

MOJO



A NEW HOPE FOR AI ?

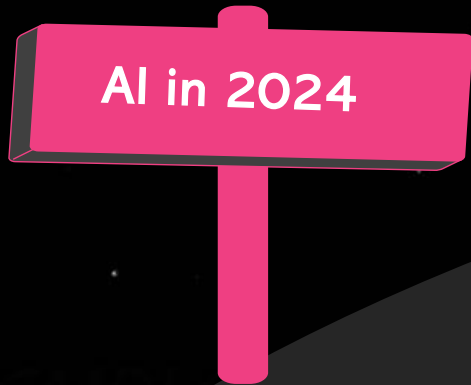
MOJO 

THE NEXT AI LANGUAGE ?



python

4.0



Value Proposition



Use Case



**What is
The current language of
AI ?**

C3-GPTo ADVISOR

L'IA, ou intelligence artificielle, peut être développée et programmée dans différents langages de programmation. Certains des langages les plus couramment utilisés pour créer des systèmes d'IA incluent Python, Java, C++, et R, entre autres. Python est particulièrement populaire dans le domaine de l'IA en raison de sa simplicité, de sa flexibilité, de sa large gamme de bibliothèques et de frameworks dédiés à l'IA (comme TensorFlow, PyTorch, scikit-learn, etc.), et de sa communauté active de développeurs.





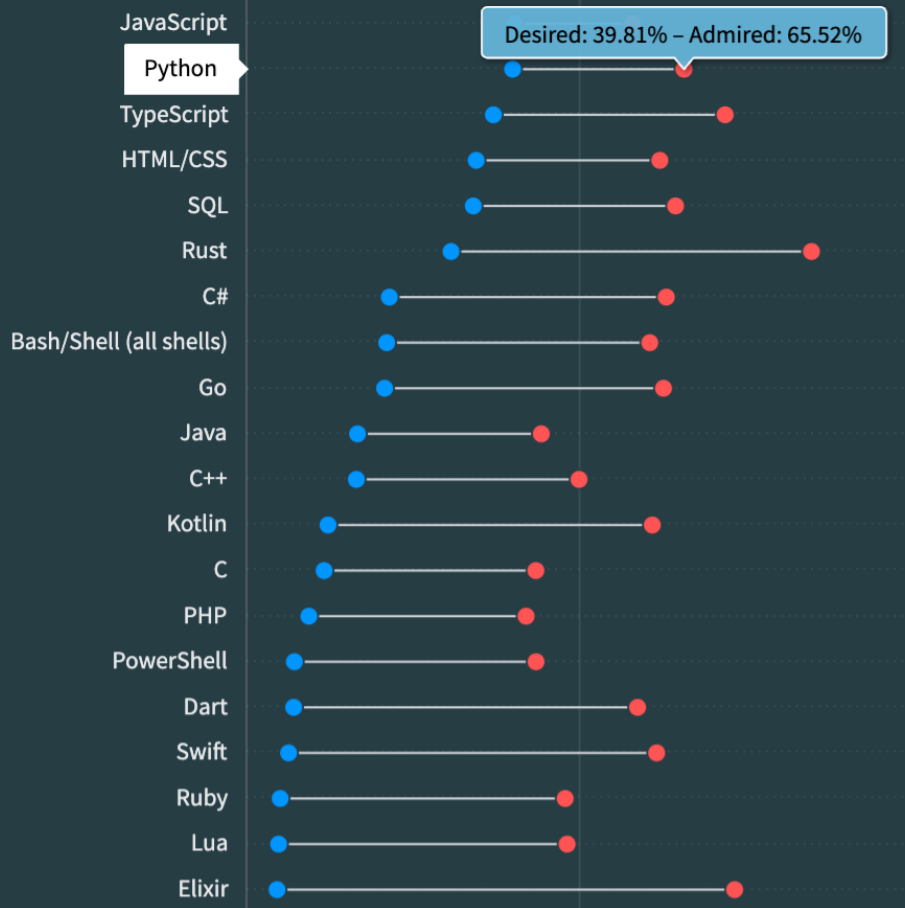
2023 Developer Survey

In May 2023 over 90,000 developers responded to our annual survey about how they learn and level up, which tools they're using, and which ones they want.

