Classification of time signals by CNN using STFT

1st Paween Pongsomboon
Computational Intelligence
Matriculation 1346048
paween.pongsomboon@stud.fra-uas.de

2nd Mahdieh Pirmoradian
Computational Intelligence
Matriculation 1323281
mahdieh.pirmoradian@stud.fra-uas.de

3rd Md Mukit Khan
Computational Intelligence
Matriculation 1347869
md.mukitkhan@stud.fra-uas.de

4th Md Rabiul Islam
Computational Intelligence
Matriculation 1345492
md.islam3@stud.fra-uas.de

***Abstract*—Time dominated signal classification is an important field that has so far covered a wide range of applications. Despite its popularity over the last few decades, it remains a difficult task that falls short of efficiency due to the nature of its randomness and dimensionality, large in data size, and continuous updating. Classifying these time signals can be beneficial in a variety of ways but Deep learning has resulted in the development of new methods, particularly Convolutional Neural Network (CNN) models. This paper proposes a method for classifying time-dominated signals from three particular objects using a two-dimensional (2D) convolution neural network (CNN) and the Short-Time Fourier Transformation (STFT).**

***Index Terms*—Convolutional Neural Network, Short Time Fourier Transform, Spectrogram, Classification of time signals**

I. INTRODUCTION

A great deal of information about our everyday surroundings and the physical actions that happens there can be transmitted by sound. Therefore, classifying this sounds can be helpful in many ways like surveillance systems or awareness and understanding of other species behaviours. But on one side, humans hearing frequency range is limited between 20 to 20,000. On the other side, handling large volumes of data is impossible or time consuming. That is when, machine learning techniques can be used. With recent great advancements in the application of automatic systems numerous researches have been done in automatic sound classification as well. For example, EnvNet [1] and speaker recognition [2]. in this research a strong deep learning machine learning approach that differs from other types of neural networks for instance having a specialty in being able to recognize patterns as well as find meaning of them names as a Convolutional Neural Network (CNN), get attentions. It behaves similarly to how our visual cortex recognizes images by Feature learning and classification. First by giving a big amount of labeled data to CNNs for training it can recognize the features which are similar in each category then at the time of testing by receiving a new data it can recognize to which category it belongs to. CNN is divided into two main sections: Feature Extraction and Classification. In CNNs the first few layers are for feature extraction and the last layers for classification. Feature extraction consists of three layers named Convolutional, batch Normalization, ReLU and Max Pooling Layer. Classification layer consists of only fully connected layer. It receives the input in the form of an image then in each layer there is a filter to recognize features then the output of each layer is the input to the next layer. When we try to train our CNN straight from row audio, we get certain issues. Firstly, long-range dependencies are difficult to record. Furthermore, ”Most mobile devices contain both cameras and microphones, and companies that develop mobile devices would like to provide functionality for classifying both videos/images and sounds; using the same technology for both these classification tasks would reduce the development cost significantly” [3]. Therefore, to reduce the cost of development it is better to use the same technology for both image and sound classification. That is the main reason that, using a Spectrogram, which is a far more compact representation of the sound and computationally simpler than raw audio is helpful. We utilize a spectrogram to visualize signals into images, which divides your signal into tiny windows and displays a spectrum of colors indicating the strength of each frequency. In reality, it employs the Short Time Fourier Transform (STFT). The fundamental aim of this paper is to show how we can use CNNs to categorize a long-sampled time domain signal. Then we reviewed what adjustments needed to be made to our Conv Net’s design in order to achieve a better result in the accuracy of model classifying, as well as what aspects we should consider while testing our program. Furthermore, in the process of creating spectrograms, selecting the window type, size, and speed of movement was critical for obtaining an accurate picture of the sound.

II. METHODOLOGY

The experiment of classification of time signals by CNN using STFT has been conducted following the flowchart diagram in (Fig. 1). For the sake of simplicity, the experiment is divided into parts. The process of experiment begins with converting audio time signal into spectrogram images using STFT. Spectrogram images are then feed into CNN model with supervised learning technique. The training process using CNN model requires the CNN model to adjust weights according to training and validated data. Lastly, the trained CNN model is then used in classification experiment.

