Orange3 Image Analytics Documentation

Biolab

Mar 01, 2022

Contents

CHAPTER 1

Widgets

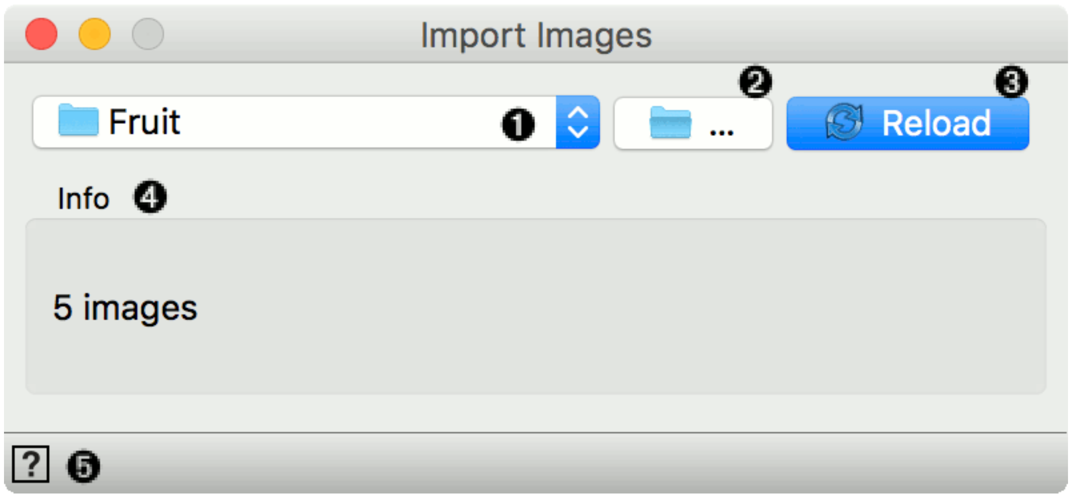
1.1 Import Images

Import images from a directory(s).

Outputs

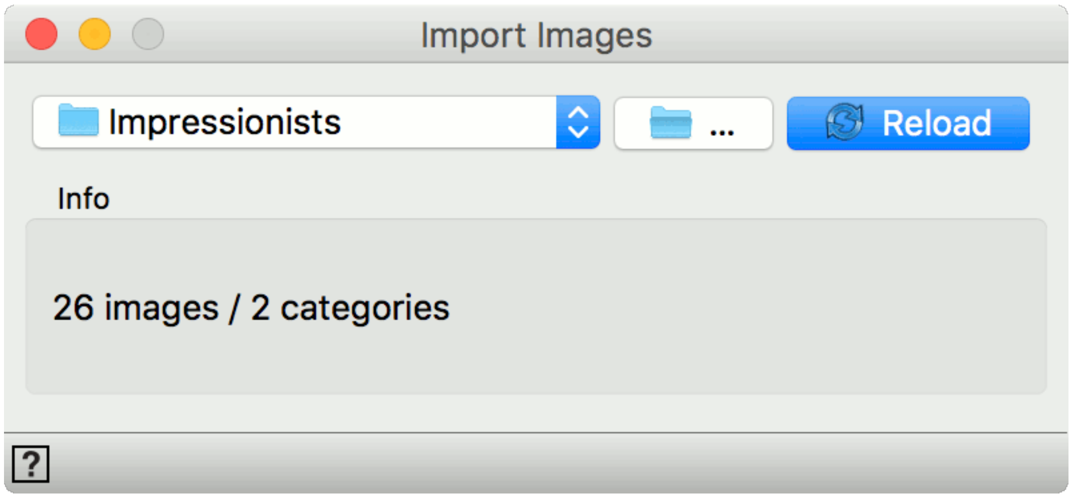
- Data: Dataset describing one image in each row.

Import Images walks through a directory and returs one row per located image. Columns include image name, path to image, width, height and image size. Column with image path is later used as an attribute for image visualization and embedding.



1. Currently loaded folder.
2. Select the folder to load.
3. Click *Reload* to update imported images.
4. Information on the input.

5. Access help.



26 images / 2 categories

You can load a folder containing subfolders. In this case Orange will consider each folder as a class value. In the example above, **Import Images** loaded 26 images belonging to two categories. These two categories will be used as class values.

