Reading the Robot Mind – Presenting Internal Data Flow Within an AI for Classification of Bird Sounds in a Format Familiar to Subject Matter Experts

Paul Nussbaum [1]

[1] *ECPI University, 800 Moorefield Park Drive, Richmond, VA 23236, United States of America*

Abstract

An interactive Jupyter notebook [1] is built for the purpose of training and deploying a deep learning neural network artificial intelligence (AI) [2] to automatically identify birds from recorded audio [3]. The notebook allows the user to modify parameters along the training and classification (inference) pipeline and observe the results. As with traditional observation methods, the notebook lets users view visual representations (spectrograms, etc.) of input vectors for similar and different birds [4]. In addition to traditional methods, this notebook also presents data in its original format (audio recordings of birds). This is common practice for a field researcher or subject matter expert (SME) testing a microphone and recording system [5]- they will want to listen to the recordings to see if they contain valid and sufficient information. The notebook [6] extends this intuitive and useful technique to individual neural network layers - working backwards towards a best estimate of the original input (referred to in this working note as "reading the robot mind"). The user can even provide just the "answer" (select a bird at the final output layer), and the reading the robot mind system will work backwards through the entire automated process and AI layers to let the SME hear a best approximation of what the AI has learned that bird sounds like.

Keywords

Deep Learning, BirdCLEF 2023, Reading the Robot Mind, Explain Ability, Quality Assurance

1. Introduction

The BirdCLEF 2023 challenge [3] is part of 2023 LifeCLEF [7] and involves identifying Eastern African bird species by sound. Specifically, the task is to develop computational solutions to process continuous audio data and recognize the species by their calls. The best entries to this contest challenge will be able to train reliable classifiers with limited training data. The training data consists of short recordings of individual bird calls generously uploaded by users of xenocanto.org. These files have been down sampled to 32 kHz where applicable to match the test set audio and converted to the ogg format. The goal of the notebook is therefore to develop this solution from this data.

The ability of AI systems to explain their classification conclusions (also called “explain ability” or “explainability”) is gaining increased research focus, both to provide support for decisions [8], as well as for “possibilities of exposing complex AI models to human users/operators in an interpretable and understandable ways.” [9]. Towards the goals of the latter, this paper focuses on the information flow through a trained AI system, including ability to observe where important information may be discarded or corrupted in some way.

The notebook [6] lets a subject matter expert (SME) hear, visualize, and compare data every step and layer on the way. It allows comparison of samples from the same type of bird so the SME can see and hear if they are similar, as well as comparison of different birds so the SME can see and hear if they are different.

[1]CLEF 2023: Conference and Labs of the Evaluation Forum, September 18–21, 2023, Thessaloniki, Greece
EMAIL: pnussbaum@ecpi.edu (A. 1)

These audio and visual recreations are available for the following steps in the AI system:

- Segmentation, feature extraction (choice of Mel Spectrogram or MFCC, both having user selectable parameters), and conversion to image format with 8-bit quantization
- Each layer of a Conv2d, MaxPool combo model, followed by Dense embedding and Dense softmax categorizer.

This notebook provides the ability to work backwards through the system from any point and let the user see a best estimation of the features that were presented as input, as well as hear a best approximation of the bird recording from which they were extracted.

Finally, the reading the robot mind system allows the above analysis method to be used to select an arbitrary bird classification, and work backwards to an approximated input so that the SME can see and hear what the AI has learned that bird sounds like.

Note that due to the time and compute limitations imposed by the computing environment and contest rules provided, the notebook is divided into four public notebooks:

- https://www.kaggle.com/code/pnussbaum/v15h-birdclef2023-mindreader - This notebook focuses on the Segmentation and Feature Extraction aspects of the AI solution, allowing users to make modifications and see and hear how much information is retained.
- https://www.kaggle.com/code/pnussbaum/v15h-all-birdclef2023-mindreader - This notebook allows the user to use their final decision related to segmentation and feature extraction, and convert and save all the BirdClef2023 data into this format.
- https://www.kaggle.com/code/pnussbaum/v16e-gpu-all-birdclef2023-mindreader - This notebook uses the final decisions noted above, and trains the entire AI for a longer period of time, achieving better accuracy, and saving the trained AI system.
- https://www.kaggle.com/code/pnussbaum/v17b-all-birdclef2023-mindreader - This notebook brings all of this together for the sake of the contest submission and scoring.

In the following sections, segmentation, feature extraction, quantization, AI model creation, training, and validation are discussed in detail. Also shown with formulas and examples are the aspects of the reading the robot mind system, including visualization of filters, recreation of input approximations based on outputs of intermediate and final layers of the AI system, and also the method whereby the output can be forced to an individual bird, and a best approximation of what that bird sounds like is created by the system.

2. Segmentation, Feature Extraction, and Image Quantization Analysis

An automated segmentation algorithm is used, however, due to the rules of the BirdCLEF 2023 competition, the SME is not permitted to modify this algorithm [3]. After segmentation, the audio data is transformed into another domain (called feature extraction in this document), and finally saved as a two-dimensional grayscale image with 8-bit quantization. The quantization was chosen due to the power and memory constraints of the edge device that will be performing inference (bird classification) in the field.

The notebook allows the SME to try several Feature Extraction Algorithms and test the following:

- Do the features visually look similar for the same bird, and different for different birds?
- Is the similarity/difference enough to be able to visually classify which bird is which?
- If the feature extraction algorithm is performed in reverse to recreate the inputted audio (or an approximation thereof, due to the lossy nature of feature extraction) - is the recreated sound clear enough for the SME to identify the bird?
- If the answer is "no" to any of the above, allow fine tuning by the user

The notebook demonstrates this with three different pre-selected methods, and also allows the SME to make further changes as they see fit. The pre-selected methods are all using the Librosa Python library feature extraction methods [10] [11] of:

1. Melspectrogram [10] with default settings and the number of mel scale frequency bands set to 16; yielding 16 features per frame.

2. Melspectrogram with default settings and the number of mel scale frequency bands set to 32; yielding 32 features per frame.
3. MFCC [11] with default settings, 64 mel scale frequency bands, and 16 final cepstral coefficients; yielding 16 features per frame.

After feature extraction, all 5 second audio samples result in a 2-dimensinal image whose width is 313 frames and whose height is equal to the number of extracted features per frame, in grayscale format, with 8-bit quantization to a positive integer from 0 to 255.

