

Timelapse Image Recognition Guide

A guide to importing and using image recognition data

The screenshot displays the Timelapse software interface, which is divided into two main panels. The left panel, titled "Timelapse: Helping You Analyze Images and Videos Captured from Field Cameras (TimelapseData.ddb)", shows a "Data entry for All files" section with fields for File (IMG_129.JPG), RelativePath (Station1), DateTime (21-Aug-2013 17:14:46), and checkboxes for Delete?, Copy previous values, Species, Episode, Person?, Animal? (checked), and Empty?. Below this is a "Data Table" tab and a large image viewer showing a savanna scene with two giraffes. One giraffe is highlighted with a blue bounding box and labeled "giraffe 1(1)", while the other is labeled "giraffe 99(99)". The bottom of the left panel shows "ScoutGuard" and a timestamp "08.21.2013 17:14:36". The right panel, titled "Select and view a subset of your files", provides instructions on how to filter files based on search criteria. It includes a "What:" section explaining the purpose, a "Solution:" section with steps, a "Result:" section, and a "Hint:" section. The "Image Recognition" section is active, showing "Use recognition" checked, "Recognized entity" set to "giraffe", and "Confidence" set from 0.80 to 1.00. A checkbox "Rank by confidence" is also present. Below this, a table lists search criteria: File, RelativePath folder (includes subfolders), DateTime, ImageQuality, Delete?, and Species. The "How multiple terms are combined" dropdown is set to "AND". The bottom of the right panel shows "22 files match your query" and buttons for "Reset to All Images", "Cancel", and "Okay".

Timelapse: Helping You Analyze Images and Videos Captured from Field Cameras (TimelapseData.ddb)

File Edit Options View Select Sort Recognitions Window Help

Data entry for All files

File: IMG_129.JPG RelativePath: Station1 DateTime: 21-Aug-2013 17:14:46 Delete? ☐ Copy previous values ☐

Species: Episode: Person? ☐ Animal? ☒ Empty? ☐

Instructions Image set Data Table

giraffe 1(1)

giraffe 99(99)

ScoutGuard 08.21.2013 17:14:36

File: 121 of 272 Select: All files Sorted by: RelativePath then by Date/Time Image set is now loaded.

Select and view a subset of your files

? Select and View a Subset of your Files ☐ Hide explanation

What: You may want to view only a subset of your images and videos that fit some criteria of interest to you.

Solution: Specify the search terms that describe your criteria.

1. Each row below reflects your data fields or (if enabled) specific recognition data.
2. Select one or more rows and adjust its values to reflect your search criteria.
3. If you have selected multiple terms in the lower area, select how those terms should be combined.

Result: Only those images and videos matching your search criteria will be displayed.

Hint: Glob expressions are case sensitive and allow wildcards as follows:

- * matches any number of characters and ? matches any single character
- [abc] matches one of the indicated characters; [a-z] matches one character in the range of indicated characters.

Image Recognition

☒ Use recognition ☐ Rank by confidence ☐ Show only those files the recognizer did not

Recognized entity: giraffe

Confidence: from 0.80 to 1.00

☐ Include all files in an episode when at least one file matches 22 files match your query

Select images and videos that match these terms

Select	Label	Expression	Value	How multiple terms are combined
<input type="checkbox"/>	File	=		These terms are combined using AND returned files match all selected con
<input type="checkbox"/>	RelativePath folder includes subfolders	=	Station1	
<input type="checkbox"/>	DateTime	≥	21-Aug-2013 17:14:46	
<input type="checkbox"/>	DateTime	≤	21-Aug-2013 17:14:46	
<input type="checkbox"/>	ImageQuality	=	Ok	These terms are combined using eith
<input type="checkbox"/>	Delete?	=		
<input type="checkbox"/>	Species	=		

Reset to All Images 22 files match your query Cancel Okay

Saul Greenberg
Greenberg Consulting Inc. / University of Calgary
saul@ucalgary.ca

Timelapse Image Recognition Guide

A guide to importing and using image recognition data¹

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, read the Timelapse Quickstart Guide (available on the [Timelapse web site](#).)
- **Other 3rd party software does the image recognition.** Timelapse does not itself do image recognition. Rather, Timelapse imports an image recognition file created by other software. Most Timelapse users currently use Microsoft's *MegaDetector*, but other recognizers are also compatible with Timelapse (e.g., *Zendo from DeepAI*). See the discussion later in this section, as well as the up-to-date details provided on the [Timelapse and Image Recognition](#) page on the Timelapse website.
- **Image recognition isn't perfect.** It's important 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](#). Presented by Saul Greenberg at *Conference: Scaling Up Camera Trap Surveys to Inform Regional Wildlife Conservation*, Columbia Mountains Institute of Applied Ecology, May 18-20, 2020. Duration: 20 minutes.

Credits. Camera trap images used in this and other guides are used with permission or obtained from public repositories. *Sources:* Parks Canada; Lila Science - Idaho Fish and Game data set; Lana M. Ciarniello, Aklak Wildlife Consulting, Snapshot Serengeti. Everything else ©Saul Greenberg, 2022.

Table of Contents

Timelapse Image Recognition Guide	2
What is image recognition?	3
Two types of image recognition: Detection vs. Classification	3
Image recognition is not full proof	3
Why use automatic image recognition?	3
Image recognizers you can use	4
Microsoft's MegaDetector	4
How to use MegaDetector	4
MegaDetector and your folder structure	4
The Image Recognition Practice Set	5
How Timelapse interprets recognition data	5
Detections are displayed as labeled bounding boxes	5
Displaying classifications	6
The meaning of confidence levels	7
Setting bounding box preferences	8
Selecting images fitting a recognition criteria	8
A recognition workflow using detections	10
Prepare the template and load everything	10
Select and filter out empty images	10
Mark images with people in them	11
Mark images with animals in them	12
Mark any remaining images	13
Detailed classification	13
A recognition workflow using classifications	13
Workflow variations	14
Skip animal verification	14
A rapid, less accurate method for tagging animals	15
Other image recognition features	15
Custom select dialog extras	15
Recognitions menu extras	15
Working with JSON recognition files	16
How Timelapse associates recognition data with an image	16
Running Megadetector in the root folder	16
Running Megadetector in sub-folders	17
Trouble shooting JSON file importing	17

What is image recognition?

