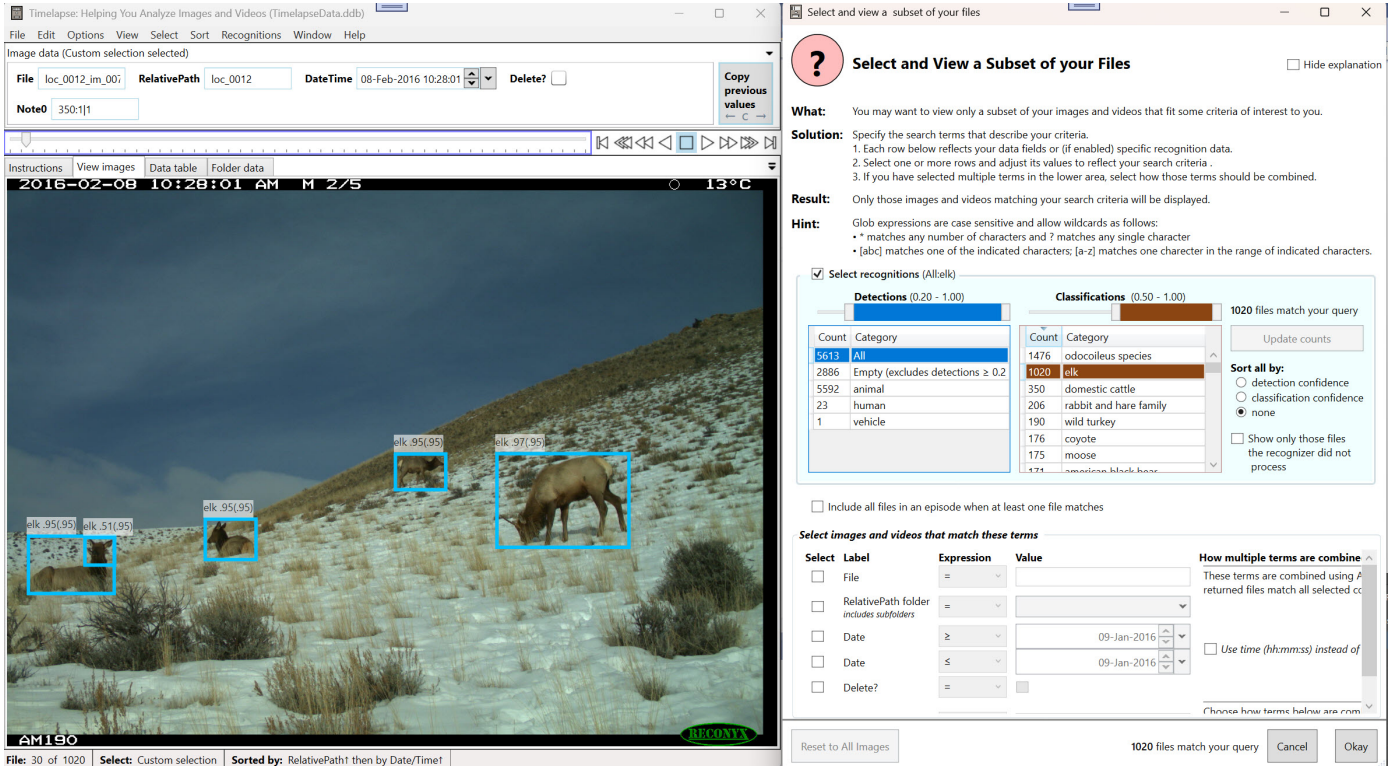
Timelapse Image Recognition Guide

A guide to importing and using image recognition data



Saul Greenberg

Greenberg Consulting Inc. / University of Calgary

saul@ucalgary.ca

Part 4 of the Timelapse Manual series. Last updated May 1, 2025. Timelapse Version 2.3.3.0

Timelapse Image Recognition Guide

A guide to importing and using image recognition data[1]

Don't Panic! The actual process for using image recognition is fairly simple. This guide is lengthy only because it goes into detail (perhaps excessively).

This guide explains how image recognition information is obtained from 3rd party software, and how that image recognition data can be incorporated into Timelapse to make your workflow even faster and more efficient. A companion video illustrates the workflow described here.

A few things you should know ahead of time before jumping into this guide (all presented in more detail shortly).

- ***You should be familiar with Timelapse.*** If not, you should at least read the Timelapse Quickstart Guide.
- ***Other software does the image recognition.*** Timelapse helps you install and run the *AddaxAI* image recognizer app on your image set, where you can choose a species identification model that best fits your needs. After importing recognitions into Timelapse, you will see bounding boxes atop of detected entities in your images and videos, and you can select and tag subsets of images or videos containing particular entities.
- ***Image recognition includes videos*** by periodically analyzing frames as images.
- ***Image recognition isn't perfect.*** It's important that you understand where recognition works and where it can fail. We strongly suggest you read this short document and watch the accompanying video presentation.
  - » Automated Image Recognition for Wildlife Camera Traps: Making it Work for You. Saul Greenberg, Research report, University of Calgary: Prism Digital Repository, 2020.
  - » Video Presentation: Image Recognition for Camera Traps: Making it Work for You. Saul Greenberg. *Conference: Scaling Up Camera Trap Surveys to Inform Regional Wildlife Conservation*, Columbia Mountains Institute of Applied Ecology, May 18-20, 2020. Duration: 20 minutes.

Table of Contents

What is image recognition?

A holy grail of camera trap analysis is automated image recognition, where a computer automatically inspects, analyzes and tags your images. While we are not quite ready to let a recognizer do all the work, image recognition can help ease your workflow, as long as you understand its many subtleties.

Image recognition also includes videos. Video frames are selectively extracted and analyzed as if they were images, where the recognizer output also includes the position of that frame in the video.

Your use image recognition will work best if you some knowledge of how it works: its benefits and weaknesses; how Timelapse displays and lets you select from image recognition results; and the various ways you can incorporate recognition data into an effective workflow.

Internally, recognizers incorporate one or more ***models,*** which it uses to detect and classify entities in your images. These models are created by training it on sample data, such as previously tagged images identifying species of interest in a geographic area. Some recognizers, including AddaxAI, let you choose a model that best fits your purposes.

Two types of image recognition: Detection vs. Classification

Under the covers, a recognizer may include ***detectors*** and/or ***classifiers.***

Detectors detect whether something is present in an image. For each suspected ***detection***, a detector:

- assigns a coarse identifying ***label*** to it (e.g., ***empty***, ***animal***, ***person***, ***vehicle***);
- assigns a ***confidence value*** indicating the likelihood that it is correct
- locates it via ***bounding box*** coordinates, which can be used to draw a rectangle on your image that identifies the suspected entity.

Classifiers analyze the sub-image contained by the detector's bounding box, where it performs a fine-grained classification chosen from a set of known categories such as wildlife species (e.g., ***bear, elk, deer***). A classifier:

- produces one (perhaps more) ***classifications*** for each detection;
- includes a ***probability value*** that roughly indicates the likelihood that the classification is correct.

Image recognition is not full proof

While image recognition can work reasonably well, it is not a magic bullet. Several types of recognition errors can occur depending upon the image recognition algorithm, how its underlying recognition model was trained, how well the images you submit match that model, and the image contents. *[Click to read more details in a separate document - its worth reading!]*.

Recognizer errors fall into these categories.

- ***False positives:*** it identifies an entity as being detected in an image when nothing is there.
- ***False negatives:*** it does not detect an entity when that entity is present.
- ***Incorrect classifications:*** it correctly detects an entity, but incorrectly labels it (e.g. an animal instead of a person for a detection, or a wrong species for a classification).
- ***Ambiguity:*** it detects several overlapping and possibly conflicting detections within an image, or across video frames.

Why use automatic image recognition?

While the ideal recognizer would simply tag all your images for you, recognition errors are a fact of life. Unless the error rate is something you can live with, you should not blindly accept all recognition results. Instead, you should consider recognitions as predictions that can augment, rather than replace, your workflow. As we will shortly see, you can use Timelapse to show only subsets of images matching a particular recognition category. You can then tag each category in bulk while still checking and correcting for occasional errors. This allows you to tag most images quickly, leaving far fewer images requiring closer inspection.

Typical benefits are listed below.

- ***Eliminate empty images from any further analysis***, such as images with no detected people or animals in them or images with very low confidence detections or classifications. This is especially valuable in cases where cameras generate large numbers of empty images, e.g., timelapse mode where images are periodically taken regardless of what is present, or motion-detection mode activated by wind effects on vegetation.
- ***Identify images containing people*** is important if your agency has a privacy policy that requires those images to handled somewhat differently from other images, or when you have a different set of analysis criteria for people vs. animals.

- ***Identify images containing vehicles***. As some camera traps are set near a roadway, vehicles passing by can trigger the camera. If these are not need, they can be selected and eliminated from further consideration.
- ***Identify images containing animals***, where you will inspect those in detail, for example, to classify or count what animals are present.
- ***Focus on a particular species.*** If classification data is available, classifications can let you quickly select and focus on those images containing the desired species. Alternately, you can eliminate species of little interest.

The AddaxAI / Megadetector image recognizer

Timelapse does not do image recognition by itself. Rather, it relies on third party image recognizers do the image processing. In this guide, we focus on *AddaxAI* as it is integrated into Timelapse, and works very well with it.

AddaxAI serves as a front end to Microsoft's ***MegaDetector,*** a reliable image recognition system. *AddaxAI* encapsulates the messy details of installing and using *MegaDetector*. It also lets you choose between pre-trained species classification models for one that best for your needs, including ***Google's SpeciesNet*** general purpose classifier. Using *AddaxAI* is simple. You select a folder containing images and videos. *AddaxAI* then analyzes them and generates a recognition file that you import into Timelapse. Timelapse then displays and lets you select images containing particular recognitions.

MegaDetector is a popular image recognizer for camera traps. It is specifically designed as a high-quality detector for wildlife-oriented camera trap images. Timelapse is normally recommended as the front-end for visualizing *MegaDetector* recognition data, and many organizations have used the two to efficiently tag massive numbers of images.

AddaxAI species identification models. *AddaxAI* contains a list of classification models that will grow over time. Most are specific to a geographic region and set of species, and thus useful only if your own needs match that. ***Google's SpeciesNet*** is the exception, as it is a general purpose model that can be used by most ecologists over broad geographic regions
If these models don't match your needs, you can still run *AddaxAI* with no model selected. It will then generate detections with broad animal/human/vehicle classifications. Even this is very useful, especially to eliminate empty images (such as those triggered through wind effects), or to manage privacy concerns by eliminating images containing people.

