THE PROCESS OF TRAINING A GENERAL-PURPOSE AUDIO CLASSIFICATION MODEL

Petar Petrović,
Nemanja Ćoso,
Sanja Maravić Čisar*,
Robert Pinter

Subotica Tech-College of Applied Sciences,
Subotica, Serbia

Abstract:
Branches of machine learning such as image classification, object detection and speech recognition are more commonly used in modern devices today than ever before. Most smartphones released in the last five years have at least one function that depends on one of the aforementioned fields. Google allows users to make a query based on a speech input which is converted into text, cameras on both iOS and Android devices have built-in object and face detection, and gallery apps can automatically sort photographs based on their content. Speech recognition falls under the category of audio classification, which also contains subfields like music genre classification, song identification, automatic audio equalization, voice-based identification, etc. This paper describes the basic steps of training a general audio classification model which can predict a limited number of distinct sounds, and it outlines the techniques that are employed during the process of training any sound classification model, regardless of its intended usage.

Keywords:
Sound classification, TensorFlow, Raspberry, Neural networks, Python.

INTRODUCTION

Sound classification is an area in which principles and techniques are applied in the field of signal processing, statistics, psychoacoustics [1]. In machine learning, the term "sound classification" can describe problems such as speech recognition, speech transcription (speech-to-text), voice recognition, music genre classification, etc., and these problems can differ in their implementations and applications. Depending on their objective and eventual outcome, these issues can be classified in a variety of ways. For example, models for music genre classification usually use convolutional neural networks (CNNs) in their architecture [2], [3], while models for speech recognition use recurrent neural networks (RNNs) [4]. The initial steps during model development of any sound classification model are almost always the same in all sound classification techniques, and they represent solutions to a few key problems that arise during the development of these models.

Correspondence:
Sanja Maravić Čisar

e-mail:
sanjam@vts.su.ac.rs

DOI: 10.15308/Sinteza-2022-81-88

2. DIGITAL REPRESENTATION OF SOUND

Sound is a mechanical wave produced by a source's periodic oscillation which affects the pressure of its surroundings or medium, and the information that they carry can be represented as an analog signal [5]. Classifications models take tensors which contain numbers which represent information regarding sound as input. As sound waves are represented as analog signals, we must first convert them to their equivalent digital representation which can be read by the model.

Signal sampling is the process where a continuous, analog signal is translated into a discrete, digital representation by recording the signal amplitudes at fixed intervals and assigning a value to the recorded amplitudes from finite range of numbers that suit their intensity [6]. The number of samples per second represents the sampling frequency, while the number of possible values for the amplitude represents the bit depth or resolution [7]. As an example, the sampling rate and bit depth of CD-quality audio files is 44100 Hz and 16 bits, respectively [8]. In other words, the audio signal is sampled 44100 times per second, and each sample might have up to 16 different values. In general, in the case of an audio file the higher these numbers are, the digital representation is more accurate and high-quality (Figure 1).

3. AUDIO FEATURE EXTRACTION

When a digital form of sound exists, it is necessary to find out how to extract its characteristics which contain information that can be used to classify it into a predefined category. In other words, we need to find out what makes an audio signal unique, which of its features give it its "identity" and how to use those features to classify it.

According to Fourier analysis, any physical signal can be decomposed into a finite number of discrete frequencies, that is the sum of these frequencies represents the original, complex signal [9]. This collection of frequencies represents the spectrum of a function and it shows the amplitude of each frequency present in the signal. In Figure 1, which shows the sampling process of an audio signal, the signal is shown in the time domain, which shows how its amplitude changes over time. Similarly, we can view the amplitudes of all the component frequencies of a signal at a specific point in time (Figure 2), and this represents frequency domain or the spectrum [10].

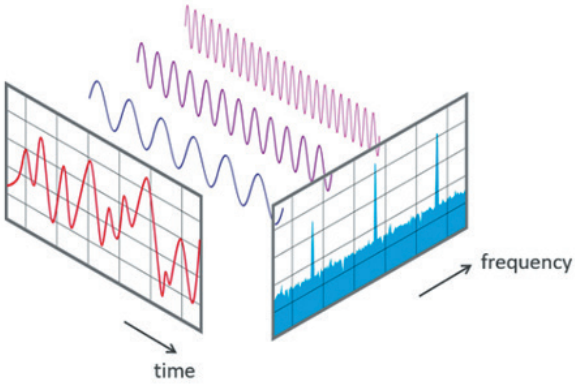

Figure 2 – The time and frequency domains of a signal.