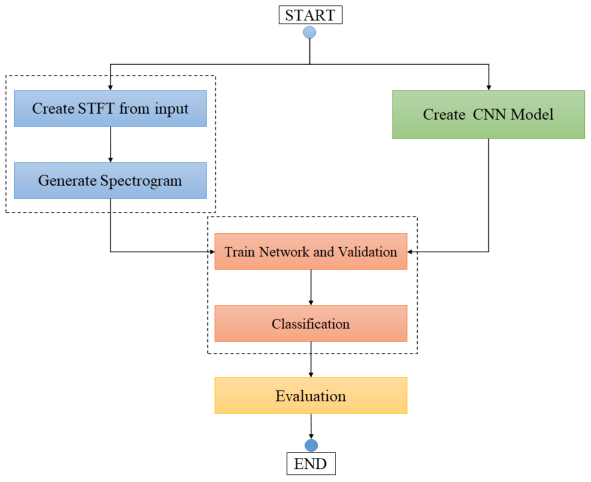

Fig. 1. Flowchart diagram of the experiment

A. Dataset

The dataset used in this experiment can be founded at [4] under ”Data” directory. We use files ”Data Object 1.xlsx” to ”Data Object 3.xlsx” as shown in (table I) to train the CNN model. While we use files ”T File 1.xlsx” to ”T File 12.xlsx” as shown in (table II) to do the classification experiment.

TABLE I
DATASET FOR TRAINING MODEL

Data	Sampling Signal	Description
Data Object 1.xlsx	315	1-D time signal data
Data Object 2.xlsx	200	1-D time signal data
Data Object 3.xlsx	400	1-D time signal data

TABLE II
DATASET FOR CLASSIFICATION

Data	Sampling Signal	Description
T File 1.xlsx	50	1-D time signal data
T File 2.xlsx	50	1-D time signal data
T File 3.xlsx	50	1-D time signal data
T File 4.xlsx	50	1-D time signal data
T File 5.xlsx	50	1-D time signal data
T File 6.xlsx	50	1-D time signal data
T File 7.xlsx	50	1-D time signal data
T File 8.xlsx	50	1-D time signal data
T File 9.xlsx	50	1-D time signal data
T File 10.xlsx	50	1-D time signal data
T File 11.xlsx	50	1-D time signal data
T File 12.xlsx	50	1-D time signal data

B. Preprocessing

Preprocessing is a method to pre-process on data by converting one dimensional data (1D data) into two dimensional data (2D data). Preprocessing composes of

1) Short-Time Fourier Transform: Preprocessing performs STFT on time signal data using (1), which is from Jont B. Allen [5], and generates spectrogram as a result as shown in “Fig. 2”, where the input is time signal data and the output is spectrogram. Spectrogram is a signal’s visual representation. It plots the amplitude of the signal’s frequency components over time.

$$X(f,t) = \int_{-\infty}^{\infty} w(t-\tau)x(\tau)e^{j2\pi f\tau}\, d\tau \tag{1}$$

where $X(f,t)$ is the spectrogram, t is the time variable, $w(t-\tau)$ is the shifted hamming window with the size of 256 sampling point, $x(\tau)$ is the input signal, and $exp(j2\pi f\tau)$ is the complex exponential.

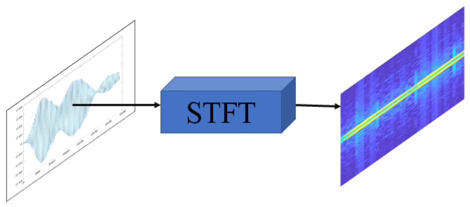

Fig. 2. Performing STFT to convert time signal data (Left) into Spectrogram (right)

2) Downsampling: Downsampling is used to reduce the size of the original spectrogram images. This process helps deduct computational cost, so the personal computer can run the training process. The original spectrogram image size is $875 \times 656 \times 3$, the size is then decreased to 20 percent of the original size, which is $175 \times 132 \times 3$. The same process has been applied to both datasets in (table I and table II).

