mesmerize-core

Using caiman more efficiently

Efficient workflows with CaImAn

- Efficient use of algorithms
  - Hyperparameter search
- Visualization
  - Lazy loading & lazy computation
- Data organization
- Mesmerize provides a high level API to perform these tasks

mesmerize-core

Caitlin Lewis

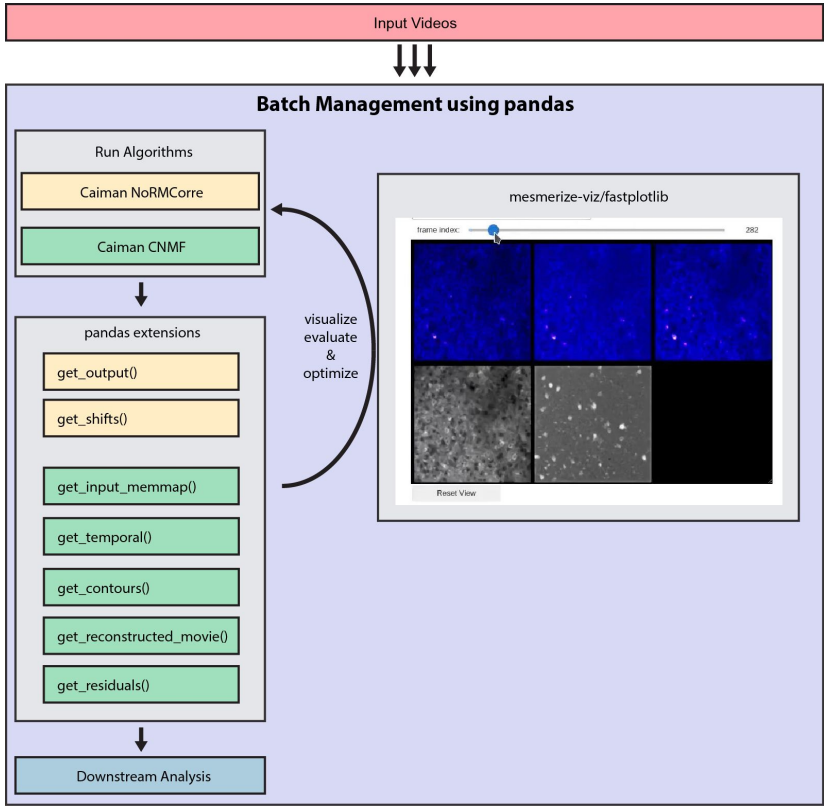
Arjun Putcha

- History
  - December 2021, got a small grant with Andrea Giovannucci
  - Mesmerize re-write using new technologies and tools
  - Developers: Arjun & Caitlin
- Repos
  - **https://github.com/nel-lab/mesmerize-core - API is now mostly stable!**
    https://github.com/kushalkolar/mesmerize-viz - WIP, do not use yet
- CI pipelines to support long-term maintenance

What parameters should I use?

Batch Management of computationally intensive tasks

- Hyperparameter optimization
  - Motion Correction
  - CNMF
  - OnACID
  - Gridsearch



What is a batch?

A DataFrame of batch items (rows) with the following columns

- input movie: raw or motion-corrected movie
- algorithm name: “mcorr” | “cnmf” | “cnmfe”
- parameters
- item name: user-defined name for your convenience

Example

index	uid	item_name	input_movie	algo	params
0	a	movie_1	exp_data/movie_1.tiff	mcorr	{“max_shifts”: (24, 24)...
1	b	movie_1	exp_data/movie_1.tiff	mcorr	{“max_shifts”: (36, 36)...
2	c	movie_2	exp_data/movie_1.tiff	mcorr	{“max_shifts”: (24, 24)...
3	d	movie_1	`uid b`	cnmf	{“gSig”: (4, 4)...
4	e	movie_2	`uid c`	cnmf	{“gSig”: (4, 4)...

