K. Lisa Yang Center for Conservation Bioacoustics

Machine Learning Detectors in Raven Pro

Introduction

Starting in release 1.6.5, Raven Pro provides a platform to integrate machine learning models as automatic detectors for bioacoustics classifications. Currently, only TensorFlow CNN (TF) models are supported. This is implemented in current Raven Pro's detector framework, e.g., a new machine learning detector called "Learning Detector" is added to existing list of available detectors in Raven Pro detector framework.

User with a trained TensorFlow CNN classification/detection model can place it in the Raven Pro user directory. If the required metadata is present in the model package (details below), Raven Pro will read the model's metadata and make the model available in its learning detector configuration UI. User can then select appropriate model for automatic detections/classifications. Detected events will be displayed/stored in Raven Pro selection table in the current Rave Pro detector framework.

Four models and their required metadata are included in Raven Pro 1.6.5 release:

- BirdNET_GLOBAL_3K_V2.2 (3000+ birds and other terrestrial species),
- Koogu_Katydid (30+ classes, Panama Katydids)
- Koogu_NARW (Northern Atlantic Right Whale)
- Koogu_Blue_Whale

This document describes the requirements of metadata and TF model format for Raven Pro to integrate a TF model. The document is intended as a reference for TF model developers when they prepare the model package to be used in Raven Pro. User should reference the Raven Knowledge Base article "MACHINE LEARNING DETECTOR – RUNNING DETECTIONS" on how to use the "Learning Detector" in Raven Pro.

Check list to prepare a model for Raven Pro

Here is a quick check list for preparing a model to be used in Raven Pro. More detailed descriptions for each item are in the following sections.

- √ Model is TensorFlow CNN, saved in protocol buffers format (saved_model.pb)
- √ Model input sound signal is unnormalized waveform
- √ Prepare model configuration file (model_config.json)

- √ Prepare model class output file(s) (e.g., birds_north_america.csv)
- √ Prepare model label file(s) (e.g., common_names.csv)
- √ Place the model package in Raven user directory in the "[user]/Raven Pro 1.6/Models" folder

---

TensorFlow CNN models and sound signals

Supported bioacoustics classification machine learning models:

- Raven Pro supports trained TensorFlow (TF) CNN models for bioacoustics classifications

Saved model in Protocol Buffers file format

- The TF CNN models must be saved in Protocol Buffers format (e.g. saved_model.pb file)

Model package and "Models" folder in user directory

- The model package is a file folder that includes the saved model pb file (saved_model.pb) and associated metadata files (more details below). The package must be placed in Raven Pro user directory, in "...\Models" folder, e.g., [user]\Raven Pro 1.6\Models, similar to other user data files such as Selections, Presets etc. Raven Pro automatically lists all the models available in the "...\Models" folder in the Learning Detector UI to allow user to select one of the models as automatic detector for classifications/detections.

Sound signals in waveform

- Raven Pro feeds the TF model sound signal in waveform and the model is expected to perform its own FFT (Fast Fourier Transform) to convert the waveform signals to spectrograms

Sound signals unnormalized in Raven Pro

- The sound signals waveform sent to TF model from Raven Pro is unnormalized, e.g., the normalization is expected to be performed by TF model.

Sound signal may be resampled in Raven Pro to match the sample rate of the model

- Sample rate supported by the model should be provided in the model's metadata in the model configuration file, the model_config.json file(more details below). If the sample rate of the sound signal doesn’t match what is supported by the model, Raven Pro will process/resample the sound data to match the sample rate before sending it to the model for classification/detection.