[Read the overview →](#)[Methodology →](#)[Overview](#)[Developer Profile](#)[Technology](#)[AI](#)[Work](#)[Community](#)[Professional Developers](#)[Methodology](#)

STACKOVERFLOW ADVISOR

87,510 responses



Overview

Developer Profile

Technology

Most popular technologies

Admired and Desired

Worked with vs. want to work with

Top paying technologies

AI

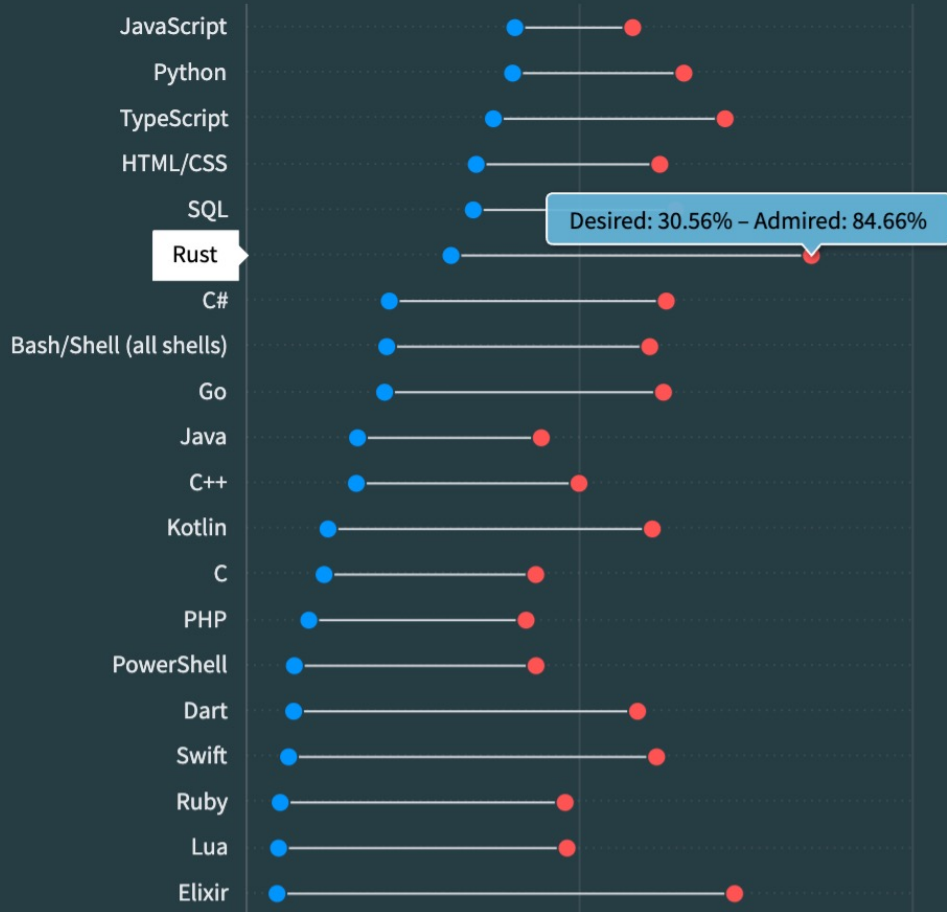
Work

Community

Developer Profile

STACKOVERFLOW ADVISOR

87,510 responses



Overview

Developer Profile

Technology

Most popular technologies

Admired and Desired

Worked with vs. want to work with

Top paying technologies

