"Environmental Sound Classification Using Spectrograms with CNN Model"

Feridun Cemre Gulten[†], Ceren Yilmaz[‡]

***Abstract*—Environmental sound classification plays a crucial role in various applications such as soundscape analysis, acoustic monitoring, and environmental impact assessment. However, accurately categorizing environmental sounds poses challenges due to the complexity and variability of the acoustic signals. In this paper, we address this problem by proposing a novel approach that leverages spectrograms and convolutional neural networks (CNNs) for environmental sound classification. The findings of this study contribute to the advancement of environmental sound classification and have potential implications for various domains such as soundscape management and environmental monitoring.**

***Index Terms*—Sound Classification, CNN Model, Deep Learning,**

I. Introduction

The purpose of this project is classifying different sound groups based on their spectrogram using CNN model. The goal of this is getting higher accuracy compared to other classification models. In the project, the CNN is used as a classification model since the image representation of the audio files are used. The project is consisting of multiple data pre-processing steps to get the image representation of the audio files and labeled the data.

In today's rapidly advancing technological landscape, sound classification has emerged as a crucial field with significant relevance and importance. As our lives become increasingly intertwined with audio-rich environments, the ability to accurately classify and analyze sounds has become fatal. Sound classification finds its applications in diverse domains such as speech recognition, environmental monitoring, acoustic surveillance, and music recommendation systems. With the advent of smart devices, virtual assistants, the need for efficient sound classification algorithms has been proposed. For this reason, there are several research and developments about this topic. (e.g. [1], [2], [3])

The sound classification can be done by various machine learning (ML), deep learning methods. The audio type to classified, can be different for each case. The different sound groups can have different specific features parameters. These parameters can be analyzed based on raw audio or image representation of audio. Researches show the image representation is more efficient way in most cases. Different spectrogram types can be used to represent audio signals in frequency domain. The deep neural network-based methods can applied to sound classification. There are various deep learning/machine learning libraries to help to build the deep learning/machine learning models. Some of the python libraries are are "PyTorch", "TensorFlow", "Keras". These libraries are generally similar usage but the advantages are different. So, while selecting the built in library, the advantages of each one of them needs to be considered.

In this project, the ESC-50 dataset is used. This dataset consist of labeled 2000 environmental audio recordings. The dataset is organized into five distinct groups. Three different classification tasks are performed using this dataset: classification on the entire ESC-50 dataset, classification on the restricted ESC-10 dataset, and classification within each of the five sound groups. Notably, spectrograms are employed as input features for classification, and a Convolutional Neural Network (CNN) is chosen as the deep learning model due to its proven effectiveness in similar research on sound classification. In this paper, we propose a CNN model implemented using the PyTorch library, providing a detailed explanation of the data preprocessing steps employed to generate spectrograms and train the CNN model.

This report (paper) is structured as follows. In Section II we describe the related works, the data models are respectively presented in Sections III. The proposed dataset and preprocessing process is detailed in Section IV and in V the learning algorithm and strategy included. In the VI, the results are shown and finally VII it includes conclusion, future work of the project.

II. Related Work

The research [1] study utilizes the ESC dataset to compare the results of human manual classification with machine learning classification methods. For machine learning classification, two significant features, namely zero-crossing rate and mel-frequency cepstral coefficients (MFCC), are selected to characterize the sounds. Three classification algorithms, namely k-nearest neighbors (k-NN), random forest ensemble, and support vector machine (SVM) with a linear kernel, are employed for the classification task. The study concludes by proposing the evaluation of deep neural networks and other deep learning models in the context of environmental sound classification, including the utilization of unsupervised pre-training with the ESC-US dataset. This proposition aims to further enhance the performance and accuracy of sound classification systems.

The primary objective of this paper [2] is to assess the

[†]Department of Information Engineering, University of Padova, email: {feriduncemre.guelten}@studenti.unipd.it

[‡]Author two affiliation, email: {ceren.yilmaz}@studenti.unipd.it

Special thanks / acknowledgement go here.

effectiveness of Convolutional Neural Networks (CNN) in classifying short audio clips of environmental sounds. The study incorporates three different publicly available datasets, including the ESC-50 dataset. To facilitate classification, log-scaled mel-spectrograms are employed as the feature representation. The research findings reveal that across all cases, models based on CNN outperform implementations utilizing manually-engineered features. Particularly notable is the superior performance of CNN models when classifying audio events from more diverse and varied categories. Based on these results, the researcher suggests that CNN models are the optimal choice for environmental sound classification tasks, highlighting their potential for accurately categorizing environmental audio clips.

