

# Classification of labeled audio events using Deep CNN model

Dharm Mehta

**Abstract—**In this report I am suggesting an approach to effectively distinguish and classify knowledge from labeled web audio data. Due to great accuracy in classification of images by the deep CNN model, it's been also embraced for sound processing. Like in the case of image processing, contiguous region in image is occupied by object, in the same way in sound processing there is a spectrogram, which contains patterns of frequency waves and is unique to particular audio. So, we need to use a neural net which can handle well the time-frequency properties of sound. With the help of a convolutional neural network based architecture, we can detect a sound event and also classify the unlabeled audio data. The technique which can be used on the CNN model to propagate is Layer-wise Relevance Propagation. The goal of this project is to analyze the different problems while classifying the audio data encountered and to think about possible improvements to overcome them.

## I. INTRODUCTION

Sound is an important part in our life, as it helps us to express our feelings, helps in interacting with environment. Hence, it is critical for the success of artificial intelligence that machines or computers are able to comprehend sounds as humans do. This has led to considerable interests in sound event detection and classification research in recent years. Also the application of sound classification has increased in past few years in security and surveillance purpose, in handling, storing and recovering the multimedia etc., also in smart assistants and many more.

Recently, there has been rapid development in field of deep learning which aims at learning more complex, higher level representations. Due to huge success of CNN model in image recognition, so CNN has also been applied in sound classification for ability to model spectral correlations and reduce spectral variations.

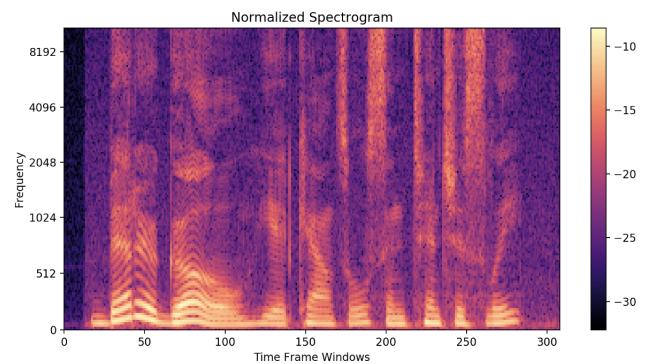
But the major problem in sound classification is due to enough availability of huge datasets. As designating or identifying the sound in an recording is a tough task. Also one of the other major issues is the uncertainty in starting and terminating of the sound event which varies from person to person. To tackle the difficulty the datasets nowadays are weakly labeled for sound detection. Even though due to availability of data being weakly labeled, but still in order to make large datasets it is very tedious and time consuming. Like in the case of image classification, by transferring the knowledge from the previously trained model from large dataset can make our task easier in sound classification.

Owing to lack of large data or dataset, it has been proposed to use transfer knowledge from visual models for sound event recognition. So the CNN models are trained to visualize the objects in order for it to teach a feature extractor network for audio.

The fundamental way to solve the above discussed problem of learning is to plot optimal representation of data without converging to a local minimum, and is to introduce sparsity in the learning algorithm. In context to processing sound waves, the receptive neurons present in the primary cortex of auditory canal in mammals are localized, and are sparsely linked, oriented and organized according to frequency, resulting in a type of sparse architecture.

There are many different techniques to represent signals for sound detection and classification, but they are mainly classified into only time and time-frequency methods. And both of these methods are not generally used in their basic form, in order to amplify the performance of the model, they undergo different mathematical operations to draw out properties that can help in sound classification.

The most common mathematical tool for time-frequency representation of sound is short time Fourier transform. In this method, for a particular fixed instance of time samples of fixed window size are Fourier transformed. For the classification part only the magnitude components are kept and phase part is discarded. The overall number of sound samples in a window denotes the frequency. And to overcome the higher frequency waves windowing technique is used. And finally to obtain the spectrogram, the different amplitude of frequency are combined to each other by using a set of band pass filter and is placed next to each other.



