# Bird Identification from Timestamped, Geotagged Audio Recordings

Jan Schlüter

Austrian Research Institute for Artificial Intelligence, Vienna, Austria
jan.schlueter@ofai.at

**Abstract.** Large-scale biodiversity monitoring would profit from automated solutions supporting or complementing human experts and citizen scientists. To foster their development, the yearly BirdCLEF scientific challenge compares approaches for identifying bird species in recorded vocalizations. The solution described in this work is based on an ensemble of Convolutional Neural Networks (CNNs) processing a mel spectrogram combined with Multi-Layer Perceptrons (MLPs) processing the recording date, time and geographic location. In BirdCLEF 2018, it achieved a mean average precision of 0.705 in detecting 1,500 South American bird species (0.785 for the foreground species), the second best entry to the challenge.

**Keywords:** Bioacoustics · Bird vocalizations · BirdCLEF 2018 · Deep Learning · Convolutional Neural Networks · Spectrograms · Metadata

**Source code:** https://github.com/f0k/birdclef2018



## 1 Introduction

Biodiversity monitoring is important to assess the impact of human actions on other species, and inform decisions on short-term projects and long-term policies. However, manually observing, classifying and counting animals is limited in scale and scope by the supply of experts, the costs of human labor and basic human needs. Automated or semi-automated approaches for recording and analyzing observations would allow monitoring more locations, over longer time spans and in forbidding terrain.

For many animals, acoustic monitoring is an interesting option – it allows to detect and classify individuals even when they are difficult to see, which is precisely the reason for some of their vocalizations. Acoustic recording devices paired with automatic classifiers could support both experts in the field in identifying species, as well as enable passive acoustic monitoring.

The CLEF (Conference and Labs of the Evaluation Forum) initiative fosters the development of automatic classifiers for the case of bird vocalizations through the yearly BirdCLEF challenge [13,8]. In 2018, it provided 36,496 recordings of 1,500 South American bird species to develop classifiers on, and challenged participants in two tasks: (1) Recognizing species in 12,347 recordings mostly

captured with monodirectional microphones aimed at individual birds, and (2) recognizing species at 5-second intervals in 51 long-term recordings taken with multidirectional recording devices.

In the following, I describe my submission to BirdCLEF 2018, which reached the second place among six participating teams. Section 2 details the layout and training of a Convolutional Neural Network (CNN) processing the audio signals, Section 3 describes a Multilayer Perceptron (MLP) predicting the species from the metadata of each recording, and Section 4 explains how the predictions of multiple audio and metadata networks are combined into a final result. Section 5 compares results on an internal validation set and the official test sets. Finally, Section 6 discusses ideas that did not turn out successful, and Section 7 concludes with a summary and an outlook.

## 2 Predictions from Audio

The foundation of my submission is an ensemble of Convolutional Neural Networks (CNNs) that process a mel spectrogram to produce occurrence probabilities for each of the 1500 possible bird species. In the following, I describe how the audio signal is preprocessed to be consumed by the network, what network architectures were chosen, how the networks were trained, and how they are applied at test time. Both for the architecture and for training I used multiple variants which were later combined in an ensemble (Section 4).

### 2.1 Spectrogram Extraction

Adopted from [21,9], I first resample the audio signal to 22.05 kHz sample rate, compute an STFT with a Hann window of 1024 samples and hop size of 315 samples (obtaining 70 frames per second), retain only the magnitudes, and apply a mel filterbank of 80 triangular filters from 27.5 Hz to 10 kHz. These settings were chosen as a starting point since they had proved to work well for bird detection, and were not varied much during the experiments: reducing the upper bound to 8 kHz diminished results, increasing it to 11 kHz did not have a significant effect, a window size of 2048 samples (for better low-frequency resolution) did not help. Note that the magnitudes are linear; they will be compressed in the network.

### 2.2 Network Architecture

The network architecture is a CNN based on my submission to a bird detection challenge (submission *sparrow* in [9]), but modified to handle classification rather than detection, and varied in different ways both in search of a better solution and to obtain multiple variants for an ensemble.

Figure 1 gives an overview of the architecture variants. The overarching idea is to have a fully-convolutional network – i.e., a network with only convolutional and pooling layers, no fixed-size fully-connected layers – process an input spectrogram into a sequence of local predictions for each bird species, followed by

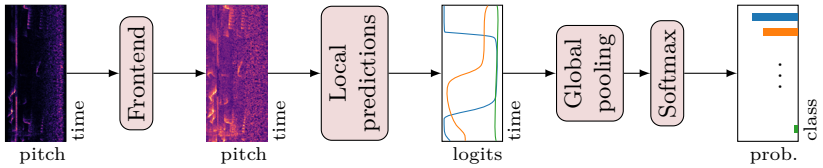


**Fig. 1.** Outline of the audio network architecture.

a global pooling strategy that produces a single prediction per species for the full recording. This type of architecture can be directly trained and tested on recordings of arbitrary length and does not require a hand-chosen averaging or majority voting procedure to produce recording-wise predictions at test time.

In the following, I will describe a common preprocessing frontend as well as the local prediction and global prediction stages in detail.

**Frontend:** The first step is a learned nonlinear magnitude transformation:

$$y = x^{\sigma(a)}, \quad \text{where } \sigma(a) = 1/\left(1 + \exp(-a)\right) \tag{1}$$

It is applied elementwise to each time-frequency bin $x$, enhancing low magnitudes similar to the more common application of a logarithmic function. $a$ is initialized to zero (so $y = \sqrt{x}$) and learned as part of the network, shared over all mel bands, typically reaching values between -1.2 and -1.7 (yielding $y = x^{0.23}$ to $y = x^{0.15}$).