The primary objective of this paper [3] is to examine the limitations of sound classification research. Through an extensive investigation of various studies and classification approaches, a significant challenge identified is the difficulty in accurately distinguishing acoustic scenes encountered in everyday environments. Moreover, the paper proposes that deep learning models, particularly those based on deep neural networks, yield more precise results in sound classification tasks. By highlighting the limitations of existing research and emphasizing the efficacy of deep learning models, this study aims to enhance our understanding of sound classification techniques and encourage further advancements in this field.

This paper [4] introduces a novel environmental sound classification model called Temporal-Frequency Attention based Convolutional Network (TFCNN). The proposed model leverages the log-mel spectrum to simultaneously capture frequency and temporal features. Experimental evaluations are conducted using two publicly available datasets. The research findings reveal that the TFCNN model exhibits lower complexity compared to other models in related works, while achieving higher accuracy in sound classification. However, the accuracy of the model varies across different groups of sounds. In light of this, the paper proposes future works to optimize the weighted combination strategy to address the varying accuracy levels observed. These proposed enhancements aim to further improve the performance and robustness of the TFCNN model in environmental sound classification tasks.

Upon reviewing the existing literature in this field, it becomes evident that the most impressive results are consistently achieved using Convolutional Neural Network (CNN) models or similar deep learning models when addressing the subject of sound classification. Moreover, spectrogram-based features have been widely recognized as an effective choice for achieving high-performance outcomes. In light of all these related works, in the project convolutional neural network (CNN) is used as a model and as a audio 2D version of the audio is used.

III. Processing Pipeline

In this paper efficient model is proposed to sound classification. In the project multiple steps is followed to get the proper results based on the proposed model. The steps consist of data preprocessing, classifying the sound groups based on the target value in the csv file, and converting each sound WAV format to PNG format which represents the spectrogram and built and train the CNN model. The flow diagram of the process can be shown visually in the figure below.

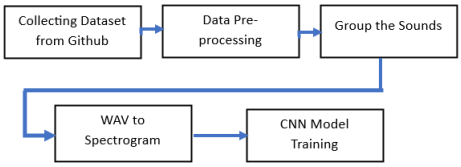

Fig. 1: Flow Chart of Sound Classification

The ESC-50 dataset [5] used in the project as a main data. All audio files are in WAV format. Originally, the audios were 5 seconds long, but they were later truncated to 3 seconds to eliminate any initial or trailing silence. Therefore, only the first 3 seconds of each audio file were retained for classification purposes. In subsequent stages of the project, these recordings were converted into spectrograms using the Librosa library. Among the various types of spectrograms, the Mel Spectrogram proved to be the most effective for this project. The project consists of three tasks, and the same process was applied to each task for sound classification.

Using PyTorch's transforms module data transformation were applied and training/ validation dataset from image files in a directory were constructed. Then, the training dataset randomly splitted into a training and validation set based on a specified split ratio. The train-test split was done as $\%70$ train set and $\%30$ test set. After that, data was loaded using PyTorch considering the batch size(25) and shuffling to prevent the model from memorizing the order of the samples.

The generated image representation of the audio files which is spectrogram, were classified using CNN model which is provided by PyTorch. It takes a 3-channel input, an RGB image of a spectrogram, and passes it through five convolutional layers, each followed by batch normalization, ReLU activation, and max pooling. The output of the last convolutional layer is then flattened and passed through a fully connected layer with output nodes, followed by a softmax activation to produce a probability distribution for output.

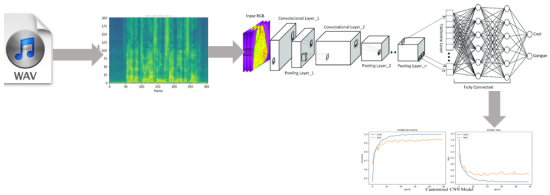

Fig. 2: Classification Process Scheme

IV. Signals and Features

The ESC-50 dataset [5] consists of 2000 labeled environmental sounds. It is organized into 5 main groups, each containing 10 subgroups. The main groups include Animals, Human/Non-speech Sounds, Natural Soundscapes & Water Sounds, Interior/Domestic Sounds, and Exterior/Urban Noises. The original dataset contains recordings that are 5 seconds long and are saved in WAV format. However, upon examining the dataset, it was observed that many recordings have empty sections at the end. Therefore, the recordings were trimmed, resulting in 3 seconds-long versions. In the CSV file of the dataset, there are target value of the sound group. While the sound files grouped for the training model, this target value of the each sound file is considered. This project encompasses three distinct classification tasks, each focusing on different targets. The classification tasks range from a more general classification to a narrower one, based on the specific objectives of the task.

