VOL. 21, NO. 1, 2022, 48-53
www.fke.utm.my/elektrika
ISSN 0128-4428

ELEKTRIKA
Journal of Electrical Engineering

Bird Sound Detection with Binarized Neural Networks

Muhammad Mun'im Ahmad Zabidi[*], **Wong Kah Liang**, **Usman Ullah Sheikh**, **Shahidatul Sadiah** and **Muhammad Afiq Nurudin**

School of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.

[*]Corresponding author: munim@utm.my

Abstract: By analysing the behavioural patterns of bird species in a specific region, researchers can predict future changes in the ecosystem. Many birds can be identified by their sounds, and autonomous recording units (ARUs) can capture real-time bird vocalisations. The recordings are analysed to see if there are any bird sounds. The sound of a bird can be used for further analysis, such as determining its species. Bird sound detection using Deep Neural Networks (DNNs) has been shown to outperform traditional methods. DNNs, however, necessitate a lot of storage and processing power. The use of Binarized Neural Networks (BNNs) is one of the most recent approaches to overcoming this limitation. In this paper, a bird sound detection architecture based on the XNOR-Net variant of BNN is used. Performance analysis of XNOR-Net in terms of the number of hidden layers used was performed, and the configuration with the highest accuracy was built. The system was tested using Xeno-Canto and UrbanSound8K datasets to represent bird and non-bird sounds, respectively. We achieved 96.06 per cent training accuracy and 94.08 per cent validation accuracy. We believe that BNNs are an effective method for detecting bird sounds.

Keywords: binarized neural networks, bioacoustics, bird sound detection, convolutional neural networks, deep learning

Article History: received 8 February 2022; accepted 1 April 2022; published 20 April 2022.

1. INTRODUCTION

Researchers can predict future changes in the environment, ecology, and population by analysing the behavioural patterns of bird species in a specific area [1]. Many birds can be identified by their sounds, and autonomous recording devices (ARUs) can capture bird vocalisations in real-time [2].

The vast amounts of data generated by ARUs have outstripped humans' ability to manually interpret it. As a result, a variety of automatic methods of bird detection and classification have emerged. The use of bioacoustics and machine learning now allows for the estimation of population densities and the location of the target species [3]. Bird detection and classification algorithms are proposed and evaluated in regular challenges such as BirdCLEF and DCASE.

Deep neural networks (DNNs) have consistently outperformed traditional methods for bird detection since 2016 [4,5]. However, DNNs require lots of storage and processing power. DNNs are especially difficult to use in low-cost embedded devices (e.g., Raspberry Pi) due to added power constraints. Binarized Neural Networks (BNNs) are one approach to addressing this deficiency. These are neural networks that use binary weights and activations rather than floating-point numbers. In this paper, we investigate the use of BNNs for the task of detecting bird sounds.

2. BACKGROUND

2.1 Bird Sound Detection

Sound is a vibration that travels through the air and conveys information such as musicality, audibility or inaudibility, softness, or loudness, and so on. Sound has a diverse frequency content and temporal structure, resulting in a wide range of features. Acoustic bird detection and classification algorithms share many similarities with audio event detection (AED) and speech recognition algorithms. However, due to the variability of bird sounds according to species, sex, age, and season, detecting whether a bird sound is present in a sound recording presents unique challenges.

A common method for pre-processing bird sounds before any kind of analysis is to apply the Short-Time Fourier Transform (STFT) to produce spectrograms, which are visual representations of sounds. Spectrograms can be used for feature extraction before applying traditional machine learning algorithms such as random forests [6] and support vector machines [7]. Bird detection can be improved by applying morphological filtering on the spectrograms [8] or by converting it to Mel-spectrograms or Mel-frequency cepstral coefficients (MFCC) [9]. For DNNs, spectrogram generation becomes an essential step as deep learning requires images or image-like inputs. Figure 1 shows the call of the Asian koel ((*Eudynamys scolopaceus*) in various forms.