Downloading AddaxAI

Downloading *AddaxAI* is straight forward. It will likely take 10-15 minutes to install, as it requires various packages to do image recognition. You can still do other work during its installing.

1. Select *Recognizer |AddaxAI Image Recognizer | Install AddaxAI image recognizer (via browser)...* An instructional dialog box will appear. Timelapse will also tell you if *AddaxAI* is already installed.
2. Click *Okay* to continue. A browser should appear displaying a *Windows Install* web page. If it doesn't, go to https:/addaxdatascience.com/addax-ai-windows. After reading the instructions, press *Download installer* on that page.
3. A command prompt window appears that may ask you a few questions. Choose whatever answer sounds right for you. Then let it run! You'll see lots of text scroll by. For the most part, you can just ignore all of it until it says its done.

Note. If your computer is managed by your organization, your IT people may have set your computer to prohibit installing applications. If *AddaxAI* does not install, ask your system person to do it for you. Similarly, anti-virus, VPN, proxy servers or other protection software might interfere with the installation. If that happens, you may want to turn those off. Contact the *AddaxAI* developer if you still have issues, as he may be able to help you get around them.

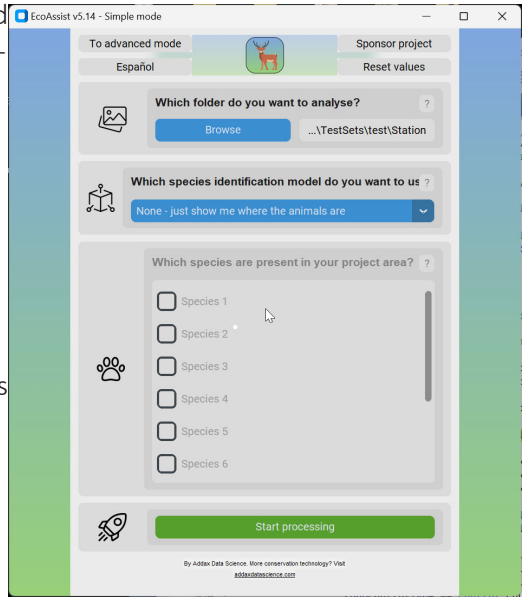
Having said that, installing *AddaxAI* on my own personal computer and laptop went flawlessly. Hopefully, you will have a similar experience.

Running AddaxAI and using its results

Running *AddaxAI* is straight forward.

1. Open Timelapse on an image set, and select *Recognizer|AddaxAI Image Recognizer|Run AddaxAI on this folder...*
2. Select the folder containing images (possibly located in sub-folders) that you want to recognize. Usually,
   - » select the root folder of your image set if you want it to recognize all images in your image set; or
   - » select a particular sub-folder if you want it to recognize just those images (e.g., you may have already run the recognizer on your images, where you now want to just run it on a folder containing new

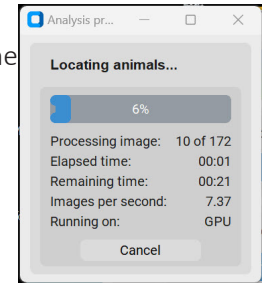
images that were retrieved from a camera).An instructional dialog box appears telling you that ***AddaxAI*** is starting up. Click *Okay*. The ***AddaxAI*** window will appear after several seconds. It asks which species identification model you want to use.



» ***None***: will show detections with broad animal/personal/vehicle classifications. Choose this if the other models don't seem to fit what you are looking for. Or,

» ***Choose a species identification model*** using the drop-down menu (optional). It then lists species known to that model. While you can select all of them, its best to select only those you know are present in your area. Choosing a model may initiate a further download. If so, just follow the instructions.

3. If your image set also includes videos, ***AddaxAI*** will, by default, sample a single frame for every second of video. For example, if your video is 15 seconds long and runs at 30 frames per second, ***AddaxAI*** will sample every 30th frame and run the recognizer on those frames, generating a total of 15 recognitions for your video.
4. Click *Start processing***.** A small progress dialog will appear. When done, ***AddaxAI*** will tell you where the recognition file is located (it should be in the same folder that you asked it to recognize).



5. Select ***Recognizer|Import recognition data for this image set…*** to load the recognition file into Timelapse. Timelapse will then display recognized entities, if any, within bounding boxes.

The recognition file. After processing your images, ***AddaxAI*** generates a file called *timelapse_recognition_file.json*. This file contain recognition data for every image file found in the folder you chose in Step 2. That data also includes the image file's *relative path* and *file name*, which Timelapse uses to locate the file. Consequently, keep the recognition file in that folder. If you move it to another folder, Timelapse may not be able to find your image files as the *relative path* would not match.

Read the *Working with JSON recognition files* at the end of this chapter. It explains how to include recognition files in your workflow. It also describes how you can create multiple *recognition* files, each representing a sub-folder within your root folder that you can add incrementally as new images become available over time. It also helps trouble-shoot cases where Timelapse cannot find the files specified in the recognition file.

Advanced mode. *AddaxAI* includes a *To advanced mode* button (top left corner), where you can adjust certain parameters. While we suggest accepting the defaults, reasonable adjustments include:

- *Using checkpoints while running* will intermittently save the recognition results when running on a large number of files. This will let you continue more or less from where you left off if the recognizer unexpectedly crashes in the middle of a long run.
- *Process videos / Process images if present* are normally both checked. You can uncheck these to process only videos or only images if desired.
- *Video options: Sample frames every N seconds* is normally set to 1 second. You can over-ride this to whatever amount you want, e.g., .5 seconds will sample 2 frames for every second of video, while 3 seconds will sample a frame every 3 seconds. The trade-off is performance vs bounding box accuracy during playback (see below).
- *Video options: Don't process every frame* if unchecked (the default), will generate a recognition for every video frame.

Video sampling rate. Timelapse is tuned to work well with the default 1-second video sampling rate. Higher sampling rates will result in longer times to run your recognizer. Lower sampling rates will mean that your recognitions will appear intermittently in a running video. For example, if you decide to recognize every frame, 30 seconds of video at 30 frames per second means that the recognizer will actually be processing 900 images. This will be slow, will produce a huge recognition file, and will only provide slight additional benefits when reviewing those videos when compared to its 1 second default sampling rate.

AddaxAI and the Practice Image Set

The *PracticeImageSet*, downloadable from the Timelapse web site's Download page, contains a template and several folders. Download and unzip it to a convenient place.

Later sections will use the *Station4* folder to illustrate workflows. That folder's mix of images and videos (collected from various camera trap locations) was chosen to illustrate image recognition capabilities and foibles. For convenience, it also includes previously generated *.json* recognition files.

Exercise: Generate your own recognition file

As a first (optional) exercise, let's use ***AddaxAI*** recognizer to recognize images in the Station1 folder in the *Practice Image Set*, and then display the results in Timelapse.

1. Start Timelapse. If you haven't already done so, install ***AddaxAI*** as described in the previous section.
2. ***File | Load template, images and video files...*** to load the Practice Image Set as explained in the previous section
3. ***Recognitions | AddaxAI Image Recognizer|Run AddaxAI recognizer on a folder...*** and select Station1.
4. When the ***AddaxAI*** interface appears:
   - » Select the ***Global - SpeciesNet - Google classification model*** (found under '*Which species identification model do you want to use?*'
   - » Click the ***To advanced mode*** button at the top left. Under ***Identification options***, select the location US-ID (Idaho) from the list.
   - » Scroll down to the bottom and select ***Start processing***.
5. A dialog box will appear showing its progress and eventually that processing is complete. Close ***AddaxAI.***
6. ***Recognitions | Import recognition data for this image set***, and select the timelapse_recognition_file.json found in the Station1 folder.
7. Navigate to the first image.

You should see something similar to the screenshot on the next page.

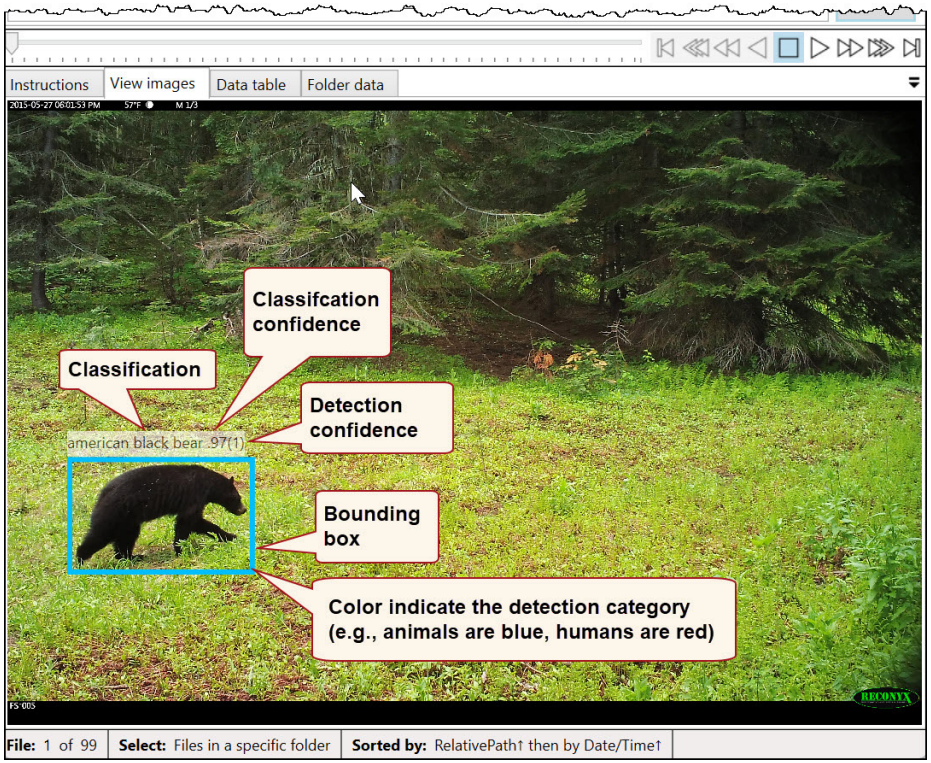
How Timelapse interprets recognition data

After you import the recognition file into Timelapse, you will be able to view recognitions atop your images. You will also be able to streamline your workflow by selecting particular image subsets based on these recognitions, as discussed shortly. The images below are selected from various image sets.