Fetching outputs - mcorr

Get mcorr output of an item at index `ix`

```
row = df.iloc[ix]
raw_movie = row.caiman.get_input_movie()
mcorr_movie = row.mcorr.get_output()
```

Visualize using fastplotlib or your favorite visualization tool

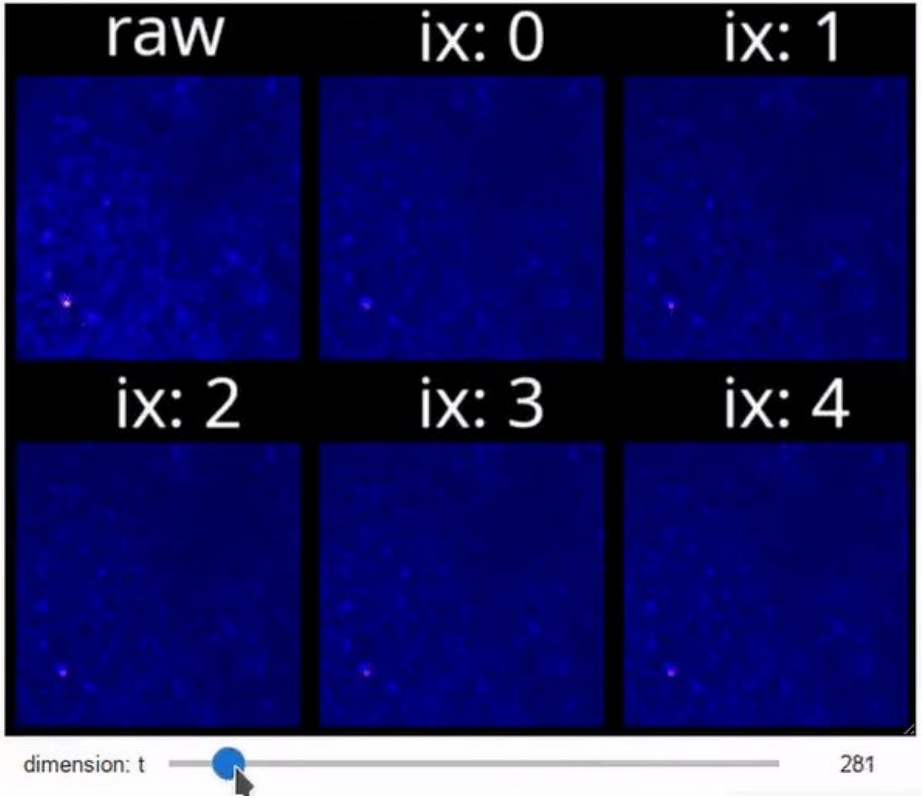
With fastplotlib (will be shown in demo)
- use `window_funcs`
- use `frame_apply`

Other extension functions that you can use:

```
row.caiman.get_projection("mean")
row.caiman.get_corr_image()
```

https://mesmerize-core.readthedocs.io/en/latest/api/mcorr.html

https://mesmerize-core.readthedocs.io/en/latest/api/common.html#mesmerize_core.CaimanSeriesExtensions.get_corr_image



How is the data saved?

