Analysis on performances of the optimization algorithms in CNN speech noise attenuator

Haengwoo Lee
Namseoul University, South Korea

Article Info

Article history:

Received Aug 1, 2021
Revised Nov 29, 2021
Accepted Dec 6, 2021

Keyword:

Noise attenuation
Deep learning
Convolutional neural network
Error back propagation
Keras Library

ABSTRACT

In this paper, we studied the effect of the optimization algorithm of weight coefficients on the performance of the CNN (Convolutional Neural Network) noise attenuator. This system improves the performance of the noise attenuation by a deep learning algorithm using the neural network adaptive predictive filter instead of using the existing adaptive filter. Speech is estimated from a single input speech signal containing noise using 64-neuron, 16-filter CNN filters and an error back propagation algorithm. This is to use the quasi-periodic nature of the voiced sound section of the voice signal. In this study, to verify the performance of the noise attenuator for the optimization algorithm, a test program using the Keras library was written and training was performed. As a result of simulation, this system showed the smallest MSE value when using the Adam algorithm among the Adam, RMSprop, and Adagrad optimization algorithms, and the largest MSE value in the Adagrad algorithm. This is because the Adam algorithm requires a lot of computation but it has an excellent ability to estimate the optimal value by using the advantages of RMSprop and Momentum SGD

Corresponding Author:

Haengwoo Lee
Namseoul University
91, daehak-ro, cheonan-si, chung-nam, South Korea 82-10-2693-5563
Email: haengwoolee@hanmail.net

1. INTRODUCTION

Noise attenuation is to attenuate noise included in speech, and various studies have been conducted on noise attenuation technology so far. As noise attenuation methods, there are the spectrum subtraction method [1,2] and the Wiener filter method [3,4] based on the short-term spectrum estimation. These methods subtract the spectrum of noise estimated from the input speech signal or estimate the clear speech spectrum, and are advantageous when the noise and statistical characteristics of the speech signal are known in advance. Another method is to use a Comb filter [5] or an adaptive filter [6, 7] using the quasi-periodic characteristics of the speech signal. The Comb filter method is used for noise having a specific frequency band, and the adaptive filter method has a function to automatically adjust the filter coefficients without knowing the statistical characteristics of the noise in advance. A single-input adaptive noise attenuator with one sensor receives a voice signal from one microphone and estimates the voice signal using the quasi-periodic characteristics of the voiced sound section.

Recently, a deep learning model is making great achievements as a technology that can learn using many hidden layers based on neural networks has been developed. By using the error back propagation algorithm to train multi-layer neural networks, even deep neural networks composed of many layers can be trained [8]. CNN [9] has finally proven to be a reliable tool for generalization of real world noise attenuation problems [10]. CNN is the most widely used deep learning model at present and can estimate the characteristics of speech well. In 2016, a model based on SNR (Signal to Noise Ratio)-aware CNN for speech enhancement was published [11]. This CNN model can efficiently process local temporal and spectral speech. Thus, the model effectively separates speech and noise from the input signal. Two SNR recognition algorithms have been proposed using CNNs to improve the generalization ability and accuracy of these models. The first algorithm incorporates a multi-task learning framework. Given a noisy speech as input to the model, the algorithm mainly reconstructs the noise-free speech and estimates the SNR level. The second algorithm does SNR adaptive noise attenuation. This algorithm first calculates the SNR level. Then, based on the calculated SNR level, an SNR-dependent CNN model is selected to reduce the noise. The proposed two SNR-aware CNN models outperform the simple deep neural network. In 2017, a CNN model for complex spectrogram enhancement was proposed to solve the phase estimation difficulties [12]. The proposed model restores clean real and virtual spectrograms from noisy spectrograms. This spectrogram is used to generate speech with very accurate phase information. The basic idea is that any signal can be represented as a function of real and virtual spectrograms.

Optimization algorithms [13, 14, 15, 16] for updating the weight coefficients of CNN filters include Stochastic Gradient Descent(SGD), Momentum SGD, Nesterov momentum SGD, Adagrad, RMSprop, and Adam. In this study, we propose the best performing algorithm by examining the effect of the optimization algorithm on the performance when noise is attenuated using the deep learning algorithm of the CNN neural network filter instead of the adaptive filter of the adaptive noise attenuator.
The content of this thesis is about the adaptive noise attenuator in Section II, the linear prediction of speech signals in Section III, the structure of the CNN neural network filter in Section IV, and the update algorithm of weight coefficients in Section V. And in Section VI, the simulation of the optimization algorithm and its results are described, and finally, a conclusion is drawn in Section VII.