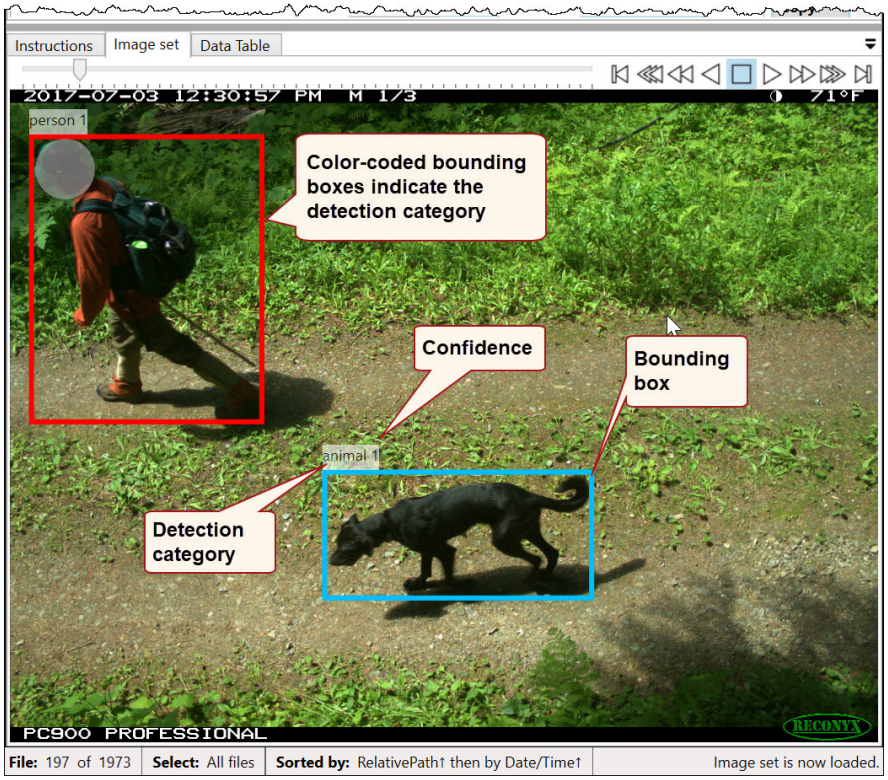
Recognitions are displayed as labeled bounding boxes

Timelapse uses the data in the recognition file to display each detected entity as a colored bounding box drawn atop each image. Each box:

- is labeled by its classification (e.g., bear, elk, goat) or (if a classification doesn't exist) its more generic detection category (e.g., person, animal);
- is colored to distinguish the detection category (e.g., red for person, blue for animal);
- includes confidence values of its guess for its detection and (if present) its classification ranging from 1 (high confidence) down to 0 (low confidence).



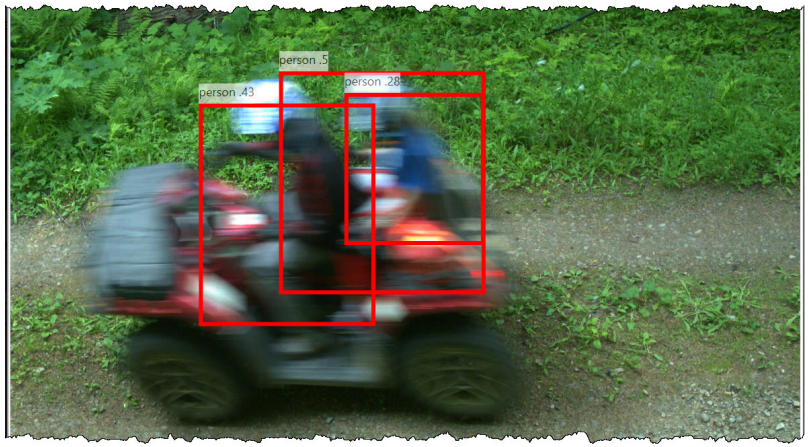
If a classification model was not selected in ***AddaxAI***, recognitions would be labeled by detection categories. This image, for example, shows two correct high confidence detections within its labeled bounding boxes. One contains a detected person (confidence score = 1, red box). The other correctly identifies a detected animal (blue box, confidence score = 1).



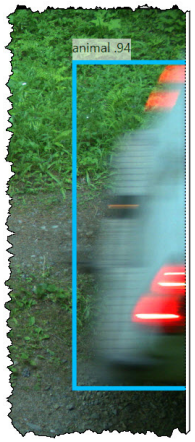
Note 1. Bounding boxes can be temporarily hidden by pressing and holding the **H** key while your cursor is over the image. This is helpful if you need to inspect a portion of the image directly under the bounding box border.

Note 2. Select *Recognitions|Set bounding box options...* to set a bounding box display threshold. Timelapse will display bounding boxes only for detections above that threshold. This helps reduce clutter.

The following image has detections of several people, although at lower confidence. It also produces somewhat ambiguous results as it shows various possibilities of what it thinks is a person. For example, you cannot simply count the bounding boxes, and assume that there are three people in the image.



Other errors are shown below: a vehicle labeled as an animal, an animal detected when nothing is there, and missed wildlife.



Incorrect identification

False positive

False negative

Displaying classifications and confidences

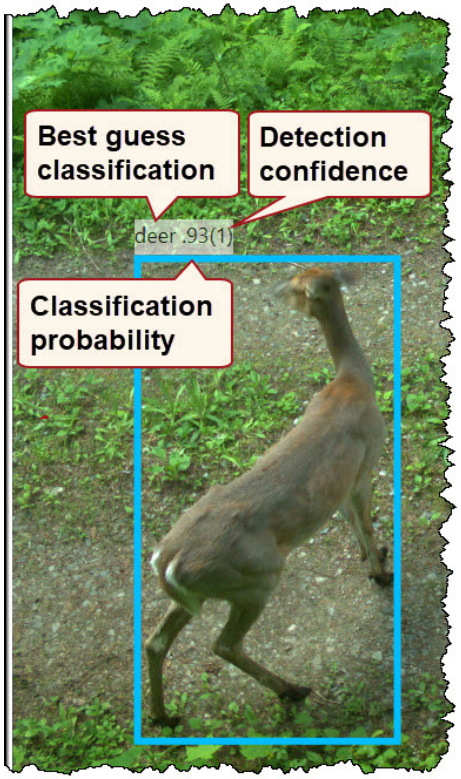
If a classification model was selected in ***AddaxAI***, recognitions would be labelled by its classification, although detection information is still available.

This image correctly classifies the detected animal as a deer. Timelapse labels the bounding box accordingly. The bounding box also uses the detection color (blue for animal).

The two accompanying numbers provide confidence values: the classification probability is .93, while the dete tion onfidence is (1).

The 'animal' detection label is not included, as it is implicit by its classification (a deer is an animal).

If no classifications are available, then the label would only shows the recognized detection category (e.g., Person, Animal, Empty).



Note. How to interpret detection vs. classification confidence. *Detection confidence* is the certainty that an entity has been correctly detected and located in an image. *Classification confidence* is the certainty that the entity within that bounding box has been correctly classified. Given an animal detection classified as a deer, these numbers would be interpretted as:
- .93(1): the detector was fairly certain that an animal was correctly detected (conf =1), and the classifier is fairly certain that it is a deer (.93).

- .4(.9): the detector was fairly certain that an animal was correctly detected (.9), but the classifier is somewhat uncertain that it is a deer (.4).

- .9(.2): there is a reasonable chance that the detection is a false positive (.2), where the classifier's best guess is that, if anything, it is likely a deer.

The meaning of confidence levels

Confidence is, at best, only a rough indicator of how likely it is that a detection or classification is correct. As explained in the technical note on the next page, he reliability of a particular confidence value varies greatly from recognizer to recognizer, and from dataset to dataset. .

To mitigate this variability, the ***json*** recognition file specifies a ***typical detection threshold***, and a ***typical classification threshold***, where a confidence and probability value is normally interpreted relative to those thresholds.

- **At or above the threshold**: the recognition is likely mostly correct, although the occasional error may occur.
- **Somewhat below the threshold**: recognitions are somewhat more suspect, where increasingly more false positives will appear.
- **Far below the threshold**: the lower the value, the more likely the recognition is wrong.

Timelapse uses the various thresholds in the json file to set defaults for:

- when to display bounding boxes, where detections below a confidence threshold are hidden;
- in the ***Select | Custom select*** dialog, what default confidence ranges are best used for selecting images that contain a particular recognized entity.

Because the thresholds in the ***json*** recognition file are not completely reliable, you can optionally set the bounding box display thresholds, and confidence values used to select a confidence range to ones that better fit your images.

Technical note

Confidence of detections vs. classifications. Each detection has a single confidence value indicating its relative likelihood that it is a correct detection. Classifications differ. As the classifier produces 1 or more predictions per detection, it assigns each classification with a relative probability of being correct, normally summing to 1. The classification probability is used as a rough indicator of confidence, i.e., how likely it is that the entity was correctly recognized.

Confidence values and 'mostly correct' thresholds. Different recognizers, and even different versions of the same recognizer use different strategies for generating confidence values. As such, the confidence threshold above which detections and classifications are 'mostly correct' can differ greatly. For example, here are differences between two MegaDetector versions:

	MegaDetector version 4	MegaDetector version5
Typical detection threshold	0.8	0.2
Typical classification threshold	0.05	0.75

As these values depend on the recognizer, version-specific values are included in the .json file. Timelapse then uses those values to determine the default confidence levels in the ***Select*** dialog.

Even so, these default values are (at best) an approximation. The best way to judge a reasonable detection or classification confidence level is by reviewing your images to get a sense of how well recognitions perform at different levels.

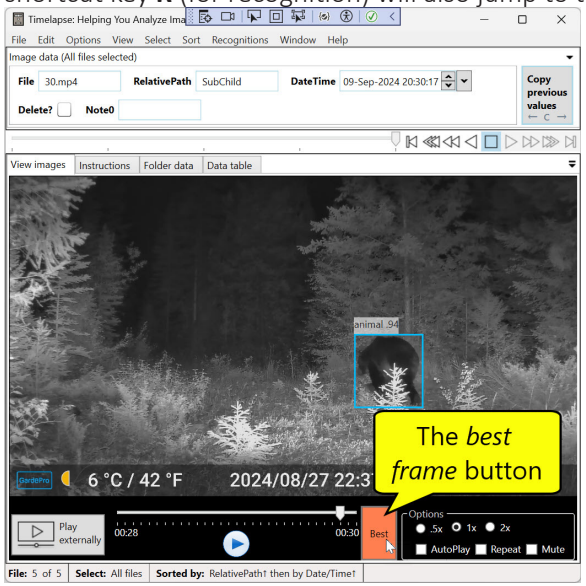
Bounding box display threshold. In the *Recognitions|Set bounding box options...* dialog(see next section), the initial default value for displaying bounding boxes is calculated using the detection confidence threshold specified in the image recognition ***.json*** file. As this value can vary considerably, we suggest that any adjustments of the bounding box display threshold should be relative to this default value. Each image set remembers this preference, where it is stored in your ***.ddb*** file.

