# IANNA « On-the-fly »

Grégory SAINTON

Observatoire de Paris | PSL

29-30 octobre 2025

IANNA Workshop 2025

# IANNA « On-the-fly »

Grégory SAINTON

IN PROGRESS

Observatoire de Paris | PSL

29-30 octobre 2025

IANNA Workshop 2025

# Why CIANNA "on-the-fly"? – The challenge

- Astronomical datasets are becoming massive (SKA, LSST, Euclid…).
- Manual or offline source detection workflows are no longer scalable.
- Visualization tools (Aladin, Carta, DS9, YaFITS…) are still disconnected from inference* models.
- Scientists need interactive source identification directly within their data viewer, without exporting images or writing scripts.

*Inférence : when we use the model after training

CIANNA DAYS

# Why CIANNA "on-the-fly"? – Our answer

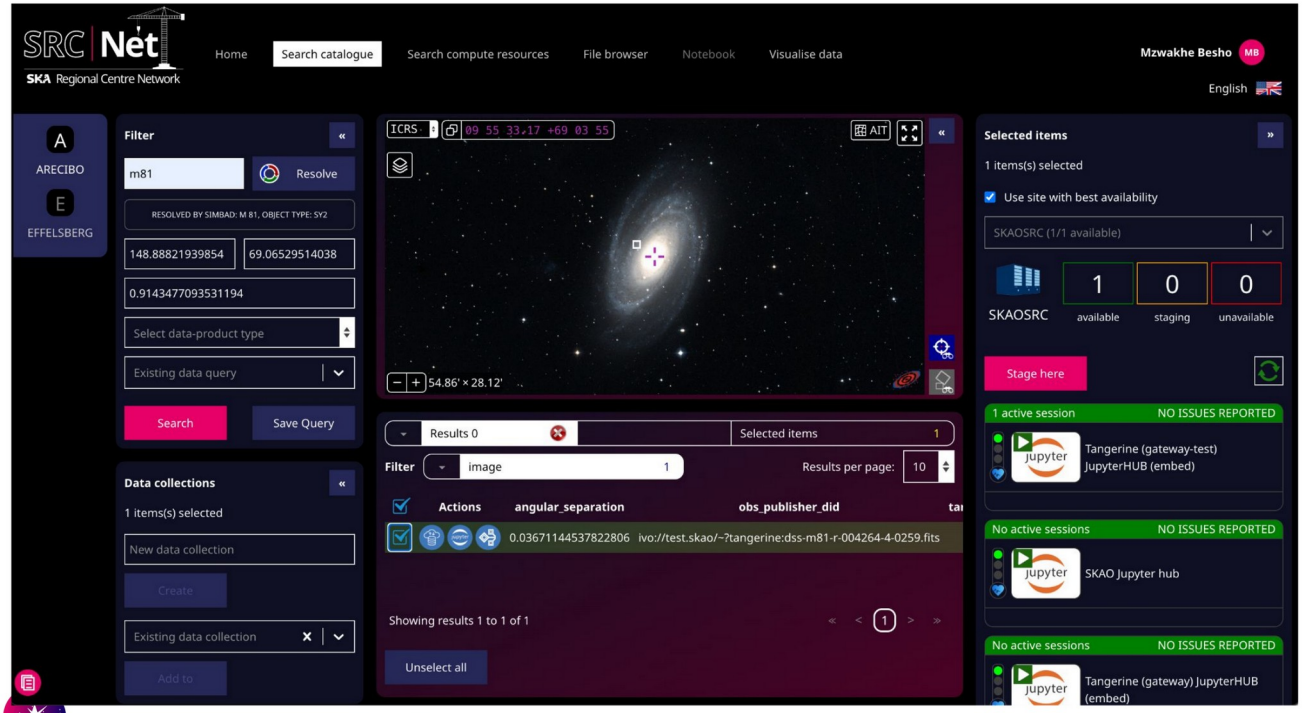
CIANNA OTF has the ambition to:

- Take advantage of CIANNA framework which is already validated on large datasets (Cornu+ 2024, Cornu+ 2025);

- Provides a standardized prediction service pluggable on any dataviz tool.
  - Respect the IVOA UWS standard sharing protocol;
  - Enables on-the-fly inference: the user clicks in his favorite dataviz tool → job sent to server → detection catalog returned quickly.

# Why CIANNA "on-the-fly"? – Our answer

## For Example... perform data product search




From R. Bolton presentation @ Gurliz, 2025

- Data are somewhere in the SRCNet
- Need to connect on a portal
- Dedicated tools will be selected (on going)
- **CIANNA-OFT has its place in this portal**

SKAO

CIANNA DAYS

# On the menu…

- Data Exchange protocol – the UWS layer
- CIANNA OTF pipeline – use case
- Next milestones

# Data exchange protocol

From dataviz tool to CIANNA OTF Server

# Using CIANNA with dataviz tools

DATAVIZ

SAO DS9

YaFITS


CARTA

...

CIANNA . DAYS

# Using CIANNA with dataviz tools

DATAVIZ

SAO DS9

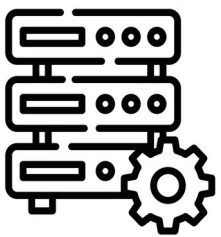
YaFITS

ALADIN

CARTA

...

PREDICTION



CIANNA

# Using CIANNA with dataviz tools

DATAVIZ

SAO DS9

YaFITS

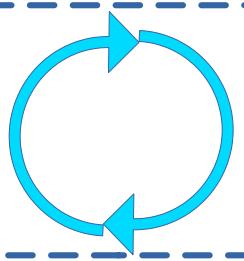
ALADIN

CARTA

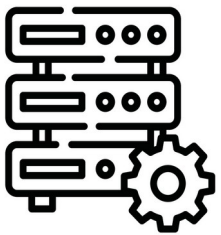
...

TRANSFERT

Standard sharing protocol ?



PREDICTION



CIANNA

CIANNA .DAYS

# Using CIANNA with dataviz tools

DATAVIZ

SAO DS9

YaFITS

ALADIN

CARTA

...

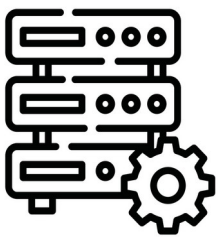
TRANSFERT

IVOA

*I* nternational
*V* irtual
*O* bservatory
*A* lliance

UWS

PREDICTION



CIANNA

CIANNA .DAYS

# UWS: Universal Worker Service


CIANNA .DAYS

▶️ Standard UWS IVOA protocol  manage asynchronous jobs over HTTP.

- Submit jobs ;
- Monitor their progress (Pending, Executing, Completed, Error, ...)
- Retrieve results later.

▶️ Key principles of UWS

- **RESTful interface** – every job is a web resource (/jobs/{id})
- Each job has a **lifecycle**: creation → execution → completion
- The client interacts via standard HTTP ;
    - **POST** (create), **GET** (check status/results), **DELETE** (remove) and **PUT** (retrieve results)
    - Exchange in **XML**
- Asynchronous jobs → your soft is not stuck

# IVOA – International Virtual Obs Alliance

- "The Virtual Observatory is the necessary "middle layer" framework connecting the Resource Layer to the User Layer in a seamless and transparent manner.";
- UWS is a standard among a lot in the IVOA architecture;
- Most of the architecture is used from Datacenter to dataviz tools.
- **CIANNA OTF is a "plugin" on these existing dataviz tools ;**
- **So far, no need to go beyond the UWS standard.**

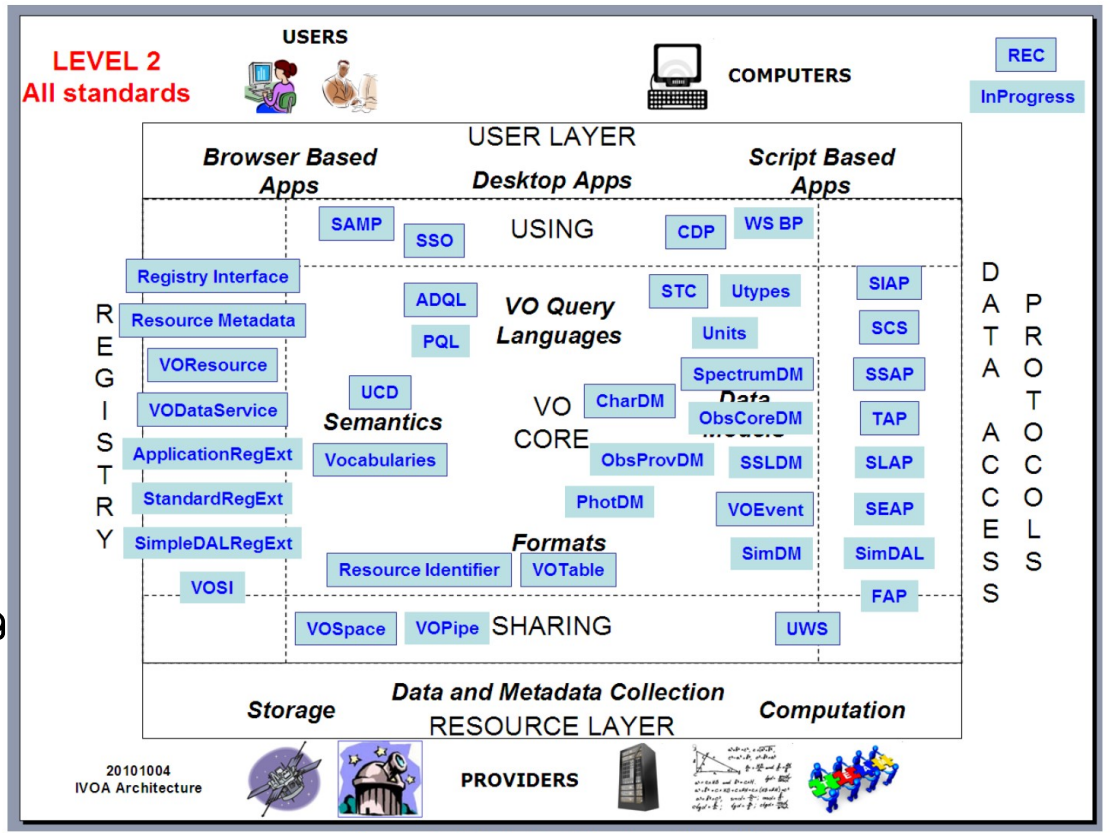



**Figure 3: IVOA Architecture Level 2**

(From IVOA Note 2010-11-23)