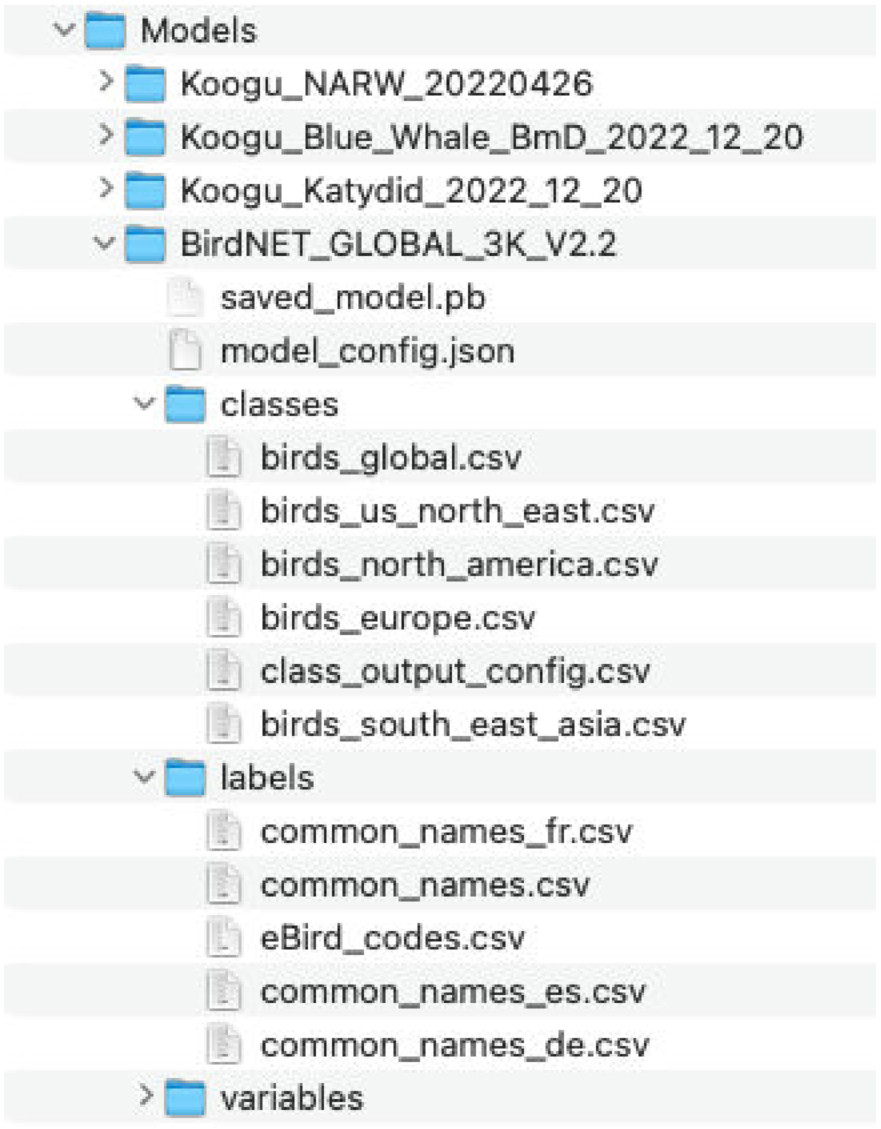
Metadata

For Raven Pro to load a trained TF model, send sound signals to the model properly and process the model’s classification outputs, the following information/metadata are required to be included in the TF model package/folder:

- Model configuration file: model_config.json
- Output class files: e.g., birds_global.csv
- Label files: e.g., common_names.csv

Model package file folder structure

The following is an example of a model package file folder structure, as seen in BirdNET model package included in Raven Pro 1.6.5:



```
Models
  Koogu_NARW_20220426
  Koogu_Blue_Whale_BmD_2022_12_20
  Koogu_Katydid_2022_12_20
  BirdNET_GLOBAL_3K_V2.2
    saved_model.pb
    model_config.json
    classes
      birds_global.csv
      birds_us_north_east.csv
      birds_north_america.csv
      birds_europe.csv
      class_output_config.csv
      birds_south_east_asia.csv
    labels
      common_names_fr.csv
      common_names.csv
      eBird_codes.csv
      common_names_es.csv
      common_names_de.csv
    variables
```

