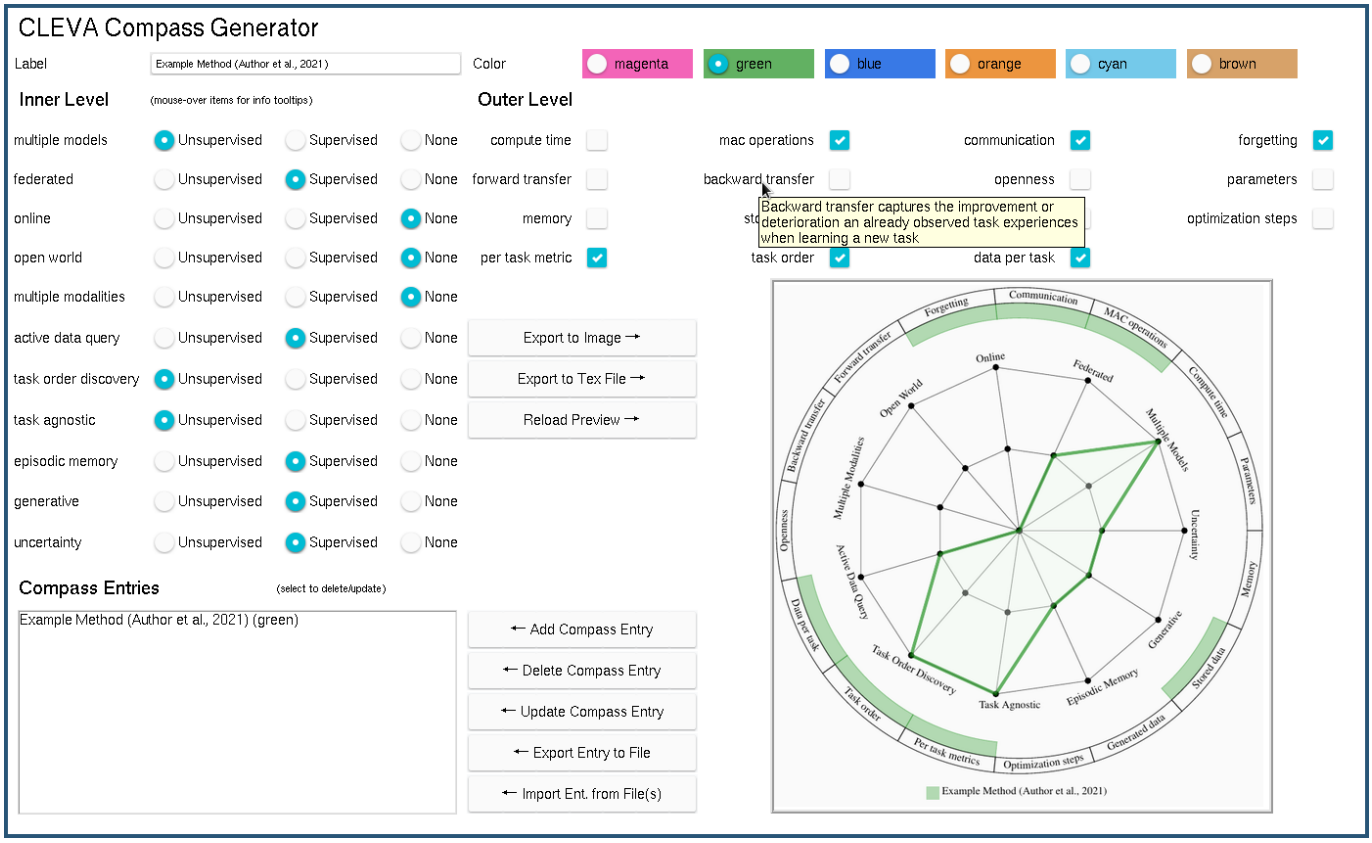
The Continual Learning EValuation Assessment (CLEVA) Compass

To make the CLEVA-Compass as accessible as possible and disseminate in a convenient way, we provide three options for practical use.

1. We provide a **LaTeX template** for the CLEVA-Compass, making use of the TikZ library to draw the compass within LaTeX. We envision that such a template makes it easy for other authors to include a compass into their future submission, where they can adapt the naming and values of the entries respectively.
2. We further provide a **Python script** to generate the CLEVA-Compass. In fact, because the use of drawing in LaTeX with TikZ may be unintuitive for some, we have written a Python script that automatically fills the above LaTeX template, so that it can later simply be included into a LaTeX document. The Python script takes a path to a JSON file that needs to be filled by the user with the CLEVA-Compass options. We further provide a default JSON file that is easy to adapt.
3. To further lower the barrier for dissemination and use, we also provide a **CLEVA-Compass Graphical User Interface (GUI)**. The GUI makes it easy for users to simply "click together" their desired compasses, save images or export to LaTeX, and conversely import already existing compass

Create the CLEVA-Compass using the GUI



The CLEVA-Compass GUI can be run using Python: `python cleva_gui.py`. The application requires at least `TkInter` to be installed (Ubuntu: `apt install python3-tk`, Fedora: `dnf install python3-tkinter`, Arch Linux: `pacman -S tk`, MacOS: `brew install python-tk`). For the interactive visualization of the CLEVA-Compass, the system-wide Poppler library is necessary (Ubuntu: `apt install poppler-utils`, Fedora: `dnf install poppler`, Arch Linux: `pacman -S poppler`, MacOS: `brew install poppler`) as well as a few additional Python dependencies are required (see `gui_requirements.txt`, install with `pip install -r gui_requirements.txt`).