Because a signal can produce different sounds over time, its component frequencies and, therefore, its spectrum can also change over time.

The visual representation of how the spectrum of a signal changes over time is called a spectrogram (Figure 3).

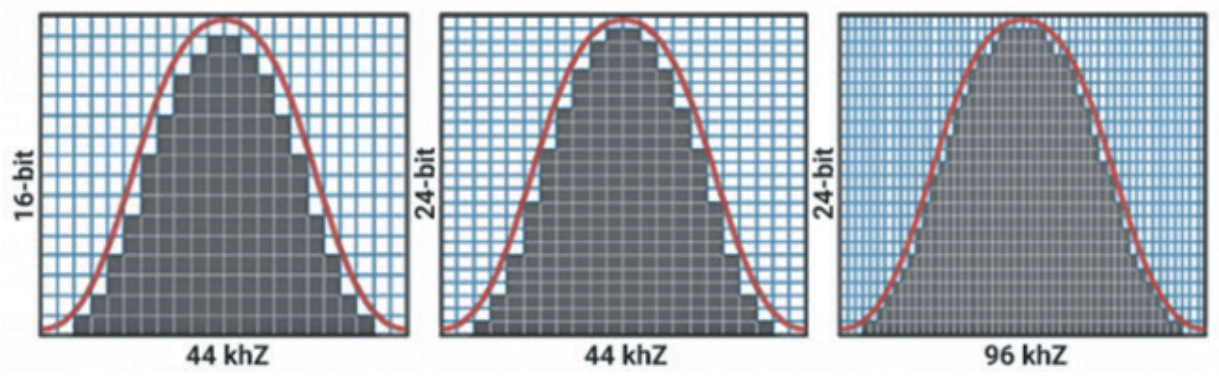

Figure 1 – Comparison of different sampling rates and bit depths.

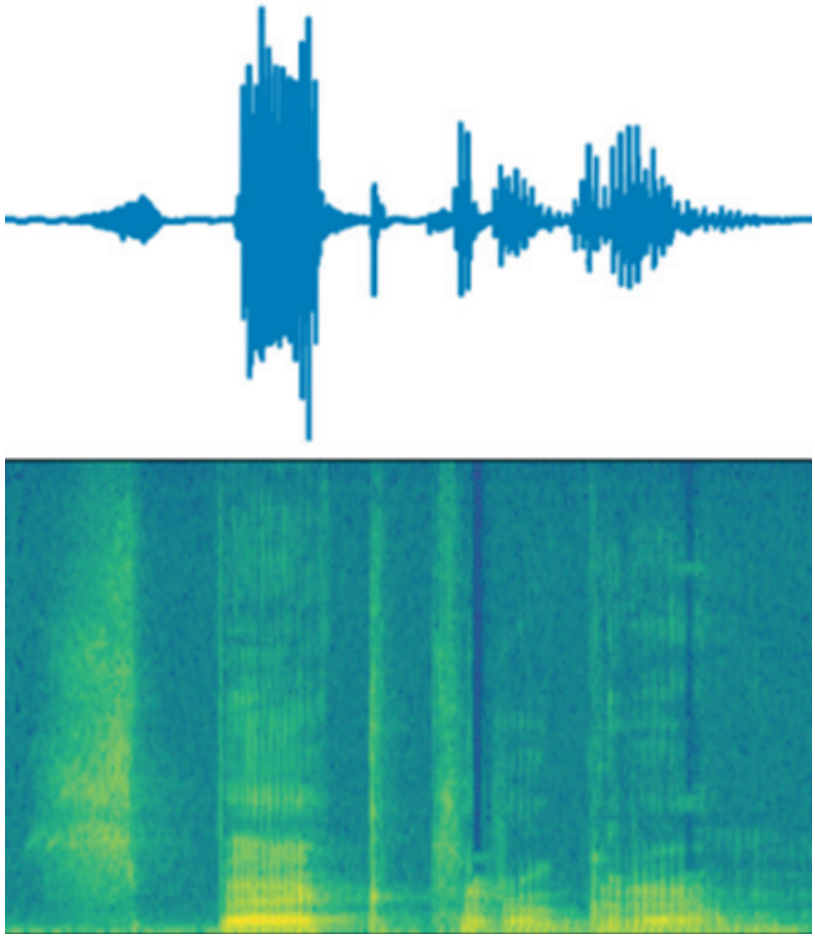

Figure 3 – The waveform and spectrogram of a signal.