Bounding boxes in videos

Bounding boxes also appear in videos, albeit with a few differences.

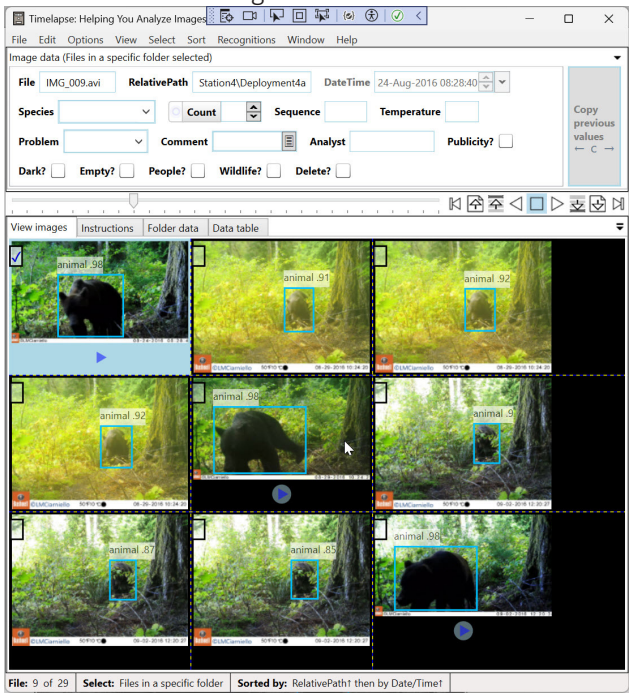
Bounding box appearance varies with sampling rate. Your bounding boxes will animate as you play or scrub your video. As videos are normally sampled only once a second, expect bounding boxes to jump around somewhat from its old to its new location, or even appear/disappear depending upon what was recognized in that sampled frame. See *Page 5, Video Sampling Rate.*

A 'Best' button lets you quickly navigate to the video frame with the highest confidence recognition. The button will appear only on videos that can display at least one recognition within your selected confidence range. The button will be rose-colored if its on the best frame, and red if it isn't. A shortcut key **R** (for recognition) will also jump to the best frame.



Note. The *best button* may only partially reveal what is going on. Consider the highest confidence frame that captures a single bear walking away (as above). Other video frames, recognized at a slightly lower confidence, may show that bear in greater fidelity, reveal interesting behaviours, or even reveal the presence of other bears or animals. Treat the highest confidence frame as hint to the video's contents. We recommend playing or quickly scrubbing the video to see if there is anything else of interest.

The overview will display the best frame in each of the video thumbnails. This can help you quickly determine those videos have good recognitions within it, and which do not (e.g., those considered empty or whose confidence value is below the threshold for displaying them). The example below includes a mix of images and videos.



Video problems happen. Video playback is complex and somewhat out of Timelapse's control as it relies on the internal Microsoft Media Player and properly installed video codecs. Problems you may see include:
- videos aren't displayed (you see only black),
- videos won't play, or stutter, or freeze.
- some videos will play fine, but others won't.

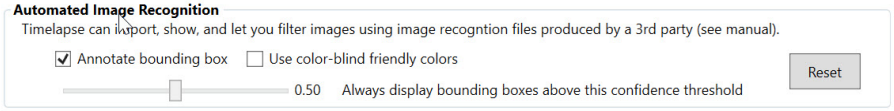
The **Timelapse FAQ page:***Video Playback* details how to update your system to better play videos. This includes:
- how to install the *Microsoft Media Player* used by Timelapse,
- how to download and install updated video codecs.

If a single video is problematic,
- press **F5** to refresh the video, or
- select **Play externally** to view the video in an external video player. However you won't see bounding boxes or be able to zoom into the video.

Setting bounding box preferences

You can customize how and when your bounding box appears in the preferences dialog (or just accept the reasonable defaults).

- select ***Recognitions|Set bounding box options...*** to set a bounding box display threshold;
- Click the ***Annotate bounding box*** checkbox to either include or exclude the recognition labels atop the bounding box.
- If you have color blindness issues, the ***Use color-blind friendly colors*** will adjust the bounding box colors accordingly.
- Use the slider to adjust the confidence threshold, where bounding boxes are displayed only for recognitions whose confidence is at or above that threshold. In the example, below, only detections with a confidence of .5 or higher will be displayed. This is discussed in more detail in the technical note on the next page.
- Pressing ***Reset*** restores these settings to their default values.



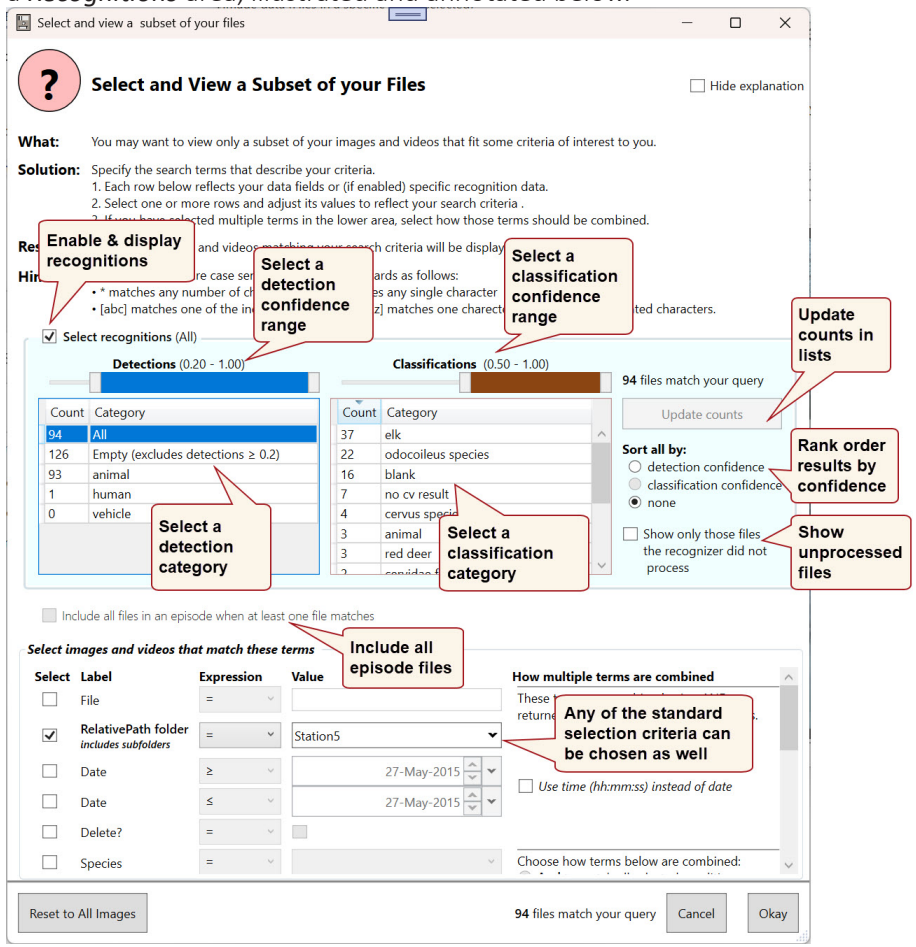
Experiment with the ***Always display bounding box*** setting by reviewing your images. While it is tempting to set the threshold to a low value, it could add considerable visual clutter by showing bounding boxes around incorrect recognitions. Alternately, setting it too high will mean that some correctly detected entities will not show their bounding boxes. The best value will likely be at, or somewhat below or above the default threshold.

Workflow tip. As discussed next, you can use ***Custom Select*** to selectively view as subset of images that include detections and classifications at a given confidence range. When you selected images using a confidence range lower than the threshold set in the ***Preference*** dialog, Timelapse will display bounding boxes down to that lower confidence range (because you are interested in those images), regardless of the ***Preference*** dialog setting.

Selecting images fitting a recognition criteria

As discussed in other Timelapse guides, ***Select | Custom Selection*** displays a dialog box that allows you to select a subset of your images to view according to some criteria.

When a recognition file has been imported, that dialog box can also include a ***Recognitions*** area, illustrated and annotated below.



The recognition features of this dialog are as follows.

- ***Select Recognitions*** checkbox, when checked, will enable and display the recognition controls. This is necessary if you set recognition criteria for selecting files. Its text also provides feedback on the current selection e.g., (All) indicates a selected detection, while (All-elk) indicates that a classification is also selected.
- ***Detections*** are a selectable list of the available detection categories.
  - » *All* – the image includes a detection of one or more animals or people within the given confidence range. All excludes images where the image recognizer has not identified any detections, even if the confidence is set to 0.
  - » *Empty* – allows you to rapidly select and inspect images where the image recognizer has not identified any detections, or that contain only low-confidence detections.
  - » *Animal*: the image includes a detection of one or more animals within the given confidence range
  - » *Person*: the image includes a detection of one or more people within the given confidence range
- ***Classifications*** are similar, where they provide a selectable list of the available classification categories.

  Note. As detection and classification categories are specified in the recognition, the categories you see may differ substantially from these. However, Detections will always have an All and Empty category.
- ***Detection and classification confidence ranges*** are adjusted by the range sliders. They allow you to specify a particular recognition confidence range of interest for each entity type. As discussed in the technical note, its default values are calculated from the typical detection and classification thresholds.
- ***Sort all by:*** includes several options. When ***detection*** or ***classification confidence*** is checked, all images of the selected entity type will be returned (ignoring the range slide settings), but sorted by confidence value. That is, the first images will be the ones where the recognizer has the highest confidence of correctness, followed by images with progressively lower confidence of correctness. For images with more than one detection, the highest detection or classification confidence value is used. Sorting by detection confidence sorts images by their detection confidence values, while sorting by classification confidence sorts them by their classification confidence. Sorting lets you quickly scan through images by confidence, where you can roughly determine a value confidence threshold below which the recognizer seems less accurate.
- ***Show only those files the recognizer did not process*** will select images that were not processed by the recognizer. This is useful for separating a mix of images, where some were processed but others were not.

The examples below illustrate working with confidence. Understanding the subtleties – especially for ***Empty*** – will be helpful to using recognition data effectively. The actual useful ranges to use depends heavily on your recognizer and data set.