Model configuration file: model_config.json

- The model_config.json file provides model configuration information needed by Raven to send sound signals to the model properly. The information provided in the file must match the configuration of the model.
- The file format is in JSON. See Appendix II for the model_config.json file schema
- The file needs to be placed at the top level of the model package (see above Model file folder structure)
- model_config.json file needs to include the following information:
  - specVersion
  - modelDescription

        - signatures
            - modelInputs
            - modelOutputs
            - semanticKeys (optional)
        - modelTypeConfig
        - globalSemanticKeys
    - When user select a TF model from Raven Pro learning detector UI, Raven Pro reads the model_config.json file in selected model package and populates the learning detector's UI with available model signatures, the selected signature's available model inputs, outputs, class outputs and labels
    - Specifications and more detailed information on the model_config.json file structure/format can be found in Appendix I.

Template to create model_config.json

Below is a template to create the model_config.json file needed by Raven Pro. You can copy and paste the text below to a text editor, make updates as needed and save it as model_config.json file. Place the file in your model's package's top level. The fields highlighted in yellow need to be updated to reflect your model's configuration. If you used default signatures to generate the TF model, most likely the signatureName, inputName and outputName are still defaults and you don't need to change them here. In that case, the most important information to update is the sampleRate and globalSemanticKeys. The globalSemanticKeys are the list of the detection/classification classes in your model. The order of the keys must match the order of the classification outputs of your model. The keys are used to specify thresholds of class output scores and label the classes. See more details in Output class files and Labels files.

Here is the template:

```
{
  "specVersion": 1,
  "modelDescription": "your model description",
  "signatures": [
    {
      "signatureName": "basic",
      "modelInputs": [
        {
          "inputName": "inputs",
          "sampleRate": 250.0,
          "inputConfig": [
            "BATCH",
            "SAMPLES"
          ]
        }
      ],
      "modelOutputs": [
        {
          "outputName": "scores",
          "outputType": "SCORES"
        }
      ],
      "semanticKeys": []
    }
  ],
  "modelTypeConfig": {
    "modelType": "RECOGNITION"
  },
  "globalSemanticKeys": [
    "class1",
    "class2"
  ]
}
```

Output Class Files

- User or model developer can configure the classification outputs to group classes based on geographic regions, interests or focused species etc. This is done by creating separate output class files and put them in "…\Classes" folder in the model package.
- The output class files are in CSV format (comma separate value)
- Examples of output class files included in BirdNET model:
  - Birds_global.cvs
  - Birds_us_north_america.cvs
  - Birds_europe.cvs
  - Birds_south_east_asia.cvs

- Output class file structure: for each species, there is a row that contains the following information:
  - Semantic key/ID of the species – string
  - Score threshold for detection/classification – positive number
  - Low limit frequency of the call – positive number
  - Upper limit frequency of the call – positive number
  - Whether the species should be suppressed/ignored during detection/classification – boolean TRUE or FALSE
  - The following is a section of the birds_north_america.csv file, as an example:

comred	0.25	0	12000	FALSE
hoared	0.25	0	12000	FALSE
coohaw	0.25	0	12000	FALSE
norgos	0.25	0	12000	FALSE
eurspa1	0.25	0	12000	FALSE
shshaw	0.25	0	12000	FALSE

- Editable in Raven:
  - Raven Pro lists all the available csv files in the model package's "…\Classes" folder in the learning detector UI to allow user to choose. The content of a user selected class output file is displayed in the learning detector UI. User can change the threshold (threshold change is applied to all specifies in Raven Pro), suppress some or all species (then reselect a few focused ones) in the UI for class output processing
- Create or edit using any text editor
  - User or model developer can also edit an existing output class file or create a new one based on their need or focus of analysis using any text editor, such as MS Excel.
- During detection/classification, Raven Pro will only produce classification/detection results in a Raven selection table for the events that have scores above the threshold specified in the selected output class file and the species is not suppressed.