The brightness of a color on a spectrogram indicates the amplitude or intensity of a frequency in a given moment in time, and every segment of the graph is a representation of the frequency domain and energy distribution of the signal in that point in time. Each segment of the graph shows the state of the spectrum, i.e. frequency domain and signal energy distribution at that time. A spectrogram can be considered of as an "image" of an audio signal. The spectrogram shows characteristics and features that are unique to this complex signal, allowing the model to classify the signal into the appropriate category.

4. TRAINING CLASSIFICATION MODEL

This section describes the process of creating a general audio classifier, a model which has the ability to differentiate a finite number of sound classes (Figure 4):

1. Loading the audio files in the correct format;
2. Creating spectrograms based on the audio files;
3. Audio or spectrogram augmentation;
4. Creating a feature map for a given spectrogram in a convolutional network;
5. Predicting the score for all supported classes in the model for the given signal based on the extracted feature map.

Firstly, we need to load the audio data from a file and convert it to the appropriate format. This is usually a single-channel audio file (mono) with a sampling frequency of 16kHz and a bit depth of 16 bits, encoded by pulse code modulation (PCM), or in other words, an uncompressed bitstream format file. A sampling frequency higher than 16kHz is not necessary due to the fact that features can easily be discerned and extracted from the spectrograms of lower frequency samples. Comparing two spectrograms of 16 and 48 kHz, we can conclude that the brighter sections of the spectrograms (the ones that often represent features) remain unchanged regardless of the sample rate (Figure 5).

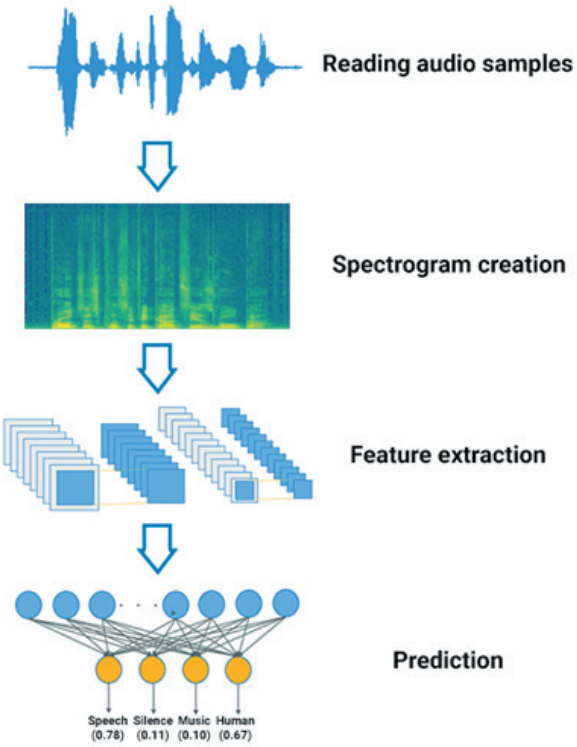

Figure 4 – Steps of model training.

To ensure the resulting classification model is accurate in its predictions, the training dataset must be large and diverse. Differences in the length of pauses, silence or noise can have a significant and often negative impact on the classification accuracy. As a result, training data are usually processed to represent the unique characteristics of the sound signal they contain better, using techniques like noise reduction, silence trimming, etc [11]. This is done to present the key sound features as accurately as possible to the neural network and to define them as clearly as possible for each category (class) that the model supports. This step can be performed before or after the conversion of the samples into spectrograms. Although it is not necessary, doing this step is recommended as excess noise or silence can easily hinder the accuracy of the classification model.

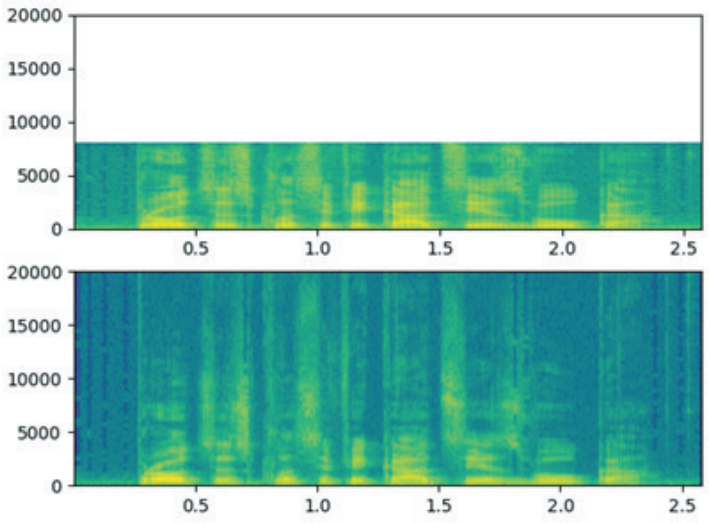

Figure 5 – Comparison of spectrograms of audio signals with a sample rate of 16kHz (upper) and 48kHz (lower).

