Bird@Edge: Bird Species Recognition at the Edge

Jonas Höchst*[1], Hicham Bellafkir*[1], Patrick Lampe[1], Markus Vogelbacher[1], Markus Mühling[1], Daniel Schneider[1], Kim Lindner[2], Sascha Rösner[2], Dana G. Schabo[2], Nina Farwig[2], and Bernd Freisleben[1]

[1] Dept. of Mathematics & Computer Science, University of Marburg, Germany
{hoechst, bellafkir, lampep, vogelbacher, muehling, schneider, freisleb}
@informatik.uni-marburg.de
[2] Dept. of Biology, University of Marburg, Germany
{kim.lindner, sascha.roesner, dana.schabo, nina.farwig}@biologie.uni-marburg.de

Abstract. We present Bird@Edge, an *Edge AI* system for recognizing bird species in audio recordings to support real-time biodiversity monitoring. Bird@Edge is based on embedded edge devices operating in a distributed system to enable efficient, continuous evaluation of soundscapes recorded in forests. Multiple ESP32-based microphones (called Bird@Edge Mics) stream audio to a local Bird@Edge Station, on which bird species recognition is performed. The results of several Bird@Edge Stations are transmitted to a backend cloud for further analysis, e.g., by biodiversity researchers. To recognize bird species in soundscapes, a deep neural network based on the EfficientNet-B3 architecture is trained and optimized for execution on embedded edge devices and deployed on a NVIDIA Jetson Nano board using the DeepStream SDK. Our experiments show that our deep neural network outperforms the state-of-the-art BirdNET neural network on several data sets and achieves a recognition quality of up to 95.2% mean average precision on soundscape recordings in the Marburg Open Forest, a research and teaching forest of the University of Marburg, Germany. Measurements of the power consumption of the Bird@Edge components highlight the real-world applicability of the approach. All software and firmware components of Bird@Edge are available under open source licenses.

Keywords: Bird Species Recognition · Edge Computing · Passive Acoustic Monitoring · Biodiversity

1 Introduction

The continuous loss of biodiversity is particularly evident from the sharp decline of bird populations in recent decades. Birds are important for many ecosystems, since they interconnect habitats, resources, and biological processes, and thus serve as important early warning bioindicators of an ecosystem's health. Thus, changes in bird species in time and space should be detected as early as possible.

*These authors contributed equally.

Traditionally, this is achieved by human experts who walk around a natural habitat to look at birds and listen to bird sounds, identify bird species present in the sounds, and take notes of their occurrence. In recent years, this is often supported by placing microphones in the natural habitats of birds and recording their sounds. The audio data recorded in this way is then evaluated either manually by human experts or by means of automatic analysis methods to recognize bird species in soundscapes. The disadvantages of this approach are: (a) there is a potentially large amount of recorded audio data that can usually only be evaluated after the end of the recording time, and (b) there is an inherent time delay between recording the audio data and delivering the recognition results.

In this paper, we combine Edge Computing and Artificial Intelligence (AI) to present Bird@Edge, an *Edge AI* system for recognizing bird species in audio recordings to support real-time biodiversity monitoring. Bird@Edge is based on embedded edge devices operating in a distributed system to enable efficient, continuous evaluation of soundscapes recorded in forests. Multiple microphones based on ESP32 microcontroller units (called Bird@Edge Mics) stream audio to a local Bird@Edge Station, on which bird species recognition is performed. The recognition results of different Bird@Edge Stations are transmitted to a backend cloud for further analysis, e.g., by biodiversity researchers.

To recognize bird species in soundscapes, a deep neural network based on the EfficientNet-B3 architecture is trained and optimized for execution on embedded edge devices and deployed on a NVIDIA Jetson Nano board using the DeepStream SDK. Our experimental results show that our deep neural network model outperforms the state-of-the-art BirdNET neural network on several data sets and achieves a recognition quality of up to 95.2% mean average precision on soundscape recordings in the Marburg Open Forest, a research and teaching forest of the University of Marburg, Germany. Measurements of the power consumption of a Bird@Edge Station and the Bird@Edge Mics highlight the real-world applicability of the approach. All software and firmware components of Bird@Edge are available under open source licenses[1]. Our contributions are:

- We present a novel Edge AI approach for recognizing bird species in audio recordings; it supports efficient live data transmission and provides high-quality recognition results.
- We propose a deep neural network based on the EfficientNet-B3 architecture optimized for execution on embedded edge devices to identify bird species in soundscapes.
- We evaluate our Edge AI approach in terms of recognition quality, runtime performance, and power consumption.

The paper is organized as follows. Section 2 discusses related work. In Section 3, the design and implementation of Bird@Edge are presented. Section 4 describes our deep neural network for bird species recognition. Section 5 discusses experimental results in terms of recognition quality, runtimes, and power consumption. Section 6 concludes the paper and outlines areas of future work.

[1] `https://github.com/umr-ds/BirdEdge`

2 Related Work

In this section, we discuss related work with respect to current machine learning approaches for bird species recognition in audio recordings and edge AI approaches for biodiversity monitoring.

2.1 Bird Species Recognition

For many years, bird populations were monitored manually by ornithologists who identified birds visually and acoustically on site. The introduction of autonomous recording units (ARU) opened new possibilities. Although such passively recorded data does not provide any visual information, the resulting bird surveys conducted by humans from sound recordings are comparable to traditional monitoring approaches in the field [2].