In this project I have trained the CNN model on the dataset of 523 audio events, which are of fixed lengths. The main goal for sound classification was to optimize the learning of sparsely network. I also tried training it for variable lengths to make it possible for transferable knowledge but was not able to achieve great accuracy.

## II. BACKGROUND

Audio classification is most generally based on representations of the data in the spectrogram but even for raw data successful classification has been done. The neural network architectures like VGG Net, AlexNet etc. which were designed specially for image classification are also nowadays used to represent spectrograms.

The sound recordings from the dataset are sampled at 43.2 KHz sampled frequency. 127 Mel bands of frequency are used. And also size of window is 22 ms and there is an overlap of 10.5 ms which is used for obtaining Mel features.

With the CNN mode, we need to use MFCC for processing the raw audio data and also use LSTM to update the model.

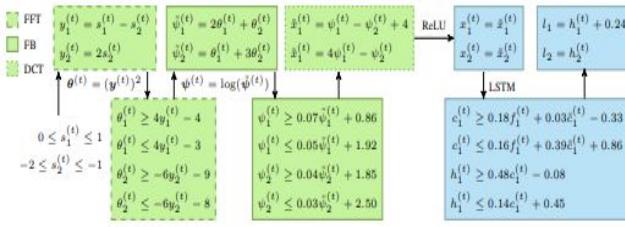


Fig. 1. Here the green part is MFCC part and the blue is the LSTM part.

**Mel-Frequency Cepstral Coefficients (MFCC):-** The MFC coefficient is found to model non-linear human acoustic perception as power spectrum filters based on certain frequencies, called Mel-frequencies. For each individual Mel-frequency, the ultimate result of transformation will contain vector coefficients whose component contains logarithmic values of filtered spectra. This resulting vector is a feature representation of the original signal and can now be used in a downstream task such as audio classification.

For preprocessing of audio, MFCC is calculated in following steps:

### 1. Pre-emphasizing and Windowing: $\mathbf{Y} = \mathbf{S}(\mathbf{I}_N - c_{pe}\mathbf{I}_N^{+1}) \odot \mathbf{H}$

Transform the signal with the pre-emphasizing and applying the Hamming window. Here,  $\mathbf{I}_N^{+1} \in \mathbb{R}^{N \times N}$  is the shifted diagonal identity matrix ( $I_{i,j}^{+1} = 1$  if  $i+1 = j$ , otherwise 0),  $\mathbf{H} \in \mathbb{R}^{T \times N}$  is the Hamming window, and  $c_{pe}$  is the pre-emphasizing constant.

### 2. Power Spectrum of Fast Fourier Transform (FFT): $\Theta = (\mathbf{Y}\mathbf{W}) \odot (\mathbf{Y}\mathbf{W})^T$

Perform FFT on the windowed data and square it to get the real-value spectrum. We can denote FFT on discrete domain (DFT) with the multiplication of  $\mathbf{Y}$  and  $\mathbf{W} \in \mathbb{C}^{N \times N/2}$ .

### 3. Filter Bank Log Energy: $\Psi = \log(\Theta\Lambda)$

Apply the Mel frequency filter bank to the power spectrum and get the log value of them.  $\Lambda \in \mathbb{R}^{N/2 \times p}$  is the filter bank given the number of filters  $p$ , and log is applied entry-wise.

### 4. DCT(Discrete Cosine Transformation): $\mathbf{X} = \Psi\mathbf{D}$

Perform DCT on the previous result. Again, this can be formulated using matrix multiplication. We use the resulting  $\mathbf{X} = [\mathbf{x}^{(1)} \cdots \mathbf{x}^{(T)}]^T$  as the input for the neural network.

**Long-Short Term Memory (LSTM):-** These networks are type of recurrent neural network capable of learning order dependence in sequence prediction problems. They are used to update in sound recognition.