2. ADAPTIVE NOISE ATTENUATOR

Figure 1 is a single-input noise attenuator that estimates the current voice sample from signals delayed by more than one sample by an adaptive prediction method using the quasi-periodic characteristics of the voice signal. A speech signal delayed by one or two pitches has a high correlation, but has little correlation with the white noise component. That is, the voice signal converges so as to have the least squares error of the target value as a relationship independent of noise.

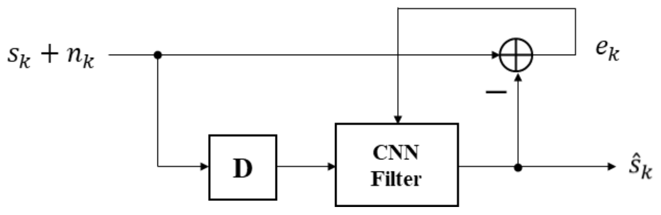

Figure 1. Single input noise attenuator

The output of the CNN filter estimates the characteristics of the voice signal included in the input signal, and this signal is subtracted from the input signal to become an error. This error signal is used as an update signal to update the weight of the CNN filter, and the average power is the same as Equation (1).

$$E\{e_k^2\} = E\{(s_k - \hat{s}_k)^2 + 2(s_k - \hat{s}_k)n_k + n_k^2\} \tag{1}$$

Here, $E\{\cdot\}$ is the average value, and assuming that the voice signal and noise are independent of each other,

$$E\{e_k^2\} = E\{(s_k - \hat{s}_k)^2 + n_k^2\} \tag{2}$$

Since the noise energy in an arbitrary section is a fixed value,

$$min(E\{e_k^2\}) = min(E\{(s_k - \hat{s}_k)^2\}) + E\{n_k^2\} \tag{3}$$

Minimizing $E\{e_k^2\}$ is to minimize the estimation error of the voice signal $E\{(s_k - \hat{s}_k)^2\}$, and the output of the filter $\hat{s}_k$ at this time estimates the voice signal best. Therefore, the minimization of $E\{(s_k - \hat{s}_k)^2\}$ means to minimize $E\{(n_k - e_k)^2\}$ and the error signal e_k is estimated the noise.

3. LINEAR PREDICTIVE CODING ANALYSIS OF SPEECH SIGNAL

Linear predictive coding analysis is a method used in various fields such as speech analysis and synthesis, and can accurately express the characteristics of speech spectrum with a relatively small number of parameters. Assuming that the speech sample at discrete time k is s_k, and the predicted value of the speech sample at time k is $\hat{s}_k$, it can be expressed as Equation (4) [17].

$$\hat{s}_k = c_1 s_{k-1} + c_2 s_{k-2} + \cdots + c_N s_{k-N} = \sum_{n=1}^{N} c_n s_{k-n} \tag{4}$$

Therefore, the present value of the voice signal can be predicted from the previous values of the previous values from Equation (4). Therefore, assuming the prediction error representing the difference between the actual input value and the predicted value is e_k, it can be expressed by Equation (5).

$$e_k = s_k - \hat{s}_k = s_k - \sum_{n=1}^{N} c_n s_{k-n} \tag{5}$$

Where c_n is the linear prediction coefficient. Therefore, the LPC coefficient is calculated so that the mean squared value of e_k is minimized. In this paper, the speech signal is estimated using the unique low-frequency spectrum structure of voiced sounds.

4. STRUCTURE OF CNN NEURAL NETWORK FILTER

The neural network filter in Fig. 2 used in this paper shows the three-layer structure when 16 CNN filters are used.