Furthermore, machine learning methods, such as Convolutional Neural Networks (CNN), are increasingly being used for automatically recognizing bird species in soundscapes. For example, BirdNET is a task-specific CNN architecture trained on a large audio data set using extensive data pre-processing, augmentation, and mixup that achieves state-of-the-art performance [14]. The audio spectrograms are generated using a Fast Fourier Transform (FFT) with a high temporal resolution. BirdNet is based on a ResNet [7] architecture and is capable of identifying 984 North American and European bird species.

More recently, BirdNET-Lite[2] has been released. This neural network is optimized for mobile and edge devices and can recognize more than 6,000 bird species. It takes raw audio as its input and generates spectrograms on-the-fly. Mühling et al. [19] proposed a task-specific neural network created by neural architecture search [24]. It won the BirdCLEF 2020 challenge [12]. It also operates on raw audio data and contains multiple auxiliary heads and recurrent layers.

Recently, Vision Transformers (ViT) achieved great improvements in computer vision tasks [4] and audio event classification[6]. Puget [20] adopted a ViT architecture for bird song recognition and achieved results comparable to CNNs. However, the annual birdcall identification challenge (BirdCLEF [13]) is currently dominated by approaches based on CNNs. The top approaches typically use ensembles of CNNs and heavy parameter tuning. The winning approach at BirdCLEF 2021, for example, uses Mel spectrograms, network architectures based on ResNet-50 [7], and gradient boosting to refine the results using metadata. The runners-up Henkel et al. [8] presented an ensemble of nine CNNs. During training, they used 30 second Mel spectrograms to mitigate the effect of the weakly labeled training data and applied a novel mixup scheme within and across training samples for extensive data augmentation. Furthermore, a binary bird call/no bird call classifier contributed to the final result. However, combining several machine learning models leads to a considerably increased computational effort.

[2] `https://github.com/kahst/BirdNET-Lite`

2.2 Edge AI for Biodiversity Monitoring

Executing machine learning algorithms on edge devices leads to a quantitative increase of data through continuous observation, where previously only individual data points could be collected with manual effort, often including a bias of individual experiences depending on, e.g., habitat or bird species. Merenda et al. [18] survey several approaches based on the execution of machine learning methods on hardware with limited resources. Gallacher et al. [5] deployed 15 sensors in a large urban park to process recorded audio data of bats locally, which allowed monitoring their activities for several months. Given that the system has only been operated in an urban environment, the limitations of this approach are that network connectivity must be available via WiFi, and that a fixed power supply must be present. Novel deep learning approaches presented by Disabato et al. [3] further improved bird song recognition at the edge. These approaches provide high accuracy while reducing computational and memory requirements, with limited battery lifetimes of up to 12.4 days on an STM32H743ZI microcontroller. Likewise, Zualkernan et al. [25] compare different edge computing platforms based on neural networks using bat species classification as an example. While the NVIDIA Jetson Nano is the only device capable of executing a TensorRT model on its GPU, both the Raspberry Pi 3B+ and the Google Coral showed good results when executing a reduced TensorFlow-Lite model.

3 Bird@Edge

Bird@Edge is designed as an *Edge AI* system based on distributed embedded edge devices to enable efficient, continuous evaluation of soundscapes recorded in forests. Multiple Bird@Edge Mics stream audio wirelessly to a local Bird@Edge Station, on which bird species recognition is performed. The recognition results of different Bird@Edge Stations are transmitted to a backend for further analysis. The results are stored in a time series database and can be visualized, as shown in Fig. 1. Using hidden microphones also supports recognizing very elusive species that are hard to detect while ecologists are present in field to conduct a census.

A Bird@Edge Station consumes significantly more power than a microphone node, but can run a neural network for bird species recognition for more than one audio stream. We can feed 1 to 10 audio streams into the neural network and thus operate a variable number of Bird@Edge Mics at one Bird@Edge Station. Different numbers of Bird@Edge Mics may be present when a new microphone node appears (e.g., by switching it on) or leaves (e.g., due to battery shortage).

To generate a list of bird species at a Bird@Edge Station, chunks of an incoming audio stream are passed to the neural network, which may return multiple results, since we process mixtures of recorded bird songs, i.e., soundscapes. These potentially multiple results per audio segment are then collected and aggregated into larger intervals in the time series database in the backend cloud. The size of the interval can be dynamically changed and visualized in near real-time. In addition, the status of the microphone nodes and potential problems can be detected much faster than collecting data only every few days.

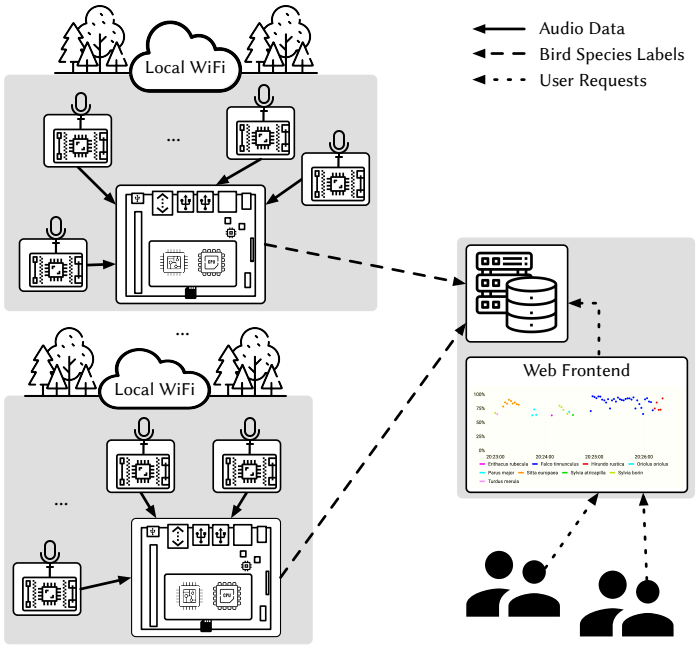

Fig. 1: Overview over the Bird@Edge system

3.1 Bird@Edge Hardware

The hardware used for Bird@Edge consists of (a) Bird@Edge Mics, which are in charge of recording and transmitting audio at the deployed location; (b) Bird@Edge Stations, which receive audio streams from multiple Bird@Edge Mics and execute the Bird@Edge processing pipeline. Figure 2 provides an overview of the hardware components used in Bird@Edge.

