	0.59	0.75
	F1-score	0.79	0.91	0.52	0.65	0.76	0.76	0.79	0.86	0.82	0.63	0.75
Existing	Precision	0.74	0.94	0.63	0.85	0.86	0.80	0.93	0.87	0.95	0.70	0.83
Model	Recall	0.83	0.79	0.71	0.80	0.81	0.84	0.89	0.84	0.83	0.73	0.81
[CNN]	F1-score	0.78	0.86	0.67	0.83	0.83	0.82	0.91	0.85	0.88	0.71	0.82

Table 6: Performance comparison of three proposed CNN architectures

	Epoch	Min loss	Max accuracy (%)	Max validate accuracy (%)	Min validate loss
Arch 1 CNN	200	0.008	99.6278	85.2891	0.027
Arch 2 CNN	200	0.008	99.6421	82.5415	0.031
Arch 3 CNN	200	0.009	97.5948	75.9588	0.040

Table 7: Classification accuracy on the UrbanSound8k dataset

Model	Classifier	Features	Loss	Optimizer	Accuracy (%)
Arch1 (Proposed Model -1)	CNN	Log Mel spectrogram	MSE	Nadam	85.29
Arch2 (Proposed Model-2)	CNN	Log Mel spectrogram	MSE	Nadam	82.54
Arch3 (Proposed Model-3)	CNN	Log Mel spectrogram	MSE	Nadam	75.95
Salamon and Bello (2017)	CNN	Log Mel spectrogram	Categorical cross-entropy	Adam	73.00
Salamon and Bello (2017)	CNN + aug	Log Mel spectrogram	Categorical cross-entropy	Adam	79.00
Piczak (2015b)	CNN	Log Mel spectrogram	MSE	Nesterov	73.00
Lezhenin *et al*. (2019)	CNN	Log Mel spectrogram	Categorical cross-entropy	Adam	80.48

Further, the comparison of performance is performed for the parameters namely min loss, max accuracy, max validation accuracy, and min validate loss for all the three proposed convolutional neural network architectures. This comparison can be seen in Table 6. Per class accuracy for all three different types of Convolution Neural Networks on the UrbanSound8k dataset is shown in Figs. 13-15.

Simulation results present the highest accuracy for the first proposed architecture. The simulation results indicate that the performance of our two proposed architectures i.e., first architecture and second architecture are better than the existing models of CNN for the classification of environment sounds. The accuracy of the third proposed model is 76% and it is better than the pretrained models of Salamon and Bello, 2017). The comparative analysis between the three proposed CNN models and the other existing models for the UrbanSound8k dataset can be presented in Table 7 and further in Fig. 16 to show the performances for validation accuracy.

The obtained simulated results are presenting better performance and optimal implementation in terms of accuracy, individual class, total accuracy, precision, recall, and F1-score with respect to the existing pre-trained models performed on the same samples of sounds dataset. Overall, the performance of the first proposed architecture is better than all the other existing pretrained models besides the proposed two other models. Therefore, the dynamically selected number of filters and size of filters improves the performance of convolutional neural networks for all the accuracy measurement parameters, with respect to the static size filters for all the convolution layers.