After converting and loading the training data, a spectrogram is constructed usually by using Fourier transforms which can be performed automatically by mathematical libraries [12]. As the model classifies sound based on its spectrogram, the classification process is very similar to image classification. As previously mentioned, spectrograms can represent the "image" of an audio signal, and in the same way specific arrangements of pixels contain information unique to a particular object in a photograph, a collection of pixels in the image of a spectrogram can also represent features of an audio signal, and because of this, very similar methods are employed when implementing image and sound classification models. The most used neural network architecture for image classification is the convolutional neural network [2], [3], [13]. Therefore, this architecture has also been used in sound classification models.

The underlying methods of training these neural networks differ from library to library, but the core principles remain the same regardless of implementation. The neural network takes a tensor of a specific rank and dimension with the data that comprises the spectrogram of the audio signal, and these tensors are passed through the convolutional layers of the network, by which another tensor, which contains the activation or feature map of the given signal, is created. The information contained within this tensor is processed by specific neurons depending on the contents of the feature map and the receptive fields of the given neurons [14]. One of the last layers in the neural network is a fully-connected layer or linear classifier, and this is where classification actually happens [15]. In other words, this is the layer that outputs the prediction scores for each supported category of the model for the input audio. After which, optionally, the prediction scores can be passed through a loss function such as softmax [16], which is similar to the sigmoid function, and it indicates the “cost” of wrong predictions – this information is used for improving the accuracy of the model. Finally, the model outputs an array of prediction scores for all classes for the given input signal.

5. A PRACTICAL EXAMPLE

The example application has the ability to classify seven different classes of sound, and it was made using the TensorFlow library for Python. When creating a classification model, the first step is to collect training data. The dataset should, ideally, be large and diverse, and the training samples should represent the ideal version of the sound in question, e.g. a small amount of noise, adequate length, etc. The samples for our model have been compiled from YouTube and Kaggle, which are sites where users can download many predefined datasets for training purposes, besides other available functions.

The model that this application uses is able to differentiate the following sounds sources: car, truck, cat, dog, human speech, crowd speech (conversation, multiple people speaking) and silence. The dataset used is quite small, and its contents are the following:

- Car - 104 samples
- Cat - 164 samples
- Crowd - 105 samples
- Dog - 113 samples
- Human - 206 samples
- Silence - 71 samples
- Truck - 107 samples

The total number of samples for this model is 870, which means that the model will not be the most accurate. Typically, when training a classification model, thousands of unique samples are used as training data. For example, Google’s YAMNet model for sound classification is trained on well over 2 million individual audio samples, so our model is somewhat modest in regards to the number of samples, but it is enough for demonstration purposes. These samples must be converted into the correct format: a .wav file with a sample rate of 16kHz and a bit depth of 16 bits. Datasets from Kaggle usually come in this format, but in the case that some data needs to be converted anyway, tools like ffmpeg can be used to easily batch convert multiple files at once.

The model training follows the procedure described in the previous section. During the model training, it is important to distinguish three different types of datasets: training dataset, validation dataset and testing dataset. The training dataset is the dataset which is used exclusively for training the neural network – the model’s weights and biases are adjusted based on the contents of this dataset [17]. The validation dataset is used for recognizing and correcting mistakes in predictions during training, and it typically consists of randomly extracted samples from the training dataset – these samples are not used for training the model [17], [18].

The number of borrowed samples may vary depending on the size of the training dataset, but the validation dataset usually takes a 20% of the samples from the training datase.

