Centre de Calcul
de l'Institut National de Physique Nucléaire
et de Physique des Particules

JAX for Machine learning

The Python Library for Accelerating NumPy

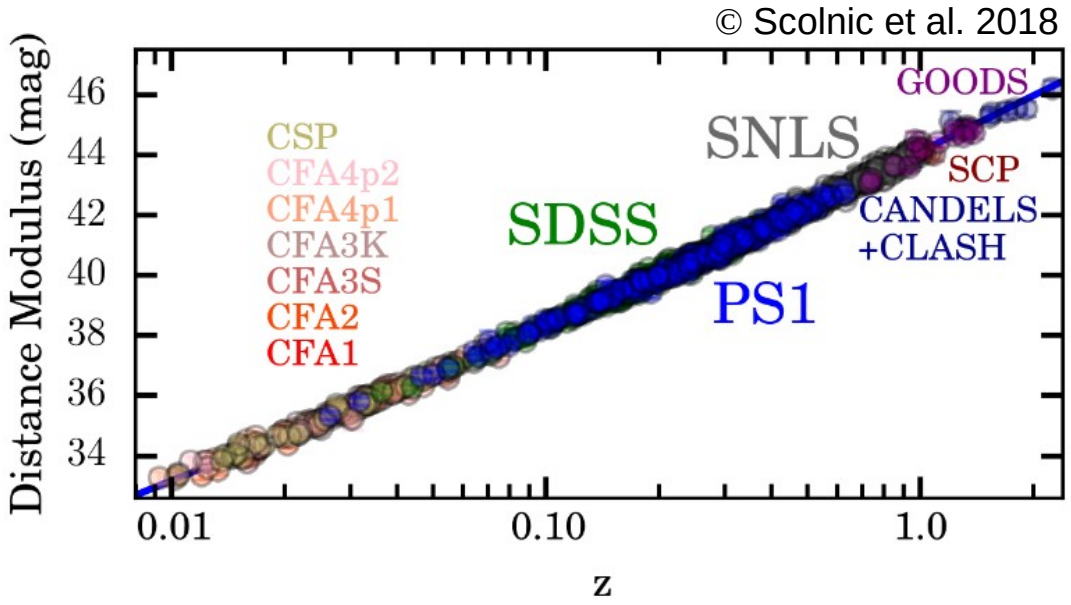
Overview

- **Introduction**
  - Cosmology and the Zwicky Transient Facility (ZTF)
  - Measurement and image calibration pipeline
  - Prospects for ZTF
- **JAX for Machine Learning**
  - Introduction to JAX
  - Applications
  - Disadvantages
- **Using JAX for ZTF**
  - Results
  - Comparison between NumPy and JAX
- **Conclusion**

Introduction

Cosmology and ZTF

- **Cosmology**



Study of Type Ia Supernovae (SNeIa) in order to understand the acceleration of the expansion of the universe

SNeIa are standard candles (fixed luminosity L)

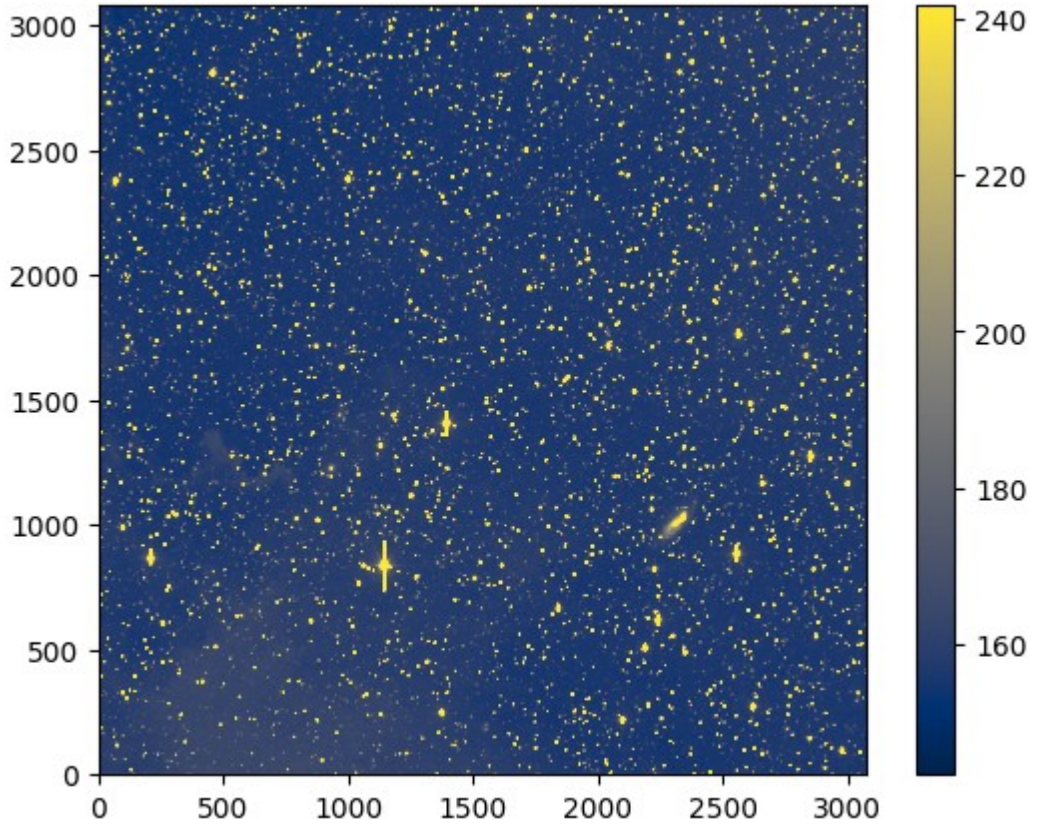
- **Zwicky Transient Facility (ZTF)**
  Palomar Mountain, California (USA), 2018