This is followed by batch normalization [12] with means and standard deviations shared over all time steps, but separated per mel band, and without learned scale and shift ($\gamma$ and $\beta$ in [12]). At test time, this is comparable to standardizing each mel band to zero mean and unit standard deviation over the training set (as in [21]), but at training time, it adds seemingly helpful noise through computing the statistics for the current minibatch only, and it dynamically adapts the standardization to any changes in the learned magnitude transformation.

**Local predictions:** The frontend is followed by one of two variants for computing a sequence of local predictions from the input spectrogram.

**A)** The first variant is based on submission *sparrow* in [9], but enlarged and adapted to perform classification rather than detection. It starts with two convolutional layers of 64 3×3 units each, followed by non-overlapping max-pooling of 3×3, and two more convolutional layers of 128 3×3 units each. All convolutions are unpadded ("valid"). At this point, the temporal resolution has been reduced to a third, and there are 21 frequency bands remaining. I apply a convolutional layer of 128 3×17 units, leaving 5 frequency bands, followed by 3×5 max-pooling, squashing all frequency bands. This combination of convolution and pooling is meant to fuse information over all frequency bands while introducing some pitch invariance, and has proven helpful in [20,9]. This is followed by a convolutional layer of 1024 9×1 units, fusing information over 103 original spectrogram frames (1.5 s), finally followed by 1024 1×1 and 1500 1×1 units for classification (the last three layers resemble what would be fully-connected layers in a fixed-input-size CNN, but here they apply to an arbitrarily long input sequence).

All layers except the last one are followed by batch normalization (the usual variant with statistics shared over all spatial locations, but separated per channel, and with learned scale and shift) and leaky rectification as $\max(x/100, x)$. The last three layers are preceded by 50% dropout [11]. The very last layer neither has batch normalization nor a nonlinearity. While its 1500 features will lead to predictions for the 1500 bird species, the nonlinearity will be applied after global pooling, for reasons discussed below.

**B)** As a variation on this architecture, I replaced the first four convolutional layers with two residual blocks with pre-activation following [10, Fig. 1b]. Each residual block consists of four 3×3 convolutional layers, each of which is preceded by batch normalization and linear rectification as $\max(x, 0)$ (except for the initial convolution following the preprocessing frontend, where pre-activation would be redundant). Convolution inputs are padded with a leading and trailing row and column of zeros so the output size matches the input size. Skip connections add the input of each pair of two convolutions to its output. If needed, the skip connection includes a $1 \times 1$ convolutional layer to adjust the number of channels. The first residual block has 64 channels throughout and is followed by 3×3 max-pooling, the second residual block has 128 channels and is followed by batch normalization, linear rectification and an unpadded convolutional layer of 128 units leaving 5 frequency bands (compared to the first CNN variant, this requires $3 \times 22$ units instead of $3 \times 17$ units since the convolutions are padded). 3×5 max-pooling and remaining layers are adopted from the first CNN variant.

**Global predictions:** The previous stage produces a sequence of vectors of size 1500, corresponding to the 1500 possible bird species. Specifically, since the network includes two pooling operations over 3 frames each, it produces a local prediction every 9 frames,[1] for the part of the input spectrogram exceeding the length of the network's receptive field (103 frames for the first, 119 for the second variant) minus padding (none for the first, 32 frames for the second variant).

**A)** Treating this as a multiple-instance learning (MIL) problem [6,1] – for each recording and species, the global occurrence label should be positive if at least one of the local occurrence labels is positive –, the natural choice is to form a global prediction by taking the maximum over time for each species. However, this leads to very sparse gradients. Instead, I use *log-mean-exp* pooling [19, Eq. 6]:

$$\mathrm{lme}(\boldsymbol{y}; a) = \frac{1}{a} \log \left( \frac{1}{T} \sum_{t=0}^{T-1} \exp(a \cdot y_t) \right) \tag{2}$$

Here, $\boldsymbol{y}$ denotes the time series of local predictions $(y_0, y_1, \ldots, y_{T-1})$ for a single species, and $a$ is a hyperparameter controlling the behaviour of the function: for $a \to \infty$, it approximates the maximum, for $a \to 0$, it approximates the mean. I set $a = 1$ ($a = 0.2$ or $a = 5$ work just as well, I did not explore it further).

[1] It would be possible to obtain a prediction for every single frame by introducing dilated convolutions and pooling as described in [22], but this increases computational costs without a strong benefit.

**B)** As an alternative to a fixed pooling strategy that prefers larger inputs, I explored pooling with temporal attention. I let the network produce a weight $\alpha_t$ for each time step, and compute a weighted sum:

$$\mathrm{att}(\boldsymbol{y}, \boldsymbol{\alpha}) = \sum_{t=0}^{T-1} \alpha_t y_t = \boldsymbol{\alpha}^T \boldsymbol{y} \tag{3}$$

The $\alpha_t$ are shared over all species, and $\boldsymbol{\alpha}$ is produced with *multi-head attention*: I extend the last convolutional layer of the local prediction stage by $K$ units, such that it produces $1500 + K$ time series. I split off the $K$ additional time series $\boldsymbol{C}$ and apply a softmax over time to each one:

$$B_{k,t} = \exp(C_{k,t}) / \sum_{i=0}^{T-1} \exp(C_{k,i}) \tag{4}$$

Finally, I average the $K$ soft-maxed time series $\boldsymbol{B}$ to obtain $\boldsymbol{\alpha}$:

$$\alpha_t = \frac{1}{K} \sum_{k=0}^{K-1} B_{k,t} \tag{5}$$

By construction, $\boldsymbol{\alpha}$ sums to 1.0 regardless of the sequence length. Using multiple heads (I chose $K = 16$ or $K = 64$) makes it easier for the network to produce attention series with multiple local maxima. Note that there is no additional supervision needed to learn what to attend to; the global loss will be backpropagated through (3), (5), (4) to reach the last layer of the local prediction stage.

