iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** IX **Month of publication:** September 2022

DOI: https://doi.org/10.22214/ijraset.2022.46884

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue IX Sep 2022- Available at www.ijraset.com

Real Time Speech Recognition in 1-Dimensional Using Convolution Neural Network

Sajad Ahmad Shah[1], Sahilpreet Singh[2]
[1]*M. Tech Scholar, Department of Computer Science and Engineering, RIMT University, Mandi Gobingarh, Punjab, India*
[2]*Asst.Professor, Department of Computer Science and Engineering, RIMT University, Mandi Gobingarh, Punjab, India*

Abstract: ***Automatic voice recognition is a hot topic in artificial intelligence and machine learning, with the goal of creating robots that can communicate with humans through speech. Speech is an information-dense communication that includes both linguistic and paralinguistic information. Emotion is a prime example of paralinguistic information that is partially communicated through speech. Developing machines that can grasp non-linguistic information like emotion simplifies human-machine communication by making it more natural and straightforward. The effectiveness of convolutional neural networks in recognizing speech has been examined in this study.. The input characteristics of the networks were wide-band spectrograms of voice samples. The networks were trained using voice signals produced by actors while acting out a given mood. Our models were trained and evaluated using English-language speech datasets. Two degrees of augmentations were applied to the training data in each database. To regularize the networks, the dropout approach was used. Our findings revealed that the gender-agnostic, language-agnostic CNN models attained state-of-the-art accuracy, beat previously published results in the literature, and matched or even surpassed human performance on benchmark databases. Future research should look at the deep learning models' capacity to recognize speech emotion using real-world speech signals.***
Keywords: Automatic Voice Recognition, CNN, Linguistic, Deep learning

I. INTRODUCTION

Speech is an information-dense communication that includes both linguistic and paralinguistic information. Identity, gender, purpose, mood, and emotion are some of the most important paralinguistic information transmitted by speech, but they have received less attention in the traditional ASR paradigm [13].. To grasp the underlying meaning of utterances and have efficient communication, the human brain uses all linguistic and paralinguistic information. In fact, any impairment in the perception of paralinguistic features has a negative impact on communication quality. Children who are unable to grasp the emotional states of those around them are said to have weak social skills and, in certain situations, psychopathological indications[5]. This emphasizes how crucial it is to recognize the emotional states of speech in order to communicate effectively. As a result, building robots that can interpret non-linguistic information like emotion is critical for creating clear, effective, and human-like communication.

Hidden Markov models (HMM), Gaussian mixture models, nearest neighbour classifiers, linear discriminant classifiers, artificial neural networks, and support vector machines are just a few examples of widely used techniques for classifying emotions based on their acoustic features of interest. The performance of these classifiers is mostly determined by feature extraction techniques and the relevant characteristics for each emotion. There is no general agreement on the acoustic correlates of emotions [2] since the acoustic correlates of emotion in speech signals differ between speakers, genders, languages, and cultures [13].. Depending on the speech corpus, this leads in a variety of "hand-crafted" qualities. One obvious method to solving this challenge is to use deep learning models. Speech emotion recognition is a multimodal process by definition. Although speech communicates a significant amount of emotional information, it is insufficient for distinguishing human affective states in everyday contexts. Other modalities, such as visual or verbal, also help to provide the necessary information for emotion recognition. People employ various paralinguistic signals such as facial expression, body language, semantics, and context in addition to speech to discern emotions in others.

In fact, when people interact with one another, body language is thought to convey 55 percent of the message [6]. We focused our research on speech and did not investigate other modalities. As a consequence, the databases played by actors were employed in the current investigation since actors conveyed emotions with exaggeration, perhaps compensating for the absence of information offered by other modalities. When opposed to utilizing real-life utterances, this gives us more control over how successful deep learning models are. The widespread availability of performed speech standards in the speech community, on the other hand, allows us to compare our work to previously investigated models. We used acted voice datasets to test the capabilities of convolutional neural networks in categorizing speech emotions within and across British and American English.

The use of wide-band spectrograms rather than narrow-band spectrograms, as well as analyzing the influence of data augmentation on the accuracy of our models across widely-used benchmark databases, are unique contributions of this work. Wide-band spectrograms and data augmentation enabled CNNs to attain state-of-the-art accuracy and outperform humans, according to our findings. To create useful features for a certain activity, domain-specific knowledge and engineering (Chiang et al., 2009; Forman, 2003) are widely utilized. In reality, feature engineering may offer benefits; nevertheless, the time spent on feature design restricts the scope of AI applications. At the end of the 1950s, the idea of training multi-layer networks to substitute hand-designed features was considered (Rosenblatt, 1958; Selfridge, 1958). In the 1980s, the advent of error back-propagation algorithms (Rumelhart et al., 1988) made it possible to train multi-layer models using a basic stochastic gradient descent technique.

