A Licenses

A.1 Code license

MIT License

Copyright (c) 2021 HumBug-Mosquito

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

A.2 Database license

CC-BY-4.0, https://creativecommons.org/licenses/by/4.0/

B Code use

B.1 Code access and structure

- The audio recordings and metadata `csv` are hosted on Zenodo http://doi.org/10.5281/zenodo.4904800. under a CC-BY-4.0 license.
- Code (and the metadata `csv` for completeness) is hosted on https://github.com/HumBug-Mosquito/HumBugDB under the MIT license.

The GitHub data directory structure is as follows:

```
HumBugDB
├── README.md
├── *requirements.txt
├── notebooks
│   ├── main.ipynb
│   └── supplement.ipynb
├── data
│   ├── metadata
│   │   └── *.csv
│   └── audio
│       └── *.wav
├── lib
│   ├── config.py
│   ├── config_keras.py
│   ├── config_pytorch.py
│   ├── feat_util.py
│   ├── runKeras.py
│   ├── runTorch.py
│   ├── ResNetDropoutSource.py
│   ├── evaluate.py
│   └── write_audio.py
└── outputs
    ├── models
    │   ├── keras
    │   └── pytorch
    ├── features
    └── plots
```

A `README` and several `requirements` are included for installing Keras, PyTorch, and dependencies for the code. The metadata is located in `/data/metadata/` as a `csv` file.

Extract the audio from Zenodo to the folder `/data/audio/` and launch the Jupyter notebook `main.ipynb` to perform train-test splitting, feature extraction, model training, and evaluation. The notebook imports from `lib` the necessary files depending on the choice of kernel and PyTorch or Keras.

B.2 Code manual

Top-level notebook `main.ipynb` performs data partitioning, feature extraction and segmentation in `get_train_test_from_df()`, model training in `train_model()`, and model evaluation in `get_results()`. The code is configured with `config.py`, where data directories are specified for the data, metadata and outputs, and feature transformation parameters are supplied. Model hyperparameters are given in `config_keras.py` or `config_pytorch.py`. The notebook supports both Keras [Chollet et al., 2015] and PyTorch [Paszke et al., 2019] with a common interface for convenience. In more detail, each top-level function is described as follows:

- `get_train_test_from_df(df_train, df_test_A, df_test_B)` extracts, reshapes, strides, and normalises `librosa` features for use as tensors, and saves them to `config.dir_out`, if features with that particular configuration do not exist already. The

Table 4: Feature transformation parameters, in samples. Audio processed with `librosa` at 8,000 Hz. The size of 1 frame in w is equal to `hop_length`. For our parameterisation this is 64 ms, resulting in an input feature slice of $w = 1.92$ s duration and $h = 128$ height.

`NFFT`	`win_size`	`hop_length`	h (`n_mels`)	w	Stride
2048	2048	512	128	30	512

data is split into train and test based on the matches of experiment ID to the audio tracks from the metadata given in `df_train, df_test_A, df_test_B`. It is important that no test recordings from these experiments are seen during training in advance, as otherwise model performance is overestimated. Appendix B.3, Table 5 shows the result of feature extraction with baseline feature parameters.

- `train_model(X_train, y_train, X_val=None, Y_val=None)` trains the BNNs on the data supplied (with validation data optional). The assumed input shape is that of the features produced by `get_train_test_from_df()`. The model architecture and training strategies may be changed in `runKeras.py` or `runTorch.py`.
- `get_results(model, X, y, n_samples=1)` evaluates the model object on test data `{X, y}` with the number of MC dropout samples as `n_samples`. If using deterministic networks, leaving the input argument blank will default to a single evaluation.

Supplementary notebook `supplement.ipynb` is used to reproduce the plots of species distribution in this paper (Figure 6) and contains utilities that were used for debugging and visualising the data, should they be helpful for researchers using their own functions.

B.3 Feature parameters

We first need to define the number of feature windows that are used to represent a sample, $\mathbf{X}_i \in \mathbb{R}^{h \times w}$, where h is the height of the two-dimensional matrix, and w is the width. The longer the window, w, the better potential the network has of learning appropriate dynamics, but the smaller the resulting dataset in number of samples. It may also be more difficult to learn the salient parts of the sample that are responsible for the signal, resulting in a weak labelling problem [Kiskin et al., 2019]. Early mosquito detection efforts have used small windows due to a restriction in dataset size. For example, Fanioudakis et al. [2018] supplies a rich database of audio, however the samples are limited to just under a second. However, despite the mosquito's simple harmonic structure, its characteristic sound also derives from the temporal variations, as is visible from spectrograms. We suspect this flight behaviour tone is better captured over longer windows, since we achieved more robust results with $w = 30, h = 128$, corresponding to 30 frames per window, each of 64 ms duration for a total audio slice of 1.92 seconds per sample. Nevertheless, we encourage researchers to make use of any data they wish to augment their model, for example by padding with noise to match the window size of this architecture, or by choosing a smaller window to extract features from.

To create an augmented dataset, we stride the input signal feature window with a step of 5 feature windows (a duration of 320 ms) Note that the training data is segmented by using overlapping strides specified with `config.step_size=5`, whereas the test data is created with no overlap. Samples that do not divide evenly into the window size are discarded (this is a very small number when using such a small step, and we prefer this option over padding with zeros or noise, though alternate solutions are welcome).

B.4 Baseline models

Keras We give the full model structure in Figure 3. Lambda layers are dropout layers which are placed to perform MC dropout at test-time. This structure bares similarity to VGGish[4], which uses 0.96 second log-mel spectrogram patches as inputs, and 11 weight layers (primarily convolutional layers and max-pool layers).

[4] https://github.com/tensorflow/models/tree/master/research/audioset/vggish

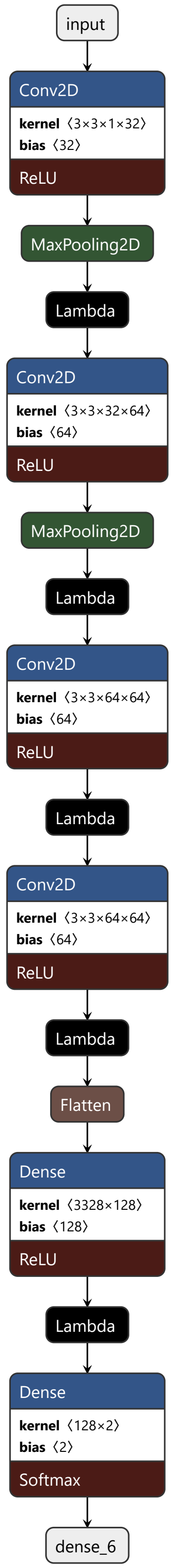


Figure 3: BCNN Keras model. Log-mel spectrograms are input with $w = 40, h = 128$, and passed through the above model. Lambda layers are dropout layers with probability 0.2. Made with https://github.com/lutzroeder/netron.