CIANNA .DAYS

# UWS: Universal Worker Service

```xml
<?xml version="1.0" encoding="UTF-8"?>
<uws:job xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0"
         xmlns:xlink="http://www.w3.org/1999/xlink"
         version="1.1">

  <!-- ========== Core metadata required by UWS ========== -->
  <uws:jobId>CIANNA_OTF_Job</uws:jobId>
  <uws:runId>Altamira_Update</uws:runId>
  <uws:ownerId>Altamira_SERVER</uws:ownerId>
  <uws:phase>PENDING</uws:phase>
  <uws:quote xsi:nil="true"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
  <uws:creationTime>2025-02-04T14:21:00Z</uws:creationTime>
  <uws:startTime xsi:nil="true"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
  <uws:endTime xsi:nil="true"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
  <uws:executionDuration>3600</uws:executionDuration>
  <uws:destruction>2025-02-05T14:21:00Z</uws:destruction>

  <!-- ========== Parameters (CIANNA model configuration) ========== -->
  <uws:parameters>
    <uws:parameter id="CIANNAVersion">1.0.0</uws:parameter>
    <uws:parameter id="ModelName">net0_s1800.dat</uws:parameter>
    <uws:parameter id="ReleaseDate">2025-02-01</uws:parameter>
    <uws:parameter id="BaseMemoryFootprint">0</uws:parameter>
    <uws:parameter id="PerImageMemoryFootprint">0</uws:parameter>
    <uws:parameter id="OriginalInputDim">256x256x1x1</uws:parameter>
    <uws:parameter id="MinInputDim">128x128x1x1</uws:parameter>
    <uws:parameter id="MaxInputDim">512x512x1x1</uws:parameter>
    <uws:parameter id="YOLOGridElemDim">16x16x1x1</uws:parameter>
    <uws:parameter id="YOLOBoxCount">8</uws:parameter>
    <uws:parameter id="YOLOParamCount">5</uws:parameter>
    <uws:parameter id="YOLOGridCount">16</uws:parameter>
    <uws:parameter id="DataNormalization">Global</uws:parameter>
    <uws:parameter id="DataNormalizationType">tanh</uws:parameter>
    <uws:parameter id="DataQuantization">float32</uws:parameter>
    <uws:parameter id="ReceptiveField">100</uws:parameter>
    <uws:parameter id="TrainingQuantization">FP32C_FP32A</uws:parameter>
    <uws:parameter id="InferenceMode">Grid</uws:parameter>
    <uws:parameter id="InferenceQuantization">FP16C_FP32A</uws:parameter>
    <uws:parameter id="InferencePatchShift">240</uws:parameter>
    <uws:parameter id="InferenceOrigOffset">128</uws:parameter>

    <uws:parameter id="ModelURL">https://zenodo.org/records/xxxxx</uws:parameter>
    <uws:parameter id="TrainingDatasetName">SKA SDC1 Data</uws:parameter>
    <uws:parameter id="TrainingDatasetDescription">Synthetic data for SKA Data Challenge 1</uws:parameter>
    <uws:parameter id="TrainingDatasetLocation">https://example.org/SDC1</uws:parameter>
    <uws:parameter id="CheckpointPath">./net_save/net0_s1800.dat</uws:parameter>
  </uws:parameters>

  <!-- ========== Results placeholder ========== -->
  <uws:results>
    <uws:result id="output"
                xlink:href="fwd_res/net0_rts_001.dat"
                mime-type="application/octet-stream"/>
  </uws:results>

  <!-- ========== Error summary (optional) ========== -->
  <uws:errorSummary type="transient" hasDetail="false">
    <uws:message>Model trained with simulated SKA data.</uws:message>
  </uws:errorSummary>

  <!-- ========== Optional free-form job information ========== -->
  <uws:jobInfo>
    <uws:note>Job created manually.</uws:note>
  </uws:jobInfo>
```

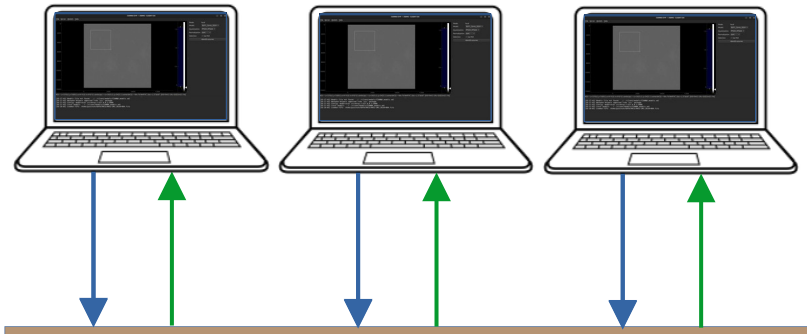
Example of UWS file (model parameters)


CIANNA DAYS

# CIANNA-OTF Pipeline

Under the hood...

# CIANNA OTF – Under the hood (0/14)



Language and framework

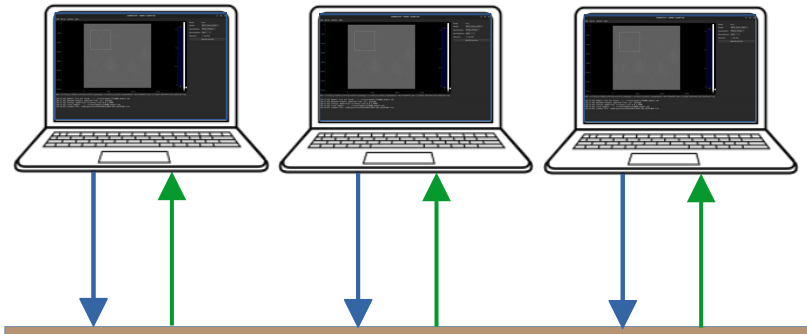
- Python 3 (client and server)
- Flask (server side) → for prototyping

Communication layer

- API REST - fully compatible with IVOA conventions ;
- Exchanges follow the UWS job lifecycle ;

CIANNA · DAYS

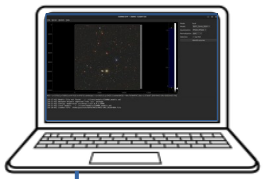
# CIANNA OTF – Under the hood (1/14)



- **Inference** : CIANNA OTF is used to apply existing models on user images or fields;
- **Trained models**: Existing trained models are stored on the server side.
- **Match image/model**: Using a Graphical Interface, user can select his/her image and the best existing model.

CIANNA .DAYS

# CIANNA OTF – Under the hood (2/14)




- Sending
  - my_image.fits
  - parameters.xml

} Preprocessing (normalisation)

*scheduler_loop*

1


2

**Pending**



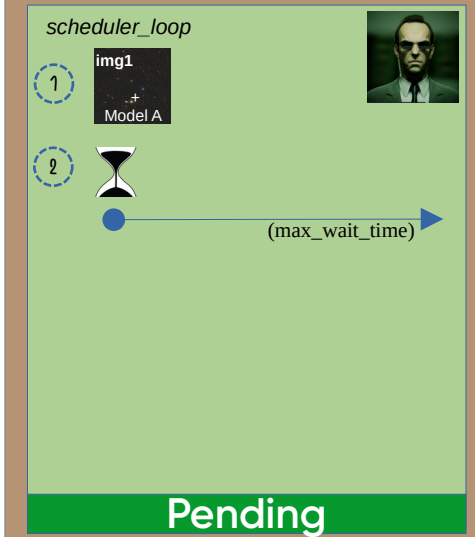
CIANNA .DAYS

# CIANNA OTF – Under the hood (3/14)

Sending
- my_image.fits
- parameters.xml

} Preprocessing (normalisation)

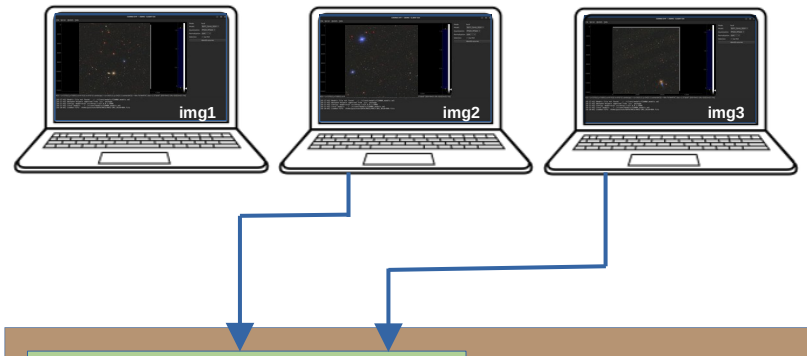
➔ Scheduler agent is waiting for more images during *max_wait_time*;*




*max_wait_time will depends on the size of the model and the time needed to load it in memory.