A. Automatic Speech Recognition

One of the early goals in AI research was automatic speech recognition (ASR), sometimes known as voice to text. Speech, often known as spoken language, is the most natural way for humans to communicate. For human-computer interaction, ASR systems can provide a more convenient and user-friendly platform. Furthermore, automated speech processing and interpretation by machines contributes to artificial intelligence's ultimate aim.. Bell Laboratories created the first ASR system, a digit recogniser, in 1952. (Davis et al., 1952). The acoustic phonetic method (Hemdal and Hughes, 1967) and pattern matching were two early techniques to voice recognition that focused on rule-based and knowledge-based heuristic approaches (Itakura, 1975). Hidden Markov models (HMMs), specifically Gaussian mixture model HMMs (GMM-HMMs), were first used in voice detection in the 1970s (Baker, 1975; Jelinek, 1976). Statistical methods have dominated this field of study since then. Given a set of characteristics x1:T, of length T, collected from raw voice stream mathematically,

II. LITERATURE REVIEW

Ming Dong et al. [7] used convolutional neural networks, which are a specific kind of deep learning models, to automatically learn discriminative features from the narrowband spectrograms of speech signals. Subsequently, they used an SVM to classify the features learned by CNN. Their results demonstrated a superior performance in learning high-level discriminative features from the low-level spectrographic representation of the speech signals

Y. Wu et al. [7] noted that discriminative testing has been used for speech recognition for many years now. The few organizations that have had the resources to implement discriminatory instructions for large-scale speech recognition assignments have mostly used the full shared information system in recent years (MMI). Instead, in the extension of the studies first presented, we reflect on the minimum classification error (MCE) paradigm for discriminatory instruction.

Papakostas et al. [8] evaluated the capabilities of deep convolutional neural networks in identifying voice emotions. They compared the performance of deep neural networks with SVM classifiers in this way. Deep neural networks outperformed SVM classifiers in distinguishing emotional states, according to their findings.

Shahnawazuddin and Sinha [10] discussed how the work presented was an extension of the current quick adaptation approaches based on acoustic model interpolation. The basis (model) weights are calculated in these methods using an iterative process that uses the maximum likelihood (ML) criterion.

D.J. Atha et al. [3] pointed out that a long-term target is the creation of an automated real-time translation device where the voice is the source. Recent developments in the area of computational translation science, however, boost the possibility of widespread adoption in the near future.

Peng et al. [4] stated that identification of speakers refers to identifying people by their voice. This technology is increasingly adopted and used as a kind of biometrics for its ease of use and non-interactivity, and soon became a research hotspot in the field of biometrics,

Vinícius Maran et al. [2] explained that learning speech is a dynamic mechanism in which the processing of phonemes is marked by continuities and discontinuities in the path of the infant towards the advanced production of ambient language segments and structures.

III. METHODOLOGY

Machine learning is a branch of ai technologies and computer science. It focuses on creating algorithms that allow a task or a skill naturally and gain knowledge via experience, such as viewing training data. Following that, machine learning should be able to explain their learnt knowledge from these observations to fresh, unknown data. Various learning methodologies, such as supervised learning, unsupervised learning, and reinforcement learning, can be used to accomplish this. The next sections provide a quick overview of supervised learning and how it was used in the current work.

We recommend Mitchell et al. [4], Bishop [5], Suton and Barto [36], and James et al. [7] for further information on various forms of learning. In supervised learning, the training data include the expected response, known as labels; that is, there is a label for each observation (training sample or instance). The purpose of learning is to properly anticipate the label of each training/test instance.
The machine learning algorithm learns to categorize the input data into two or more categories, which is an example of supervised learning. Based on observed training data, the algorithm learns the discriminant characteristics or properties across distinct categories or classes. These characteristics are then utilized to categorize additional test input data.
Although some ANNs can be trained using machine learning technique, artificial neural networks (ANNs) are a well-known example of classifier [3]. The way biological neural networks, such as the human central nervous system, function has inspired ANNs. That is, they are made up of neurons, which are closely linked processing units. ANNs are the fundamental components of deep learning, a powerful current machine learning approach. Deep learning models are, in reality, ANNs with a large number of neurons and layers. Deep learning models have excelled in a variety of machine learning applications, including classification.. Deep learning models' capacity to learn complex characteristics from basic data is a crucial aspect that makes them effective [24]. That is, the earliest layers of deep learning models represent the training data's simplest and most fundamental properties. These low-level properties are used by the deeper layers to create a complicated representation. This ability to construct high-level features from low-level data has the potential to minimize the amount of preprocessing necessary for extracting hand-crafted features prior to classifier creation. We used a deep learning model termed convolutional neural network to categorize emotional states of voice data in the current study.

The rest of this chapter is laid out as follows:

A. Multi-layer Perceptron Networks

One of the characteristics of artificial neural networks is their network design (ANNs). It controls how neurons, which are the fundamental processing units, are linked. The multi-layer perceptron (MLP) is a well-known ANN design made up of neurons known as linear threshold units (LTUs) [6]. A linear threshold unit is shown in Figure .1. An LTU accepts weighted inputs from several neurons (in this case, three neurons) and computes the linear combination of the inputs as z = w1x1 + w2x2 + w2x2 + b, where b is the bias factor.. The linear combination is then applied to a step function (e.g., Heaviside function H(x) or Sign function sgn(x)) to give the output y = f(z), where f is a step function. The LTU will provide an output if the weighted total is larger than a threshold value (which is determined by the bias term).