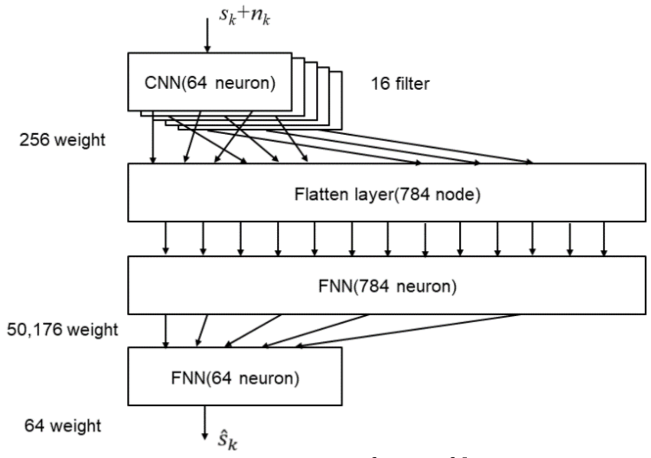

Figure 2. Structure of CNN filter

The CNN layer of the first layer consists of 64 neurons and 16 feature filters, and the size of the kernel is 16 samples, and a kernel exists at every sample interval. The input signal consists of 64×16 data for every sample, and ReLU is applied as an activation function at the output. The output of the CNN layer is flattened in one dimension through the next Flatten layer and spreads out to 49×16 = 784 nodes. These signals are input to the Fully-connected Neural Network (:FNN) layer with 784 neurons, and the ReLU function is applied again at the output. Then, it goes through the last layer, the FNN layer with 64 neurons, and is output as one signal. To reduce the amount of computation, the batch size was set to 30 and the bias parameter of each layer was omitted. The weight parameters to be calculated in this model are 256 (=16×16) in the CNN layer, 50,176 (=784×64) in the hidden layer, and 64 in the output layer, for a total of 50,496. The weight update algorithm uses Adam and the error backpropagation algorithm. This system is classified as supervised learning and prepares training data and learning target values with single input data.

5. THE ALGORITHM FOR WEIGHT COEFFICIENTS UPDATING

A multi-layer perceptron has the structure of a multi-layer neural network having one or more hidden layers. A multi-layer perceptron consisting of an input layer with l input neurons, a hidden layer with m hidden neurons, and an output layer with n output neurons, the values of the input neurons are represented by the l-dimension vector $x = [x_1, x_2, \cdots, x_i, \cdots, x_l]$, the values of the hidden neurons are represented by the m-dimension vector $a^1 = [a_1^1, a_2^1, \cdots, a_j^1, \cdots, a_m^1]$, and the values of the output neurons are represented by the n-dimensional vector $y = [y_1, y_2, \cdots, y_k, \cdots, y_n]$. The weight between the input layer and the hidden layer is represented by w_{ij}^1, the weight between the hidden layer and the output layer is expressed by w_{jk}^2, and bias is omitted. Also, the weighted sum inputted to the j-th hidden neuron is called u_j^h, the weighted sum inputted to the k-th output neuron u_k^o, and the activation function of the hidden neuron uses the ReLU function denoted by $\emptyset_{relu}$, and the output neuron does not use the activation function.

Therefore, the output values of the hidden neuron and the output neuron can be expressed by Equations (6) and (7).

$$a_j^1 = \emptyset(u_j^h) = \emptyset_{relu}\left(\sum_{i=1}^{l} w_{ij}^1 x_i\right) \tag{6}$$

$$y_k = u_k^o = \sum_{j=1}^{m} w_{jk}^2 a_j^1 \tag{7}$$

If we denote all weights as a parameter θ, we can express the value of the k-th output neuron as a function $f_k(x,\theta)$ given the input x.

$$f_k(x,\theta) = y_k = \sum_{j=1}^{m} w_{jk}^2 \emptyset_{relu}\left(\sum_{i=1}^{l} w_{ij}^1 x_i^1\right) \tag{8}$$

The error backpropagation learning algorithm[18] is an algorithm for learning the multi-layer perceptron, and the supervised learning of the multi-layer perceptron defines an error function that is the difference between the values output by the multi-layer perceptron and given the learning target value. When the learning data and the target output value are given as a pair of input and output orders $(x_i, t_i)(i = 1, \cdots, N)$, the error for the whole learning data X can be defined as a mean square error as shown in the following equation.

$$E(X,\theta) = \frac{1}{2N}\sum_{i=1}^{N} \|t_i - f(x_i,\theta)\|^2 \tag{9}$$

In the above equation, the error function $E(X,\theta)$ is set to one value given the data set X and the parameter θ. The data set X is the value given from the outside, and the target to be optimized is θ. Therefore it can be written $E(\theta)$. The back-propagation learning algorithm uses the gradient descent method to find the parameters to minimize the error function $E(\theta)$. The gradient descent method is an algorithm that finds a parameter that minimizes the value of a cost function iteratively.