The definition of the update for LSTM unit is:-

For training the CNN model,a sound recording is divided into small segments and are put inside the network one after

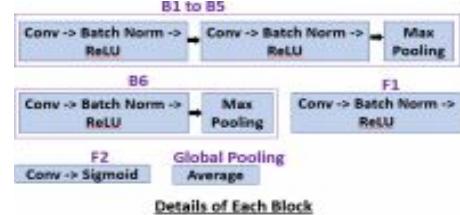
$$\begin{aligned} f_0^{(t)} &= [\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}]W_f + b_f & o_0^{(t)} &= [\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}]W_o + b_o \\ i_0^{(t)} &= [\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}]W_i + b_i & \tilde{c}_0^{(t)} &= [\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}]W_{\tilde{c}} + b_{\tilde{c}} \\ c^{(t)} &= \sigma(f_0^{(t)}) \odot c^{(t-1)} + \sigma(i_0^{(t)}) \odot \tanh(\tilde{c}_0^{(t)}) & h^{(t)} &= \sigma(o_0^{(t)}) \odot \tanh(c^{(t)}) \end{aligned}$$

where  $[\cdot, \cdot]$  is the horizontal concatenation two row vectors,  $W$ , and  $b$ , are kernel and bias of the cell, respectively. At timestep  $t$ , vectors  $f_0^{(t)}, i_0^{(t)}, o_0^{(t)}, \tilde{c}_0^{(t)}$  represent pre-activations of the forget, input, and output gate, respectively, and the pre-calculation of cell state. Cell state  $c^{(t)}$  and hidden state

the another and labels of the whole recordings are made the same as the selected labels for all fragments. This procedure for training will be referred to as strong label assumption training (SLAT).

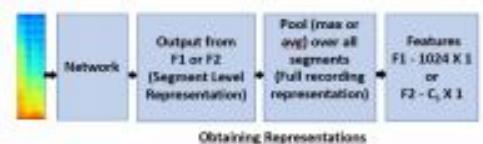
## III. NETWORK ARCHITECTURE

The framework consists of the deep CNN network for the labeled sound. The first five Blocks consists of two layers which are convoluted (with batch is normalized) which is then followed by a max pooling. The sixth block consists of one convolutional layer, followed by a max pooling layer. Re LU (max(0, x))[24] activation is used in all cases. For convolutional layers in all six blocks, 3 × 3 filters are used. Stride and padding are fixed to 1. The number of filters used in convolutional layer(s) of blocks B1 to B6 are, B1 : 16, B2 : 32, B3 : 64, B4 : 128, B5 : 256, B6 : 512. Max pooling are done over a 2 × 2 window, with a stride of 2 by 2. F1 is also a convolutional layer with ReLU activation. 1024 filters of size 2 × 2 are used with a stride of 1. No padding is used in F1.

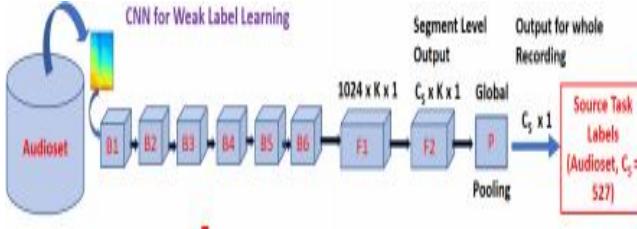


F2 is the secondary output layer, a convolutional layer of Cs filters of size 11 and sigmoid output. This layer produces segment level output (Cs K 1), where Cs is the number of classes (in source task) and K is the number of segments. The segment level outputs are aggregated using a global pooling layer to produce Cs1 dimensional output for the whole recording.

The network will scan through the complete input and will produce results corresponding to segments of 128 input-frames moving by 64 frames. For example, an input logmel spectrogram consisting of 896 logmel-frames, that is X ∈ R 896128 (128 Mel frequency bands as stated before), will produce K=13 segments at F1 and F2.



The labeled sound in dataset have labels for the complete recording, the outputs at the fragment level are needed to obtain the full recorded level output. To calculate the loss, it is found out in regards to the output at corresponding recorded level. The loss is then computed with respect to this recording level output. The network is a VGG based style



CNN for label learning of audio. The network is designed to control the fragment size and hop size. For NS, these are 128 (1.5 s) and 64 (0.75s) frames respectively in spectograms.