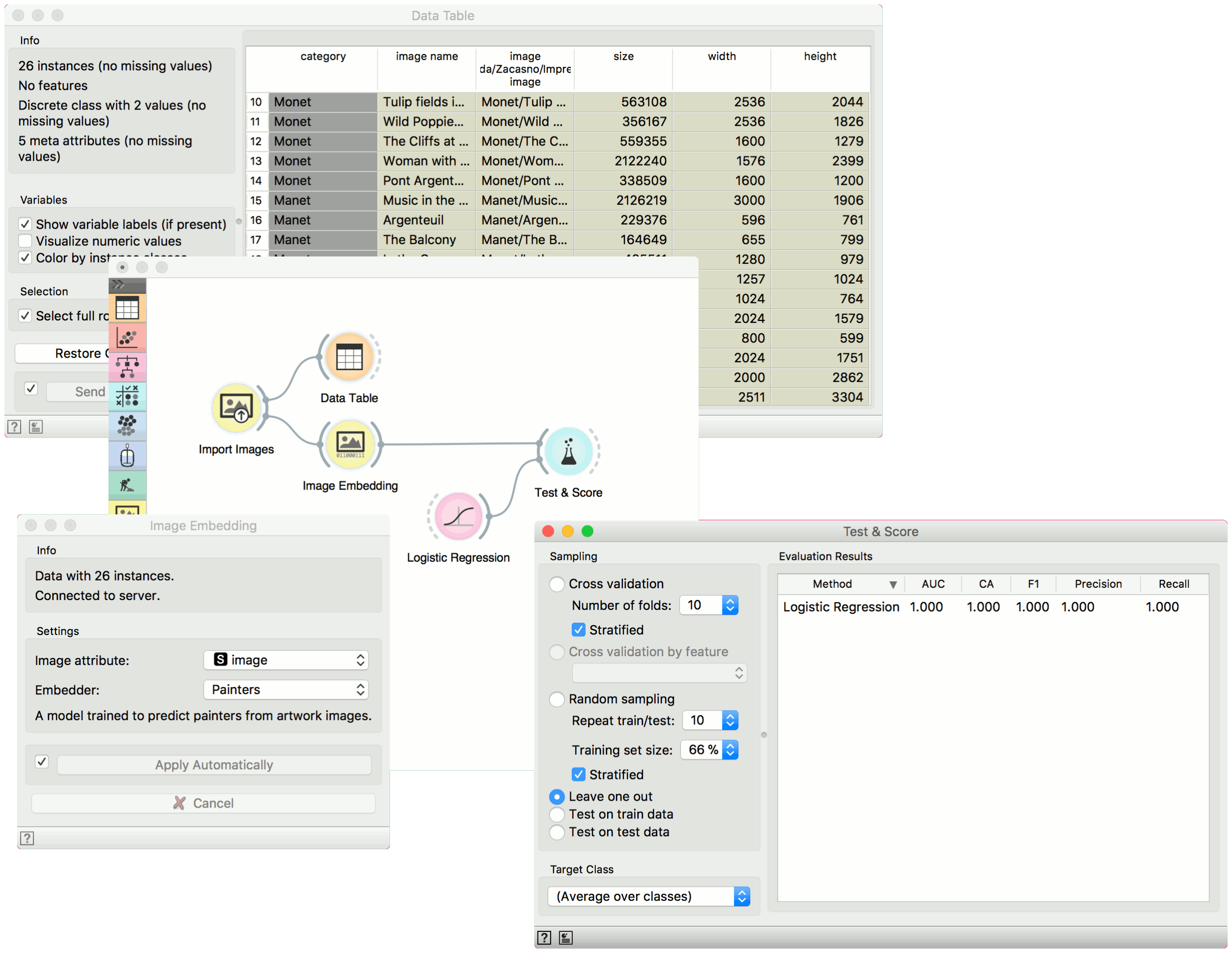
1.1.1 Example

Import Images is likely the first widget you will use in image analysis. It loads images and creates class values from folders. In this example we used **Import Images** to load 26 painting belonging to either Monet or Manet.

We can observe the result in a **Data Table**. See how Orange added an extra class attribute with values Monet and Manet?

Now we can proceed with standard machine learning methods. We will send images to Image Embedding, where we will use *Painters* embedder to retrieve image vectors.

Then we will use **Test & Score** and **Logistic Regression**, to build a model for predicting the author of a painting. We get a perfect score? How come? It turns out, these were the images the *Painters* embedder was trained on, so a high accuracy is expected.



Data Table

Info

26 instances (no missing values)
No features
Discrete class with 2 values (no missing values)
5 meta attributes (no missing values)

Variables

✓ Show variable labels (if present)
Visualize numeric values
✓ Color by instance classes

Selection

✓ Select full rows

	category	image name	image da/Zacasno/Impre image	size	width	height
10	Monet	Tulip fields i...	Monet/Tulip ...	563108	2536	2044
11	Monet	Wild Poppie...	Monet/Wild ...	356167	2536	1826
12	Monet	The Cliffs at ...	Monet/The C...	559355	1600	1279
13	Monet	Woman with ...	Monet/Wom...	2122240	1576	2399
14	Monet	Pont Argent...	Monet/Pont ...	338509	1600	1200
15	Manet	Music in the ...	Manet/Music...	2126219	3000	1906
16	Manet	Argenteuil	Manet/Argen...	229376	596	761
17	Manet	The Balcony	Manet/The B...	164649	655	799
					1280	979
					1257	1024
					1024	764
					2024	1579
					800	599
					2024	1751
					2000	2862
					2511	3304

Image Embedding

Info

Data with 26 instances.
Connected to server.

Settings

Image attribute: image
Embedder: Painters
A model trained to predict painters from artwork images.

✓ Apply Automatically

Cancel

Test & Score

Sampling

Cross validation
Number of folds: 10
✓ Stratified
Cross validation by feature
Random sampling
Repeat train/test: 10
Training set size: 66 %
✓ Stratified
[✓ Leave one out]
Test on train data
Test on test data

Target Class

(Average over classes)

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Logistic Regression	1.000	1.000	1.000	1.000	1.000

1.2 Image Viewer

Displays images that come with a data set.

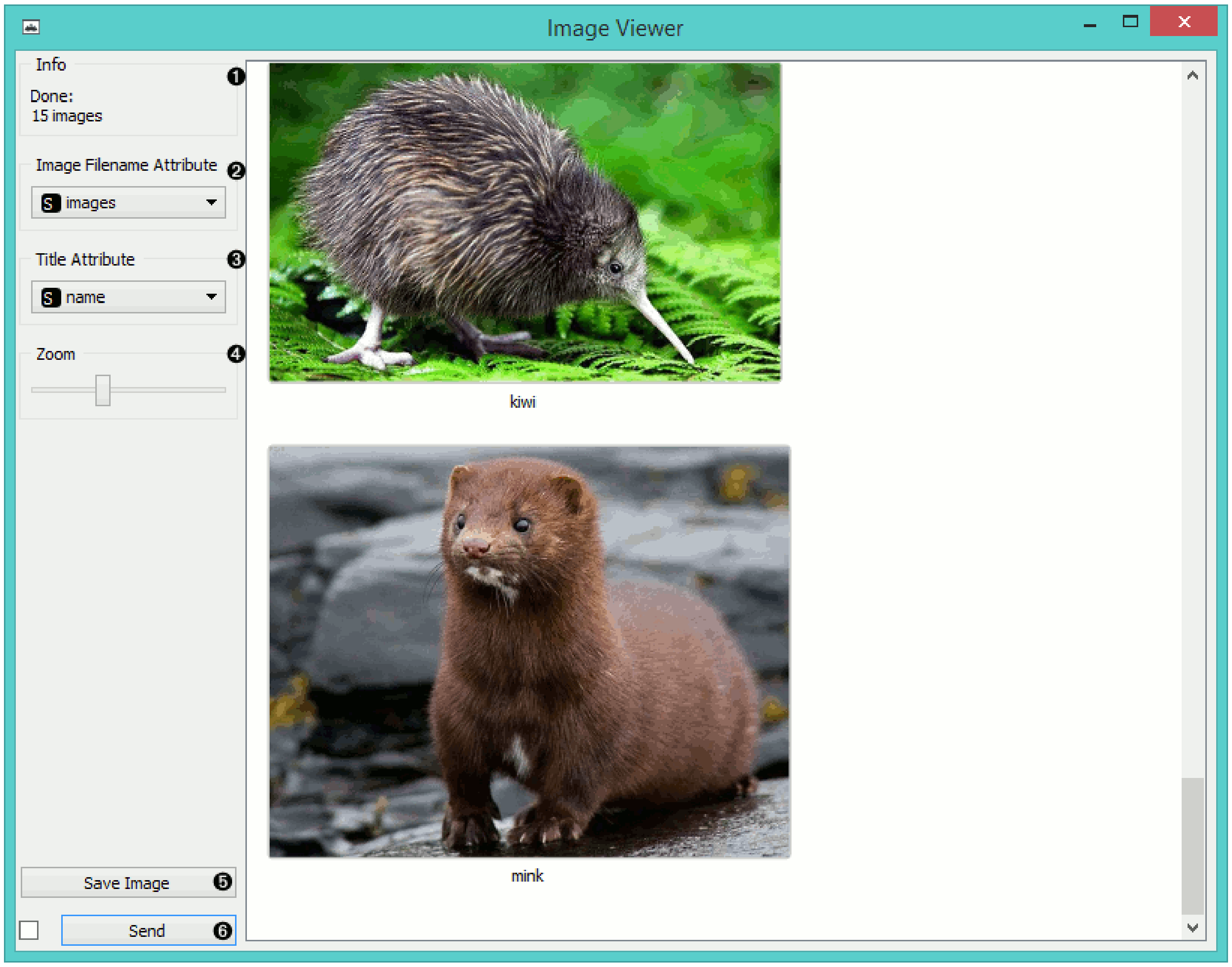
Inputs

- Data: A data set with images.