Table 5: Statistics of samples passed to models. Shown in number of data samples $\mathbf{X}$ resulting from log-mel feature transformation with a window size of 1.92 s, a step size of 0.32 s over the data of Table 7. All the variables affecting the size of the dataset created are found in `config.py`. and documented in `main.ipynb`.

Data	Mosquito	Negative	Total
Train	164,677	138,784	303,461
Test A: Tanzania (bednet)	1,714	2,068	3,782
Test B: Oxford Zoology (caged)	430	1,015	1,445

PyTorch ResNet-X We modify the final layers for compatibility with our data (in `runTorch.py`). Furthermore, we have augmented the construction blocks `BasicBlock()` and `Bottleneck()`, as well as the overall model construction, to feature dropout layers to act as an approximation for the model posterior at test-time. Dropout is implemented implicitly in `ResNetSource.py`, to not interfere with the behaviour of `model.eval()`, which by default disables dropout layers at test-time, removing the necessary stochastic component. Select which version of ResNet to use by modifying the class `ResnetDropoutFull` in `runTorch.py`. For ResNet-18, 34 the final `self.fc1` layer is of size `[512,1]`, whereas ResNet-50 the size is `[2048,1]`. A quick way to check this is to print `x.shape()` before the creation of the `fc1` layer.

Model training To select the loss that is used to define the best performing model, edit `runTorch.py` to make use of `train_acc` (or any other metric as desired) by replacing line 126. Similarly, amend the training epoch loop starting at line 85 to change other metrics or properties during training. In `runKeras.py`, supply arguments and any other desired callbacks and model checkpointing strategies to `model.fit()` in line 105.

Memory optimisation Note that the default settings require at least 16 GB RAM to load into memory for ResNet-50 processing, as channels are replicated 3 times to match the pre-trained weights model. To reduce the strain on memory, increase the `step_size` parameter in `config.py` to reduce the number of windows created by feature extraction. This reduces the overlap between samples.

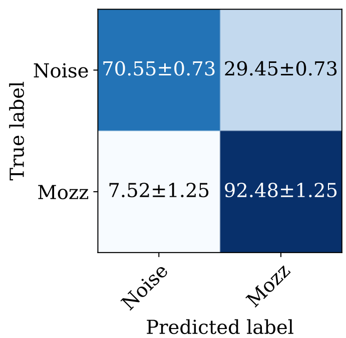
Alternatively, it is possible to use a non-pretrained architecture and change the tensor creation code in `build_dataloader()` from `runTorch.py` to remove `.repeat(1,3,1,1)` as there will be no need to copy over identical data over three channels.

Note that once the tensors have been created, VRAM is not an issue due to the batching over the dataloaders (this code has been run on a GTX970 with 3.5 GB useable VRAM).

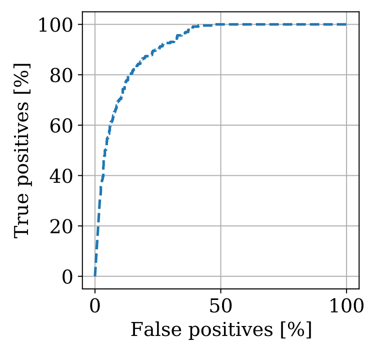
Hyperparameters Configure the hyperparameters in `config_pytorch.py` and `config_keras.py`. The number of epochs was set by observing the learning rate of the network. Within a few epochs, the models began to strongly overfit, with the training accuracy failing to improve validation accuracy. For this reason, both models are set to a low epoch number, and have a fairly low `max_overrun` counter, which determines the maximum number of steps taken for which the target metric fails to improve. The dropout rate and batch size were set to 0.2 and 32, values which are generally risk-free. We note here that the point at which we stop training the model made a fairly significant difference to the balance between true positive and true negative errors (despite a similar overall ROC AUC score). In this respect, the optimisation procedure for the models could be improved with more careful thought about the metrics used for training. If error types are important, consider using loss-calibrated approaches such as that of Cobb et al. [2018].

B.5 Test performance

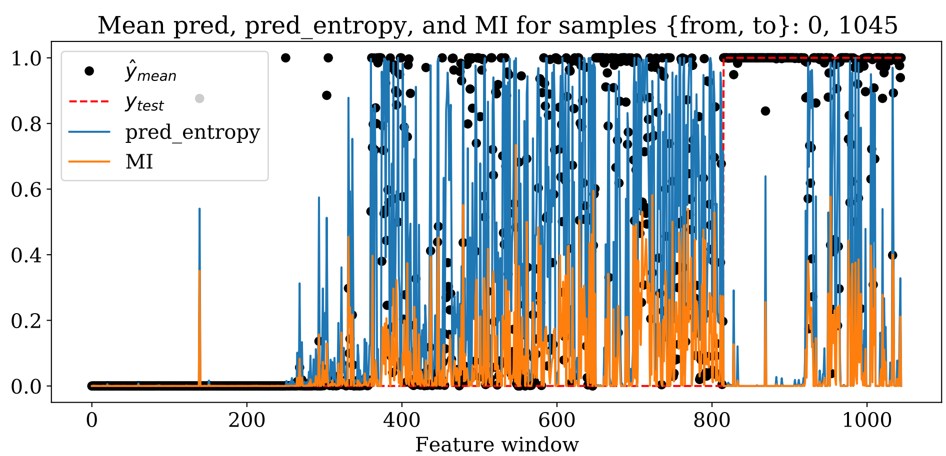
Verifying data integrity of Test B To support the validity of this data, we train the Keras model on half of Test B and test on the other half (with recordings held out), with the settings: `epochs = 7, tau = 1.0, dropout = 0.2, validation_split = None, batch_size = 32, lengthscale = 0.01` to achieve the results of Figure 4.



(a) Confusion matrix



(b) ROC AUC: 0.915 ± 0.003



(c) Mean prediction, predictive entropy and mutual information for feature windows

Figure 4: Performance on half of a hold-out test set constructed from Test B. Confusion matrices given in normalised percentage, and ROC in the form of mean $\pm$ standard deviation, across $N = 10$ MC dropout samples.

C PostgreSQL Database

C.1 Database metadata

The data presented in this paper are regularly maintained in a PostgreSQL database. For completeness, we include the full schema in Figure 5. We note that since data upload is a constant work in progress, some fields have not yet been populated sufficiently to be useful upon data extraction. We thus restrict the metadata to the fields that have been verified, and are most likely to be of greatest use. The command we use to extract all the metadata for this paper is as follows:

```
\copy (SELECT label.id, fine_end_time-fine_start_time, name,
sample_rate, record_datetime, sound_type, species, gender, fed, plurality,
age, method, mic_type, device_type, country, district, province, place,
location_type
FROM label
LEFT JOIN mosquito ON (label.mosquito_id = mosquito.id)
RIGHT JOIN audio ON (label.audio_id = audio.id)
RIGHT JOIN device ON (audio.dev_id = device.id)
WHERE type = 'Fine'
AND fine_start_time IS NOT NULL AND sound_type in
('mosquito', 'background', 'audio', 'wasp','fly') AND
(path LIKE '%Kenya%'
OR path LIKE '%Thai%'
OR path LIKE '%Tanzania%'
OR path LIKE '%LSTMH%'
OR path LIKE '%CDC%'
OR path LIKE '%Culex%')
ORDER BY path) to '/data/export/neurips_2021_zenodo_0_0_1.csv' csv header;
```

We will now break down each metadata field in the data release by the table it originated from and its column heading.