C. Training and Testing

In the experiment, the supervised learning has been used to train the CNN model, which is shown as (Fig. 3). We train the model with the spectrogram images from preprocessing. By segregating 80 percent of each data from (table I) for training and 20 percent for testing or validation, the CNN model then adjusts weights accordingly. Note that each time we do the experiment, the spectrogram images are selected randomly.

(Fig. 3) shows the CNN layer diagram used in this experiment, while the spectrogram image is the input and classification result is the output. CNN layers’s details can be founded at (table III).

The process of training CNN model requires optimizer to updates weights and biases, which we use Stochastic Gradient Descent with Momentum (SGDM) algorithm to minimise the loss function [8]. SGDM algorithm is shown as (2)

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) + \gamma(\theta - \theta_{l-1}) \tag{2}$$

Where l is the iteration number, $\alpha > 0$ is the learning rate, θ is the parameter vector, and $\nabla E(\theta)$ is the loss function. $\gamma(\theta - \theta_{l-1})$ is the momentum term, which help reduce the oscillation in each iteration.

The algorithm evaluates loss function ($\nabla E(\theta)$ using the entire training set, which is denoted by Epoch value. the training algorithm requires parameters as (table IV).

Initial Learning Rate denotes how fast the training can process. If the Initial Learning Rate is too high, the training

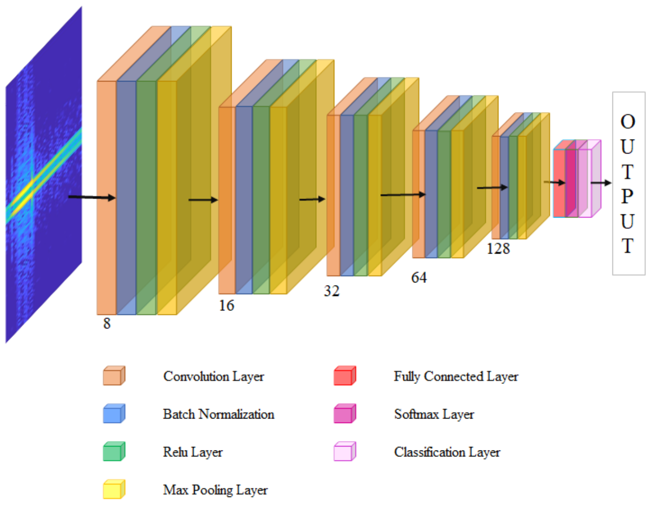

Fig. 3. CNN Layer Diagram

TABLE III
CNN LAYERS IN DETAIL

Layer	Description and size
Image Input	132×175×3 (width × height × depth) images with 'zerocenter' normalization
Convolution	8 filters 3×3×3 convolutions with stride [width = 1 height = 1] and padding 'same'
Batch Norm	Batch normalization with 8 channels
ReLU	Rectified Linear Units
Max Pooling	2×2 max pooling with stride [width = 2 height = 2] and padding [0 0 0 0]
Convolution	16 filters 3×3×8 convolutions with stride [1 1] and padding 'same'
Batch Norm	Batch normalization with 16 channels
ReLU	Rectified Linear Units
Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
Convolution	32 filters 3×3×16 convolutions with stride [1 1] and padding 'same'
Batch Norm	Batch normalization with 32 channels
ReLU	Rectified Linear Units
Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
Convolution	64 filters 3×3×32 convolutions with stride [1 1] and padding 'same'
Batch Norm	Batch normalization with 64 channels
ReLU	Rectified Linear Units
Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]
Convolution	128 filters 3×3×64 convolutions with stride [1 1] and padding 'same'
Batch Norm	Batch normalization with 128 channels
ReLU	Rectified Linear Units
Fully Connected	3 fully connected layer
Softmax	softmax
Classification Output	cross entropy loss function with 'object1' and 2 other classes

TABLE IV
TRAINING SETTING

Parameter	Value
Initial Learning Rate	0.01
Epoch number	30
Batch size	16

may exceed the optimal point too soon and has the low performance, while the low Initial Learning rate will result in long training time.