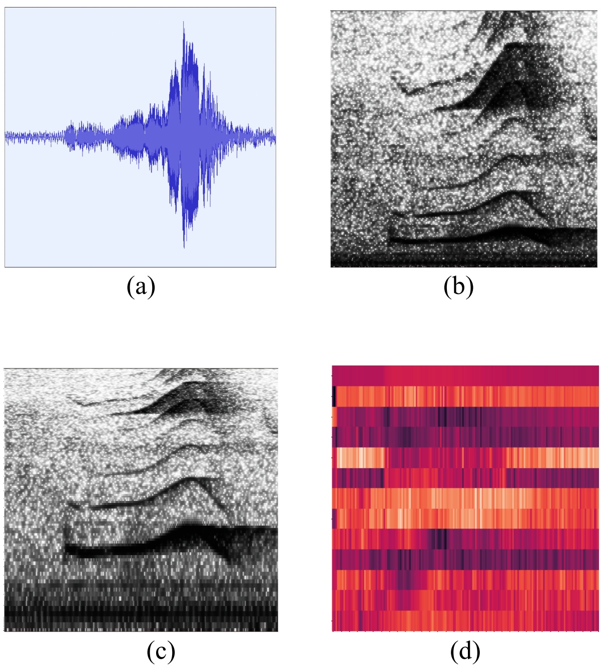


Figure 1. Various representations of the sound of the Asian koel (*Eudynamys scolopaceus*): (a) time series (b) STFT spectrogram, (c) Mel spectrogram, and (d) Mel-frequency cepstral coefficients (MFCC)

2.2 Convolutional Neural Networks

Many new intelligent applications rely on DNNs especially in the form of convolutional neural networks (CNNs) to achieve high accuracy in image recognition tasks [10]. CNNs detect bird acoustic events after sounds have been converted into spectrograms.

A basic artificial neural network (ANN) consists of three main layer types: input layer, hidden layers, and output layer. Each node inside the three layers is known as a neuron. The operation of each neuron is the summation of input value multiplied with weight *w*. This operation is also known as multiply-accumulate (MAC). There is a bias offset *b* which serves the purpose of ensuring that there will be activation in the neuron even when all the input values are zero. The output is then computed using the sigmoid or hyperbolic tangent (TanH) activation function. When the number of layers exceeds one, the ANN is classified as a DNN. When the hidden layers use convolutions, the DNN transforms into a CNN.

The performance of the CNNs comes at the expense of high memory and energy needs. This has hindered the use of CNNs on embedded and mobile devices with limited storage capacity and processing power. This has motivated TinyML, a field of study that explores machine learning models that can run on small, low-powered devices.

Two small CNN architectures were proposed by [11] for AED. Both models were fed with a large input field to model entire audio events end-to-end. Architecture A consists of 4 convolutional and 3 fully connected layers and architecture B was made up of 6 convolutional and 3 fully connected layers. All the convolutional layers are made up of 3·3 kernels. Taking the cue from automatic speech recognition (ASR), the input sub-word unit's length is less than a few hundred milliseconds. By modelling the sub-word units in several seconds long signal, this gives the CNN an accuracy of 92.8%.

To improve the accuracy of the CNN model in detecting avian flight calls of nocturnal bird migration from a ten-hour recording, [12] proposed two noise adaptive techniques that integrate short-term (60 milliseconds) and long-term (30 minutes) context. The first technique uses the per-channel energy normalization (PCEN) model in the time-frequency domain. This model applies short-term automatic gain control to every sub-band in the Mel-frequency spectrogram. The second technique is to replace the last dense layer in the network with a context-adaptive neural network (CA-NN) layer. This architecture consists of two branches: the main branch and the auxiliary branch. The architecture of the main branch originated from the state-of-the-art urban sound classification using Urban8K datasets [13] and the species classification from clips of avian flight calls using CLO-43SD dataset [14].

2.3 Binarized Neural Networks

Besides making CNNs shallower, other methods to reduce computational complexity while maintaining acceptable accuracy have been proposed. To reduce the number of computations, most methods employ network pruning and overparameterization. To reduce the number of weights, channel-wise separable convolutions were proposed [15]. Weight sharing and low rank were also proposed [16]. Squeezenet [17] uses network topologies that are specifically designed to reduce the number of parameters.