A holy grail of camera trap analysis is automated image recognition, where a computer automatically inspects and tags your images. While we are not quite there yet, you can use it to ease your workflow as long as you understand its many subtleties.

You can experiment with image recognition in Timelapse (and we do encourage that). Still, you will get better results if you have: some knowledge of how image recognition works; its benefits and weaknesses; how Timelapse shows you image recognition results; and the various ways you can incorporate recognition data into an effective workflow.

Two types of image recognition: Detection vs. Classification

There are two types of image recognition.

- A **detector** detects whether something is in an image or not. For each suspected **detection**, a detector:
 - » assigns a coarse identifying label to it (e.g., **empty**, **animal**, **person**, **vehicle**)
 - » locates it via **bounding box** coordinates, which can be used to draw a rectangle around the suspected entity
 - » assigns a **confidence value** indicating the likelihood that it is correct.
- A **classifier** analyzes the sub-image contained by the detector's bounding box, where it performs a fine-grained classification chosen from a set of known categories. A classifier will typically produce a list of possible **classifications** for each detection. For example, wildlife classification, categories will be wildlife species (e.g., elk, wolf, bear, dog). The recognizer includes a **probability value** that very 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.

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 identifications:** it 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.

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 images that are likely empty from any further analysis,** such as images with no detected people or animals in them or images with only very low confidence. This is especially valuable in cases where your cameras generate a large number of empty images, e.g., because you have your camera on a timelapse mode, or because wind effects frequently trigger your camera in motion-detection mode.
- **Identify images containing people.** This is especially important if your agency has a privacy policy that requires those images to be 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 needed, 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, use the classification categories to eliminate species of little interest, or to quickly focus on those images containing a particular classified species.

Image recognizers you can use

Timelapse does not do image recognition by itself. Rather, a third party image recognizer does the actual image processing.

Microsoft's MegaDetector

The most popular image recognizer that works with Timelapse is Microsoft's **MegaDetector**, a free open source image recognition system. **MegaDetector** analyzes your images and produces a recognition file containing detections for each submitted image (see [MegaDetector web site](#) for details). Timelapse can import and use that file, as described shortly. **MegaDetector**:

- is specifically designed for wildlife-oriented camera trap images, and has been used by many ecologists around the world;
- is currently best at handling detections;
- has a person in charge that can help you as needed (Dan Morris at cameratraps@lila.science)
- does have limited classification capabilities for certain species (on special request only to Dan Morris), as the classifier is still being fine-tuned;
- is free!

MegaDetector works very well with Timelapse, where its inter-operation is a result of extensive collaborations between their developers. Indeed, Timelapse is recommended as the 'front-end' for visualizing and using **MegaDetector** recognition data. Many organizations have used both successfully to efficiently tag massive numbers of images.

Note. The [Timelapse and Image Recognition](#) page on the Timelapse website provides details on how to download and use **MegaDetector**, and how to contact Dan Morris if needed.

Note. We do expect other companies and agencies to develop their own image recognition systems, where they will produce Timelapse-compatible recognition files. One such agency is **Zendo from DeepAI**, whose software allows you to train your own recognizer and run it on your own images. The [Timelapse and Image Recognition](#) web page provides details.

How to use MegaDetector

Because we are most familiar with Microsoft's **MegaDetector**, only that image recognition system will be discussed here.

There are two options for using **MegaDetector**.

1. You can install the Megadetector recognition software on your machine and generate the recognition file yourself (fairly straight-forward). See the [Downloading the MegaDetector Image Recognition System](#) on the Timelapse website for details.
2. Alternately, you can submit your images to Dan Morris (contact him at cameratraps@lila.science), who will do it 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 the **MegaDetector** analyze your images and will return a data file (suffixed with `.json`) containing the image recognition data.

MegaDetector and your folder structure

When your images have been processed by **MegaDetector**, you will end up with one or more `.json` files containing recognition data. These files contain recognition data describing every file. Importantly, that data also includes the file's relative path and file name, which Timelapse uses to locate the file.

To import recognition data (try this with the image recognition practice set):

- Place the `.json` file into the appropriate folder so you can find it. (see Important Note below);
- start Timelapse and load your image set as normal;
- import the `.json` file by selecting **Recognitions/Import image recognition data for this image set**.

Important Note. Read the *Working with JSON recognition files* at the end of this chapter. It describes critical points about how to make sure Timelapse can find the files specified in the recognition file, and how to include those recognition files in your workflow. It also describes how you can create multiple json files, each representing a sub-folder within your root folder that you can add incrementally as new images become available over time.

The Image Recognition Practice Set

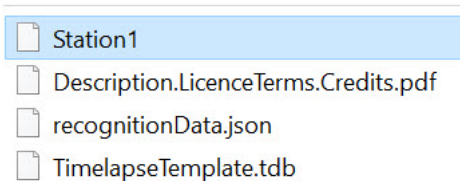
Follow along with the upcoming explanations using the *ImageRecognitionPracticeSet*, downloadable from the [Timelapse web site](#). It contains a template and a single *Station1* folder containing image files from the [Snapshot Serengeti](#) project. It also contains a *.json* recognition file, which was generated by running the images through MegaDetector. This particular file contains both detections and classifications data.

To use the Image recognition practice set

- download and unzip the folder to a convenient place
- start Timelapse and load the template in the practice image set as normal;
- import the *.json* file by selecting *Recognitions/Import image recognition data for this image set*.