- ***Entity = Empty, confidence range is 0 - 1*** selects all images where the recognizer has not detected an entity within it. You may use this to rapidly inspect (and occasionally correct) images that the recognizer thinks has nothing in them
- ***Entity = Empty, confidence range is .2 - 1*** is broader, where it now includes all images with a detection confidence excepting those that have a detection ≥ 0.2 (i.e., images with a confidence < 0.2). You may use this to rapidly inspect (and occasionally correct) images with either nothing in them, or where its low confidence detection was likely a false positive.
- ***Entity = Empty, confidence range is 0.5 - 1*** is similar, except that it broadens the range to exclude images with a detection ≥ 0.5. This set may include a few more empty images, while increasing the risk of including an image with a correctly detected entity.
- ***Entity = animal, confidence range is 0.5 – 1.0*** displays images containing at least one detected animal at 0.8 confidence or above (i.e. high confidence animal images that you may want to rapidly inspect for correctness). You may use this to rapidly inspect images to ensure animals are in it, and correct the occasional error.
- ***Entity = animal, confidence range is 0.2 – 0.8*** displays images containing at least one detected animal between the 0.2 – 0.8 confidence range. You may use this to rapidly inspect detections that are more error-prone.
- ***Entity = all, confidence range is 0.0 – 0.2*** displays images containing at least one detected entity up to 0.2 confidence. You may use this to rapidly inspect and eliminate detections that are likely wrong.
- ***Entity = Bear, confidence range is 0.8 – 1*** selects all high probability bear classifications. You would use this a part of a workflow to perhaps check and quickly label those images that are highly likely to contain a bear.
- ***Sort by all*** is useful in your workflow for determining – somewhat qualitatively – the threshold where the recognizer no longer provides generally useful recognition results. You can then set the confidence sliders to select the mostly correct range.

A recognition workflow using detections

Note. The following workflow description uses the *Station4* folder in the *Practice Image Set*, which contains only a small set of images and videos collected from a variety of camera traps. It also illustrates the power of using only detections, e.g., if no classification model was available for your desired region or species.

This first workflow only uses detection information. Its purpose is:

- *identify and eliminate 'empty' images* captured via false triggers or by a camera set in Timelapse mode;
- *identify images with people in it* for privacy purposes,
- *classify and count the animal images* by species.

The workflow also assumes you have a low tolerance for errors, where you need to thoroughly inspect and correct recognitions for errors.

The general recognition workflow strategy is:

- select a subset of your images to display based on the recognizer's predictions (via ***Select | Custom Selection***),
- bulk-tag that sub-set in a single action
- rapidly review those images for tag accuracy and correct any errors.

While it may seem like the following steps adds work, you will find that – depending on your images – you can quickly reduce the number of images you need to manually inspect and tag to a small subset of the original files.

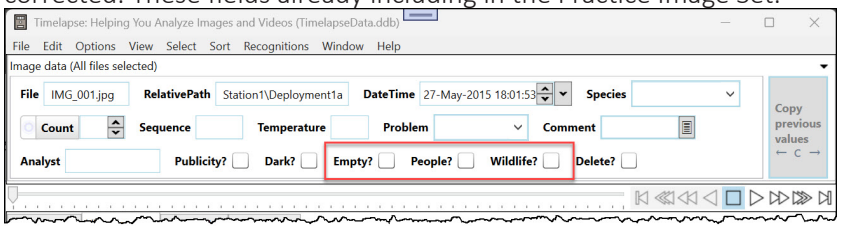
Workflow Tips:
The efficiency of using a recognition workflow can be substantial when used over large and very large image sets. The small demonstration set in the *Station4* folder is just there to illustrate the process.

This workflow is just one of several possible, where the 'best' one will depend on the nuances of your work. A subsequent section describes possible variations that may better fit different needs.

We fully expect you to create your own workflow protocol that fits your own needs: these workflows are included just to give you a flavour of the possibilities.

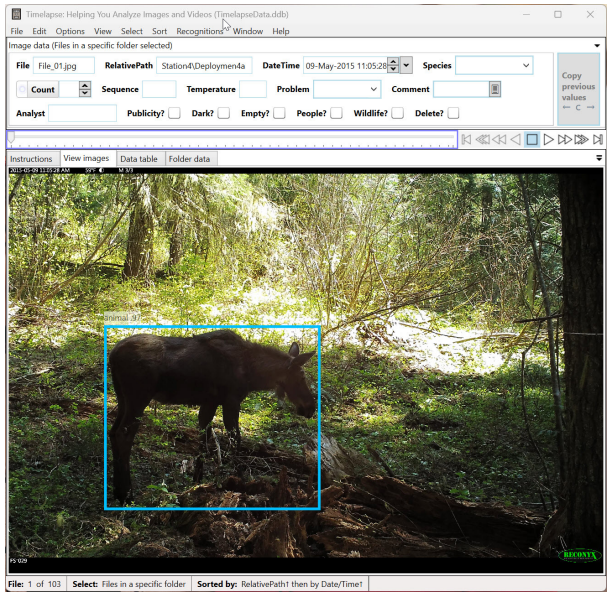
Preparation

1. Three flag fields should be included in the template, named something like ***Person***, ***Wildlife***, and ***Empty***. Their role is to track and bulk-tag what detections have been accepted and corrected. These fields already including in the Practice Image Set.



2. You would normally generate a *.json* detection file in the desired folder using ***AddaxAI*** (classification models are not required for this workflow). One has already including in the Station4 folder of the Practice Image Set, called *timelapse_recognition_file_detections_only.json*.
3. Load in your image set into Timelapse as normal.
4. Select ***File|Import image recognition data for this image set*** to import the .json file.
5. Select ***Custom Selection | All files in a folder and its subfolders | Station 4*** to narrow your selection to just the images in the Station 4 folder.

You should now see bounding boxes atop the Station4 images and videos, each box indicating where the detector has spotted an entity.



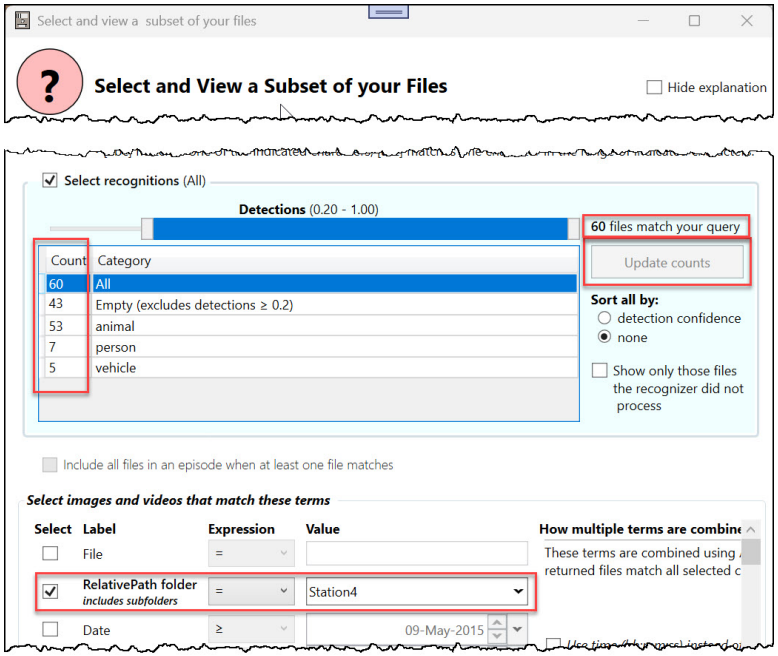
Select and filter out empty images

Your goal in this step is to identify empty images (i.e., where the detector has not detected anything in it) and remove them from any further consideration, while correcting any detection errors. You could also use the same step to permanently delete those images from your data set.

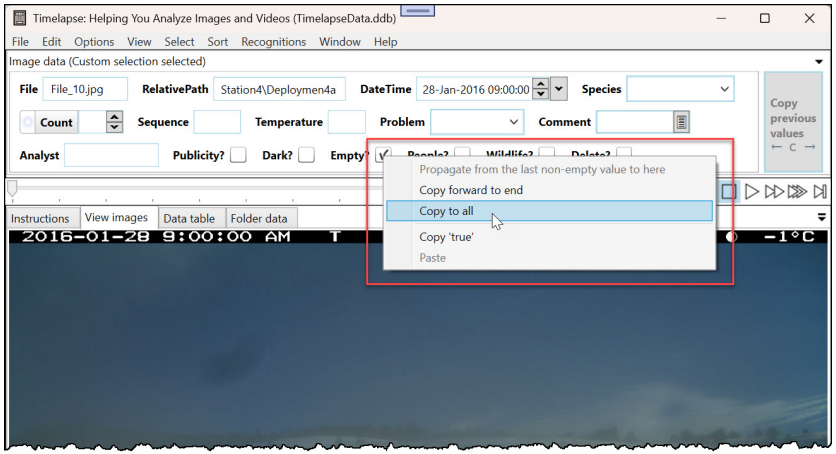
1. Select ***Custom Select***, and click the ***Select recognitions*** checkbox.
2. Click ***Update Counts*** to see a preview of how many images were recognized in each category within the specified confidence range. For example, the results indicate that the recognizer identified 53 images containing an animal within the 0.2-1 confidence range, 7 containing people, 43 that are likely empty, etc.

Note. Counts are based on the current *Custom selection* settings. As the *RelativePath folder* = Station4 is set, counts only includes Station4 images.

Because counting all categories can be slow, you will have to update counts manually using the *Update counts* button. However, the count shown above the button is updated automatically as you change a *Custom Select* field: it reflects the number of images to be returned in the current selection. The same number is repeated at the bottom of the *Custom Select* dialog box



3. Selected ***Empty*** in the Detections list. This will then display the 43 images that either have no detections, or that have very low confidence detections (i.e., < 0.2). While we would expect most of these to be empty, they have to be checked.
4. Tentatively tag these 43 images as empty by setting all of their 'Empty' fields to true.
   a. Checkmark the ***Empty*** flag of the current image.
   b. Select ***Copy to all*** on the ***Empty*** flag (right-click raises its context menu).