In the first task, the sounds were classified based on the 5 main groups, resulting in a more generalized classification. The second task utilized the ESC-10 dataset, which is a subset of the ESC-50 dataset. This dataset contains a specific column for the ESC-10 subset. Finally, in the last task, the entire ESC-50 dataset was used for classification, aiming to provide the subcategory label as the output instead of the general category.

WAV(Waveform Audio File Format) format is a type of digital audio file format like MP3. However, WAV is an uncompressed audio format that stores audio data in a lossless manner. This means it captures and store the original audio features. So, it takes more space compared to the MP3 format. For this reason, it generally used in professional audio production. Since, the features in audio format can be hard to distinguish, extracting the features from any type of spectrogram more proper and efficient way.

A spectrogram is a visual representation of the frequency content of an audio signal over time. It is essentially a 2D graph where x-axis represents time, the y-axis represents frequency, and the color or intensity of each point in the graph represent the amplitude of the signal at that point in time and frequency. WAV format is time-domain representation of audio signal. Fourier transform is used to convert time-domain to frequency-domain. It essentially breaks down the audio signal into its individual frequency components allowing to analyze the frequency content of the signal over time. After this transformation, the signal can be used to create a spectrogram by dividing the audio signal into short time windows. While transforming the WAV to spectrogram, there are several methods in python. The different type of spectrograms can be used as different analysis approach. Each spectrogram save as PNG format which is a image representation.

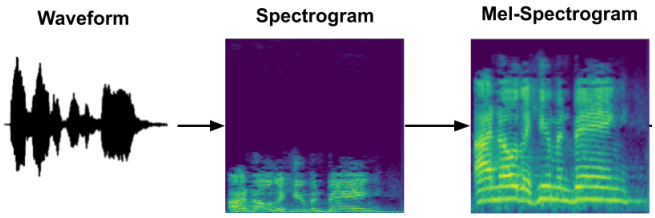

Fig. 3: WAV Format to Spectrogram Format Representation

The images of the spectrograms were taken from the train data folders which have the size as 288x384 . After the images loaded to the tensors the images shape became as ([25, 3, 288, 384]). This is the size of the image while it used as input to model. The feature extraction was done in the model part, the number of the feature is based on batch size.

V. Learning Framework

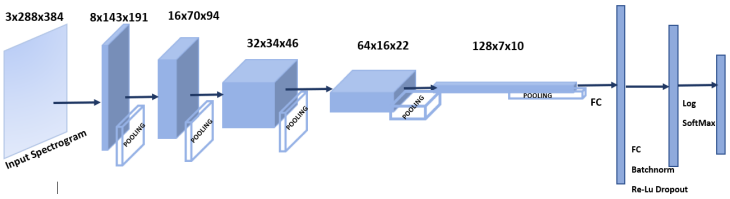

Fig. 4: CNN Model Diagram with Layer

A. General Explanation

In the project a baseline convolutional neural network (CNN) model designed for sound classification. It takes a 3-channel input, an RGB image of a spectrogram, and passes it through five convolutional layers, each followed by batch normalization, Re-LU activation, and max pooling. The output of the last convolutional layer is then flattened and passed through a fully connected layer with 4 output nodes, followed by a Softmax activation to produce a probability distribution for output.

The architecture of this model follows a common pattern for image classification tasks, where the initial convolutional layers extract local features from the input image, while the subsequent layers combine these features to form increasingly abstract and high-level representations. The batch normalization layers help to reduce internal co-variate shift, while the dropout layer helps to regularize the model and prevent over fitting.

The choice of kernel size, stride, and padding in each

convolutional layer determines the receptive field of the filters and the degree of down sampling. In this model, the kernel size is fixed at 3x3, the stride is fixed at 1, and no padding is used. This leads to a gradual reduction in the spatial dimensions of the feature maps through the pooling layers, resulting in a more compact representation that can be more easily processed by the fully connected layer.

B. Layer's Channels Explanation

in˙channels and out˙channels are parameters in the convolutional layers that define the number of input and output channels, respectively. In a CNN, each convolutional layer learns a set of filters, where each filter is a small matrix of weights that is convolved with the input feature maps to produce a set of output feature maps.