Of course, if you have submitted your own files to the *MegaDetector* to produce your own *.json* recognition file, you can use that as well.

ImageRecognitionPracticeSet



How Timelapse interprets recognition data

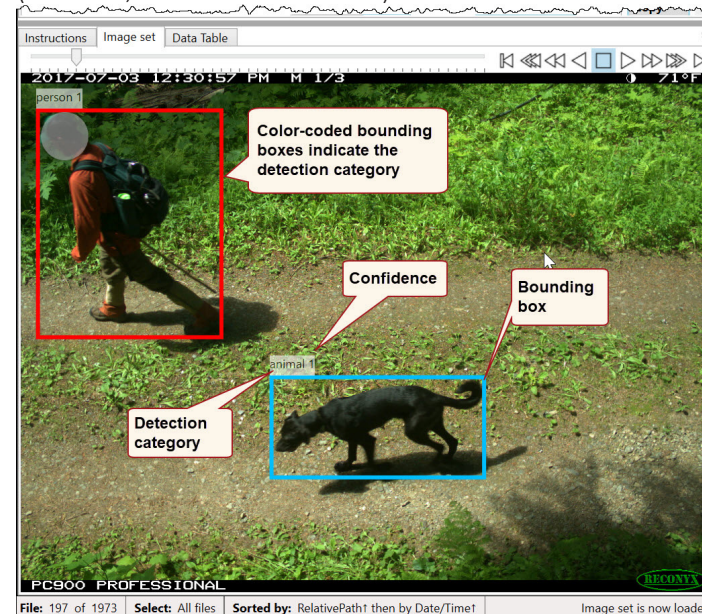
After you generate the *MegaDetector* recognition file, you can easily import it into Timelapse. You can then view and operate on your images' recognition information to streamline your workflow, as discussed shortly.

Detections 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:

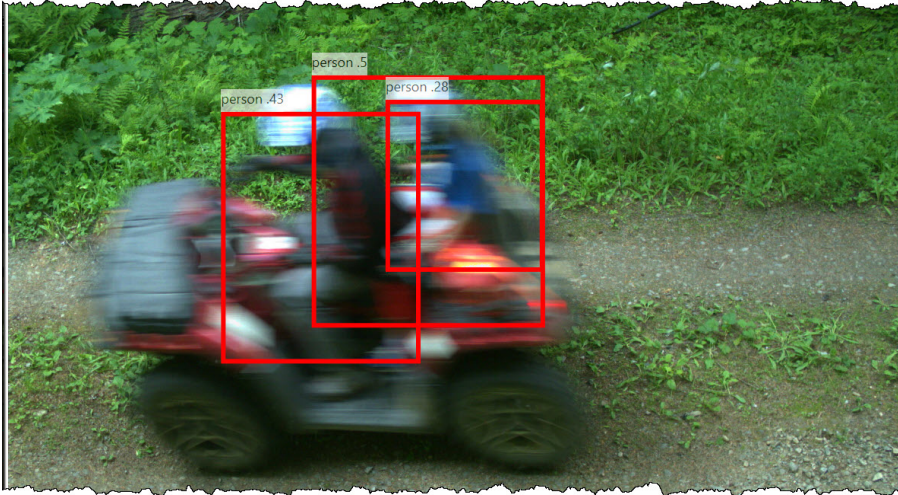
- is labeled with what it thinks the entity is (e.g., person, animal);
- is colored to distinguish the detection category (e.g., red for person, blue for animal);
- includes a confidence value of correctness ranging from 1 (high confidence) down to 0 (low confidence).

This image, for example, shows two correct high confidence detections within its labelled 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: 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.

This 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.



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

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



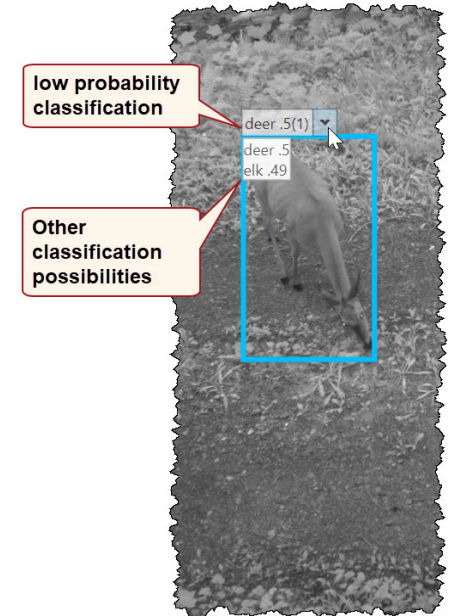
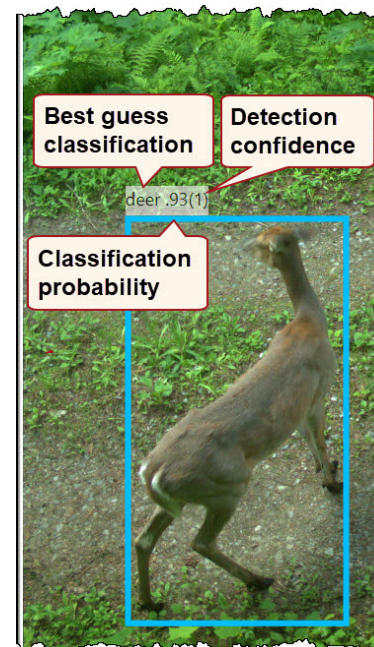
False negative

Displaying classifications

Classifications, if available, work with detections. Their visualizations are similar, except that extra information is added.

The left image correctly classifies the detected animal as a deer. The numbers that follow provide the individual confidence values as probabilities: the classification probability is .93, while the detection confidence is (1). While the 'animal' label is not included, it is implicit by its classification (a deer is an animal). The bounding box also uses the animal color.

