C - PY

Integration for AI

Why C Helps Optimization in AI with Python

- **Foundation Performance:**
  - Direct Hardware Access
  - Memory Management
  - Low-level Performance Tuning
  - Suitability for Computation-Intensive Tasks

Why C Helps Optimization in AI with Python

- **Execution Speed**
- Compiled Language
- Machine Code Translation
- Reduced Overhead
- Rapid Data Processing

Why C Helps Optimization in AI with Python

- **Seamless Integration**
- **Performance Optimization**
- **Cython Tool**
- **C-written Modules**
- **Efficiency and Convenience**

Python in AI

Simplicity and Readability

Extensive Libraries and Frameworks

Strong Community and Support

Role of C in Performance

- **Efficient Memory Management**
- **Close-to-Hardware Operations**
- **Advantages of a Compiled Language**

The Case for Integration

Limitation of Python in AI

- Its interpreted nature leads to slower processing speeds for computationally intensive tasks.

C for Enhancement

- Performance Boost: faster speed and efficiency,
- Better Resource Allocation: Reduce the overall memory footprint

The Case for Integration

Why we need integration?

- Use Python for rapid development and C for performance-critical sections.
- Improves processing efficiency and speeds.
- Opens up possibilities for real-time data processing, complex tasks, and advanced machine-learning models

Creating C Extensions in Python

Writing

Writing C Functions

Creating

Creating Python Wrappers

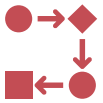
Compiling

Compiling C extensions

Importing

Importing in Python

Writing C Function



Find heavy time intensive processes

Data processing, loops, mathematical computations

Rewrite in C for efficiency

```
import cProfile
cProfile.run('your_function()')


void process_data(int* data, int size) {
  for (int i = 0; i < size; i++) {
    data[i] = /* Some intensive computation */;
  }
}
```

Wrap C Functions with Python/C API

- Include Python Header
- Wrapper function
  - Convert Python args to C
  - Call C function
  - Convert results back to Python args
- Method Table
  - Map python function names to C wrapper functions

```c
#include <Python.h>
```

```c
static PyObject* py_process_data(PyObject* self, PyObject* args) {
  int size;
  int* data;
  /* Parse arguments from Python to C */
  if (!PyArg_ParseTuple(args, "ii", &data, &size)) {
    return NULL;
  }
  process_data(data, size);
  return Py_BuildValue(""); // Return None in Python
}

static PyMethodDef ModuleMethods[] = {
  {"process_data", py_process_data, METH_VARARGS, "Process data efficiently."},
  {NULL, NULL, 0, NULL} // Sentinel
};
```

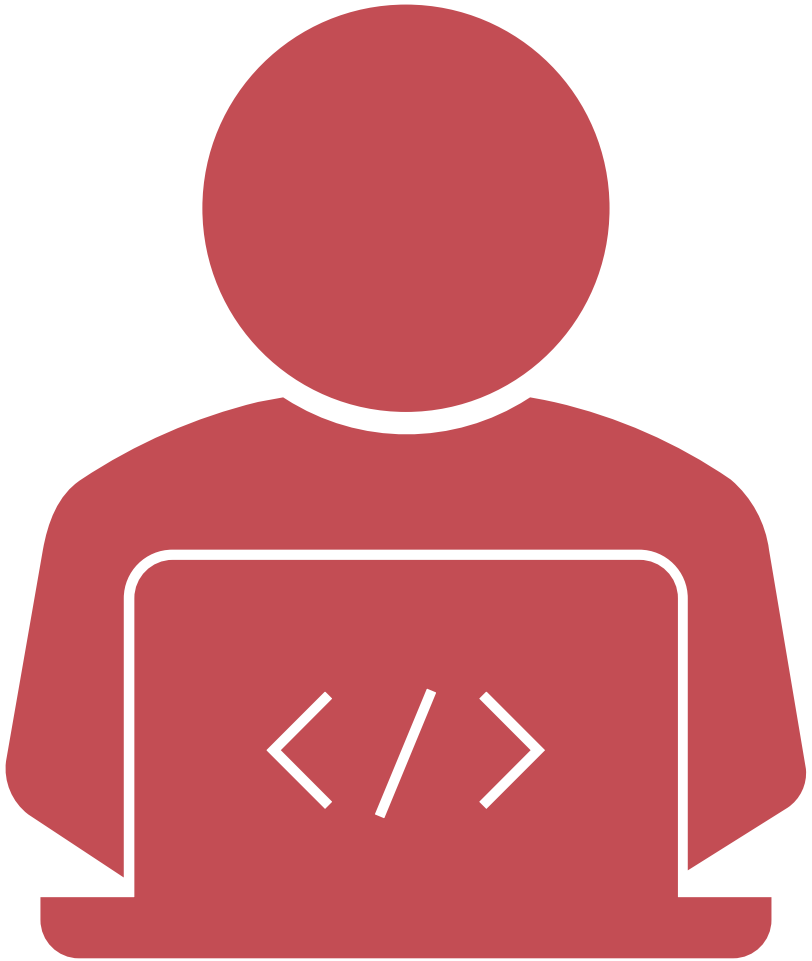
Compiling and Importing into Python

```
from setuptools import setup, Extension

module1 = Extension('your_module_name',
                    sources = ['your_module.c'])

setup(name = 'PackageName',
      version = '1.0',
      description = 'This is a demo package',
      ext_modules = [module1])

-------------------------------------------------------------

import your_module_name
your_module_name.process_data(data, size)
```