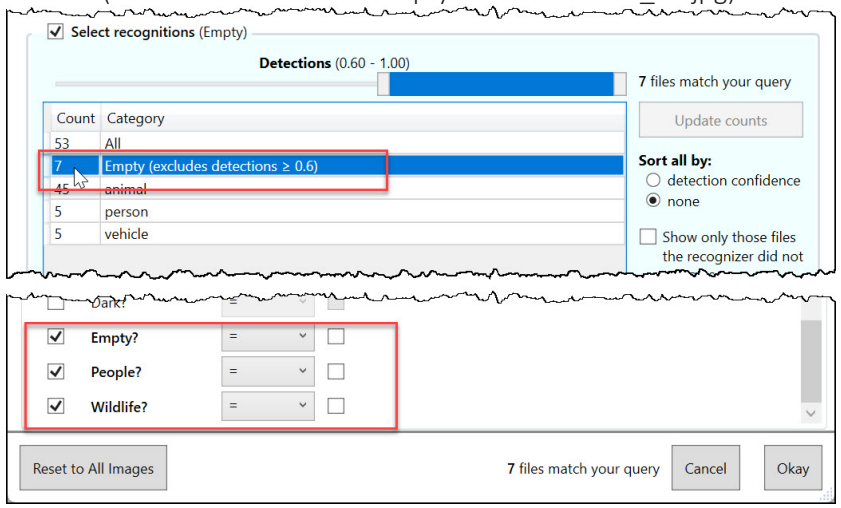
5. Starting from the 1st file, review the images until you are satisfied that the detector was reasonably accurate at identifying images without animals or people in it. You can do this efficiently by using one or more of these strategies:
   » use the ***FilePlayer*** to automatically play the images at various speeds. ***Options|Adjust File Player Speeds*** controls playback speed.
   » use the ***Overview*** to display multiple images at a time, where you can also use the ***FilePlayer*** control to scroll through pages of images;
   » use the arrow keys to rapidly move through images.

   Reminder. Custom selection will show the last previously selected image, or the image closest to it, which may not be the 1st image. Remember to scroll back to the first image before reviewing them, as otherwise you will miss some of them.

6. Correct erroneous ***Empty*** detections as needed. If the detector is working well, the vast majority of images you see will be empty. If you do see images that contain animals or people (false negatives), you can uncheck the ***Empty*** field for those images. At the same time, you can broadly classify those by checking their ***Person*** or ***Wildlife*** field and (if you want) set its species. For example, the last two images (File_72.jpg and File_73.jpg) contain a very small rodent (middle right- these images were intentionally included in this image set to illustrated missed detections).

   Workflow tip. Create two ***QuickPaste*** entries to do the above with a single mouse click. The first clears ***Empty*** and sets ***Person***, while the second clears ***Empty*** and sets ***Animal***.

7. You can alter the detection confidence ranges to search for more empties. For example,
   » set the Detection range slider to 0.6- 1
   » select the following fields so that the selection only includes those images that have not yet been classified as Empty, Wildlife, or People images
   » repeat step 6 to tag, review and correct those images as needed (all but one are indeed empty: a deer is in File_75.jpg).



Workflow tips.

Tip 1. Depending on your tolerance for errors, you may want to review all images carefully and correct for false negatives, or just look for glaring errors and correct those. Alternately, you may want to stop reviewing your selected images at some point if you feel the detector's accuracy is good enough for your needs, and just tag all of them.

Tip 2. With very large image sets, you can *Select | Create a random sample from the current selection* to get a sense of how well the particular recognition selection is doing. If it appears mostly correct and you can accept the occassional error, you can then decide to bulk-tag everything. Alternately, if there are many errors, you can readjust your confidence thresholds to a lower range.

Tip 3. This particular image set did not have many empty images in it. In practice, images sets can include an extreme number of empty images caused by false triggers (wind effects on vegetation) or because the camera is set to Timelapse mode. Some users have reported that the above steps can help them remove 90% or more of their empty images from further consideration, which is an incredible increase in efficiency.

Tip 4. An alternate strategy: Select *Empty* and clicked *Sort by: detection confidence*. Working backwards from the last image, you could then tag the empties until you started seeing more and more images that have correctly detected entities.

Tip 5. Another alternate strategy uses the range slider. Move the ***lower*** confidence of the range slider slowly, and monitor the count of how many files match that confidence range and the current selection. Use that information to decide where you should set the confidence ranges. For example, if only a few images match a particular range, you may want to expand the range to generate more images.

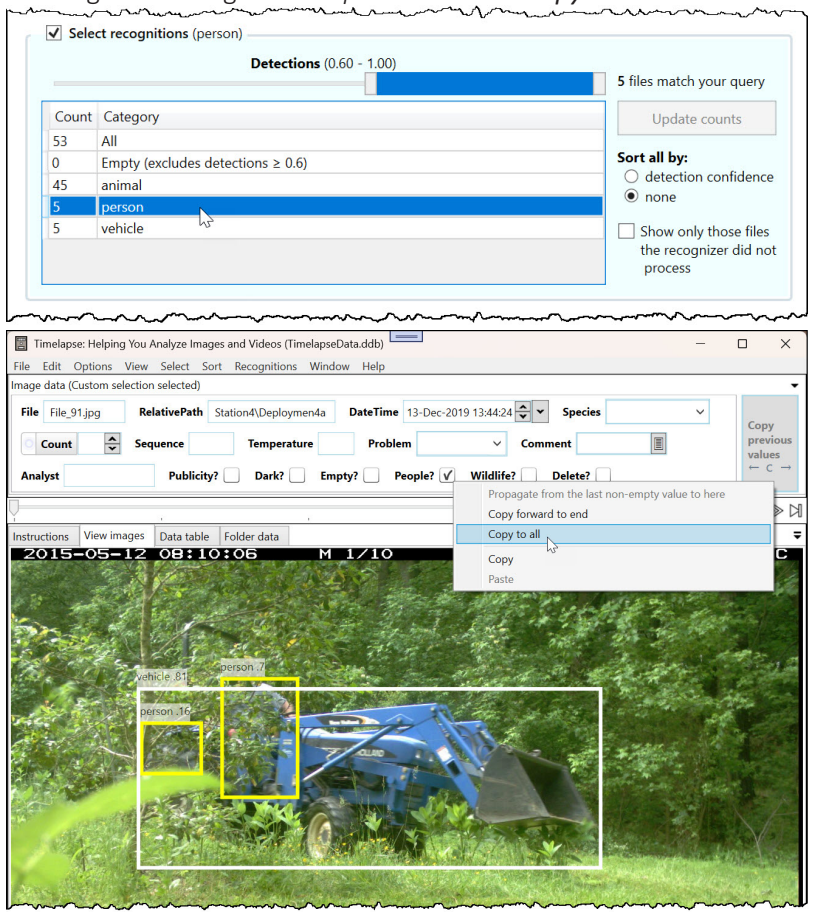
Tip 6. Deleting empties. Empties can take up considerable space on your disk, as well as add clutter to your collected data. While we will not do it here, you can delete all images and corresponding data tagged as Empty by selecting the Empty = √ and then selecting from the Edit | Delete... menu to delete those selected empty images.

Mark images with people and vehicles in them

Your next goal is to identify as many images as possible with people (and vehicles) in them. The process is similar to how we dealt with empty images.

Reasons vary. Some organizations require, usually due to privacy policies, that images with people in it be identified and even discarded. Alternately, you may be counting people in your images, or you may not be interested in images with people in the context a wildlife survey.

8. Select the images where the detector has a reasonable confidence that a person is in them, as illustrated below. Keep the Empty? People? and Wildlife? checkboxes in the custom selection dialog selected, as this will omit images that you have already classified. Notice that videos are also included. In this case, all selected images and videos had people in them.
9. Bulk tag those images as *People?* and select ***Copy to All.***



10. Review this subset (remember to start at the first file). If you see an image that does not contain a person (false positive), clear its ***Person*** flag where you can then set its ***Empty*** or ***Wildlife*** flag accordingly.

Again, depending on your tolerance for errors, you may want to review all images carefully and correct for false positives, or just look for glaring errors and correct those. Or, you may want to stop reviewing the images if you feel the detector's accuracy was good enough. As you've previously set them all to ***Person***, you don't have to do anything more.

11. For completeness, repeat the same step but selecting for Vehicles. Similar to step 7, also select Empty, Wildlife, or People checkboxes set to unchecked so that previously classified images are not included. Mark these as also containing people.

Mark images with wildlife in them

Your goal in this step is to identify as many images as possible with animals in them for later classification. It is identical to the way you tagged images with people in it.

12. Repeat the above process, but this time selecting the Animals option. An occasional false positive may appear (you should find one). The final video does not have an animal in its first frame, but if you select the 'Best Rec'n' button, you will see that the video indeed does have an animal in it.

Note that some bounding boxes reveal animals that could have been easily missed without them. Bounding boxes can hint that these should be investigated more closely, for example by magnifying them or by *Options | Temporarily adjust image appearances | Gamma correction* to increase contrast. These include:

- camouflaged animals or animals in dark areas
- far-away and / or small animals,
- partially obscured animals either hidden in the underbrush, or where only a small portion of them is in the camera's view.

Check for unclassified images

13. To see if you missed any images, check the ***Person***, ***Animals***, and ***Empty*** checkboxes with values unchecked. If the count is not 0, manually tag them. For example, you should find two files that have not been tagged (these two files were not processed by MegaDetector, and would have appeared if you had selected ***Show only those files the recognizer did not process***).
14. To make sure you have classified all images, unselect the *Select recognitions* checkbox, and make sure that the Empty? People? and Wildlife? checkboxes are selected with unchecked values. This will select all remaining unclassified images in the Station4 folder. If the count is not 0, manually tag them.

Workflow tip. You could have started this process by selecting Animal, and then checking the ***Rank by confidence*** checkbox. The selected files will then be sorted from the highest to lowest confidence. Scrolling through them quickly then gives a sense of where the detector doesn't do as well, which would hint at what confidence intervals would give the best results. The benefits of this are clearer when used in larger image sets.

Detailed classification

You have now broadly classified the ***Empty***, ***Person*** and ***Animal*** images.

15. To classify the ***Animal*** images by species, use ***Custom selec***t, and check ***Animals*** = checked. All the tagged Animal images (which could be a considerable number) will now be selected, where non-animal images are excluded. You can now go through the standard manual tagging mechanism to set the species and other pertinent information for these animal images.

Workflow tip. Using the above workflow can significantly reduce the number of images requiring close inspection. In practice, many image sets can have a massive number of empty images in them: 90% empty is not unusual. Similarly, the above workflow can quickly separate images with people in them from other images.

Discarding un-needed images

Camera traps can collect many images containing nothing of relevance. Some agencies keep all images regardless of their value, while others prefer to only keep relevant images. The next few steps apply if your agency prefers to discard irrelevant images, such as those that are empty, and/or - perhapse for privacy purposes- those containing people.