For both log-mean-exp pooling and attentional pooling, we now end up with a single vector of 1500 values. Assuming that we want to predict the occurrence of each species independently, the natural choice would be to apply the logistic sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$ to squash each value into the range $(0, 1)$ and treat it as a binary prediction. However, since the training examples strongly focus on capturing a single prominent species per recording, it turns out to work slightly better to treat this as a categorical classification problem, so I apply the softmax function to produce a distribution over 1500 classes instead.

Note that when planning to train with categorical cross-entropy loss, it is crucial to apply the softmax *after* the global temporal pooling operation, otherwise the competition between classes is introduced at the wrong level. Considering the case of global temporal max pooling, having the softmax last ensures the loss gradient reaches the frame of maximal activation for each species. Having the softmax first (i.e., on each local prediction) would propagate the gradient to the frame where the correct species is most dominant over the others, and then push down competing species for that frame only. Similar considerations apply to log-mean-exp pooling and attentional pooling.

### 2.3 Training

To deal with the fact that training recordings are usually a few minutes long, but only sparsely populated with vocalizations of the target bird (without any

information on where in a recording the target bird is audible), several authors employed hand-designed methods to find potential bird calls and trained on these passages only [17,23,14,5]. Here I take a different route: I randomly choose extracts long enough to hopefully contain at least one vocalization of every annotated bird for a recording, and directly train on those, relying on the global pooling operation to distribute the gradient to the relevant local predictions.

**Optimization:** The network is trained on mini-batches of 16 excerpts of up to 30 seconds to minimize categorical cross-entropy between network output and targets (see below for three variants of forming targets). 10% of the recordings are reserved as a validation set, selected to contain at least one recording per species (the same split as [14] through personal communication). Weights are updated with ADAM [15], with an initial learning rate of 0.001 dropped to a tenth whenever the validation loss did not improve for 10 consecutive mini-epochs of 1000 updates each. Training is stopped at the third drop. The learning rate for the magnitude compression parameter $a$ (Equation 1) is kept 10 times as high throughout training, otherwise it learns too slowly.

**Loss and targets:** Optimization minimizes the cross-entropy loss between predictions $\boldsymbol{y}$ and targets $\boldsymbol{t}$:

$$\mathrm{ce}(\boldsymbol{y}, \boldsymbol{t}) = -\sum_{s=0}^{S-1} t_s \log(y_s) \tag{6}$$

I employ three different variants for defining the target vector $\boldsymbol{t}$ for a recording.

**A)** The most obvious choice for a categorical classification task is to set $t_s$ to 1 for the annotated foreground species for the recording, and to 0 otherwise. Denoting the foreground species as $S_f$, we have:

$$t_s = \begin{cases} 1 & \text{if } s = S_f \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

**B)** To factor in the annotated background species, I set $t_s$ to 2 for the foreground species, to 1 for any of the background species, and to 0 otherwise. Denoting the set of background species as $S_b$, we have:

$$t_s = \begin{cases} 2 & \text{if } s = S_f \\ 1 & \text{if } s \in S_b \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Obviously, the network can never reach this target. However, as categorical cross-entropy (Equation 6) is linear in the targets, this is equivalent to adding the cross-entropy losses against each background species and the foreground species, weighting the latter twice. Furthermore, since ADAM is invariant to the scale of the gradient, it is unimportant if we set the foreground target to 2 and background to 1, or the foreground to 1 and background to 0.5, for example.

**C)** As a third variant, I train a network as a Born-Again Network (BAN), following [7]. Using a trained model, I compute predictions $\hat{\boldsymbol{y}}$ for the excerpt encountered during training, and set the target as follows:

$$t_s = \begin{cases} \hat{y}_t + 1 & \text{if } s = S_f \\ \hat{y}_t & \text{otherwise} \end{cases} \tag{9}$$

Again, note that this is equivalent to adding the loss against the target of variant A to the loss against $\hat{\boldsymbol{y}}$ (or averaging the two losses, if training with ADAM). Even if the model computing $\hat{\boldsymbol{y}}$ has the same architecture as the new model to be trained, this provides a helpful regularization, leading to faster convergence and possibly improved generalization (or an additional model for ensembling). Of course the same modification is possible for variant B (omitted for brevity).

**Length bucketing:** While aiming to train on 30-second excerpts (assuming that even for long recordings, a random 30-second excerpt will often contain vocalizations of all annotated birds), about 60% of the recordings are actually shorter than 30 seconds. For those, we can just take the full recording. To avoid wasted computation, I ensure to collect mini-batches of similarly-sized recordings: All recordings between 3 s and 30 s length are ordered by length and split into 8 same-sized buckets; recordings shorter than 3 s and longer than 30 s are sorted into 2 additional buckets. When preparing a mini-batch, a random bucket is chosen with probability proportional to its size, and 16 recordings within are selected uniformly at random. Recordings shorter than 3 s are looped to reach 3 s, recordings longer than 30 s are randomly cropped to 30 s, other recordings are randomly cropped to match the length of the shortest among the 16. This produces mini-batches of same-sized excerpts without excessive cropping or padding.