The most important point is that fragmented level outputs can give temporary localization of events.

For the CNN architectures to be used for SLAT it should contain dense layers are fully connected. But for sound recording of varying length these layer must be completely convolutional to make it best fit to transfer the knowledge.

#### IV. PROPAGATION

In many sector and field of research where simplification of the problem is very important, there linear models are still used for training and classification due to its easily explainable architecture. But if we look for higher complex queries, here the linear model will have poor accuracy in prediction.

So to overcome this problem a method called LRP is used which will propagate each layer according to its relevance and it will also potentially scale the complex layers along with making it more simpler to understand. In this method it will get all the relevance values  $R_i$  that are related to the elements  $i$  present in the input  $x$ . LRP uses top-down approach as it will start propagating from the last hidden layer which is close to output towards the input through each layer one by one and calculating values of relevance  $R_i$ . Each  $R_i$  will describe the contribution by every input or hidden variable  $x_i$  that has affected the final prediction.

Layer-wise relevance conservation

$$\sum_i R_i = \dots = \sum_i R_i^{(l)} = \sum_j R_j^{(l+1)} = \dots = f(x)$$

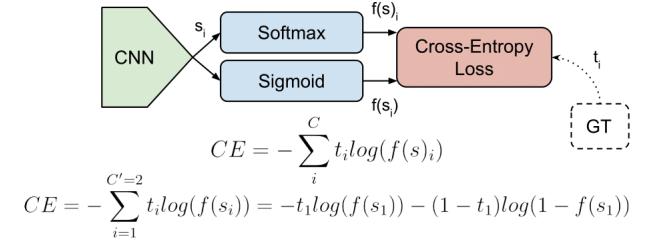
And then the important part of this method is to redistribute the relevance values  $R_j$  from the uppermost layer to back towards the input layer  $x_i$ , and also calculate the contributions in activating the output neuron  $j$  by all the inputs.

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$$

For the output neuron  $R_j$  the initial value of relevance is decided by us and its attributes are defined according to the position of the layer and also its types. And when  $R_k = 0$ , it means that there will be no contribution fro that variable.

#### V. LOSS FUNCTION

As there are multiple labels training loss in the dataset, I have used binary cross entropy loss.



This loss function is also known as Sigmoid cross entropy loss. As other than the cross entropy loss it also contains activation by sigmoid. And also the speciality of this function is that the loss is calculated for each resultant vector component of the CNN model and is totally independent i.e does not have any impact on other components , which is not in the case of softmax function. This is the main reason on why it is used for classification in multi-label models, where one label must not affect the output of the other.

And another type of cross- entropy loss used is Focal loss that calculates the loss based in the error in classification by taking the weight of each sample according to their respective contribution. The concept behind it is if the model has correctly identified the sample with the label, then that sample will have very less contribution to the overall loss. And they also consider the weight of the sample in the contribution of each and every samples in the total loss.

$$FL = -\sum_{i=1}^{C=2} (1 - s_i)^\gamma t_i \log(s_i)$$

case:

when gamma =0 , Focal loss will be equal to binary cross entropy loss.

$$\text{For } (1 - s_i)^\gamma : -$$

and gamma is greater than equal to zero than this term will act as the modulating factor to decrease the effect of correctly classified sample on the loss.

The loss is also described as:-

$$FL = \begin{cases} -(1 - s_1)^\gamma \log(s_1) & \text{if } t_1 = 1 \\ -(1 - (1 - s_1))^\gamma \log(1 - s_1) & \text{if } t_1 = 0 \end{cases}$$

And its gradient expression is:-

$$\frac{\partial}{\partial s_i} (FL(f(s_i))) = (1 - f(s_i))^\gamma (\gamma f(s_i) \log(f(s_i)) + f(s_i) - 1) \quad \text{if } t_1 = 1$$

## VI. OPTIMIZATION

To optimize the result Adam optimizer is used, this algorithm is used to update weights iteratively while training the data in place of the gradient descent.