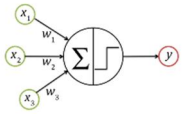


Figure . 1Linear threshold unit

An MLP typically consists of one input layer, one or more LTU layers (also known as hidden layers), and one output layer. The data goes from the input layer (lower level) to the output layer (higher level) (higher level). They are known as feedforward artificial neural networks because of this. The values of one training sample are represented in several dimensions by the input layer. This might be the amplitude of an audio stream at various sample points or the picture intensity at various pixels. In most cases, the input is represented as a vector x, with the length indicating the number of dimensions (e.g., the number of sampling points in an audio signal or the number of pixels in an image).
The output can be either a real integer, y, or a vector, y, which displays the input label. The layers of neurons in an MLP are designated as l [0], l[1], l[2],.., l[n], l[n+1], where l [0] is the input layer, l [1], l[2],.., l[n], l[n+1], where l [0] is the input layer, l [1], l[2],.., l[n], l[n Every neuron in the layer l [j] 1j(n+1) (all layers save the input layer) gets weighted input directly from every neuron in the layer one level down, i.e., l [j1]. W1 and W2 are matrices related with the values that weight the inputs of the neurons in Figure 3.2, which shows an MLP with one hidden layer.

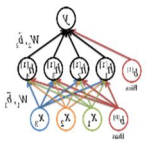


Figure 2 Multi layer perceptron

The parameters of interest are the weight matrices (W1 and W2) as well as the bias vectors (b1 and b2). That is, by modifying the values of these parameters, the network learns to categorize the incoming data. There are a number of learning approaches that may be used to determine the best values for these parameters. One of these learning approaches that has been frequently utilized to train MLP networks is encourage low. The backpropagation method is based on gradient descent, which is briefly discussed in the next section.

B. Gradient Descent

The search for the best weight parameter values may be thought of as an optimization issue. The purpose is to determine the parameters of interest that minimize the error function after obtaining the error function or loss function. The error function can be defined in a variety of ways. W, d, and D represent weights, one training instance, and all training data, respectively, in Equation 3.1, which shows a well-known error function termed sum of squared errors [34]. The squared of error between the actual (td) and projected (yd) labels is totaled across all training data, as illustrated. The purpose is to look for values in the weight space that minimize this function..

The MLP networks' error function is not convex like a linear unit's. As a result, establishing global minima isn't a foregone conclusion. Because we don't have access to the hidden neurons' output, the backpropagation method uses the chain rule of calculus to determine the contribution of hidden neurons to the output error and update the hidden layers' weights. 24 and 38 . It should be emphasized that the MLP networks' step function makes determining the derivative with regard to the weights difficult. As a result, the step function is substituted by the differentiable sigmoid function (x) = 1 1+ex. We suggest the reader to [8] for further information. There are various gradient descent versions that accelerate the pace of convergence, such as gradient descent with momentum [9] or root mean square propagation (RMSprop) [10]. Instead of using gradients to update the weights, gradient descent with momentum employs an exponentially declining weighted average of gradients.. That is, the gradient across the number of iterations can be viewed as a timeseries signal, $\partial E\ \partial w\ (t)$, where t stands for the number of iterations. The gradients can be replaced with the exponentially weighted average of gradients as $vt = \beta vt-1+ (1-\beta)\ \partial E\ \partial w\ (t)$, where $v0 = 0$ and $\beta \in [0, 1]$, to update the weights. RMSprop optimization computes the exponentially decaying weighted average of the squared gradient, $m(t) = \beta mt-1 + (1 - \beta)(\ \partial E\ \partial w\ (t))2$, where $m0 = 0$ and $\beta \in [0, 1]$, and use its square root to scale the learning rate as $\eta 0 = \sqrt{}\ \eta\ m(t)$. These gradient descent versions can be used to smooth out oscillations around local optima and speed up convergence. The Adam method, developed by Kingma and Ba [11], is another gradient-based optimization technique that minimizes the loss function by combining the benefits of gradient descent with momentum and RMSprop optimization [2]. To update the weights, it uses first-order momentum as in gradient descent with momentum and second-order momentum as in RMSprop optimization. This method gets its name from "adaptive moment estimation." This approach is extremely efficient and is commonly used in deep learning models..

C. Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep learning model that has had a lot of success in fields including object identification [3], face recognition [4], handwriting recognition [5], speech recognition [6], and natural language processing [7]. The word convolution stems from the fact that neural networks use convolution, a mathematical technique. The convolutional layer, the pooling layer, and the fully connected layer are the three basic building elements of CNNs. Following are descriptions of various building elements, as well as some fundamental concepts like softmax, rectified linear unit, and dropout.