- Write a setup.py using setuptools
- Compile using python setup.py build_ext --inplace
- Import regularly as a module

Things to Keep in Mind

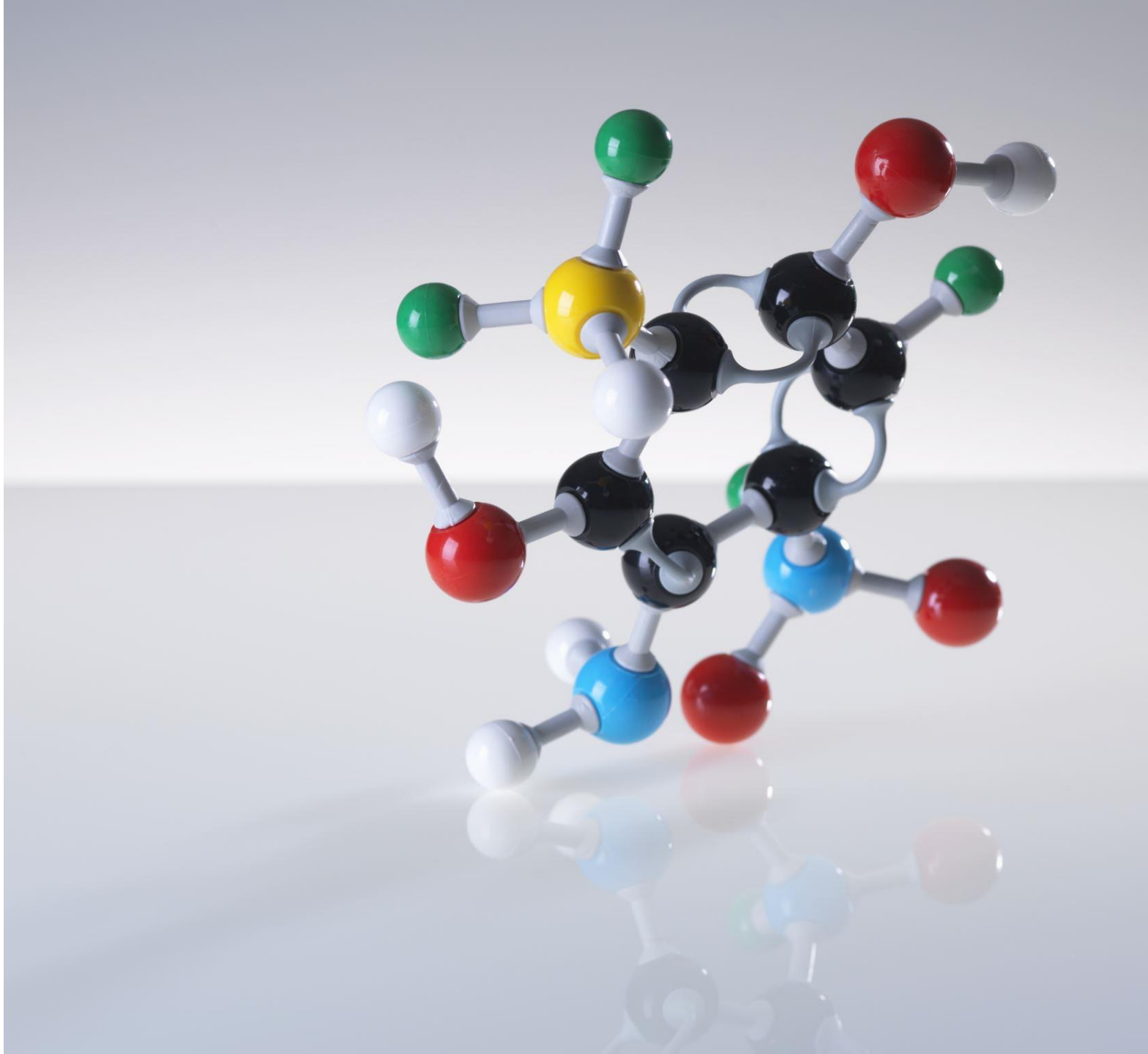
- Memory Allocation to avoid leaks
  - Python handles references to objects automatically.
  - Python objects in C:
    - `Py_INCREF()`
    - `Py_DECREF()`
- `Error handling`
  - `C doesn't catch as many errors as Python automatically`
    - `e.g. out of bounds array access`
- `Prioritize efficiency and simplicity`
  - `Make sure the C extensions improve performance and don’t bottleneck`