Collaboration: US National Science Foundation & universities and institutes of Europe and Asia

Scan the Milky Way plane twice a night and scan the entire northern sky in 3 night

Designed to detect transient objects (ex: SNe)

Measuring photometric luminosity



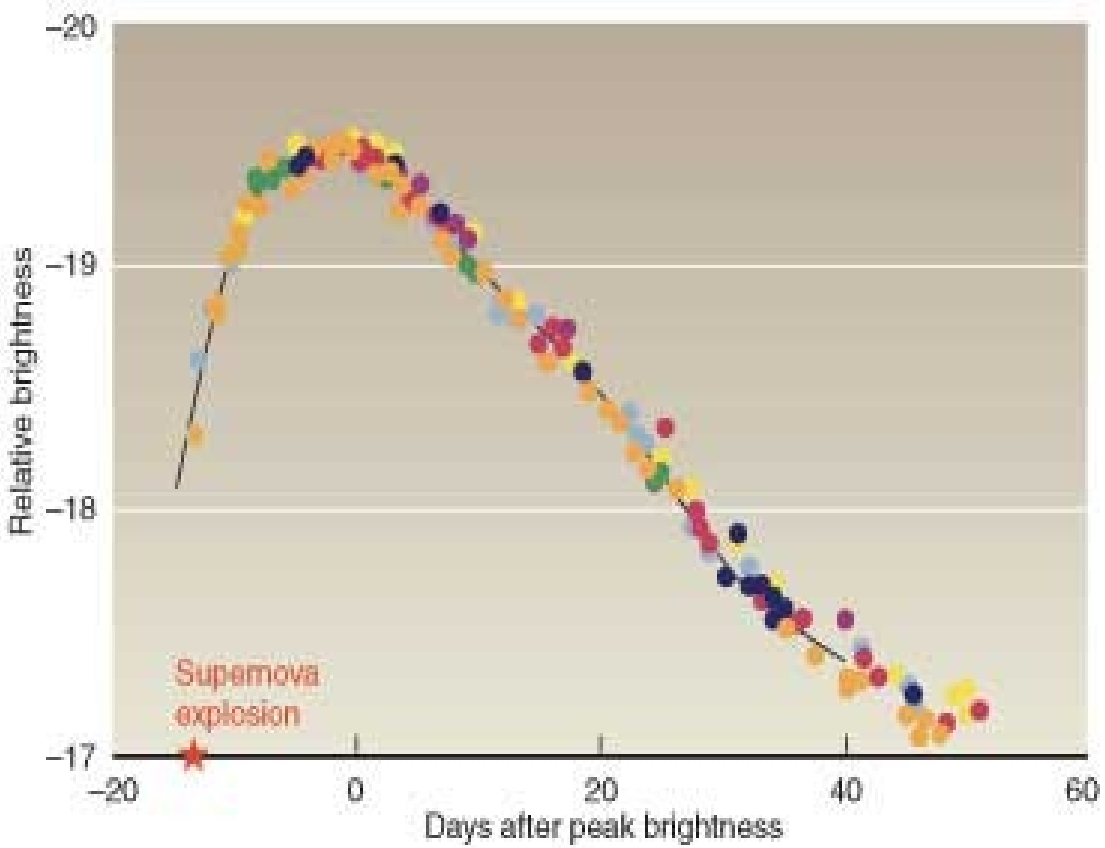
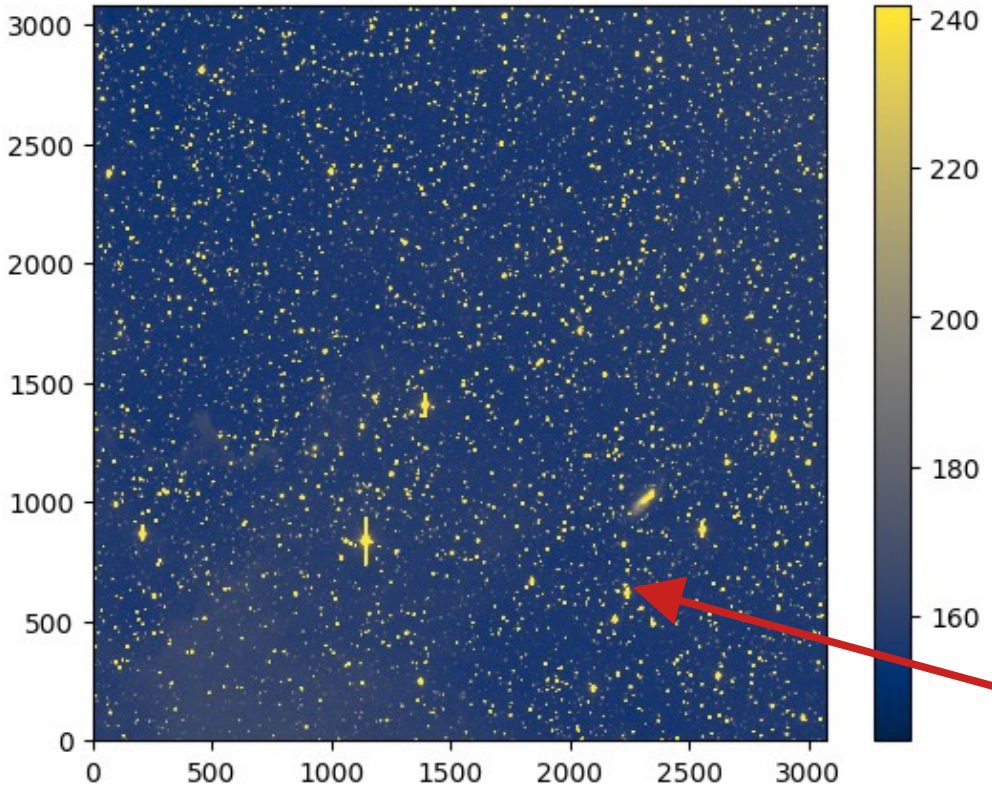