Finally, a subset of samples from like birds, as well as a subset of samples of different birds are presented visually and via audio playback to the SME.

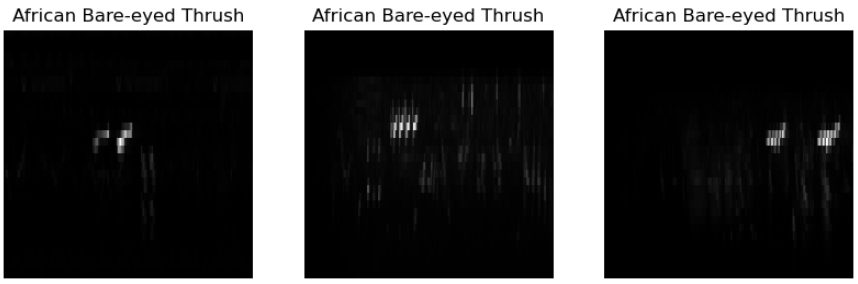


Figure 1: 32 band Mel Scale Spectrogram showing similarities between audio samples from the same type of bird.

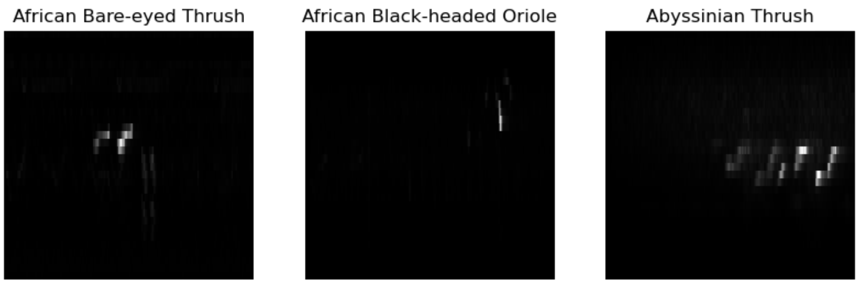


Figure 2: 32 band Mel Scale Spectrogram showing differences between audio samples from the different types of bird.

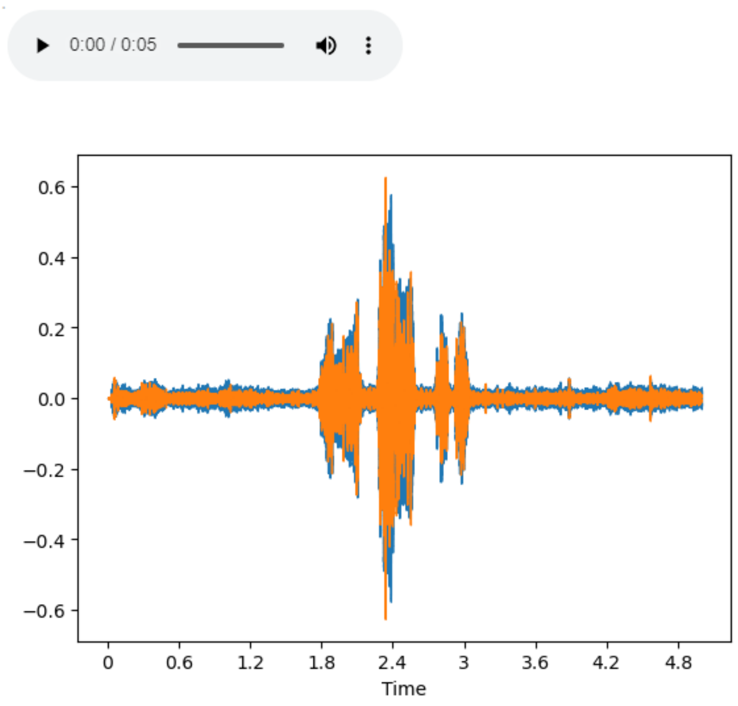


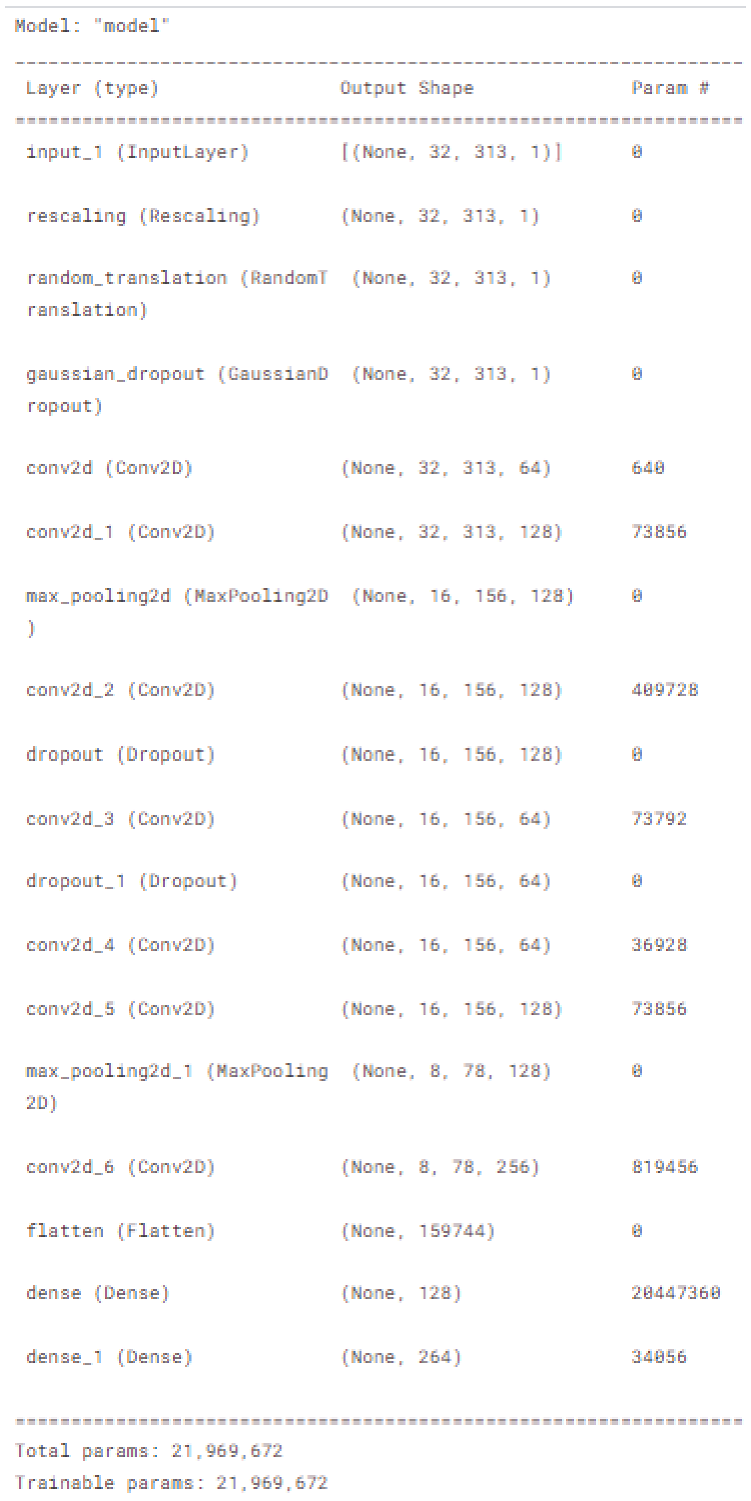
Figure 3: Audio envelope (In BLUE is the original audio, in ORANGE is the recreated audio) as well as audio playback tool.