A Bird@Edge Mic consists of an Espressif ESP32 microcontroller that has a dual-core CPU running at 80 MHz, Bluetooth and WiFi connectivity, as well as multiple input and output options, including an I2S (Inter-IC Sound) bus. Connected to it is a Knowles SPH0645LM4H microphone capable of recording audio in the range between 50 Hz and 15 kHz[3]. A Bird@Edge Mic can be powered either using single 18650 Li-ion cells or using one of the widely available USB power banks. The price of a Bird@Edge mic of 22€ to 50€ is composed of the ESP32, depending on the offer and model 5€ to 15€, the I2S microphone 7 - 12€ and a battery for 10€ - 20€. All components can be placed in a small case of 10 x 10 x 5 centimeters, which does not exceed the weight of 500 grams.

At the heart of a Bird@Edge Station is a NVIDIA Jetson Nano. It allows the efficient execution of machine learning models in a low power environment. A Realtek RTL8812BU-based USB WiFi is used to enable wireless networking with

[3] `https://www.knowles.com/docs/default-source/default-document-library/sph0645lm4h-1-datasheet.pdf`

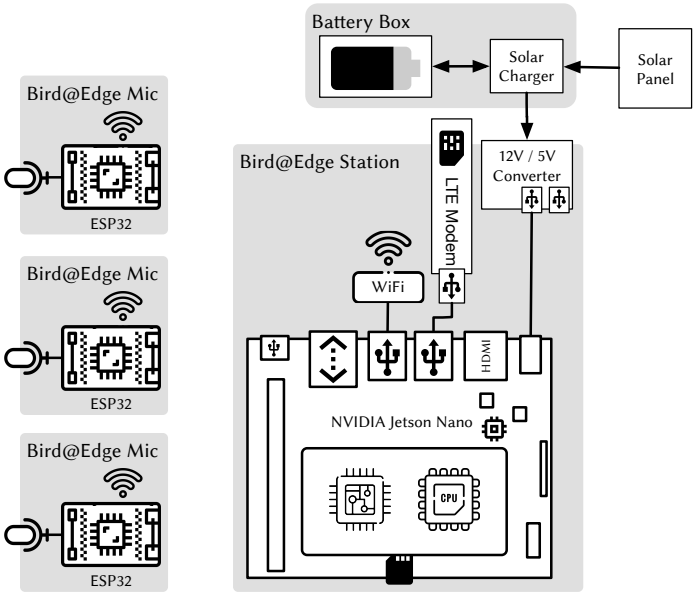

Fig. 2: Bird@Edge hardware components

the board and allow connection to the Bird@Edge Mics. In addition, a Huawei E3372H LTE modem is installed to connect to the Internet in rural areas. The station is powered by 12V solar battery system connected to the Jetson Board via a 12V/5V step down converter. The hardware of a Bird@Edge Station costs about 110€, with 50€ for the Jetson Nano, 20€ for the USB WiFi adapter, and 40€ for the LTE modem. The components of an Bird@Edge Station, including a solar charge controller, can be fitted into an industrial enclosure measuring 25 x 18 x 12 centimeters, weighing less than 1.5 kilograms in total.

3.2 Bird@Edge Software

Bird@Edge consists of a variety of software components that enable its smooth configuration and operation. Figure 3 shows these software components, as well as the data flows and interaction possibilities of the users with the system.

The software for the Bird@Edge Mics is built using components of the Espressif Development Framework (ESP-IDF), i.e., HTTP Server, Multicast DNS Implementation, and I2S drivers. When booting up, the Bird@Edge Mic connects to the WiFi network (SSID: BirdEdge) with the best signal strength and reports its own accessibility via the service identifier mDNS. Then, the HTTP server is started, which provides the audio stream of the microphone for the Bird@Edge station. To detect connection interruptions, the WiFi connection is also checked

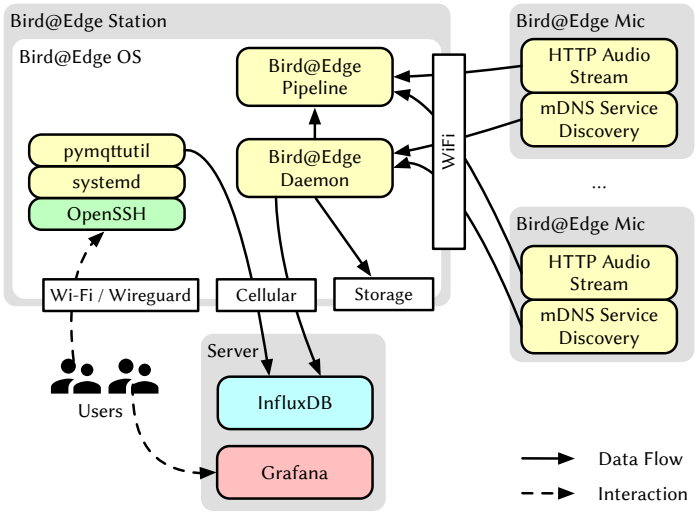

Fig. 3: Bird@Edge software components

at intervals of one second with the aid of ICMP and, if necessary, the WiFi connection is re-established. The Bird@Edge Mic software is available online[4].