Outputs

- Data: Images that come with the data.
- Selected images: Images selected in the widget.

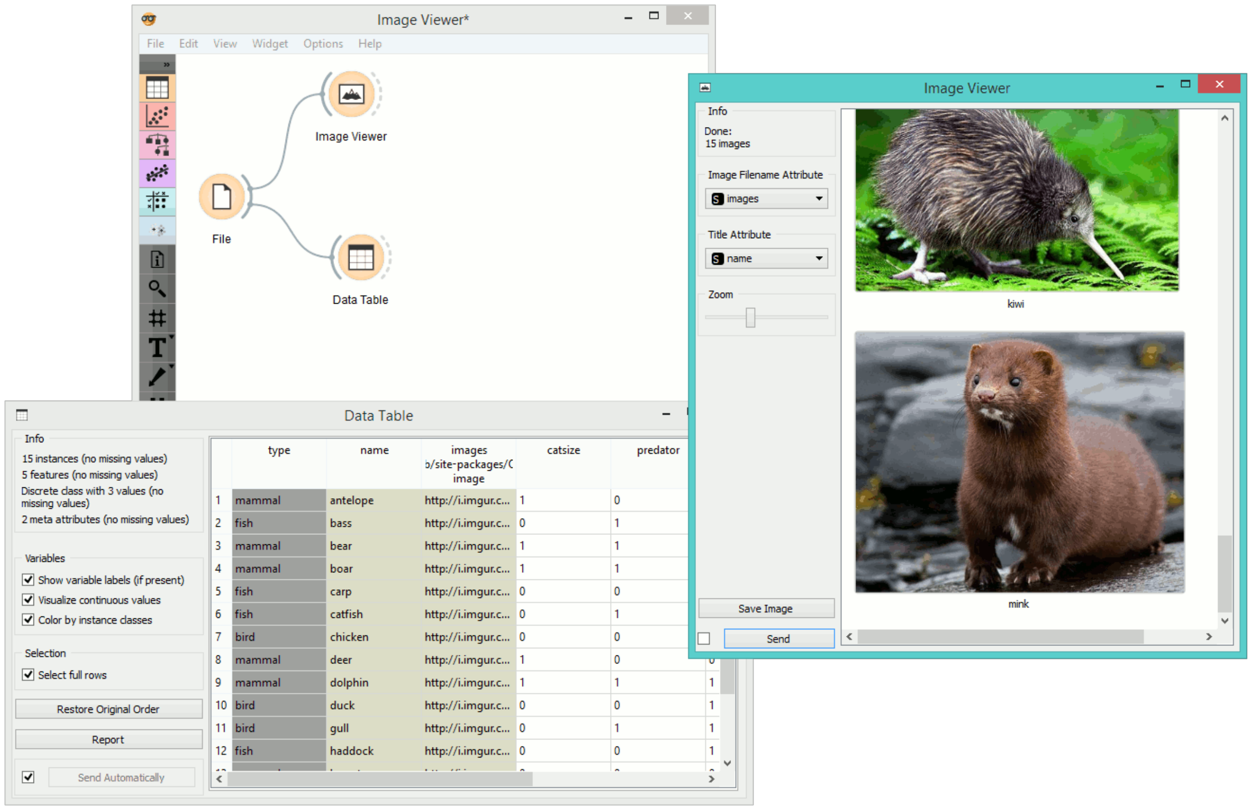
The **Image Viewer** widget can display images from a data set, which are stored locally or on the internet. The widget will look for an attribute with *type=image* in the third header row. It can be used for image comparison, while looking for similarities or discrepancies between selected data instances (e.g. bacterial growth or bitmap representations of handwriting).



1. Information on the data set
2. Select the column with image data (links).
3. Select the column with image titles.
4. Zoom in or out.
5. Saves the visualization in a file.
6. Tick the box on the left to commit changes automatically. Alternatively, click *Send*.

1.2.1 Examples

A very simple way to use this widget is to connect the **File** widget with **Image Viewer** and see all the images that come with your data set. You can also visualize images from Import Images.



Data Table

Info
15 instances (no missing values)
5 features (no missing values)
Discrete class with 3 values (no missing values)
2 meta attributes (no missing values)

Variables
☑ Show variable labels (if present)
☑ Visualize continuous values
☑ Color by instance classes

Selection
☑ Select full rows

Restore Original Order
Report
☑ Send Automatically

	type	name	images b/site-packages/C image	catsize	predator	
1	mammal	antelope	http://i.imgur.c...	1	0	
2	fish	bass	http://i.imgur.c...	0	1	
3	mammal	bear	http://i.imgur.c...	1	1	
4	mammal	boar	http://i.imgur.c...	1	1	
5	fish	carp	http://i.imgur.c...	0	0	
6	fish	catfish	http://i.imgur.c...	0	1	
7	bird	chicken	http://i.imgur.c...	0	0	
8	mammal	deer	http://i.imgur.c...	1	0	0
9	mammal	dolphin	http://i.imgur.c...	1	1	1
10	bird	duck	http://i.imgur.c...	0	0	1
11	bird	gull	http://i.imgur.c...	0	1	1
12	fish	haddock	http://i.imgur.c...	0	0	1