AI

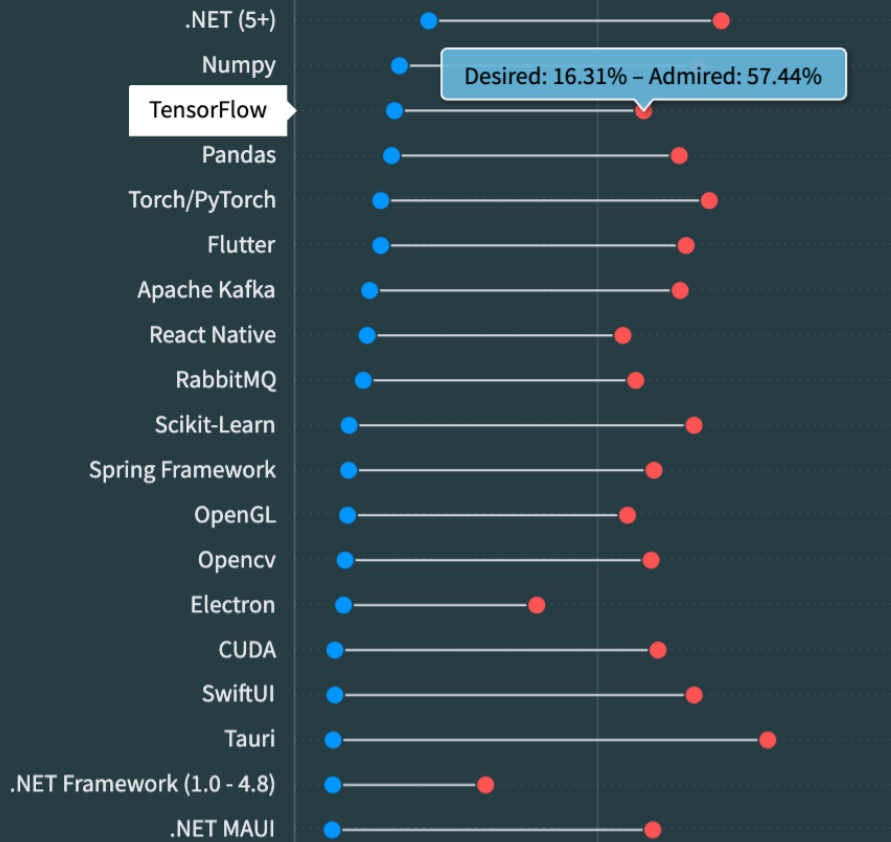
Work

Community

Professional Development

STACKOVERFLOW ADVISOR

66,235 responses



Overview

Developer Profile

Technology

Most popular technologies

Admired and Desired

Worked with vs. want to work with

Top paying technologies

AI

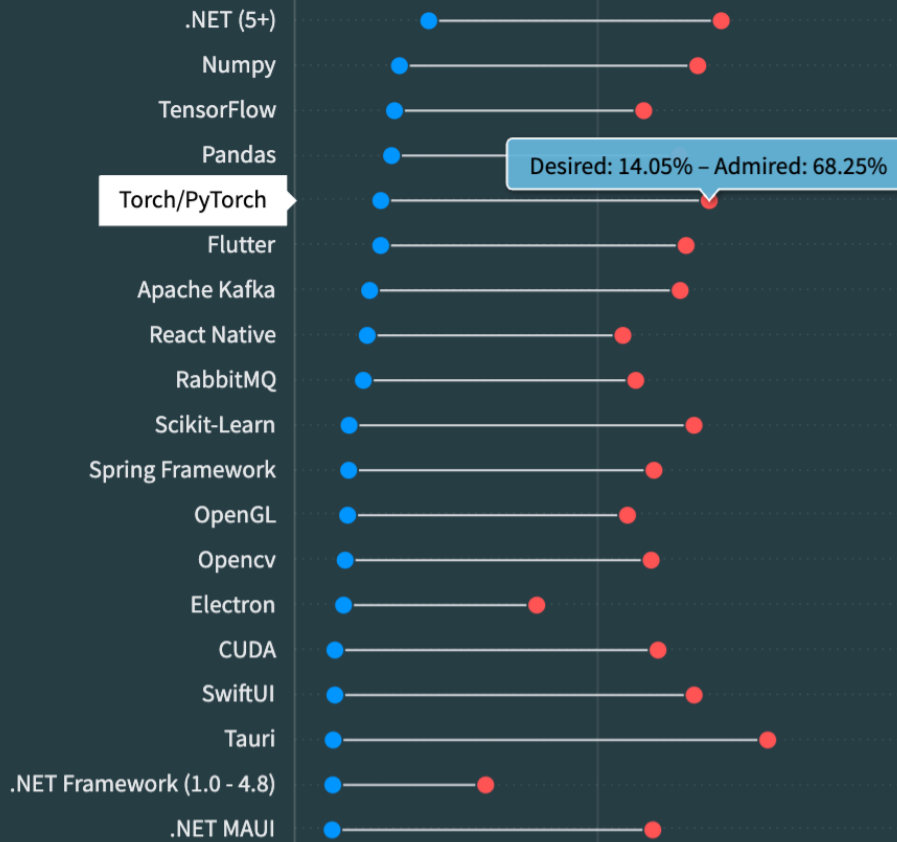
Work

Community

Developer Profile

STACKOVERFLOW ADVISOR

66,235 responses



Overview

Developer Profile

Technology

Most popular technologies

Admired and Desired

Worked with vs. want to work with

Top paying technologies

AI

Work

Community

Developer Profile

HOW TO CHOOSE A FRAMEWORK ?