Image calibration pipeline

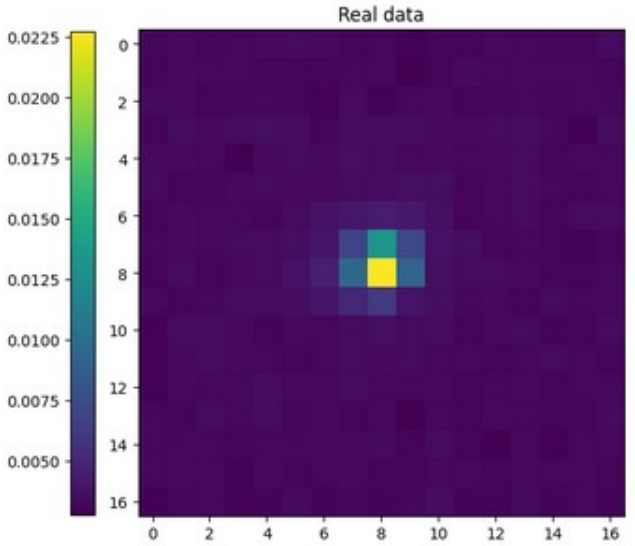


X 50 000 000 images

→ Master-bias (~ ms)

→ Master-flat (~ ms)

→ Flux Extraction (~ 10s)



X ~ 20 000

Prospects for ZTF

- **Requirements:**
  - Easy-to-use language to implement in existing official code
  - Ability to switch between CPU and GPU depending on machine availability
  - fast execution language

- **Available frameworks:**
  - TensorFlow (developed by Google Brain in 2015)
  - PyTorch (developed by Facebook AI Research in 2017)
  - JAX (developed by Google Research in 2018)

JAX for Machine Learning

Introduction to JAX

- **NumPy-like:** vector operations, linear algebra, arrays/matrices… (jax.numpy)

- **SciPy-like:** numerical computation, linear algebra… (jax.scipy)

- **Built to optimize performance and perform machine-learning tasks**:
  - Auto-differentiation (jax.grad)
  - Auto-vectorization (vmap)
  - Auto-parallelization (pmap)
  - Just-in-time compilation (@JIT)
  - Accelerated Linear Algebra (XLA)

- **JAX uses a GPU if available, by default:** possibility to specify CPU or GPU (jax.device_put)

Application

- **Auto-differentiation (jax.grad):** Obtain the gradient of a function of any order without having to write the derivative formula.
- **Automatic vectorization (vmap):** Automatically vectorizes all data points for fast calculations (matrix multiplication instead of scalar)
- **Parallelization (pmap):** Allows parallel programming on multiple GPUs
- **Just-in-time compilation (JIT):** tool for optimizing intensive calculations by transforming Python code into machine code
- **Accelerated Linear Algebra (XLA):** Executes elementary operations continuously instead of storing them in memory, which improves execution speed and saves memory.

Disadvantages

- Currently in full development (version 0.4.35, **last release on October 22th, 2024**)
- **Lot of changes** from one version to the next: older versions not documented
- JAX does not support all operating systems (need to use a virtual machine)

Supported platforms

	Linux x86_64	Linux aarch64	Mac x86_64	Mac aarch64	Windows x86_64	Windows WSL2 x86_64
CPU	yes	yes	yes	yes	yes	yes
NVIDIA GPU	yes	yes	no	n/a	no	experimental
Google TPU	yes	n/a	n/a	n/a	n/a	n/a
AMD GPU	yes	no	experimental	n/a	no	no
Apple GPU	n/a	no	n/a	experimental	n/a	n/a