The right image also correctly classifies the detected animal as a deer, except this time with a much lower probability value (.5). Because of its low probability, a drop down menu is included that lists alternate possibilities of lesser probabilities, in this case an elk at .49. When multiple classifications appear, the bounding box label displays the highest probability prediction. If no classifications are available, then the label only shows the recognized detection category (e.g., Person, Animal, Empty).



The meaning of confidence levels

Confidence is best seen as a very rough indicator of how likely it is that a detection or classification is correct. The catch is that the reliability of a particular confidence value can vary greatly from dataset to dataset, as explained in the technical note on the next page.

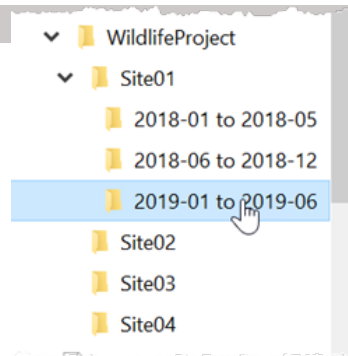
To mitigate this variability, the *json* recognition file specifies a **typical detection threshold**, and a **typical classification threshold**, where a confidence and probability value is normally interpreted relative to those thresholds.

- **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 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 confidence values you want to use to ones that better fit your images, as discussed next.



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, all 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 versions of MegaDetector use different strategies for generating confidence values. As such, the confidence threshold above which detections and classifications are 'mostly correct' can differ greatly, as they depend upon the MegaDetector version used. For example,

	MegaDetector version 4	MegaDetector version 5
Typical detection threshold	0.8	0.2
Typical classification threshold	0.05	0.75

As these values can change across future versions of MegaDetector, version-specific values are included in the *.json* file. Timelapse then uses those values to determine the default confidence levels in the *Select* dialog.

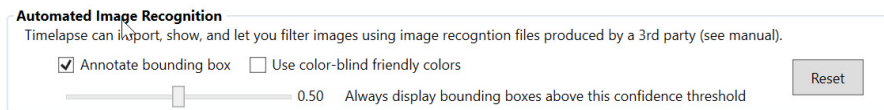
As these default values are (at best) an approximation, the best way to judge a reasonable detection or classification confidence level is by reviewing your images to get a sense of how well recognitions perform at different levels.

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.

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.

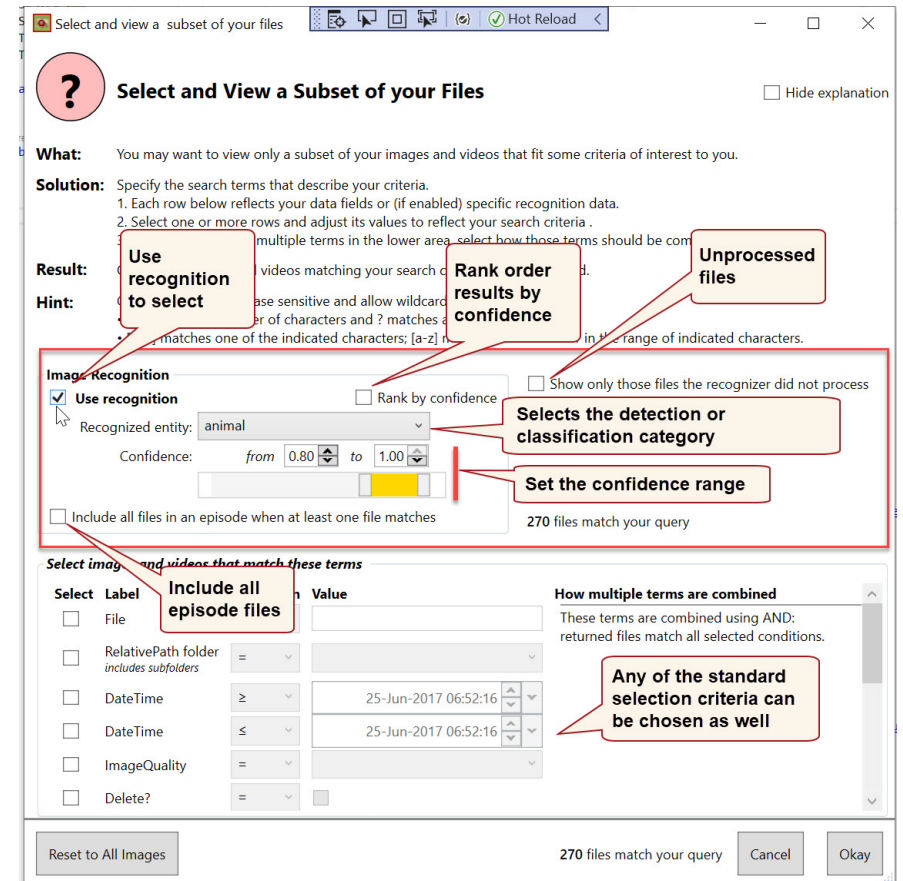


Experiment with the **Always display bounding box** setting by reviewing your images. While it is tempting to set the threshold to a low value, it could add considerable visual clutter by showing bounding boxes around incorrect recognitions. Alternately, setting it too high will mean that some correctly 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.

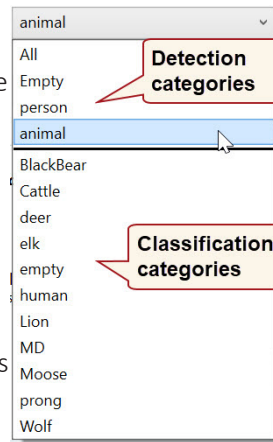
Selecting images fitting a recognition criteria

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 will now include an **Image Recognition** area, illustrated and annotated below. (Confidence defaults are from MegaDetector version 4).



The various recognition features of this dialog are detailed below.