Label Files

- Label files contain the labels/names of the classes/species in the learning detector's detection/classification outputs. Raven translates the class ID from semantic key to the label provided in the selected label file.
- The label files are in CSV format, comma separated value.
- User or model developer can create multiple label files based on language, common names etc. and placed them in the "…\Labels" folder in the model package.

- Raven will list the available label files in the "…\Labels" folder in the learning detector UI, allowing user to select.
- Example of label files included in BirdNET model
  - Common_names.csv
  - Common_names_de.csv (German)
  - Common_names_es.csv (Spanish)
  - Common_names_fr.csv (French)
- The following is a portion of common_names.csv, as an example

rufwar1	Rufous-faced Warbler
yebwar1	Yellow-bellied Warbler
watgua1	Wattled Guan
spchon1	Spiny-cheeked Honeyeater
lesred1	Lesser Redpoll
comred	Common Redpoll
hoared	Hoary Redpoll
yertho1	Yellow-rumped Thornbill
tastho1	Tasmanian Thornbill
brotho1	Brown Thornbill

- The label file is a list of one-to-one translation for each class/species, from semantic key to label. The label is used in the Raven detection/classification output selection table in the "Label" annotation column. See example below, a screenshot of selection table with result of Learning Detector detections using BirdNET model.

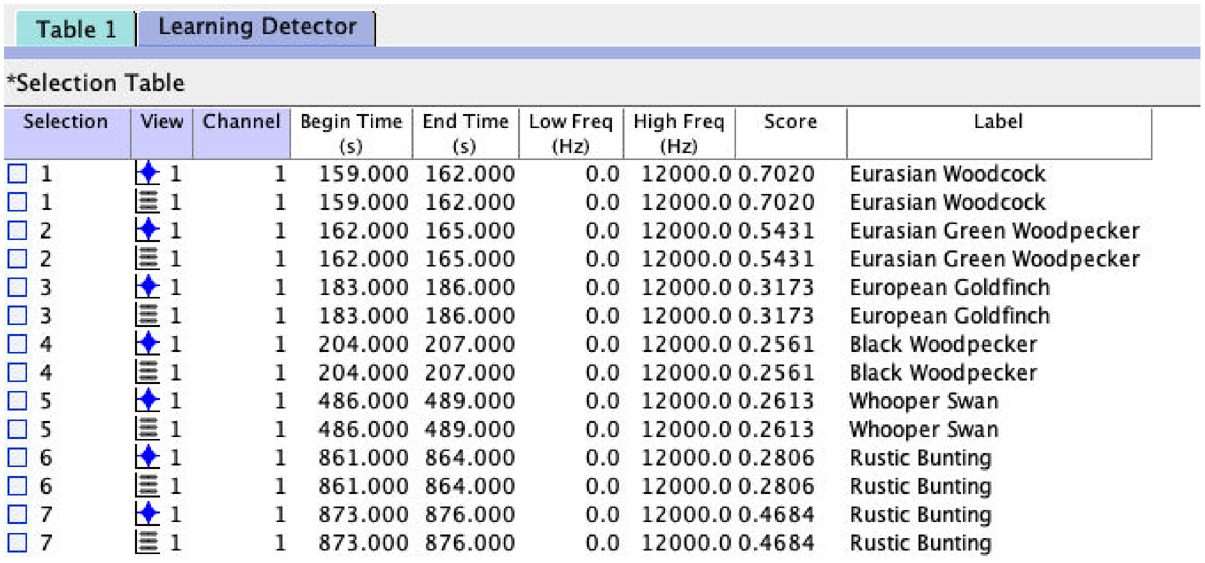


Table 1 | Learning Detector

*Selection Table

Selection	View	Channel	Begin Time (s)	End Time (s)	Low Freq (Hz)	High Freq (Hz)	Score	Label
☐ 1	1	1	159.000	162.000	0.0	12000.0	0.7020	Eurasian Woodcock
☐ 1	1	1	159.000	162.000	0.0	12000.0	0.7020	Eurasian Woodcock
☐ 2	1	1	162.000	165.000	0.0	12000.0	0.5431	Eurasian Green Woodpecker
☐ 2	1	1	162.000	165.000	0.0	12000.0	0.5431	Eurasian Green Woodpecker
☐ 3	1	1	183.000	186.000	0.0	12000.0	0.3173	European Goldfinch
☐ 3	1	1	183.000	186.000	0.0	12000.0	0.3173	European Goldfinch
☐ 4	1	1	204.000	207.000	0.0	12000.0	0.2561	Black Woodpecker
☐ 4	1	1	204.000	207.000	0.0	12000.0	0.2561	Black Woodpecker
☐ 5	1	1	486.000	489.000	0.0	12000.0	0.2613	Whooper Swan
☐ 5	1	1	486.000	489.000	0.0	12000.0	0.2613	Whooper Swan
☐ 6	1	1	861.000	864.000	0.0	12000.0	0.2806	Rustic Bunting
☐ 6	1	1	861.000	864.000	0.0	12000.0	0.2806	Rustic Bunting
☐ 7	1	1	873.000	876.000	0.0	12000.0	0.4684	Rustic Bunting
☐ 7	1	1	873.000	876.000	0.0	12000.0	0.4684	Rustic Bunting

- For a selected label file, if Raven can’t find a label for a species, the species’ semantic key will be used as Label in the detection/classification output selection table

Appendix I

Specifications for model configuration file model_config.json

This section provides detailed specifications for model_config.json file. For most cases, user can use the template above to create a model_config.json file to support Raven Pro's need. The information below is reference on the full capabilities of the model_config.json file format, many of them are place holders for future developments and improvements.