CIANNA · DAYS

# CIANNA OTF – Under the hood (4/14)




Sending
- my_image.fits
- parameters.xml

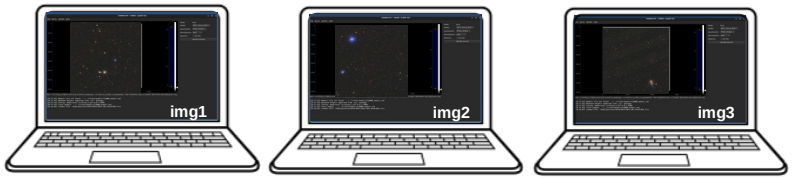
} Preprocessing (normalisation)

Scheduler agent is waiting for more images during *max_wait_time;*

→ Some images arrive from other clients;

CIANNA .DAYS

# CIANNA OTF – Under the hood (5/14)




Sending
- my_image.fits
- parameters.xml

} Preprocessing (normalisation)

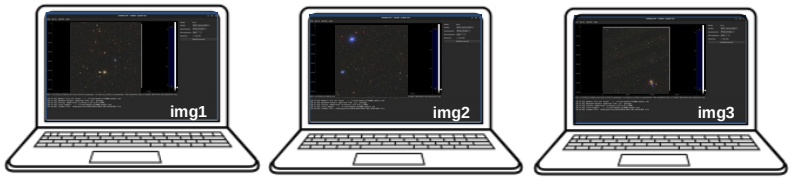
Scheduler agent is waiting for more images during *max_wait_time;*

Some images arrive from other clients;

→ Another agent is grouping jobs into batches with same models and same parameters (quantization);


CIANNA · DAYS

# CIANNA OTF – Under the hood (6/14)




- parameters.xml

Scheduler agent is waiting for more images during *max_wait_time;*

Some images arrive from other clients;

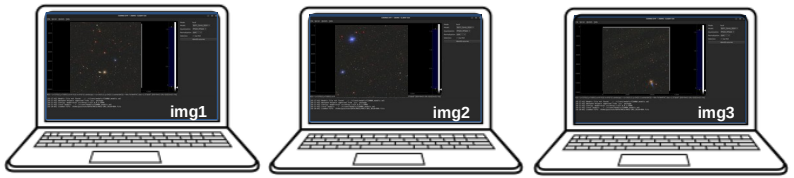
Another agent is grouping jobs into batches with same models and same parameters (quantization);

→ Model A is loaded in memory;




CIANNA .DAYS

# CIANNA OTF – Under the hood (7/14)

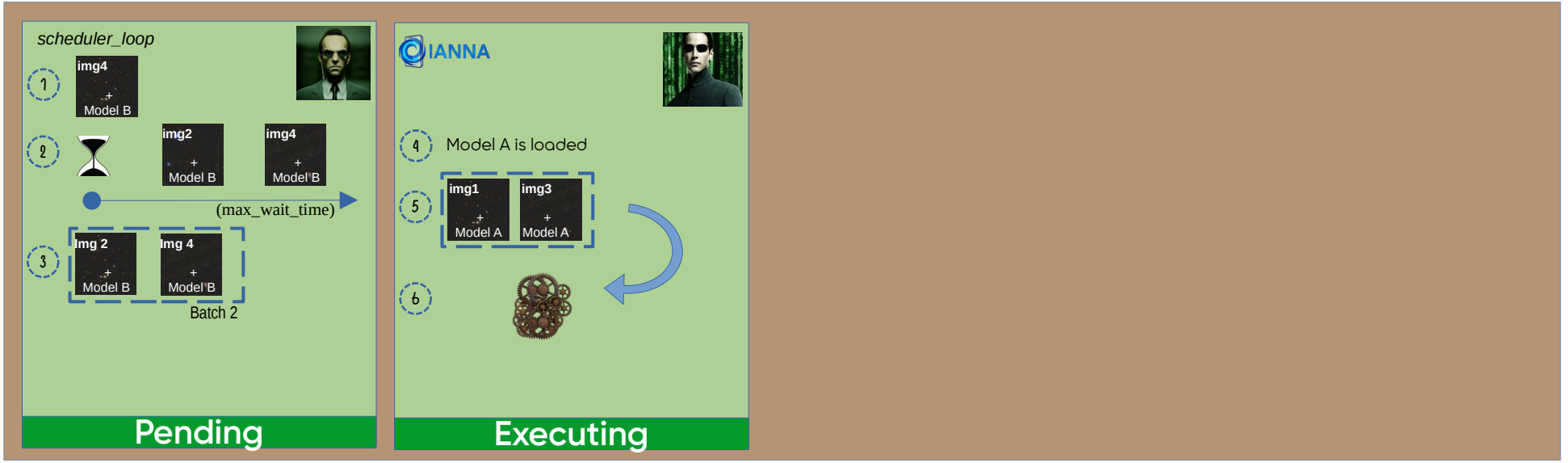



Some images arrive from other clients;

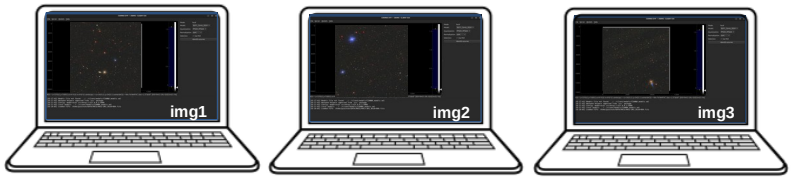
Another agent is grouping jobs into batches with same models and same parameters (quantization);

Model A is loaded in memory and images copied in "EXECUTING" directory ; Job status is now "EXECUTING";

- Images are stacked to be processed in the same time; Inference starts;



# CIANNA OTF – Under the hood (8/14)

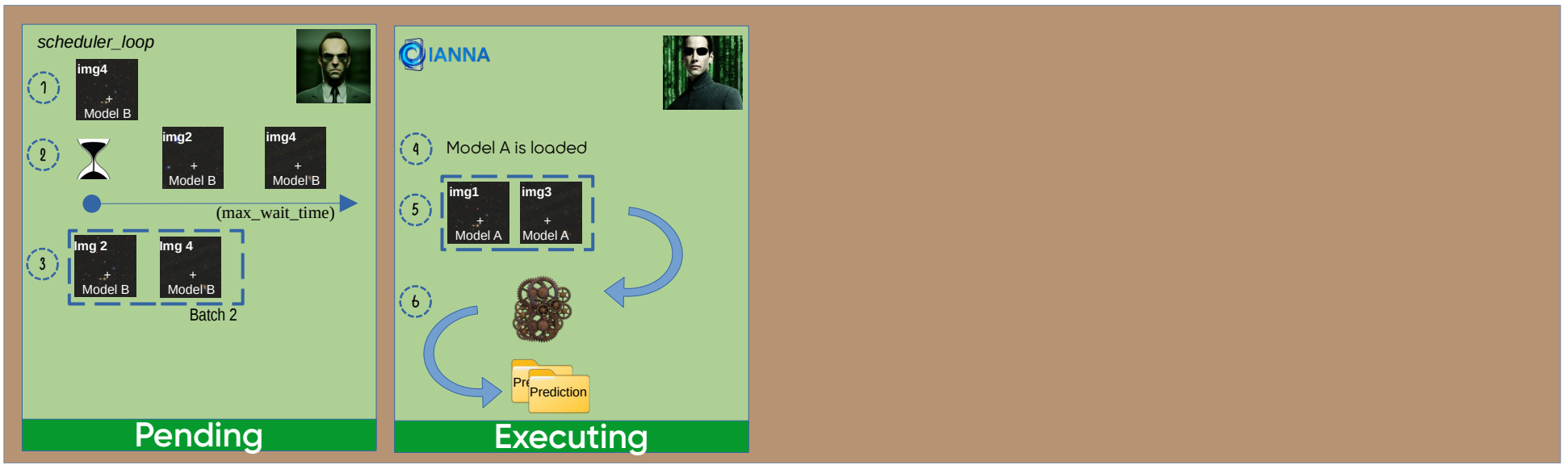



Another agent is grouping jobs into batches with same models and same parameters (quantization);

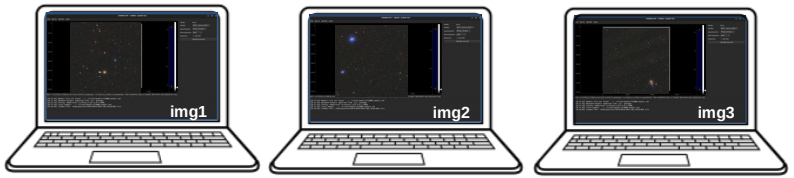
Model A is loaded in memory and images copied in "EXECUTING" directory ; Job status is now "EXECUTING";

Images are stacked to be processed in the same time; Inference starts;

- Raw prediction files are post-processed to create a catalog;