Image Viewer

Info
Done:
15 images

Image Filename Attribute
images

Title Attribute
name

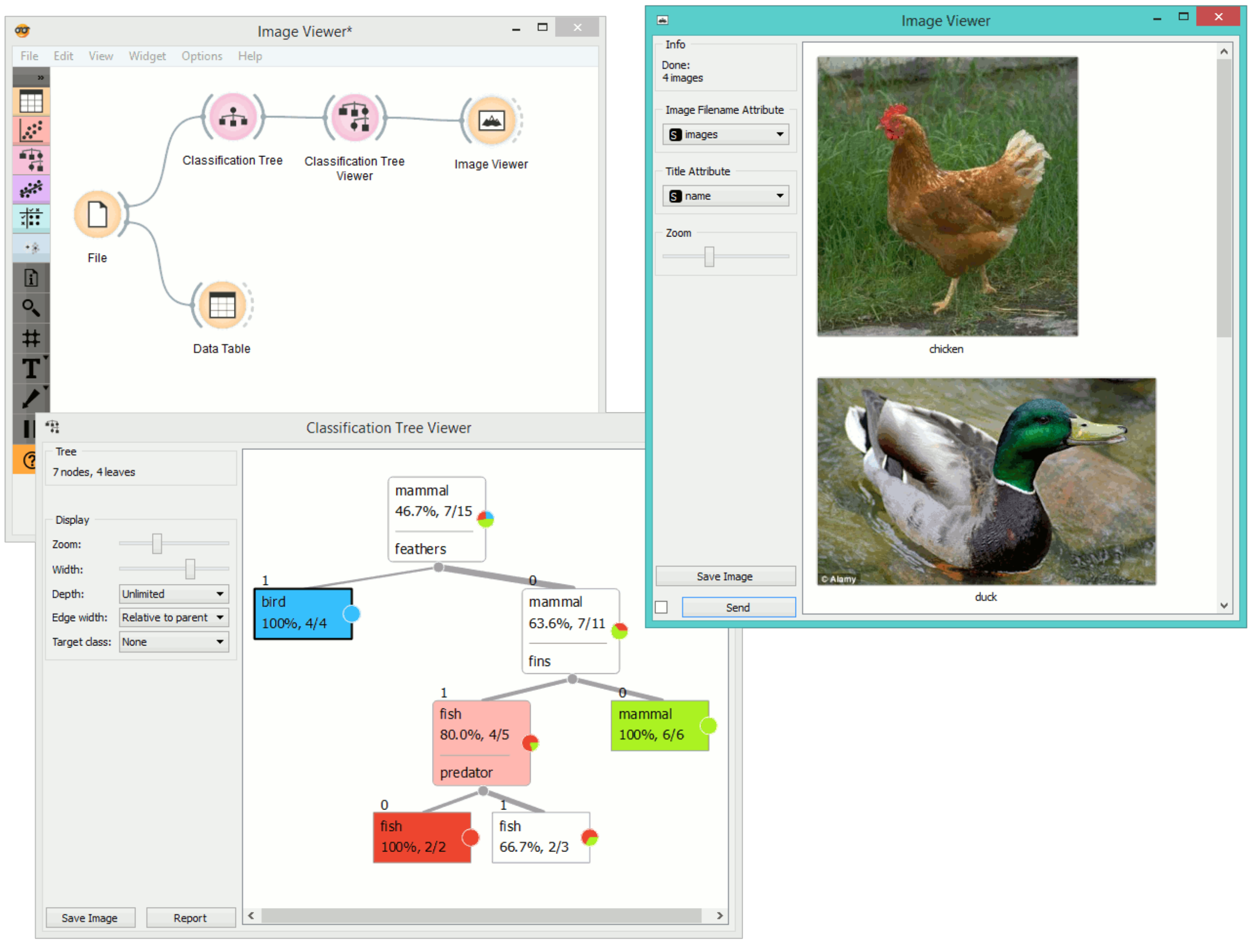
Zoom

kiwi

mink

Save Image
☐ Send

Alternatively, you can visualize only selected instances, as shown in the example below.



Tree
7 nodes, 4 leaves

Display
Zoom:
Width:
Depth: Unlimited
Edge width: Relative to parent
Target class: None

Info
Done:
4 images

Image Filename Attribute: images

Title Attribute: name

Zoom

1.3 Image Embedding

Image embedding through deep neural networks.

Inputs

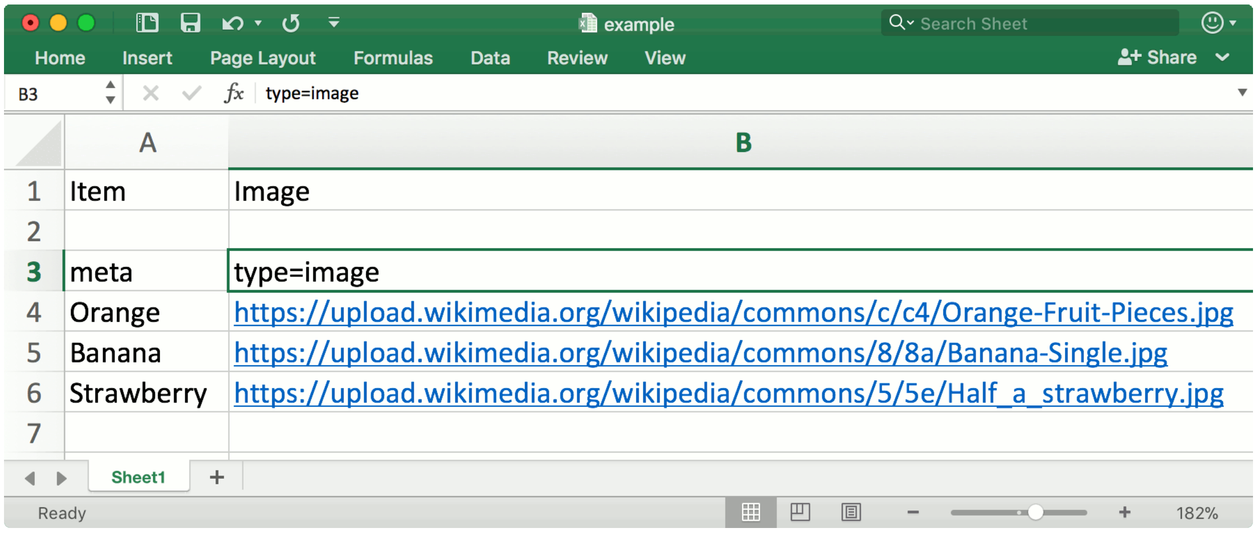
- Images: List of images.

Outputs

- Embeddings: Images represented with a vector of numbers.
- Skipped Images: List of images where embeddings were not calculated.

Image Embedding reads images and uploads them to a remote server or evaluate them locally. Deep learning models are used to calculate a feature vector for each image. It returns an enhanced data table with additional columns (image descriptors).

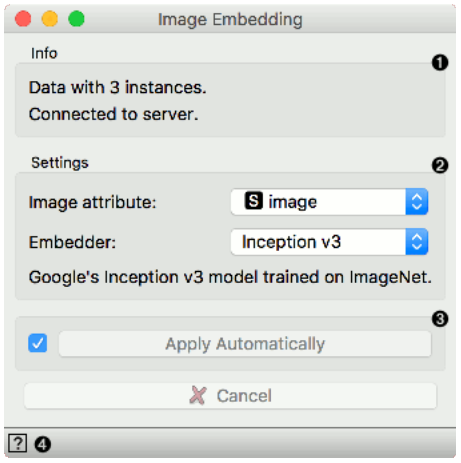
Images can be imported with Import Images widget or as paths to images in a spreadsheet. In this case the column with images paths needs a three-row header with *type=image* label in the third row.



B3 fx type=image

	A	B
1	Item	Image
2		
3	meta	type=image
4	Orange	https://upload.wikimedia.org/wikipedia/commons/c/c4/Orange-Fruit-Pieces.jpg
5	Banana	https://upload.wikimedia.org/wikipedia/commons/8/8a/Banana-Single.jpg
6	Strawberry	https://upload.wikimedia.org/wikipedia/commons/5/5e/Half_a_strawberry.jpg
7		

Image Embedding offers several embedders, each trained for a specific task. Images are sent to a server or they are evaluated locally on the user's computer, where vectors representations are computed. SqueezeNet embedder offers a fast evaluation on users computer which does not require an internet connection. If you decide to use other embedders than SqueezeNet, you will need an internet connection. Images sent to the server are not stored anywhere.



Data with 3 instances.
Connected to server.

Image attribute: image
Embedder: Inception v3
Google's Inception v3 model trained on ImageNet.