- model_config.json file includes the following information (see json schema):
    - specVersion
    - modelDescription
    - signatures
        - modelInputs
        - modelOutputs
        - semanticKeys (optional)
    - modelTypeConfig
    - globalSemanticKeys
- File structure specifications:
    - specVersion
        - Integer number, specify the model_config.json file's schema version or specification version number that is supported by Raven.
        - Current supported version: "1"
        - Raven Pro uses specVersion to handle file compatibility.
    - modelDescription
        - string, words or sentences to provide brief descriptions about the model.
        - Example: "Automated scientific audio data processing and bird ID."
        - Raven Pro displays the model description in the learning detector UI when a model is selected
    - signatures: array of signature objects
        - signature: object to specify the configurations of model's inputs and outputs. The information provided in the signature **must match those in the model**
            - signatureName:
                - string, name of a signature of the model
                - Example: "basic" is BirdNET model's one signature's name
                - Used in Raven Pro learning detector UI for signature selection

- modelInputs: array of modelInput objects
  - modelInput: object to describe the model input configuration. It contains the following:
    - inputName:
      - string, name of an input of a model's signature
      - Example: "inputs" is one input's name of a model's signature
      - Used in Raven Pro learning detector UI for model input selection
    - sampleRate:
      - positive number, describes the sound recording sample rate supported by the model.
      - Example: 48000.0 (in Hz)
      - Raven Pro display the sample rate of a selected model's input in the learning detector UI. Raven Pro will display a resample message in UI if the sample rate of a sound file doesn't match the model supported sample rate
    - inputConfig:
      - Array of strings from the enumeration: "batch", "channels", "samples", "blank". It tells Raven how to structure the data of the model input array, e.g. the shape of the input data.
      - Example: ["batch", "samples"]
      - Above example tells Raven sample size can be obtained from index 1 of the model input shape array. Currently, Raven only uses "samples".
- modelOutputs: array of modelOutput objects that describe the model output configurations.
  - modelOutput: object to describe the model output configuration. It contains the following:
    - outputName:
      - string, describe name of the output,
      - Example: "scores"
      - used in Raven Pro learning detector UI for model output selection