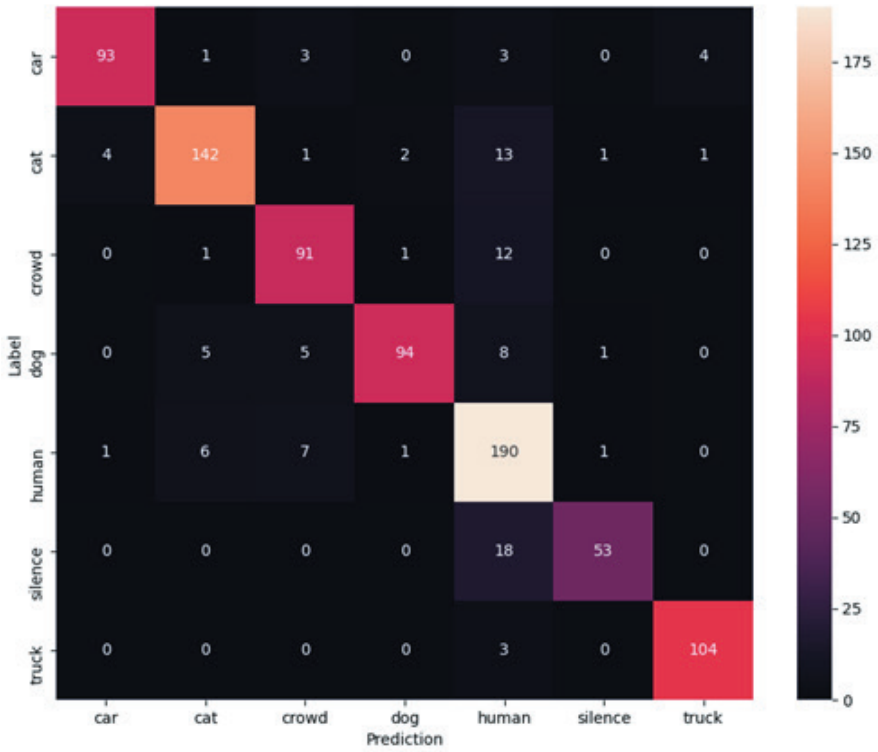

Figure 6 – The confusion matrix for the predictions made during training.

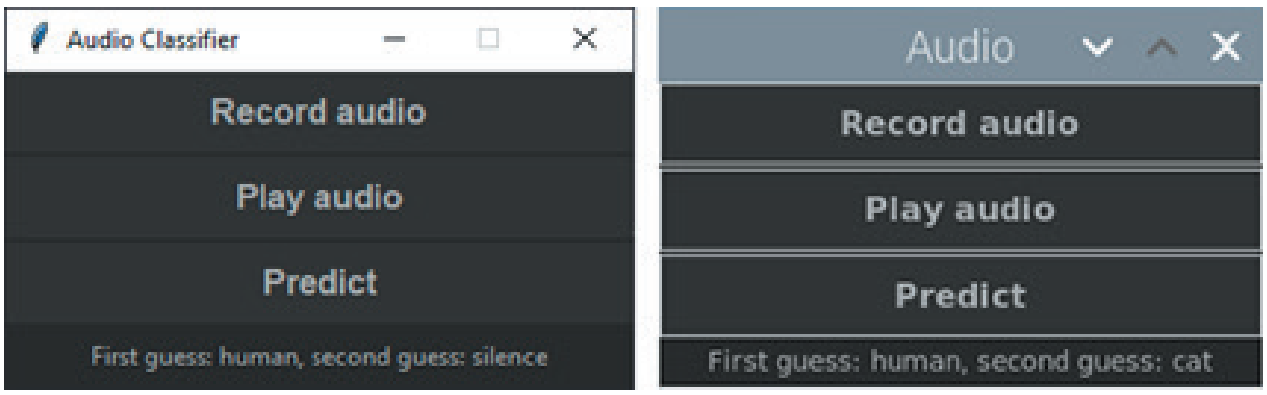

Figure 7 – The GUI applications on Windows 10 (left) and Raspberry Pi OS (right).

Finally, the test dataset is usually a completely separate dataset which is used for the evaluation of the final model after training ends, and it does not contain samples from either of the two previously mentioned datasets [17].

After training the model, we can display a confusion matrix which indicates the model's accuracy when predicting samples from the validation dataset (Figure 6):

The user interface of the application was made using the cross-platform Tkinter library for Python. In the application, the user can record audio from their microphone, play the recorded sound and call the model to output predictions for the recorded audio (Figure 7).

After recording some audio, clicking on the "Predict" button the model is called, and the classification of the recorded clip is performed and the first two classes with the highest prediction scores will be displayed. The application is designed to run on a Raspberry Pi OS, and has been tested on a Raspberry Pi 4 device, where, on average, it uses between 150 and 200 MB of RAM, though with extended use, that number sometimes increases up to 300 MB.