DEVOXX FRANCE 2023

1^{ère} Edition - 12 AU 14 AVRIL 2023



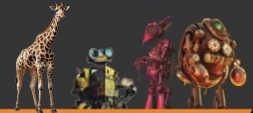
Devovx France 2023

Deep learning en Python : comment choisir un framework ?

Justine Bel-Létoile
HelloWork

jeudi 13 avril 2023

@Devovxfr



<https://www.youtube.com/watch?v=k4Tfg6-7cyQ>

AI PROGRAMMING LANDSCAPE

Model



System



Hardware

CUDA, OpenCL, ROCm


NEW KID IN TOWN!

Mojo 

02/05/2023

Modular


Contact sales Sign In Menu




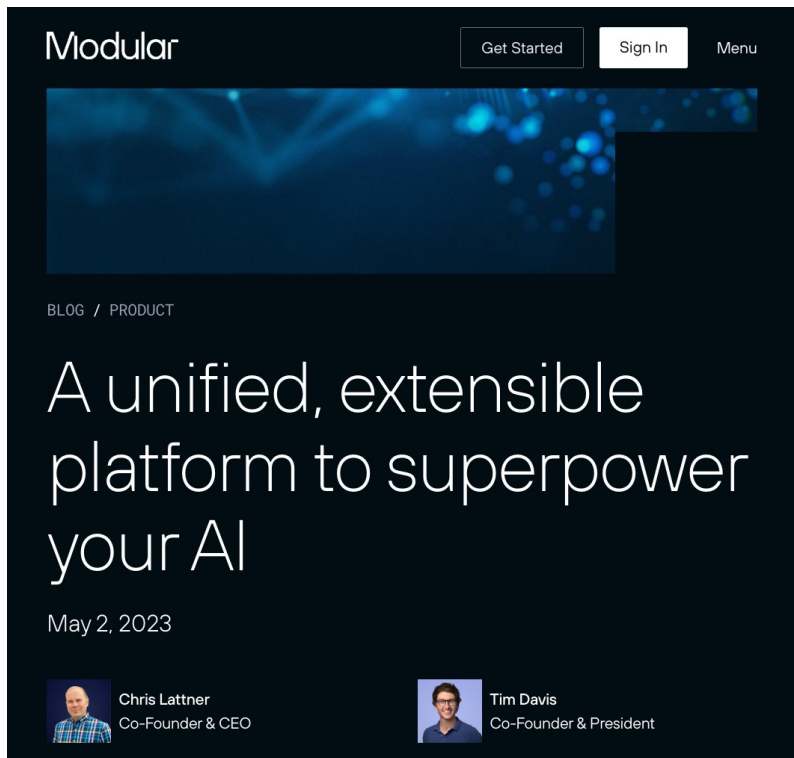
BLOG / COMPANY

The future of AI depends on Modularity

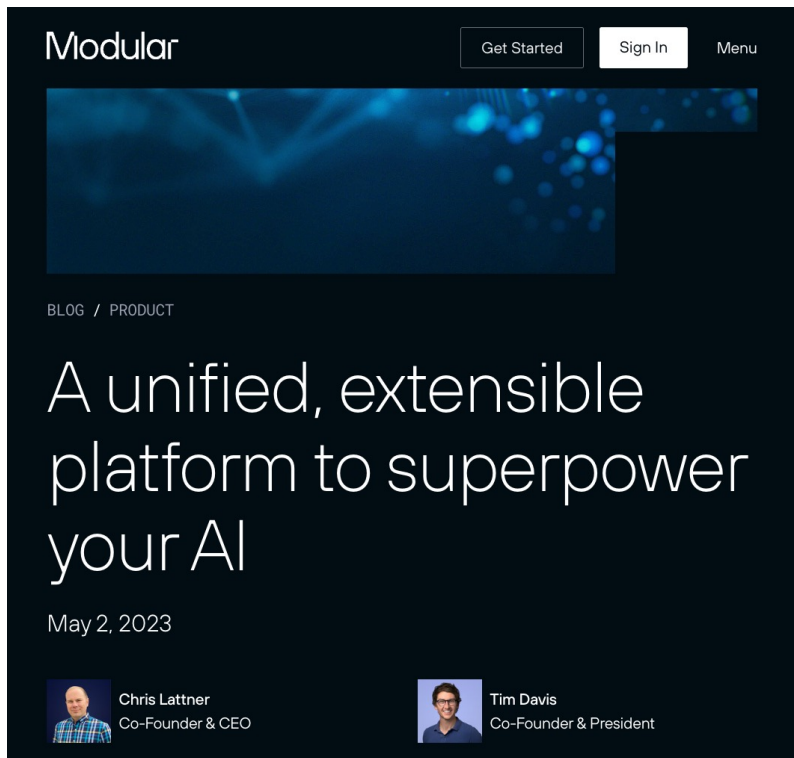
April 26, 2022

 Chris Lattner
Co-Founder & CEO

 Tim Davis
Co-Founder & President

A screenshot of a web page header for a blog post. The page has a dark background. At the top left, the word "Modular" is written in white. To its right are two buttons: "Get Started" and "Sign In", both in white text on dark backgrounds. Further right is a "Menu" link. Below the navigation is a large, abstract blue and white graphic with glowing points and lines. Underneath the graphic, the text "BLOG / PRODUCT" is displayed in white. The main headline reads "A unified, extensible platform to superpower your AI" in large white font. Below the headline is the date "May 2, 2023". At the bottom, there are two author profiles: "Chris Lattner, Co-Founder & CEO" and "Tim Davis, Co-Founder & President", each with a small circular profile picture.

Modular Accelerated eXecution platform



The image shows a screenshot of a web page header for a blog post. At the top left, the word "Modular" is displayed in a white sans-serif font. To its right are three navigation elements: a "Get Started" button, a "Sign In" button, and a "Menu" link. Below the navigation is a large, dark blue banner image featuring a network of glowing nodes and connecting lines. Underneath the banner, the