The software running on a Bird@Edge Station is based on the NVIDIA Jetson Nano Development Kit Operating System, which in turn is based on Ubuntu Linux. The central component responsible for detecting the Bird@Edge Mics, executing the processing pipeline and transmitting the results is called birdedged (Bird@Edge Daemon). It continuously searches for newly connected Bird@Edge devices and restarts the processing pipeline accordingly when devices are found or dropped. Bird species recognition results from the processing pipeline are captured and transmitted to the InfluxDB server system. The server system that collects data from potentially multiple Bird@Edge implementations runs Grafana, a dashboard visualization WebUI designed specifically for stream data[5].

The operating system running on Bird@Edge Stations is built using pimod [10] and is available online[6]. NVIDIA's licenses do not allow to redistribute complete operating system images, however pimod allows to reduce the necessary steps and easily create the images.

4 Recognizing Bird Species in Soundscapes

In this section, we describe our deep learning approach to bird species recognition in audio recordings including the preprocessing steps, the neural network as well as its optimization and deployment on the NVIDIA Jetson Nano edge device. The

[4] `https://github.com/umr-ds/BirdEdge/tree/main/BirdEdge-Client`
[5] `https://grafana.com`
[6] `https://github.com/umr-ds/BirdAtEdge-OS`

deep neural network model is designed to recognize 82 bird species indigenous in Germany and background noise that is typical for German forests.

4.1 Audio Preprocessing

We selected 44.1 kHz as the sampling rate and analyzed frequencies up to 22.05 kHz to cover the frequency ranges of the bird song patterns. The task is considered as a classification problem, aiming to recognize bird species in 5-second audio snippets. To avoid overfitting and enrich our data set, we randomly select these 5-second snippets and add randomly selected noise from up to four background samples. This encourages our model to focus on the patterns that are important for species recognition. The recognition is based on visual representations of the frequency spectrum as it changes over time, called spectrograms. In our case, we use Mel spectrograms that are generated using 128 Mel bands and an FFT window size of 1,024.

4.2 Neural Network Architecture

Our approach to recognize bird species relies on an EfficientNet-B3 [22] architecture pre-trained on ImageNet [21]. The model is fine-tuned in two phases to target domain using the Adam [15] optimizer. In the first phase, we only train the last, randomly initialized layer for 40 epochs with an initial learning rate of 0.004, while the remaining layers with pre-trained weights are frozen. In the second phase, we train all layers of the model until convergence, while reducing the initial learning rate by a factor of 10. Furthermore, a binary cross-entropy loss combined with modulation factors motivated by the success of focal loss [16] in the field of object detection are used to emphasize difficult samples during the training process. Since the underlying data set is only weakly labeled, we use positive training samples for one species as negative samples for the others. Furthermore, samples labeled negative from expert feedback are defined as hard negatives in the following. Our loss function is defined as follows:

$$L = \sum_{k=1}^{K} l(y_k, p_k),$$

$$l(y,p) = \begin{cases} -\alpha_{pos}(1-p)^{\gamma}\log(p) & \text{if } y \text{ is positive} \\ -\alpha_{n}p^{\gamma}\log(1-p) & \text{if } y \text{ is negative} \\ -\alpha_{hn}p^{\gamma}\log(1-p) & \text{if } y \text{ is hard negative} \end{cases}$$

where K is the number of bird classes, p_k is the predicted probability for the k-th class, y_k is the k-th ground truth label, α_{pos} is the weighting factor for positive labels, α_n for negative or undefined labels, α_{hn} for hard negative labels and γ is the focusing parameter.

We implemented our approach using the TensorFlow deep learning framework [1]. For audio processing and especially spectrogram generation, we use the librosa library [17].

4.3 Optimizing the Neural Network for Edge Devices

To speed up inference, we optimized our model using the TensorRT[7] library. This library includes an inference optimizer for CUDA-capable target devices that applies various operations, such as quantization and memory optimization, to reduce the inference time. In particular, the floating point precision is reduced by quantizing to FP16 or INT8, while maintaining high accuracy. We optimized our model by using FP16 quantization in addition to the original FP32 weights, since the NVIDIA Jetson Nano does not support INT8 computations natively. Furthermore, we applied target-specific auto-tuning to select the best algorithms and quantization for each layer.

4.4 Inference

We use the DeepStream SDK[8] to deploy our optimized model on the NVIDIA Jetson Nano board with high throughput rates. DeepStream is based on the GStreamer framework and provides a pipeline that takes an input stream and performs hardware accelerated inference on it. An overview of our pipeline com-