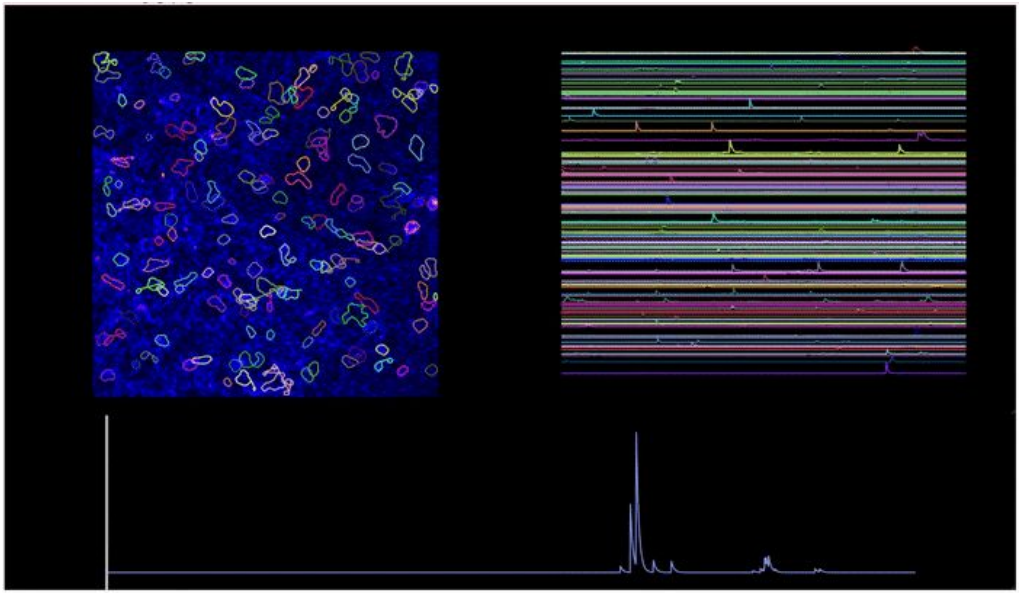
- **Exactly the same way that caiman saves them!**
  - Because it just uses the caiman API.
    - (use the caiman API, there's ***a lot*** of stuff in there :D)
- For motion correction this is the F-order memmap
- mesmerize-core organization
  - dataframe
    - directory *per row*
  - API to fetch outputs with lazy-loading and lazy-compute

Fetching outputs - cnmf

Get cnmf outputs of an item at index `ix`

```
row = df.iloc[ix]
temporal = row.cnmf.get_temporal("good")
contours = row.mcorr.get_contours("good")
```

https://mesmerize-core.readthedocs.io/en/latest/api/cnmf.html



How is the data saved?

- **Exactly the same way that caiman saves them!**
  - Because it just uses the caiman API.
    - (use the caiman API, there's ***a lot*** of stuff in there :D)

- For CNMF this is the cnmf-formated hdf5 file
  - Because it uses the caiman API to save them!

Lazy arrays

- numpy-like interface
  - slicing
  - properties
    - shape
    - ndim
    - …
- Lazy computing
  - reconstructed movie
    - $A \otimes C$
  - reconstructed background
    - $b \otimes f$
  - residuals
    - $Y - A \otimes C - b \otimes f$
- LazyTiff (not relevant to cnmf)

- LazyArrays will work with data that was NOT processed within mesmerize.