**Pipeline:** Before training, all recordings are transformed to monophonic signals at 22.05 kHz sample rate and written to fast persistent storage. During training, multiple background threads read excerpts of these signals and apply an STFT to prepare mini-batches of magnitude spectrograms. The mel filterbank is applied on GPU as part of the network.

Mel spectrograms are not precomputed before training to enable easy experimentation with different mel spectrogram settings (and learned mel filterbanks). STFTs are not precomputed because that would increase I/O costs (for the settings in Section 2.1, a minute-long magnitude spectrogram in 32-bit floating-point precision takes 8.2 MiB, while a same-length 22.05 kHz signal in 16-bit integer precision takes 2.5 MiB), cancelling out savings in computation.

### 2.4 Testing

Since the network can process input of arbitrary size into a vector of 1500 class probabilities, we can directly apply it to full test recordings for the first subtask of the challenge (monodirectional recordings), and to 5-second excerpts for the second subtask (long-term multidirectional recordings at 5-second intervals).

## 3 Predictions from Metadata

Not all 1500 bird species of the challenge are likely to occur across the whole South American continent. Similarly, some migrating birds are not likely to occur in specific seasons, and some species are more likely to be heard during dawn than at noon. To learn about and use these dependencies, I train additional networks on the metadata supplied with the training recordings, which is also available for the test recordings: the date, time, geocoordinates, and elevation.

### 3.1 Data Encoding

As a first step, I cleaned up and unified the different formats for the time (e.g., "14:20", "11am", "morning", "PM") and elevation (e.g., "1400", "300-400", "1.270m", "to 1100 m") data, both of which seem to be drawn from freeform text fields. Date and geocoordinates were already unified, but not all fields are filled for each recording.

To process the metadata with a neural network, I encode it as a fixed-size vector. The numeric values for longitude, latitude and elevation are used directly. The date is converted to the day of the year, and the time to the minute of the day. To avoid a discontinuity from December 31, to January, 1 and from 23:59 to 0:00, I employ a $K$-dimensional circular encoding of these values using phase-shifted sinusoid functions:

$$\hat{d}_k = \sqrt{2}\sin(2\pi d/D + \pi k/K), \tag{10}$$

where $d$ is the value to encode (e.g., the minute of the day), $D$ is the number of possible values (e.g., 1440), and $\hat{\boldsymbol{d}}$ is the resulting $K$-dimensional encoding of $d$. I experimented with two settings: $K = 12$ and $K = 3$. Missing metadata fields are represented as zeros, and 5 additional binary values indicate which fields are valid. In total, this results in a 14-dimensional vector for $K = 3$.

As a final step, all values are standardized to zero mean and unit variance using statistics over the training set (except for the circular-encoded values, which are approximately standardized due to the $\sqrt{2}$ factor in Equation 10, assuming a uniform distribution of days of the year and minutes of the day).

### 3.2 Network Architecture

The network architecture for this part is very simple: The 14-dimensional input vector is processed by two fully-connected hidden layers of 256 and 512 leaky rectified units each, and finally classified by a fully-connected output layer of 1500 softmax units. Dropout or batch normalization were not helpful.

### 3.3 Training

Optimization uses the same settings as for the audio network, except that monitoring for early stopping uses the mean average precision (MAP, see Section 5) on the validation set rather than the cross-entropy loss, and mini-epochs of 5000 instead of 1000 updates.

Training targets are chosen according to strategy A or B of Section 2.3, with strategy B turning out to be more useful for ensembling.

To obtain additional variations for ensembling, I varied training in two ways: **A)** I added Gaussian noise to the metadata values (before circular encoding, if applicable), either with small standard deviations (3 days, 3 minutes, 5 meters, 0.3 degrees), medium standard deviations (7 days, 10 minutes, 20 meters, 1 degree) or large standard deviations (14 days, 30 minutes, 100 meters, 3 degrees). **B)** I trained networks on a subset of metadata fields, either leaving out one of the fields or all but one field.

As an additional regularization, I experimented with randomly dropping single fields (handling them like missing fields, i.e., replacing with zeros and flipping the validity indicator variable), but this only worsened results.

## 4 Ensembling of Predictions

Both to combine audio-based and metadata-based predictions and to benefit from multiple variations among the audio-based and metadata-based models, it is crucial to be able to merge predictions of multiple models.

In its simplest form, predictions by multiple models for the same recording can be merged by averaging them. There are different stages at which this could happen; empirically, it worked best to average predictions after global temporal pooling, but before applying the softmax output nonlinearity.

Averaging is sufficient for few similar models, but not optimal for larger sets of diverse models, or for combining audio-based and metadata-based predictions: in these cases, it pays off to weight the predictions of each model differently. To automate the choice of weights, I employ hyperopt [2], a general-purpose hyperparameter optimization tool. Given a set of models, I use it to search a weight in $[0, 1)$ for each model that optimizes the ensemble's mean average precision (MAP) on the validation set (specifically, the average of the MAPs for the foreground and background species; see the next section for details on the evaluation). When the set of models becomes large, hyperopt can also be used to choose which models to include in the ensemble at all. To support this, I extend the search space by a binary choice for each weight, allowing hyperopt to more easily set it to zero. While this strategy greatly helps in finding a good ensemble, of course it bears the risk of over-optimizing towards the validation set.