Once the SME is satisfied that the feature extraction algorithm contains sufficient quality information, all of the data is converted and saved to disk using the selected segmentation, feature extraction, and quantization method.

If the SME makes no changes, there is also a pre-selected method which uses option 2 above and therefore converts 16,941 audio files into 147,248 eight bit greyscale images of width 313 pixels by 32 pixels in height, representing 5 second audio segments, organized in sub-directories by bird classification, to be used in the training/validation pipeline described in the next section (also called “image files”).

3. Layer-wise Analysis of the AI (Convolutional, Max Pooling, and Dense Layers)

A simple sequential (not residual) convolutional AI is used, with dimensions and architecture shown in the figure below. This was found to yield a satisfactory accuracy measure for the purposes of demonstrating the reading the robot mind system, while still falling within the running time and memory constraints imposed by the development platform and contest rules.



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 313, 1)]      0

 rescaling (Rescaling)       (None, 32, 313, 1)        0

 random_translation (RandomT  (None, 32, 313, 1)       0
 ranslation)

 gaussian_dropout (GaussianD  (None, 32, 313, 1)       0
 ropout)

 conv2d (Conv2D)             (None, 32, 313, 64)       640

 conv2d_1 (Conv2D)           (None, 32, 313, 128)      73856

 max_pooling2d (MaxPooling2D  (None, 16, 156, 128)     0
 )

 conv2d_2 (Conv2D)           (None, 16, 156, 128)      409728

 dropout (Dropout)           (None, 16, 156, 128)      0

 conv2d_3 (Conv2D)           (None, 16, 156, 64)       73792

 dropout_1 (Dropout)         (None, 16, 156, 64)       0

 conv2d_4 (Conv2D)           (None, 16, 156, 64)       36928

 conv2d_5 (Conv2D)           (None, 16, 156, 128)      73856

 max_pooling2d_1 (MaxPooling  (None, 8, 78, 128)       0
 2D)