1. Information on the number of embedded images and images skipped.
2. Settings:
   - *Image attribute*: attribute containing images you wish to embed
   - *Embedder*:
     - SqueezeNet: Small and fast model for image recognition trained on ImageNet.
     - Inception v3: Google's Inception v3 model trained on ImageNet.
     - VGG-16: 16-layer image recognition model trained on ImageNet.
     - VGG-19: 19-layer image recognition model trained on ImageNet.
     - Painters: A model trained to predict painters from artwork images.
     - DeepLoc: A model trained to analyze yeast cell images.
3. Tick the box on the left to start the embedding automatically. Alternatively, click *Apply*. To cancel the embedding, click *Cancel*.

4. Access help.

1.3.1 Embedders

InceptionV3 is Google's deep neural network for image recognition. It is trained on the ImageNet data set. The model we are using is available here. For the embedding, we use the activations of the penultimate layer of the model, which represents images with vectors.

SqueezeNet is a deep model for image recognition that achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters. The model is trained on the ImageNet dataset. We re-implemented the SqueezeNet by using weights from the author's pretrained model. We use activations from pre-softmax (`flatten10`) layer as an embedding.

VGG16 and **VGG19** are deep neural networks for image recognition proposed by Visual Geometry Group from the University of Oxford. They are trained on the ImageNet data set. We use a community implementation of networks with original weights. As an embedding, we use activations of the penultimate layer - `fc7`.

Image Embedding also includes **Painters**, an embedder that was trained on 79,433 images of paintings by 1,584 painters and won Kaggle's Painter by Numbers competition. Activations of the penultimate layer of the network are used as an embedding.

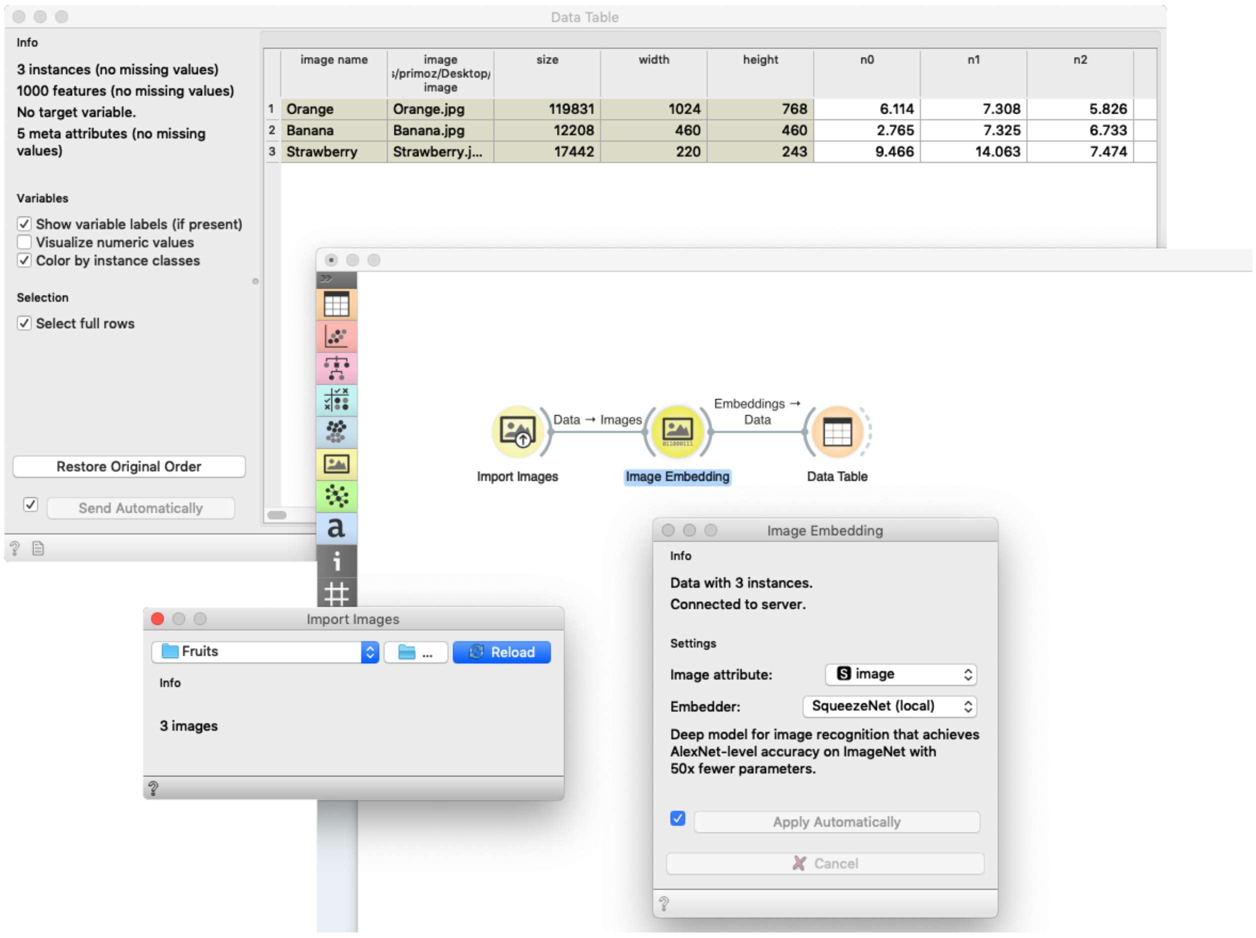
DeepLoc is a convolutional network trained on 21,882 images of single cells that were manually assigned to one of 15 localization compartments. We use the pre-trained network proposed by authors. The embeddings are activations of penultimate layer `fc_2`.

An article by Godec et al. (2019) explains how the embeddings work and how to use it in Orange.

1.3.2 Example

Let us first import images from a folder with Import Images. We have three images of an orange, a banana and a strawberry in a folder called Fruits. From **Import Images** we will send a data table containing a column with image paths to **Image Embedding**.

We will use the default embedder *SqueezeNet*. The widget will automatically start retrieving image vectors from the server.


Data Table

Info

3 instances (no missing values)
1000 features (no missing values)
No target variable.
5 meta attributes (no missing values)

Variables

✓ Show variable labels (if present)
Visualize numeric values
✓ Color by instance classes

Selection

✓ Select full rows

	image name	image ;/primoz/Desktop/ image	size	width	height	n0	n1	n2
1	Orange	Orange.jpg	119831	1024	768	6.114	7.308	5.826
2	Banana	Banana.jpg	12208	460	460	2.765	7.325	6.733
3	Strawberry	Strawberry.j...	17442	220	243	9.466	14.063	7.474

Import Images

Fruits

Info

3 images

Image Embedding

Info

Data with 3 instances.
Connected to server.