Label:

- `label.id` selects the column `id` from the table `label`, which is joined to `audio` on `label.audio_id = audio.id`. This allows us to now extract a labelled section of audio as indicated by the start and end times of the label.
- `fine_start_time`, `fine_end_time` are the tags for start and end of the audio label, with reference to the original audio recording. Once audio is extracted, we assign the labelled section the filename set to the `label.id`, and define a column `length` which takes the value of `fine_start_time - fine_end_time` for each new label.

Audio:

- `name`: The original filename of the recording (including file extension).
- `sample_rate`: The sample rate of the recording.
- `record_datetime` The time of recording, as SQL `DATETIME` object (easy to parse with either `pandas` or built-in `datetime.datetime`). For newer data, this timestamp is exact, however data collected prior may only be correct to the month.

Mosquito:

- `species` is the species of the mosquito, either the species complex, or more specifically the species if available (e.g. *An. arabiensis* of the complex *An. gambiae s.s.*). If no species information is available, this field is blank (or `NaN` when imported by `pandas` with default settings). A full breakdown of the available species per experimental group is given in Figure 6 and Table 6.
- `gender`: Gender of mosquitoes (`M` or `F`) or blank if not known.
- `fed`: Whether mosquito has been fed (`t` or `f`) or blank.

- `plurality`: The quantity of mosquitoes recorded at one instant: `single`, `plural` or blank if unknown.
- `age`: The age of mosquito in days.
- `sound_type`: denotes whether the label corresponds to a mosquito event if `mosquito`, but can take the value of `background` for corresponding background, `audio` for sections of dense audio events not containing mosquito or `wasp` and `fly`. When parsing data, a binary distinction between `mosquito` and `NOT mosquito` can be made safely.

Device:

- `method`: The method of capture of mosquitoes, taking values `HBN`, `LT`, `ABN`, `LC`, `HLC` or none if not known (or applicable). Human-baited nets (`HBN`) are a form of mosquito intervention where humans are surrounded by a mosquito net. As part of the HumBug project, adapted bednets were used where an additional canopy to hold smartphones for recording was sewn on (from 2020 onwards) [Sinka et al., 2021].

  Animal-baited nets follow the same concept but involve an animal as the main attractant for mosquitoes.

  CDC light traps (`LT`) use several attractants to lure mosquitoes into the collection chamber. Light is the primary source, but bottled CO_2, gas or dry ice can also be used.

  Larval collections, where the eggs of young mosquitoes are collected, are denoted `LC`.

  Human landing catches, where mosquitoes that landed on humans are caught, are denoted `HLC`.

  For mosquitoes raised from culture and not released into the wild and/or near any nets, this field is blank.
- `mic_type`: The microphone used. Takes values `telinga`, `phone` to denote the microphone type. Use this field to filter audio by the type of sound produced, if you wish to check for bias arising from recording device. Further refine the search with the phone model as specified in `device_type`.
- `device_type`: the device to which the microphone was connected. E.g. the field microphone (Telinga) was connected to a `Tascam` or `Olympus` recorder. If a smartphone was used, the device is the phone model (e.g. `itel A16` or `Alcatel 4015X`).

Location:

- `location_type`: The environment in which the mosquitoes were recorded in, taking values `cup` for mosquitoes recorded in sample cups, `field` for mosquitoes recorded free-flying in the field (applicable to Tanzania 2020 bednet recordings), or `culture` for mosquitoes recorded in culture cages.
- `country`, `district`, `province`, `place`: The country, district, province, and name of the recording site (e.g. `USA`, `Georgia`, `Atlanta`, `CDC insect culture, Atlanta`). Use these values combined with `location_type` to filter data by recording experiment.

Figure 5: Relational tables of the full PostgreSQL database which was used to generate the data for this paper. The structured nature of the database enforces a standard in label format, ensuring we can efficiently mix and match data from a wide range of experiments with differing protocols. For example, if we wish to investigate the effect of mosquito gender or microphone type on the ability to detect mosquitoes, we may sub-select data with the appropriate metadata with one query. Database schema generated with `dbdiagram.io` from with `pg_dump -s`.