In 2015, Courbariaux *et al.* proposed BinaryConnect, a method to train deep neural networks using binary weights [18]. Only the weights are binarized in this approach, and inputs can remain non-binarized (real). The first layer is usually not completely binarized. The weights are binary, but the inputs are real. The outputs are also real, but they are converted to binary after activation before being used by other layers. The first layer can also be made fully binary by stochastic computing [19]. Additionally, the thresholding is not performed on the last layer. Instead, the output neuron with the maximum pop count value gives the output of the neural network. Courbariaux *et al.* expanded their work and in 2016, proposed Binary Neural Networks (BNNs) [20]. BNNs use binary values (+1 or -1) for both weights and activation during inference and backpropagation training. This drastically reduces the hardware requirements to operate the networks. Multiplication is replaced with XNOR gates, eliminating hardware multipliers. Accumulation, normally done by adders, is replaced by simple bit counting (pop count). Finally, the activation function is a simple thresholding or sign function.

Two different binarization functions are used to convert real values to binary [15]. The first function is deterministic:

$$x^b = \begin{cases} +1, & if\ x \geq 0, \\ -1 & otherwise. \end{cases} \tag{1}$$

where x is the real-valued variable and x^{b} is the binarized variable (weight or activation). The second binarization function is stochastic:

$$x^{b} = \begin{cases} +1, & with\,probability\ \ p = \sigma(x), \\ -1 & with\,probability\ 1 - p. \end{cases} \quad (2)$$

where σ is the "hard sigmoid" function:

$$\begin{aligned} \sigma(x) &= \text{clip}\left(\frac{x+1}{2}, 0{,}1\right) \\ &= \max\left(0, \min\left(1, \frac{x+1}{2}\right)\right) \end{aligned} \quad (3)$$

The deterministic binarization function is more commonly used in practice as it is easier to implement.

Rastegari *et al.* proposed two variants of the BNN: Binary Weight Networks and XNOR-Networks [21]. In a standard Convolutional Neural Network, both the weights and the inputs to the convolutional layers are real values (Figure 2 (a)). For Binary Weight Networks, binarized values were used in the weights of the neural network, shown in the weights as ±1 (Figure 2 (b)). The inputs are still real values. In the XNOR-net, both the inputs and the weights are binarized (Figure 2 (c)). With binary weights, memory usage is reduced by approximately 32x compared to the single-precision filters. Using binarized weights, multiplications can be replaced by additions and subtractions. As a result, this gives 2x improvement in a computation speed up. In the XNOR-Network, both weights and the inputs to the convolutional network are binary. This produces 58x speedup and 32x memory savings.

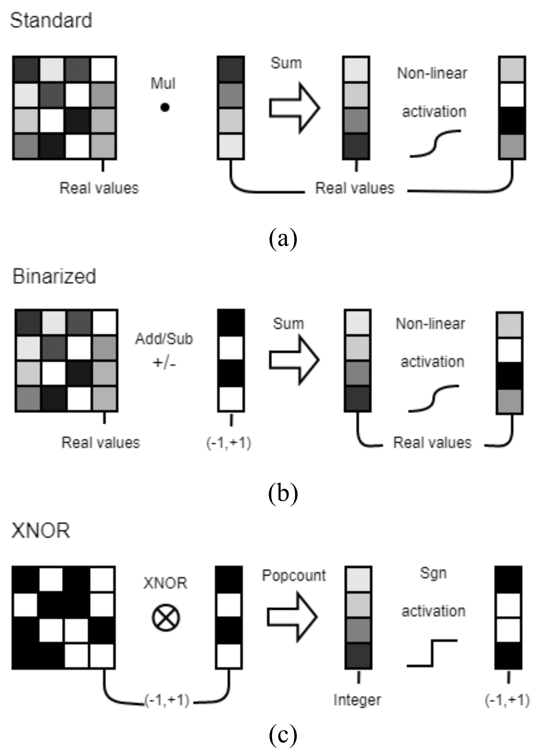


Figure 2. Comparison of (a) standard and (b-c) binarized neural networks.

The Binarized Convolutional Neural (BCNN) was proposed as a bird sound detector by Song and Li [22]. Based on the architecture used by the winner of the DCASE 2017 bird audio detection challenge, the authors proposed two networks (CNNs and BCNNs) [5]. It was tested on the DCASE 2018 dataset and it has 4 convolutional layers with 5·5 filters and 2 fully connected layers. In the BCNN network, the Batch Normalization layer was used to normalize the data and binarize the activation layer before convolution operation. This prevents the activation layer's result from becoming a single value after pooling. The activation function of the BCNN uses the Sign(x) function with a gradient of zero. To void having the gradient disappear during backpropagation, Htanh(x) function was used as the activation function.