D. Convolutional Layer

Convolutional layers in CNNs compute output using mixture rather than duplication. As a result, the neurons in the convolutional layers aren't all linked to the neurons in the layers above them. The fact that neurons in the visual cortex have a local receptive field inspired this layout [9]. That is, the neurons are trained to respond to stimuli that are specific to a certain region and structure. As a result, convolutional neural networks have sparse connection and parameter sharing, which substantially reduces the number of parameters in deep neural networks. The convolution of a kernel, which is a 2 2 matrix, with a one-channel 3 3 picture is shown in Figure 3.3. The size of the output is 2 2 1. In general, the output size is (nh f + 1) (nw f + 1) nf, where nh is the input height, nw is the input width, and nf is the number of kernels. The depth of the input determines how deep the kernel is. The depth of the input in the case shown in Figure 3.3 is nc = 1. As a result, the kernel depth is 1. Also, because there is just one kernel, the output depth is 1. Each output neuron is the weighted sum of the input neurons inside the appropriate receptive field, as can be observed, resulting in sparse connectivity in CNNs. . Furthermore, the kernel is shared across the layer, allowing CNNs to exchange parameters. Stride is the amount by which the kernel glides along the input.

The stride in our case (Figure 3) is s = 1, which indicates the kernel moves one step across the picture. It's worth noting that after each convolutional layer, the input volume reduces. We can pad the input's outside border with zero to avoid this reduction.
Overall, the output's height and breadth are governed by

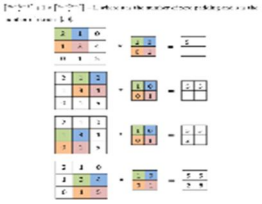


Figure 2 Convolution 3x3image

Local filtering in convolutional layers allows for the detection of numerous low-level characteristics of interest and the generation of various feature maps. These feature maps are used by the deep layers to build a high-level representation of the inputs.

E. Pooling Layer

A pooling layer is the second crucial component of CNNs. This layer is designed to reduce the sensitivity of the outputs to local variations in the inputs. In some applications, this variability to tiny local translational might reduce spatial resolution and contribute to underfitting. When precise spatial characteristics are not required, pooling can help CNNs extract the features of interest more quickly. Pooling can also help to avoid overfitting by reducing the number of dimensions and parameters [2]. Pooling, in a sense, extracts subsamples from the outputs [3].. Pooling layers, like convolutional layers, employ a kernel (a linear receptive field) to summary the values of the neurons inside the pooling kernel using an aggregation function like maximum, average, L2-norm, or weighted average. In order to create a pooling layer in CNNs, we must first establish the size of the pooling kernels, the shifting step, and the amount of padding. Figure .4 shows maximum pooling over a 3 3 matrix with a pooling kernel of size 2 2 that moves one pixel across the matrix (i.e., stride of 1).

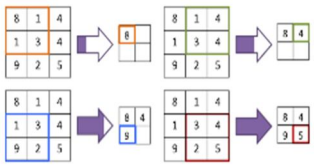


Figure. 3 Pooling of a 3x3 using 2x2 kernel

F. Fully Connected Layer

A typical CNN is made up of many convolutional layers, each of which is followed by a pooling layer. The fully linked layer, which is essentially a regular MLP, is the last building piece of CNNs. This component is used to either build a more abstract representation of the inputs by processing the features further or to categorize the inputs based on the features retrieved by the precedinglayers [51]. Softmax Unit 3.2.4 The output of the completely linked layer is generally a softmax unit.. To describe the posterior distribution of k classes, a softmax unit uses the softmax function (normalized exponential). In reality, the softmax function is an extension of the sigmoid function, which depicts the probability distribution of two potential classes [24]. The softmax function is shown in Equation 3.5.

$\sigma(z) = 1\ 1+e{-}z = e\ z\ 1+e\ z$

$$\text{softmax}(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}, \qquad (3.5)$$

where $\vec{z}$ is the output of the k-way softmax unit, $\text{softmax}(\vec{z})_i$ is the probability that the input instance is in class i, and z_i is the ith element of the vector $\vec{z}$.

1) Rectified Linear Unit

The activation functions have been the linchpin of artificial neural networks (ANNs). That is, they incorporate nonlinearity into ANNs, which makes ANNs an effective tool to learn complex models. Sigmoid function $g(z) = \sigma(z) = 1\ 1+e{-}z$ and hyperbolic tangent function $g(z) = \tanh(z) = 2\sigma(2z) - 1$ are two popular activation functions, especially in traditional ANNs (see Figure .5).

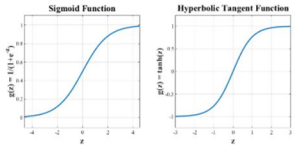


Figure 4 Sigmoid Activaton Function

One drawback of these functions is that they saturate for zs with high absolute values, causing gradient-based learning to suffer. The rectified linear unit (ReLU)—g(z) = max(z, 0)—can be used as an activation function to address this issue [4].

The ReLU function is shown in Figure 3.6. It acts linearly for z > 0 and z 0 despite its nonlinearity around z = 0. ReLU functions provide a balance between nonlinearity, which is required to train complicated models, and linearity, which is required for gradient-based learning. As a result, ReLU functions are frequently utilized in deep learning models as the perceptron. Before subsampling the pooling layers, ReLU functions are applied to the feature maps in CNN convolutional layers. The zero output for zs with negative value is the major flaw of the ReLU functions. The dying ReLUs [3] is an issue in which the output of the neurons reaches zero during training and remains zero. As a result, the neurons lose their ability to function. To address this issue, a leaky ReLU function has been developed, ReLU(z) = max(z, z), where specifies the slope of the ReLU function for z 0 [8]. According to previous research, utilizing leaky ReLU increases the risk of overfitting the training data for a small number of training occurrences [38]. This is due to the fact that as the number of parameters grows, the network's sensitivity to nuisance variations grows as well..