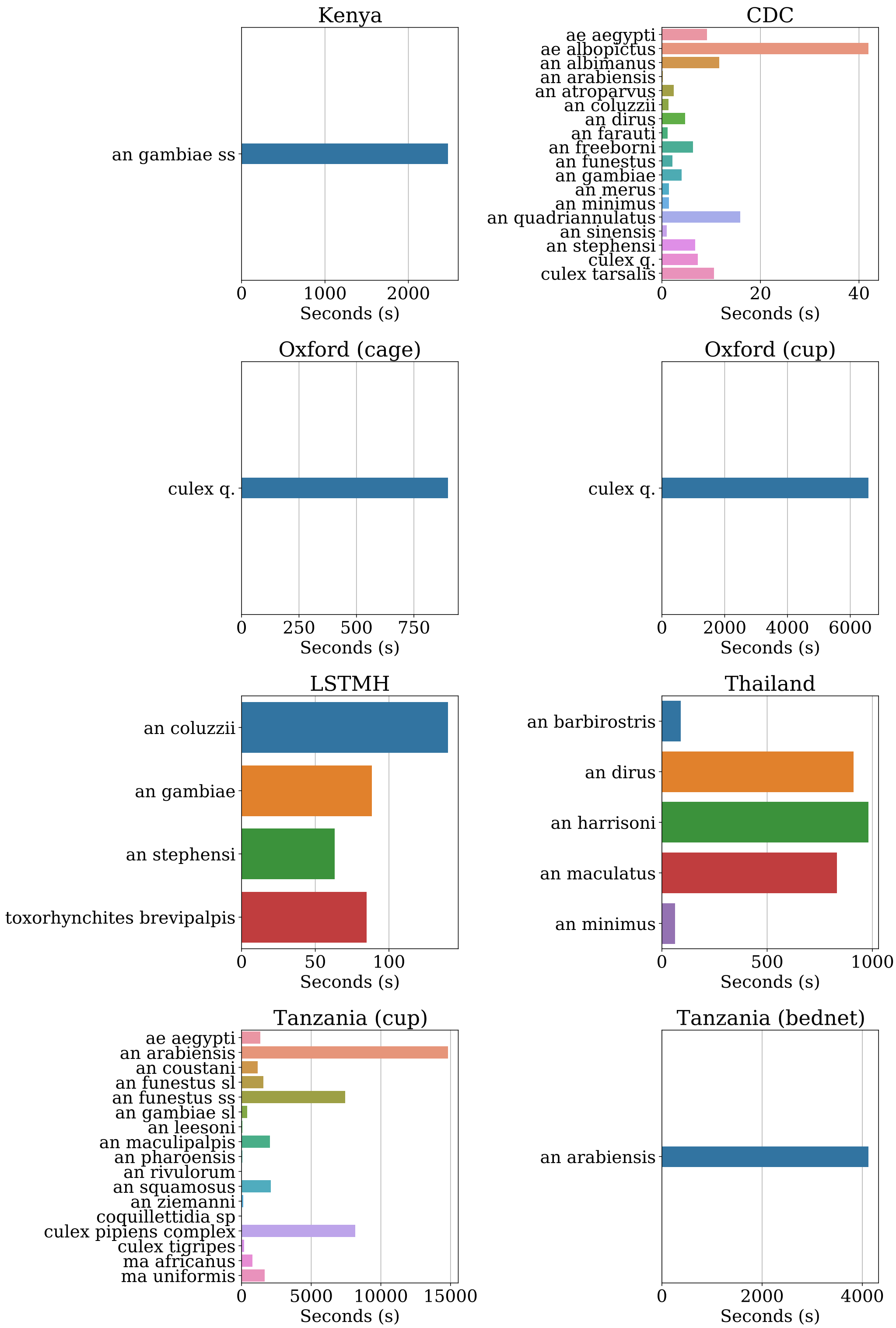


Figure 6: Species distribution per experiment corresponding to Table 7.

Table 6: Species distribution per experimental group corresponding to Table 7 and Figure 6.

Species	Species per location (seconds)								
	Kenya	USA (CDC)	Oxford (cup)	Oxford (cage)	LSTMH	Thailand	Tanzania (cup)	Tanzania (bednet)	**Total**
Ae. aegypti	0	9.1	0	0	0	0	1322.4	0	**1333.6**
Ae. albopictus	0	41.9	0	0	0	0	0	0	**41.9**
An. albimanus	0	11.6	0	0	0	0	0	0	**11.6**
An. arabiensis	0	0.1	0	0	0	0	14815.2	4118.2	**18933.6**
An. atroparvus	0	2.3	0	0	0	0	0	0	**2.4**
An. barbirostris	0	0	0	0	0	87.8	0	0	**87.8**
An. coluzzii	0	1.3	0	0	140.0	0	0	0	**141.3**
An. coustani	0	0	0	0	0	0	1140.6	0	**1140.6**
An. dirus	0	4.7	0	0	0	909.8	0	0	**914.5**
An. farauti	0	1.1	0	0	0	0	0	0	**1.1**
An. freeborni	0	6.3	0	0	0	0	0	0	**6.2**
An. funestus	0	2.1	0	0	0	0	0	0	**2.1**
An. funestus s.l.	0	0	0	0	0	0	1542.1	0	**1542.1**
An. funestus s.s.	0	0	0	0	0	0	7414.2	0	**7414.2**
An. gambiae	0	4.0	0	0	88.2	0	0	0	**92.2**
An. gambiae s.l.	0	0	0	0	0	0	406.7	0	**406.7**
An. gambiae s.s.	2474.2	0	0	0	0	0	0	0	**2474.2**
An. harrisoni	0	0	0	0	0	980.4	0	0	**980.4**
An. leesoni	0	0	0	0	0	0	43.5	0	**43.5**
An. maculatus	0	0	0	0	0	830.4	0	0	**830.4**
An. maculipalpis	0	0	0	0	0	0	2013.0	0	**2013.0**
An. merus	0	1.4	0	0	0	0	0	0	**1.4**
An. minimus	0	1.4	0	0	0	61.5	0	0	**63.0**
An. pharoensis	0	0	0	0	0	0	56.3	0	**56.3**
An. quadriannulatus	0	15.9	0	0	0	0	0	0	**15.9**
An. rivulorum	0	0	0	0	0	0	5.1	0	**5.1**
An. sinensis	0	1.0	0	0	0	0	0	0	**1.0**
An. squamosus	0	0	0	0	0	0	2091.8	0	**2091.8**
An. stephensi	0	6.7	0	0	63.1	0	0	0	**69.9**
An. ziemanni	0	0	0	0	0	0	110.0	0	**110.0**
Coquillettidia sp.	0	0	0	0	0	0	25.6	0	**25.6**
Culex pipiens	0	0	0	0	0	0	8157.8	0	**8157.8**
Culex q.	0	7.3	6573.1	898.1	0	0	0	0	**7478.5**
Culex tarsalis	0	10.5	0	0	0	0	0	0	**10.5**
Culex tigripes	0	0	0	0	0	0	158.7	0	**158.7**
Ma. africanus	0	0	0	0	0	0	785.2	0	**785.2**
Ma. uniformis	0	0	0	0	0	0	1654.5	0	**1654.6**
Toxorhynchites br.	0	0	0	0	84.6	0	0	0	**84.6**

C.2 Miscellaneous commands

To generate the metadata of Table 7, we include a list of commands used to generate one row for completeness.

Count total length of labelled audio for a certain path and sound type:

```
SELECT SUM(fine_end_time-fine_start_time)
FROM label
LEFT JOIN mosquito ON (label.mosquito_id = mosquito.id)
RIGHT JOIN audio ON (label.audio_id = audio.id)
RIGHT JOIN location ON (audio.loc_id = location.id)
WHERE path LIKE '%Thai%' and sound_type='mosquito';
```

Count number of audio files for a certain path and sound type:

```
SELECT COUNT (DISTINCT path)
FROM label
LEFT JOIN mosquito ON (label.mosquito_id = mosquito.id)
RIGHT JOIN audio ON (label.audio_id = audio.id)
RIGHT JOIN location ON (audio.loc_id = location.id)
WHERE path LIKE '%Thai%' and sound_type='mosquito';
```

Return location, device types, and recording methods for dataset:

```
SELECT DISTINCT country, location_type, method, mic_type, device_type
FROM audio
RIGHT JOIN location ON (audio.loc_id = location.id)
RIGHT JOIN device ON (audio.dev_id = device.id)
WHERE path LIKE '%Tanzania%';
```

D Datasheet for dataset

We follow the structure outlined in Datasheets for Datasets by Gebru et al. [2018]. D.1 gives the motivation for the data. D.2 describes the composition of the data. D.3 describes the collection process. D.4 describes the preprocessing involved in the data curation. D.5 lists past uses, and suggests a range of future uses in depth. D.6 describes potential sources of data bias and relevant mitigation strategies. Database maintenance policies are given in D.7.

D.1 Motivation

For what purpose was the dataset created? This dataset was created for academic research, and applications of machine learning for global health. One such application is the monitoring of deadly mosquito species from their acoustic signature, for which quality training data is required to capture the variation that species may exhibit.

Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)? This dataset was curated by the Machine Learning Research Group of the University of Oxford. Data was collected by the Department of Zoology, University of Oxford, the Centers for Disease Control and Prevention, Atlanta, the United States Army Medical Research Unit in Kenya (USAMRU-K), at the London School of Tropical Medicine and Hygiene, the Dept of Entomology, Kasesart University, Bangkok, and by the Ifakara Health Institute in Tanzania.

Who funded the creation of the dataset? A Google Impact Challenge Award 2014, The Bill and Melinda Gates Foundation (2019–present), available on `https://www.gatesfoundation.org/about/committed-grants/2019/07/opp1209888` (last accessed: June 2021).

D.2 Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)? This dataset is a collection of acoustic recordings in `wav` PCM format. We also supply all the metadata, generated in PostgreSQL to a `csv` file.

How many instances are there in total (of each type, if appropriate)? 9,295 `wav` audio files, 1 `csv`.

Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set? The audio files are a sub-sample of complete audio recordings, with the recordings corresponding to one complete label defined with a label ID, extracted from the original audio with the markers `start_time, end_time`. We are unable to release the full unlabelled audio due to potential issues with privacy and personally identifiable information. The metadata is a curated sub-sample of all available metadata, where fields which were not sufficiently populated or unverified are excluded.

What data does each instance consist of? Each instance corresponds to a labelled section of audio with the event times originally tagged in the original recording with a `start_time, end_time`, either manually by human domain experts, or by machine learning models. The label type is supplied in the metadata.

Is there a label or target associated with each instance? Yes, every recording matches a label.

Is any information missing from individual instances? Though every single sample has a label, some recordings have greater availability of metadata than others; see the metadata `csv` for details.

Are there recommended data splits (e.g., training, development/validation, testing)? Yes, see Table 7. The splits are carried out to increase the chance of generalisation to recordings conducted in varying conditions. The validation split is part of the challenge of this benchmark, left to the discretion of the users. The test data is automatically split in the supplied code.

Table 7: Key audio metadata and train-test partition. *'Wild'* mosquitoes captured and placed into paper *'cups'* or attracted by bait surrounded by *'bednets'*. *'Culture'* mosquitoes bred specifically for research. Total length (in seconds) of mosquito recordings per group given, with the availability of species meta-information in parentheses. Total length of corresponding non-mosquito recordings, with matching environments, given as *'Negative'*. Full metadata is given in Appendix C.

Data (mosquitoes)	Site (country)	Recorded in	Device (sample rate)	Mosquito (s) (with species)	Negative (s)
Train (wild)	Kasetsart (Thailand)	cup (2018)	Telinga (44.1 kHz)	9,306 (2,869)	7,896
Train (wild)	IHI (Tanzania)	cup (2020)	Telinga (44.1 kHz)	45,998 (45,998)	5,600
Train (culture)	Zoology (Oxford, UK)	cup (2017)	Telinga (44.1 kHz)	6,573 (6,573)	1,817
Train (culture)	LSTMH (UK)	cup (2018)	Telinga (44.1 kHz)	376 (376)	147
Train (culture)	CDC (USA)	cage (2016)	phone (8 kHz)	133 (127)	1,121
Train (culture)	USAMRU (Kenya)	cage (2016)	phone (8 kHz)	2,475 (2,475)	31,930
Test A (culture)	IHI (Tanzania)	bednet (2020)	phone 8 kHz	4,118 (4,118)	3,979
Test B (culture)	Zoology (Oxford, UK)	cage (2016)	phone (8 kHz)	737 (737)	2,307
All	All	All	All	71,286 (64,843)	53,227

Are there any errors, sources of noise, or redundancies in the dataset? To our knowledge there are no redundancies, duplicate files, corrupt files or unintended bugs. Despite comprehensive manual checks, label errors due to human entry and ambiguity in sound type may remain.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)? The data is self-contained, generated from a PostgreSQL database which is hosted on University of Oxford servers. The data itself is hosted on Zenodo, and the code is accessible on GitHub.

Does the dataset contain data that might be considered confidential? No, explicit permission was obtained where speech is present.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? The audio recordings of mosquitoes may cause distress or discomfort to individuals with medical issues that pertain to mosquito sound.

Does the dataset identify any subpopulations (e.g., by age, gender)? The metadata identifies subpopulations of species complexes by species, and further by gender, age and plurality type (for example, if there was more than one mosquito recorded at a label). Further discriminating factors are described in Appendix C.

Is it possible to identify individuals (i.e., one or more natural persons), either directly or indirectly (i.e., in combination with other data) from the dataset? Yes, the speakers may announce the recording ID at the start of a recording, however explicit consent was obtained. It may be possible to trace to the person conducting the experiment indirectly.

D.3 Collection Process

How was the data associated with each instance acquired? The data was collected globally at numerous research facilities. We summarise the data collection efforts below:

- **UK, Kenya, USA**: Recordings from laboratory cultures at the London School of Tropical Medicine and Hygiene (LSTMH), the United States Army Medical Research Unit-Kenya (USAMRU-K); Center for Diseases Control and Prevention (CDC), Atlanta as well as with mosquitoes raised from eggs at the Department of Zoology, University of Oxford. Mosquitoes were recorded by placing a recording device into the culture cages where one or multiple mosquitoes were flying, or by placing individual mosquitoes into large sample cups and holding these close to the recording devices.
- **Tanzania i)**: Mosquitoes recorded at Ifakara Health Institute's semi-field facility (*'Mosquito City'*) at Kining'ina. The facility houses six chambers containing purpose-built experimental huts, built using traditional methods and representing local housing constructions, with grass roofs, open eaves and brick walls. Four different configurations of the HumBug Net, each with a volunteer sleeping under the net, were set up in four chambers. Budget smartphones were placed in each of the four corners of the HumBug Net. Each night of the study, 200 laboratory cultured *An. arabiensis* were released into each of the four huts and the MozzWear app began recording.
- **Tanzania ii)** A collection and recording project in the Kilombero Valley, Tanzania. HBNs, larval collections and CDC-LTs were used to sample wild mosquitoes and record them in sample cups in the laboratory. *Anopheles gambiae* and *An. funestus* (another highly dangerous mosquito found across sub-Saharan Africa), are also siblings within their respective species complexes. Thus, standard PCR identification techniques [Scott et al., 1993] were used to fully identify mosquitoes from these groups. The Tanzanian sampling has collected 17 different species including: *An. arabiensis* (a member of the *gambiae* complex), *An. coluzzii*, *An. funestus*, *An. pharoensis* (see Appendix C, Figure 6 for a full breakdown).
- **Thailand**: Mosquitoes were sampled using ABNs, HBNs and larval collections over a period of two months during peak mosquito season (May to October 2018). Sampling was conducted in Pu Teuy Village (Sai Yok District, Kanchanaburi Province, Thailand) at a vector monitoring station owned by the Kasetsart University, Bangkok. The mosquito fauna at this site include a number of dominant vector species, including *An. dirus* and *An. minimus* alongside their siblings (*An. baimaii* and *An. harrisoni*) respectively (Appendix C, Figure 6 gives a species histogram for this dataset). Sampling ran from 6 pm to 6 am, as most anopheline vectors prefer to bite during the night. Mosquitoes were collected at night, carefully placed into large sample cups and recorded the following day using the high-spec Telinga field microphone and a budget smartphone.