# CIANNA OTF – Under the hood (9/14)

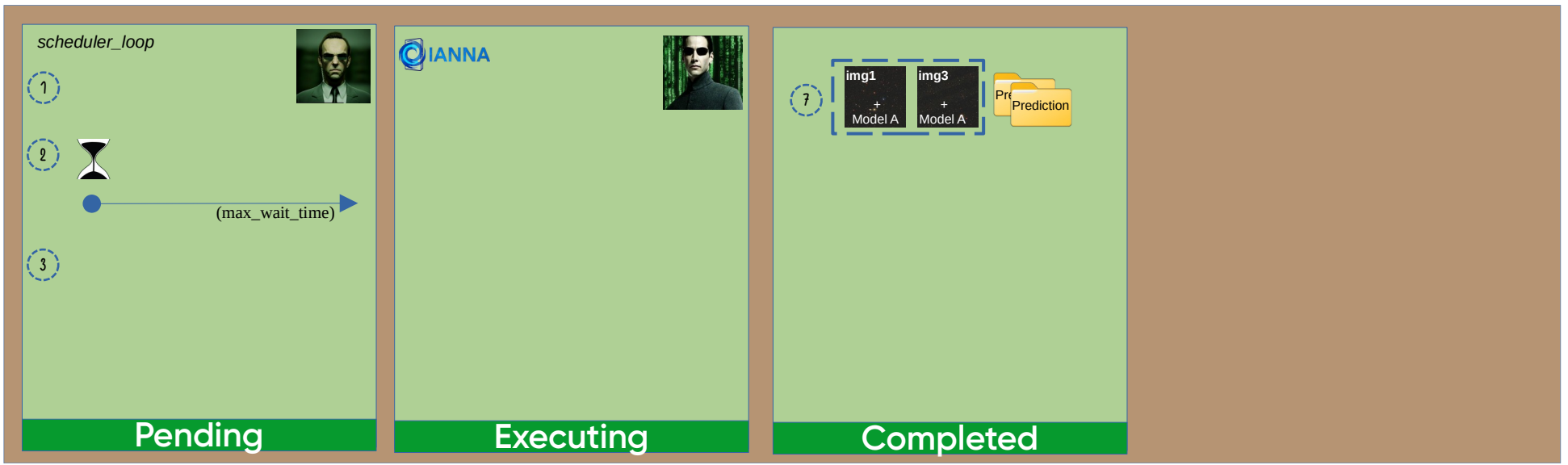



Model A is loaded in memory and images copied in "EXECUTING" directory ; Job status is now "EXECUTING";

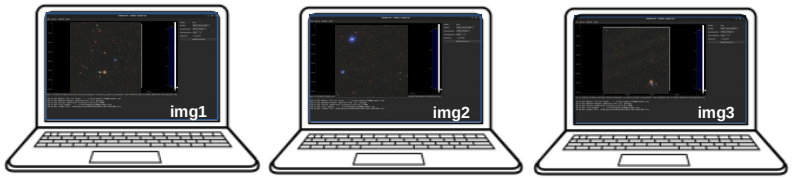
Images are stacked to be processed in the same time; Inference starts;

Raw prediction files are post-processed to create a catalog;

→ Catalogs and images are copied to a completed directory and the status of the job is updated to "COMPLETED";



# CIANNA OTF – Under the hood (10/14)

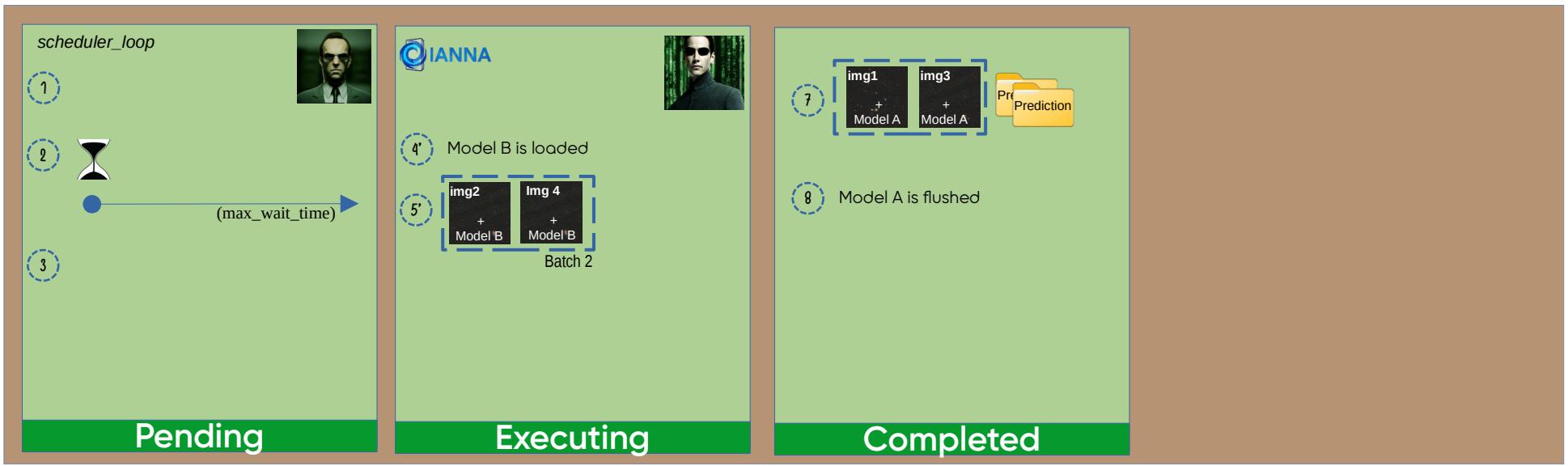



Images are stacked to be processed in the same time; Inference starts;

Raw prediction files are post-processed to create a catalog;

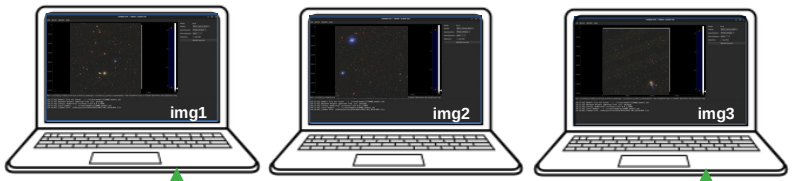
Catalogs and images are copied to a completed directory and the status of the job is updated to "COMPLETED";

→ Model A is flushed then Model B is loaded and batch 2 images copied in the "Executing" directory ; job status is "EXECUTING";




CIANNA · DAYS

# CIANNA OTF – Under the hood (11/14)

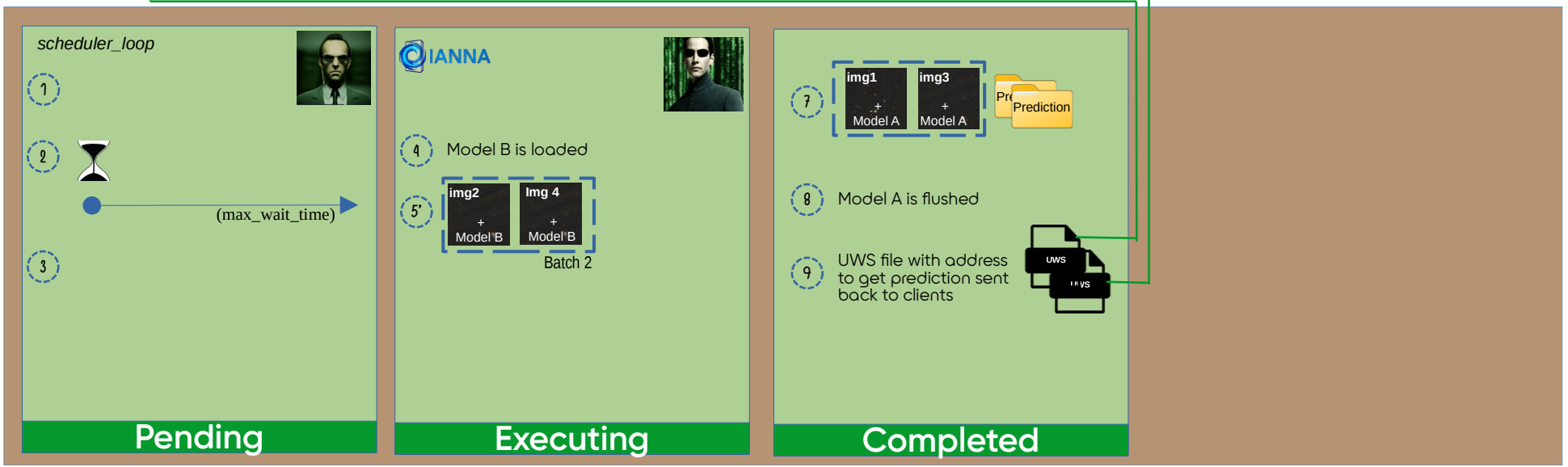



Raw prediction files are post-processed to create a catalog;

Catalogs and images are copied to a completed directory and the status of the job is updated to "COMPLETED";

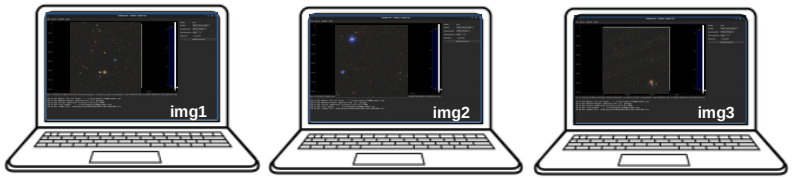
Model A is flushed then Model B is loaded and batch 2 images copied in the "Executing" directory ; job status is "EXECUTING";