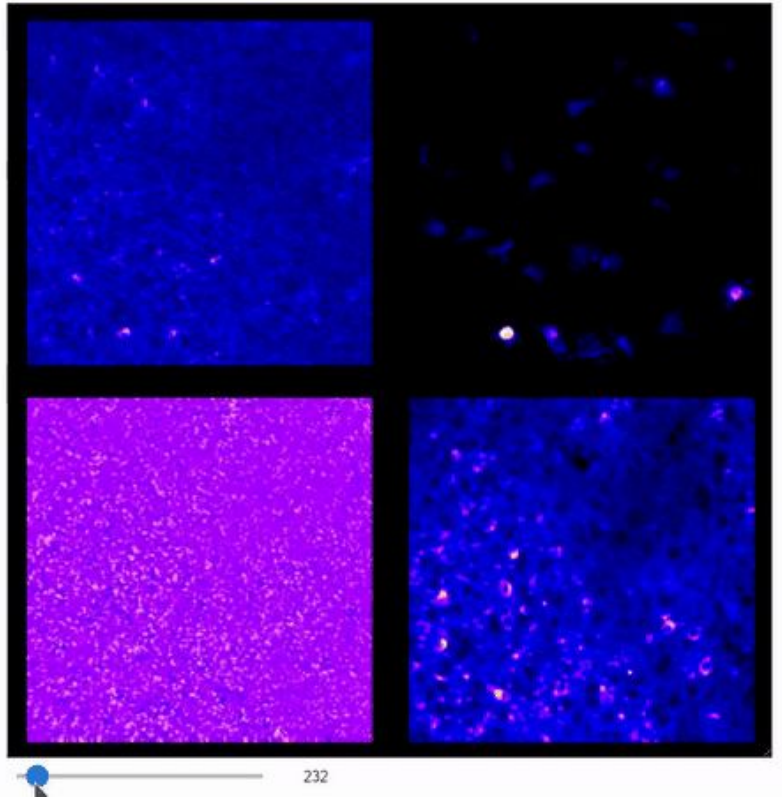
Fetching outputs

Get rcm, rcb, residuals of an item at index `ix`

```
row = df.iloc[ix]

mcorr = row.caiman.get_input_movie()

# all components; can also use “good”, “bad” etc.
rcm = row.cnmf.get_rcm()
rcb = row.cnmf.get_rcb()
res = row.cnmf.get_residuals()
```



Downstream analysis

You can continue to use the batch dataframe for downstream analysis

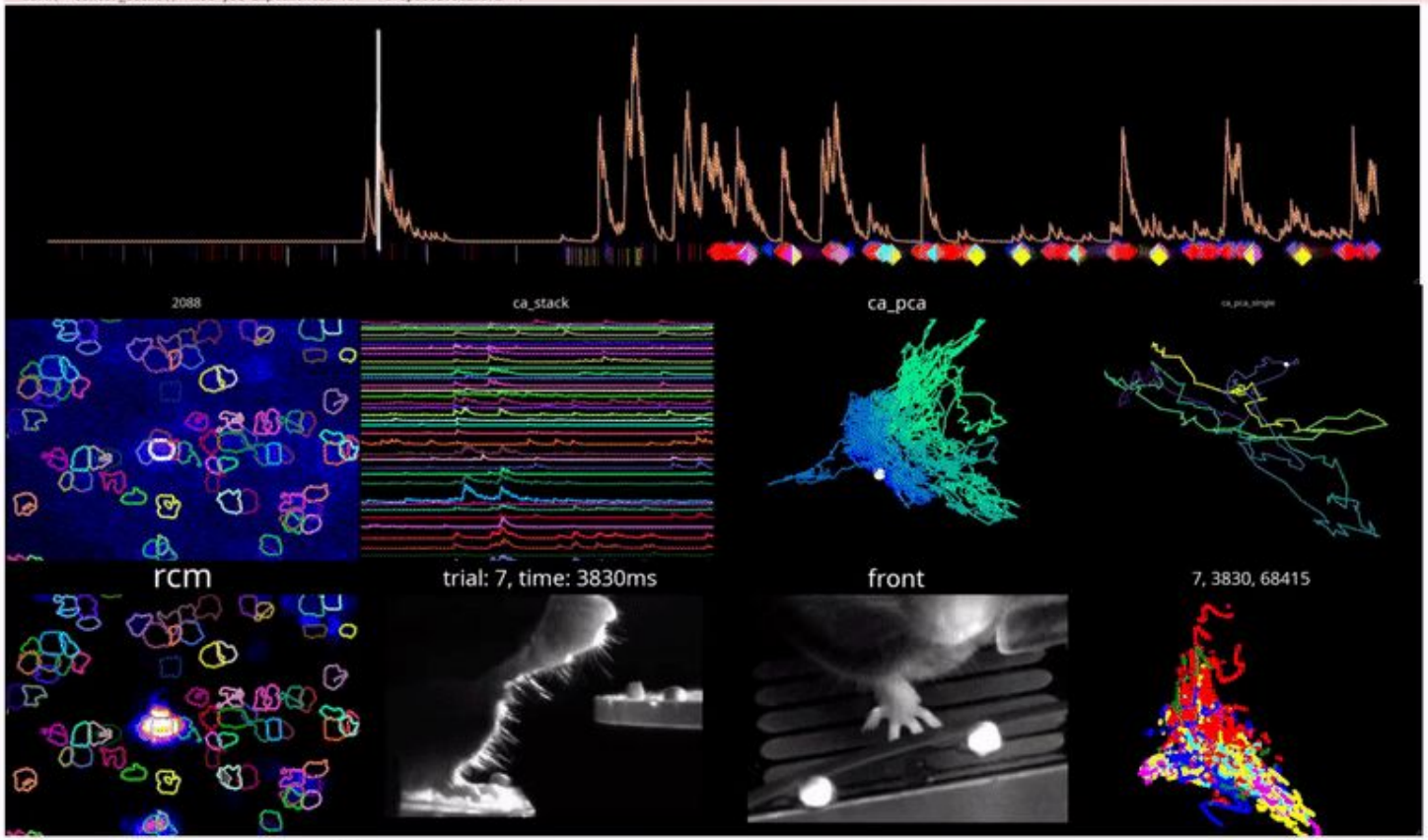
- Remove batch items that have “bad” parameters
- Keep only 1 mcorr and 1 cnmf item per input movie