## 5 Results

Challenge submissions are evaluated in terms of mean average precision (MAP), both when using only the annotated foreground species as ground truth and when including the background species as additional correct classes. Given class probabilities $\boldsymbol{y}$ and a set of annotated species $S$ for a recording, the average precision is computed as

$$\mathrm{ap}(\boldsymbol{y}, S) = \frac{1}{|S|} \sum_{s \in S} \mathrm{p}(\boldsymbol{y}, S, \mathrm{rank}(\boldsymbol{y}, s)), \tag{11}$$

where $\text{rank}(\boldsymbol{y}, s)$ is the number of values in $\boldsymbol{y}$ less than or equal to $y_s$:

$$\text{rank}(\boldsymbol{y}, s) = \sum_i \left[ y_i \leq y_s \right], \tag{12}$$

and $\text{p}(\boldsymbol{y}, S, k)$ is the precision at $k$:

$$\text{p}(\boldsymbol{y}, S, k) = \frac{1}{k} \sum_{s \in S} \left[ \text{rank}(\boldsymbol{y}, s) \leq k \right]. \tag{13}$$

The mean average precision is the mean of the average precisions of all recordings. Note that when $|S| = 1$ (e.g., when only evaluating against foreground species), the average precision is equal to the reciprocal rank of the correct species, as easy to see when inserting (13) into (11).

As an additional evaluation measure, I use the classification accuracy with respect to the foreground species, and the top-5 classification accuracy (which treats a prediction as correctly classified when the foreground species is among the 5 largest class probabilities for a recording).

Table 1 lists validation results for all candidate model variants partaking in the ensemble search. In addition, it shows the ensembling weights, validation and official test results for three ensembles formed from the candidate models. We can draw several interesting insights from these results:

- All else kept equal, the residual network outperforms the plain CNN (variants A and B in Section 2.2, "Local predictions"). It still pays off to combine them in an ensemble.
- All else kept equal, log-mean-exp pooling tends to outperform temporal attention (variants A and B in Section 2.2, "Global predictions"). Again, it still pays off to combine them in an ensemble.
- All else kept equal, including background species in the targets tends to decrease the foreground MAP against foreground species (and the classification accuracy), while increasing the MAP against background species (variants A and B in Section 2.3, "Loss and targets").
- All else kept equal, retraining a model as a Born-Again Network (BAN) tends to improve its performance (variant C in Section 2.3, "Loss and targets").
- The metadata alone suffices to classify 21% of validation recordings correctly.
- Including all metadata fields gives better performance than omitting some.
- When training on a single metadata field, it seems location is the most informative cue.
- There is no systematic difference between 12-dimensional and 3-dimensional date/time encodings (but there is only a single pair of results with all else kept equal).
- Adding Gaussian noise to metadata diminishes results. This might indicate that the models learn to exploit unwanted correlations between training and validation data, such as recordings done on the same day at the same location, rather than learning robust occurrence patterns. However, when ensembling with audio models, blurred (noisy) metadata is superior to unmodified data.

**Table 1.** Results for several model variants on the validation set (columns "MAP-fg" to "Top-5"), both for audio (upper part) and metadata networks (lower part). The last three columns show model selections for three ensembles, with corresponding weights. The last four rows display ensemble results on the validation set and official test set.