 conv2d_6 (Conv2D)           (None, 8, 78, 256)        819456

 flatten (Flatten)           (None, 159744)            0

 dense (Dense)               (None, 128)               20447360

 dense_1 (Dense)             (None, 264)               34056

=================================================================
Total params: 21,969,672
Trainable params: 21,969,672
```

Figure 4: Model parameters for the simple CNN AI method used.

The AI starts with a rescaling layer (converting 8-bit unsigned integers ranging from 0 to 255, to floating point numbers ranging from 0 to 1). After that, data augmentation layers are used to reduce overfitting, including a time shift (left to right) of up to 20% with 0 fill, and a Gaussian dropout multiplier randomly ranging from 0 to 10%. The remaining layers are sequentially calculated, and have dimensions and parameters shown below. Finally, not shown in in the image is the fact that all layers have a rectified linear unit (relu) activation except for the last dense layer, which uses a softened winner take all (softmax) activation for final bird classification.

Training of the AI is performed with a randomly selected test/training split of 80%/20% from the entire set of previously saved image files containing the extracted features. The training is run for 16 epochs for a total of 30,682 seconds (just over 8.5 hours) using the GPU P100 option yielding a final categorical cross entropy loss of 0.9117 on the training set and 1.4665 on the validation set, and final accuracy measures of 0.7790 on the training set and 0.7120 (or 71%) accuracy on the validation set. Below, the history of these values over the training epochs is graphed, showing some overfitting towards the later epochs, despite the data augmentation employed.

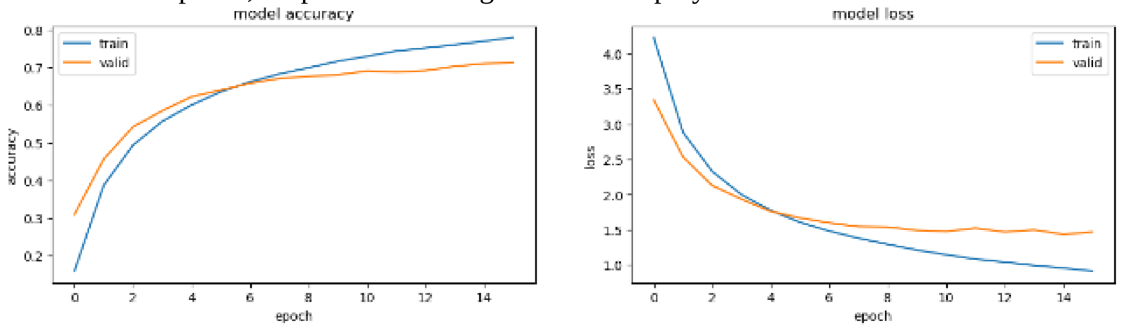


Figure 5: History of training and validation loss and accuracy over the 16 epochs of training

3.1. Visualizing Filter Patches

The notebook allows visualization of the convolutional filter patches. Although this information is more useful to the AI programmer than it is to the SME, it is presented here since it is a step in the system of reading the robot mind. The algorithms used are similar to the "expansion" method [12].

The first convolutional layer filter patches are visualized as simply the weights of each position in the filter.

$$P_{(l,n,x,y)}|_{l=1} = F_{l,n,pl,x,y}|_{l=1}$$

Where

- P is the image patch to be displayed
- F is the filter in question, along with its weights
- l is the convolutional layer (in this case, the first layer where l = 1)
- pl is the channel/filter depth of the previous layer. In this case, there is no previous layer, and the color channel depth of the input is 1 (grayscale) so pl=1
- n is the filter in question, from 1 to the number of filters in that layer
- x and y are the positions of the weights and/or image patch, and can range from 1 to the size of that filter patch (only square filters are used in this example)

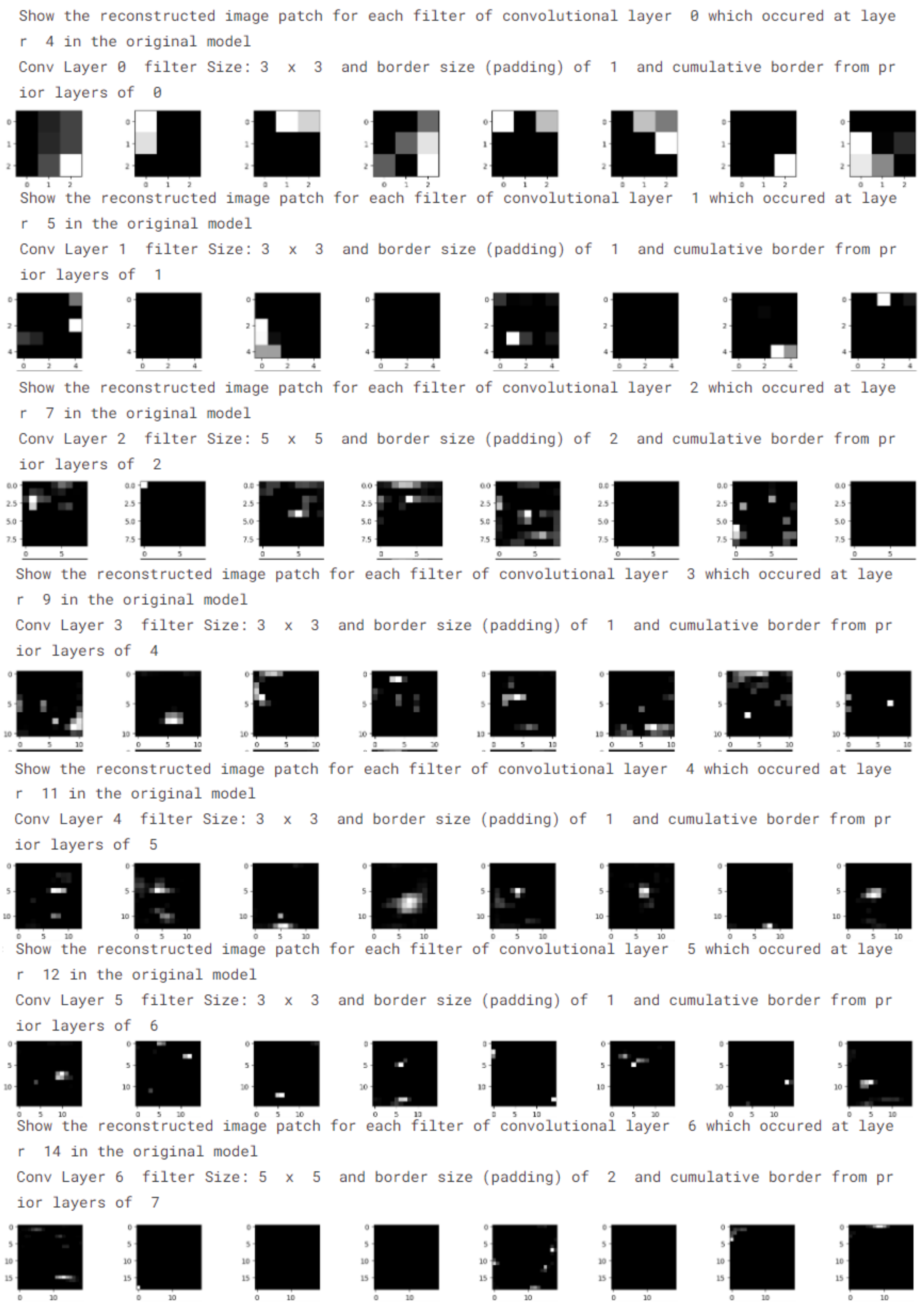
For subsequent layers, the visualizations are weighted sums of filter patches, with the weights coming from the layer being examined, and the filter patches coming from the cumulative prior layer visualizations. Note that since padding with zeroes was used, each subsequent convolutional layer will have a border.

$$P_{l,n,x,y} = \sum_{n=1}^{num\_filt_l} \sum_{x=cb_l}^{siz_l+cb_l} \sum_{y=cb_l}^{siz_l+cb_l} \sum_{pl=1}^{num\_filt_l} F_{l,n,pl,x-cb_l,y-cb_l} * P_{pl,n,x,y}$$

Where, in addition to the aforementioned variable definitions,

- l is the convolutional layer, starting with 2 (since the first layer image patches were already calculated for layer 1) and going up to the number of convolutional layers in the AI
- num_filtl is the number of filters for convolutional layer l
- sizl is the size of the filters at convolutional layer l (only square filters are used in this example).
- cb_l is the cumulative border for layer l, found by adding the border size of previous convolutional layers. For example, if the filter size is 3x3, then the border size is 1. If the filter size is 5x5 then the border size is 2.

The first eight filter patches for each convolutional layer are shown below.