iloc	uid	item_name	input_movie	algo	params	stereotac_coors	region	genotype
0	b	movie_1	exp_data/movie_1.tiff	mcorr	{“max_shifts”: (36, 36)...	{“ml”: 1.2, …	M1	wt
1	c	movie_2	exp_data/movie_1.tiff	mcorr	{“max_shifts”: (24, 24)...	{“ml”: 1.2, …	M1	mutant
2	d	movie_1	`uid b`	cnmf	{“gSig”: (4, 4)...	…	…	wt
3	e	movie_2	`uid c`	cnmf	{“gSig”: (4, 4)...	…	…	mutant
…	…	…	…	…	…			…

```
time_slider.observe(slider_changed, "value")
plot_temporal.renderer.add_event_handler(resize_slider, "resize")

VBox([plot_temporal.show(), gp.show(), time_slider])
```

```
/home/kushalk/repos/fastplotlib/fastplotlib/layouts/_base.py:205: UserWarning: `center_scene()` not yet implemented for `PerspectiveCamera`
  warn(f"`center_scene()` not yet implemented for `PerspectiveCamera`")
```




```
plot_temporal["temporal"][:].present = False
gp["ca_movie"]["img"].link("click", target=gp["ca_movie"]["contours"], feature="colors", new_data="w", callback_function=euclidean)
gp["rcm"]["img"].link("click", target=gp["rcm"]["contours"], feature="colors", new_data="w", callback_function=euclidean)
```

Limitations of the original Mesmerize desktop application

- **The Mesmerize desktop application is now legacy software**
  - Limited support is provided, no longer in development, migrate to mesmerize-core
  - This was the old repo: https://github.com/kushalkolar/MESmerize
- pyqtgraph and OpenGL are getting very old
  - Slows down beyond > ~300 neurons
- Difficult to install, expand, and maintain large Qt applications longterm
- Inefficient visualizations
  - Memmaps not used
  - Datasets larger than RAM
- Limited functionality
  - cannot interactively explore CNMF eval params
- Newer tools exist for more elegant data-organization systems
  - pandas extensions
- Other
  - Jupyter notebooks are better for downstream analysis than flowcharts

tl:dr: Stop using the old mesmerize!

(... and matlab)

Caveats - file safety

- Do not lose your batch pickle file!
- Not guaranteed to be transferable between computers, we might move to hdf5 for dataframe files in v0.2
  - Will maintain backwards compatibility by loading v0.1 pickle files and saving them in a new hdf5 format
- Do not run multiple batch items simultaneously!
  - This would probably require us to interface the dataframe with a real database

Getting help

- Please use GitHub for issues/bugs, and use the form
  - The more details you can provide the faster we can help you!
- Use gitter for smaller questions.
  - Gitter isn’t easily searchable so we strongly prefer GitHub for issues!

Things we can implement TODAY! :D

- OnACID
- Ain - spatial masks for initialization