| | model variant | | | | MAP-fg | MAP-bg | Top-1 | Top-5 | Ens. A | Ens. B | Ens. C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | local | global | target | BAN | | | | | | | |
| audio | plain | att16 | fg | | 0.655 | 0.567 | 0.57 | 0.76 | – | – | – |
| audio | plain | att16 | fg/bg | | 0.643 | 0.580 | 0.55 | 0.75 | 0.08 | 0.07 | – |
| audio | plain | att16 | fg/bg | ✓ | 0.672 | 0.606 | 0.59 | 0.77 | 0.72 | 0.63 | – |
| audio | plain | att64 | fg | | 0.648 | 0.565 | 0.56 | 0.75 | – | – | – |
| audio | plain | att64 | fg | ✓ | 0.676 | 0.589 | 0.59 | 0.78 | 0.38 | 0.21 | – |
| audio | plain | lme | fg | | 0.671 | 0.594 | 0.58 | 0.78 | – | 0.65 | – |
| audio | plain | lme | fg | ✓ | 0.686 | 0.608 | 0.60 | 0.80 | – | 0.78 | – |
| audio | plain | lme | fg/bg | | 0.668 | 0.621 | 0.58 | 0.77 | 0.06 | – | – |
| audio | plain | lme | fg/bg | ✓ | 0.679 | 0.631 | 0.59 | 0.78 | – | – | – |
| audio | resnet | att16 | fg | | 0.676 | 0.585 | 0.59 | 0.77 | – | 0.42 | – |
| audio | resnet | att16 | fg/bg | | 0.661 | 0.597 | 0.58 | 0.76 | 0.60 | 0.87 | – |
| audio | resnet | att64 | fg | | 0.667 | 0.580 | 0.58 | 0.77 | – | – | – |
| audio | resnet | att64 | fg | ✓ | 0.690 | 0.600 | 0.61 | 0.79 | 0.41 | 0.87 | – |
| audio | resnet | lme | fg | | 0.701 | 0.620 | 0.62 | 0.80 | 0.74 | 0.93 | 1.00 |
| audio | resnet | lme | fg | ✓ | 0.691 | 0.611 | 0.60 | 0.80 | – | 0.92 | – |
| audio | resnet | lme | fg/bg | | 0.677 | 0.634 | 0.59 | 0.78 | 0.45 | 0.65 | – |
| audio | resnet | lme | fg/bg | ✓ | 0.690 | 0.642 | 0.61 | 0.79 | 0.83 | – | – |
| | fields | K | noise | target | BAN | | | | | | |
| metadata | all | 12 | med | fg | | 0.240 | 0.231 | 0.17 | 0.31 | – | – | – |
| metadata | all | 12 | no | fg/bg | | 0.267 | 0.270 | 0.20 | 0.34 | – | 0.30 | – |
| metadata | all | 3 | high | fg/bg | | 0.191 | 0.195 | 0.12 | 0.25 | – | – | – |
| metadata | all | 3 | med | fg/bg | | 0.245 | 0.247 | 0.17 | 0.32 | – | 0.02 | 0.33 |
| metadata | all | 3 | no | fg | | 0.269 | 0.272 | 0.20 | 0.34 | – | – | – |
| metadata | all | 3 | no | fg | ✓ | 0.274 | 0.275 | 0.20 | 0.35 | – | – | – |
| metadata | all | 3 | no | fg/bg | | 0.272 | 0.276 | 0.21 | 0.34 | – | 0.22 | – |
| metadata | all | 3 | no | fg/bg | ✓ | 0.277 | 0.280 | 0.21 | 0.34 | – | – | – |
| metadata | no date | 12 | no | fg/bg | | 0.218 | 0.224 | 0.14 | 0.29 | – | – | – |
| metadata | no date | 3 | high | fg/bg | | 0.134 | 0.142 | 0.07 | 0.18 | – | 0.80 | – |
| metadata | no date | 3 | med | fg/bg | | 0.194 | 0.201 | 0.12 | 0.26 | – | – | – |
| metadata | no elev | 12 | no | fg/bg | | 0.232 | 0.235 | 0.16 | 0.30 | – | 0.22 | – |
| metadata | no elev | 3 | high | fg/bg | | 0.129 | 0.136 | 0.07 | 0.18 | – | – | – |
| metadata | no elev | 3 | med | fg/bg | | 0.190 | 0.194 | 0.12 | 0.26 | – | – | – |
| metadata | no loc | 12 | no | fg/bg | | 0.190 | 0.187 | 0.13 | 0.25 | – | – | – |
| metadata | no loc | 3 | high | fg/bg | | 0.121 | 0.124 | 0.07 | 0.16 | – | – | – |
| metadata | no loc | 3 | med | fg/bg | | 0.160 | 0.162 | 0.10 | 0.21 | – | – | – |
| metadata | no time | 12 | no | fg/bg | | 0.213 | 0.219 | 0.13 | 0.29 | – | 0.27 | – |
| metadata | no time | 3 | high | fg/bg | | 0.135 | 0.144 | 0.07 | 0.18 | – | 1.00 | – |
| metadata | no time | 3 | med | fg/bg | | 0.175 | 0.182 | 0.09 | 0.25 | – | 0.16 | – |
| metadata | only date | 12 | no | fg/bg | | 0.039 | 0.046 | 0.02 | 0.05 | – | – | – |
| metadata | only elev | 12 | no | fg/bg | | 0.054 | 0.064 | 0.02 | 0.06 | – | – | – |
| metadata | only loc | 12 | no | fg/bg | | 0.123 | 0.131 | 0.06 | 0.16 | – | – | – |
| metadata | only time | 12 | no | fg/bg | | 0.025 | 0.031 | 0.01 | 0.03 | – | – | – |
| | | | | | | | | | Ensemble MAP-fg (valid) | 0.754 | 0.809 | 0.764 |
| | | | | | | | | | Ensemble MAP-bg (valid) | 0.677 | 0.740 | 0.693 |
| | | | | | | | | | Ensemble MAP-fg (test) | 0.724 | 0.752 | 0.693 |
| | | | | | | | | | Ensemble MAP-bg (test) | 0.655 | 0.692 | 0.636 |

The official submissions are based on the three ensembles shown in the table. Specifically, I created four submissions for the monodirectional task:

**Run 1:** Ensemble A, an ensemble of only audio-processing networks
**Run 2:** Ensemble B, an ensemble of audio and metadata networks
**Run 3:** Ensemble B, but with an additional filter. The metadata of each training and test recording includes the year the respective file was added to BirdCLEF (2014, 2015 or 2017). Since past editions used a smaller set of classes, this information can be used to exclude particular species for 2014 and 2015, by forcibly setting their probabilities to zero. While this is only reliable for the foreground species, it noticeably improves results: Compared to Run 2, MAP-fg raises from 0.752 to 0.785, and MAP-fg from 0.692 to 0.705. The same technique was used by Runs 2–4 of the leading team in BirdCLEF 2018 (personal communication).
**Run 4:** Ensemble C, the best combination of a single audio and metadata network, but combined into a joint model and carefully fine-tuned (with initial learning rate 0.00001, halved whenever the validation error did not improve for 10 consecutive mini-epochs of 1000 updates, and trained until the third learning rate drop). Fine-tuning slightly improved validation results from a MAP-fg of 0.764 to 0.768 and a MAP-bg of 0.693 to 0.698. Test set results reported in Table 1 are for the fine-tuned model.

For the multidirectional task, I submitted predictions using Ensembles A, B and C applied to 5-second excerpts (Run 3 was not included since there is no information to use for filtering). They achieved c-MAP[2] scores of 0.113, 0.146 and 0.097, respectively.

Note that for both tasks, ensemble A outperformed ensemble C, while the latter was superior on the internal validation set. This might indicate that the validation set results are not a reliable indicator for choosing models after all.[3]

## 6 Dead Ends

During experimentation, I tried several ideas that did not turn out to be helpful in the end, and were not discussed above. I will describe a few in the following.

**mixup:** Several authors augmented training examples by mixing them with noise, based on a hand-designed approach for segmenting recordings into passages containing birds and passages only containing background noise [23,14,5]. When such a separation stage is not available, an alternative is to mix training examples regardless of bird vocalizations and update the labels accordingly. An interesting proposal in this regard is *mixup* [27]: Produce

[2] c-MAP computes the average precision per species over all segments, rather than per segment over all species, and then takes the mean over species.