→ Clients are notified by a UWS file that their job has ended and that the prediction file can be retrieved at a given address;




CIANNA DAYS

# CIANNA OTF – Under the hood (11/14)

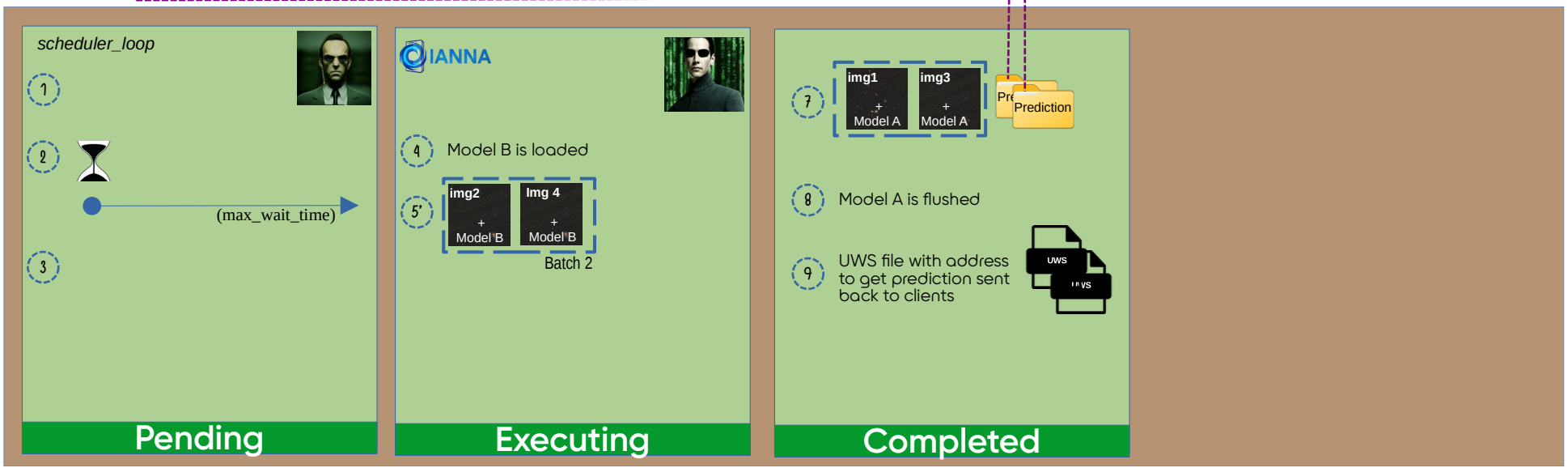



Catalogs and images are copied to a completed directory and the status of the job is updated to "COMPLETED";

Model A is flushed then Model B is loaded and batch 2 images copied in the "Executing" directory ; job status is "EXECUTING";

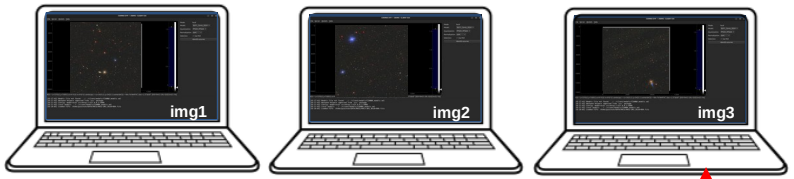
Clients are notified by a UWS file that their job has ended and that the prediction file can be retrieved at a given address;

- Catalogs are downloaded by clients ;




CIANNA DAYS

# CIANNA OTF – Under the hood (13/14)



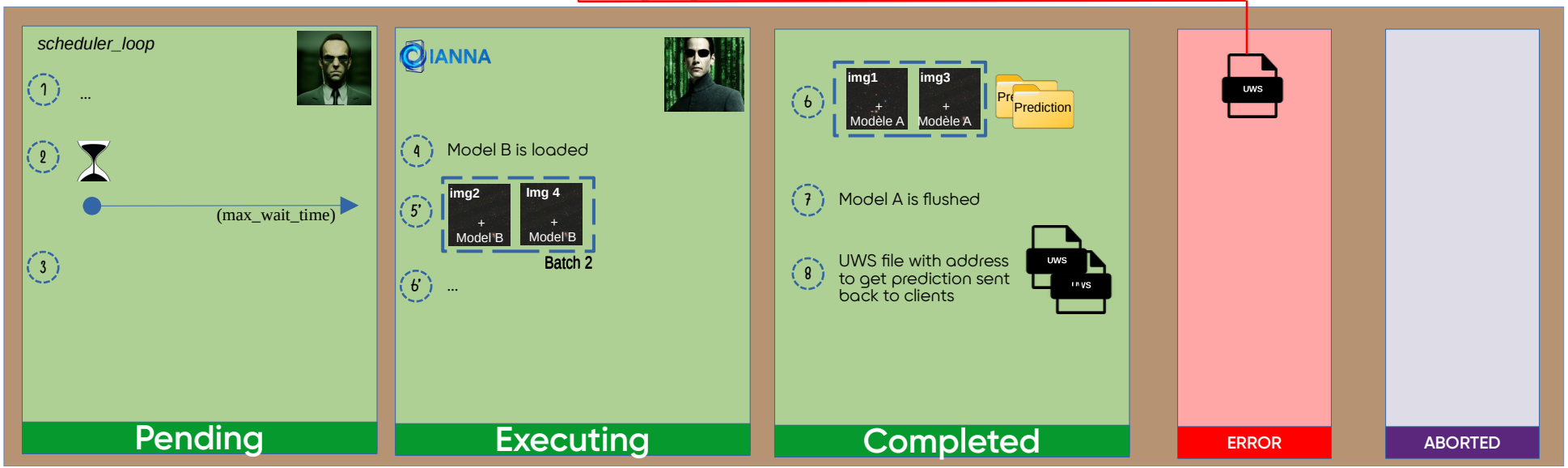

Model A is flushed then Model B is loaded and batch 2 images copied in the "Executing" directory ; job status is "EXECUTING";

Clients are notified by a UWS file that their job has ended and that the prediction file can be retrieved at a given address;

Catalogs are downloaded by clients;

❌ In case of error, the job take the "ERROR" status and the client is notified;

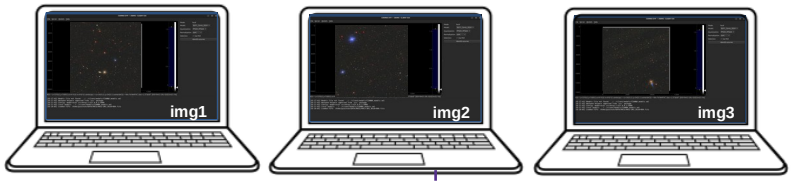
At any stage, if an error occurs, client is notified






CIANNA .DAYS

# CIANNA OTF – Under the hood (14/14)

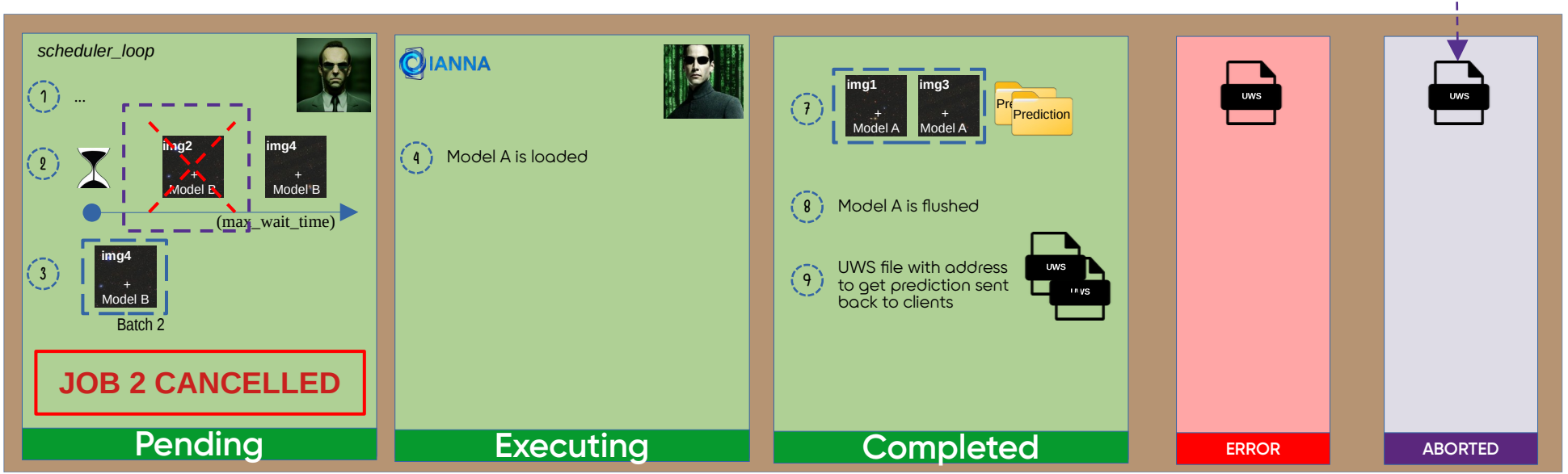



Clients are notified by a UWS file that their job has ended and that the prediction file can be retrieved at a given address;

Catalogs are downloaded by clients;

❌ In case of error, the job take the "ERROR" status and the client is notified;

🚮 Client can also request the abortion of his/her job *(not yet implemented)* with the status ABORTED;




CIANNA · DAYS

# Next milestones

Roadmap to be followed

# Next steps (1/3): what has been done so far ?

✅ Flask server architecture up and running (locally and on a GPU-based server);

✅ Multi-job scheduler (PENDING/EXECUTING/COMPLETED & ERROR) ;

✅ Exchanges compliant with the IVOA UWS 1.1 sharing protocol ;

✅ Run YOLO-CIANNA inference on a single GPU on multiple images ;

✅ Functional prototype client (“toy app”) [not presented here] ;

CIANNA .DAYS

# Next steps (2/3)

👷 ON GOING

- Finish tests on the catalog creation;
- Finish tests on ROI selection [feature developped but not tested];
- Free other some parameters in the toy app and use them fron the server side (EXPERT MODE)

  +

- Write a proper technical documentation (this presentation is a good draft) ;
- Write a user documentation ;

# Next steps (3/3)

🗓️ Short term

- Test performed only on a single 2D model (based on SDC1) → improve to 3D models;
- Add features to select NMS from client interface;
- Implement missing UWS status for better notification (QUEUED, ABORTED, SUSPENDED, UNKNOWN…)
- Improve monitoring and logging;
- Distribute a first version for test and debugging.