- **Use Recognition** checkbox, when checked, will enable the recognition controls, which will let you set recognition criteria for selecting files
- **Recognized entity** contains a drop-down menu that lets you select files by detection or classification category. Detections are always listed at the top.
 - » *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.
 - » *Person*: the image includes a detection of one or more people within the given confidence range
 - » *Animal*: the image includes a detection of one or more animals within the given confidence range



Note. The detection and classification categories are specified in the recognition file produced by the *MegaDetector*. The ones used above are an example. Your own categories may differ from these.

- **Confidence Range** is adjusted by either a range slider or the up-down boxes. It allows you to specify a 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, .
- **Rank by confidence.** When checked, all images of the selected entity type will still be returned, 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. If the selected entity is a detection, the detection confidence is used. If the selected entity is a classification, the classification probability is used.

The examples below illustrate working with confidence. Understanding the subtleties – especially for *Empty* – will be helpful to using recognition data effectively. This example illustrates confidence defaults based on MegaDetector version 4, which uses a typical detection threshold of 0.8 and a typical classification threshold of .75. You will need to calibrate these ranges to the default that actually appears when you use TImelapse.

- **Entity = Empty, confidence range is 1 - 1** selects all images where the image recognizer has not identified any detections. You may use this to rapidly inspect (and occasionally correct) images with nothing in them.
- **Entity = Empty, confidence range is 0.8 - 1** selects all images where the image recognizer has not identified any detections, and all images with 0.2 or less confidence (i.e., $1.0 - 0.8 = 0.2$). While similar to the above, it should include more images that likely don't have anything in them (i.e., erroneous detections). A few more correct detections may also appear.
- **Entity = Empty, confidence range is 0.8 - 0.99** selects all images with 0.2 or less confidence (i.e., $1.0 - 0.2 = 0.8$) but excludes images where the image recognizer has not identified any detections. You would use this a part of a workflow where you first wanted to go through images with no detections (1 - 1), and then separately go through images with poor detections.
- **Entity = animal, confidence range is 0.8 - 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 = all, confidence range is 0.5 - 0.8** displays images where at least one detected entity is between 0.5 - 0.8 confidence range. You may use this to rapidly inspect detections that are more error-prone.
- **Entity = all, confidence range is 0.1 - 0.5** displays images containing at least one detected entity between 0.1 - 0.5 confidence, (i.e., images containing detections where none of them exceed 0.5 confidence. You may use this to rapidly inspect and eliminate detections that are likely wrong.
- **Entity = Black bear, confidence range is 0.8 - 1** selects all high probability black bear classifications. You would use this a part of a workflow to perhaps check and quickly label those images as a bear.
- **Rank by confidence** is useful in your workflow for determining – somewhat qualitatively – the confidence range where that recognizer

seems to work well vs. poorly. Since files are now sorted by confidence, you can play the images from the beginning, and scan them to check if they have the correct entity. When errors increasingly appear, you can check the confidence value (displayed on the bounding box) to see 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 image recognition practice set which contains only 272 files. It also used confidence values based on MegaDetector version 4, so the confidence values you use in other MegaDetector versions may differ. In practice, the suggested workflow reaps substantial benefits when used over larger image sets.

If you have your own image set and .json file, you can mimic these steps.

Recognition workflow strategies depend upon you:

- selecting a subset of your images to display based on the detector's predictions (via *Select / Custom Selection*),
- bulk- tagging that sub-set in a single action
- rapidly reviewing those images for tag accuracy and for correcting 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.

The workflow described below is based on using detection information only. This particular workflow assumes that there is a very low tolerance for errors, and is perhaps the most thorough one you can do. It also uses the typical detection and recognition defaults, which can differ from your own images.

You will inspect and tag: all empty images, images with people in it, and images with animals in it. Afterwards, you can classify just the animal images.

Workflow Tip: This workflow is just one of several possible. The one you choose depends on the nuances of your work. A subsequent section will describe several other workflows that may better fit different needs. We fully expect you to create your own workflow protocol that fits your own needs – these descriptions are included just to give you a flavour of the possibilities. Still, go through this one as it described the details that should be considered in all workflows.

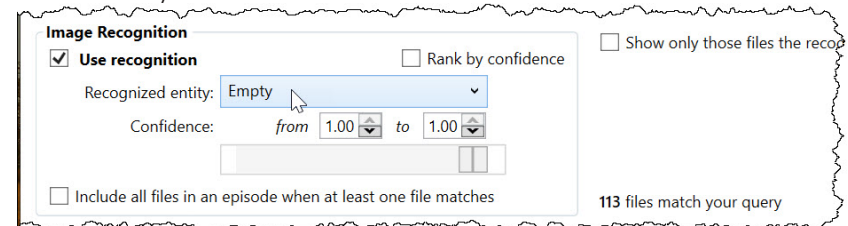
Prepare the template and load everything

1. Add the following fields to the Timelapse Template: three flags called *Person*, *Animal*, and *Empty*. These are already including in the image recognition practice set. Note that these labels have no special meaning to Timelapse. Their role in this example workflow is to track and bulk-tag what detections have been accepted and corrected.
2. Check that the appropriate .json detection file is in the same folder as your template.
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. After this step, you should see blue or red bounding boxes in images, each box indicating where the detector has spotted a person or animal.

Select and filter out empty images

Your goal in this step is to remove empty images (i.e., where the detector has not detected anything in it) from any further consideration, while correcting any detection errors.