```
Show the reconstructed image patch for each filter of convolutional layer  0 which occured at laye
r  4 in the original model
Conv Layer 0  filter Size: 3  x  3  and border size (padding) of  1  and cumulative border from pr
ior layers of  0
```

```
Show the reconstructed image patch for each filter of convolutional layer  1 which occured at laye
r  5 in the original model
Conv Layer 1  filter Size: 3  x  3  and border size (padding) of  1  and cumulative border from pr
ior layers of  1
```

```
Show the reconstructed image patch for each filter of convolutional layer  2 which occured at laye
r  7 in the original model
Conv Layer 2  filter Size: 5  x  5  and border size (padding) of  2  and cumulative border from pr
ior layers of  2
```

```
Show the reconstructed image patch for each filter of convolutional layer  3 which occured at laye
r  9 in the original model
Conv Layer 3  filter Size: 3  x  3  and border size (padding) of  1  and cumulative border from pr
ior layers of  4
```

```
Show the reconstructed image patch for each filter of convolutional layer  4 which occured at laye
r  11 in the original model
Conv Layer 4  filter Size: 3  x  3  and border size (padding) of  1  and cumulative border from pr
ior layers of  5
```

```
Show the reconstructed image patch for each filter of convolutional layer  5 which occured at laye
r  12 in the original model
Conv Layer 5  filter Size: 3  x  3  and border size (padding) of  1  and cumulative border from pr
ior layers of  6
```

```
Show the reconstructed image patch for each filter of convolutional layer  6 which occured at laye
r  14 in the original model
Conv Layer 6  filter Size: 5  x  5  and border size (padding) of  2  and cumulative border from pr
ior layers of  7
```

Figure 6: Filter patch visualization for the first 8 filters from each of the 7 convolutional layers of the AI

Observing the above figure, we can see that successive convolution layers manage successive levels of complexity. These are often described as edge detection first, then these are combined to form textures. Finally, the last convolution layers combine these textures to form patterns the AI is looking for in order to correctly identify birds.

Examining the filters, a programmer might be able to see if there are too many filters (many of them are similar), too many layers (filters are similar from layer to layer), and also specific patterns the CNN is looking for. Although filter visualization may be of use to the AI programmer, it can be a struggle for an SME who is not an AI programmer to understand this information, and so the reading the robot mind system will convert this information into a format familiar to the SME, as described in the next section.

3.2. Using Filter Patches to Reconstruct Approximation of Input (for Convolutional and Max Pooling layers)

Using convolutional neural networks to extract patterns from image data necessarily involves loss of information. Each convolutional layer has only a limited number of filters. Even though that limited number of filters is greater than would be needed to form an ortho-normal basis set (each filter having all 0 weights except for a weight of one at a different individual pixel), the AI is not being trained as an auto-encoder, and is instead being trained to minimize the loss when classifying birds. Also, information is being lost through the use of the relu activation function, which makes all negative output values a 0. Most of all, information is being lost at the max pooling layers.

The notebook providing the reading the robot mind system attempts to present this reconstructed best approximation of the input in a manner common and usable by the SME. This includes using the same visualization and audio playback mechanisms described in the earlier section that allowed the SME to qualitatively measure the segmentation, feature extraction and quantization algorithms.

The below image shows a three examples from the image set, proceeding left to right, first showing the original input, and then the subsequent reconstruction of the input from the output of the seven convolutional layers.

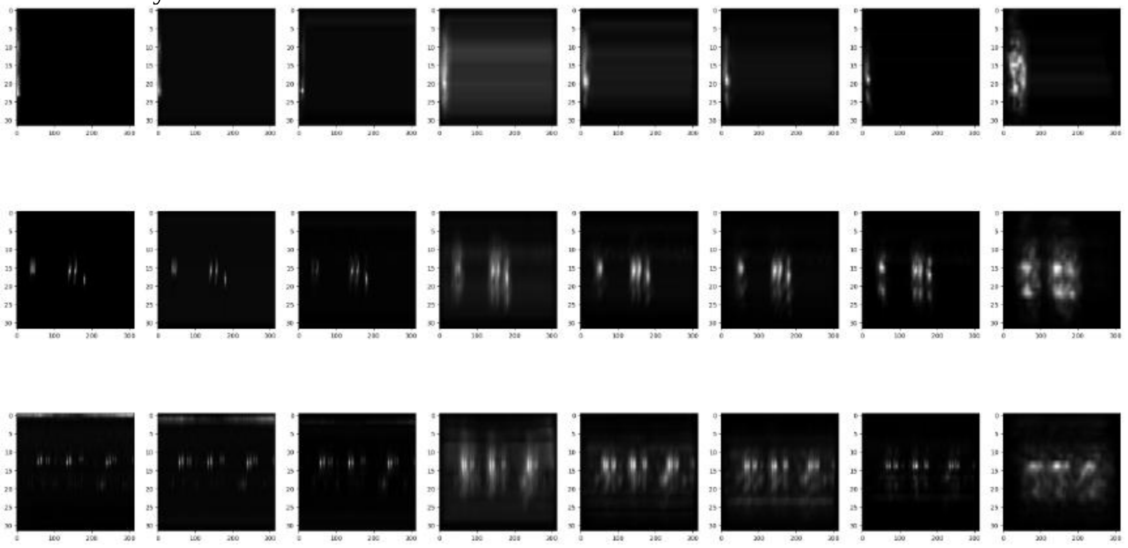


Figure 7: Three examples of Input reconstruction. Left is input, followed by reconstructions made from the outputs of convolutional layers 1 to 7.

Looking at the last row of images in the above figure, the SME can see the original input on the left. The outputs of the first two convolutional seem to show a very good reconstruction of the original input, but the third convolutional layer shows a definite lossy blurring of the reconstructed input. This is due to the max pooling layer that preceded this convolutional layer. The SME can see a similar blurring at the last convolutional layer, which is also preceded by a max pooling layer. These blurring's represent a loss of data, and the SME may want to determine if too much information is lost.

To create the above reconstructions, first the AI is subdivided into many models, each ending with the output of a different convolutional layer. This is done after the training is completed, so the configuration and trained weights remain the same, just the final output is now one particular convolutional layer, instead of the bird classification output of the originally trained AI model.

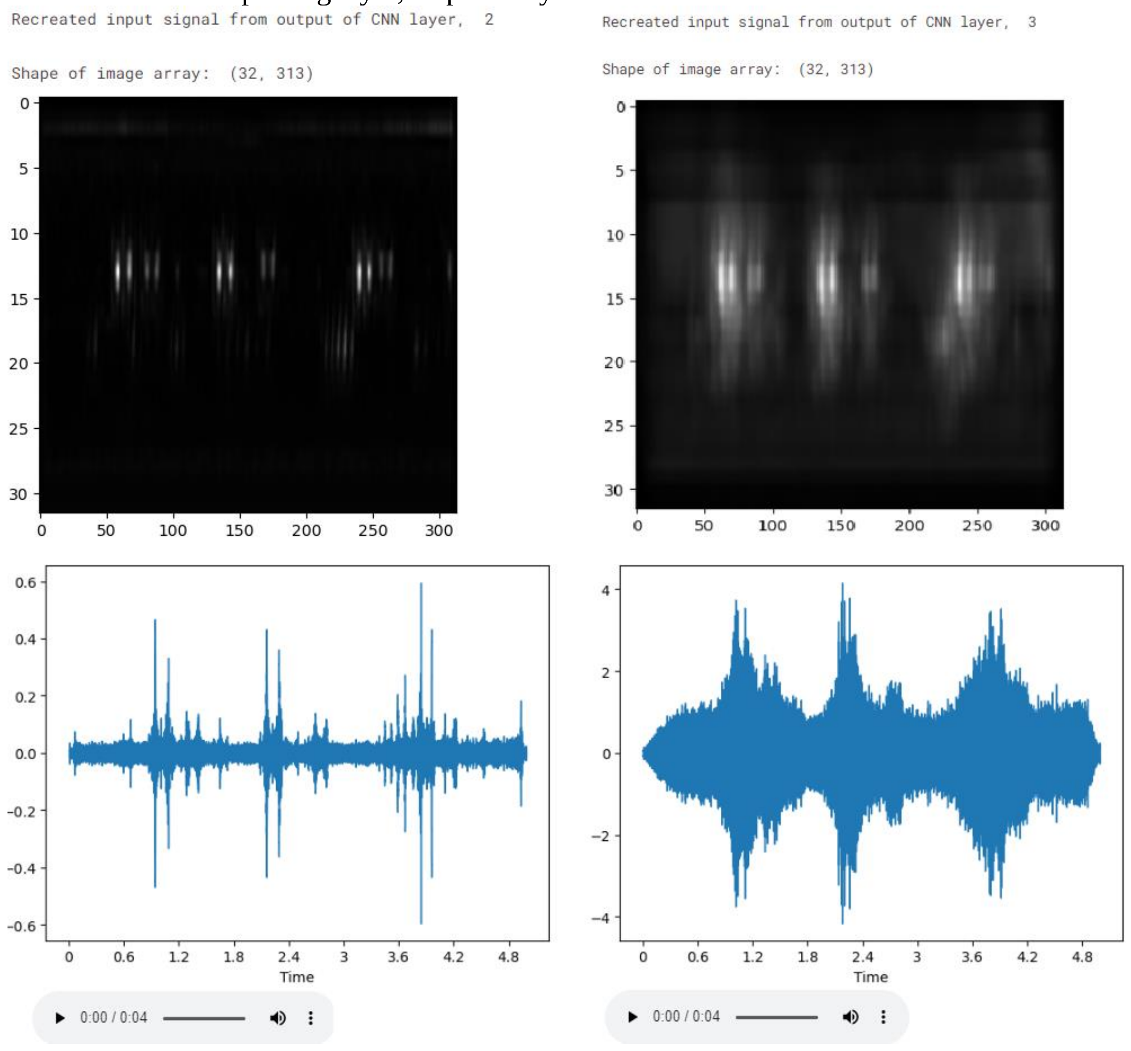
Once the output of a particular convolutional layer is calculated for a particular input, an approximation of the original input is made from that output as per the following formula.

$$IA_l = \sum_{n=1}^{num\_filt_l} \sum_{x=cb_l}^{siz_l+cb_l} \sum_{y=cb_l}^{siz_l+cb_l} O_{l,n,x-cb_l,y-cb_l} * P_{l,n,x,y}$$

Where, in addition to the aforementioned variable definitions,

- IA_l is the Image Approximation of the input to the AI based on the output from layer l
- P are the image patches for layer l, filter n, of width x and height y previously calculated and shown earlier.
- O is the output of layer l, filter n, at position x, and y

Using the reading the robot mind system, the SME can view the approximated input (visually as shown above) and can also hear and view the envelope of the reconstructed audio. Shown below is the reconstructed input based on the outputs of convolutional layers 2 and 3, discussed above, appearing before and after a max pooling layer, respectively.