# Next steps (3/3)

CIANNA.DAYS

### 🗓️ Short term

- Test performed only on a single 2D model (based on SDC1) → improve to 3D models ;
- Add features to select NMS from client interface;
- Implement missing UWS status for better notification (QUEUED, ABORTED, SUSPENDED, UNKNOWN…)
- Improve monitoring and logging;
- Distribute a first version for test and debugging.

### 🗓️ Mid term

- Scalability:
  - Flask is just for prototyping, can't manage multithreading → modify HTTP front-end;
  - Manage multi-GPU server to distribute jobs.

- Direct integration into YaFITS (API & "Identify sources" menu); → Collab' w/ YaFITS Team

# Next steps (3/3)

CIANNA.DAYS

### 🗓️ Short term

- Test performed only on a single 2D model (based on SDC1) → improve to 3D models ;
- Add features to select NMS from client interface;
- Implement missing UWS status for better notification (QUEUED, ABORTED, SUSPENDED, UNKNOWN…) ;
- Improve monitoring and logging;
- Distribute a first version for test and debugging.

### 🗓️ Mid term

- Scalability:
  - Flask is just for prototyping, can't manage multithreading → modify HTTP front-end;
  - Manage multi-GPU server to distribute jobs.

- Direct integration into YaFITS (API & "Identify sources" menu); → Collab' w/ YaFITS Team

### 🗓️ Long term

- Integration in the SRCNet (at least in the french Node).

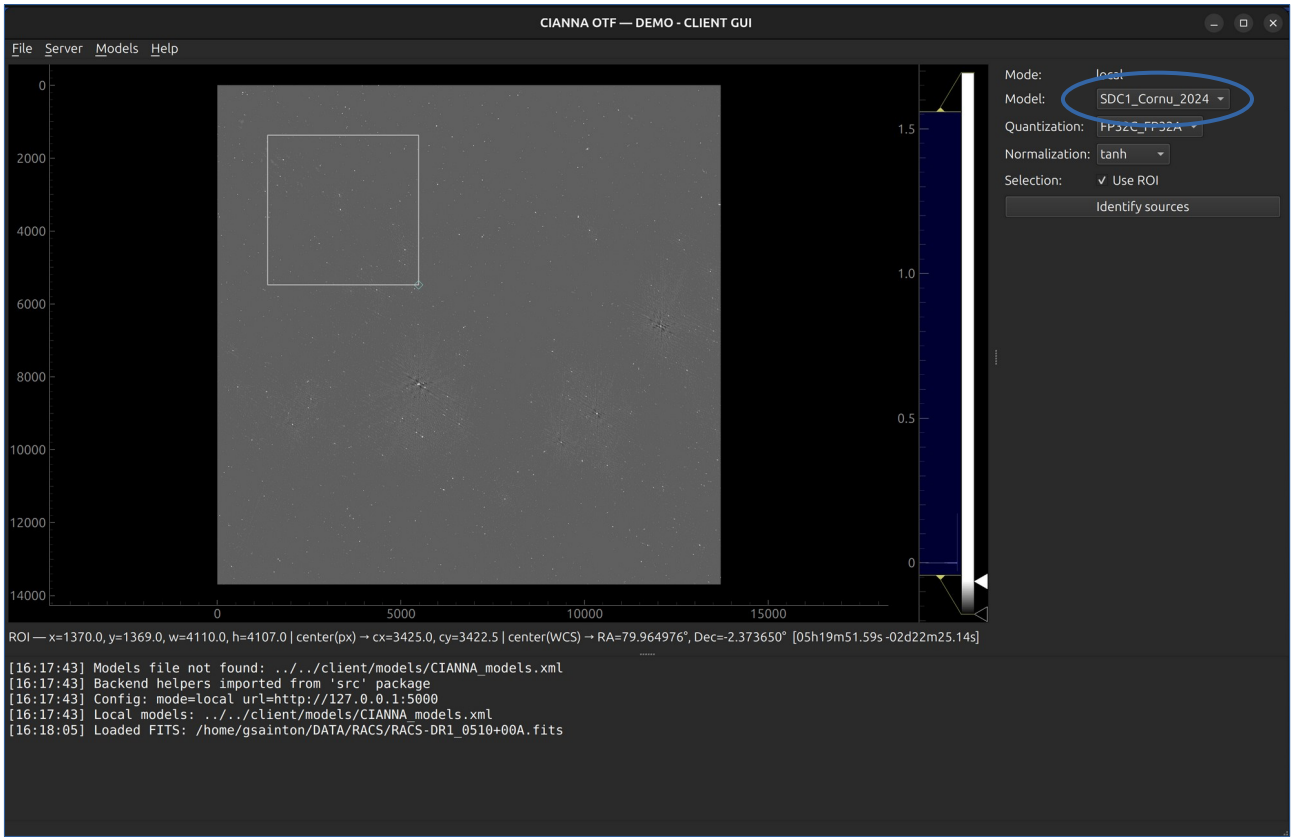
# Thank you.

# Back-up slides.

# About the "Toy app"

Quick & dirty tool to test CIANNA-OTF

# Toy App for CIANNA-OTF

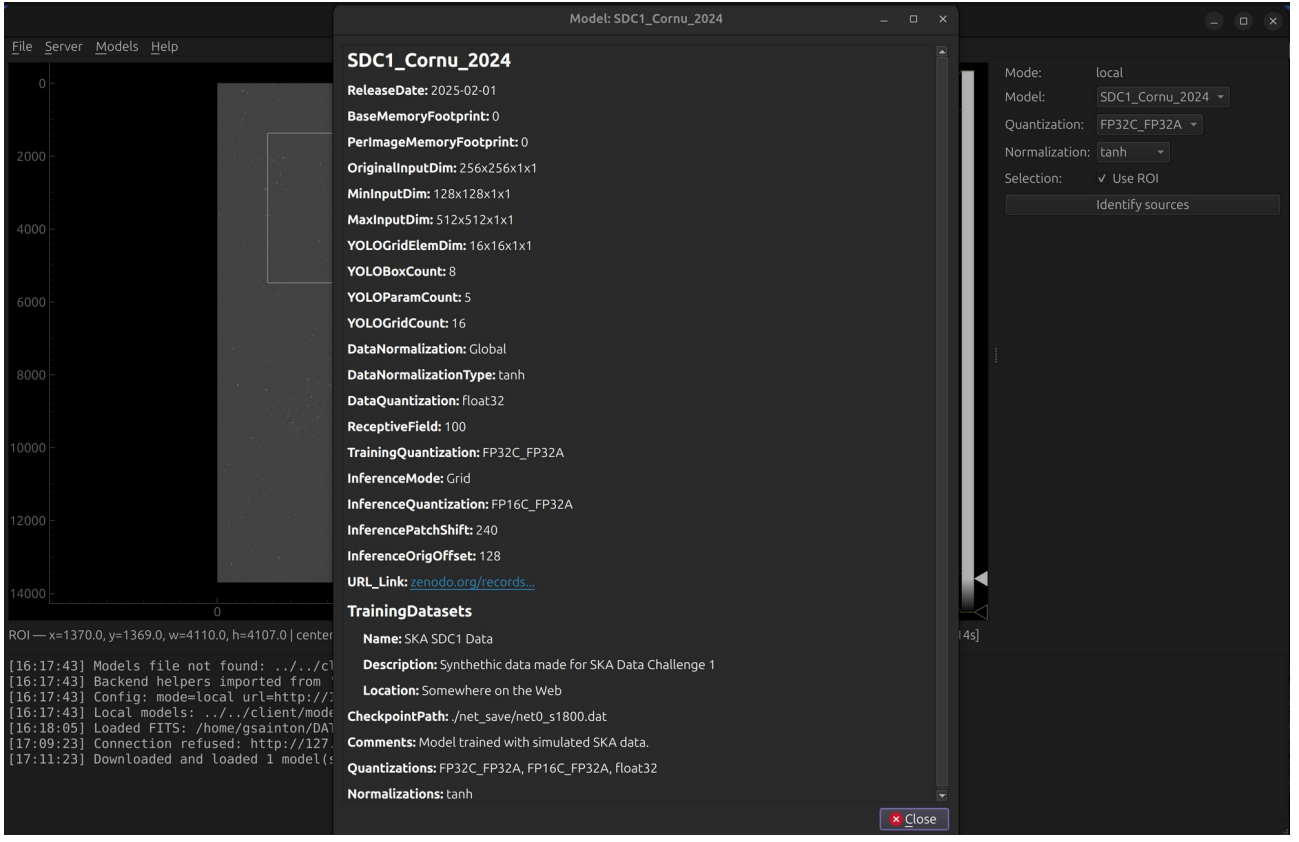



> Dev in PyQt 6

When the app starts up: query the server to retrieve the list of available models.