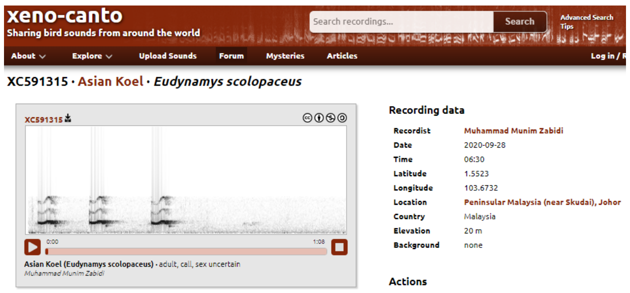
3. EXPERIMENTS AND RESULTS

3.1 Dataset Preparation

The preparation of datasets starts by downloading data files Xeno-Canto and Urban8k databases for positive and negative datasets, respectively [13, 23]. A web page for a sound file on Xeno-Canto contains metadata for the sound as well as spectrogram of the sound as shown in Figure 3. Multiple accesses were required to fetch the required data. The Urban8K dataset, on the other hand, is a single 6GB download that contains all files and a CSV containing metadata. The data files are pre-processed using MATLAB code to obtain 1000 files for each class. The complete dataset preparation steps are depicted in Figure 4.

Due to the varying durations of the audio recordings, the techniques for positive and negative datasets are different. For positive datasets, most of the audio files downloaded from Xeno-canto are long, in which there are time frames where birds are detected and there are quiet passages. The audio wave's amplitude is first normalised between -1 and +1. The audio files are segmented into many 1-second timeframes without overlapping. Then, five clips with an absolute amplitude of at least 0.5 were selected.

The audio files from Urban8k are not always 1-second long. Prior to performing audio segmentation, files shorter than two seconds are rejected. Then audio segmentation is performed on the remaining audio files. Hence, 1000 files for each class were obtained.



XC591315 · Asian Koel · *Eudynamys scolopaceus*

Asian Koel (Eudynamys scolopaceus) · adult, call, sex uncertain
Muhammad Munim Zabidi

Recording data

Recordist	Muhammad Munim Zabidi
Date	2020-09-28
Time	06:30
Latitude	1.5523
Longitude	103.6732
Location	Peninsular Malaysia (near Skudai), Johor
Country	Malaysia
Elevation	20 m
Background	none

Figure 3. A sound file entry from the Xeno-Canto repository.

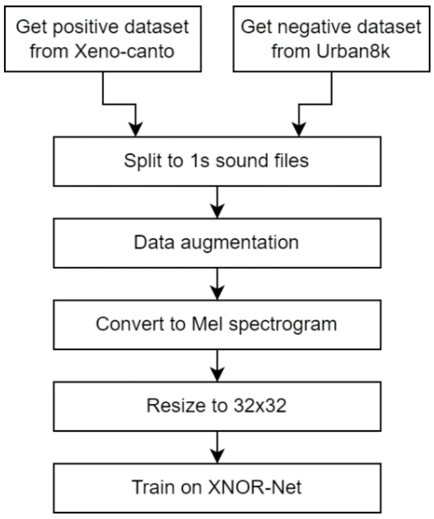

Get positive dataset from Xeno-canto
Get negative dataset from Urban8k
Split to 1s sound files
Data augmentation
Convert to Mel spectrogram
Resize to 32x32
Train on XNOR-Net

Figure 4. Stages in dataset preparation.

The audio files are then converted to the frequency domain by 2048-point Fast Fourier Transform with 25% overlap. These outputs were then converted into Mel-Spectrogram using Librosa library coded in Python [25]. Even though MFCC was the preferred method of audio feature extraction for traditional machine learning algorithms, we opted to use Mel-spectrogram as it requires fewer computations and works just as well as MFCC for deep learning algorithms [26].

Next, the Mel-spectrograms were converted into binary using OpenCV image binary thresholding library and were reshaped to 32·32 images before being fed to the BNN for training. The size of 32·32 was chosen as it produces sufficiently accurate results within the limitations of the Google Colab platform. Figure 5 shows the effect of binarization on Mel-spectrograms.