```
Recreated input signal from output of CNN layer,  2

Shape of image array:  (32, 313)
```

```
Recreated input signal from output of CNN layer,  3

Shape of image array:  (32, 313)
```

Figure 8: Information about data content presented to SME for qualitative analysis. Both left and right images show reconstructed input from the same audio segment, using the output of two different convolutional layers (left is before a max pooling layer, and right is afterwards) to reconstruct the input approximation. The SME is presented with the approximations of the extracted features input image, as well as a playback button and visual envelope of the audio.

When looking at the images above, the SME can see that quite a bit of information is discarded at the max pooling layer. When listening to the audio playback the SME can hear that it is possible to distinguish a bird call from the leftmost recreation, but it sounds like a very noisy signal to the right. The SME might be able to hear some aspects of a bird call (change of frequency, bursts of sound) but it is clearly not distinguishable as being a bird call, and it is quite impossible to use the audio to classify which bird is making the audio.

In this way, the SME can clearly see and hear that the AI solution is throwing away a great deal of information that would otherwise be used by the SME to classify birds. There are some possible conclusions from this information.

- The AI is only trained on limited sounds from the real world, unlike the SME who has listened to many other audio sounds other than those from the training set, both relating to bird calls, as well as other everyday sounds the SME can categorize. These can include the sound of a person talking, the sound of crickets, etc. which are not specifically classified in the training set. There might be an opportunity to train the AI on other sounds like these.
- The AI may be throwing away too much **feature information**, such as in the case of the max pooling layer. When max pooling is performed in the column direction, different frequency bands are effectively pooled, which may remove the ability to distinguish important frequency information that can be used to categorize the bird. There might be an opportunity to throw away less frequency information by reducing the max pooling in the column direction.
- The AI may be throwing away too much **time information**, such as in the case when max pooling is performed in the row direction, different fast occurring events are effectively pooled, which may remove the ability to distinguish different events happening in sequence; information that can be used to categorize the bird. There might be an opportunity to throw away less time information by reducing the max pooling in the row direction.
- As discussed above, the AI is purposely throwing away information to reduce training resources and minimize overfitting that might happen with a larger network. This reduction of dimensionality might be accomplished a different way, other than by max pooling, which could lead to less loss of information, as perceived by the SME.

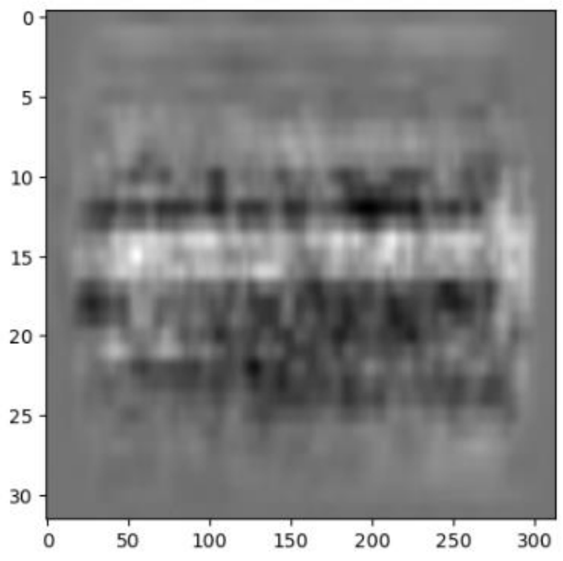
3.3. Reconstructing an Approximation of Input from Dense Layer Outputs

The output of the dense layer just after the flatten layer also represents a loss of information, because the flatten layer has (using the example settings) has 159,744 values, whereas the output of the dense layer following this (also called the embedding layer) has only 128 neurons. This represents a great reduction in dimensionality, and therefore a great opportunity for loss of information.

To calculate this input approximation based on the output of this dense layer, we estimate the output of the prior convolutional layer by multiplying the output of the dense layer by it's trained weights, and then performing the flatten function in reverse. This gives us approximations of the output of that prior convolutional layer, after which the above formulas are used to reconstruct and approximation of the input.

The same can be done from the output of the last (classification) Dense layer, working backwards as noted above. Below is a figure showing what the SME can see and hear based on the output of the dense layers. Once again, we are using the same input data as described in the previous section.