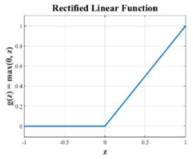


Figure 5 Rectified linear function

2) Mini-batch Learning

Deep learning models include a huge number of training instances, which slows learning and puts a strain on computational resources. The concept of employing tiny batches of examples from the training set aims to address the issue of data size [24]. That is, instead of one batch of training data for each iteration, the network is trained over numerous batches. Instead of a single error function, the optimization problem occurs on numerous error subfunctions. Mini-batch learning is a middle ground between batch and stochastic learning. All training instances are utilized to train the model for each iteration in batch learning, as we said before. In stochastic learning, on the other hand, each instance is utilized independently to create an error function.. The main advantage of stochastic and mini-batch learning is that it saves time and resources. The stochastic or mini-batch algorithms, on the other hand, never converge to the global optimum; instead, they approach or fluctuate about it [8]. Because it considers a greater fraction of the training data while optimizing the error functions, mini-batch learning is more accurate than stochastic learning. However, stochastic learning is more likely than mini-batch learning to avoid local optima [3]. In deep learning models like convolutional neural networks (CNNs), mini batches are commonly set to 64, 128, 256, 512, or 1024.

3) Dropout

Gradient-based learning has two major properties: variance and bias [7]. Depending on the project set use it for learning, the estimated underlying model of a system can be altered. Variance is a term used to describe these changes. The ideal situation is to have a model that is a broad representation of system behavior and hence less susceptible to the unique training set. High variance models overfit the training data by following noise-induced changes in the training set. In general, sophisticated models have a lot of volatilityOn the other hand, models constructed on assumptions that simplify the system's true activity have a strong bias. These models underfit the training data and fail to represent the system's underlying diversity. Due to the large quantity of training data and model parameters in deep neural networks, there is a substantial danger of overfitting the training set (high variance) in deep learning models. There is a trade-off between variance and bias in classical machine learning; that is, lowering unpredictability leads to greater bias and vice versa. However, using deep architectures and large training sets, this long-standing variance-bias problem has been overcome. The variance may be minimized without affecting the bias as long as the deep networks are adequately regularized [2].. Models with a lot of variation do well on the training set, but they don't generalize well on the test set. Traditional ways to remedying overfitting include L1-norm regularization and L2-norm regularization. As illustrated below for L1-norm regularization and L2-norm regularization [4], the objective function adds weight decay into the loss function.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathscr{L}(l^{(i)}, s^{(i)}) + \frac{\lambda}{2m}\|w\|_1, \qquad (3.9)$$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathscr{L}(l^{(i)}, s^{(i)}) + \frac{\lambda}{2m}\|w\|_2^2, \qquad (3.10)$$

where m denotes the number of research data, L designates the calculation, such as cross entropy or mse error (MSE), l I conveys the true label of the ith learning algorithm, s I denotes the predicted label of the ith training instance, w denotes the parameters of interest, and is a hyperparmeter that controls the regularization strength. When the regularization term is included, the optimization is limited to small values of w.. This may result in weight assignment to a reduced number of characteristics, which reduces the model's complexity. Dropout is a strong regularization approach that is commonly employed in deep learning models to reduce overfitting. Dropout is a powerful and effective approach for controlling deep neural networks by removing neurons in the hidden layers at random during training. The neurons are removed with a propensity, such as p [55]. The networks' outputs and error function may be represented as random variables. In other ways, dropout is similar to bagged ensemble training [14], which uses training data to train several networks.

As a result, the network with dropout's output may be seen as an average ensemble of several networks with various designs [13]. According to Baldi and Sadowski [6], the predicted value of the network's error function with dropout contains a regularization factor comparable to weight decay in L1-norm and L2-norm regularization. They also claim that when the hidden neurons are excluded with a probability of p = 0.5, the best amount of regularization is attained. Data Enhancement Deep learning algorithms strive for data because their true potential is amplified when massive data sets are used to train them [5].

IV. SYSTEM ARCHITECTURE

We used convolutional neural networks (CNNs) to identify speech interjections based on their content in the current study. We trained and evaluated our models using a proprietary database in addition to three commonly used benchmarks for voice utterance recognition [2]. For our CNN models, we used TensorFlow (an open-source library developed in Python and C++ [5]) as the programming framework. The present work's experimental setup is described in this chapter.

A. Pre-processing

Prior to any processing, all utterances were resampled and filtered using an antialiasing FIR lowpass filter to have a frequency rate of 16 kHz to ensure consistency across all databases. The spectrograms of all audio utterances were then created. A spectrogram is a visual representation of energy fluctuation at different frequencies over time. The ordinate (vertical axis) indicates frequency, whereas the left half (horizontal axis) represents time. The amount of darkness or the colors are used to encode the energy or intensity..