The in˙channels parameter specifies the number of input feature maps to the convolutional layer. For example, in the first convolutional layer in this model, in˙channels is set to 3, since the input is a 3-channel image (likely an RGB spectrogram).

The out˙channels parameter specifies the number of filters to be learned by the convolutional layer. Each filter produces one output feature map, so the total number of output feature maps is equal to out˙channels. In this model, the number of output channels increases from 8 to 128 in each successive convolutional layer, indicating that the model is learning increasingly complex and abstract features as it progresses through the layers.

C. Batch Normalization

In this model, each batch normalization layer is applied after its corresponding convolutional layer. Since the number of output channels from the convolutional layers increases from 8 to 128, the num˙features parameter is set accordingly for each batch normalization layer to match the number of output channels from the corresponding convolutional layer.

For example, in the first batch normalization layer (self.batchnorm1 = nn.BatchNorm2d(num˙features=8)), num˙features is set to 8, which matches the number of output channels from the first convolutional layer (out˙channels=8). This ensures that the statistics (mean and variance) used for normalization are calculated separately for each channel, and that each channel is normalized independently of the others.

D. Numerical Explanation

- The input image size is 288x384.
- After passing through the first convolutional layer with a kernel size of 3 and stride of 1, the output size will be (288-3+2*0)/1 + 1 = 286x382.
- After passing through the first max pooling layer with a kernel size of 2, the output size will be 143x191.
- After passing through the second convolutional layer with a kernel size of 3 and stride of 1, the output size will be (143-3+2*0)/1 + 1 = 141x189.
- After passing through the second max pooling layer with a kernel size of 2, the output size will be 70x94.
- After passing through the third convolutional layer with a kernel size of 3 and stride of 1, the output size will be (70-3+2*0)/1 + 1 = 68x92.
- After passing through the third max pooling layer with a kernel size of 2, the output size will be 34x46.
- After passing through the fourth convolutional layer with a kernel size of 3 and stride of 1, the output size will be (34-3+2*0)/1 + 1 = 32x44.
- After passing through the fourth max pooling layer with a kernel size of 2, the output size will be 16x22.
- After passing through the fifth convolutional layer with a kernel size of 3 and stride of 1, the output size will be (16-3+2*0)/1 + 1 = 14x20.
- After passing through the fifth max pooling layer with a kernel size of 2, the output size will be 7x10.

Finally, the output of the last convolutional layer is flattened and passed through a fully connected linear layer. The number of input features for the model is 7 x 10 x 128 = 8960.

E. Forwarding The Model

Given an input tensor x, the forward method applies a sequence of convolutional, batch normalization, activation, and max pooling layers to extract features from the input.

1) Convolutional layer 1: x = self.conv1(x) applies the first convolutional layer to the input tensor.
2) Batch normalization layer 1: x = self.batchnorm1(x) applies batch normalization to the output of the first convolutional layer. This normalizes the output to have zero mean and unit variance.
3) Re-LU activation layer 1: x = F.relu(x) applies the rectified linear unit (ReLU) activation function to the output of the first batch normalization layer. This introduces non linearity to the model and helps to prevent the vanishing gradient problem.
4) Max pooling layer 1: x = F.max_pool2d(x, kernel_size=2) applies max pooling with a 2x2 kernel to the output of the first ReLU activation layer. This reduces the spatial size of the output tensor by a factor of 2.
5) Repeat steps 1-4 for convolutional layers 2-5.
6) Flatten: x = torch.flatten(x, 1) flattens the output tensor into a 1D vector for input to the fully connected layer.
7) Dropout: x = self.dropout(x) applies dropout regularization to the flattened output tensor. This randomly drops out some units to prevent over fitting.
8) Fully connected layer 1: x = self.fc1(x) applies the fully connected layer to the output of the dropout layer. The output tensor has 4 units, which correspond to the 4 possible classes.
9) Softmax activation: x = F.softmax(x) applies the softmax activation function to the output of the fully connected layer. This converts the output into class probabilities.

F. Training The Model

A training method was constructed that takes the model, a device (GPU or CPU), training data loader, validation data loader, and number of epochs as inputs.We initialized the cross-entropy loss function and Adam optimizer with a learning rate of 0.0001. In each epoch, the model is set to training mode using model.train(). The loop then iterates over each batch of data in the training data loader. For each batch, the data and target labels are sent to the specified device. The model is called with the data batch to get the output. The optimizer's gradients are zeroed out with optimizer.zero˙grad(). The loss is computed using the cross-entropy loss function. The gradients are then computed using loss.backward(), and the weights are updated using optimizer.step(). The running loss and accuracy for this batch are recorded and printed.