```
Weights array from the last Conv2D layer to the Embed layer has the shape  (159744, 128)
Weights array shape to the embed layer after undoing the Flatten  (8, 78, 256, 128)
```



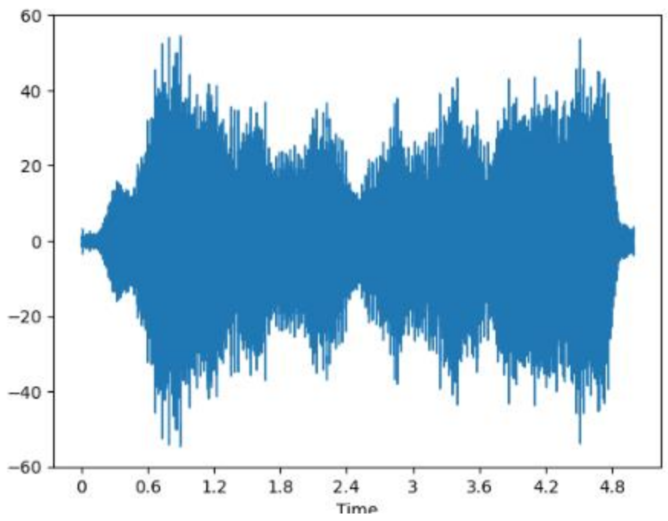


Figure 9: Information about data content presented to SME for qualitative analysis showing a reconstructed input from the same audio segment discussed earlier, using the output of the dense layer after the flatten.

When looking at the images above, the SME can see that quite a bit of information is discarded by the dense layer. When listening to the audio playback the SME can hear a very noisy signal. The SME might be able to hear some aspects of a bird call (some specific frequencies, some recognizable frequency changes, bursts of sound that roughly mimic the bursts of sound from the original audio sample, etc.) but it is clearly not distinguishable as being a bird call, and it is quite impossible for the SME to use the audio to classify which bird is making the sound.

Not described here are additional details and possible conclusions that can be observed by performing the above using different inputs, as well as possibly selection of different hyperparameters (such as the volume and row wise max pooling parameters discussed earlier) as well as possibly entirely different AI architectures. The notebook leaves this to the user, allowing experimentation with different methods.

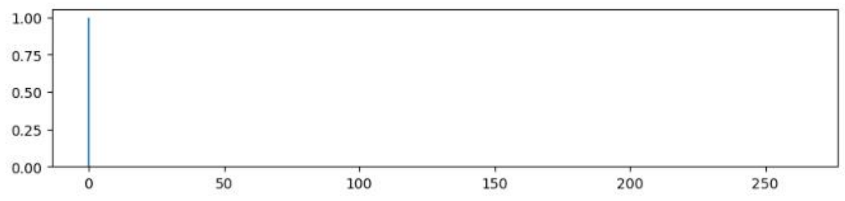
4. Working Backwards from a Bird Classification (What Does This Bird Sound Like?)

The reading the robot mind system also allows one more function that can prove useful to the SME who is helping the AI programmer improve the system. This function is the ability to specify a particular output (bird classification) and work backwards through the entire AI to recreate an approximation of the original audio input, even when no input is provided.

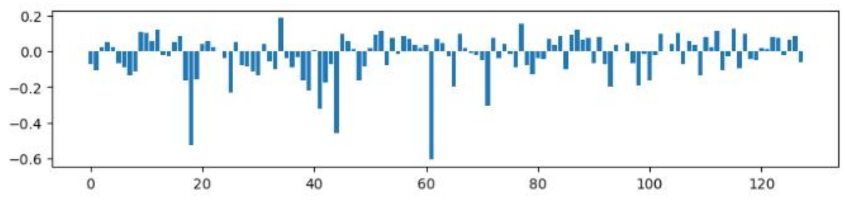
Instead of using an actual output of the final dense layer from a given input to work backwards and recreate an approximation of that input, the reading the robot mind system allows the SME to simply select a bird. This value is forced as an output of the last dense layer, by making all of the output values 0 except for the output value for that bird, which is set to 1 (similar to a 100% classification accuracy estimation of a particular bird that might be made by the automated AI system).

In the below figure, we show such an input estimation. The SME forces the output of the AI to be the first bird in our dataset whose feature extraction examples were shown earlier in **Figure *1***: 32 band Mel Scale Spectrogram showing similarities between audio samples from the same type of bird..

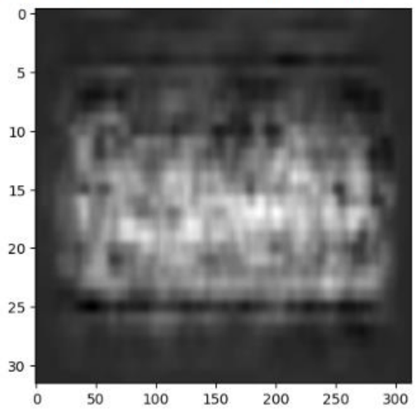
```
Here is the forced output of our classification layer (the last Dense layer with Softmax activation)
```