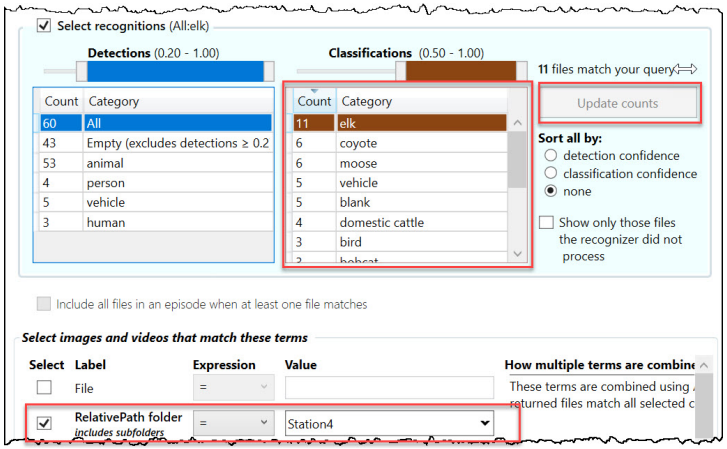
16. To discard empty images (if desired)
    - Select the ***Empty*** checkbox
    - In your data entry area, check the ***Delete?*** flag
    - Right-click that flag to raise the context menu, and select ***Copy to all***.
    - Use the ***Delete*** menu to delete all those files (and optionally its data).
17. To discard People images (if desired), repeat step 20 expect with ***Person*** selected.

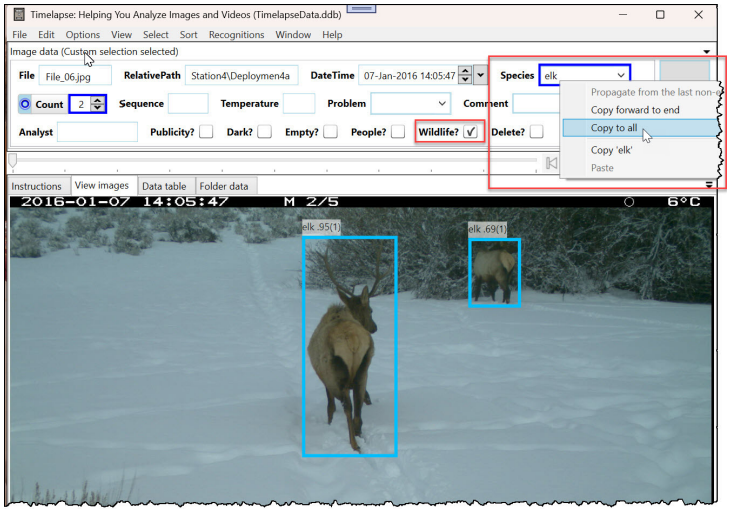
A workflow using species classifications

If the recognizer produces both detections and classifications (as with the image recognition practice set), further efficiencies can be achieved. The process is very similar to what was described in the previous section.

1. You would normally generate a *.json* detection file in the desired folder using ***AddaxAI*** , where you would select a particular classification model (e.g., SpeciesNet). One has already including in the Station4 folder of the Practice Image Set, called *timelapse_recognition_file_detections_and_classfications.json*. Import that file.
2. Follow the same steps above to broadly classify images as Empty, People, and Animals (we will later show some workflow variations that can shorten this step).
3. You will now tag the most frequently occurring species in your images, where you will use the classifier to select, bulk-tag and review particular species.
   a. In the ***Custom select,*** choose ***Animals***, and click ***Update counts***. The classification list will display each species ordered by a count of how many images it believes has that species. For example, the most commonly detected species is Elk.
   b. Select Elk, where the Elk images will now be displayed.
   c. Similar to the previous workflow, fill in the Species field to Elk, and then tentatively tag all selected images a Elk by via the ***Copy to all*** in the ***Species*** field's context menu.
   d. Verify that these images are all elk. For those that aren't correct them by filling in the correct species.
   e. At either the same time or after verification is completed, fill in each image's count box (e.g., via the ***Count*** field's increment buttons or marking option, or by creating ***Quickstart*** entries to do this in a single click).
   f. Repeat the above process for other frequently seen species.

Workflow tip. The idea of using classifications is to bulk-tag as many images as possible. Using high-probability classifications means that reviewing and correcting errors should also be fairly fast. Images that remains should be a much smaller set of images containing rarer or harder to classify species.





4. At some point, you will have already labeled many images with your most frequently seen species. The remaining images should be a much smaller set of infrequently seen species, or which were under the confidence limit you specified. At this point, you may want to revert to manual tagging.

5. Turn off ***Use recognition***, and then select all remaining untagged species (i.e., ***Animals***, but with ***Species*** not set). Tag the ***Species*** and ***Count*** field manually.

Workflow variations

The above workflow is the most thorough, where time is spent reviewing images for occasional recognition errors. However, depending on your needs, you may choose to follow one of the simpler workflows below. These are just examples, as many variations are possible.

A rapid, less accurate method for managing empties

This approach assumes you can tolerate some errors, where occasional images are mis-tagged as empties.

1. Select Empty detections as in the first workflow.
2. Instead of checking all returned images, choose ***Select | Create a random sample from the current selection***, which will randomly sample from those returned images. If you choose (say) 100 as your random sample, you can quickly review and count the number of errors in that sample and infer that the entire set has that error rate.
3. If that error rate is acceptable, bulk tag all returned images as empty. If not, you may want to try again at a more restrictive confidence value.

> ***Note.*** This and similar workflow shortcuts trade off accuracy with tagging speed. Occasional mis-classifications will occur. Some of the images detected as ***Empty*** for example, may have animals in it.

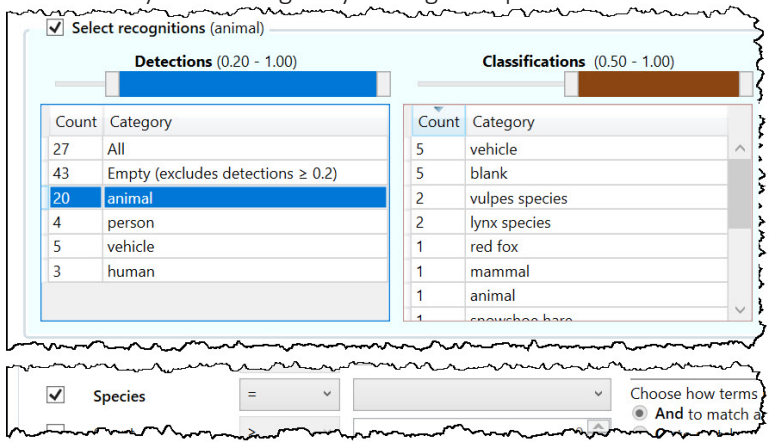
Variation. If you want to ignore images with people in it vs. tagging them, don't bother creating a 'person' flag in your template. Just use the empty field instead.

A rapid workflow focused on Species classification

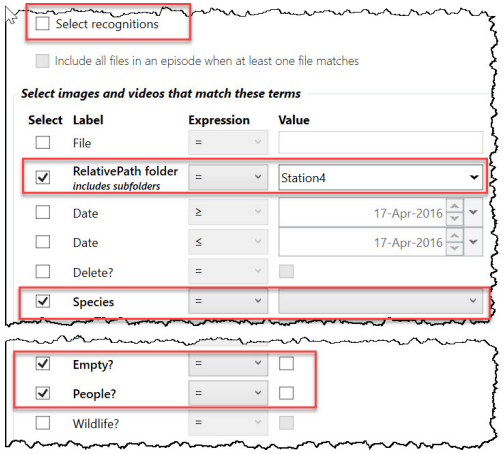
Instead of broadly classifying animals, you can immediately start classifying species using the ***Species*** field. The Animal? flag is not used.

The main idea is that you classify the most frequent species using image recognitions, and those that remain manually.

1. Start by using Custom Select to select the most frequently seen species in the Classifications list (e.g., Elk), seen by using the ***Update counts*** button.
2. Review and correct that classification, e.g., if an image does not contain an Elk, enter the correct species. If it contains a Person or is Empty, select the appropriate flags.
3. Repeat this for the next frequent classification.
4. When counts for the remaining classifications are small or 0, you may decide to revert to manual tagging rather than doing each species one by one. First, use the Detector to select Animals, and the Species as Blank (selecting animal should unselect a Classification item, if any). This will select all files classified as Animal that have not yet been tagged (i.e., those shown in the Count list if the Update counts button was selected). Then classify those images by setting the Species field.



5. Now check all remaining images by choosing these settings, which will select all untagged images in the Station4 folder, i.e., the *Species*, *Empty?* and *People?* fields are not yet set. Most will be either empty or have people in them. If you see the occasional wildlife image, tag its species field.



If you are only interested in species classification, there is no need to further classify the images that are empty or that have people in them. Essentially, you will be left with images of interest that have a non-empty species field filled in, and all other images.

Incorporating recognitions over time

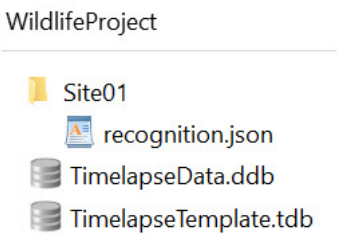
In practice, most agencies add images incrementally over time, usually because new cameras have been added and/or images from SD cards have been retrieved. This section offers a workflow that efficiently incorporates recognitions of these incrementally added images.

Initial recognitions

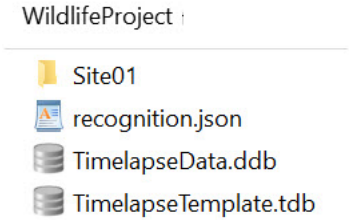
When you first collect images and load them into Timelapse, you can run *AddaxAI* on your image set's root folder or on each individual folder. This will generate recognition file(s) for all your initial images, which you can load into Timelapse using the *Recognitions | Import recognitions* option.

For example, the Wildlife project below initially contained images from a single camera, stored in the folder Site01. *AddaxAI* was run on the Station1 folder, where it generated a recognition file in Station1. Alternately, if there was more than one folder, it could have been run in the root folder, as shown on the right).

File in the Site1 folder



File in root folder



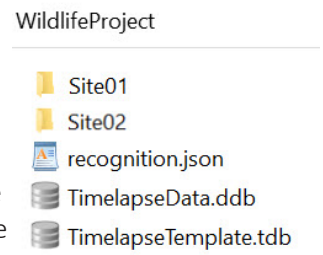
Subsequent recognitions: run AddaxAI on the sub-folder

As you add folders containing new images over time , you can then run *AddaxAI* on those new folders and import the new recognitions.

For example, consider this update to the WildlifeProject folder. The analyst has recently installed a new camera, and retrieved its images from its SD card. She created a new folder (Site02) to hold those images, where she added them to Timelapse using Timelapse's ***File| Add image and video files to this image set***.

The question is: how can we best use *AddaxAI* to import Site02's recognitions into Timelapse?