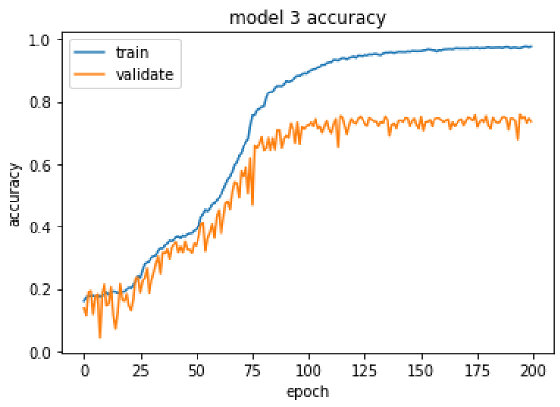


Fig. 11: Accuracy for third CNN evaluated on training and testing data

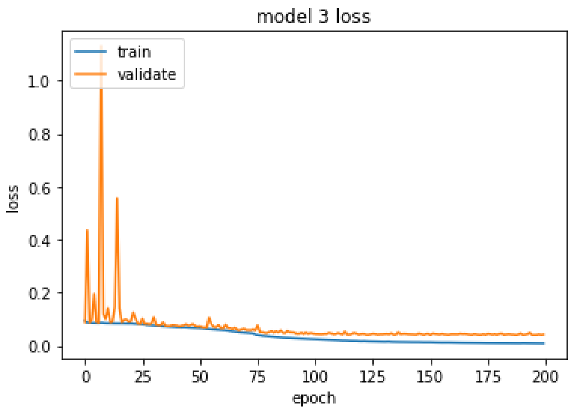


Fig. 12: Loss for third CNN evaluated on training and testing data

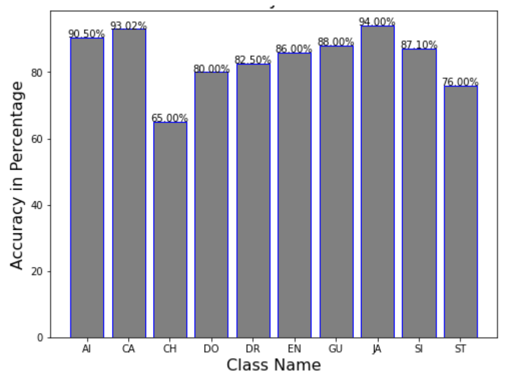


Fig. 13: Class-wise accuracy for the first CNN architecture

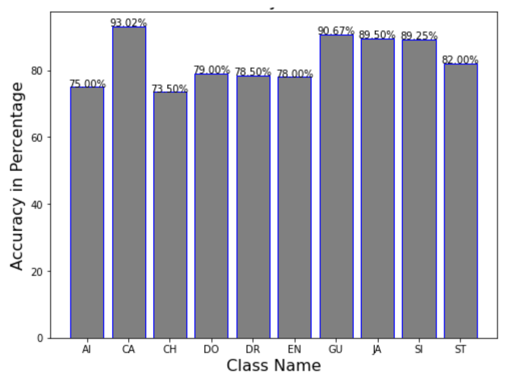


Fig. 14: Class-wise accuracy for second CNN architecture

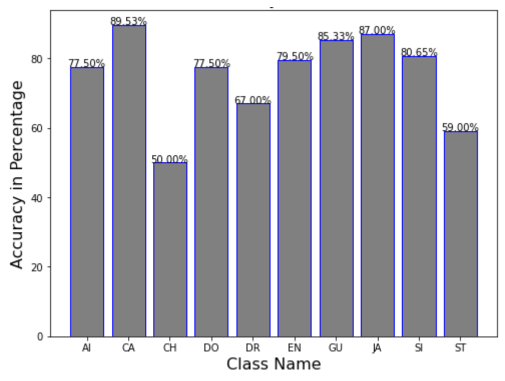


Fig. 15: Class-wise accuracy for the third CNN architecture

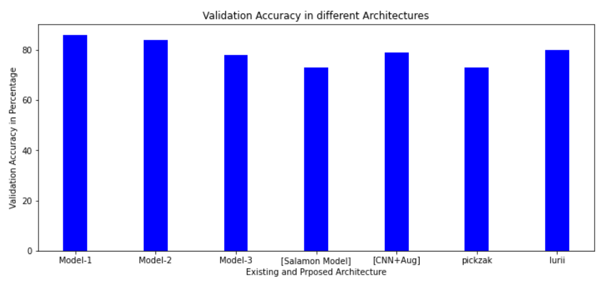


Fig. 16: Three proposed convolution neural networks and other existing model's accuracy results on the UrbanSound8k dataset