text "BLOG / PRODUCT" is shown in a small, white, uppercase font. The main headline is "A unified, extensible platform to superpower your AI", written in a large, white, sans-serif font. Below the headline, the date "May 2, 2023" is displayed in a smaller white font. At the bottom of the header, there are two author profiles. The first profile includes a small square portrait of Chris Lattner, followed by his name "Chris Lattner" and his title "Co-Founder & CEO". The second profile includes a small square portrait of Tim Davis, followed by his name "Tim Davis" and his title "Co-Founder & President".

- Member of the python family (superset of python)
- Support modern chip architectures (thanks to MLIR)
- Predictable low level performance



Chris Lattner

2000 beginning of the project LLVM

2003 release of LLVM 1.0

🍏 2007 release of Clang 1.0

2008 XCode 3.1

2011 Clang replace gcc on macos

2014 release of Swift 1.0

🇸 2018 beginning of the MLIR

2022 creation of Modular cie

2023 🔥

MOJO IS BLAZING FAST!

Modular Contact sales Sign In Menu



BLOG / DEVELOPER

How Mojo 🔥 gets a 35,000x speedup over Python – Part 1

August 18, 2023



Abdul Dakkak
AI Compiler Engineer

<https://www.modular.com/blog/how-mojo-gets-a-35-000x-speedup-over-python-part-1>



MOJO IS BLAZING FAST!



Modular

Contact sales Sign In Menu

BLOG / DEVELOPER

How Mojo 🔥 gets a 35,000x speedup over Python – Part 1

August 18, 2023

Abdul Dakkak
AI Compiler Engineer

<https://www.modular.com/blog/how-mojo-gets-a-35-000x-speedup-over-python-part-1>

Changelog

2022/01 incorporation

2022/07 seed round (30 M\$)

2023/05 announce MAX & Mojo

2023/08 serie B (100 M\$)

🐦 2023/09 release mojo 0.2.1

🍏 2023/10 release mojo 0.4.0

..

2024/01 release mojo 0.7.0

2024/02 release MAX & mojo 24.1

🍏 2024/06 release MAX & mojo 24.4

MOJO IS BLAZING FAST!