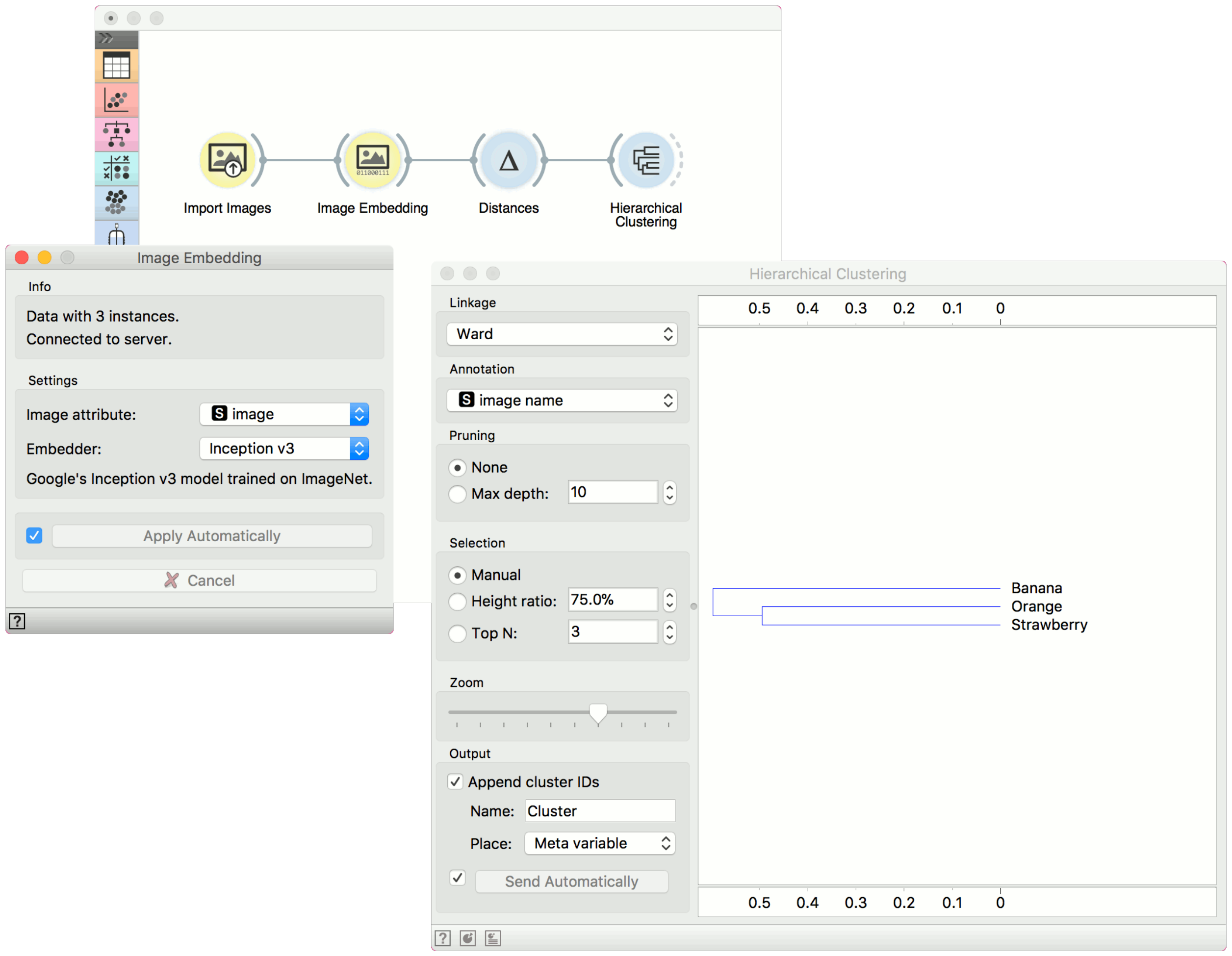
Settings

Image attribute: image

Embedder: SqueezeNet (local)

Deep model for image recognition that achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters.

Once the computation is done, you can observe the enhanced data in a **Data Table**. With the retrieved embeddings, you can continue with any machine learning method Orange offers. Below is an example for clustering.



Info

Data with 3 instances.
Connected to server.

Settings

Image attribute: image

Embedder: Inception v3

Google's Inception v3 model trained on ImageNet.

Linkage: Ward

Annotation: image name

Pruning: [● None] Max depth: 10

Selection: [● Manual] Height ratio: 75.0% Top N: 3

Zoom

Output

✓ Append cluster IDs

Name: Cluster

Place: Meta variable

1.4 Image Grid

Displays images in a similarity grid.

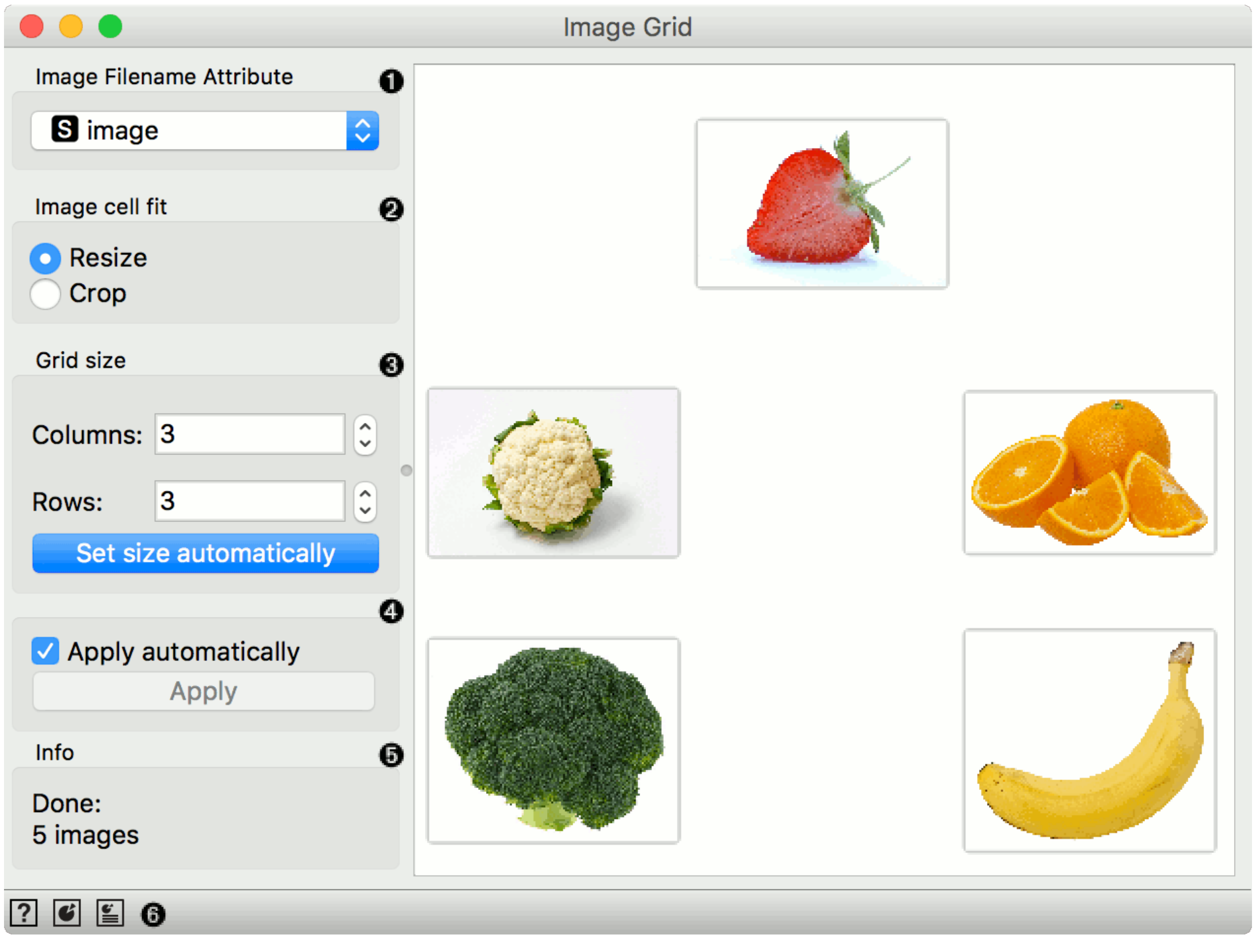
Inputs

- Embeddings: Image embeddings from Image Embedding widget.
- Data Subset: A subset of embeddings or images.