Conclusion

In this presented work, a novel approach is used for the construction of convolutional neural network architectures. In this approach, the size of the kernels and the numbers of kernels are considered with a variable length scale i.e., the size of the kernels and numbers of channels in convolutional layers are selected dynamically with a variable length scale instead of static size filters and channels. Besides this, the filters are arranged in both ascending and descending order as per the dynamic scale to measure the accuracy of the networks for the classification of environmental sounds. The samples of environmental sound are used from the existing dataset & the 2D pattern vectors of time and frequency are constructed with a spectrogram. The log-scaled Mel-spectrogram technique is used to represent the pre-processed audio data into the time-frequency patches. Thus, a two-dimensional feature vector in the form of TF patches is used as input to the proposed convolutional neural network architectures. Three convolutional neural network architectures have been presented with dynamically selected filters of the receptive field. Thus, the proposed architectures are different from the existing pre-trained architectures due to their variability in the size of filters and number of filters. The proposed third architecture used the maximum variable size kernel as per the scale to distribute the feature map into the convolution layer of three convolution blocks of dynamic size filters. The mini-batch stochastic gradient descent learning with the Adagrad method is reformulated as per the proposed three architectures. The experimental results are obtained for proposed CNNs networks for the sound samples collected from the dataset of UrbanSound8k. The simulation is performed to analyze the effect of change in the variable length scale of filters and size of filters. The role of regularization and optimization are also considered in the performance analysis for the classification. Simulated results exhibit that the proposed architecture of dynamic variable length of filters and channels with reformulated stochastic gradient descent optimizer shows good accuracy for classification. The first proposed architecture has better accuracy than all the other existing models of classification for environmental sound samples. It is also observed that the dynamically sized filters of the receptive field and dynamically sized channels arranged in ascending order as per variable length scale performed better with respect to other proposed and existing models and it reports the classification accuracy of 85.29% for the existing dataset. It reflects that, as the variable scale increases, the size of filters and number of filters also increase, and if it increases in ascending order i.e., the last layer contains a smaller size filter and a smaller number of channels with respect to the first layer then better accuracy is obtained. In the first convolution architecture, this mechanism is implemented and the performance of the network is found better than others. This interesting observation reflects that the relationship of feature extraction with variable size filters arranged in a specific order and the role of redesigned optimizer according to dynamically sized filters are improving the classification accuracy. These proposed architectures of dynamically sized filters of the receptive field and reformulated mini-batch stochastic gradient descent learning with the Adagrad method are applied only for the environmental sounds. In the future, the same models and learning can be applied to the classification of human sounds.

Acknowledgment

The authors thankfully acknowledge the financial support of the Uttar Pradesh government, Lucknow, India in the form of a major research project: 47/2021/606/seventy/4-2020-4(56)/2021.

Funding Information

This research is funded by the Uttar Pradesh government, Lucknow, India in the form of a major research project: 47/2021/606/seventy/4-2020-4(56)/2021.

Author’s Contributions

Manu Pratap Singh: Participated in Conceptualization, methodology, supervision, mathematically modeled and contributed to the written of the manuscript.

Pratibha Rashmi: Participated in data curation, implementation, visualization, results, and validation.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and that no ethical issues are involved.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this study.

Availability of Data and Material

Data collection of voice samples of environmental sounds has been considered from the existing dataset UrbanSound8k.

Conflicts of Interest Statement

The authors whose names are listed immediately below certify that they have no affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers’ bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

Abdoli, S., Cardinal, P., & Koerich, A. L. (2019). End-to-end environmental sound classification using a 1D convolutional neural network. *Expert Systems with Applications*, *136*, 252-263. https://doi.org/10.1016/j.eswa.2019.06.040

Adavanne, S., & Virtanen, T. (2017). Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network. *arXiv preprint arXiv:1710.02998*. https://doi.org/10.48550/arXiv.1710.02998

Bhujel, A., & Pant, D. R. (2017). Dynamic convolutional neural network for image super-resolution. *Journal of Advanced College of Engineering and Management*, *3*, 1-10. https://doi.org/10.3126/jacem.v3i0.18808

Boddapati, V., Petef, A., Rasmusson, J., & Lundberg, L. (2017). Classifying environmental sounds using image recognition networks. *Procedia Computer Science*, *112*, 2048-2056. https://doi.org/10.1016/j.procs.2017.08.250