Modular Contact sales Sign In Menu



BLOG / DEVELOPER

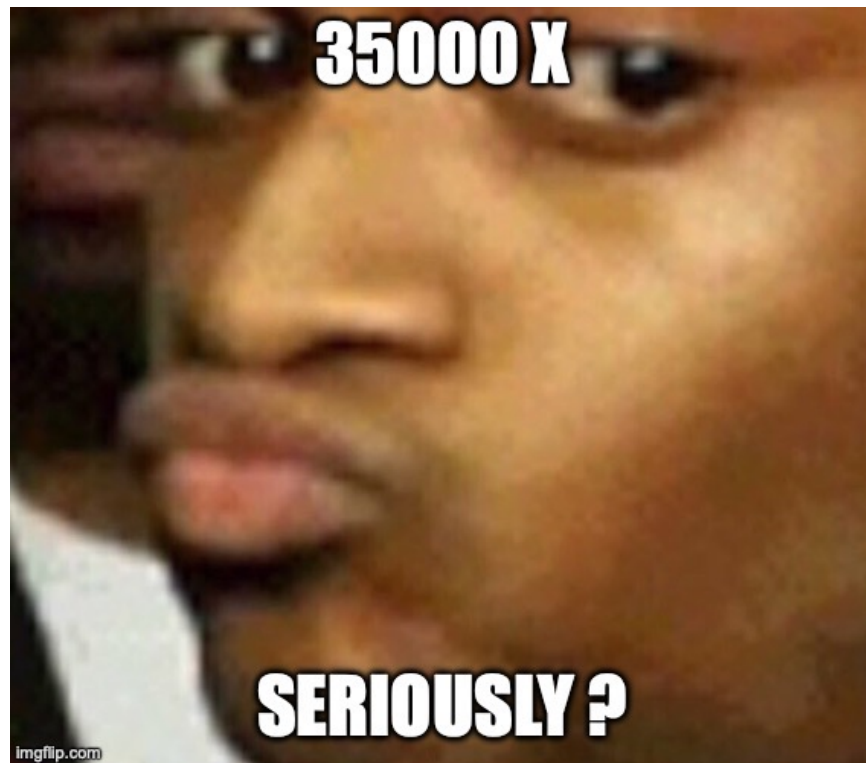
How Mojo 🔥 gets a 35,000x speedup over Python – Part 1

August 18, 2023



Abdul Dakkak
AI Compiler Engineer

<https://www.modular.com/blog/how-mojo-gets-a-35-000x-speedup-over-python-part-1>



Performance matters :

- for our users

Your resume is being processed



PERFORMANCE MATTERS!

Performance matters :

- for our users
- for (artificial) intelligence

PERFORMANCE MATTERS!

CodhGame
by CoderPad

ÉVÉNEMENTS SOLO MULTI COMMUNAUTÉ

Rechercher

SE CONNECTER S'INSCRIRE

ABALONE

Par DomiKo

97 CodinGamers dans l'arène de Abalone

Approuvé par [trlctrac](#), [field3](#), [VulpesCorsac](#)

















REJOINDRE

DÉTAILS **CLASSEMENT** DISCUSSIONS PARTAGER

Monde

Rechercher un ami, une école, ...

Ligue CodinGamer Langage Score École / Entreprise Pays

1	 trlctrac VOIR DERNIERS COMBATS	C	46,24	Centrale Lille / Thales	
2	 LeRenard VOIR DERNIERS COMBATS	C++	44,24	N/A	
3	 MuddySneakers VOIR DERNIERS COMBATS	Rust	42,74	Cornell University - Ithaca, NY	
4	 EricSMO VOIR DERNIERS COMBATS	C++	42,70	Ensimag - Grenoble INP	
5	 KaluTheFrog VOIR DERNIERS COMBATS	C	36,68	ISAE-ENSMA / Mensa France	
6	 Benoit_Belasco VOIR DERNIERS COMBATS	Python 3	34,11	FANB - Monaco-Ville / Mensa France	
7	 field3 VOIR DERNIERS COMBATS	Python 3	31,26	Chiba University - Chiba-shi / freelance	
8	 _Royale	C++	30,96	N/A	

PERFORMANCE MATTERS!

Performance matters :

- for our users
- for (artificial) intelligence
- for the planet

PERFORMANCE MATTERS!

Ranking Programming Languages by Energy Efficiency

Rui Pereira^{a,b}, Marco Couto^{c,b}, Francisco Ribeiro^{c,b}, Rui Rua^{c,b}, Jácome Cunha^{c,b}, João Paulo Fernandes^d, João Saraiva^{c,b}

^a*CI — Centro de Competências em Cloud Computing (CI-UBI), Universidade da Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-901, Covilhã, Portugal*

^b*HASLab/INESC Tec, Portugal*

^c*Universidade do Minho, Portugal*

^d*Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto & CISUC, Portugal*

Abstract

This paper compares a large set of programming languages regarding their efficiency, including from an energetic point-of-view. Indeed, we seek to establish and analyze different rankings for programming languages based on their energy efficiency. The goal of being able to rank programming languages based on their energy efficiency is both recent, and certainly deserves further studies. We have taken rigorous and strict solutions to 10 well defined programming problems, expressed in (up to) 27 programming languages, from the well known Computer Language Benchmark Game repository. This repository aims to compare programming languages based on a strict set of implementation rules and configurations for each benchmarking problem. We have also built a framework to automatically, and systematically, run, measure and compare the energy, time, and memory efficiency of such solutions. Ultimately, it is based on such comparisons that we propose a series of efficiency rankings, based on single and multiple criteria.