Using JAX for ZTF

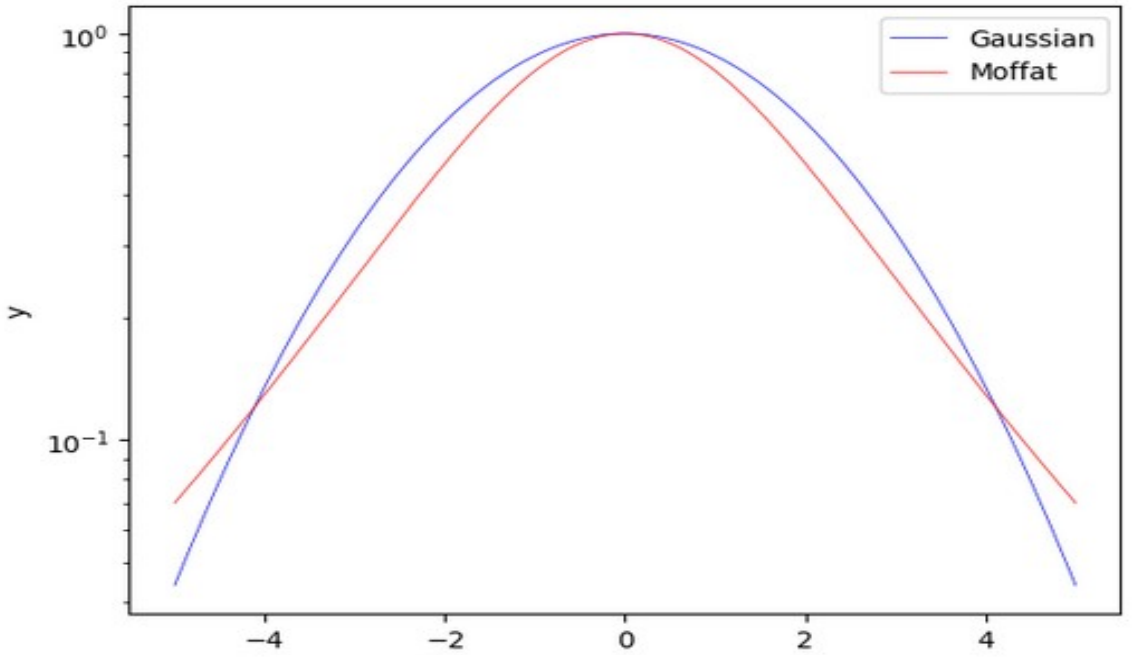
Models

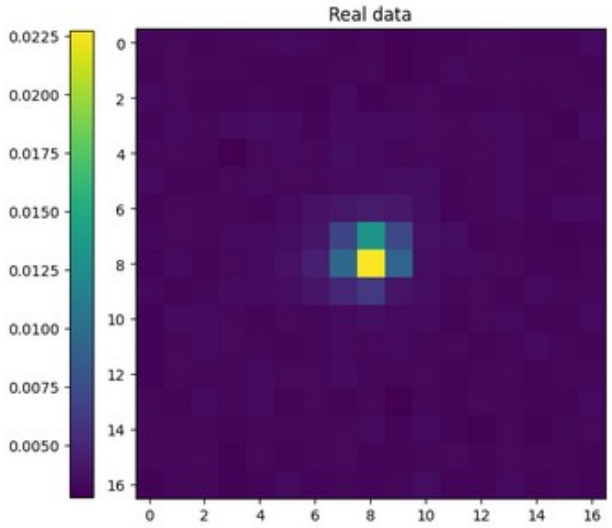
Gaussian distribution: First approach

$$f(x) = A \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right] \quad \text{with} \quad A = \frac{1}{\sqrt{2\pi \det(\Sigma)}}$$

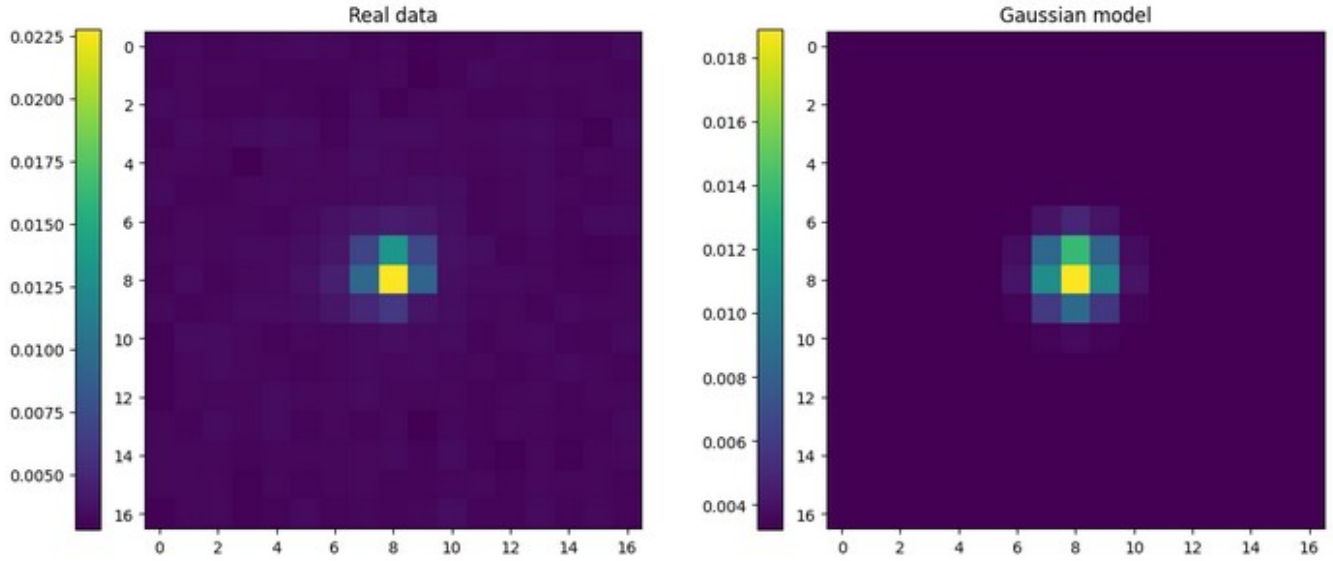
Moffat distribution: better point spread model