[3] On a related note, beware of trying to interpret the ensemble weights – they are not indicative of which models are optimal; there are wildly different model selections and weights that perform about the same on the validation set.

weighted combinations of training examples paired with weighted combinations of their target vectors. While this seems to be especially well-suited to audio recordings, in my experiments, it slowed down convergence and lead to worse results.

**Data augmentation:** While data augmentation is often reported to be helpful for audio classification (using transformations such as time stretching, pitch shifting or equalization), in my experiments, it only diminished results.

**Grouped convolution/dropout:** Several authors proposed to design a network architecture to have multiple separate computation paths processing the same input that are merged in the end [4,16]. Some even consider this to be a design dimension on the same level as network depth and width, under the term *multiplicity* [24] or *cardinality* [26].

An easy way to implement such an architecture is to use grouped convolutions – a variation of convolutional layers that divides the input channels into non-overlapping groups to be processed separately. Multiple such layers with the same number of groups result in separated deep computation paths. Even with only two groups, this can promote the development of different feature sets (e.g., edge and color features in AlexNet [16, Fig. 3]). I experimented with different numbers of groups, keeping either the total number of feature maps or total number of parameters constant, but a single group always performed best. I also tried combining this with grouped dropout, i.e., randomly dropping complete groups at training time, to promote independence between computation paths, but to no avail.

**Per-Channel Energy Normalization (PCEN):** As an alternative to the basic learned magnitude transformation in Equation 1, I tried PCEN as proposed in [25]. It also includes a learned root compression, but with separate exponents per mel band, and in combination with an automatic gain control meant to improve robustness to different recording conditions. However, it worked significantly worse on the internal validation set.

## 7 Discussion

Combining an ensemble of audio networks that individually achieve a Mean Average Precision (MAP) score of up to 0.701 with an ensemble of metadata networks that achieve up to 0.277, it is possible to reach a MAP score of 0.809 (for detecting the foreground species in an internal validation set split off from the BirdCLEF 2018 training data). Compared to an audio-only ensemble, including the metadata improved scores by 3 to 5 percent points and helped securing the second place in the official challenge results.

Results could be improved in a number of ways. For one, I hardly varied the spectrogram settings – especially the sampling rate of 22.05 kHz (and upper frequency bound of 10 kHz for the mel filterbank) may be a suboptimal choice. While this was enough to win a bird detection challenge [9], it discards a lot of information from the original 44.1 kHz or 48 kHz signals. Secondly, seeing the improvement from going from a plain CNN to a residual network, there probably

still is a lot to gain from larger networks or more modern design strategies such as dual-path networks [3]. Thirdly, for convenience, I used a single validation set both for early stopping during training and for model selection and weighting for ensembling. This bears a strong risk of overfitting to the validation set, and means a subset of recordings will never be used for training. Furthermore, I used the same validation set as [14] (to be able to compare results during development), which was designed to contain at least one recording per species and otherwise picked randomly. This does not match the distribution of the test set and disregards the fact that some recordists upload several recordings from the same day and location, which may partly explain the high performance of the metadata-only networks (especially when not corrupting metadata with slight noise). Using cross-validation with more careful splits as [18, Sec. 2.2] would lead to more informed model selection and more diverse model variants for ensembling (trained on different sets of recordings). Finally, it is surprising that data augmentation did not help in the variants I tried; a further exploration of augmentation methods may improve generalization.

When judging the relevance of results for practical applications, it is important to distinguish the two subtasks. The MAP score used for the first subtask is purely rank-based (Equation 11): it rewards models giving the correct species higher scores than incorrect ones, but does not require a clear decision on which species occur in a recording. Furthermore, as the test set only contains files that include at least one of the possible 1500 bird species, models will never need to decide whether a file contains a bird (or a known bird) at all. Finally, the score takes an average over all test recordings, so species with more recordings will be given higher importance than rare species. Thus, the first subtask only captures the setting of an expert in the field interested in classification suggestions for a particular vocalization, not a passive acoustic monitoring scenario. The c-MAP score for the second subtask weights all species equally, and rewards models giving segments that contain a particular species higher scores than those that do not contain that species. This is closer to the requirements for passive monitoring, and still has a lot of room for improvements.

## Acknowledgements

I would like to thank Hervé Glotin, Hervé Goëau, Willem-Pier Vellinga, and Alexis Joly for organizing this challenge, supported by Xeno-Canto, Floris'Tic, SABIOD and EADM MaDICS. This research is supported by the Vienna Science and Technology Fund (WWTF) under grants NXT17-004 and MA14-018. I also gratefully acknowledge the support of NVIDIA Corporation with the donation of two Tesla K40 GPUs and a Titan Xp GPU used for this research.