Our results show interesting findings, such as how slower/faster languages can consume less/more energy, and how memory usage influences energy consumption. We also present a simple way to use our results to provide software engi-

Email addresses: rui.a.pereira@inesctec.pt (Rui Pereira), marco.l.couto@inesctec.pt (Marco Couto), francisco.j.ribeiro@inesctec.pt (Francisco Ribeiro), rui.a.rua@inesctec.pt (Rui Rua), jacome@di.uminho.pt (Jácome Cunha), jpf@dei.ucp.pt (João Paulo Fernandes), jsa@di.uminho.pt (João Saraiva)

Preprint submitted to Elsevier

January 4, 2021

<https://haslab.github.io/SAFER/scp21.pdf>

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58



MEETUP PYTHON-RENNES

meetup

Créer un groupe
- 30 % de
réduction !



PRO

Essayer gratuitement



Contacts



Messages



Notifications



python
RENNES



Python Rennes

Rennes, France

339 membres · Groupe public

Organisé par **Nicolas Ledez** and 1 other

Partager:

À propos Événements Membres Photos

Vous êtes membre

De quoi s'agit-il

Tu débutes le fourchelangue ou tu le pratiques depuis longtemps ? Retrouve-nous pour échanger autour :

[En savoir plus](#)

Événements passés (10)

[Tout voir](#)

VEN. 23 FÉVR. 2024, 18:30 CET

Python : il buono, il brutto, il cattivo

L'événement est passé

Organizers

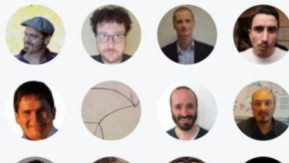


Nicolas Ledez and 1 other

[Message](#)

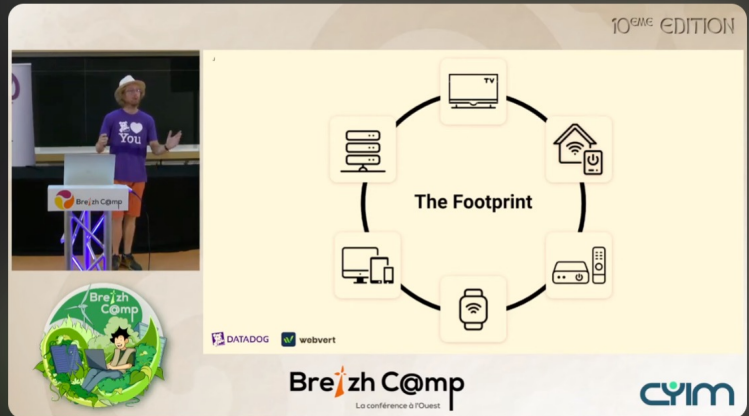
Membres (339)

[Tout voir](#)



YouTube

FR



L'eco-conception, c'est bien, mais si on parlait un peu du backend ? (Jérémy Drouet, Youen Chéné)



BreizhCamp
5.06K subscribers

Subscribe

23



Share



1K views 1 year ago

A l'heure actuelle, l'éco-conception des applications web est un domaine qui monte.

Vous pouvez faire des tests de performance de vos pages web, savoir combien

<https://www.meetup.com/fr-FR/python-rennes/>

<https://www.youtube.com/watch?v=gE6HUsmh554>

PERFORMANCE MATTERS!

Performance matters :

- for our users
- for (artificial) intelligence
- for the planet

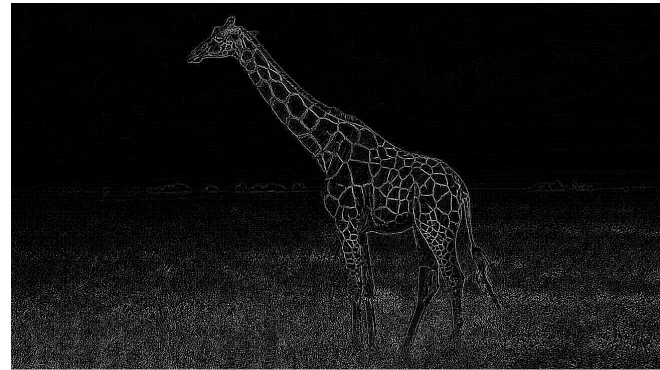
**IT'S
DEMO TIME!**

Laplacian filter
(edge detection)