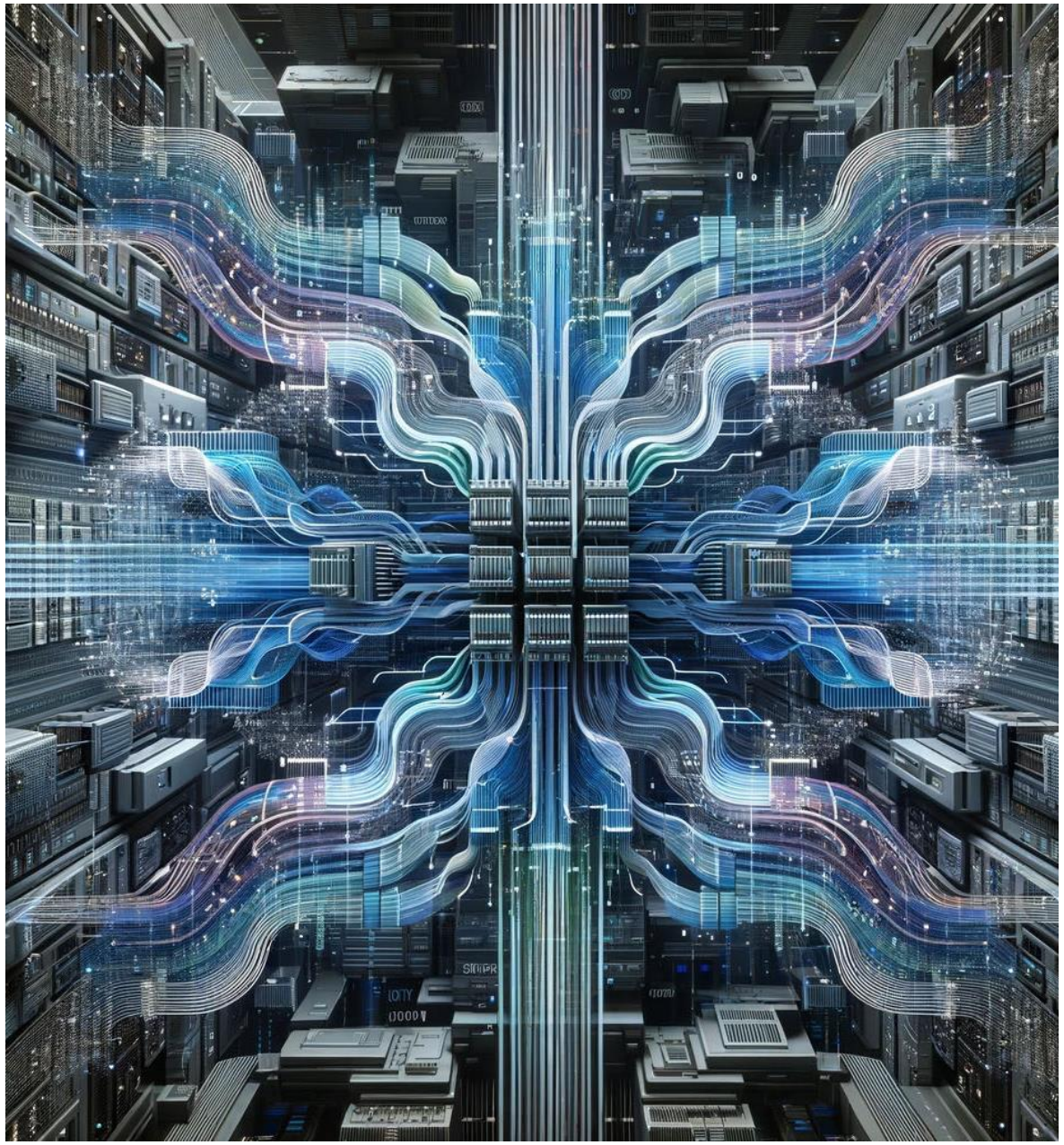


Practical AI Applications

What is Machine Learning

ML is a branch of artificial intelligence (AI) that focuses on developing algorithms and statistical models for tasks without explicit programming





C Extensions for Dataset Manipulation

Utilizing C extensions can significantly speed up data processing tasks in machine learning. By implementing critical functions in C and creating Python wrappers, developers can harness the performance benefits of C while retaining Python's ease of use.

Image processing

a collection of techniques and methods used to digitally manipulate images to improve their quality, enhance features, or achieve specific objectives.