The model is then set to evaluation mode using model.eval(). The loop then iterates over each batch of data in the validation data loader. For each batch, the data and target labels are sent to the specified device. The model is called with the data batch to get the output. The loss is computed using the cross-entropy loss function. The running loss and accuracy for this batch are recorded and printed.

VI. Results

A. 5 Main Category Classification Task

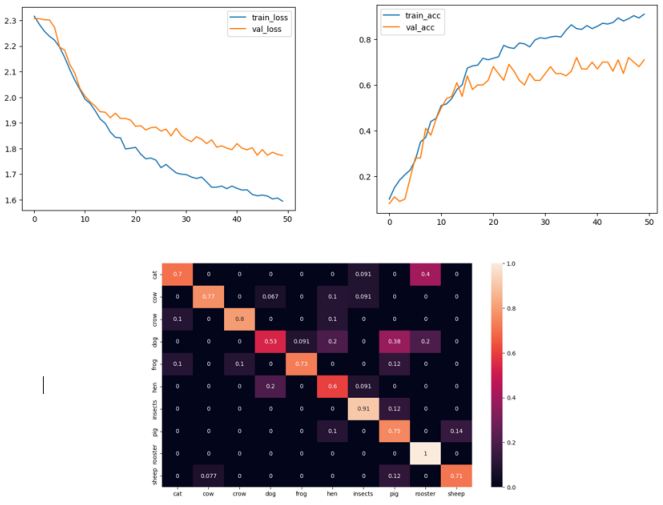

Fig. 5: Result of the Animal Sounds Classification

1) Animal Sounds: The loss and the accuracy values were plotted as shown in the Figure 5, for the first thematic group which is Animals. From the plot it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time. Training accuracy is observed as higher than the validation accuracy which expected. At the and the validation accuracy is found as %70. Finally, the confusion matrix was plotted with normalized version and the relation between true and predicted values were observed. Correct predictions (diagonal cells) for some of the classes were yielded higher than the others. Off-diagonal rates show that there are some errors while predicting the correct class.

2) Natural Soundscapes & Water Sounds: The loss and the accuracy values were plotted as shown in the Figures 6 for the second thematic group which is Natural soundscapes & water sounds. From the plots it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time. Training accuracy is observed as higher than the validation accuracy which is expected. At the and the validation accuracy is found as %70. Finally, the confusion matrix was plotted with normalized version and the relation between true and predicted values were observed. Correct prediction (diagonal cells) for some of the classes were yielded higher than the others. Off-diagonal rates show that there are some misclassifications. The crickets sound obtained with very low prediction rate for this group.

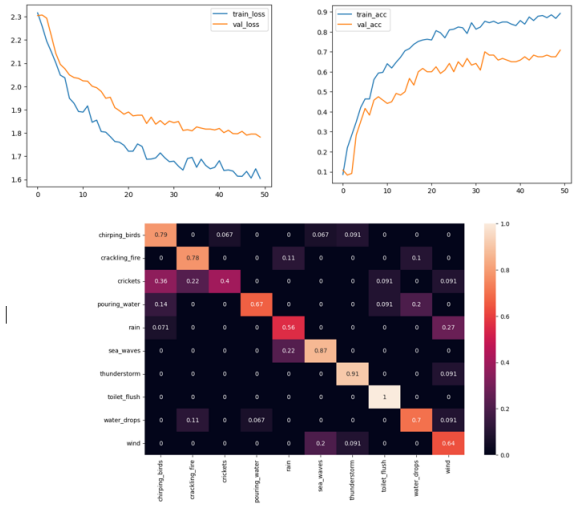

Fig. 6: Result of the Natural Soundscapes & Water Sounds Classification

3) Human Sounds: The loss and the accuracy values were plotted as shown in the Figure 7 for the third thematic group which is Human, non-speech sounds. From the plots it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time. Training accuracy is observed as higher than the validation accuracy which is expected. At the and the validation accuracy is found as %70. Finally, the confusion matrix was plotted with normalized version and the relation between true and predicted values were observed. Correct prediction (diagonal cells) for some of the classes were yielded higher than the others. There are %100 correctly predicted 3 classes. Off-diagonal rates show that there are some misclassifications. For this group the drinking-sipping sound has the lowest prediction rate.