The advantage of ADAM optimizer are:-

- It is very easy to implement.
- very efficient
- Less memory space required.
- Suitable for large number of data and parameters
- Optimum for objectives that are non-stationary.
- Gives good results on noisy gradients

There is only a single learning rate for updating all the weights and it remains constant throughout training the data by using gradient descent. But using ADAM the benefit is that it uses combine ability of adaptive gradient algorithm and root mean square propagation.

### A. Root mean square propagation:-

This algorithm does very well on the noisy problems like the online or non-stationary dataset as it maintains the learning rate based on the mean of gradients per parameter.

$$\begin{aligned}\nu_t &= \rho\nu_{t-1} + (1 - \rho) * g_t^2 \\ \Delta\omega_t &= -\frac{\eta}{\sqrt{\nu_t + \epsilon}} * g_t \\ \omega_{t+1} &= \omega_t + \Delta\omega_t\end{aligned}$$

### B. Adaptive gradient algorithm:-

This algorithm works well for sparse gradients and it balances the learning rate per the weights of the parameter.

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

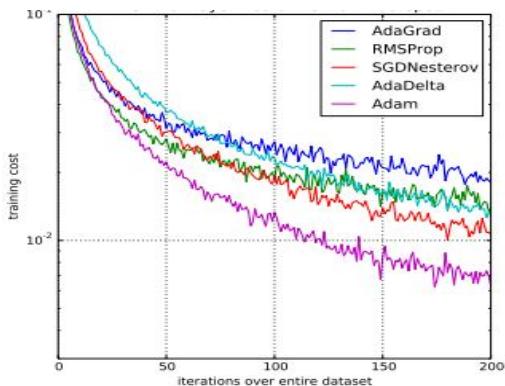


Fig. 2. Performance of other optimizer with respect to Adam

So in Adam algorithm for maintaining the learning rate instead of using the the mean average of the first moment, it uses the incentered variance of the second moment. It will calculate an exponentially moving average of the gradient

and also the squared value of gradient, and the parameters like  $\beta_1$  and  $\beta_2$  will control the decay rates of these moving averages.

## VII. RESULTS

So the performance for sound event detection and classification on all 525 sound events in dataset for which labeled data is available. And after comparing performance of  $s$  with SLAT, we are easily able to determine that SLAT is pretty similar to  $s$  except that F1 and F2 are now fully connected layers of size 1024 and  $C_s$ . Training is done with fixed size segments of 128 logmel frames as inputs, segments overlap by 64 frames. Loss is computed for each input segment by using recording level labels.

After doing all the coding with Pytorch . Adam optimization was used for training the parameters in the networks. And validation set is used for tuning parameters and selecting the best model. Learning rate during this process is fixed to 0.0002 and updates are done for 50 epochs, after which the network is used to obtain representations.

Linear SVMs are then trained on the representations obtained from different methods. The slack parameter C in SVMs is tuned by cross validation on the training set.

Dataset consists of labels for 525 audio events on videos. We use the balanced training set for training NS. Balanced training set provides a total of around 22, 000 training audio recordings, with at least 59 examples per class. However, due to multi-label nature of the data the actual number of examples for several classes is much higher. A small subset from Unbalanced set of dataset is used as the validation set in experiments. Results are reported on the full evaluation set of dataset, which has around 20,000 test audio recordings, again with at least 59 examples per class.

Area under ROC curves (AUC) and Average Precision (AP) for each class are used as evaluation metrics.