CIANNA · DAYS

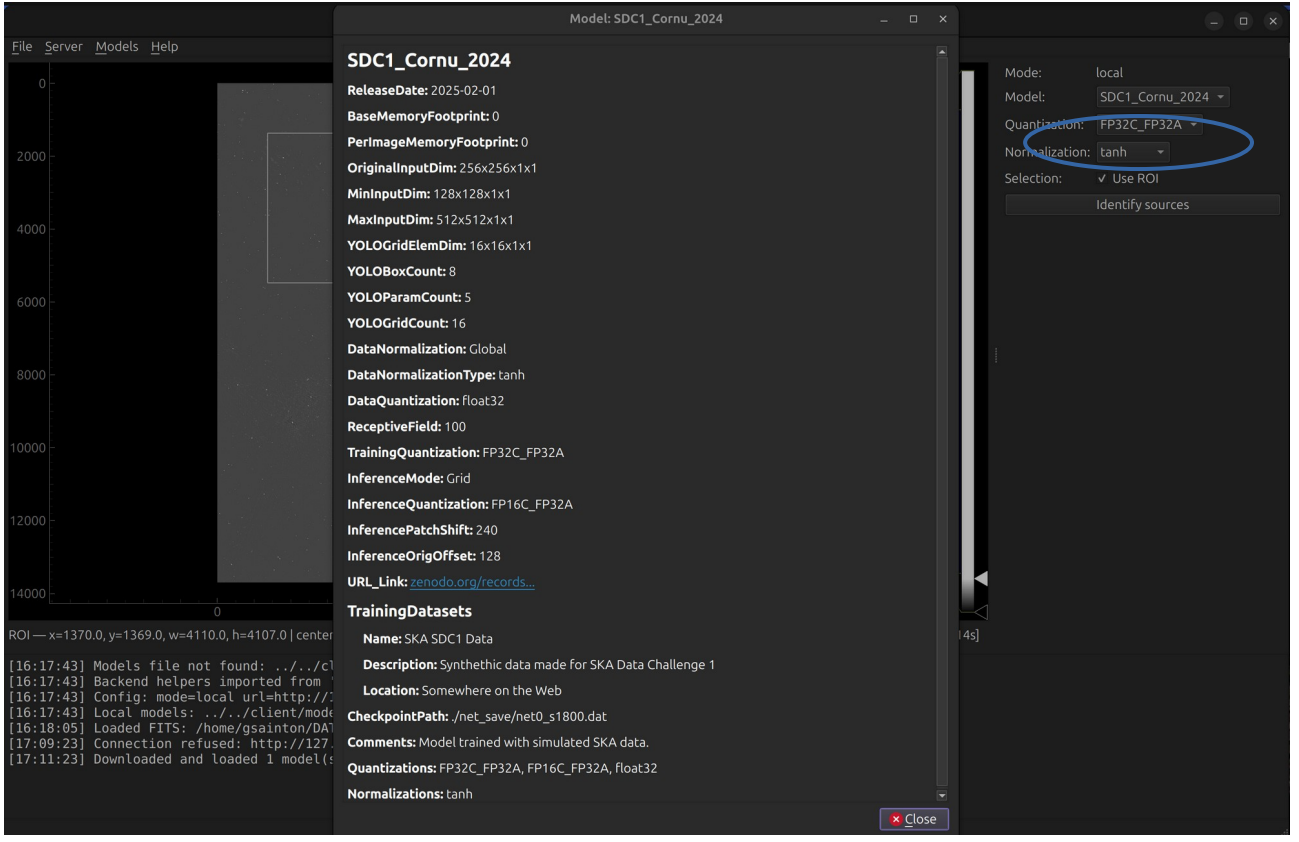
# Toy App for CIANNA-OTF




- When the app starts up: query the server to retrieve the list of available models;
- Each model contains information specific to its training;

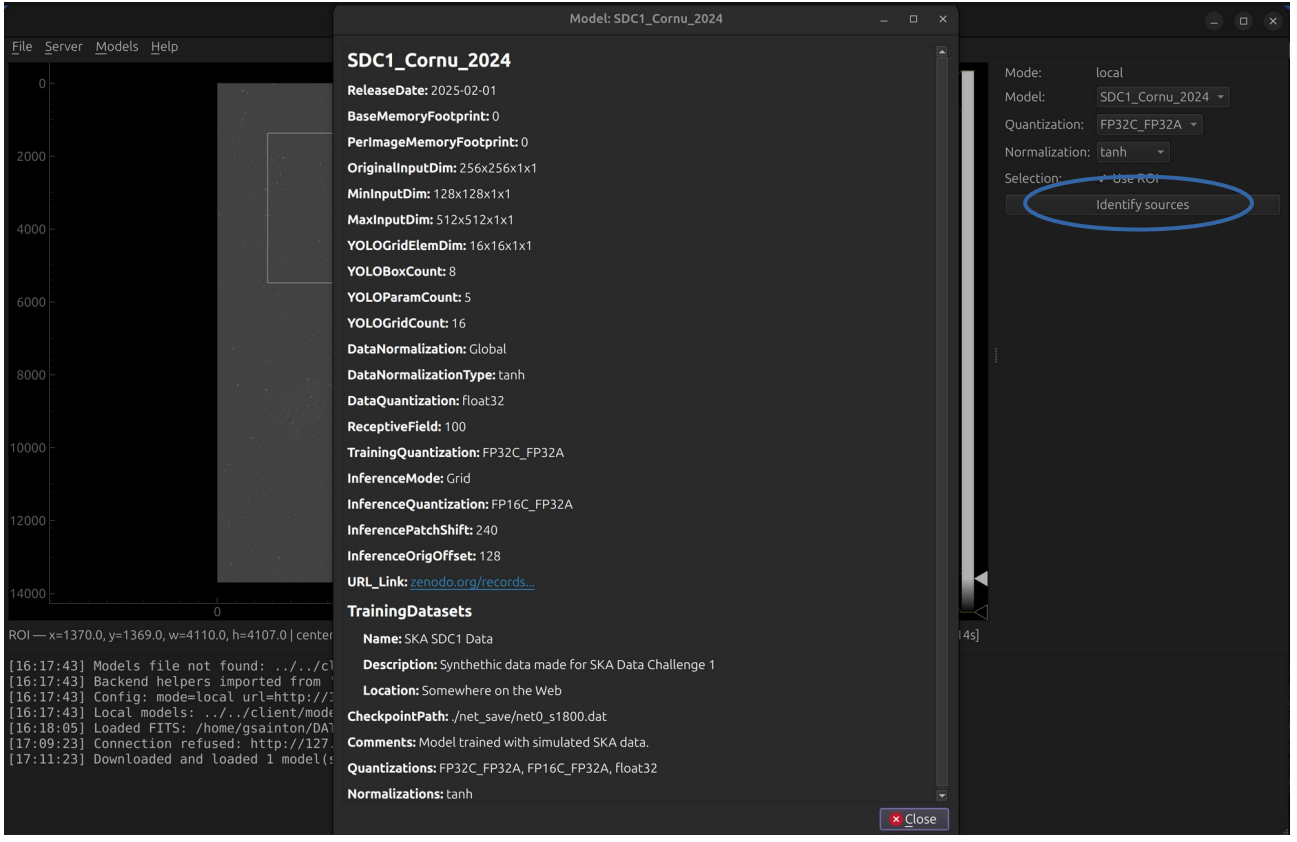
CIANNA DAYS

# Toy App for CIANNA-OTF




- When the app starts up: query the server to retrieve the list of available models;
- Each model contains information specific to its training;
- Certain options can be modified to perform inference (quantization, normalization, etc.).