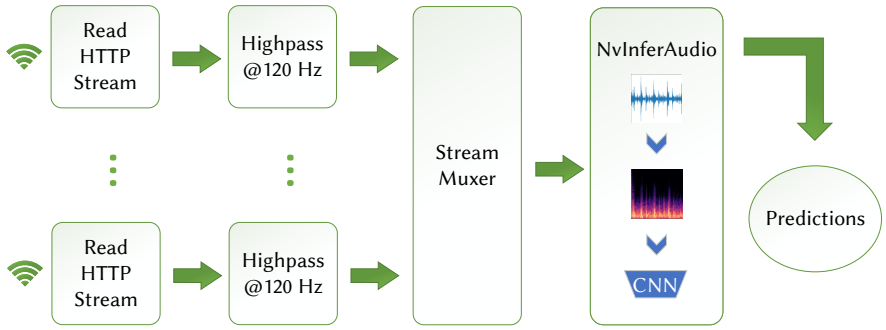

Fig. 4: Overview of the Bird@Edge processing pipeline

posed with DeepStream is presented in Figure 4. First, the N HTTP streams are read and parsed from the WiFi signal. Since the microphone we used (see Section 3 for details) induces noise in the lowest frequency bands, we apply a highpass filter that attenuates all frequencies below 120 Hz to each stream. These frequencies are irrelevant for bird species recognition and can therefore be neglected. We prefer the Chebyshev highpass filter over the windowed sinc filter, because it is much faster. Next, we use DeepStream's stream muxer to bundle our streams into one batch and forward the data to the NvInferAudio plugin. This plugin provides inference for audio streams and automatically generates the respective Mel spectrograms. Finally, the spectrograms are passed to our model with a batch size of N, and the obtained predictions are retrieved. To be able to process the streams in real-time with a high temporal resolution, we take 5 second snippets with a stride of one second.

[7] `https://developer.nvidia.com/tensorrt`
[8] `https://developer.nvidia.com/deepstream-sdk`

5 Experimental Evaluation

In this section, we present experimental results in terms of (a) bird species recognition quality and execution speed, (b) visualization of bird recognition results, as well as (c) power consumption measurements of a Bird@Edge Station and a Bird@Edge Mic.

5.1 Bird Species Recognition Quality and Execution Speed

Data Sets. Our neural network models were evaluated and compared to BirdNET [14] and BirdNET-Lite[2] on data sets collected from three sources. We recorded a first data set with AudioMoth devices [9] in the Marburg Open Forest (MOF). The recordings were labeled on a 5 second basis by human experts. In total, 33 species occur in the MOF data set. Since the amount of labeled data in the MOF data set is not sufficient to train a deep learning model, we acquired further data sets by crawling data from the online bird song collections Xeno-Canto [23] and iNaturalist [11]. The assets included in these data sets have often higher quality and contain less background noise. In our evaluation, we took up to 10% of the files of each class. To make sure that we do not feed snippets without bird calls, we first applied the heuristic used by Kahl et al. [14] and selected up to three 5 second snippets containing a bird call for each test file. Table 1 shows an overview of the training and test data.

Data Set	MOF	Xeno-Canto	iNaturalist
Training	4,294	104,989	30,631
Test	913	2,144	1,365

Table 1: Overview of the training and test data.

Quality Metrics. To evaluate the performance of our bird species recognition approach, we use average precision (AP) as our quality metric. The AP score is the most commonly used quality measure for retrieval results and approximates the area under the recall-precision curve. The task of bird call recognition can be considered as a retrieval problem for each species where the annotated audio samples represent the relevant documents. Then, the AP score is calculated from the list of ranked documents as follows:

$$AP(\rho) = \frac{1}{|R \cap \rho^N|} \sum_{k=1}^{N} \frac{\left|R \cap \rho^k\right|}{k} \psi(i_k),$$

$$\text{with} \quad \psi(i_k) = \begin{cases} 1 & \text{if } i_k \in R \\ 0 & \text{otherwise} \end{cases}$$

where N is the length of the ranked document list (total number of analyzed audio snippets), $\rho^k = \{i_1, i_2, \ldots, i_k\}$ is the ranked document list up to rank k, R is the set of relevant documents (audio snippets containing a bird call), $\left|R \cap \rho^k\right|$ is the number of relevant documents in the top-k of ρ and $\psi(i_k)$ is the relevance function. Generally speaking, AP is the average of the precision values at each relevant document. To evaluate the overall performance, the mean AP score is calculated by taking the mean value of the AP scores from each species.

Method	MOF	XC	iNat
BirdNET[14]	0.833	0.725	0.725
BirdNET-Lite[2]	0.859	0.737	0.714
EfficientNet-B3	0.952	0.820	0.811
Bird@Edge	0.952	0.816	0.819

Table 2: Results (mAP).

Model	Device	Inference time (ms)
BirdNET-Lite[2]	Raspberry Pi-4B	279
Bird@Edge (FP32)	Jetson Nano	64
Bird@Edge	Jetson Nano	54

Table 3: Model inference runtimes.

Results. First, we evaluated the recognition quality of our models, namely the original trained model (EfficientNet-B3) as well as the optimized model (Bird@Edge) and compare the results to BirdNET and BirdNET-Lite. While EfficientNet-B3 is evaluated with TensorFlow on a workstation, the Bird@Edge model is run on the NVIDIA Jetson Nano for inference.

BirdNET and BirdNET-Lite take the recording location as additional metadata along with the corresponding audio input. As longitude and latitude, we take the coordinates of the Marburg Open Forest for all data sets, since we only use bird species resident in this specific forest for evaluation. Since the length of the audio input of the BirdNET models differs from our approach, the 5 second samples are split into two 3 second snippets with an overlap of 1 second and the results are averaged for the final prediction.