```
Weights array from the embed layer to the class layer has the shape  (128, 264)
Here is the forced output of our embed layer (the first Dense layer after flattened last Conv layer)
```



```
Weights array from the embed layer to the class layer has the shape  (159744, 128)
Weights array shape from the embed layer after undoing the Flatten  (8, 78, 256, 128)
Here is the best approximation of the input for this bird
```



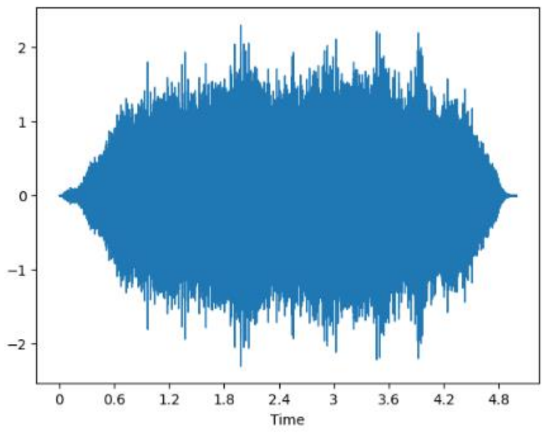


Figure 10: Reconstructed Input from Forced Output (top) working backwards to prior dense layer (next row), then un-flattening and using aforementioned methods to work backwards through convolutional layers (recreated input image approximation, followed by audio envelope and playback tool).

The SME can see that some of the aspects from the original bird samples are present. These include:

- The central band of frequencies being loudest, similar to the same band of frequencies being the loudest in the original extracted features from this bird.
- The time varying cadence of sounds, similar to the time varying sounds being present in the original samples.

Nevertheless, there are a large number of distortions that show great differences to the SME from the original samples. These include:

- So many other frequencies also having similar loudness to the “important” central ones, which make the vertical features extracted blurry, and the audio playback quite noisy.
- A compressed version of the sound with regard to time, blurring the distinctions where different sounds start and end (horizontal blurring).
- The final audio when listened to by the SME cannot be used to distinguish which bird is making the sound, and the noise and aforementioned distortions making the sound not recognizable as a bird call at all.

4. Conclusions

The reading the robot mind system is implemented, allowing the SME to observe and qualitatively analyze the internal data flow of deep learning neural networks in a format familiar to them.

With this, the SME can understand where in the pipeline the most information is being discarded by the AI system, and possibly help the programmer make improvements in future systems.

5. Acknowledgments

Thanks to all of the SME’s whose work before this have allowed all of this to be possible. Special thanks to my daughter, Nicole Nussbaum, BS Biology, James Madison University and bird identification expert, who was instrumental in helping me understand what SME’s look for. Thanks also to ECPI University and to my family for allowing me the time to work on this software. Finally, thanks to all of the people who took the time to read this document, and who may find it interesting, insightful, fun, and perhaps even useful in improving future AI systems.

6. References

[1] J. Perkel, "Why Jupyter is data scientists' computational notebook of choice," *Nature,* vol. 563.7732, no. (2018), pp. 145-147, 2018.

[2] IBM, "AI vs. Machine Learning vs. Deep Learning v. Neural Networks: What's the Difference?," 2023. [Online]. Available: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks.

[3] S. D. T. K. H. R. H. C. F. G. H. G. H. V. W. P. R. J. A. Kahl, "Overview of BirdCLEF 2023: Automated bird species identification in Eastern Africa.," *Working Notes of CLEF 2023 – Conference and Labs of the Evaluation Forum,* 2023.

[4] Audibon Society, "Start using Spectrograms to Read Bird Songs and Calls," 2023. [Online]. Available: https://www.audubon.org/news/start-using-spectrograms-read-bird-songs-and-calls.

[5] Acoustic Nature, "Best way to record birdsong: Gear guide and tips," 2020. [Online]. Available: https://www.audubon.org/news/start-using-spectrograms-read-bird-songs-and-calls. [Accessed 2023].

[6] P. Nussbaum, "v15h BirdClef2023 Mindreader," 2023. [Online]. Available: https://www.kaggle.com/code/pnussbaum/v15h-birdclef2023-mindreader.

[7] C. B. L. P. S. K. H. G. B. D. D. M. J. E. C. L. T. L. R. C. M. Š. M. H. M. S. H. G. R. P. W.-P. V. H. K. T. D. I. E. P. B. H. M. A. Joly, "Overview of LifeCLEF 2023: evaluation of ai models for the identification and prediction of birds, plants, snakes and fungi," in *International Conference of the Cross-Language Evaluation Forum for European Languages*, 2023.

[8] J. M. K. M. S. a. M. P. W. Wu, "Explainable AI for Early Detection of Health Changes Via Streaming Clustering," in *2022 IEEE International Conference on Fuzzy Systems*, Padua, 2022.

[9] S. P. a. S. A. S. Sutthithatip, "(Explainable) Artificial Intelligence in Aerospace Safety-Critical Systems," in *IEEE Aerospace Conference*, Big Sky. MT, 2022.

[10] Librosa, "librosa.feature.melspectrogram," 2023. [Online]. Available: https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html. [Accessed 2023].

[11] Librosa, "librosa.feature.mfcc," 2023. [Online]. Available: https://librosa.org/doc/main/generated/librosa.feature.mfcc.html. [Accessed 2023].

[12] e. a. Voss, "Visualizing Weights," *Distill*, vol. 10.23915/distill.00024.007, no. 10.23915/distill.00024.007, p. 10.23915/distill.00024.007, 2021.

[13] Google, 2023. [Online]. Available: https://distill.pub/2020/circuits/visualizing-weights/.