$$\theta(t+1) = \theta(t) + \Delta\theta(t) = \theta(t) - \eta\frac{\partial E(\theta)}{\partial\theta} \tag{10}$$

Where η is the learning rate that controls the speed of learning. In the multi-layer perceptron, the back-propagation learning uses the error function $E(x, w_{jk})$ for a data by applying a stochastic gradient descent method of updating a data for each weight.

$$w_{jk}(k+1) = w_{jk}(k) - \eta\frac{\partial E(x,w_{jk})}{\partial w_{jk}} \tag{11}$$

$$E(x,w_{jk}) = \frac{1}{2}(t_k - y_k)^2 = \frac{1}{2}\left(t_k - \sum_{j=1}^{m} w_{jk}^2 a_j^1\right)^2 \tag{12}$$

In the above equation, the weight w_{jk}^2 between the hidden layer and the output layer, and the weight w_{ij}^1 between the input layer and the hidden layer are parameters which should be corrected through learning.

As another method, the momentum method obtains and updates the current error and the error that reflects the previously used error to some extent.

$$v_{jk}(k) = \gamma v_{jk}(k-1) - \eta\frac{\partial E(x,w_{jk})}{\partial w_{jk}} \tag{13}$$

$$w_{jk}(k+1) = w_{jk}(k) + v_{jk}(k) \tag{14}$$

Here, γ is 0< γ <1 as the reflection coefficient. And the Nesterov momentum method predicts the direction of movement before calculating $v(k)$, and calculates the gradient after moving in that direction in advance.

$$v_{jk}(k) = \gamma v_{jk}(k-1) - \eta\frac{\partial E(x,w_{jk})}{\partial(w_{jk} + \gamma v(k-1))} \tag{15}$$

In neural network training, if the learning rate value is too small, it takes a long time to learn, and if it is too large, the learning is not performed properly. A simple way to solve this problem is the learning rate decay method, which lowers the learning rate value of all parameters. The Adagrad method is a method of adjusting the learning rate according to the number of updates of the weights to give larger changes to parameters with fewer changes.

$$g_{jk}(k) = g_{jk}(k-1) + \left[\frac{\partial E(x, w_{jk})}{\partial w_{jk}}\right]^2 \tag{16}$$

$$w_{jk}(k+1) = w_{jk}(k) + \eta \frac{1}{\sqrt{g(k) + \epsilon}} \frac{\partial E(x, w_{jk})}{\partial w_{jk}} \tag{17}$$

Where ϵ is a very small value and prevents division by zero. $g(k)$ squares the existing gradient value and adds it continuously. Among the elements of the parameter, the significantly updated element has a lower learning rate, and is applied differently for each element of the parameter. However, as the learning is repeated, the value of the gradient squared gradually decreases, the update intensity becomes weak, and eventually becomes 0 at some point, and there is a disadvantage that learning may not proceed any more.

And the RMSprop changed the $g(k)$ obtained by adding the square value of the slope in the Adagrad to an exponential moving average instead of a sum. Like Adagrad, $g(k)$ does not grow indefinitely, and the relative magnitude difference of recent changes between variables is maintained.

$$g_{jk}(k) = \gamma g_{jk}(k-1) + (1-\gamma)\left[\frac{\partial E(x, w_{jk})}{\partial w_{jk}}\right]^2 \tag{18}$$

Where γ is called a decaying factor and has a value of 0.9 to 0.999. In the Adagrad, since $g(k)$ is defined as the sum of changes up to the current time, it increases as time passes and the learning rate decreases. However, in the RMSprop, it is defined as the exponential average of the previous change and the current change, so that a sudden decrease in the learning rate is prevented.

Finally, the Adam is an optimizer created by combining the RMSprop, which changes the learning rate, and the Momentum, which changes the update path by optimization. Like Momentum, it stores the exponential average of the gradients calculated so far, and stores the exponential average of the square values of the gradients like RMSprop.

$$v_{jk}(k) = \gamma_1 v_{jk}(k-1) + (1-\gamma_1)\frac{\partial E(x, w_{jk})}{\partial w_{jk}} \tag{19}$$

$$g_{jk}(k) = \gamma_2 g_{jk}(k-1) + (1-\gamma_2)\left[\frac{\partial E(x, w_{jk})}{\partial w_{jk}}\right]^2 \tag{20}$$

$$w_{jk}(k+1) = w_{jk}(k) + \eta \frac{v_{jk}(k)/(1-\gamma_1)}{\sqrt{g_{jk}(k)/(1-\gamma_2) + \epsilon}} \tag{21}$$

Where γ_1 = 0.9, γ_2 = 0.999, and $\epsilon = 10^{-8}$ are recommended.