Table 2 summarizes the experimental bird species recognition results. Our original model (EfficientNet-B3) outperforms BirdNET-Lite as well as BirdNET by roughly 10% in terms of mAP on all data sets considered. While keeping the recognition quality, the optimized Bird@Edge model achieves an inference runtime of 64 ms per spectrogram, as shown in Table 3. Adding model quantization with 16-bit floating point precision where appropriate effectively reduces the inference runtime on the NVIDIA Jetson Nano board by 10 ms. We also compared

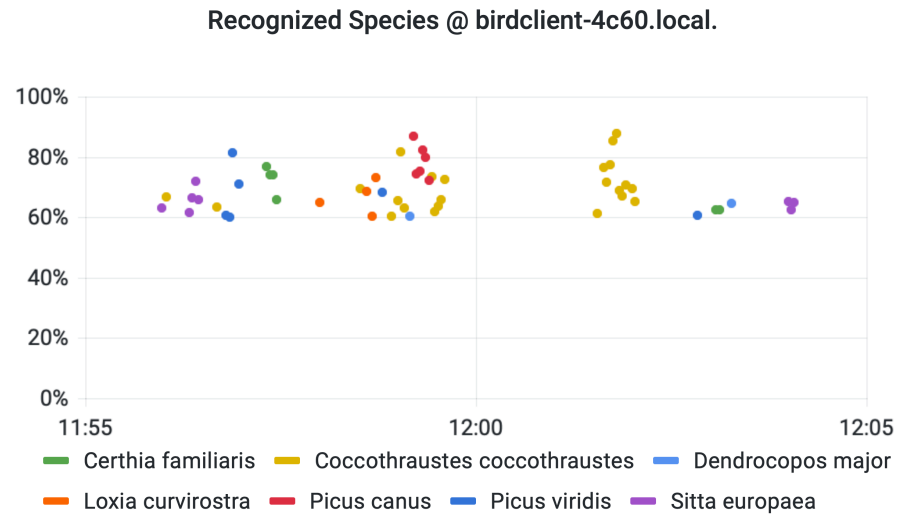

Fig. 5: Grafana panel (x-axis: clock time; y-axis: recognition confidence) showing recognized bird species of a certain Bird@Edge Mic, based on Xeno-Canto file XC706150, recorded by user *brickegickel*

the runtimes of our models to BirdNET-Lite. Similar to BirdNET-Pi[9], we ran BirdNET-Lite on a Raspberry Pi-4B with 4 CPU threads in parallel. Table 3 reveals that our setting is more than four times faster.

5.2 Visualization of Bird Species Recognition Results

Figure 5 shows a Grafana screenshot of an automatically generated graph of the recognized bird species of a Bird@Edge station. To generate the figure, the publicly available soundscape audio file XC706150 from Xeno-Canto[10] of the target area was played back and captured by the Bird@Edge Mic. The clock time is shown on the x-axis. The confidence of the recognition is plotted on the y-axis. The data is grouped according to the recognized bird species labels, distinguished by color. For every Bird@Edge Mic, a separate figure is generated, and its parameters, e.g., plotted time frame or selection of species, can be configured.

Some observations can be derived from this simple visualization. First, there are several recognized occurrences of *Coccothraustes coccothraustes* (hawfinch) in two clusters. *Picus canus* (grey-headed woodpecker) is detected multiple times over the duration of 12 seconds, and *Sitta europaea* (Eurasian nuthatch) is detected in two clusters each at the beginning and end of the observation period.

[9] `https://github.com/mcguirepr89/BirdNET-Pi`
[10] `https://xeno-canto.org/706150`

All three observations indicate that individuals were heard on the recordings and were in the area at these times. For *Loxia curvirostra* (red crossbill) and *Dendrocopos major* (great spotted woodpecker), only 4 and 2 observations were made, respectively; these were probably heard only in the background. More sophisticated analyses can be performed based on researchers' requirements, such as heat maps of the occurrence of species based on their geo-positions, or time-based plots. This can include both short-term considerations, such as the time of day at which certain species are active, or long-term aspects, such as during which period a particular species is particularly active.

5.3 Power Consumption

An important aspect for the applicability of Bird@Edge in real applications is its power consumption. Therefore, we measured the power consumption of a Bird@Edge Station and a Bird@Edge Mic.

To measure the power consumption of a Bird@Edge Station, we used the internal power monitors of the NVIDIA Jetson Nano, since these enable the differentiation between CPU, GPU, and total power consumption. The power measurements were performed in different profiles: a) the 10 Watt maximum performance profile (default), b) the 5 Watt low power profile from NVIDIA, and c) a custom low power profile created for Bird@Edge. In this custom power profile, only 2 of the 4 CPU cores were used, running at a maximum frequency of 614 MHz, and the GPU was limited to a maximum of 230 MHz. As a baseline, the power consumption is measured with 5 connected Bird@Edge mics, and only the measured values while the pipeline is running are averaged. In this setup, the maximum performance mode requires 4.86 W, the low power profile 4.19 W and the custom low power mode only requires 3.16 W, i.e., roughly 35% compared to the maximum performance mode with no performance degradation observable. Our observations during the execution of the experiments suggest that the GPU's dynamic frequency scaling algorithm tends to be too conservative to permanently lower the clock and thus prevents the possible lower power consumption.