Image processing operations

original

grayscale

enhanced

Edge detection

denoised

binarized

Python Limitations

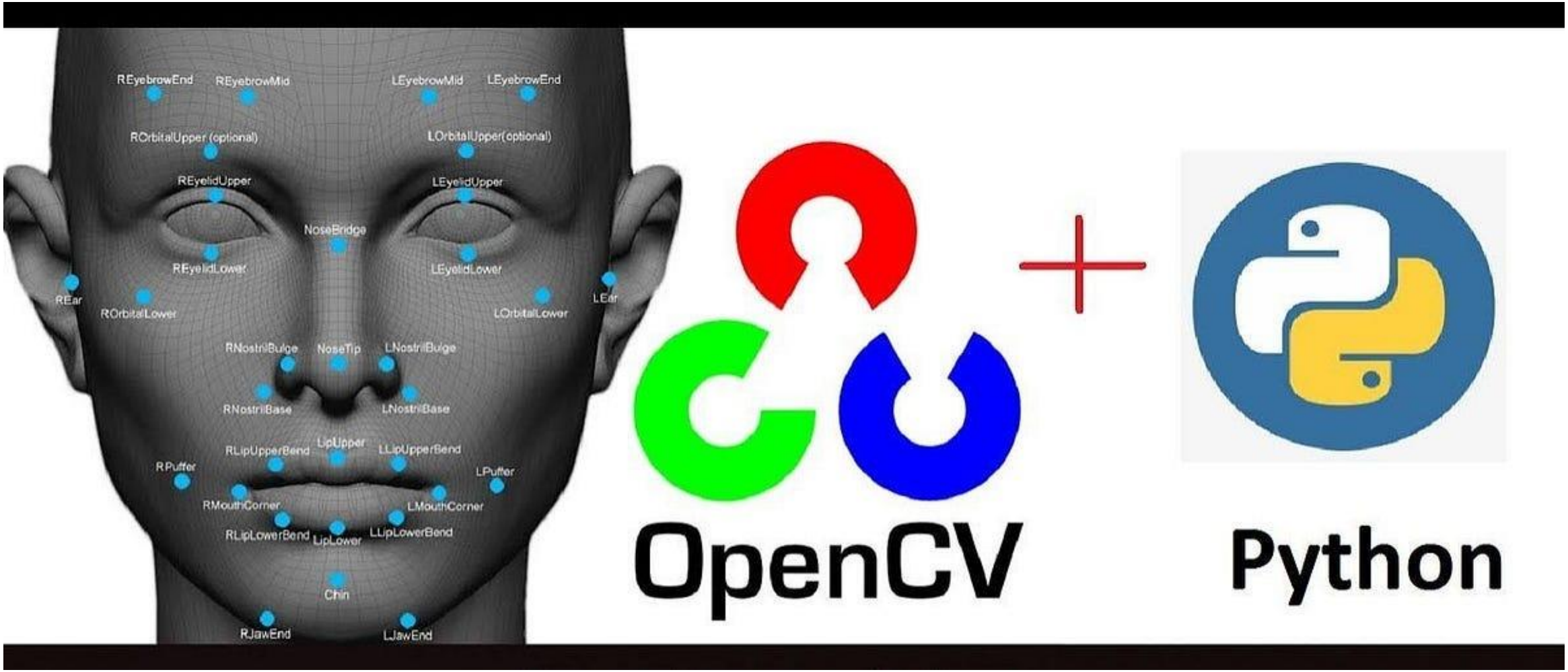
HANDLING LARGE-SCALE DATA

REAL-TIME PROCESSING

HIGH PERFORMANCE

C Extensions for Image processing

Python OpenCV Library



README.md — Space mistake in README.md — 3 weeks ago

SECURITY.md — Updated PGP key for security reports — last year

README | Apache-2.0 license | Security

OpenCV: Open Source Computer Vision Library

Resources

- Homepage: https://opencv.org
  - Courses: https://opencv.org/courses
- Docs: https://docs.opencv.org/4.x/
- Q&A forum: https://forum.opencv.org
  - previous forum (read only): http://answers.opencv.org
- Issue tracking: https://github.com/opencv/opencv/issues
- Additional OpenCV functionality: https://github.com/opencv/opencv_contrib
- Donate to OpenCV: https://opencv.org/support/

Contributing

Please read the contribution guidelines before starting work on a pull request.

Contributors 1,570

+ 1,556 contributors

Languages

C++ 87.4% C 3.3%
Python 2.7% CMake 2.0%
Java 1.6% Objective-C++ 0.8%
Other 2.2%

The underlying code of OpenCV is actually written in C and C++, accounting for 90% of its composition.

Conclusion

Key benefits of integrating C with Python in AI applications

- By leveraging the benefit of C for performance-critical tasks, we can significantly improve the processing speed and efficiency.