            - outputType:
                - string from enumeration “EMBEDDINGS”, “”SCORES”. It tells Raven the model’s output type. SCORES is the only type supported by Raven right now.
                - Example: “SCORES”
                - Raven processes all model’s output scores, checking them against the threshold specified in Raven user interface. For class with score above threshold, Raven creates a selection/detection.
        - semanticKeys:
            - Array of strings, listing all the semantic keys/codes of the species that the selected signature of the model can classify/detect.
            - Raven uses this to pre-filter the list of global keys, since some signatures may only output some of the classes listed there
            - This is optional. Generally, they are a signature specific subset of the global keys (see below).
    - modelTypeConfig: object that specifies the model type. It contains the following:
        - modelType: string from enumeration “RECOGNITION”, “STREAM”. Currently Raven only supports model with RECOGNITION type.
    - globalSemanticKeys: array of strings, listing all the semantic keys/IDs of all the species the model can classify. The list is required and must contain output class keys in the specific order of the output tensors. That is, the **order of the list must match the order of output classes in the model.**

- Example model_config.json (abridged from BirdNET model’s model_config.json file with most semanticKeys removed)

```
{
   "specVersion": "1",
   "modelDescription": " Automated scientific audio data processing and bird
ID.\nDeveloped by the K. Lisa Yang Center for Conservation Bioacoustics at the Cornell Lab
of Ornithology.",
   "signatures": [{
      "signatureName": "basic",
            "modelInputs": [{
                "inputName": "inputs",
                "sampleRate": 48000.0,
                "inputConfig": [
                    "batch",
                    "samples"
                ]
            }],
            "modelOutputs": [{
                "outputName": "scores",
                "outputType": "SCORES"
            }]
        }],
        "modelTypeConfig": {
            "modelType": "RECOGNITION"
        },
        "globalSemanticKeys": [
            "rufwar1",
            "yebwar1",
            "watgua1",
            "spchon1",
            "lesred1",
            "capwhe2"
        ]

}
```

Apendix II

model_config.json schema

The following is the schema generated from the ModelConfiguration class in Raven Pro.

```
{
  "type" : "object",
"properties" : {
  "specVersion" : {
    "type" : "integer"
  },
  "modelDescription" : {
    "type" : "string"
  },
  "signatures" : {
    "type" : "array",
    "items" : {
      "type" : "object",
      "properties" : {
        "signatureName" : {
          "type" : "string"
        },
        "modelInputs" : {
          "type" : "array",
          "items" : {
            "type" : "object",
            "properties" : {
              "inputName" : {
                "type" : "string"
              },
              "sampleRate" : {
                "type" : "number"
              },
              "inputConfig" : {
                "type" : "array",
                "items" : {
                  "type" : "string",
                  "enum" : [ "BATCH", "CHANNELS", "SAMPLES", "BLANK" ]
                }
              }
            }
          }
        },
        "modelOutputs" : {
          "type" : "array",
          "items" : {
            "type" : "object",
            "properties" : {
              "outputName" : {
                "type" : "string"
              },
              "outputType" : {
                "type" : "string",
                "enum" : [ "EMBEDDINGS", "SCORES" ]
                }
              }
            }
          },
          "semanticKeys" : {
            "type" : "array",
            "items" : {
              "type" : "string"
            }
          }
        }
      }
    },
    "modelTypeConfig" : {
      "type" : "object",
      "properties" : {
        "modelType" : {
          "type" : "string",
          "enum" : [ "RECOGNITION", "STREAM" ]
        }
      }
    },
    "globalSemanticKeys" : {
      "type" : "array",
      "items" : {
        "type" : "string"
      }
    }
  }
}
```