Figure 6 shows the power consumption of a Bird@Edge station in a short scenario with a changing number of connected Bird@Edge Mics. At the beginning, the system is switched on, but the neural network for bird species is not running; the system needs 2.1 W in this state. At t=0, the neural network is started with a Bird@Edge Mic already connected to the station. The neural network model is loaded into memory from t=6, for which the CPU requires up to 0.59 W. From time t=36, i.e., 30 seconds after the start of the pipeline, the neural network model runs and forwards results to the backend. In this phase, the Bird@Edge station requires an average of 3.18 W. At t=120 and t=180, 4 and 5 additional Bird@Edge Mics are switched on, which first connect to the Bird@Edge Station via WiFi, then are discovered via mDNS, which results in the reconfiguration of the processing pipeline and its restart. In both cases, the reboot took about 35 seconds, with 5 seconds for WiFi connection and discovery, and 30 seconds for pipeline reboot. With 5 and 10 Bird@Edge Mics connected, the Bird@Edge Station requires 3.16 W and 3.13 W, respectively. Particularly noteworthy is

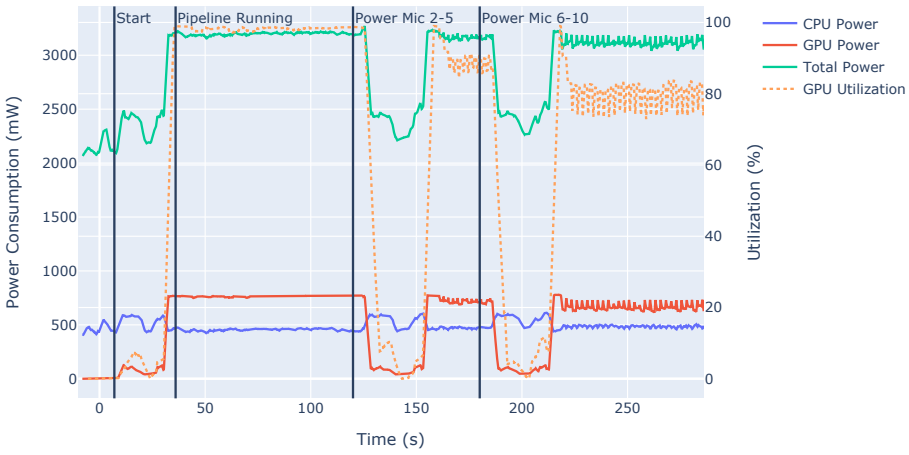

Fig. 6: Power consumption of a Bird@Edge Station in a dynamic scenario.

the station's lower power consumption when a larger number of Bird@Edge Mics are connected. Figure 6 indicates that GPU utilization is lower with many Bird@Edge Mics connected, compared to smaller numbers of Bird@Edge Mics (98% with 1 client, 88% with 5 clients, 80% with 10 clients). This is probably due to internals of the DeepStream SDK and may be influenced by the implementation, e.g., with respect to the handling of unused streams.

The magnitude of the Bird@Edge Station's power consumption necessitates the use of an LFP, VRLA or AGM battery, which at 12 Volts typically have a capacity of 50 to 200 Ah. The available 60 to 2400 Wh allow the operation of a Bird@Edge Station from 7 up to 31 days. In combination with a solar panel between 50 and 100 watt, a continuous operation is also possible during periods of weak sunlight.

To measure the power consumption of a Bird@Edge Mic, we used a Monsoon High Voltage Power Monitor[11] and connected the ESP32 using the 3.3 volt input. The measurements of a Bird@Edge Mic show a power usage of 665 mW whenever the stream is activated and data is sent to the station. The power measurements were performed with three different off-the-shelf ESP32 boards, since the additional electronics present on the boards can have an additional impact on power consumption. The three boards differed only slightly in terms of power consumption: 0.452 W was needed by the *Adafruit HUZZAH32*, 0.452 W by the *Joy-IT NodeMCU ESP32*, and 0.421 W by the *SparkFun ESP32 Thing Plus*. The latter[12] is the board of choice for our application, due to the lowest power consumption, a direct LiPo battery connector, and an external WiFi antenna.

To get a realistic estimation of the battery life of a Bird@Edge Mic, further measurements were performed with 3.7 Volt via the corresponding connectors for

[11] `https://www.msoon.com/high-voltage-power-monitor`
[12] `https://www.sparkfun.com/products/15663`

LiPo batteries. The SparkFun board required 0.468 W or 126.6 mA in operation, whereas the Adafruit board required 132.9 mA, or 0.492 W. LiPo batteries are available in a wide range of capacities, from 100 mAh to over 30000 mAh. Typical capacities, as they are found in smartphones and can be purchased cheaply, are around 3500 mAh, which allow a runtime of 27.6 hours. In combination with a small solar panel of around 10 Watts, continuous operation is thus easily feasible.

6 Conclusion

We presented Bird@Edge, an *Edge AI* system for recognizing bird species in audio recordings to support real-time biodiversity monitoring. Bird@Edge is composed of embedded edge devices, such as ESP32-based microphones and NVIDIA Jetson Nano boards, operating in a distributed system to enable efficient, continuous evaluation of soundscapes recorded in forests. We presented a deep neural network based on the EfficientNet-B3 architecture and optimized for execution on a NVIDIA Jetson Nano board to recognize bird species in soundscapes. It outperforms the state-of-the-art BirdNET neural network on several data sets and achieves a recognition quality of up to 95.2% mean average precision on soundscape recordings in the Marburg Open Forest, a research and teaching forest of the University of Marburg, Germany. Measurements of the power consumption of Bird@Edge Station and Bird@Edge Mics show that the system has an acceptable demand of 3.18 W plus 0.492 W for each Bird@Edge Mic, which can be covered by reasonably sized batteries and solar panels, highlighting the real-world applicability of the approach. All software and firmware components of Bird@Edge are available under open source licenses[13].

There are several areas for future research. For example, self-supervised learning could be used to leverage the vast amount of unlabeled data and to improve the recognition quality on the target domain. Furthermore, continual and federated learning of machine learning models at the edge are interesting future research topics. Finally, a real-world test of several Bird@Edge deployments should be performed to identify potential problems in harsh environments.