Outputs

- Images: Images from the dataset with an additional column specifying if the image is selected or the group, if there are several.
- Selected Images: Selected images with an additional column specifying the group.

The **Image Grid** widget can display images from a dataset in a similarity grid - images with similar content are placed closer to each other. It can be used for image comparison, while looking for similarities or discrepancies between selected data instances (e.g. bacterial growth or bitmap representations of handwriting).



1. *Image Filename Attribute*: Attribute containing paths to images.
2. *Image cell fit*: Resize scales the images to grid, while Crop crops them to squares.
3. *Grid size*: Set the size of the grid. Click *Set size automatically* to optimize the projection.
4. Tick the box to commit the changes automatically. Alternatively, click *Apply*.
5. Information on the input.
6. Access help, save image, and report (in that order).

1.4.1 Example

Image Grid can be used to visualize similarity of images in a 2D projection. We have used 5 images of fruits and vegetables, namely orange, banana, strawberry, broccoli and cauliflower.

We loaded the images with Import Images and embedded them with Inception v3 embedder in Image Embedding.

Finally, we visualized the images in **Image Grid**. It is obvious that broccoli and cauliflower and much more alike than strawberry and banana.

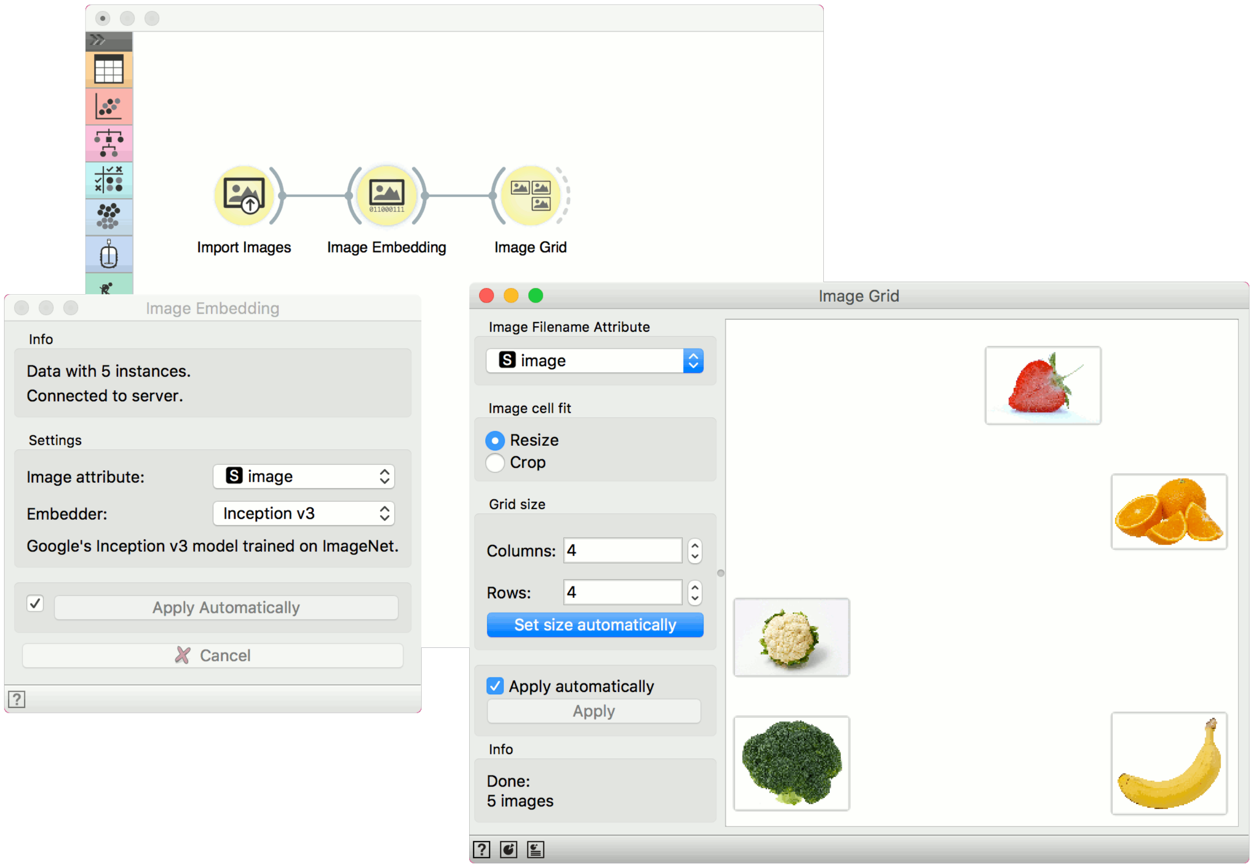


Image Embedding

Info

Data with 5 instances.
Connected to server.

Settings

Image attribute: image

Embedder: Inception v3

Google's Inception v3 model trained on ImageNet.

[✓] Apply Automatically

Cancel

Image Grid

Image Filename Attribute: image

Image cell fit: [✓ Resize] Crop

Grid size

Columns: 4

Rows: 4

Set size automatically

[✓] Apply automatically

Apply

Info

Done:
5 images

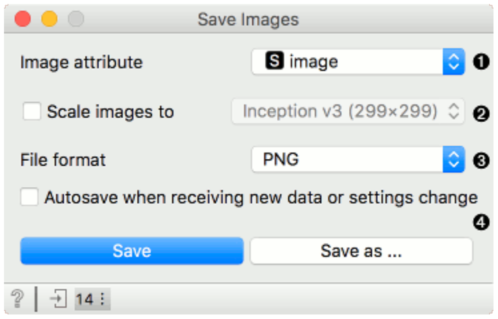
1.5 Save Images

Save images in the directory structure.

Inputs

- Data: images to save.

Save Images is a simple widget that saves images sent to its input. Images will be saved as separate files in their own directory. When a class is present in the data, images will be saved in subdirectories based on the class variable.



Save Images

Image attribute: image ❶

[ ] Scale images to: Inception v3 (299×299) ❷

File format: PNG ❸

[ ] Autosave when receiving new data or settings change ❹

Save | Save as ...