There are two general types of spectrograms:

- Wide-band spectrograms
- Narrow-band spectrograms

The temporal resolution of wide-band spectrograms is higher than that of narrow-band feature extraction. Individual glottal pulses can be seen in wide-band spectrograms because of this characteristic. Narrow-band spectrograms, on the other hand, have a better frequency resolution than wide-band spectrograms. Individual harmonics can be resolved using narrow-band spectrograms because of this feature [3].

A voice utterance's wideband and narrowband spectrogram pictures are shown. We opted to transform all utterances into wide-band spectrograms because of the relevance of vocal fold vibration and the fact that glottal pulse is related with one period of vocal fold vibration [7]. The length of Hamming windows was set at 5 ms with a 4.4 ms overlap as a result. A total of 512 DFT points were used.

We also removed frequencies over 8 kHz from spectrograms since frequencies below 8000 Hz are sufficient for speech comprehension in many cases [6]. In early investigations, removing energy over 8000 Hz increased the algorithms' performance. This resulted in 129 frequencies. All spectrogram pictures were scaled to 129 129 pixels and then z-normalized to have a zero mean and close to one standard deviation.

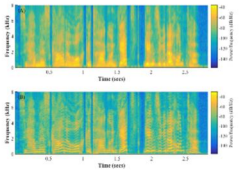


Figure 6 Wide band spectrogram

1) Training and Test Sets

5-fold cross-validation was used to train and test our models. That is, the data was split into five folds. The first fold served as a training set for our models, while the others served as a test set. The second fold was then utilized to test our models, with the remaining folds being used for training, etc. The data sets were enhanced by introducing white Gaussian noise with a +15 signal to noise ratio (SNR) to each audio signal either 10 times or 20 times to prevent overfitting and the negative effect of limited database sizes.. The SNR is equal to 10 log10(Pspeech Pnoise), where P is the signal's average power. Data sets with 10 times augmentation (10x) and data sets with 20 times augmentation (20x) were utilized to train our models as a consequence of the data augmentation (20x). To test our models, I utilized the original data that was free of noise. The increased data was exclusively utilized for training purposes. Finally, the training and test data labels were encoded as single-hot vectors.

The total number of training epochs ranged from 100 to 4000. Because of the computation and time costs, the favorable training epoch was set to 100.

2) Architecture

In the current study, the baseline design of the deep neural network was a convolutional neural network with two convolutional layers and one fully connected layer with 1024 hidden neurons. The probability distribution of the classes was estimated using either a 5-way or a 7-way softmax unit, depending on the number of classes. A max-pooling or average-pooling layer was placed after each convolutional layer. Rectified Linear Units (ReLU) were utilized as activation functions in convolutional and fully connected layers to impart nonlinearity to the model. Convolutional layers' starting kernel size was set to 55 with a stride of 1.. For the first and second convolutional layers, the initial number of kernels was set at 8 and 16, respectively. Pooling layers' kernel size was set at 22 with a stride of 2. The loss function was cross-entropy, and the Adam optimizer was used to minimize the loss function across the training data's mini batches. The tiny batches were set to 512 characters. There were 100 training iterations. Using Graphics Processing Units, the networks constructed in this study required anywhere from 30 minutes to two days to train (GPUs). GPUs are utilized instead of CPUs to enhance processing speed since GPUs have several cores and can manage a high number of concurrent tasks

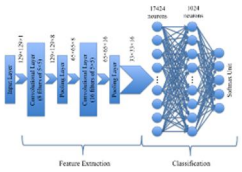


Figure 7 The baseline architecture of CNN

Our studies were done on the Holland Computing Center's Crane cluster at the University of Nebraska-Lincoln. We also added the dropout method into the fully connected layer to boost network performance if overfitting symptoms were detected.

V. SIMULATION

Import the required libraries

```
import os
import librosa
import IPython.display as ipd
import numpy as np
from scipy.io import wavfile
```

Check the sample rate of recordings

```
In [2]: trainDir = '/Users/Lenovo/OneDrive/Desktop/speech/train/'
        samples, sample_rate = librosa.load(trainDir + 'bed/00f0204f_nohash_0.wav', sr = 16000)
        ipd.Audio(samples, rate=sample_rate)
        print(sample_rate)

        16000
```

Change the sample rate to 8000 and check whether the recording is audible.

```
samples = librosa.resample(samples, sample_rate, 8000)
ipd.Audio(samples, rate=8000)
```

These are the labels of the recordings

```
[4]: labels=["bed", "bird", "dog", "four"]
```

Change the sample rate to 8000 and filter the recordings with the.wav extension. Append all recodings to the all wave list and all labels to the all label list after that

```
In [5]: trainDir = '/Users/Lenovo/OneDrive/Desktop/speech/train/'
        all_wave = []
        all_label = []
        for label in labels:
            print(label)
            waves = [f for f in os.listdir(trainDir + '/'+ label) if f.endswith('.wav')]
            for wav in waves:
                samples, sample_rate = librosa.load(trainDir + '/' + label + '/' + wav, sr = 16000)
                samples = librosa.resample(samples, sample_rate, 8000)
                if(len(samples)== 8000) :
                    all_wave.append(samples)
                    all_label.append(label)
```