## References

1. Amores, J.: Multiple instance classification: Review, taxonomy and comparative study. Artificial Intelligence **201**, 81–105 (2013). https://doi.org/10.1016/j.artint.2013.06.003, http://refbase.cvc.uab.es/files/Amo2013.pdf

2. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of the 30th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 28, pp. 115–123. Atlanta, GA, USA (Jun 2013), https://github.com/hyperopt/hyperopt
3. Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J.: Dual path networks. In: Advances in Neural Information Processing Systems 30, pp. 4467–4475. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/7033-dual-path-networks
4. Dan Cireşan, U.M., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3642–3649. Providence, RI, USA (Jun 2012), http://www.idsia.ch/~ciresan/data/cvpr2012.pdf
5. Fazekas, B., Schindler, A., Lidy, T., Rauber, A.: A multi-modal deep neural network approach to bird-song identification. In: Working Notes of CLEF. Dublin, Ireland (Sep 2017), http://ceur-ws.org/Vol-1866/paper_179
6. Foulds, J., Frank, E.: A review of multi-instance learning assumptions. Knowledge Engineering Review **25**(1), 1–25 (Mar 2010). https://doi.org/10.1017/S026988890999035X, http://www.cs.waikato.ac.nz/~ml/publications/2010/FouldsAndFrankMIreview.pdf
7. Furlanello, T., Lipton, Z.C., Itti, L., Anandkumar, A.: Born again neural networks. In: Proceedings of the 35th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 80. Stockholm, Sweden (Jul 2018), https://arxiv.org/abs/1805.04770
8. Goëau, H., Glotin, H., Planqué, R., Vellinga, W.P., Kahl, S., Joly, A.: Overview of BirdCLEF 2018: monophone vs. soundscape bird identification. In: Working Notes of CLEF. Avignon, France (Sep 2018)
9. Grill, T., Schlüter, J.: Two convolutional neural networks for bird detection in audio signals. In: Proceedings of the 25th European Signal Processing Conference (EUSIPCO). Kos Island, Greece (Aug 2017), http://ofai.at/~jan.schlueter/pubs/2017_eusipco.pdf
10. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Proceedings of the 14th European Conference on Computer Vision (ECCV). pp. 630–645. Springer International Publishing, Amsterdam, Netherlands (2016). https://doi.org/10.1007/978-3-319-46493-0_38, preprint http://arxiv.org/abs/1603.05027
11. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv e-prints **abs/1207.0580** (Jul 2012), http://arxiv.org/abs/1207.0580
12. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 37, pp. 448–456. PMLR, Lille, France (Jul 2015), http://proceedings.mlr.press/v37/ioffe15.html
13. Joly, A., Goëau, H., Botella, C., Glotin, H., Bonnet, P., Planqué, R., Vellinga, W.P., Müller, H.: Overview of LifeCLEF 2018: a large-scale evaluation of species identification and recommendation algorithms in the era of AI. In: Proceedings of CLEF. Avignon, France (Sep 2018)
14. Kahl, S., Wilhelm-Stein, T., Hussein, H., Klinck, H., Kowerko, D., Ritter, M., Eibl, M.: Large-scale bird sound classification using convolutional neural networks. In: Working Notes of CLEF. Dublin, Ireland (Sep 2017), http://ceur-ws.org/Vol-1866/paper_143.pdf

15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR). San Diego, CA, USA (May 2015), http://arxiv.org/abs/1412.6980
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012), http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks
17. Lasseck, M.: Bird song classification in field recordings: Winning solution for NIPS4B 2013 competition. In: NIPS Workshop on Neural Information Scaled for Bioacoustics. Lake Tahoe, NV, USA (2013), http://www.animalsoundarchive.org/RefSys/Nips4b2013NotesAndSourceCode/WorkingNotes_Mario.pdf
18. Lasseck, M.: Improved automatic bird identification through decision tree based feature selection and bagging. In: Working Notes of CLEF. Toulouse, France (Sep 2015), http://ceur-ws.org/Vol-1391/160-CR.pdf
19. Pinheiro, P.O., Collobert, R.: From image-level to pixel-level labeling with convolutional networks. In: Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1713–1721. Boston, MA, USA (Jun 2015), http://openaccess.thecvf.com/content_cvpr_2015/html/Pinheiro_From_Image-Level_to_2015_CVPR_paper.html
20. Schlüter, J.: Learning to pinpoint singing voice from weakly labeled examples. In: Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR). New York City, NY, USA (Aug 2016), http://ofai.at/~jan.schlueter/pubs/2016_ismir.pdf
21. Schlüter, J., Grill, T.: Exploring data augmentation for improved singing voice detection with neural networks. In: Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR). Málaga, Spain (Oct 2015), http://ofai.at/~jan.schlueter/pubs/2015_ismir.pdf
22. Sercu, T., Goel, V.: Dense prediction on sequences with time-dilated convolutions for speech recognition. In: NIPS Workshop on End-to-end Learning for Speech and Audio Processing. Barcelona, Spain (Nov 2016), http://arxiv.org/abs/1611.09288
23. Sprengel, E., Jaggi, M., Kilcher, Y., Hofmann, T.: Audio based bird species identification using deep learning techniques. In: Working Notes of CLEF. Évora, Portugal (Sep 2016), http://ceur-ws.org/Vol-1609/16090547.pdf
24. Veit, A., Wilber, M., Belongie, S.: Residual networks are exponential ensembles of relatively shallow networks. arXiv e-prints **1605.06431v1** (May 2016), https://arxiv.org/abs/1605.06431v1
25. Wang, Y., Getreuer, P., Hughes, T., Lyon, R.F., Saurous, R.A.: Trainable frontend for robust and far-field keyword spotting. In: Proceedings of the 42nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). pp. 5670–5674 (Mar 2017). https://doi.org/10.1109/ICASSP.2017.7953242, preprint http://arxiv.org/abs/1607.05666
26. Xie, S., Girshick, R., Dollar, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jul 2017), http://openaccess.thecvf.com/content_cvpr_2017/html/Xie_Aggregated_Residual_Transformations_CVPR_2017_paper.html
27. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: Proceedings of the 6th International Conference on Learning Representations (ICLR). Vancouver, Canada (May 2018), https://arxiv.org/abs/1710.09412