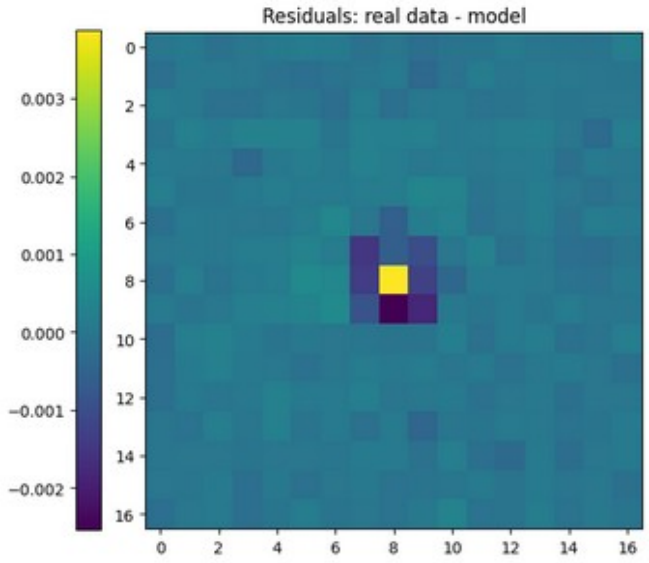
$$f(x, y, \alpha, \gamma) = A\left[1 + \left(\frac{(x-x_0)^2 + (y-y_0)^2}{\gamma^2}\right)\right]^{-\alpha} \quad \text{with} \quad A = \frac{\alpha - 1}{\pi \gamma^2}$$