4) Interior/Domestic Sounds: The loss and the accuracy values were plotted as shown in the Figure 8 for the fourth

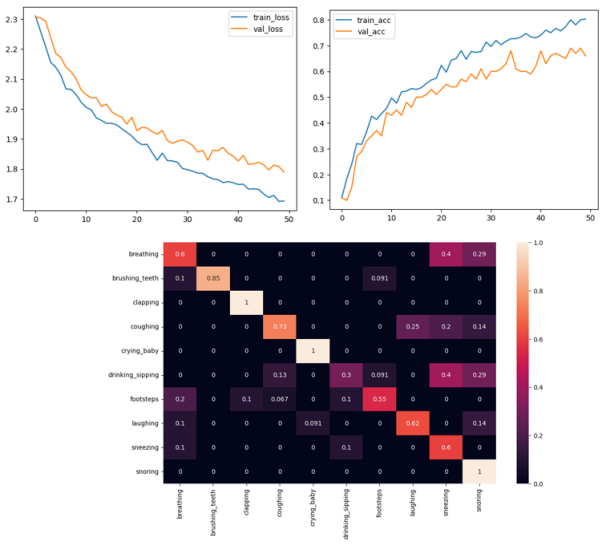

Fig. 7: Result of the Human Sounds Classification

thematic group which is Human, non-speech sounds. From the plots it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time. Training accuracy is observed as higher than the validation accuracy which is expected. At the and the validation accuracy is found as %70. Finally, the confusion matrix was plotted and the relation between true and predicted values were observed. Correct prediction (diagonal cells) for some of the classes were yielded higher than the others. There are %100 correctly predicted 2 classes. Off-diagonal rates show that there are some misclassifications.

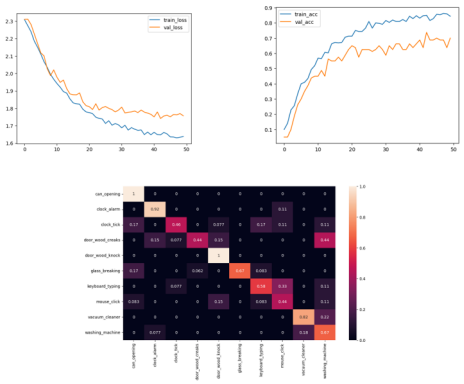

Fig. 8: Result of the Interior/Domestic Sounds Classification

5) Exterior/Urban Sounds: The loss and the accuracy values were plotted as shown in the Figure 9 for the fifth thematic group which is Exterior/urban noises. From the plots it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time. Training accuracy is observed as higher than the validation accuracy which is expected. At the and the validation accuracy is found as round %80. Finally, the confusion matrix was plotted with normalized version and the relation between true and predicted values were observed. Correct prediction (diagonal cells) for some of the classes were yielded higher than the others. Off-diagonal rates show that there are some misclassifications. For this group the engine sound has the lowest prediction rate. The engine sound mostly confused with helicopter sound.

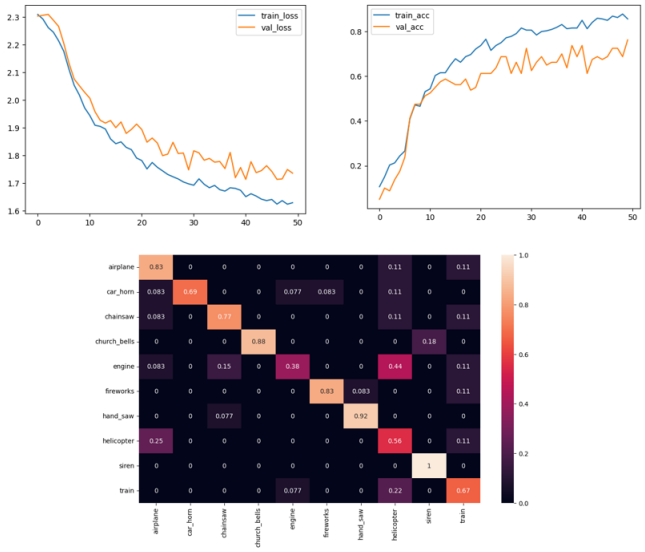

Fig. 9: Result of the Exterior/Urban Sounds Classification

B. *ESC-10 Classification Task*

The loss and the accuracy values were plotted as shown in the Figure 10 for the ESC-10 group. From the plots it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time. Training accuracy is observed as higher than the validation accuracy which is expected. At the and the validation accuracy is found as %70. Finally, the confusion matrix was plotted and the relation between true and predicted values were observed. Correct prediction rates (diagonal cells) for some of the classes were yielded higher than the others. Off-diagonal rates show that there are some misclassifications between some classes.