Acknowledgments

This work is funded by the Hessian State Ministry for Higher Education, Research and the Arts (HMWK) (LOEWE Natur 4.0, LOEWE emergenCITY, and hessian.AI Connectom AI4Birds), the German Academic Exchange Service (DAAD) (Transformation Partnership Program; Project OLIVIA), and the German Research Foundation (DFG, Project 210487104 - Collaborative Research Center SFB 1053 MAKI).

[13] `https://github.com/umr-ds/BirdEdge`

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`
2. Darras, K., Batáry, P., Furnas, B., Celis-Murillo, A., Van Wilgenburg, S.L., Mulyani, Y.A., Tscharntke, T.: Comparing the sampling performance of sound recorders versus point counts in bird surveys: A meta-analysis. Journal of Applied Ecology **55**(6), 2575–2586 (2018). https://doi.org/10.1111/1365-2664.13229
3. Disabato, S., Canonaco, G., Flikkema, P.G., Roveri, M., Alippi, C.: Birdsong detection at the edge with deep learning. In: 2021 IEEE International Conference on Smart Computing (SMARTCOMP). pp. 9–16. IEEE (2021)
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: 9th Int. Conference on Learning Representations, ICLR 2021, Austria (2021)
5. Gallacher, S., Wilson, D., Fairbrass, A., Turmukhambetov, D., Mac Aodha, O., Kreitmayer, S., Firman, M., Brostow, G., Jones, K.: Shazam for bats: Internet of things for continuous real-time biodiversity monitoring. IET Smart Cities (2021)
6. Gong, Y., Chung, Y., Glass, J.R.: AST: audio spectrogram transformer. In: Interspeech 2021. pp. 571–575 (2021). https://doi.org/10.21437/Interspeech.2021-698
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016). https://doi.org/10.1109/CVPR.2016.90.
8. Henkel, C., Pfeiffer, P., Singer, P.: Recognizing bird species in diverse soundscapes under weak supervision. In: Faggioli, G., Ferro, N., Joly, A., Maistro, M., Piroi, F. (eds.) Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21-24, 2021. CEUR Workshop Proceedings, vol. 2936, pp. 1579–1586. CEUR-WS.org (2021), `http://ceur-ws.org/Vol-2936/paper-134.pdf`
9. Hill, A.P., Prince, P., Snaddon, J.L., Doncaster, C.P., Rogers, A.: Audiomoth: A low-cost acoustic device for monitoring biodiversity and the environment. HardwareX **6**, e00073 (2019). https://doi.org/10.1016/j.ohx.2019.e00073
10. Höchst, J., Penning, A., Lampe, P., Freisleben, B.: PIMOD: A Tool for Configuring Single-Board Computer Operating System Images. In: 2020 IEEE Global Humanitarian Technology Conference (GHTC 2020). pp. 1–8. Seattle, USA (Oct 2020). https://doi.org/10.1109/GHTC46280.2020.9342928
11. iNaturalist: A community for naturalists, `https://www.inaturalist.org/`
12. Kahl, S., Clapp, M., Hopping, W.A., Goëau, H., Glotin, H., Planqué, R., Vellinga, W., Joly, A.: Overview of birdclef 2020: Bird sound recognition in complex acoustic environments. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020. CEUR Workshop Proceedings, vol. 2696. CEUR-WS.org (2020), `http://ceur-ws.org/Vol-2696/paper_262.pdf`

13. Kahl, S., Denton, T., Klinck, H., Glotin, H., Goëau, H., Vellinga, W., Planqué, R., Joly, A.: Overview of birdclef 2021: Bird call identification in soundscape recordings. In: Faggioli, G., Ferro, N., Joly, A., Maistro, M., Piroi, F. (eds.) Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21-24, 2021. CEUR Workshop Proceedings, vol. 2936, pp. 1437–1450. CEUR-WS.org (2021), http://ceur-ws.org/Vol-2936/paper-123.pdf
14. Kahl, S., Wood, C.M., Eibl, M., Klinck, H.: Birdnet: A deep learning solution for avian diversity monitoring. Ecological Informatics **61**, 101236 (2021). https://doi.org/10.1016/j.ecoinf.2021.101236
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), https://arxiv.org/abs/1412.6980
16. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. 2017 IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
17. McFee, B., Raffel, C., Liang, D., Ellis, D.P., McVicar, M., Battenberg, E., Nieto, O.: librosa: Audio and music signal analysis in python. In: Proceedings of the 14th python in science conference. vol. 8 (2015)
18. Merenda, M., Porcaro, C., Iero, D.: Edge Machine Learning for AI-enabled IoT devices: A Review. Sensors **20**(9), 2533 (2020)
19. Mühling, M., Franz, J., Korfhage, N., Freisleben, B.: Bird species recognition via neural architecture search. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020. CEUR Workshop Proceedings, vol. 2696. CEUR-WS.org (2020), http://ceur-ws.org/Vol-2696/paper_188.pdf
20. Puget, J.F.: Stft transformers for bird song recognition. In: Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21-24, 2021. CEUR Workshop Proceedings, vol. 2936. CEUR-WS.org (2021), http://ceur-ws.org/Vol-2936/paper-137.pdf
21. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
22. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (2019). https://doi.org/1905.11946
23. Xeno-canto: Sharing bird sounds from around the world, https://www.xeno-canto.org/
24. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017)
25. Zualkernan, I., Judas, J., Mahbub, T., Bhagwagar, A., Chand, P.: An aiot system for bat species classification. In: 2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS). pp. 155–160 (2021). https://doi.org/10.1109/IoTaIS50849.2021.9359704