6. RESULTS OF SIMULATIONS

In this study, a simulation program was written using the Keras library to verify the performance of the noise attenuator for the optimization algorithms. The input signal mixed with voice and noise was sampled at 8 kHz and consisted of 300,000 samples (37.5 sec). Since this system corresponds to supervised learning, the input data is internally composed of an input array of 64×499,901 samples and a target value of 499,901 samples. To evaluate the performance of the system, the mean square error MSE for the error between the target value, the input signal and the speech prediction value, was used. The MSE curves were compared.
Figure 3 shows the MSE curves for the Adam, RMSprop, and Adagrad algorithms when the SNR is 20dB. Here, the Adam curve is black, the RMSprop curve is blue, and the Adagrad curve is red. From this figure, as the update progresses, it can be seen that the MSE decreases rapidly at first, and then gradually decreases from the number of batches of 3,000. The Adam algorithm represents the smallest MSE and the Adagrad algorithm represents the largest MSE.

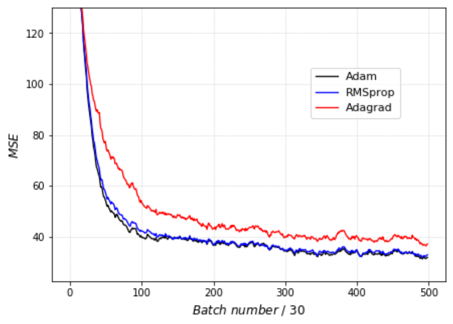

Figure 3. Curves of MSE for SNR = 20dB

Next, Figure 4 shows the MSE curves for the three algorithms when the SNR is 10dB, and shows similar performance to the curves when the SNR is 20dB. That is, the Adam algorithm shows the best performance and the Adagrad algorithm shows the lowest performance.

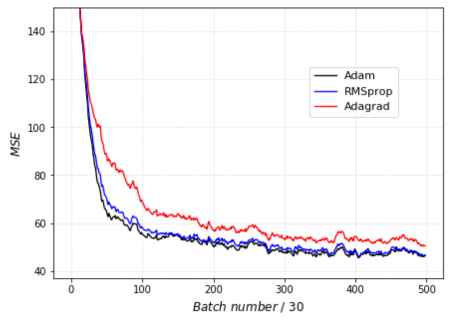

Figure 4. Curves of MSE for SNR = 10dB

And Figure 5 shows the MSE curve when the SNR is 5dB because more noise is mixed. It can be seen that the difference in performance for each algorithm is reduced compared to the previous figure. However, the performance of Adam algorithm is still the best and the performance of Adagrad algorithm is the lowest.

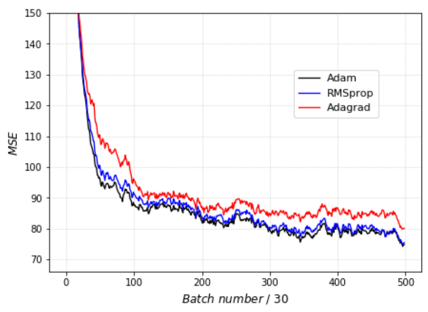

Figure 5. Curves of MSE for SNR = 5dB

Finally, Figure 6 shows the MSE curve when the noise is very mixed and the SNR is 1dB. In this case, it was found that MSE significantly increased no matter which algorithm was used. It can be seen that if the noise is large, there is almost no difference according to the algorithm, and the performance is poor, so the noise is not removed well.