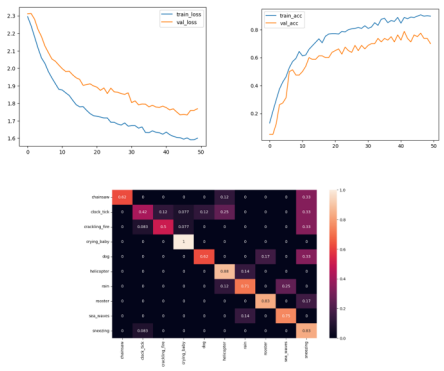

Fig. 10: Result of the ESC-10 Classification Task

C. ESC-50 Entire Dataset

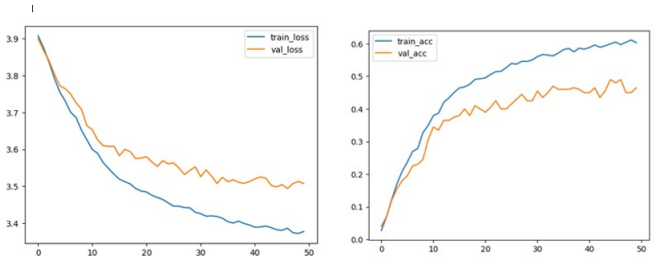

Fig. 11: Result of the ESC-50 Classification Task

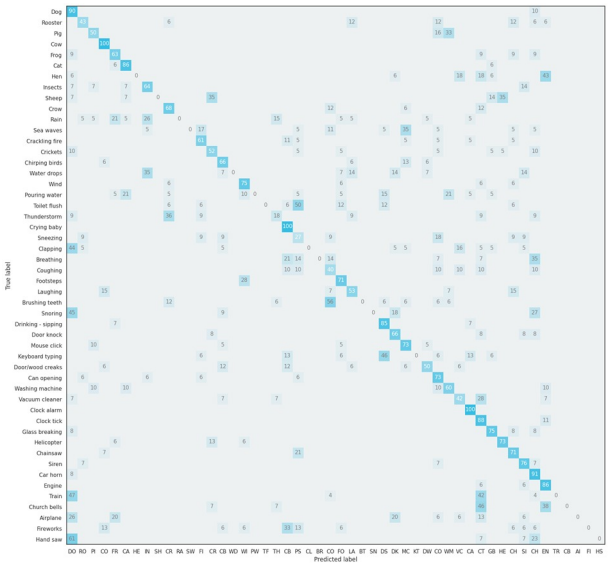

Fig. 12: Result of the ESC-10 Classification Task

The loss and the accuracy values were plotted as shown in the Figure 11 and 12 for the ESC-50 group. From the plots it can be observed that the loss is decreasing while the accuracy is increasing through the epochs time . Training accuracy is observed as higher than the validation accuracy which is expected. The loss amount was observed more than the other classification tasks.Because there are 50 class higher amount loss was expected. Accuracy is also found as 0.45 which is very low then the other classification tasks. Because the training data amount is increased the training of the model with this data was more challenging. Optimizing and fine-tuning the parameters may resulted more efficient classification task for this data set. Finally, the confusion matrix were plotted and the relation between true and predicted values were observed. Correct prediction rates(diagonal cells) for some of the classes were yielded higher than the others.Off-diagonal rates show that there are some misclassification between some classes. As shown in the figure there are many non classified classes also.However, correctly predicted classes still observed.

D. Discussion

Different accuracies were observed for different classification tasks. Even the accuracy is not higher than %90, appropriate predictions and performance can be obtained using the model. From the plots of loss and accuracy, under fitting or over-fitting problems was not observed significantly. In the plots small peaks or noises in the training and validation loss curves may indicate several possibilities. Firstly, the learning rate; the learning rate determines the step size taken during gradient descent optimization. If the learning rate is too high, it can lead to oscillations or overshooting, causing the loss curve to exhibit peaks or irregularities. Even the learning rate was not too high, it is tried to be optimized but some oscillations were observed. Secondly, model complexity: may the model be complex relative to the available data or the problem's complexity, it may exhibit small peaks or irregularities in the loss curves. This could be a sign of the model trying to fit noise in the data or struggling to generalize effectively. Another effect is regularization strength; the regularization technique used (e.g., dropout, weight decay) may affect the loss curves. Finally, optimization convergence; the loss curves may exhibit small peaks during the optimization process, even when the model is not over-fitting or under-fitting. These fluctuations can occur due to the optimization algorithm searching for a better solution in the parameter space.