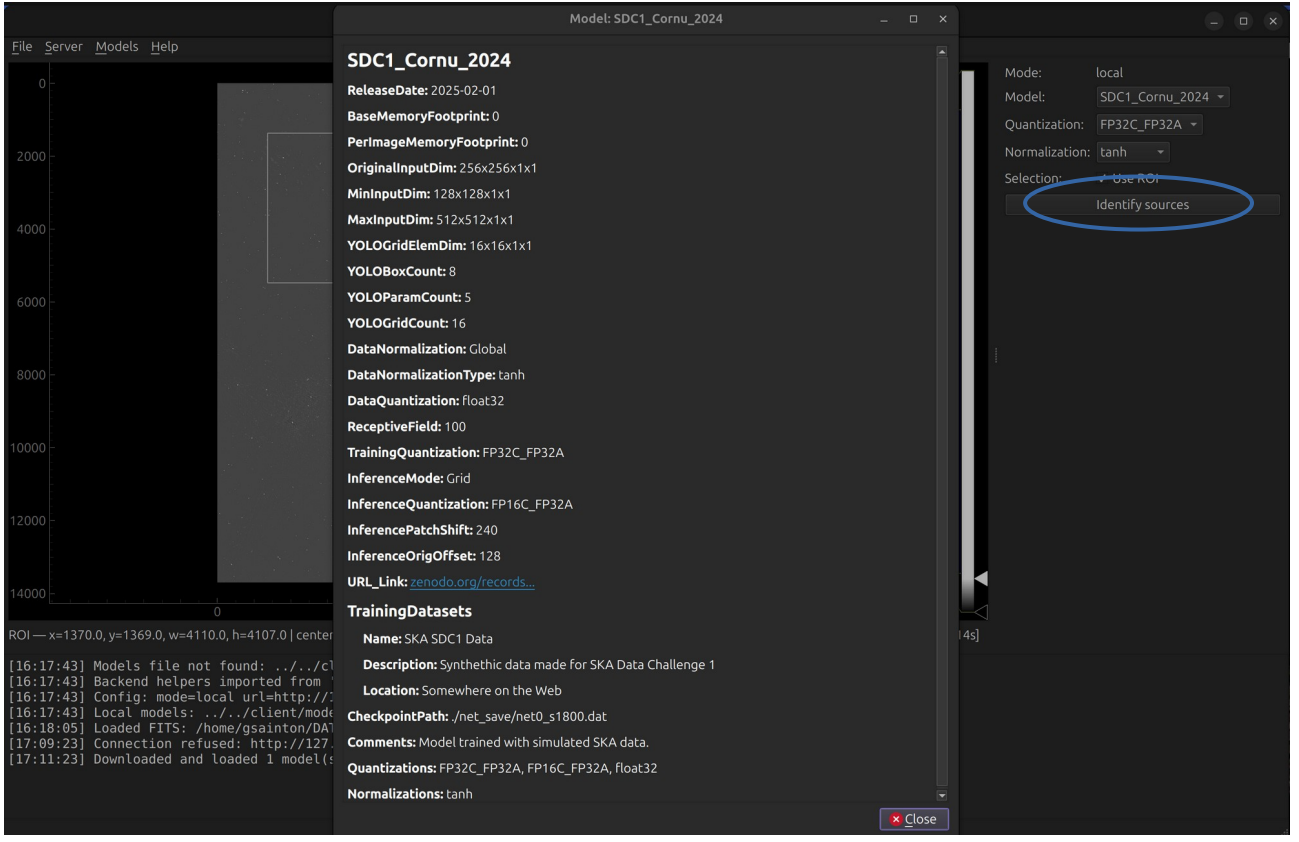
# Toy App for CIANNA-OTF




- When the app starts up: query the server to retrieve the list of available models;
- Each model contains information specific to its training;
- Certain options can be modified to perform inference (quantization, normalization, etc.).
- Once the parameters have been selected, you can request source identification.

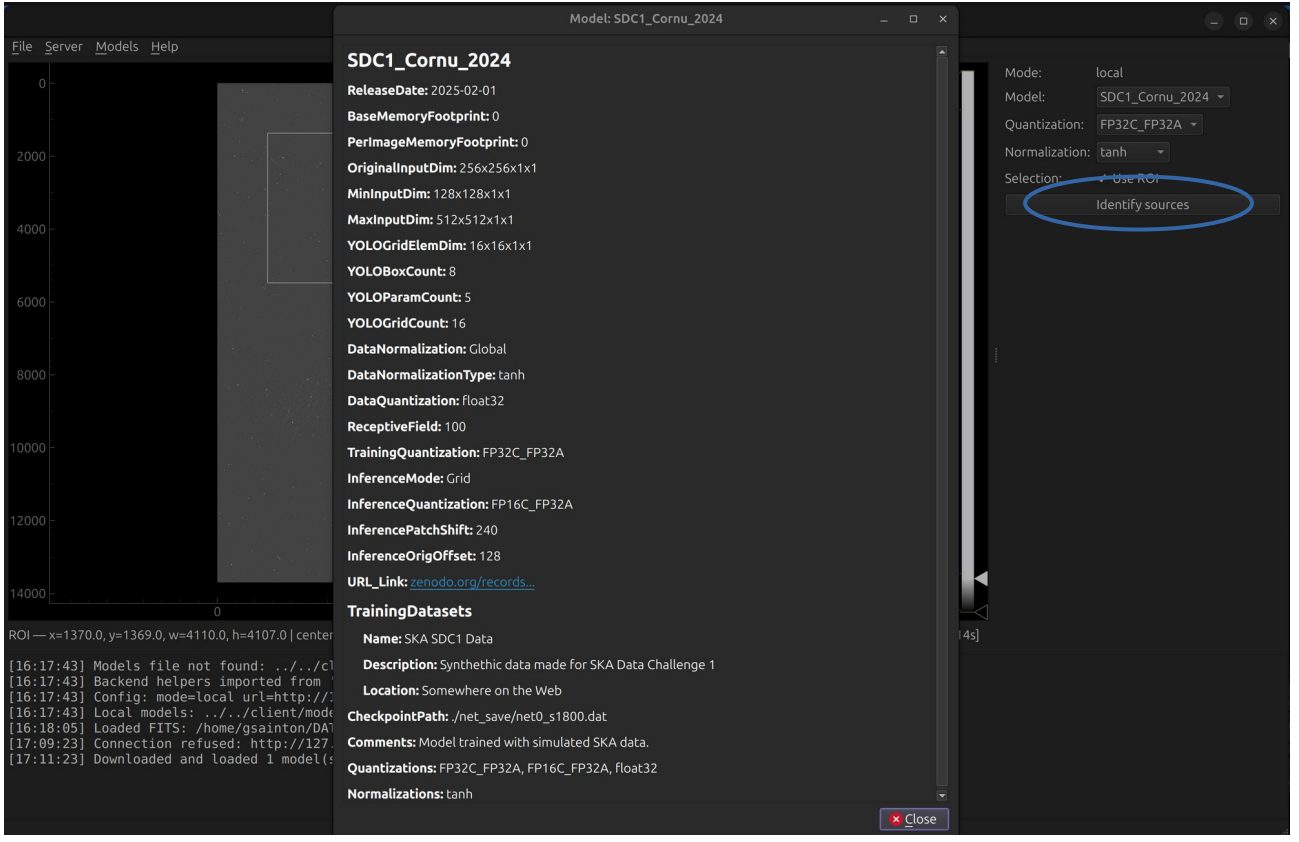
CIANNA DAYS

# Toy App for CIANNA-OTF




- When the app starts up: query the server to retrieve the list of available models;
- Each model contains information specific to its training;
- Certain options can be modified to perform inference (quantization, normalization, etc.).
- Once the parameters have been selected, you can request source identification;
- UWS file with your parameters and the image are sent to the server;

# Toy App for CIANNA-OTF




- When the app starts up: query the server to retrieve the list of available models;
- Each model contains information specific to its training;
- Certain options can be modified to perform inference (quantization, normalization, etc.).
- Once the parameters have been selected, you can request source identification;
- UWS file with your parameters and the image are sent to the server;
- The program is now waiting for a sign of life from the server.

CIANNA DAYS

# (Hyper)-Parameters sent

*via* the UWS file

# Client – (hyper)-parameters

| Parameters | Meaning | Definition |
|---|---|---|
| <OriginalInputDim> | Original input dimensions | Size of the images during the training |
| <MinInputDim>/ <MaxInputDim> | Input dimension range | Minimum and maximum size that the model can accept. |
| <YOLOGridElemDim> | Size of the YOLO grid cell | Each image is divided into small regions (grids) to detect multiple sources at once. |
| <YOLOGridCount> | Total number of grid cells | Number of subdivisions on which the model makes its predictions. |
| <YOLOBoxCount> | Number of boxes (detections) per cell | Each cell can detect multiple objects or sources (8 in this case). |
| <YOLOParamCount> | Number of parameters per box | In each boxes, several parameters are fitted |

# Client – (hyper)-parameters

| Parameter | Signification | Explicitation |
|---|---|---|
| <DataNormalization> | Type of normalization | Data scaling method |
| <DataNormalizationType> | Function used | Ex : *tanh* means that images are scaled from [-1 ; +1] |
| <DataQuantization> | Digital accuracy | Format of numbers used (e.g., float32, i.e., 32-bit floating point) |
| <TrainingQuantization> | Format used in training | Indicates the accuracy of calculations during training |
| <InferenceQuantization> | Format used for inference | Precision for inference. |

# Definition

**Quantization**: technique that reduces the numerical precision of calculations performed by a machine learning model.

- This reduces its memory consumption and
- Speeds up its execution without any significant loss of performance.

| Type | Numeric Format | Precision | Common usage | Effect |
|---|---|---|---|---|
| FP32 | 32-bit floating-point | Very high | Training phase | Reference precision |
| FP16 | 16-bit floating-point | High | Fast inference on GPUs | 2x faster, 2x less memory |
| Int8 | 8-bit integer | Medium | Embedded deployment | 4x smaller, small accuracy loss |
| BF16 (Bfloat-16) | 16-bit « brain float » | High | HPC and cloud training | Wide dynamic range, efficient |

FP32C_FP32A ???

| Code | Meaning | Description |
|---|---|---|
| FP32C | FP32 Convolution | Convolutional computations use 32 bit precision |
| FP32A | FP32 Activation | Activation (layers output) use 32-bit precision |

# References

- UWS : https://ivoa.net/documents/UWS/20161024/REC-UWS-1.1-20161024.html