Bogdanov, D., Wack, N., Gómez Gutiérrez, E., Gulati, S., Boyer, H., Mayor, O., ... & Serra, X. (2013). Essentia: An audio analysis library for music information retrieval. In *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil. [place unknown]: ISMIR; 2013. p. 493-8*. International Society for Music Information Retrieval (ISMIR). https://doi.org/10.1145/2502081.2502229

Bologna, G., & Pellegrini, C. (1998, May). Constraining the MLP power of expression to facilitate symbolic rule extraction. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)* (Vol. *1*, pp. 146-151). IEEE. https://doi.org/10.1109/IJCNN.1998.682252

Botev, A., Lever, G., & Barber, D. (2017, May). Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 1899-1903). IEEE. https://doi.org/10.1109/IJCNN.2017.7966082

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics Paris France, August 22-27, 2010 Keynote, Invited and Contributed Papers* (pp. 177-186). Physica-Verlag HD. https://doi.org/10.1007/978-3-7908-2604-3_16

Bühlmann, P., & Van De Geer, S. (2011). *Statistics for high-dimensional data: Methods, Theory and Applications*. Springer Science and Business Media. https://doi.org/10.1007/978-3-642-20192-9

Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., & Liu, Z. (2020). Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11030-11039). https://doi.org/10.1109/CVPR42600.2020.01104

Choi, K., Fazekas, G., & Sandler, M. (2016). Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*. https://doi.org/10.48550/arXiv.1606.00298

Chu, S., Narayanan, S., & Kuo, C. C. J. (2009). Environmental sound recognition with time frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing*, *17*(6), 1142-1158. https://doi.org/10.1109/TASL.2009.2017438

Cotton, C. V., & Ellis, D. P. (2011, October). Spectral vs. spectro-temporal features for acoustic event detection. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (pp. 69-72). IEEE. https://doi.org/10.1109/ASPAA.2011.6082331

Dai, W., Dai, C., Qu, S., Li, J., & Das, S. (2017, March). Very deep convolutional neural networks for raw waveforms. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 421-425). IEEE. https://doi.org/10.1109/ICASSP.2017.7952190

Darken, C., Chang, J., & Moody, J. (1992, August). Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing* (Vol. *2*, pp. 3-12). Helsinoger, Denmark: Citeseer. https://doi.org/10.1109/NNSP.1992.253713

Deng, S., Han, J., Zhang, C., Zheng, T., & Zheng, G. (2014, May). Robust minimum statistics project coefficients feature for acoustic environment recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 8232-8236). IEEE. https://doi.org/10.1109/ICASSP.2014.6855206

Geiger, J. T., & Helwani, K. (2015, August). Improving event detection for audio surveillance using gabor filterbank features. In *2015 23rd European Signal Processing Conference (EUSIPCO)* (pp. 714-718). IEEE. https://doi.org/10.1109/EUSIPCO.2015.7362476

Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6645-6649). IEEE. https://doi.org/10.1109/ICASSP.2013.6638947

Hoshen, Y., Weiss, R. J., & Wilson, K. W. (2015, April). Speech acoustic modeling from raw multichannel waveforms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4624-4628). IEEE. https://doi.org/10.1109/SLT.2018.8639585

Kim, T., Lee, J., & Nam, J. (2018, April). Sample-level CNN architectures for music auto-tagging using raw waveforms. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 366-370). IEEE. https://doi.org/10.1109/ICASSP.2018.8462046

Kons, Z., Toledo-Ronen, O., & Carmel, M. (2013, August). Audio event classification using deep neural networks. In *Interspeech* (Vol. *2013*, pp. 1482-1486). https://doi.org/10.21437/Interspeech.2013-384

Kukačka, J., Golkov, V., & Cremers, D. (2017). Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*. https://doi.org/10.48550/arXiv.1710.10686

Lezhenin, I., Bogach, N., & Pyshkin, E. (2019, September). Urban sound classification using long short-term memory neural network. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)* (pp. 57-60). IEEE. https://doi.org/10.15439/2019F185

Luo, W., Li, Y., Urtasun, R., & Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *29*. https://proceedings.neurips.cc/paper/2016/hash/c8067ad1937f728f51288b3eb986afaa-Abstract.html

