Bird Sound Classification using Convolutional Neural Networks

Inferences:

Analyzing and Eliminating of models:

- The input should consists of features derived from statistics of the spectrograms, these are to be applied on Decision trees, CNN and RNN's
- Ranking the feature importance returned from the decision trees, one can find relevant segments to identify each sound class
- Avoid decision trees due to large computational load that it takes
- Avoid LSTM, because of the gradient vanishing and explosion problems associated with the sigmoid gate function, the model is difficult to reach convergence.
- RNN, preprocessing and augmentation are difficult to implement.
- CNN is the best approach, where the spectrogram of the bird sound audio is the input.

CNN Modifications:
Reduce sparsity of the feature map
Replace the general convolution layer with an inception module
Instead of large filters in the c-layer for feature extraction small filters are preferred and run in parallel.

Preprocessing:

- Commonly used technique is the MEL-scale log-amplitude spectrogram, a kind of time-frequency representation that takes human auditory sensitivity with respect to frequency into consideration
- Apply band-pass filter with cut-off frequencies of 500Hz and 15kHz, since most bird species vocalize within this frequency range
- Hop length was determined so that each clip of one second contains exactly 255 frames
- Apply SNR(Signal-to-Noise ratio) threshold, based on that we include only sufficiently high spectrograms for training
- Applying augmentation methods as there is class imbalance, data augmentation is a strategy that enables us to significantly increase the diversity of data available for training models, without actually collecting new data. By performing augmentation we can prevent our neural network from learning irrelevant patterns, essentially boosting the overall performance.(Can be done offline for small datasets, here transforms are applied to the whole dataset whereas in online mode for large datasets they are applied in mini batches)
- Here we use noise as a regularization method on the input and hidden layers. The spectrograms contain noise, so we can add gaussian noise to the model because it does accurately reflect many systems and it is very easy to deal with mathematically, making it an attractive model to use. It is to modify our learning algorithm to reduce its generalization error but not its training error. Here we use noise but we can also use Dropout, Early stopping and Weight constraint.

- Apply Normalization, The mean and the variance for transformation was calculated from the entire training dataset.

Network Architecture:

ResNet:
As we add more and more layers we encounter the problem of Vanishing gradient that degrades the model, as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitely small.
Resnet addresses the issue of model degradation, bp gradients vanish which degrades the CNN, so introduce a highway pass between upper and lower layer and that result is residual block. The core idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers.

Inception module:
Allows for the utilization of *varying convolutional filter* sizes and concatenating them to learn spatial patterns at different scales

Output:
Output of the neural network is a one-hot vector.

Audio Based Bird Species Identification using Deep Learning Techniques

Challenges:

- Background noise
- Multiple birds singing at the same time (multi-label)
- Difference between mating calls and songs
- Inter-species variance [9]
- Variable length of sound recordings
- Large number of different species

Feature Generation:

Convolutional neural network with five convolutional and one dense layer
Every C-Layer uses activation function followed by max-pooling layer
Split data into Noise part and signal(birds sound) part
Compute spectrograms of both parts and split each of them into equal sized chunks
Use these chunks for training and testing and augment it with chunk from noise spectrogram

Split data into Noise part and signal part:

- First the signal is passed through a STFT, dividing every element by the maximum value, such that all values end up in the interval [0, 1]. With the spectrogram at hand, we are now able to look for the signal/noise intervals
- Select all pixels that are three times bigger than the row median and three times bigger than the column median
- These give us the signal that we need and avoid the noise, so we set these pixels to 1 and others to 0.
- Applying binary erosion and dilation filters to get rid of the noise and join segments, Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries.
- For noise follow the same steps but, 2.5 times bigger rather than 3 for signals and invert the result at the end.
- Some columns may not be part of either the noise or signal part because we use different thresholds; this is good as those parts do not contribute either way to the model.
- The signal and noise masks split the sound file into many short intervals. We simply join these intervals together to form one signal- and one noise-sound-file. Everything that is not selected is disregarded and not used in any future steps
- From the two resulting sound files we can now compute a spectrogram for both signal and noise.

Now, we compute the spectrograms for the noise as well as the signal file.
We split both spectrograms into chunks of equal size, this helps us in having equal sized inputs for our architecture and there are no empty chunks so each chunk can be used for training and testing.

Since the data size is quite small, compared to the number of classes we apply different data augmentation methods like shifting pitch, adding noise, shifting time and combining same class audio files.

Network Architecture:

Network contains five convolutional layers each followed by a max pooling layer which calculates the maximum, or largest, value in each patch of each feature map. After these we insert one dense layer which is a layer that is deeply connected with its preceding layer and the network concludes with a softmax layer which is typically the final output layer in a neural network that performs multi-class classification.
Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks, this normalization technique is applied before every convolutional and dense layer.

Learning methods and the batch size are Nesterov momentum methods and 8 or 16 training samples respectively.