5. Use *Custom Select* to select the images where the detector has not detected any entities.

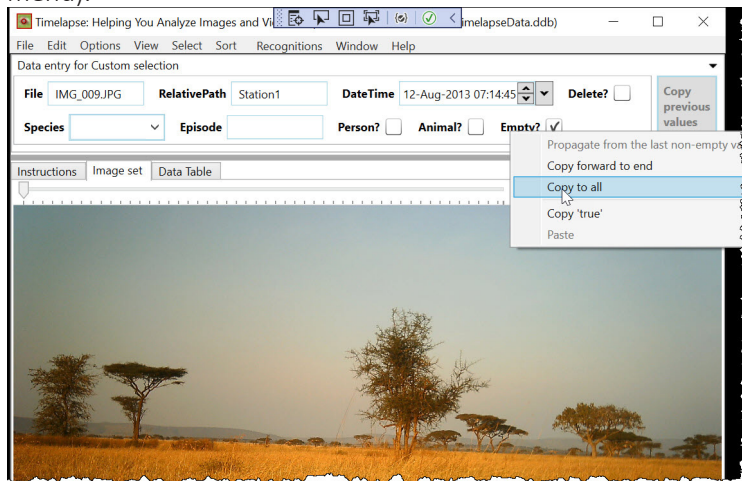


- a. Click *Use Recognition*
- b. Set *Recognized entity* to *Empty* and a confidence range from 1 to 1.
- c. In the image recognition practice set, this will match 113 out of the 272 total files. Click *Okay*.

Note. The number at the bottom right above indicates how many images match that selection. The same number is repeated at the bottom of the dialog box.

Timelapse is now displaying the 113 images that the detector has considered **Empty** at a high confidence. While we would expect most of these to be empty, they have to be checked.

6. Tentatively tag these 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).



7. 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. A custom selection will show the last image (or as close to it as it could get) that was previously selected. Remember to scroll back to the first image before reviewing them, as otherwise you will miss some of them.

8. 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 **Animal** field and (if you want) set its species. For example, IMG_148-149 has an unidentifiable animal in it (very small, on the middle far left), and IMG_261-263 has a zebra (you may have to use the **Gamma** setting on the **Options | Temporarily adjust image appearance** to see it in the dark).

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**.

9. 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 empty images at some point, if you feel the detector's accuracy is good enough for your needs.

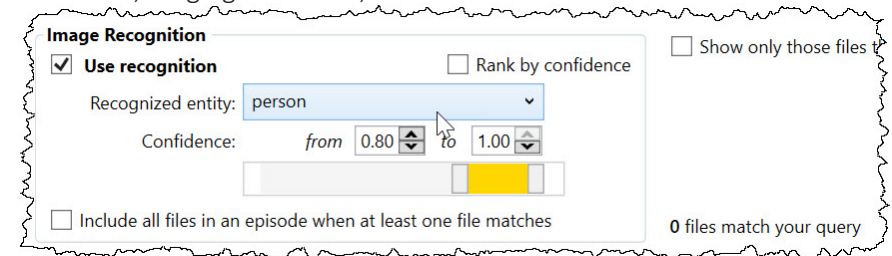
At this point, we have identified 109 empty images, and corrected 4 errors in just a few minutes. 40% of the total files have now been processed!

Mark images with people in them

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. In any of these scenarios, it's useful to rapidly identify the subset of images that contain people.

Your goal in this step is to identify as many images as possible with people in them. The process is somewhat similar to how we deal with empty images.

10. Select the images where the detector has a high confidence that a person is in them, ranging from .8- 1, as illustrated below.



Note. As mentioned, the image recognition practice set does not contain any images with people in it. The detector has likewise not detected any people (if reducing the confidence, there will still be 0 matches). However, to show how the workflow works, the explanation will continue as if there were matches.

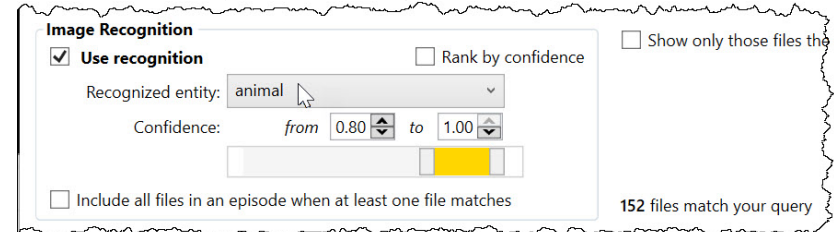
11. Similar to how you had previously tagged and verified empty images, tentatively indicate a person is present in all these images by setting the **Person** flag to true on one image, and then use **Copy to all** to set it for all images in the current selection.
12. Review this subset (remember to start at the first file). If you see an image that does not contain a person (false positive), clear its **Person** flag. If the image is empty or has an animal, set its **Empty** or **Animal** flag to indicate that. 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.
13. Repeat the above, but this time by setting the confidence from .8 to .9, then between .7 and .8, and so on down to 0.
 - » When you begin to see that the detector is getting things more wrong than right, don't set the **Person** flag via **Copy to All**. Rather, review the images and click only on the ones with a person in it.
 - » At some point, you can stop reviewing images below a certain confidence threshold if there are many errors.

Workflow tip. Move the **from** confidence of the range slider slowly, and monitor how many files match that confidence range. Use that information to decide how 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.

Mark images with animals 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.

14. Select the images where the detector has a high confidence that an animal is in them, ranging from .8 - 1, as illustrated below. 152 files (of the 272 total files) are in this range. Click Okay.



15. As before, tentatively set the **Animal?** flag for all those images. Review them and alter the flags as needed. Some things to note:
 - » None are empty, although a few have animals that are fairly obscure and hard to identify.
 - » Some of them show bounding boxes on animals that could have been easily missed otherwise. IMG_195, 214, for example, detected animals in dark areas, where the human eye would not see them unless we investigated it with the Gamma correction facility.
 - » Another example that would have been easily missed without the bounding boxes is IMG_277: a far-away animal obscured in the underbrush.
16. Repeat the process, but this time setting the upper confidence to .8, and then drag the lower confidence slider downwards. When it is at its lowest limit, we see only 5 matching files. Instead of using **Copy to all**, just go through them and tag whether they are **Empty**, **Animal**, or **Person** files.