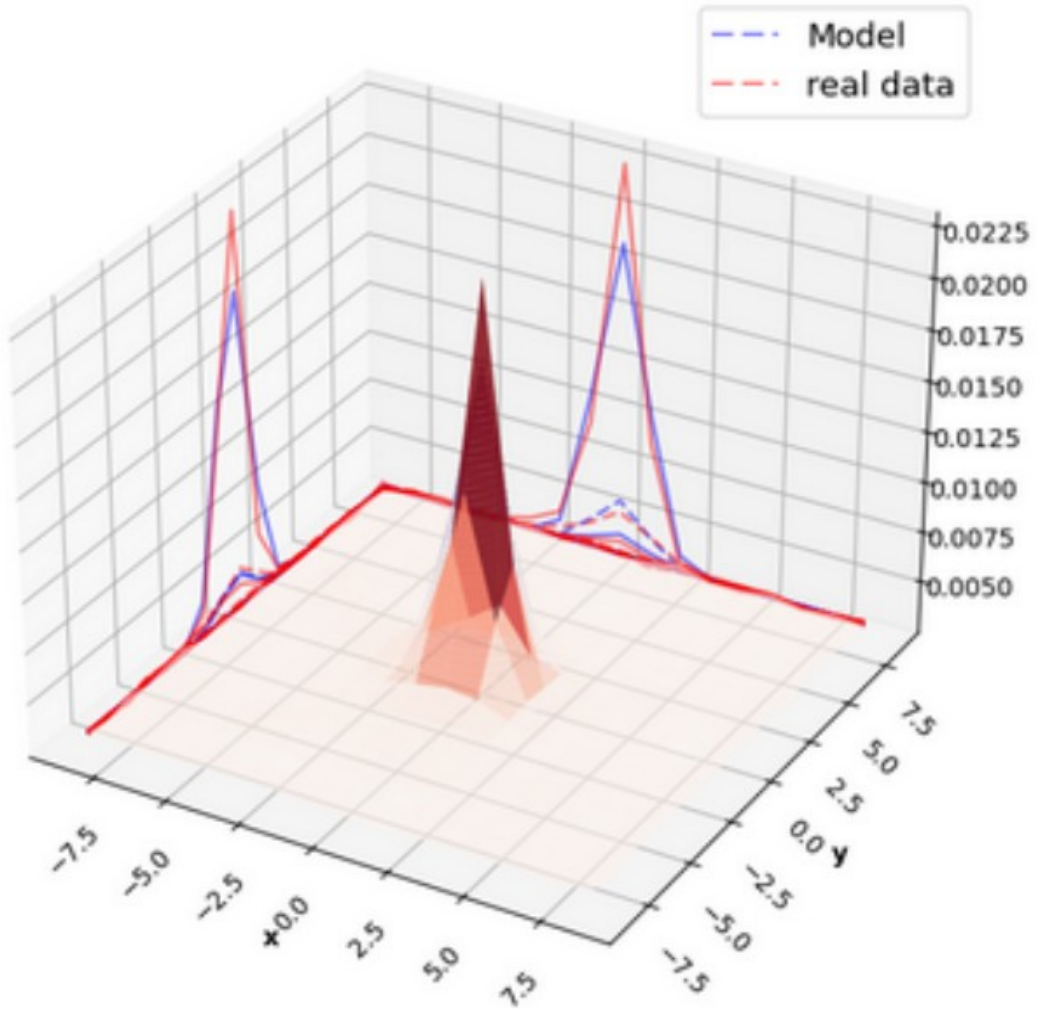




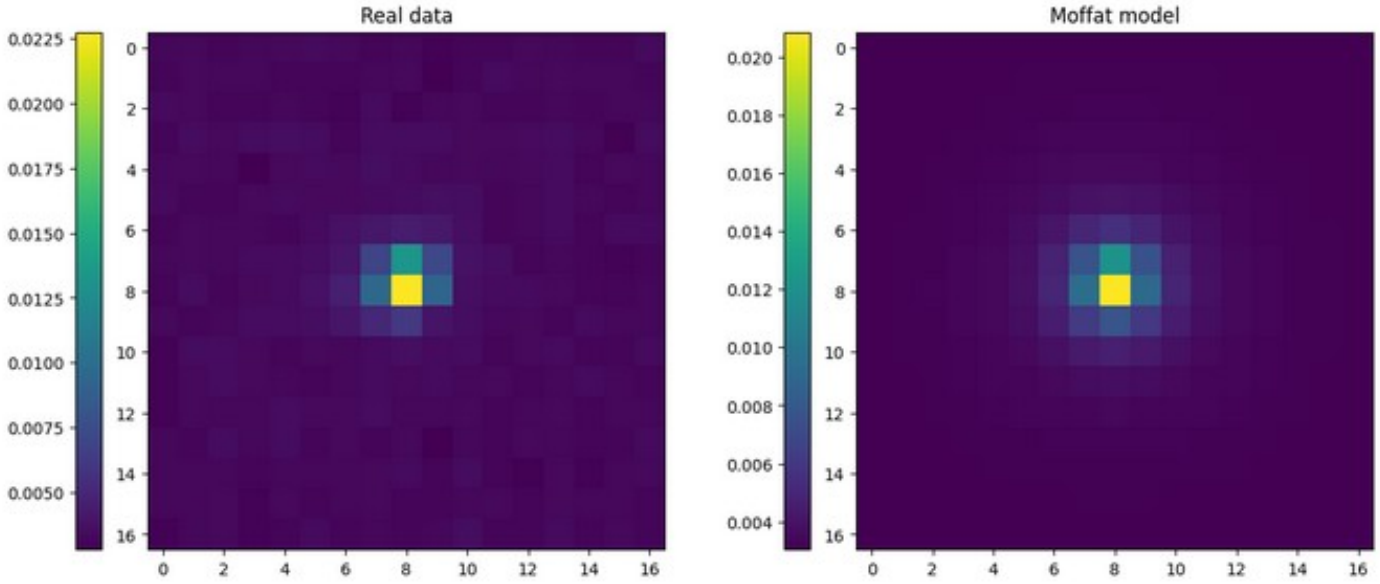
Gaussian model

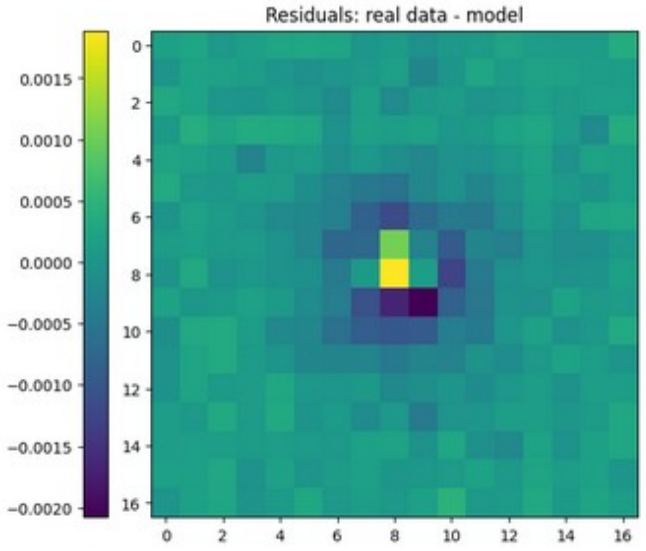