- Handle larger datasets, perform more complex computations, and achieve faster processing times.

Conclusion

Potential future applications

- ❖ The optimization of image processing through C extensions could lead to breakthroughs in real-time computer vision applications.

- ❖ Significant implications for autonomous vehicles-safe navigation.

- Other fields
  Medical field, improve the quality of assessment tools.

Question？

Thanks！

- [1] “Machine Learning on AWS,” Amazon Web Services, Available: https://aws.amazon.com/what-is/machine-learning/?nc1=h_ls. [Accessed: 2024.03.25].

- [2] OpenCV. (2024.03.25). OpenCV. [Online]. Available: https://github.com/opencv/opencv. [Accessed: 2024.03.25].
- Smith, Ross. *Performance of MPI Codes Written in Python with NumPy and Mpi4py*. 13 Nov. 2016, pp. 45–51, https://doi.org/10.5555/3019083.3019089. Accessed 27 Mar. 2024.
- “Wrapping C/C++ for Python — Intermediate and Advanced Software Carpentry 1.0 Documentation.” *Intermediate-And-Advanced-Software-Carpentry.readthedocs.io*, intermediate-and-advanced-software-carpentry.readthedocs.io/en/latest/c++-wrapping.html. Accessed 27 Mar. 2024.
- Authority, Python Packaging. “Setuptools: Easily Download, Build, Install, Upgrade, and Uninstall Python Packages.” *PyPI*, pypi.org/project/setuptools/.

- All code generated by ChatGPT