Epoch number indicates a full pass of the data in each iteration.

Batch Size is size for each training iteration, it is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights [8].

D. Evaluation

After we train CNN model, we evaluated the CNN model's performance with the following measurements derived from [6], [7], which uses confusion matrix to evaluate predictions and actual data. Confusion matrix consists of four values. True Positive (TP) represents the value of correct predictions of positive out of actual positive cases. False Positive (FP) represents the value of incorrect positive predictions. True Negative (TN) represents the value of correct predictions of negatives out of actual negative cases. False Negative (FN) represents the value of incorrect negative predictions.

1) Accuracy: Accuracy indicates the CNN model's performance on prediction. It is calculated as the ratio between correct prediction and overall prediction. The accuracy is measured by the equation from (3).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

2) Precision: Precision indicates the CNN model's ability in order to predict positive predictions correctly out of all positive predictions. Precision is measured by the equation (4). Increasing precision means increasing correctness of the positive prediction.

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

3) Recall: Recall is similar to precision, however recall measures the correctly positive predictions out of correct prediction. Recall is measured by the equation (5). Increasing recall value means minimising the incorrect predictions

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

4) F1-Score: F1-Score involves precision and recall. It balances between both measurements. F1-Score is measured by the equation (6). F1-Score is useful in the scenario where model gives high FP and FN.

$$F1-Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

E. Classification Experiment

The classification experiment uses the previous trained model from training and testing. The experiment is as simple as feed all spectrogram images from the untrained and untested dataset (table II) into the trained CNN model and then we acquire the classification result.

III. Experiment Result

There are two experiment results, which are resulted from CNN model training part and classification experiment.

1) CNN Model's Training Experiment: The CNN Model's training experiment has been conducted for 100 times and evaluated using the evaluation method from methodology part. The result of training experiment is as shown in (table V), which is the average result of 100 experiment times. The evaluation result can be found at [4] under MATLAB directory and filename "experiment_result.csv".

TABLE V
AVERAGE CNN MODEL TRAINING EVALUATION

Accuracy	Precision	Recall	F1-Score
0.879454	0.905669	0.890216	0.90067591

One of the confusion matrix of the experiment is shown as (Fig.4). There are total 915 spectrogram images from 3 objects (table I) used in training and testing. By the ratio of 80 percent training and 20 percent testing results in 732 data images for training and 183 data images for testing.

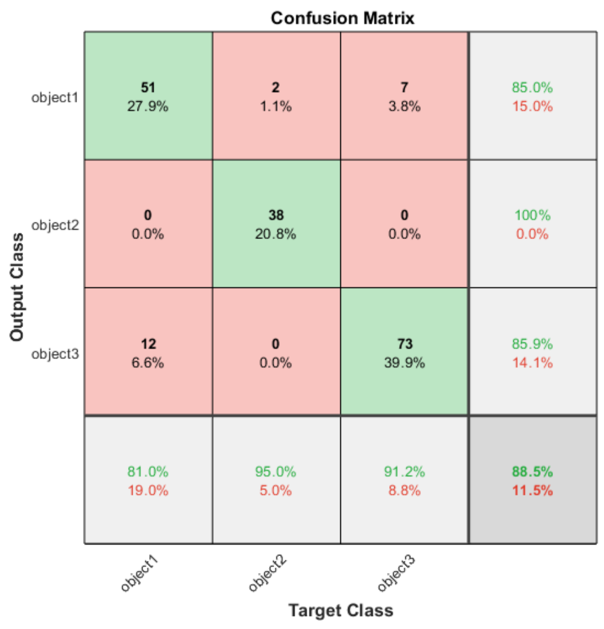

Fig. 4. Confusion matrix from one of the experiment

2) Classification Result: Classification results are the output of classification experiment. The process uses classification dataset (table II) and classifies using the trained CNN model from training and testing part. The dataset consists of 12 Objects, each object has 50 sampling signals. The result of classification is shown by (tableVI) and a bar graph shows the potentiality (Fig. 5). The classification result can be found at [4] under MATLAB directory and filename is "classified_result.csv";