1. Attribute containing the path to the image.
2. If *Scale images to* is ticked, images will be resized to the size used in the selected embedder:

   - Inception v3: 299x299

   - SqueezeNet: 227x227
   - VGG-16: 224x224
   - VGG-19: 224x224
   - Painters: 256x256
   - DeepLoc: 64x64
   - openface: 256x256

3. File format to save images in. See the next section for information on supported formats.
4. If *Autosave when receiving new data or settings change* is on, images will be saved upon every change. *Save* will save images, while *Save as…* enables the user to set the name and the folder where to save the images.

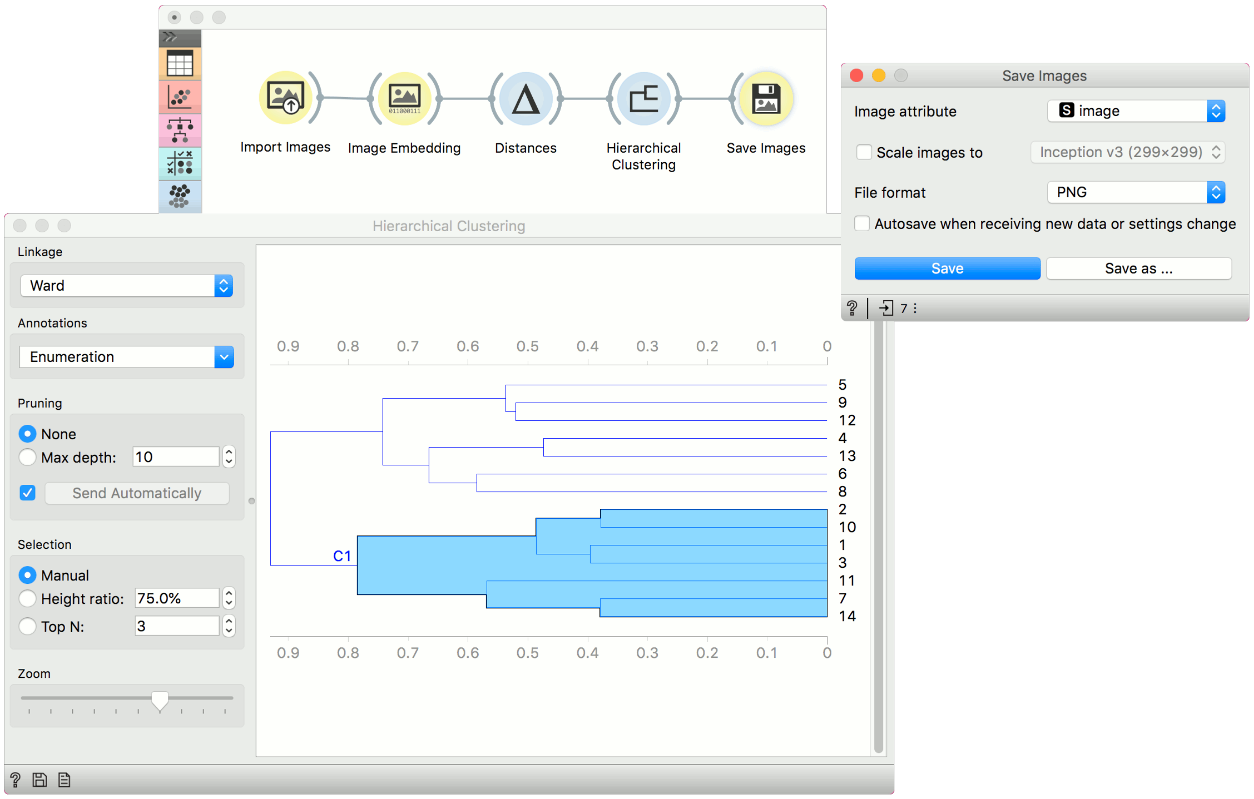
1.5.1 Supported Formats

Save Images can save images in the following formats:

- .png
- .jpeg
- .gif
- .tiff
- .pdf
- .bmp
- .eps
- .ico

1.5.2 Example

Here is a simple example how to use **Save Images**. We loaded 14 paintings from Picasso, sent them to Image Embedding using *Painters* embedder, then to Distances using cosine distance and finally to Hierarchical Clustering to construct a dendrogram. Then we selected a cluster from the plot and saved the images belonging to the selected cluster with **Save Images**.



Save Images

Image attribute: image
[ ] Scale images to: Inception v3 (299×299)
File format: PNG
[ ] Autosave when receiving new data or settings change

Hierarchical Clustering

Linkage: Ward
Annotations: Enumeration
Pruning: [✓ None] Max depth: 10
[✓] Send Automatically
Selection: [✓ Manual] Height ratio: 75.0% Top N: 3

CHAPTER 2

Scripting

2.1 Image Embedding module

class orangecontrib.imageanalytics.image_embedder.**ImageEmbedder**(*model: str = 'inception-v3', server_url: str = 'https://api.garaza.io/'*)

Client side functionality for accessing a remote image embedding backend.

model
Name of the model, must be one from MODELS dictionary

server_url
The url of the server with embedding backend.

Examples

```
>>> import Orange
>>> from orangecontrib.imageanalytics.image_embedder import ImageEmbedder
```

```
>>> # embedding from list of paths
>>> image_file_paths = ['image001.jpg', 'image001.jpg']
>>> with ImageEmbedder(model='model_name') as emb:
...     embeddings = emb(image_file_paths)
```

```
>>> # embedding from orange tabl
>>> table = Orange.data.Table('Table_with_image_path.csv')
>>> with ImageEmbedder(model='model_name') as emb:
...     embeddings = emb(table, col="image_path_column")
```

`clear_cache`() → None

Function clear cache for the selected embedder. If embedder is loaded cache is cleaned from its dict otherwise we load cache and clean it from file.

static `construct_output_data_table`(*embedded_images: Orange.data.table.Table*, *embeddings_: numpy.ndarray*) → Orange.data.table.Table

Join the orange table with embeddings.

Parameters

- `embedded_images` – Table with images that were successfully embedded
- `embeddings` – Embeddings for images from table

Returns

Return type Table with added embeddings to data.

`from_table`(*data: Orange.data.table.Table*, *col: Union[str*, *Orange.data.variable.Variable] = 'image'*, *callback: Callable = None*) → Tuple[Orange.data.table.Table, Orange.data.table.Table, int]

Calls embedding when data are provided as a Orange Table.

Parameters

- `data` – Data table with image paths
- `col` – The column with image paths
- `callback` – Optional callback - function that is called for every embedded image and is used to report the progress.

`is_local_embedder`() → bool

Tells whether selected embedder is local or not.

static `prepare_output_data`(*input_data: Orange.data.table.Table, embeddings_: List[List[float]]*) → Tuple[Orange.data.table.Table, Orange.data.table.Table, int]

Prepare output data when data table on input.

Parameters

- `input_data` – The table with original data that are joined with embeddings
- `embeddings` – List with embeddings

Returns

- *Tuple where first parameter is table with embedded images, the second*
- *table with skipped images and third the number of skipped images.*

`set_canceled`() → None

Cancel the embedding

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index