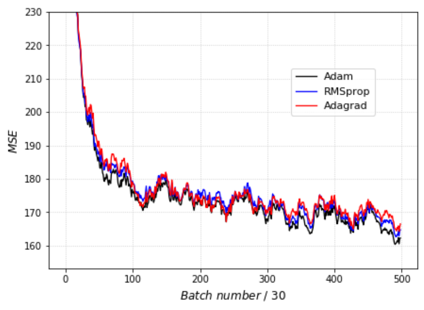

Figure 6. Curves of MSE for SNR = 1dB

In addition, Table 1 summarizes the MSE values for each optimization algorithm in a table. From this table, it can be seen that as the SNR decreases, the MSE increases, and when the SNR becomes 1 dB, almost no noise is removed. In addition, the performance of the Adam algorithm is the best, the performance of the RMSprop algorithm is slightly inferior to that of the Adam algorithm, and the performance of the Adagrad algorithm is the lowest

Table 1. MSE values of optimization algorithms for SNRs

SNR[dB]	20	10	5	1
Adagrad	49.31	62.94	91.83	176.41
RMSprop	41.78	56.05	87.28	176.04
Adam	40.83	54.45	85.63	173.72

7. CONCLUSIONS

In this paper, the effect of the optimization algorithm on the performance of the noise attenuator using CNN deep learning technology was investigated. The noise attenuator was implemented using a 64-neuron, 16-filter CNN filter and an error backpropagation algorithm. The model was coded using the Keras library, and how the MSE value changes according to the optimization algorithm was observed.
As a result of simulation, this system showed the smallest MSE value when using the Adam algorithm among the Adam, RMSprop, and Adagrad optimization algorithms, and the largest MSE value in the Adagrad algorithm. This is because the Adam algorithm requires a lot of computation, but it has an excellent ability to estimate the optimal value by using the advantages of RMSprop and Momentum SGD.

ACKNOWLEDGEMENTS

Funding of this paper was provided by Namseoul University

REFERENCES

[1] S. F. Boll, "Suppression of acoustic noise in speech using spectral subtraction," IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-29, pp.113-120, Apr. 1979.
[2] A. Schaub and P. Schaub, "Spectral sharpening for speech enhancement/noise reduction," IProc. of Int. Conf. on Acoust., Speech, Signal Processing, vol.2, pp.993-996, May 1991.
[3] J. S. Lim and A. V. Oppenheim, "All-pole modeling of degraded speech," IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-26, pp.197-210, Jun. 1978.
[4] J. Hansen and M. Clements, "Constrained iterative speech enhancement with to speech recognition," IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-39, no.4, pp.21-27, Apr. 1989.
[5] J. S. Lim, A. V. Oppenheim and L. D. Braida, "Evaluation of an adaptive comb filtering method for enhancing speech degraded by white noise addition," IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-26, no.4, pp.354-358, Apr. 1991.
[6] S. F. Boll and D. C. Pulsipher, "Suppression of acoustic noise in speech using two microphone adaptive noise cancellation," IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-28, no.6, pp.752-753, Dec. 1989.
[7] W. A. Harrison, J. S. Lim and E. Singer, "A new application of adaptive noise cancellation," IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-34, pp.21-27, Feb. 1986.
[8] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol.61, pp.85–117, 2015.
[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, vol.86, no.11, pp.2278-2324, November 1998.
[10] T. M. F. Taha, A. Adeel, and A. Hussain, " A Survey on Techniques for Enhancing Speech," Cognitive modeling, vol.179, no.17, pp.1-14, 2018.
[11] S.-W. Fu, Y. Tsao, and X. Lu, "Snr-aware, convolutional neural network modeling for speech enhancement," INTERSPEECH, pp. 3768–3772, 2016.
[12] S.-W. Fu, T.-Y. Hu, Y. Tsao, and X. Lu, "Complex spectrogram enhancement by convolutional neural network with multi-metrics learning," 2017.
[13] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, Dive into Deep Learning, pp. 435-514, Jun. 2021.
[14] N. A. Rahmad, N. A. Jasmin Sufri, M. A. As'Ari, A. Azaman, "Recognition of Badminton Action Using Convolutional Neural Network," Indonesian Journal of Electrical Engineering and Informatics, vol. 7, no. 4, pp. 750-756, Dec. 2019.
[15] A. M. Prasanna Kumar, S. M. Vijaya, "Noise Cancellation Employing Adaptive Digital Filters for Mobile Applications ," Indonesian Journal of Electrical Engineering and Informatics, vol. 8, no. 1, pp. 112-122, Mar. 2020.
[16] H. Lee, " Noise reduction system by using CNN deep learning model," Cognitive modeling, vol.9, no. 1, pp.84-90, 2021.
[17] P. B. Patil, "Multilayered network for LPC based speech recognition", IEEE Transactions on Consumer Electronics, vol. 44, no. 2, pp. 435-438, 1998.
[18] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," Cognitive modeling, vol.5, pp.3, 1988.