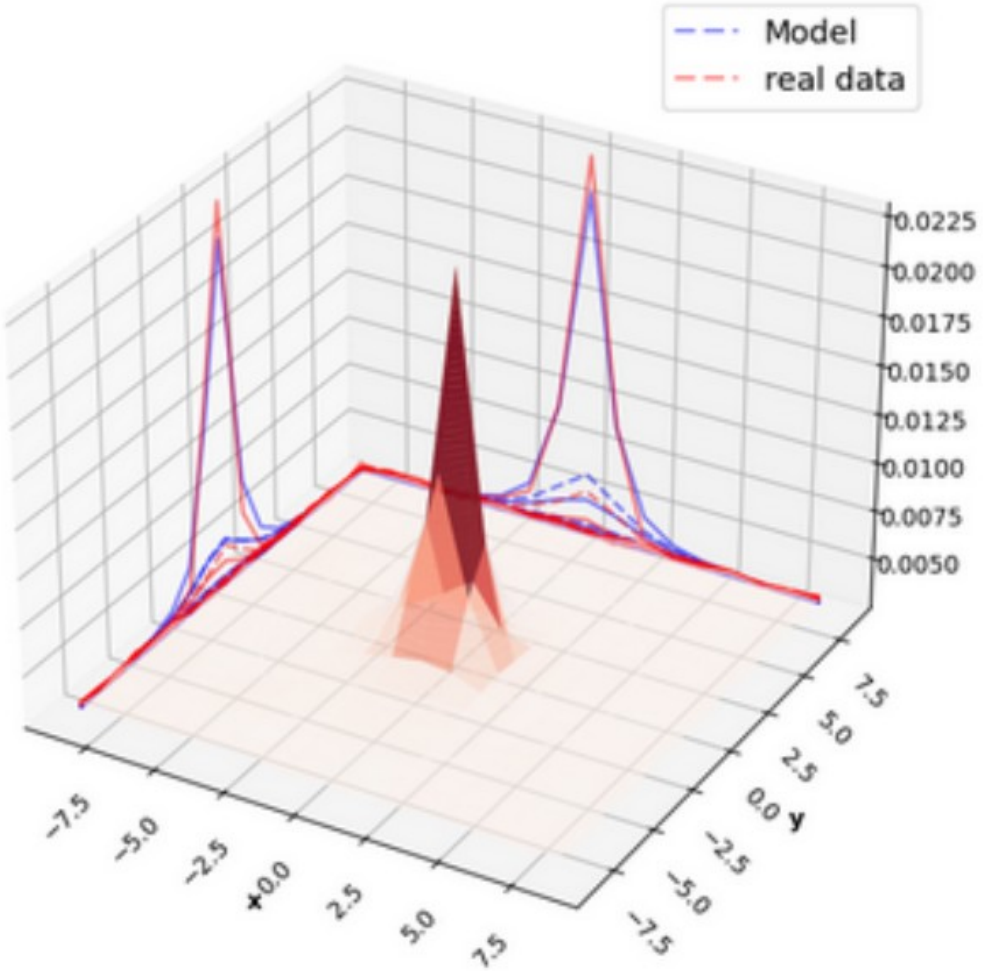




Moffat model

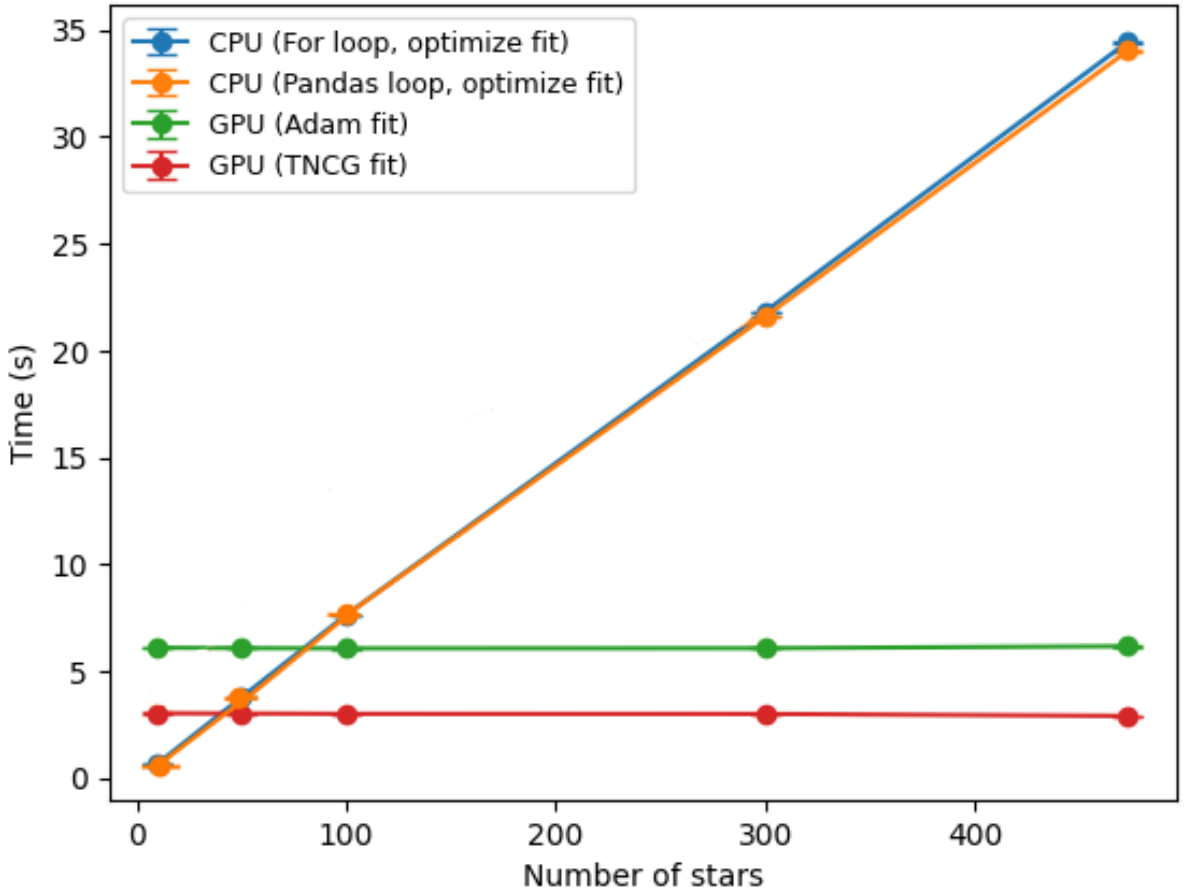




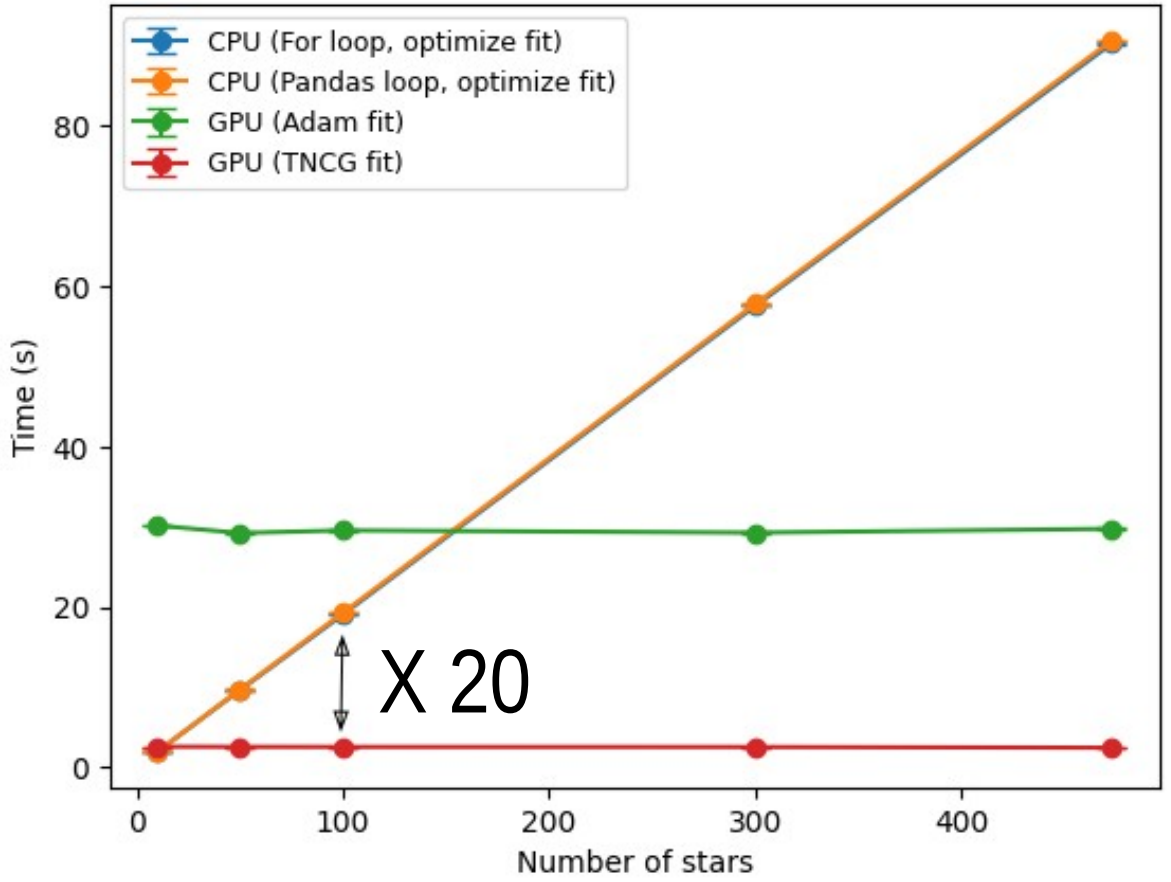


Execution speed