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.

Mark any remaining images

17. 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*).



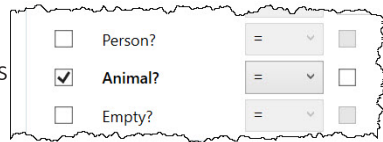
<input checked="" type="checkbox"/>	Person?	=	0
<input checked="" type="checkbox"/>	Animal?	=	0
<input checked="" type="checkbox"/>	Empty?	=	0

Note: You can double check your tagged images at any time by selecting the **Empty**, **Person** or **Animal** checkbox only.

Detailed classification

At this point, you have isolated the **Empty**, **Person** and **Animal** images.

18. To classify the **Animal** images, use **Custom select**, and check **Animals** = unchecked. All the tagged Animal images (which could be a considerable number) will now be filtered from view. You can now go through the standard manual tagging mechanism to set the species for the remaining untagged images.



<input type="checkbox"/>	Person?	=	0
<input type="checkbox"/>	Animal?	=	0
<input type="checkbox"/>	Empty?	=	0

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.

19. 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).

20. To discard People images (if desired), repeat step 20 expect with **Person** selected.

A recognition workflow using 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. Follow the same steps above for:
 - a. preparing the template and loading everything;
 - b. selecting and filtering out empty images;
 - c. marking images with people in them;
 - d. marking images with animals in them;
 - e. marking any remaining images.
2. Instead of manually tagging animal species (step 19 above), you can use the classifier to bulk-tag frequently occurring species.
 - a. In the **Custom select**, choose **Animals**, and click **Rank by confidence**. Scan the returned images quickly to get a sense of which species are most frequently captured. For example, you should see many images of Thomson Gazelles, then perhaps Giraffes, followed by a small number of other species.
 - b. Uncheck **Rank by confidence** and select **gazelleThomsons** with a high confidence range. Repeat the **Copy to all** process to tentatively tag the **Species** and **Animal** data fields. Verify / correct those as before (reminder: create Quickstart entries to do this in a single click).
 - c. Repeat the above at lower confidence ranges, but only while classification errors remain fairly low.
 - d. Repeat the above process for other frequently seen species (e.g., **Giraffe**).
 - e. At this 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.

When you start getting to rarer species, revert to manual tagging. Turn off **Use recognition**, and then set the following:

A screenshot of a classification interface. It features a list of checkboxes on the left: **Species** (checked), **Episode** (unchecked), **Person?** (unchecked), and **Animal?** (checked). To the right of each checkbox is a dropdown menu with an equals sign (=) and a downward arrow. Further right is a search bar. On the far right, there are radio buttons labeled **Any** (selected) and **Or** (unselected).

This will now select all remaining untagged species (i.e., *Animals*, but with *Species* not set). Tag the *Species* field manually.

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

Workflow variations

The above workflow is the most thorough. However, depending on your needs, you may choose to follow one of the simpler workflows below.

Skip animal verification

You can skip the step for tagging and verifying animal detections, where you instead start classifying species immediately in the *Species* field. The difference is that you will handle mis-detected animals as you do your detailed classification, rather than as a separate step. To do this,

1. When creating the template, don't bother including an *Animals* data field.
2. Follow the workflow above for verifying and correcting detected *Empty* and *People* images.
3. To classify wildlife, set the following parameters. This will return all files with detected animals, and exclude files you've already tagged as being either empty or having a person in it.

A screenshot of the 'Image Recognition' settings interface. It includes a checkbox for **Use recognition** (checked). To its right is a checkbox for **Rank by confidence** (unchecked). Further right is a checkbox for **Show only those files the** (unchecked). Below 'Use recognition' is a dropdown menu for 'Recognized entity:' set to 'animal'. Below that is a 'Confidence:' section with 'from' and 'to' sliders set to 0.10 and 1.00 respectively. A yellow bar represents the confidence range. At the bottom left is a checkbox for **Include all files in an episode when at least one file matches** (unchecked). At the bottom right, it says '157 files match your query'. Below this is another section with checkboxes for **Episode** (unchecked), **Person?** (checked), and **Empty?** (checked), each followed by a dropdown menu with an equals sign (=) and a downward arrow.

4. Start classifying these animal images by setting its *Species* field, where you correct the occasional image that are either Empty or Animal).
5. Alternately, you can rapidly inspect and eliminate many of the animal false positives by first selecting images where the detected entity is animal, and the confidence range is (say) between 0 – 0.5. Then continue with the previous step but with the confidence range between 0.5 – 1.0.

A rapid, less accurate method for tagging animals

This approach assumes you can tolerate some errors. Note that the animal/people/empty tags in the template are required only if you want to tag them.

1. Quickly check the detector's **Person** and **Empty** results. Determine if the error rate is tolerable.

Workflow tip. You can check if the error rate is acceptable by quickly selecting the Empty and Person images, and scanning them for accuracy. For a more rigorous approach, follow this with **Select | Create a random sample from the current selection**. If you choose (say) 100 as your random sample, you can count the number of errors in that sample and infer that the entire set has that error rate.

2. If the error rate is tolerable, focus only on files with detected animals.
 - a. **Custom select** the recognized entity as **Animal**.
 - b. Slide the confidence downwards until the number of matching images does not increase significantly.
 - c. Start classifying the species of the returned images immediately.
 - d. if the returned images include some without animals (false positives), which you can just skip over when you come across them.

The above method trades off accuracy with tagging speed. There will be missed tags in this method. Some of the images detected as **Empty** or **Person** may have animals in it, as are some of the low confidence **Animal** images you have ignored.

Variation. If you simply 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.

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, 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.

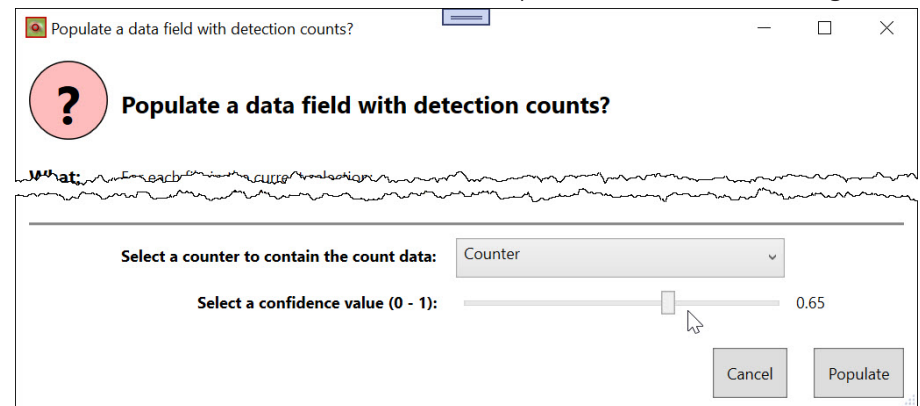
Show only those files the recognizer did not process checkbox. When checked, the custom select will return all files with absolutely no detection data, i.e., files that were not included in the *.json* file. If any exist, they will have to be tagged manually.

Recognitions menu extras

Reconitions | 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).

- You can always inspect your images and alter that count if needed.
- 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..



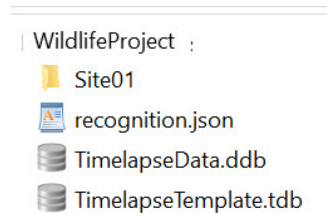
Working with JSON recognition files

Timelapse and Megadetector 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.



The somewhat tricky bit when generating a Json file is to ensure that Megadetector defines a file path that Timelapse can use to locate the file. This usually means that the paths are identical, although an exception occurs for json files in sub-folders, as will be described shortly.

Note. *MegaDetector* should be run on a sub-folder and filename structure identical to what you will be using in Timelapse. For example, if you change some of your sub-folder names between importing images into Timelapse vs. running MeaDetector on your folder, the names will no longer match.

Running Megadetector in the root folder

If *MegaDetector* is run in the root folder, it should generate the same file paths as used by Timelapse. Thus Timelapse should be able to match the files recorded in the json file to those it knows about. This assumes that there were no changes made to the subfolder and file names between when you first loaded them into Timelapse and when you ran MegaDetector.

Technical Note. The Json file structure

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,
- 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: ...
images:
  file: Site01/IMG001.jpg               the file's location
  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
    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...
```

Running Megadetector in sub-folders

While running Megadetector in the root folder is a reasonable strategy for static images sets, there is a better approach if you plan to add new images- and run Megadetector on those new images- over time.

For example, consider how the WildlifeProject folder on the right has evolved over time: the user has added a new folder (Site02), and have incorporated those images to Timelapse using Timelapse's *File | Add image and video files to this image set*.

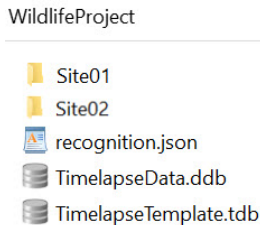
The question is: how can Site02 recognitions be created by MegaDetector so they can be imported by Timelapse?

Approach 1. Rerun Megadetector on all files. The simplest is to simply rerun Megadetector in the root folder on all files. That recognition file will 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 replace the recognitions already recorded for images in the Site1 folder.

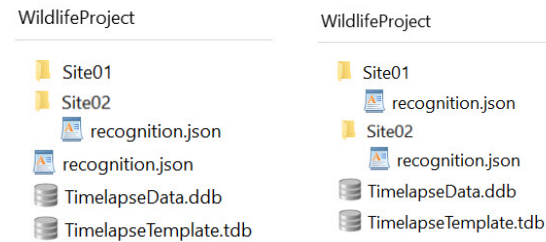
However, this approach is very inefficient, as it means you are re-running the recognizer for images that already have known recognition data. This can add considerable time to the recognition task.

Approach 2. Run Megadetector in the sub-folder. The second approach is to run Megadetector in the new Site2 sub-folder, and to place the json recognizer in that sub-folder. When you import the recognition file, Timelapse checks to see if it is in a sub-folder under the root folder. If so, it then checks the file paths to see if it can resolve the locations of the images identified in that sub-folder.

To explain, the recognition file will not include 'Site2' in an image's file path, as all its files are recorded relative to the Site2 folder. However, Timelapse can accommodate that as long as that json file is located within the Site2 folder. When you import that file using Timelapse's *File | Import image recognition data for this image set*, Timelapse recognizes that the json file is in the Site2 sub-folder relative to the root folder, and examines the image paths to see if adding a 'Site2' prefix will locate the image files. If so, it repairs the paths.



The left image illustrates how a new recognition file was created in the Site02 folder and then imported. This will work, but it does leave some organizational questions about which folders that original recognition.json file (located in the root folder) are covered. This could be handled, of course, by renaming the file (e.g., recognition-Site1.json)



The right image illustrates a better organizational approach. Here, the user has recognized that folders will be added periodically, and thus creates the recognition file within each sub-folder.

In summary:

- *If your image set is complete* where you don't expect to add any images to it over time, run Megadetector on the root folder.
- *If you plan to add images to your image set over time*, try to organize each incremental addition as its own sub-folder. Run Megadetector in that sub-folder, where you can then incrementally add those images import that json file into your image set. However, Timelapse requires that the json file is located in its corresponding sub-folder in order to repair its paths.

Trouble shooting JSON file importing

After trying to import a .json recognition file, You may 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. Debug it as follows.

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 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".

Often, a prefix to the path is either added or missing due to how files were submitted to MegaDetector. 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, MegaDetector provides a [web page and downloadable application](#) for adjusting paths, as it recognizes this as a common problem. 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.