MAUC		MAP		Train Time		Inference Time	
$N_S^{stat}$	$N_S$	$N_S^{stat}$	$N_S$	$N_S^{stat}$	$N_S$	$N_S^{stat}$	$N_S$
0.915	0.927 (+4.3%)	0.167	0.213 (+27.5%)	1.0	0.61 (-39%)	1.0	0.67 (-33%)
<b>Lowest 10</b>						<b>Highest 10</b>	
Event	$N_S^{stat}$	$N_S$	Event	$N_S^{stat}$	$N_S$		
Scrape	0.0058	0.0092	Music	0.728	0.749		
Crackle	0.0078	0.0097	Siren (Civil Defense)	0.671	0.641		
Man Speaking	0.0080	0.0202	Bagpipes	0.646	0.786		
Mouse	0.0092	0.0368	Speech	0.631	0.661		
Buzz	0.0095	0.0077	Purr (Cats)	0.575	0.600		
Squish	0.0102	0.0122	BattleCry	0.575	0.651		
Gurgling	0.0111	0.0125	Heartbeat	0.559	0.569		
Door	0.0115	0.0685	Harpsichord	0.544	0.630		
Noise	0.0116	0.0107	Ringing (Campanology)	0.538	0.690		
Zipper	0.0121	0.0161	Timpani	0.538	0.528		
Mean	0.0097	0.0203	Mean	0.600	0.651		

In the above table, it shows Mean AUC (MAUC) and Mean AP (MAP) over all 525 classes in dataset. An absolute improvement of 1.2 ( 1.3% relative) in MAUC and 4.6 (27.5% relative) in MAP is obtained using NS. The top right table shows relative computational times, normalized for comparison. NS is 33% faster on an average during inference. Hence, more suitable for real applications. During

training, on an average it is 39% faster for 1 full pass over training data. The overall accuracy achieved is 82.5%

Performance of all classes are available here. Bottom table in Tables shows comparison for 10 sound events for which SLAT achieved least and highest APs. For low performance classes, on an average NS doubles the AP (0.0097 to 0.0203). For easier sound classes 8.5% relative improvement is obtained using  $s$ .

### VIII. FUTURE PROSPECTS

The given model is only trained for audio events of fixed time length, but if it is designed in such a way that it can even work for the events with variable time length, then it can even be used for transferred knowledge.

To achieve the above task, there are many methods like:-

- 1. In the already present architecture, if we can make a new network that will directly link the F1 block to the output task. So, doing this F2 part is being replaced by a new convolutional layer (FT) of CT filters. Parameters in F1 and FT are then updated using the training set of the output task. We will call this network n1.
- 2. Another approach could be that altogether after the pooling layer in NS if we add a newly layer of size CT that is fully connected to FT. And here also like the above method only the F1, F2 and FT will be updated during training. The reason for doing this network is that instead of learning in fragment level it will try to map the full audio event. And let this network be called n2.
- 3. The last approach would be to completely add a new layer of size CT and it is convoluted to CT. And here we will update all the three F1, F2 and FT layers. Here the plan is to go to the output, it will catch the specific task and then it will transition to F2. We will call this network as n3.

The conclusion is that for all the three newly formed network n1, n2 and n3: if the target problem has a multi-label task, then the activation in final layer is kept as sigmoid and loss function is the same that is used for this model. But if we also have the problem where it has only one label then the activation used will be softmax and for loss function we will use categorical cross entropy loss. Thus these methods can be used to efficiently handle the sound events with variable length.

There is quite a possibility that the dataset can contain weak labels then also the above methods will help in classification. And even if the dataset is very small the SVM can be trained accordingly to handle the task.

Another improvement that can be done is to increase the efficiency of the model. After going through many literature I have come to the point that in place of using a CNN model for sound classification, we can use a CLNN model or a MCLNN model. They are specially built in order to take advantage of sound in the time-frequency domain.

The conditional neural net is non-linear in nature for time but it is linear in nature for frequency. And so it

will condition its inference at a fixed time based on the surrounding time slices.

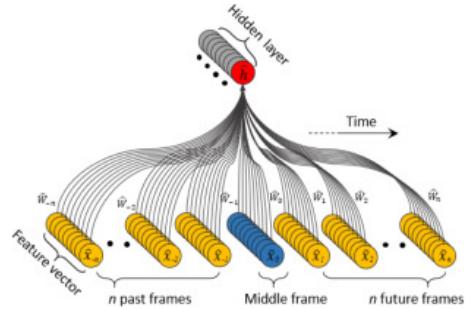


Fig. 3. CLNN

Whereas, the masked conditional neural net has a filter like attribute which will give inference in a controlled manner within the given network. And also, the added advantage using this is it will automate any exploration which is concurrent in nature and has several feature like different combinations, which are analogous to hand-picking the perfect combination of features that can be helpful in the recognition task.