While the classes for each classification tasks were obtained correctly some incorrectly predicted instances yielded. The reasons may be attributed to some reasons such as similarity between some characteristics. For example, in the animal group, for the dog class, there is wrong prediction as a pig, it is not too high but should be investigated. Also, for the helicopter and the engine classes have similarities when investigated. No need to show all of them but here are some samples taken from the dataset to show similarities between the classes.

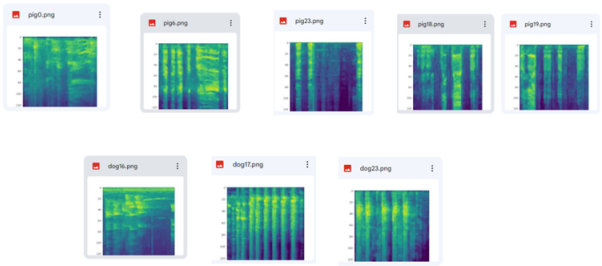

Fig. 13: Pig vs. Dog Similarities from Spectrogram

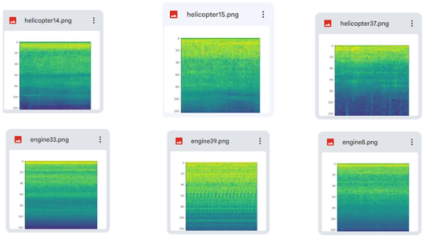

Fig. 14: Helicopter vs. Engine Similarities from Spectrogram

The accuracies for the tasks may also be affected from the quality and size of the image. While saving the spectrograms adding some qualification process may yield with different results. Additionally, the batch size and image size have the following effects: The first one is batch size. Batch size determines the number of samples that will be propagated through the network at once during the training process. A larger batch size can lead to faster training because it allows for parallel processing on the GPU. However, using a very large batch size may require more GPU memory. On the other hand, a smaller batch size may result in slower training but can provide more frequent weight updates, potentially leading to better convergence. In this project because Google Colab is used, the memory usage was limited so bigger batch size was not used. Different batch sized were tried such as 32 or 64 but efficient results could not be obtained as much as 25 which is used for this project. The second one is Image Size. Resizing the images to a specific size (in this case, 288) has implications for memory usage and computational efficiency. Increasing the image size can lead to more detailed representations but requires more memory. It can also increase the computational cost of training since larger images require more computations for convolutions and pooling operations. Resizing the images to a smaller size can help reduce memory usage and training time but may result in a loss of some information.

VII. Concluding Remarks

CNN model gave efficient result for the environmental sound classification task. There were some struggles points during the process which are GPU and capacity problems. Solving these problems may give higher accuracy by using more dataset samples. In the end, the classification was successfully done by CNN model with PyTorch library. For the future improvements, more than one model can be combined, even more than one spectrogram type can be used. These combination of different approaches can give higher accuracy. However, it may cause more complicated problems too. In the project deep learning models are deeply investigated. Even model was used in the project one, while searching the proper methods various models was studied. The different Python libraries were deeply investigated to use them by efficiently. Understand the general problem and concept is an advantage of the learning outcome. Another learning point is analysing and preprocessing the data. This point is one of the hardest part but learning outcome is high too. Since, the dataset includes different classes with 40 sounds, they must be replaced to the correct class and obtaining spectrograms of each sounds needs time. Capacity was also a struggling part of the project. Since we do not have Google Collab Pro, the GPU was not usable in sometimes. This caused the process to slow down. For example, changing the epochs parameter can affect the learning accuracy. However, when its increased system can crush because of the overload. However, generally this problem was solved with different approaches such as using less epochs or using CPU.

References

[1] K. J. Piczak, “ESC: Dataset for Environmental Sound Classification,” in *23rd ACM international conference on Multimedia*, (), 2015.

[2] K. J. Piczak, “ENVIRONMENTAL SOUND CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS,” in *2015 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING*, (Boston, USA), 2015.

[3] H. M. J. T. J. Nogueira, A.F.R.; Oliveira, “Sound Classification and Processing of Urban Environments: A Systematic Literature Review.,” in *Sensors 2022*, (), 2022.

[4] Y. B. H. X. e. a. Mu, W., “Environmental sound classification using temporal-frequency attention based convolutional neural network,” in *Sci Rep 11*, ().

[5] K. J. Piczak, “ESC: Dataset for Environmental Sound Classification,” in *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, ACM Press.