The GUI exposes tooltips on mouse-hover to display information on button actions and inner/outer level options. The main idea is that users create their own CLEVA-Compass visualization by interactively generating entries for specific methods. The list of current entries is shown on the bottom right in the GUI. A method entry consists of a unique label, a selected color, and inner/outer level options. If all is set, the `Add Compass Entry` button can be pressed and the entry will be listed below. Entries can be deleted when selected with the `Delete Compass Entry` button. A selected entry can also be change and its updated options stored when the `Update Compass Entry` button is pressed. The Compass preview can be generated explicitly, based on the current set of entries, using the `Reload Preview` button. Furthermore, entries can be imported (`Import Ent. from File(s)`) and exported (`Export Entry to File`) as a JSON file for serialization purposes, as well as SVG/PNG images (`Export to Image`) or as TikZ LaTeX code (`Export to Tex File`) which can be readily included into LaTeX documents.

Create the CLEVA-Compass using the Python Script

You can use the `create_compass.py` python script to generate a compass and specify how it is filled for each continual approach in a JSON file:

```
$ python create_compass.py -h
usage: create_compass.py [-h] [--template TEMPLATE] [--output OUTPUT] [--data DATA]

CLEVA-Compass Generator.

optional arguments:
  -h, --help           show this help message and exit
  --template TEMPLATE  Tikz template file. (default: cleva_template.tex)
  --output OUTPUT      Tikz filled output file. (default: cleva_filled.tex)
  --data DATA          Entries as JSON file. (default: data.json)
```

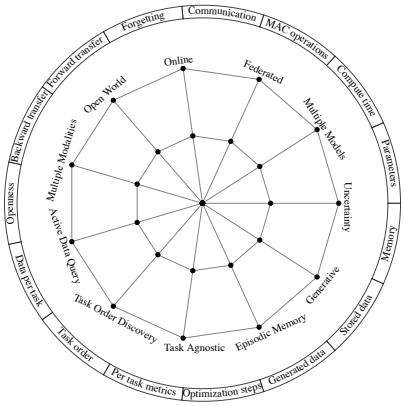
For this purpose we provide the template file `cleva_template.tex`.

Example Usage

The default reads the template file from `cleva_template.tex` and writes the filled output file into `cleva_filled.tex` with the data specified via `--data <JSON_FILE>`:

```
$ python --data examples/compass_data_0.json
```

An "empty" CLEVA-Compass, generated with `examples/compass_data_0.json`, looks like this:



In the next section, we specify how to add methods to the json file to fill the compass.

JSON Data Format

The JSON file specifies a list of `entries`, where each element defines a `color`, `label` (can contain escaped TeX commands, such as citations), `inner_level`, and `outer_level`. The latter two specify the attributes visualized in the compass.

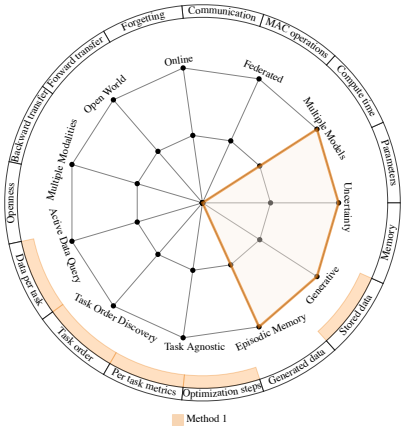
- `color`: Can be one of `["magenta", "green", "blue", "orange", "cyan", "brown"]`.
- `label`: A label describing the compass entry (can contain arbitrary, escaped TeX commands such as citations).
- `inner_level`: Specifies the inner compass level attributes. Attribute values must be on of:
  - `0`: does not apply
  - `1`: supervised
  - `2`: unsupervised
- `outer_level`: Specifies the outer compass level attributes. Attribute values must boolean (`true` / `false`).

The `compass_data_1.json` file is given as an example:
`json { "entries": [ { "color": "magenta", "label": "FedWeIT", "inner_level": { "multiple_models": 1, "federated": 2, "online": 0, "`

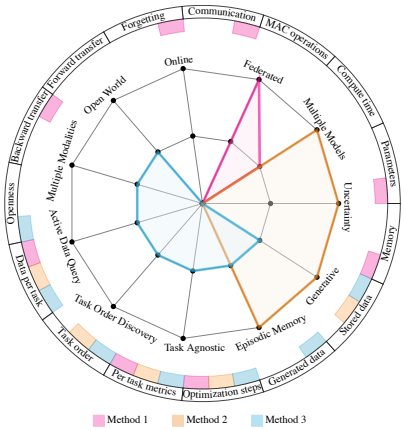
The resulting file `compass_filled.tex` can then be included into any LaTeX document, e.g.:

```
\begin{figure}
    \input{compass_filled.tex}
    \caption{CLEVA-Compass for methods Foo, Bar, and Baz.}
\end{figure}
```

With the above example, this results in the following visualization:



A JSON file with multiple entries (`examples/compass_data_3.json`) produces the following compass:



If you want to directly modify a filled template without using the Python script described below, we also provide `cleva_filled.tex` as an example with three entries, which corresponds to the results of `python create_compass.py --data examples/compass_data_3.json` .