What mechanisms or procedures were used to collect the data (e.g., hardware apparatus or sensor, manual human curation, software program, software API)? A summary of equipment is as follows:

- Smartphone (Itel, Alcatel, and others) audio recording with the MozzWear application. Smartphone devices may have variable sample rates, as denoted by the sample rate column of the metadata. The version of MozzWear used in the curation of this dataset recorded audio in 8,000 Hz mono wave format.
- Telinga EM-23 field microphone, and Tascam, Olympus recording devices recording at 44,100 Hz. The Telinga is a very sensitive, low-noise microphone which was widely adopted in bioacoustic studies.
- Human labelling with Excel.
- Human labelling with Audacity (GNU GPLv2 license).
- Labels produced by a Bayesian convolutional neural network (our own, MIT license, included in paper).
- Voice activity detection and removal with WebRTC (BSD license).

- Python (BSD-style license), MongoDB (Server Side Public License), Django (BSD license), Apache (GPLv3 license), PostgreSQL (BSD/MIT-like license), Unix for databases, HTML dashboards, and post-processing.

Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)? Researchers from the locations previously mentioned, paid salary from their respective institutions, through the grants disclosed previously.

Over what timeframe was the data collected? 2015 to 2020 (and ongoing).

Were any ethical review processes conducted (e.g., by an institutional review board)? We have obtained the ethics approval from the following committees:

- Oxford Tropical Research Ethics Committee (OxTREC Ref. 548-19) – University of Oxford (UK).
- Ifakara Health Institute (IHI)-IRB – Tanzania
- National Institute for Medical Research – Tanzania
- School of Public Health at the University of Kinshasa (KSPH) – DRC

D.4 Preprocessing/cleaning/labeling

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)? The data underwent rigorous curation, from manual adjustment to labels supplied in text files, to commands in the database to deal with incorrectly entered label times resulting in missing data. To encourage reproducibility and compatibility for future data release, all the label and audio quality control is performed before uploading to the database, and within the dataset itself.

Example of quality control code to check that the label end does not exceed the length (which happens frequently when labels are entered by hand into Audacity with end times longer than the recording and then exported to a text file):

```
SELECT path, fine_start_time, fine_end_time, sound_type, length
FROM label
LEFT JOIN mosquito ON (label.mosquito_id = mosquito.id)
RIGHT JOIN audio ON (label.audio_id = audio.id)
RIGHT JOIN location ON (audio.loc_id = location.id)
WHERE fine_end_time > length;
```

Sources with low estimated label quality were either removed or manually re-labelled and amended in the database.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)? Yes, all data that may have future utility (and has not been yet used for that purpose) has been released. Unprocessed, and currently unlabelled data is also all stored on the database server, but requires further curation and data entry to the specific data tables before release. We plan to periodically update the database as more data becomes available.

Is the software used to preprocess/clean/label the instances available? The software to do so included Audacity, PostgreSQL, Python, Excel, and is available and well-maintained. We will make use of it in future for future data curation.

D.5 Uses

Has the dataset been used for any tasks already? A subset of this data (recorded in Thailand, Kenya, UK, USA) has been used to train a machine learning model to distinguish and detect a mosquito from its acoustic signature. The model was a 4-layer Bayesian convolutional neural network implemented in Keras. The predictive entropy and mutual information were used to screen predictions over thousands of hours of data. Hand labels were added to correct predictions, and the labels were fed back into the database [Kiskin et al., 2021]. Code for the training and resulting predictive pipeline is available on `https://github.com/HumBug-Mosquito/MozzBNN`.

Other past use cases and publications can be found in related works from the link of the following section. We summarise these here as:

- Bioacoustic classification with wavelet-conditioned neural networks [Kiskin et al., 2017, 2018].
- Cost-sensitive mosquito detection [Li et al., 2017a]
- A case study of species classification with field recordings [Li et al., 2018]
- A release of a subset of this database for crowdsourcing (with baseline mosquito detector model) [Kiskin et al., 2019, 2020]

Is there a repository that links to any or all papers or systems that use the dataset? Yes, the Zenodo data directory `https://zenodo.org/record/4904800` contains all the references to projects, papers and code which are associated with this dataset.

What (other) tasks could the dataset be used for? A list of use cases is not limited to, but may include:

1. **Species classification with HumBugDB.** Training a machine learning model to distinguish between various species.
2. **Validating species classification models from the literature.**
3. **Frequency analysis.** Identifying the fundamental and harmonic frequencies of flight tone for a particular species, to improve upon the understanding of bioacoustics literature, and entomological research.
4. **Examining inter-species (or similar) variability.** For example, the effect on the sound of flight as a result of age, gender, or any field supported in the database.

We now expand upon each point:

1. **Species classification with HumBugDB.** There a multitude of mosquito species, however only certain species have potential to transmit certain pathogens. It is imperative therefore to accurately locate and identify the few dangerous mosquito species amongst the many benign ones to achieve efficient mosquito control. Dangerous species include primary malaria vectors (e.g. *Anopheles gambiae*, *An. arabiensis*), arbovirus vectors including primary vectors of dengue virus (*Aedes albopictus*), yellow fever virus (*Aedes aegypti*) and west nile virus (*Culex quinquefasciatus*). The species available and their associated occurrence per experimental group is given in Figures 6 and Table 6 in Appendix C.

   When designing experiments, it would be useful to take into account the method of capture and whether mosquitoes are from lab cultures or captured as individuals in the wild. The experimental groups are given in Table 7, and further details can be found in the database metadata as detailed in Appendix C. As a starting point, we recommend using the Tanzanian cup recordings of wild individuals, denoted by the metadata `country = Tanzania`, `location_type = field`. Each recording (denoted by the `name` metadata field) is of an individual mosquito. You may partition the train/validation/test splits by recording name, to ensure appropriately disjoint subsets (avoiding repetition of individuals when validating or testing models).

   A very useful property of this dataset is that there is overlap in species recorded in entirely different experimental conditions (Table 6). This gives the opportunity to test the robustness of the model when generalising across datasets.
2. **Validating species classification models from the literature.** As a result of procuring curated data with species meta-information of both wild and lab mosquitoes, this dataset serves as an ideal test-bed to verify the effectiveness of existing species classification

approaches. We encourage researchers to validate their models by making use of these data to form their own test sets without re-training their models on any parts of this dataset. Strong species discrimination performance would signify a great opportunity to utilise acoustics as a wide-scale surveillance tool. If you encounter any issues, or require further information do not hesitate to contact the database maintainers (Appendix D.7).