Gaussian model



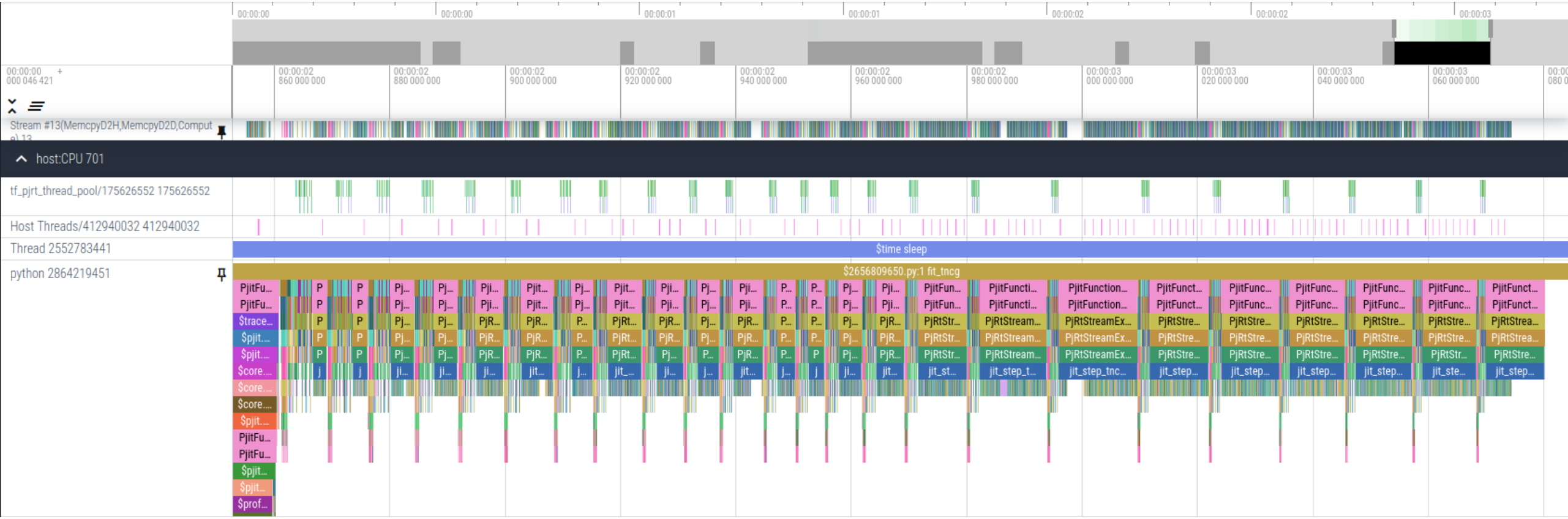
Moffat model



Conclusion

Profiling

https://perfetto.dev/docs/



Thank you for your attention