Encode all labels and recordings to machine language form and split the data into train and test.

```
In [6]: from sklearn.preprocessing import LabelEncoder
        le = LabelEncoder()
        y=le.fit_transform(all_label)
        classes= list(le.classes_)

In [7]: from keras.utils import np_utils
        y=np_utils.to_categorical(y, num_classes=len(labels))
        Using TensorFlow backend.

In [8]: all_wave = np.array(all_wave).reshape(-1,8000,1)

In [70]: from sklearn.model_selection import train_test_split
         x_tr, x_val, y_tr, y_val = train_test_split(np.array(all_wave),np.array(y),stratify=y,test_size = 0.2,random_state=777,
```

Created CNN 1D

Start Training the model and after enough epochs and least validation loss, save the model.
Testing

Load the saved model and get a random recording from the test data and test it.

VI. EXPRIMENTS RESULT

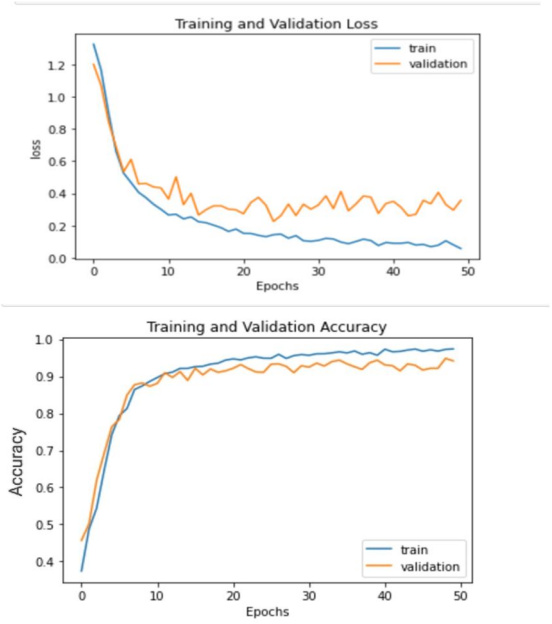


Fig 8: Training and validation results (loss & accuracy)

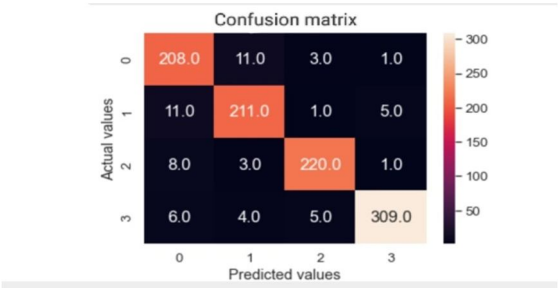


Fig 9:Confusion matrix of validation data

```
In [40]: print(classification_report(y_val, pred))
```

	precision	recall	f1-score	support
0	0.89	0.93	0.91	223
1	0.92	0.93	0.92	228
2	0.96	0.95	0.95	232
3	0.98	0.95	0.97	324
accuracy			0.94	1007
macro avg	0.94	0.94	0.94	1007
weighted avg	0.94	0.94	0.94	1007