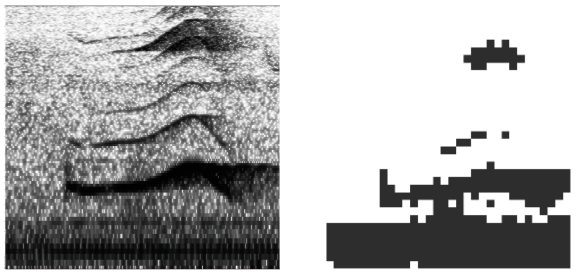


Figure 5. The result of applying binarization: (a) source Mel-spectrogram image, and (b) 32·32 binary image

3.2 Proposed BNN Model

Our proposed architecture is based on XNOR-Net as shown in Table 1. The Binary Activation layer and Binary Convolution layer are wrapped as a layer named XNORnetConv. Other details about each layer are provided below:

- *Convolutional 2D layer* — The input layer produces a tensor of receptive fields
- *Batch Normalization layer* — Optimizes the activation of the previous layer to mean activation close to 0 and the standard activation close to 1.
- *Binary Activation layer* — Uses Sign activation from TensorFlow to return -1 if $x < 0$, 1 if $x \geq 0$.
- *The Max-Pooling layer* — Down-samples the input representation by taking the maximum value over the window defined by the pool_size for each dimension along the features axis.
- *Flatten layer* — Transform 2D matrix of features into a vector that can be fed into a fully connected neural network classifier.
- *Dropout layer* — Randomly set the input units to 0 with a frequency of rate at each step during training time to prevent over-fitting.
- *Dense layer* — Fully connected the output layer.

The BNN architecture is modelled with Keras/TensorFlow for training and validation, running on the Colab platform. The initial set of training parameters for one binary convolutional layer XNOR-Net is as shown in Table 2.

Table 1. XNOR-Net parameters

Layer type	Output shape	Parameter
Input layer	32·32	0
Conv2D	30·30	448
Batch normalization	30·30	64
XNORnetConv	28·28	4617
Max Pooling	14·14	0
Flatten	6272	0
Dropout	6272	0
Dense	2	12546
Total trainable parameters	17,643	

Table 2. XNOR-Net initial hyperparameters

Parameter	Value
Binary Activation	Sign Function
Epoch	30
Batch Size	32
Learning Rate	0.0003
Decay	0.001

3.3 Impact of Network Depth and Activation Function

The output of XNOR-Net varies with different numbers of binary convolutional layers. The impact of varying the number of layers is shown in Table 3.

To improve training accuracy, an additional activation layer was introduced after Batch Normalization and before the Binary Convolutional Layer. Table 4 shows that the addition of the activation layer between the output of Batch Normalization and at the input of Binary Convolutional Layer does improve the training accuracy. By adding only one TanH layer, the training and validation accuracy is the highest at 98.27% and 95.56% with a validation loss of 12.01%. By adding a ReLU activation at the input of the TanH activation layer, the training accuracy is reduced, but the difference between training and validation accuracies has narrowed and hence giving the network a better generalization performance. Hence, the TanH activation function was chosen for the final model.

Table 3. Performance of XNOR-Net with different numbers of Binary Convolutional Layers

	Number of hidden binary convolutional layers		
	1	2	3
Trainable Params	17,643	16,628	23,805
Training Accuracy (%)	94	93	83
Validation Accuracy (%)	95	96	95
Training Loss (%)	14	19	36
Validation Loss (%)	2	1	19
Time (sec)	75.19	84.54	86.38
Memory resources (MByte)	234.33	243.20	248.55

Table 4. Performance of XNOR-Net with different types of activation functions

Activation Layer	None	ReLU	TanH	ReLU+ TanH
Training Accuracy (%)	94.39	96.49	98.27	96.49
Validation Accuracy (%)	95.06	94.32	95.56	93.58
Training Loss (%)	14.40	9.59	5.60	9.94
Validation Loss (%)	2.29	23.84	12.01	19.43
Time (sec)	75.19	85.46	88.94	86.49
Memory (MByte)	234.33	236.56	236.33	236.02

Table 5. Hyperparameters for the best model.