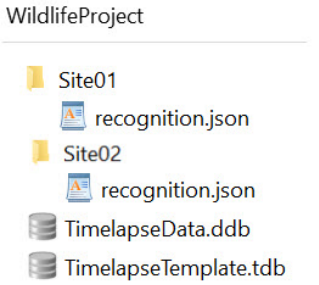
Approach 1 (inefficient). Rerun *AddaxAI* on all files. The simplest but highly inefficient way is to rerun *AddaxAI* in the root folder on all files. That recognition file would then include recognitions for all old and new files. When Timelapse reads in the recognition file, it will add recognitions for the images in the new Site2 folder, and will update the recognitions already recorded for images in the Site1 folder.



However, this approach is very inefficient, as it means you are re-running the recognizer for images that already have known recognition data. This can add considerable time to the recognition task.

Approach 2. Run *AddaxAI* in the just-added sub-folder. The more efficient way is to run *AddaxAI* in the new Site2 sub-folder. *AddaxAI* will analyze only the Site2 images, and will generate a new recognition file in the Site2 sub-folder. You can then import Site2's recognition file, where Timelapse will merge the new recognitions into its existing set of images.

The image illustrates how a new recognition file was created in the Site02 folder and then imported. Organizationally, the best way to do this is to have a recognition file in each corresponding folder.



In summary:

- *If you plan to add images to your image set over time*, try to organize each incremental addition as its own sub-folder. Run *AddaxAI* in that sub-folder, where you can then incrementally add those images import that json file into your image set.
- *Don't move the json file.* Timelapse requires that the json file is located in the folder where recognitions were done, as otherwise it won't be able to locate the images associated with its recognitions.

Other image recognition features

Custom select dialog extras

Include all files in an episode when at least one file matches checkbox. When checked, all files in an episode will be selecting even if only one (or more) files in that episode match the recognition criteria. This is feature is useful for cases when the analyst wishes to review all files in an episode in order to best identify and validate the recognition prediction (e.g., some images may only show only a small, difficult to identify part of the animal as it enters the scene), or to uniformly tag the images in that episode.

For this to work properly:

- a note field must have been previously populated with episode data (using *Edit | Populate a field with Episode data...*);
- when populating, you set the episode data format to include the episode number (e.g., 24:1|7) - this is the default setting;
- populating was applied to all your images, as the episode number needs to be unique.

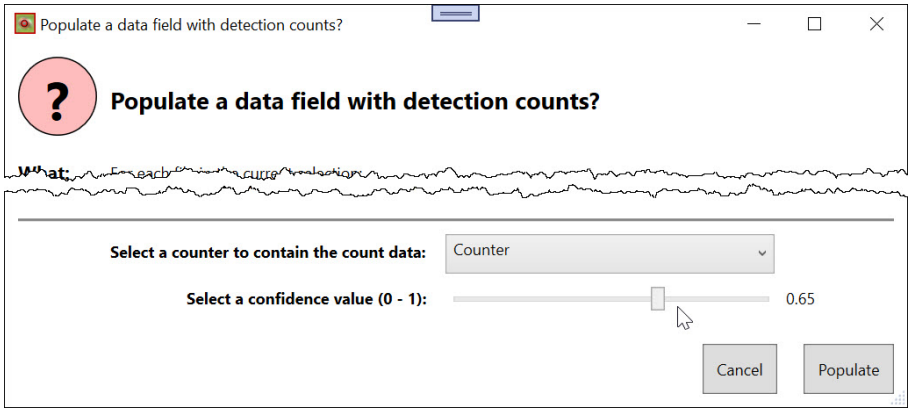
Show only those files the recognizer did not process checkbox. When checked, the custom select will return all files with absolutely no detection data, i.e., files that were not included in the *.json* file. If any exist, they will have to be tagged manually.

Recognitions menu extras

Recognitions | Populate a data field with detection counts... This facility, shown below, applies to all your currently selected files. It counts all detections in an image at or above a particular confidence value, and places that count in a counter data field of your choosing.

The number of detections provides a rough count of how many entities are in each image. It is an approximation, and is very much affected by the chosen confidence value (low values likely over-estimate the count, while high values likely under-estimate it).

- An approximate count for certain species may suffice. To avoid manual counting of (say) unimportant cow images, custom select on the 'Cows' classification. Then use the count facility on the returned cow images.
- You can always inspect your images and alter that count if needed.



Appendix 1: Recognition file format

If you use ***AddaxAI***, you don't really need to know anything about the Json recognition files, as that file should be in the correct image folder location. However, if you use ***MegaDetector*** directly (see Appendix 3), the information below may help you fix problems that can occur due to your recognition file not being able to find your images.

Timelapse and recognition files are similar in that both fundamentally associate data with images found in folders and sub-folders. For the two to work together, the file paths describing an image location must match.

How Timelapse associates recognition data with an image

As Timelapse reads in the json file, it compares the file path location of the image (recorded in each json image 'file' field) with the file path location it knows about (see technical note). If there is a match, Timelapse adds that file's recognition data to the database, and associates it with the corresponding Timelapse file image. If there is no match, it skips that file.

Recall that Timelapse records its file location relative to the root folder (i.e., the folder that contains the template file). For example, consider the WildlifeProject folder on the right, which is a root folder. If the Site01 folder contained the IMG001.jpg file, Timelapse would have recorded its location as Site01/IMG001.jpg. The path indicates the location of that image relative to (i.e., within) the folder containing the TimelapseTemplate.tdb file. It does not record the 'WildlifeProject' name, as it doesn't need that information. This allows Timelapse to find the files contained within the WildlifeProject folder even if that file is moved around.

Technical Note. The Json file structure (or see the formal specification)
It is helpful (although not necessary) to know what a json file normally contains. The image below portrays a stylized example. The file:

- contains some internal information about the recognizer,
- defines the detection and classification categories associated with a recognition, and optionally a description for each classification category
- provides a list of image files, where
- each image file includes its location as a ***relative path*** name, and its associated detection and classification information, if any.

```
info:                              various information fields provided by the recognizer
detection_categories:              defines several high level categories of what
    1: animal,                     the detected item could be
    2: person,
    3: vehicle …
classification_categories:         defines more precise categories associated with
    0: bear,                       a particular classification
    1: bird,
    2: canid,
    3: …
classification_descripton:         defines the taxonomy of each classification; currently
    0: <taxonomic id>,             a particular classification
    1: <taxonomic id>,
    2: <taxonomic id>,
    3: …                           only generated by Google's SpeciesNet classifier.
images:
   file: Site01/IMG001.jpg         the file's location
   frame_rate: 30.0                the frame rate in fps if the file is a video file, otherwise omitted
   max_detection_conf: 0.9         maximum confidence across all detections
   detections:                     detections (if any) associated with this file
      category: 1                  this detection's category(animal)
      conf: 0.9                    this detection's confidence
      frame_number: 30,            the frame number if the file is a video file, otherwise omitted
      bbox: 0.3,0.5,0.1,0.4        bounding box coordinates for the detection
      classifications:             this detection's classifications (if any)
         1,0.9                     this classification's category (bird) & confidence value
         more classifications for this detection, if any
      more detections for this image, if any
   more images...
```

Appendix 2: Trouble-shooting file importing

Import problems are unlikely if you use ***AddaxAI***, as it creates recognition file in the correct folder. However, if you try to import a .json recognition file, and get a warning that no recognition data was found, the problem is almost certainly due to a mismatch between how files and paths are named in the *.json* file vs. how they are named in the Timelapse database. This is likely due to your recognition file being moved, or your folders being renamed before the recognition file is loaded into Timelapse

1. As explained earlier in this document, Timelapse locates files by its relative path to the folder holding the template. For example, if the template is in the folder *MyImages*, and *IMG_01.JPG* is located in a subfolder named *Station1*, the image location is recorded relative to the *MyImages* folder:
   - » its ***RelativePath*** is *Station1*
   - » its ***File*** name is *IMG_01.JPG*
   - » so its location within Timelapse becomes Station1/IMG_01.JPG.
2. In the *json* file, each file has an entry that describe that file's location, which must exactly match the above location. The only exception is a json located in a sub-folder whose paths are relative to that sub-folder, which Timelapse recognizes as a special case.
3. To check and possibly repair path errors, make a copy of the recognition. *json* file. Open that copy in a text editor.
   - » the free EditPad Lite editor works really well for this, as the usual Windows-based editors are poor at handing really large files.
4. Check if the paths are what would be expected. For each file, the path is recorded in the "***file***" field. For example, a correct path to the above file would be recorded in the line:
   "file": "Station1/IMG_01.JPG".

   If you are using MegaDetector by itself, the prefix to the path is likely either added or missing due to how files were submitted to it. For example, an incorrect path to the above file (which in this case includes the added root folder name), would be
   "file": "MyImages/Station1/IMG_01.JPG"

   If a prefix is missing from all paths because MegaDetector was run in a sub-folder (e.g., within Station1), moving that json into the sub-folder and then importing it would work, as Timelapse recognizes this as a special case.
   "file": "IMG_01.JPG"
5. Do a global replace to add or remove the appropriate prefix, for example, to remove ***MyImages/*** from all files paths. Then try to import the .json file back into Timelapse.
6. Alternately, Dan Morris provides an application for adjusting paths, as he

recognizes this as a common problem (ask him for it). Its a bit complex to use, but it does handle complex situations, such as when you split your folders up to contain multiple Timelapse database.

Appendix 3. MegaDetector without AddaxAI

If for some reason you don't want to use ***AddaxAI,*** you can install the ***MegaDetector*** recognition software directly on your machine and generate the recognition file yourself (a bit more complex). See the Timelapse website: Downloading the MegaDetector Image Recognition System.

Alternately, you can submit your images to Dan Morris (contact him at cameratraps@lila.science), who will run ***MegaDetector*** for you. Dan is incredibly helpful. He will:

- check that ***MegaDetector*** is appropriate for what you are trying to do;
- tell you how to submit your camera trap images, either through the cloud (preferred) or by sending a hard drive;
- once submitted, will have ***MegaDetector*** analyze your images and will return a data file (suffixed with *.json*) containing the image recognition data, which you can then load into Timelapse.

Notes
1. The only reason for not using ***AddaxAI*** to do the heavy lifting is if you are knowledgeable about MegaDetector and want to tweak its recognition parameters, or use it to develop your own model.

2. Several versions of MegaDetector are currently available. One is maintained by Dan Morris email:cameratraps@lila.science. Dan has actively supported MegaDetector and its community for a long time. Another version is maintained by Microsoft AI for Good research group. While it should be compatible with Timelapse (and I am working with them on that), it has not yet been thoroughly tested for all recognition features.

3. We do expect other companies and agencies to develop their own image recognition systems, where they will produce Timelapse-compatible recognition files.