6. CONCLUSION

Sound classification models are currently most often used for speech and music recognition, but they could also be used in, for example, IP cameras. Many IP cameras today can automatically detect human figures or silhouettes and execute specific functions depending on what they "see", e.g., activating alarm or lighting systems, and this is powered by image classification models. Similarly, sound classification models may be used for activating specific functions depending on what the microphone "hears" but the camera can't immediately "see" or detect glass breaking or a loud noise above a certain threshold. Sound classification also has the potential to be employed in natural environments, such as detecting illegal deforestation operations [17] or tracking the activity of difficult-to-see wildlife like insects in their natural habitats [18].

The basic steps outlined in this paper can be used to implement a sound classification model using most of the popular machine learning libraries such as PyTorch and TensorFlow. They are easily configurable and the models can be adapted for a relatively wide range of applications. Options such as the number and order of layers in the neural network, tensor rank and dimensions, loss functions and the datasets themselves all have an effect on the prediction accuracy and quality of the final model. The process described in this paper, however, is mostly used for training a general-purpose audio classification model which can classify a few different types of audio. When implementing a model for a different use case, such as speech recognition, the architecture of the neural network may be different, but the core principles of digital audio representation and discerning the inherent attributes of an audio signal through feature extraction rarely ever change when implementing a sound classification model.

7. REFERENCES

[1] D. Gerhard, "Audio Signal Classification: History and Current Techniques," 2003.

[2] C. Senac, T. Pellegrini, F. Mouret and J. Pinquier, "Music Feature Maps with Convolutional Neural Networks for Music Genre Classification," in *CBMI '17: Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing*, 2017.

[3] T. Nakashika, C. Garcia and T. Takiguchi, "Local-feature-map Integration Using Convolutional Neural Networks for Music Genre Classification," in *INTERSPEECH 2012 ISCA's 13th Annual Conference*, Portland, 2012.

[4] T. Robinson, M. Hochberg and S. Renals, "The Use of Recurrent Neural Networks in Continuous Speech Recognition," in *Automatic Speech and Speaker Recognition. The Kluwer International Series in Engineering and Computer Science (VLSI, Computer Architecture and Digital Signal Processing)*, vol. 355, Springer, 1996, pp. 233-258.

[5] "Acoustics and electroacoustics," [Online]. Available: https://electropedia.org/iev/iev.nsf/index?openform&part=801. [Accessed 25 March 2022].

[6] M. Weik, Communications Standard Dictionary, Springer Science & Business Media, 1995.

[7] "Understanding DVD-Audio," [Online]. Available: http://patches.sonic.com/pdf/white-papers/wp_dvd_audio.pdf. [Accessed 18 March 2022].

[8] J. Watkinson, The Art of Digital Audio, Taylor&Francis, 2001.

[9] D. Morin, "Fourier analysis," 2009.

[10] J. C. Santamarina and D. Fratta, Discrete Signals and Inverse Problems: An Introduction for Engineers and Scientists, John Wiley&Sons.

[11] "Audio Data Preparation and Augmentation," TensorFlow, [Online]. Available: https://tensorflow.org/io/tutorials/audio. [Accessed 18 March 2022].

[12] J. O. Smith III, Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications, Second Edition, 2007.

[13] N. I. Chervyakov, P. A. Lyakhov, M. A. Deryabin, N. N. Nagornov, M. V. Valueva and G. V. Valuev, "Residue Number System-Based Solution for Reducing the Hardware Cost of a Convolutional Neural Network," *Neurocomputing*, vol. 407, pp. 439-453, 2020.

[14] W. Luo, Y. Li, R. Urtasun and R. Zemel, "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks," in *30th Conference on Neural Information Processing Systems* (*NI*, Barcelona, 2016.

[15] "Convolutional Neural Networks (CNNs / ConvNets)," [Online]. Available: https://cs231n.github.io/convolutional-networks/. [Accessed 18 March 2022].

[16] "Linear Classification," [Online]. Available: https://cs231n.github.io/linear-classify/. [Accessed 18 March 2022].

[17] T. White, "The fight against illegal deforestation with TensorFlow," 21 March 2018. [Online]. Available: https://blog.google/technology/ai/fight-against-illegal-deforestation-tensorflow/. [Accessed 18 March 2022].

[18] V. Poblete, D. Espejo, V. Vargas, F. Otondo and P. Huijse, "Characterization of Sonic Events Present in Natural-Urban Hybrid Habitats Using UMAP and SEDnet: The Case of the Urban Wetlands," *Appl. Sci.*, vol. 11, no. 17, 2021.