TABLE VI
CLASSIFICATION RESULT IN PERCENTAGE

Object Name	Classified as Object 1 (%)	Classified as Object 2 (%)	Classified as Object 3 (%)
T1	100	0	0
T2	100	0	0
T3	84	16	0
T4	88	12	0
T5	88	12	0
T6	12	0	88
T7	70	0	30
T8	60	0	40
T9	64	0	36
T10	80	0	20
T11	82	0	18
T12	0	100	0

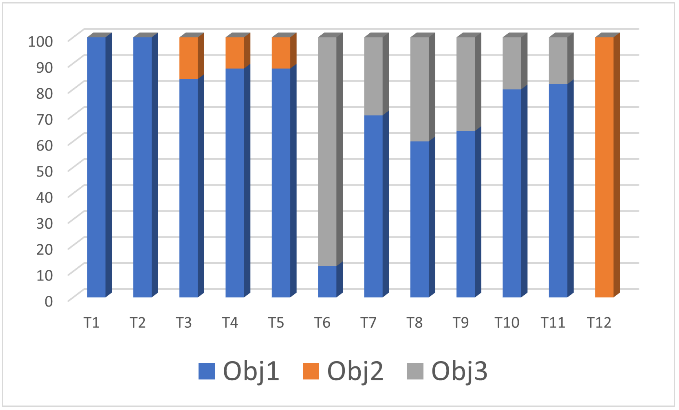

Fig. 5. Potentiality from classification result

IV. Discussion and Conclusion

In the paper, we proposed the method to classify time signals with CNN model using STFT. The methodology in our paper converts the 1-D time signals into 2-D spectrogram images, which contain time, amplitude, and frequency. We trained the CNN model with three known objects, which has overall 915 images. Our CNN model then extracted features and adjust weights using supervised learning technique, which the label objects are known. After performing training for 100 times, we evaluated the CNN model. It achieves high evaluation score, accuracy 0.88, precision 0.90, recall 0.90, and F1-Score 0.89. The trained CNN model is then used to classify another unlabeled dataset.

V. Encounter Problems

Although the evaluation of training experiment shows the promising result, as we conduct the experiment for 100 times, we found the low evaluation result in 1 of 100 times. The cause is still unknown, but we make an assumption that since the data from training dataset (table I) is selected randomly, seldom the process selects poor samples for training and resulting in low adjusting in weights and biases.

VI. Commitment

The commitments of each member in this experiment can be found at [4]

References

[1] Y. Tokozume and T. Harada, “Learning environmental sounds with end-to-end convolutional neural network,” in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017, pp. 2721–2725.

[2] Ravanelli, M., & Bengio, Y. (2018). Speaker recognition from raw waveform with sincnet. In 2018 IEEE Spoken Language Technology Workshop (SLT) (pp. 1021–1028).

[3] Venkatesh Boddapatia, Andrej Petefb, Jim Rasmussonb, Lars Lundberg, ”Classifying environmental sounds using image recognition networks”, International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

[4] P. Pongsomboon, M. Pirmoradian, Md Mukit Khan, and Md Rabiul Islam ”Classification-of-time-signals-by-CNN-using-STFT” Retreived from Github - https://github.com/paweenp/Classification-of-time-signals-by-CNN-using-STFT, 2021.

[5] Jont B. Allen ”Short Time Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform”. IEEE Transactions on Acoustics, Speech, and Signal Processing. ASSP-25 (3): 235–238, June 1977.

[6] Ahmed Fawzy Gad, ”Evaluating Deep Learning Models: The Confusion Matrix, Accuracy, Precision, and Recall”, published on 2020, accessed on 25 Aug 2021.

[7] Ajitesh Kumar, ”Accuracy, Precision, Recall & F1-Score”, published on 3 September 2021, accessed on 20 September 2021.

[8] Kevin Patrick Murphy, ”Machine Learning: a Probabilistic Perspective”, The MIT Press, Cambridge, Massachusetts, 2012.