3. **Frequency analysis.** Earlier works in the literature proposed more hand-crafted approaches to building detection or classification models. These may be especially useful in very low-power embedded devices which require fast and efficient algorithms. These approaches were typically centered around specific harmonic inter-peak ratios (See Kiskin [2020, Sec. 3.2] for an overview of relevant prior work). Frequency analysis may be performed on any parts of this dataset, including on species which are under-represented. In particular, the CDC dataset contains a wide range of unique species which are sparsely labelled, however the labelled sections have very high signal-to-noise ratio. As with previous suggested use cases, we recommend trialling approaches on disjoint sets of experiments (or at the very least individual mosquito recording within an experimental set). Once again, there exists an excellent opportunity to validate models from the literature on their ability to distinguish species on this dataset.

4. **Examining the effect of species variability on their flight tone.** It is well known that mosquitoes exhibit significant variability in their physical (and therefore acoustic) properties within a species. These occur due to a multitude of factors including the age, wingspan, gender. Additionally, confounding factors such as the temperature, humidity, and potentially their fed status, can increase the difficulty in distinguishing individuals within and across species. As we maintain as much metadata as possible, this dataset provides the opportunity to examine such factors. In future releases, temperature and humidity will be added where possible, and this data is expected to be available in an update on the Tanzanian cup recordings which has already good metadata coverage including `species`, `age`, `gender`, `fed`, `method`. If you wish to have early access to additional metadata, please contact us and we will make the availability of such metadata a higher priority.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? No, the dataset is specifically organised in PostgreSQL in a way to be consistent with future release. However, in future more metadata may become available for legacy datasets, and larger subsets may become available upon addition of labels.

Are there tasks for which the dataset should not be used? No.

D.6 Data bias

Data is collected with varying recording paradigms, and is sampling a broad (and not fully understood) population of mosquitoes. This induces inherent biases which may affect an algorithm's performance when acting as either a mosquito detector or species discriminator. We attempted to capture biases as well as possible with comprehensive metadata coverage, which we encourage users to explore for their own use cases. We discuss potential sources of bias and suggest mitigation strategies as follows:

1. **Nature of mosquitoes: lab or wild**. These are denoted by `location_type`. The controlled conditions of laboratory cultures produce uniformly sized fully-developed adult mosquitoes which are used for a variety of purposes, including trialling new insecticides or examining the genome of these insects. Models trained on purely lab cultures run the risk of overfitting to this artificial subpopulation, encountering difficulty when generalising even to the same species. Wild mosquitoes on the other hand exhibit greater variation, at the cost of a much more laborious collection procedure. When constructing models, it is advised to train on wild data, but caution needs to be taken when testing on mosquitoes from uniform subpopulations.

2. **Recording device** corresponding to metadata from `device`, It is crucial that datasets are not constructed in a way where one device is used for only positive or negative instances (e.g. all noise is from one device, and all mosquito from another). If trained in such a manner, it will be easy for a high-dimensional model such as a neural network to learn the characteristics of

the microphone response and use this confounding factor for classification. To mitigate this, we have included a negative control group for each experiment, and therefore also device. This issue becomes especially critical for species classification, where different species may be captured with different devices. Careful consideration and construction of data with the use of the device metadata will help avoid, or at the very least alert to possible confounding. If it is not possible to control the device, it may be desirable to use features which are (more) invariant to microphone type, e.g. MFCCs or high-level pre-trained feature representations such as VGGish embeddings [Gemmeke et al., 2017].

3. **Data imbalance.** Biased models for either species classification or mosquito detection may arise when trained naively without balancing distributions of species, or positive and negative samples. In the case of mosquito detection, a predominance of one species will likely increase the model's ability to detect mosquitoes of that particular species, while performing worse on less well-represented groups. This is a potential source of improvement worth investigating, especially for the data split suggested in Table 7. Additionally, a closer look at species-specific performance may reveal areas for further model improvement. We recommend benchmarking against the baselines supplied to investigate areas of improvement.

   If using multi-class classifiers, it is possible to begin by weighting class samples by the inverse of their frequency. There are however well-known drawbacks to this. Bayesian models which take into account asymmetrical cost functions aim to alleviate this problem [Cobb et al., 2018]. A further option is to use different step functions in the data partitioning/augmentation pipelines. A starting point would be to modify `step_size` in `feat_util.py` in a class-specific function, to artificially balance the relative frequency of data samples of desired classes.

D.7 Database maintenance

Who is supporting/hosting/maintaining the dataset? Please contact Dr. Ivan Kiskin at `ivankiskin1@gmail.com`, who is maintaining the dataset. Alternative contacts include Professor Steve Roberts at `sjrob@robots.ox.ac.uk` at the University of Oxford Machine Learning Research Group.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)? The data will be updated as new data from new trials is obtained and curated. We expect updates to occur every few months in 2021. Updates will be communicated by commits and pushes to the GitHub repository. Please also follow the link on Zenodo for versioning details, where older versions will continue to be hosted.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? If so, please provide a description. If you would like to contribute to this data, please contact the database host and supervising professor. We would be happy to curate data and provide requirements which would help qualify a dataset for hosting. All contributions will be credited appropriately in future work.

References

H. Ali. Real-time Communication Using WebRTC. Technical report, Georgia Institute of Technology, 2018.

R. J. Bomphrey, T. Nakata, N. Phillips, and S. M. Walker. Smart wing rotation and trailing-edge vortices enable high frequency mosquito flight. *Nature*, 544(7648):92–95, 2017.

Y. Chen, A. Why, G. Batista, A. Mafra-Neto, and E. Keogh. Flying insect classification with inexpensive sensors. *Journal of Insect Behavior*, 27(5):657–677, 2014.

F. Chollet et al. Keras, 2015. URL https://keras.io. Accessed: 2018-06-07.

A. D. Cobb, S. J. Roberts, and Y. Gal. Loss-calibrated approximate inference in Bayesian neural networks. *arXiv preprint arXiv:1805.03901*, 2018.

E. Fanioudakis, M. Geismar, and I. Potamitis. Mosquito wingbeat analysis and classification using deep learning. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2410–2414, 2018.

N. Friederici, S. Ojanperä, and M. Graham. The impact of connectivity in Africa: Grand visions and the mirage of inclusive digital development. *The Electronic Journal of Information Systems in Developing Countries*, 79(1):1–20, 2017.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford. Datasheets for datasets. *arXiv preprint arXiv:1803.09010*, 2018.

J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: an ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

GSMA. The mobile economy-sub-saharan africa, 2020. URL https://www.gsma.com/mobileeconomy/sub-saharan-africa/. Last accessed: 2021-07-08.

R. Harbach. Mosquito taxonomic inventory, 2013. URL http://mosquito-taxonomic-inventory.info/. Last accessed: 2021-06-07.

S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135. IEEE, 2017.

A. A. Hoffmann and P. A. Ross. Rates and Patterns of Laboratory Adaptation in (Mostly) Insects. *Journal of Economic Entomology*, 111(2):501–509, 03 2018. ISSN 0022-0493. doi: 10.1093/jee/toy024. URL https://doi.org/10.1093/jee/toy024.