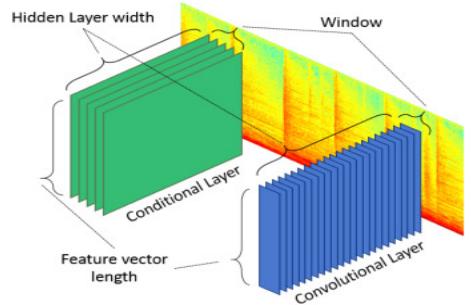


Fig. 4. MCLNN

$\widehat{\mathcal{X}}$  : two-dimensional input segment of size  $q$   
 $\widehat{\mathcal{Y}}$  : two-dimensional output segment of size  $q - 2n$   
 $\widehat{\mathcal{W}}$  : three-dimensional weight tensor of shape  $[2n + 1, t, e]$   
 $\widehat{\mathcal{M}}$  : two-dimensional mask of binary patterns of shape  $[t, e]$ .  
 for  $u < 2n + 1$   
 $\widehat{\mathcal{Z}}_u = \widehat{\mathcal{W}}_u \circ \widehat{\mathcal{M}}$  (weight matrix masked with the binary mask)  
 $\widehat{\mathcal{X}}' = \widehat{\mathcal{X}}' + \widehat{\mathcal{X}}_{[u-u+(q-2n),l]} \cdot \widehat{\mathcal{Z}}_u$  (slice of  $\widehat{\mathcal{X}}$  from  $u$  to  $u+(q-2n)$ )  
 $\widehat{\mathcal{Y}}' = f(\widehat{\mathcal{b}} + \widehat{\mathcal{w}}')$  (transfer function  $f$ , bias  $\widehat{\mathcal{b}}$ )

Fig. 5. Algorithm of MLCNN

### IX. CONCLUSION

I have built a VGG type CNN based model which can classify the audio events in the given dataset within a very quick time. Due to its efficient architecture it can handle large datasets which needs to be of fixed time length. Although the accuracy is only 82.5% but it is all in all pretty good for a sound classifier. In addition to the above achieved result I did train the model for varying length, and got less accuracy. With few modifications in the model these can be used for real time application.

## REFERENCES

- [1] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
- [2] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017.
- [3] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
- [4] Justin Salamon and Juan Pablo Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24.
- [5] Karol J Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*. ACM, 2015.
- [6] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2009.
- [7] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.
- [8] A. Temko, R. Malkin, C. Zieger, D. Macho, C. Nadeu, and M. Omologo, "Clear evaluation of acoustic event detection and classification systems," in *International Evaluation Workshop on Classification of Events, Activities and Relationships*. Springer, 2006.
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.
- [10] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for ReLU networks. In Proc. International Conference on Machine Learning (ICML), volume 80.
- [11] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw waveform cldnns. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [12] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In *Proc. International Joint Conference on Artificial Intelligence, IJCAI*
- [13] M. Popescu, A. Mahnot, Acoustic fall detection using one-class classifiers, in: *Annual International Conference of the Engineering in Medicine and Biology Society, EMBC*, 2009.
- [14] T. Lidy, A. Rauber, A. Pertusa, J.M. Inesta, Improving genre classification by combination of audio and symbolic descriptors using a transcription system, in: *International Conference on Music Information Retrieval*, 2007.
- [15] Hinton G.E., Salakhutdinov R.R. Reducing the dimensionality of data with neural networks Science, 313 (2006).
- [16] LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition Proc. IEEE, 86 (1998).
- [17] Oppenheim A.V., Willsky A.S., Nawab S.H. *Signals Systems* Prentice-Hall, USA (1996)