Parameter	Value
Batch size	32
Learning Rate	0.0003
Validation dataset (%)	35
Decay	0.001
Training Accuracy (%)	96.06
Validation Accuracy (%)	94.08
Training Loss (%)	12.07
Validation Loss (%)	1.01
Time (sec)	247.1
Memory resources (MByte)	234.34

3.4 Optimization of XNOR-Net Hyperparameters

After determining the model structure, hyperparameters such as batch size, learning rate, decay rate, and validation datasets can be changed to optimize the model. To have more data to be analyzed, 100 epochs was set for this run. Table 5 shows the most suitable hyperparameters to employ this variant to train 32·32 binary images. The training and validation results are shown in Table 6.

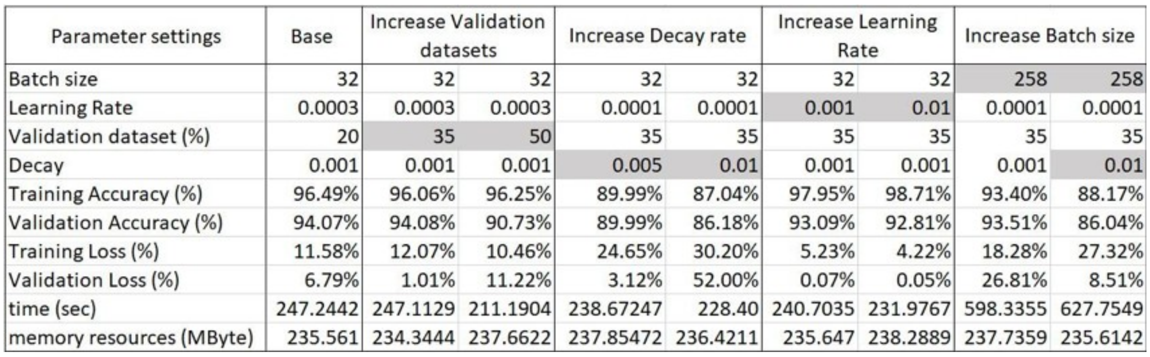
4. CONCLUSION

When compared to typical convolutional neural networks, the XNOR-Net binarized neural network allows classification tasks to be completed with significantly fewer logic operations. In this paper, a variant the XNOR-Net binarized neural network was used in the bird acoustic event detection task. For the positive and negative datasets, audio files from the Xeno-Canto and UrbanSound8K databases were employed, respectively. The audio files were segmented into 1-second clips, which were then converted into to 32·32 binarized Mel-Spectrograms for use by the BNN.

Following testing, the XNOR-Net model with 7 layers produced the best results. The hyperparameters were optimized to achieve 96.06 percent training accuracy and 94.08 percent validation accuracy. Because the XNOR-net only has two output classes, binary images with a size of 32·32 pixels are sufficient. When there are more output classes, the input size may need to be raised for the network to generalize more effectively.

The datasets utilized in this model are considered small because the dropout rate must be adjusted to 50%. To improve generalization, the size of the positive and negative datasets could be increased. One way to enlarge the training datasets is by data augmentation techniques such as SpecAugment [26] which includes warping the features, masking blocks of frequency channels, masking blocks of the time step, and adding noise are used. The work could also be advanced by transferring the BNNs to embedded platforms such as microcontrollers and field-programmable gate arrays (FPGAs).

Table 6. Training and validation of XNOR-Net by varying the model parameters.



Parameter settings	Base	Increase Validation datasets		Increase Decay rate		Increase Learning Rate		Increase Batch size	
Batch size	32	32	32	32	32	32	32	258	258
Learning Rate	0.0003	0.0003	0.0003	0.0001	0.0001	0.001	0.01	0.0001	0.0001
Validation dataset (%)	20	35	50	35	35	35	35	35	35
Decay	0.001	0.001	0.001	0.005	0.01	0.001	0.001	0.001	0.01
Training Accuracy (%)	96.49%	96.06%	96.25%	89.99%	87.04%	97.95%	98.71%	93.40%	88.17%
Validation Accuracy (%)	94.07%	94.08%	90.73%	89.99%	86.18%	93.09%	92.81%	93.51%	86.04%
Training Loss (%)	11.58%	12.07%	10.46%	24.65%	30.20%	5.23%	4.22%	18.28%	27.32%
Validation Loss (%)	6.79%	1.01%	11.22%	3.12%	52.00%	0.07%	0.05%	26.81%	8.51%
time (sec)	247.2442	247.1129	211.1904	238.67247	228.40	240.7035	231.9767	598.3355	627.7549
memory resources (MByte)	235.561	234.3444	237.6622	237.85472	236.4211	235.647	238.2889	237.7359	235.6142

REFERENCES

[1] D. Hayhow, F. Burns, M. Eaton, N. Al Fulaij, T. August, L. Babey, L. Bacon, C. Bingham, J. Boswell, K. Boughey, et al., "State of nature 2016", The State of Nature partnership (2016).