N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

B. Huho, K. Ng'habi, G. Killeen, G. Nkwengulila, B. Knols, and H. M. Ferguson. Nature beats nurture: a case study of the physiological fitness of free-living and laboratory-reared male Anopheles gambiae sl. *Journal of Experimental Biology*, 210(16):2939–2947, 2007.

HumBug. The HumBug Project, 2021. URL https://humbug.ox.ac.uk/. Accessed: 2021-06-21.

S. Jakhete, S. Allan, and R. Mankin. Wingbeat frequency-sweep and visual stimuli for trapping male Aedes aegypti (Diptera: Culicidae). *Journal of medical entomology*, 54(5):1415–1419, 2017.

B. J. Johnson and S. A. Ritchie. The siren's song: exploitation of female flight tones to passively capture male Aedes aegypti (Diptera: Culicidae). *Journal of medical entomology*, 53(1):245–248, 2016.

R. Karrer. Google WebRTC Voice Activity Detection module, 2020. URL https://github.com/rafaelkarrer/mex-webrtcvad/releases/tag/v0.1. Accessed: 2021-06-05.

I. Kiskin. *Machine learning for acoustic mosquito detection*. PhD thesis, University of Oxford, 2020.

I. Kiskin, B. P. Orozco, T. Windebank, D. Zilli, M. Sinka, K. Willis, and S. Roberts. Mosquito detection with neural networks: the buzz of deep learning. *arXiv preprint arXiv:1705.05180*, 2017.

I. Kiskin, D. Zilli, Y. Li, M. Sinka, K. Willis, and S. Roberts. Bioacoustic detection with wavelet-conditioned convolutional neural networks. *Neural Computing and Applications: Special Issue on Deep Learning for Music and Audio*, Aug 2018. ISSN 1433-3058.

I. Kiskin, U. Meepegama, and S. Roberts. Super-resolution of time-series labels for bootstrapped event detection. *Time-series Workshop at the International Conference on Machine Learning*, 2019.

I. Kiskin, L. Wang, A. Cobb, et al. Humbug Zooniverse: a crowd-sourced acoustic mosquito dataset. *International Conference on Acoustics, Speech, and Signal Processing 2020, NeurIPS Machine Learning for the Developing World Workshop 2019*, 2019, 2020.

I. Kiskin, A. D. Cobb, M. Sinka, and S. J. Roberts. Automatic acoustic mosquito tagging with Bayesian neural networks. *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2021.

Y. Li, I. Kiskin, D. Zilli, M. Sinka, H. Chan, K. Willis, and S. Roberts. Cost-sensitive detection with variational autoencoders for environmental acoustic sensing. *NeurIPS Workshop on Machine Learning for Audio Signal Processing*, 2017a.

Y. Li, D. Zilli, H. Chan, I. Kiskin, M. Sinka, S. Roberts, and K. Willis. Mosquito detection with low-cost smartphones: data acquisition for malaria research. *NeurIPS Workshop on Machine Learning for the Developing World*, 2017b.

Y. Li, I. Kiskin, M. Sinka, D. Zilli, H. Chan, E. Herreros-Moya, T. Chareonviriyaphap, R. Tisgratog, K. Willis, and S. Roberts. Fast mosquito acoustic detection with field cup recordings: an initial investigation. *Detection and Classification of Acoustic Scenes and Events*, 2018.

T. Marinos, S. Lin, D. Zilli, and H. Chan. MozzWear, 2021. URL https://github.com/HumBug-Mosquito/MozzWear. Pending update on Google Play store, GitHub private, accessed: 2021-06-05.

MongoDB Inc. Mongodb, 2021. URL https://www.mongodb.com/. Accessed: 2021-06-05.

H. Mukundarajan, F. J. H. Hol, E. A. Castillo, C. Newby, and M. Prakash. Using mobile phones as acoustic sensors for high-throughput mosquito surveillance. *eLife*, 6:e27854, Oct 2017. ISSN 2050-084X.

K. Palanisamy, D. Singhania, and A. Yao. Rethinking CNN models for audio classification. *arXiv preprint arXiv:2007.11154*, 2020.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

V. P. Perevozkin and S. S. Bondarchuk. Species specificity of acoustic signals of malarial mosquitoes of anopheles maculipennis complex. *International Journal of Mosquito Research*, 2(3):150–155, 2015.

J. Pons and X. Serra. musicnn: Pre-trained convolutional neural networks for music audio tagging. *arXiv preprint arXiv:1909.06654*, 2019.

J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra. End-to-end learning for music audio tagging at scale. *arXiv preprint arXiv:1711.02520*, 2017.

PostgreSQL Global Development Group. PostgreSQL, 2021. URL https://www.postgresql.org/docs/9.3/app-psql.html. Accessed: 2021-06-05.

J. Ramirez, J. M. Górriz, and J. C. Segura. Voice activity detection. fundamentals and speech recognition system robustness. *Robust speech recognition and understanding*, 6(9):1–22, 2007.

A. Sahoo. Voice activity detection for low-resource settings. *Department of Electrical Engineering, Stanford University*, 2020.

J. A. Scott, W. G. Brogdon, and F. H. Collins. Identification of single specimens of the anopheles gambiae complex by the polymerase chain reaction. *The American journal of tropical medicine and hygiene*, 49(4):520–529, 1993.

K. Shimada, N. Takahashi, S. Takahashi, and Y. Mitsufuji. Sound event localization and detection using activity-coupled cartesian doa vector and rd3net. Technical report, DCASE2020 Challenge, July 2020.

P. M. Simões, R. A. Ingham, G. Gibson, and I. J. Russell. A role for acoustic distortion in novel rapid frequency modulation behaviour in free-flying male mosquitoes. *Journal of Experimental Biology*, 219(13):2039–2047, 2016.

M. Sinka, D. Zilli, I. Kiskin, Y. Li, D. Kirkham, W. Rafique, H. Chan, B. Gutteridge, E. Herreros-Moya, H. Portwood, S. J. Roberts, and K. J. Willis. HumBug – An Acoustic Mosquito Monitoring Tool for use on budget smartphones. *Methods in Ecology and Evolution*, 2021. doi: 10.1111/2041-210X.13663.

M. E. Sinka, M. J. Bangs, S. Manguin, Y. Rubio-Palis, T. Chareonviriyaphap, M. Coetzee, C. M. Mbogo, J. Hemingway, A. P. Patil, W. H. Temperley, et al. A global map of dominant malaria vectors. *Parasites & vectors*, 5(1):1–11, 2012.

D. Vasconcelos, N. J. Nunes, and J. Gomes. An annotated dataset of bioacoustic sensing and features of mosquitoes. *Scientific Data*, 7(1):1–8, 2020.

World Bank Organisation. Listening to Africa, 2017. URL https://www.worldbank.org/en/programs/listening-to-africa. Last accessed: 2021-07-08.

World Health Organization. Fact Sheet, 2020. URL https://www.who.int/news-room/fact-sheets/detail/vector-borne-diseases. Accessed: 2020-01-26.