Lydia, A., & Francis, S. (2019). Adagrad-an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci*, *6*(5), 566-568. https://www.scirp.org/(S(czeh2tfqw2orz553k1w0r45))/reference/referencespapers.aspx?referenceid=2912025

Mydlarz, C., Salamon, J., & Bello, J. P. (2017). The implementation of low-cost urban acoustic monitoring devices. *Applied Acoustics*, *117*, 207-218. https://doi.org/10.1016/j.apacoust.2016.06.010

Nair, V., & Hinton, G. E. (2009). 3D object recognition with deep belief nets. *Advances in Neural Information Processing Systems*, *22*. https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=118367363EA44B29C43721319DD43787?doi=10.1.1.167.3450&rep=rep1&type=pdf

Nemirovski, A., Juditsky, A., Lan, G., & Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, *19*(4), 1574-1609. https://doi.org/10.1137/070704277

Phan, H., Hertel, L., Maass, M., Mazur, R., & Mertins, A. (2015, September). Representing nonspeech audio signals through speech classification models. ISCA. https://doi.org/10.1145/2647868.2655045

Piczak, K. J. (2015a, October). ESC: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia* (pp. 1015-1018). https://doi.org/10.1145/2733373.2806390

Piczak, K. J. (2015b, September). Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)* (pp. 1-6). IEEE. https://doi.org/10.1109/MLSP.2015.7324337

Pons, J., & Serra, X. (2019, May). Randomly weighted cnns for (music) audio classification. In *ICASSP 2019-2019 IEEE International Conference On Acoustics, Speech and Signal Processing (ICASSP)* (pp. 336-340). IEEE. https://doi.org/10.1109/ICASSP.2019.8682912

Radhakrishnan, R., Divakaran, A., & Smaragdis, A. (2005, October). Audio analysis for surveillance applications. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* (pp. 158-161). IEEE. https://doi.org/10.1109/ASPAA.2005.1540194

Ravanelli, M., & Bengio, Y. (2018, December). Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)* (pp. 1021-1028). IEEE. https://doi.org/10.1109/SLT.2018.8639585

Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2016). Introduction to Anaconda and Python: Installation and setup. *Quant. Methods Psychol*, *16*(5), S3-S11. https://doi.org/10.20982/tqmp.16.5.S003

Ruan, X., Liu, Y., Li, B., Yuan, C., & Hu, W. (2021, May). DPFPS: Dynamic and progressive filter pruning for compressing convolutional neural networks from scratch. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. *35*, No. 3, pp. 2495-2503). https://doi.org/10.1609/aaai.v35i3.16351

Salamon, J., & Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, *24*(3), 279-283. https://doi.org/10.1109/LSP.2017.2657381

Salamon, J., Jacoby, C., & Bello, J. P. (2014, November). A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 1041-1044). https://doi.org/10.1145/2647868.2655045

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. https://doi.org/10.48550/arXiv.1409.1556

Srisuk, S., Boonkong, A., Arunyagool, D., & Ongkittikul, S. (2018, March). Handcraft and learned feature extraction techniques for robust face recognition: A review. In *2018 International Electrical Engineering Congress (iEECON)* (pp. 1-4). IEEE. https://doi.org/10.1109/IEECON.2018.8712272

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958. https://jmlr.org/papers/v15/srivastava14a.html

Wang, J., Li, C., Xiong, Z., & Shan, Z. (2014). Survey of data-centric smart city. *Journal of Computer Research and Development*, *51*(2), 239-259. https://doi.org/10.7544/issn1000-1239.2014.20131586

Zhang, X., Zou, Y., & Shi, W. (2017, August). Dilated convolution neural network with LeakyReLU for environmental sound classification. In *2017 22nd international conference on digital signal processing (DSP)* (pp. 1-5). IEEE. https://doi.org/10.1109/ICDSP.2017.8096153

Zhu, Z., Engel, J. H., & Hannun, A. (2016). Learning multiscale features directly from waveforms. *arXiv preprint arXiv:1603.09509*. https://doi.org/10.21437/Interspeech.2016-256