Fig 10: Performance matrix result

VII. CONCLUSION

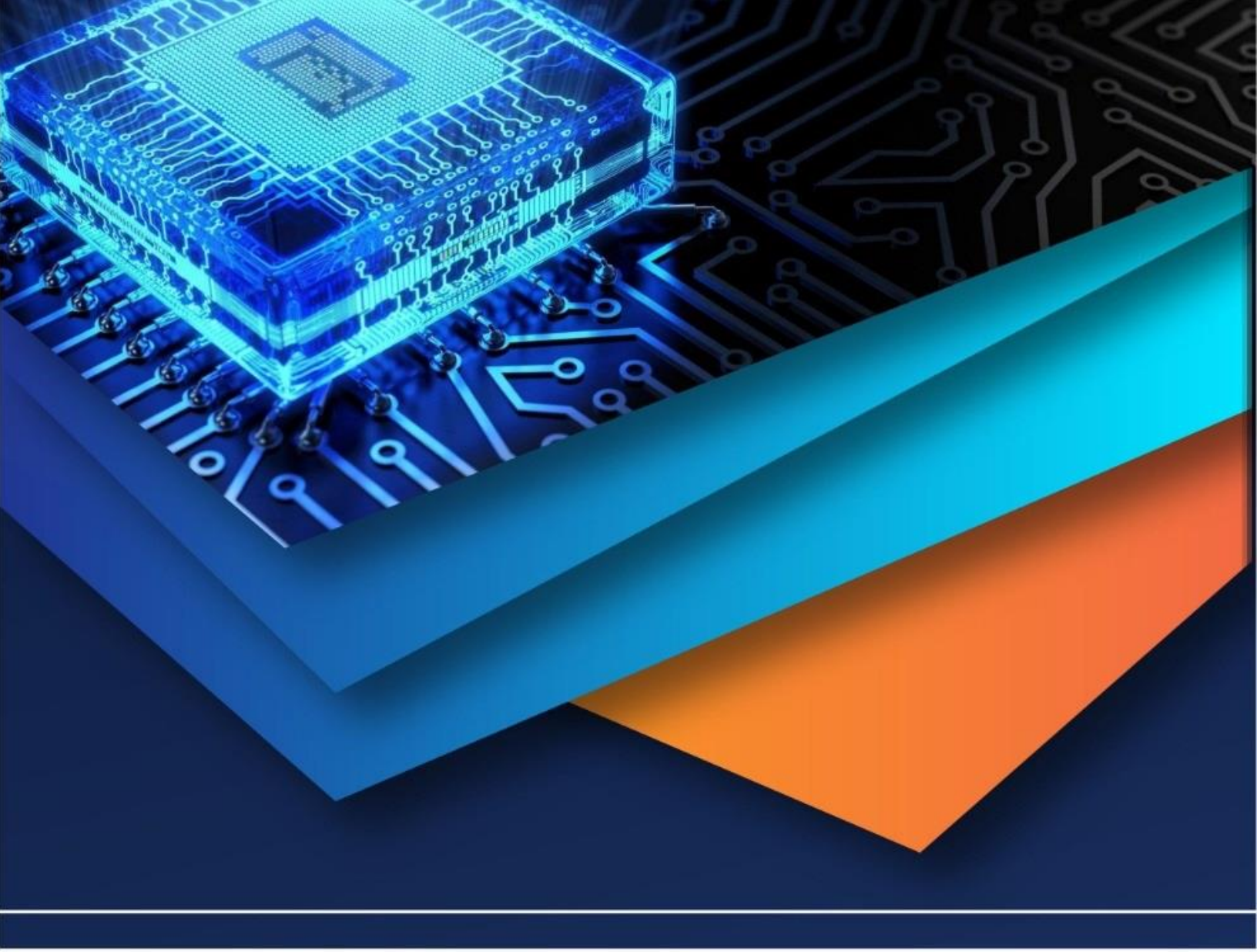
In this paper ,First convolution layer learn the useful features from the raw speech signal and remaining convolution layers further process these features into the useful information.

After processing the speech signal, CNN 1D estimates the class conditional probability by using these acoustic model send to recognizer and it convert to text which will be understand by a phenome where a discrete and distinctive unit of language that can be used to differentiate between words .

The classification accuracy of the model climbed to 94%. For learning purposes, CNN needs a lot of data the more data provided, the better and more accurate categorization accuracy CNN will provide. Based on the proportion of data division, our classification results showed that 85% of the input data was used for training and 15% for validation.

REFERENCES

[1] Vinícius Maran, Marcia Keske-Soares, towards a speech therapy support system based on phonological processes early detection, Computer Speech & Language, Volume 65, 2021, 101130, ISSN 0885- 2308, Article (CrossRefLink)
[2] D. J. Atha, M. R. Jahanshahi, Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection, Struct. Health Monit. 17 (5) (2018) 1110–1128. Article (CrossRef Link)
[3] Shuping Peng, Tao Lv, Xiyu Han, Shisong Wu, ChunhuiYan, Heyong Zhang,Remote speaker recognition based on the enhanced LDV-captured speech, Applied Acoustics,Volume 143, 2019, Pages 165-170, ISSN 0003-682X, Article (CrossRefLink)
[4] George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A Nicolaou, Bj¨orn Schuller, and Stefanos Zafeiriou. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, pages 5200–5204. IEEE, 2016.
[5] Qirong Mao, Ming Dong, Zhengwei Huang, and Yongzhao Zhan. Learning salient features for speech recognition using convolutional neural networks. IEEE Transactions on Multimedia, 16(8):2203–2213, 2014.
[6] Mehmet Berkehan Akçay, Kaya Oğuz, Speech emotion recognition: Emotional models, databases,features, preprocessing methods, supporting modalities, and classifiers, Speech Communication, Volume116, 2020, Pages 56-76, ISSN 0167-6393, Article (CrossRefLink).
[7] Y. Wu et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," arXiv preprint arXiv:1609.08144, pp. 1-23, 2016.
[8] Michalis Papakostas, Evaggelos Spyrou, Theodoros Giannakopoulos, Giorgos Siantikos, DimitriosSgouropoulos, Phivos Mylonas, and Fillia Makedon. Deep visual attributes vs. handcrafted audio features on multidomain speech emotion recognition. Computation, 5(2):26, 2017.
[9] WQ Zheng, JS Yu, and YX Zou. An experimental study of speech emotion recognition based on deep convolutional neural networks. In Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on, pages 827–831. IEEE, 2015.
[10] S. Shahnawazuddin, Rohit Sinha, Sparse coding over redundant dictionaries for fast adaptation of speech recognition system, Computer Speech & Language, Volume 43, 2017, Pages 1-17, ISSN 0885-2308, Article (CrossRefLink).
[11] Phoemporn-lakhanawannakum,chaluemwut Noyunsan,Speech recognition using deep learning ,IEEE 87933338, Article (Cross Reflink).



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY