Particle Classification

Release 2.4.0

Feb 02, 2022
Getting started

1 Overview
   1.1 MISO ......................................................... 3
   1.2 ParticleTrieur ............................................. 3

2 Getting started ........................................... 5

3 Citing ......................................................... 7

4 Table of contents ......................................... 9
   4.1 Introduction ................................................ 9
   4.2 Installation ................................................ 13
   4.3 Overview .................................................. 14
   4.4 Adding images ............................................ 17
   4.5 Selection and Sorting ................................... 21
Welcome to the help and tutorial documentation for the ParticleTrieur program and the MISO python library.
CHAPTER 1

Overview

1.1 MISO

MISO is a library of python scripts that simplify training a CNN from a set of labeled images. A variety of common CNN topologies can be chosen, such as variations of ResNet or using transfer learning. The scripts take a folder of images and output a trained model along with statistics on the model performance. The system is optimised for particle images.

Github repository

1.2 ParticleTrieur

ParticleTrieur is a cross-platform java program to help organise, label, process and classify images, particularly for particle samples such as microfossils. It can be used for both the creation of the training set required to make a CNN classifier, and classification of image using a trained CNN. It also includes some image processing functions, morphology calculations and statistical graph generation. ParticleTrieur allows the user to configure and launch training using the MISO library.

ParticleTrier is release under the open-source GPL v2 licence and the source code can be found at Github repository
• *Read* the introduction on ParticleTrieur, MISO and how to create a good training set for CNN models.
• *Install* ParticleTrieur and MISO
Citing

If using ParticleTrieur with the MISO library please cite our paper on CNNs for foraminifera classification at the Journal of Micropalaeontology

@article{jm-39-183-2020,
  author = {Marchant, R and Tetard, M and Pratiwi, A and Adebayo, M and de Garidel-Thoron, T},
  doi = {10.5194/jm-39-183-2020},
  journal = {Journal of Micropalaeontology},
  number = {2},
  pages = {183--202},
  title = {{Automated analysis of foraminifera fossil records by image classification using a convolutional neural network}},
  url = {https://jm.copernicus.org/articles/39/183/2020/},
  volume = {39},
  year = {2020}
}
4.1 Introduction

4.1.1 Overview

ParticleTrieur (PT) is a java program for organising and labelling images, particularly particle images such as micro-fossils or other organisms, but can be used for any image-level labelling task.

MISO is a python library for easy training of convolutional neural networks (CNNs). Convolutional neural network (CNN) based classification systems are currently the state-of-the-art in image classification, outperforming previous methods based on engineered features. ParticleTrieur contains a graphical user interface (GUI) for launching CNN training with the MISO library. The trained CNN model can then be imported into PT to classify new images.

The procedure to create a CNN using PT and MISO is:

1. **Creation**: Create an dataset consisting of a wide variety of images of all the different classes you wish to identify. The images should span the range of normal variations you would expect to see in each class.

2. **Labeling**: Label the images according to their class using PT. PT can suggest labels based on the images you have already labelled.

3. **Training**: Train a CNN using the labeled images. MISO comes with a range of CNN models to try.

4. **Validation**: The training scripts produce graphs of the accuracy of the network, as well as estimating if any images have been mislabeled. We use these results to tweak the CNN parameters or double-check the image labeling, respectively.

5. **Inference**: The final output of training is a *frozen* CNN model which can now be used for classifying unknown images. The model is loaded into PT to help train unseen images.

6. **Export**: The labeled or classified images information can be exported as a CSV for further analysis.

PT and MISO have many other features outside of this workflow, such as calculating particle morphology, running as server, exporting abundance counts, and so on.
4.1.2 CNN primer

Convolutional neural networks (CNNs) are very good pattern recognisers. During training they learn which image features correspond to which classes. Then once trained, they classify images according to which features are found in the image.

This leads to a few important observations:

1. **We cannot expect the CNN to accurately classify an image whose distinguishing features are NOT in the training set.**

For example, the following images show the dorsal (left) and umbilical (right) views of a *N. dutertrei* foraminifera microfossil particle. If only the dorsal view is in the training set, the CNN may have difficulty classifying images of the umbilical view, as the image contains a different set of features.

2. **Images not in the training set may be classified with high probability into one of the classes in the training set.**

The CNN tries to classify an image into one of the learnt classes, according to the features in the image. If an image is from a class that is not one of the learnt classes (i.e. an ‘invader’) it may still be classified with high “probability” as one of the learnt classes.

For example, say we had a CNN trained on the following four classes:
and used it to classify this image:

The image might be classified with high probability as belonging into the last class, due to having the same round shape. This is because the output of CNNs (trained as multi-class classifiers using softmax) are a vector, consisting of one probability value for each class, that all add up to 1. These “probabilities” are not probabilities in the sense of likelyhood of belonging to a distribution, as with some traditional classifiers. Thus we cannot use the network to flag images *not* in the training set.

3. **The CNN will learn features that differentiate each class, even if they do not belong to the object.**

The CNN sees the images as patterns of pixels, not as a 2D representation of an arrangement of objects. This mean it will learn any patterns that differentiate classes, regardless if they belong to the object of interest or not.

For example, of these five classes, only the last has the reflection of the ring light, due to its shiny surface. The CNN may learn that the presence of a ring light reflection means the image is from class 5. Therefore the CNN may have trouble recognising images of the same particle taken with a different lighting system where this pattern is not present.
Other things such as changes in background can also significantly affect classification performance if they are not in the training set.

The images used for training should cover all the variations in the class that you need to be able to predict. These variations could include:

- Intra-class variations, e.g. morphology, damage, preservation
- Position variations, e.g. pose (lying on top / side etc), rotation, location in image, size in image
- Acquisition system variations, e.g. brightness, contrast, colour, focus

Acquiring images covering all of the permutations of these variations would be difficult and time-consuming. Fortunately, we can use other techniques to reduce this load.

- Pre-processing can be used to remove variations, such as the size and location of the particle in the image. It is performed before training.
- Augmentation is used to simulate variations in the brightness, contrast, rotation, zoom and offset of the particle in the image. It is performed during training.

**Important:** The ParticleTrieur and MISO libraries perform the following pre-processing steps:

1. Rescale the image to the range 0-1 by dividing by 255 (e.g. for normal 8-bit images)
2. Resize and pad the image to the square shape required by the CNN. The padding value is the median of the pixels along the border of the image.

And the following augmentations are used by default:

- rotation
- brightness
- contrast
- zoom
4.1.3 Image acquisition

As a general guide, images of particles should be acquired:

- Using the same brightness, contrast, zoom and white balance
- With the particle centred in the image with a small buffer around them
- Without the addition of extra image features such as scale bars or captions
- With black or white borders

Enough images of each class should be captured to cover the morphological variations present, such as:

- shape
- damage
- preservation
- pose
- colour

We recommend at minimum of 50 and preferably at least 200 images per class of simple particles such as foraminifera or plankton. In particular, one must pay attention to have enough images of each pose. It may not be possible to obtain 50 images of some rare classes. It is ok to still include these in the selection, as they can be excluded later in the training procedure.

4.2 Installation

4.2.1 ParticleTrieur

ParticleTrieur is distributed as a Java JAR file. It requires the installation of Java 8 to run.

1. Install Amazon Corretto 8 or another Java 8 JRE / JDK
2. Test the java installation by opening command prompt (Windows) or terminal (macOS) and running `java -version`. It should return something like:

```bash
C:\Users\rossm>java -version
openjdk version "1.8.0_212"
OpenJDK Runtime Environment Corretto-8.212.04.2 (build 1.8.0_212-b04)
OpenJDK 64-Bit Server VM Corretto-8.212.04.2 (build 25.212-b04, mixed mode)
```

3. Download the latest release of ParticleTrieur.jar from the github repository releases page
4. Save the JAR file in a convenient location

To run ParticleTrieur:

1. Open command prompt (Windows) or terminal (macOS / linux)
2. Change directory to the one containing ParticleTrieur.jar

```
cd /PATH/TO/PARTICLETRIEUR
```

3. Execute the jar file

```
java -jar ParticleTrieur.jar
```
Note: The latest version of ParticleTrieur may be called ParticleTrieur-dev.jar or similar. Update the instructions above accordingly.

### 4.2.2 MISO

The MISO library is required to perform CNN training using ParticleTrieur but can also be used stand alone.

1. Install Anaconda from [here](#). Use the default installation options, making sure that *Install for: Just me* is selected.
2. Open the Anaconda command prompt (Windows) or terminal (MacOS / Linux).
3. Create a new python environment called *miso2*:
   
   ```
   conda create -n miso2 python=3.7
   ```
4. Activate the *miso2* environment:

   ```
   conda activate miso2
   ```
5. Install Tensorflow v1.14, the version depends on if you have an NVIDIA GPU:

   With GPU:
   
   ```
   conda install tensorflow-gpu==1.15
   pip install tensorflow-gpu==1.15.2
   ```

   Without GPU:
   
   ```
   conda install tensorflow=1.15
   pip install tensorflow==1.15.2
   ```
6. Install the MISO library:

   ```
   pip install miso2
   ```

   Setup is complete!

   To update the MISO library in the future, perform steps 2, 4 and 6, or click the *Update Library* button in the *Training* dialog in ParticleTrieur

### 4.3 Overview

The primary functions of ParticleTrieur (PT) are:

- Organise images according to their sample, resolution and other metadata.
- Label images by their taxonomic class, and tag them according to their properties.
- Use in-built AI to help predict the label of an image based on the most similar already labeled images.
- Load a external trained convolutional neural network (CNN) to help label images.
- Process particle images, such as by removing borders, normalising intensity, or centering the particle in the image.
- Calculate morphology information such as circularity or solidity.
• Export images, by processing them and sorting them into directories by their label.
• See graphs of statistics such as label counts.
• Configure and launch CNN training with the MISO particle classification library.

4.3.1 Glossary

PT allows the organisation of particle images according to their metadata. PT projects are saved in human-readable XML format.

A project consists of settings data and a list of particle metadata. The particle metadata is:

• **Filename:** The path to the image. This will be *relative to the project file* if saved on the same drive as the project file, otherwise it will be absolute. [Path (wikipedia)]

• **Sample:** The name of the sample from which the image was taken.

• **Index 1:** An index value used to sort images and generate statistics. For example, if index 1 may be set to the depth at which a foraminifera sample was taken.

• **Index 2:** A secondary index.

• **Resolution:** The resolution of the image in pixels per millimetre.

• **GUID:** A globally unique identifier for the image.

• **Classifications:** Labels and their confidence scores for this image, along with the id of the classifier. When manually labelling and image there will be only a single classification with the score set to 1.0. The classification items consist of two values:
  – **code:** The classification label code of the class.
  – **score:** The confidence score of the classification [0-1].

• **Tags:** Tags to help sort images, see below.

• **Validator:** The name of the person who validated the image label.

• **Morphology:** The calculated morphology of the particle.

• **Parameters:** All other metadata for the particle.

The project metadata also contains a list of labels used for classification and tags:

• **Labels:** Labels are the names of the various classes to which an image can be assigned. There are also non-taxonomic labels such as *unlabeled* and *unsure* which are used for images that have not been labelled or cannot be identified confidently. Taxonomic labels are used to train the CNN model.

• **Tags:** Tags are used to help sort the images, but are not used in CNN training. There are some build in tags such as *auto*, which is given to images that are automatically classified by a CNN or other, and *duplicate* which is given to images identified as duplicates by the *Find duplicates…* command.

4.3.2 Launching PT

If not already installed, follow the instructions in *Installation*

1. Open command prompt (Windows) or terminal (macOS / linux)
2. Change directory to the one containing ParticleTrieur.jar

```
cd /PATH/TO/PARTICLETRIEUR
```
3. Execute the jar file

```java
java -jar ParticleTrieur.jar
```

The startup window will show:

On the left:
- **New project** will start a new project with the default settings
- **New project from template** will prompt to open an existing project, from which a new project will be created with no images but using the same settings.
- **Open project** will prompt to open and existing project.

On the right:
**Recent projects** show as list of up to the last 5 opened projects.

### 4.3.3 Main Window

Choosing and option will start the program after a moment to load the internal AI algorithms. On the left is a list of particles and their metadata, on the right are the labelling and processing tools and common functions are on a toolbar at the top.
4.4 Adding images

4.4.1 Adding

Click *Add* to start adding images to the project. The add dialog will pop up:
There are three options:

**From folder**
Select Add images from folder to add images from a folder and its subfolders. Click Choose... to select the folder.

**From files**
Select Add images from files to add individual images. Click Choose... to select the files.

**From CSV**
Select Add images from CSV to add images from information contained in a CSV. Click Choose... to select the CSV.

The CSV must contain a column with the header file, filename, or dossier. The entries in this column must be the full path to the image, as the images will be loaded from this location.

If the CSV contains any columns with the headers sample, label, score, index1, index2, resolution or guid, the information from these columns will be added to the particle metadata. See Glossary for the meanings.

All other information in the CSV will be added to the particle parameters metadata.
**Note:** Use **Add images from CSV** to add the CSV created by the Flowcam Segmenter script. This will then add all the Flowcam morphology and other parameters for the image.

**Random selection**

You can add only a random selection of images (e.g. if the number is too large) by checking **Randomly select images** and entering the number of images in the adjacent text box.

Once images have been selected, they will appear in the the dialog window:

Click **OK** to add the images.

### 4.4.2 Metadata

Once added, the **Update Metadata** dialog will appear, allowing you to set the metadata of the added images.

On the left you can set the `sample`, `index 1`, `index 2` and `resolution` values for the images. Leaving an input box blank will ignore that metadata and leave the value for the images unchanged.
On the right is a more complex interface for extracting the metadata from the image path.

![Update Metadata](image)

Ticking **Extract image metadata from its path and filename** will enable this feature.

It works by trying to match parts of the filename and path to the main particle metadata of *label*, *sample*, *index1*, *index2*, and *GUID*.

The path used is the filename and the previous two directories. E.g. if the path to a file is

C:\Users\rossm\Data\OLZO\images\F44_80_micron\copepode\OLZO_F44_80_micron_00000135.png

the part that will be used to match is

F44_80_micron\copepode\OLZO_F44_80_micron_00000135.png

**NOTE: backslashes \ are changed to forward slashes / for consistency between Windows and macOS / Linux machines.**

In this path, the root directory is the sample, the parent directory is the label of the image, and filename is a unique value. We can match the sample and label using the following pattern:

$sample$/label/$skip$

The $ signs delineate a metadata parameter. Possible values are:

- $label$
- $sample$
- $index1$
- $index2$
- $GUID$

There are two other tokens that can be used:

- $skip$ is used to skip a part of the path.
- $end$ matches the remainder of a path.
By default, the parameters will match including the ‘_’ or dash ‘-’ characters. This can be difficult if for example, the sample or label contains underscores which are also used to separate other information. For example, consider the path:

MD972138_v2/images/MD972138_fragment_round_100_101_0001385.png

In this MD972138 is the sample, and fragment_round is the label. The correct matching pattern is:

$sample$_$skip$/$skip$/$skip$_$label$_$skip$_$skip$_$end$

Notice the multiple $skip$ and ‘_’ values around the $label$. This is needed to match the last part of the filename. If instead the following pattern was used:

$sample$_$skip$/$skip$/$skip$_$label$_$end$

The label would be matched as fragment_round_100_101 which is incorrect.

A green tick will appear next to the pattern input box if the pattern matches the entire path correctly, a red cross will appear if it doesn’t. Click New random file to show a new filename from the added images, and Test if matching to test if the pattern matches for all the filenames.

Click OK to update the metadata, or if you do not want to change anything, click Cancel

4.5 Selection and Sorting

Particle images added to the project appear on the left side of the main window:
There are three tabs:

- Images presented in a list view
- Images presented in a grid view
- A single large image of the currently select image

Above the three tabs is the filtering input box

Below the three tabs is a Select All button for conveniently selecting all the images and two buttons for decreasing / increasing the size of the images in the list and grid views.

4.5.1 List view

<table>
<thead>
<tr>
<th>#</th>
<th>Image</th>
<th>Sample</th>
<th>Label/Tag</th>
<th>Annotator</th>
<th>Filename</th>
<th>Folder</th>
<th>Sample</th>
<th>Index 1</th>
<th>Index 2</th>
<th>Result</th>
<th>GUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F44_B0_micron</td>
<td>[1,0]</td>
<td>[0,0]</td>
<td>from_csv</td>
<td>F44_B0_micron-0000001.png</td>
<td>C:\Users\rossen\Documents\Data\Plankton\OLZ01\OLZ01\images\F44_B0_micron\fermeur</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>nZEf5gYrH0SN</td>
<td>Qk9gZy</td>
</tr>
<tr>
<td>2</td>
<td>F44_B0_micron</td>
<td>[1,0]</td>
<td>[0,0]</td>
<td>cropped</td>
<td>rossem</td>
<td>F44_B0_micron-0000002.png</td>
<td>C:\Users\rossen\Documents\Data\Plankton\OLZ01\OLZ01\images\F44_B0_micron\fermeur</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>nZEf5gYrH0SN</td>
</tr>
<tr>
<td>3</td>
<td>F44_B0_micron</td>
<td>[1,0]</td>
<td>[0,0]</td>
<td>debris ni</td>
<td>from_csv</td>
<td>F44_B0_micron-0000003.png</td>
<td>C:\Users\rossen\Documents\Data\Plankton\OLZ01\OLZ01\images\F44_B0_micron\fermeur</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>nZEf5gYrH0SN</td>
</tr>
</tbody>
</table>

The list view contains a thumbnail image of the particle and the main metadata parameters. The columns are:

- **#: Image number in project**
- **Image**: Image thumbnail
- **Sample**: Sample name and indicies
  - 1st line: sample name
  - 2nd line: index 1 value
  - 3rd line: index 2 value
- **Label/Tag**: The label and tags of the image
  - 1st line: label
  - 2nd line: list of tags
  - 3rd line: the validation status - blank if not validated, green tick if validated
- **Annotator**: Username of person who labeled the image, and optionally, who validated it.
  - 1st line: label username
  - 2nd line: validator username
- **Filename**: Filename of image
- **Folder**: Folder where the file is stored
- **Resolution**: Resolution of the image in pixels per millimetre
- **GUID**: Globally unique identifier of the image

Use the Shift and Ctrl (Command on macOS) keys to select multiple images.
4.5.2 Grid view

The grid view contains a thumbnail image of the particle with the image number in the project and its label. If the image has been validated, a green tick will appear in the top right corner.

Use the Shift and Ctrl (Command on macOS) keys to select multiple images.

Note: Right-clicking on an image will show a larger version of it with more metadata.
### 4.5.3 Image view

The image view contains a large size image of the (first) selected image(s) along with the image metadata.

**Image #3**

debris ni

<table>
<thead>
<tr>
<th>Label</th>
<th>debris ni</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tags:</td>
<td></td>
</tr>
<tr>
<td>Classifier</td>
<td>from_csv</td>
</tr>
<tr>
<td>Validated:</td>
<td>rossm</td>
</tr>
<tr>
<td>Sample:</td>
<td>F44.80_micron</td>
</tr>
<tr>
<td>Filename:</td>
<td>OLZO_F44.80_micron_00000003.png</td>
</tr>
<tr>
<td>Path:</td>
<td>C:\Users\rossm\Documents\Data\Plankton\OLZO\OLZO\images\F44.80_micron\debris ni</td>
</tr>
<tr>
<td>Info:</td>
<td>74 x 64 pixels</td>
</tr>
<tr>
<td>GUID:</td>
<td>6Wq4QrGvYYxMjF8cSPV1Js</td>
</tr>
</tbody>
</table>

The image view contains a large size image of the (first) selected image(s) along with the image metadata.

### 4.5.4 Filtering

The filtering input box can be used to filter the list of images according to their metadata. Multiple filtering options can be concatenated to narrow down the results.

The possible filtering fields are:

- # (image number)
- sample
• label
• tag
• index1
• index2
• file
• folder
• guid
• valid (if the image has been validated, use true or false)

The possible filtering operators are:

• == contains
• != does not contain
• === exact match
• !== not an exact match

Enter a combination of fields to filter the list, for example:

• label==fragment returns all images with the label fragment as part of their name, e.g.*fragment_round* or fragment_bilobal.
• label===fragment returns all the images with label fragment exactly.
• label!=fragment valid==true returns all the images whose label does not contain fragment and have been validated.

**Important:** Values with spaces in them need to be replaced with underscores ‘_’. For example, if you wish to filter for images with label fragment round then the filter would be label==fragment_round
<table>
<thead>
<tr>
<th>#</th>
<th>Gasteropode</th>
<th>#</th>
<th>Gasteropode</th>
<th>#</th>
<th>Gasteropode</th>
<th>#</th>
<th>Gasteropode</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td><img src="image1" alt="Gasteropode 2" /></td>
<td>9</td>
<td><img src="image2" alt="Gasteropode 9" /></td>
<td>10</td>
<td><img src="image3" alt="Gasteropode 10" /></td>
<td>23</td>
<td><img src="image4" alt="Gasteropode 23" /></td>
</tr>
<tr>
<td>25</td>
<td><img src="image5" alt="Gasteropode 25" /></td>
<td>31</td>
<td><img src="image6" alt="Gasteropode 31" /></td>
<td>34</td>
<td><img src="image7" alt="Gasteropode 34" /></td>
<td>37</td>
<td><img src="image8" alt="Gasteropode 37" /></td>
</tr>
</tbody>
</table>