[2] J. Shonfield and E. Bayne, "Autonomous recording units in avian ecological research: current use and future applications", Avian Conservation and Ecology 12 (2017).

[3] T. A. Rhinehart, L. M. Chronister, T. Devlin, and J. Kitzes, "Acoustic localization of terrestrial wildlife: Current practices and future opportunities", Ecology and Evolution 10, 6794-6818 (2020).

[4] H. Goeau, H. Glotin, W.-P. Vellinga, R. Planque, and A. Joly, "LifeCLEF bird identification task 2016: The arrival of deep learning", in CLEF: Conference and Labs of the Evaluation Forum, 1609 (2016) pp. 440-449.

[5] D. Stowell, M. D. Wood, H. Pamula, Y. Stylianou, and H. Glotin, "Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge", Methods in Ecology and Evolution 10, 368-380 (2019).

[6] C. Bravo, R. Berríos, and T. Aide. "Species-specific audio detection: a comparison of three template-based detection algorithms using random forests", PeerJ Computer Science 3 (2017): e113.

[7] S. Fagerlund. "Bird species recognition using support vector machines", EURASIP Journal on Advances in Signal Processing (2007).

[8] G. de Oliveira, T. M. Ventura, T. D. Ganchev, J. M. de Figueiredo, O. Jahn, M. I. Marques, and K.-L. Schuchmann, "Bird acoustic activity detection based on morphological filtering of the spectrogram", Applied Acoustics 98, 34-42 (2015).

[9] D. Steven, and P. Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", IEEE transactions on acoustics, speech, and signal processing 28, no. 4 (1980): 357-366.

[10] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning book", MIT Press 521, 800 (2016).

[11] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection", arXiv preprint arXiv:1604.07160 (2016).

[12] V. Lostanlen, J. Salamon, A. Farnsworth, S. Kelling, and J. P. Bello, "Robust sound event detection in bioacoustic sensor networks", PLoS ONE 14, e0214168 (2019).

[13] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research", in Proceedings of the 22nd ACM international conference on Multimedia (2014) pp. 1041-1044.

[14] J. Salamon, J. P. Bello, A. Farnsworth, M. Robbins, S. Keen, H. Klinck, and S. Kelling, "Towards the automatic classification of avian flight calls for bioacoustic monitoring", PLoS ONE 11, 1-26 (2016).

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications", CoRR arXiv preprint arXiv:1704.04861 (2017).

[16] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions", arXiv preprint arXiv:1405.3866 (2014).

[17] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size", arXiv preprint arXiv:1602.07360 (2016).

[18] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations", in Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2 (2015) pp. 3123-3131.

[19] T. Hirtzlin, B. Penkovsky, M. Bocquet, J.-O. Klein, J.-M. Portal, and D. Querlioz, "Stochastic computing for hardware implementation of binarized neural networks", IEEE Access 7, 76394-76403 (2019).

[20] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1", arXiv preprint arXiv:1602.02830 (2016).

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks", in European Conference on Computer Vision (Springer, 2016) pp. 525-542.

[22] J. Song and S. Li, "Bird sound detection based on binarized convolutional neural networks", in Proceedings of the 6th Conference on Sound and Music Technology (CSMT) (Springer, 2019) pp. 63-71.

[23] W.-P. Vellinga and R. Planque, "The Xeno-canto collection and its relation to sound recognition and classification", in CLEF (Working Notes) (2015).

[24] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in Python", in Proceedings of the 14th Python in Science conference, Vol. 8 (Citeseer, 2015) pp. 18-25.

[25] H. Fayek. "Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between", URL: https://haythamfayek.com/2016/04/21/speech-processingfor-machine-learning. html (2016).

[26] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition", arXiv preprint arXiv:1094.08779 (2019).