EDGE DETECTION



$$\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

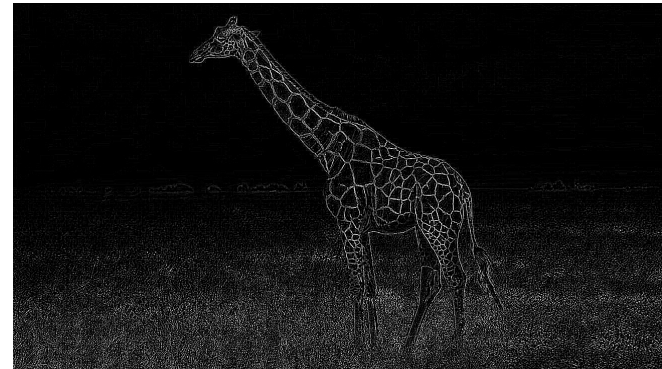


EDGE DETECTION



Convolve

kernel

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$


HOP HOP HOP

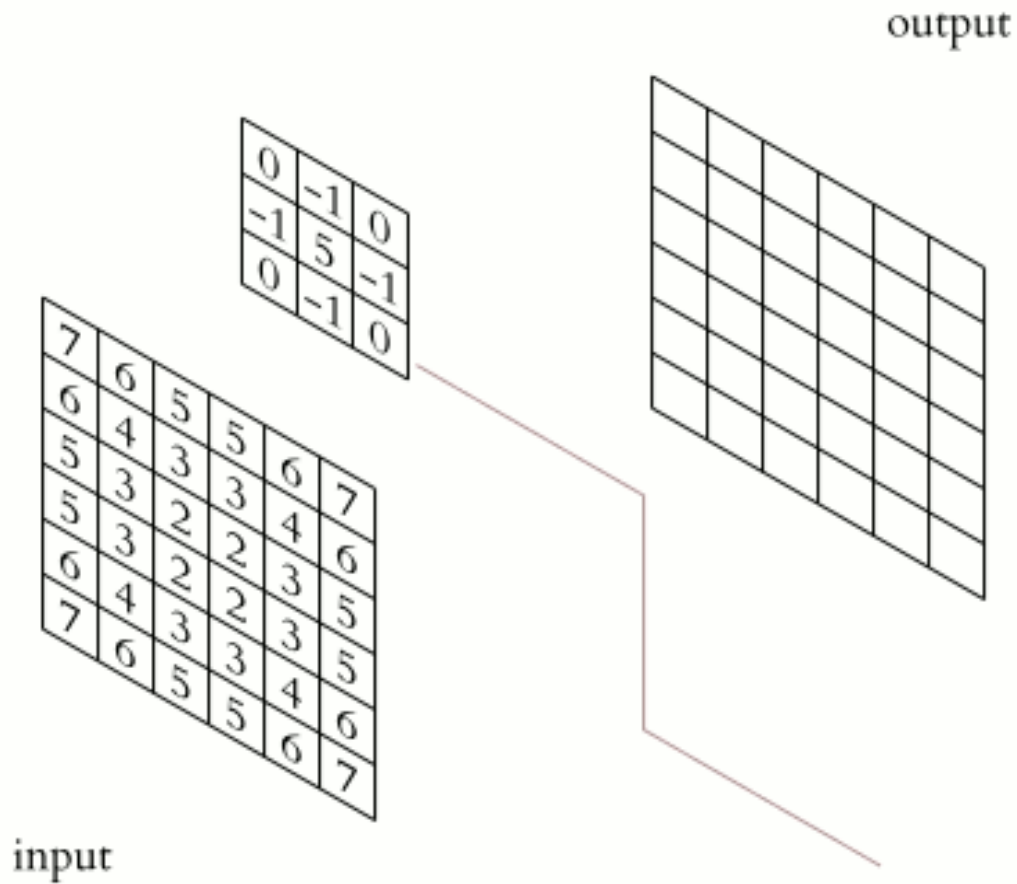
7	6	5	5	6	7
6	4	3	3	4	6
5	3	2	2	3	5
5	3	2	2	3	5
6	4	3	3	4	6
7	6	5	5	6	7

0	-1	0
-1	5	-1
0	-1	0

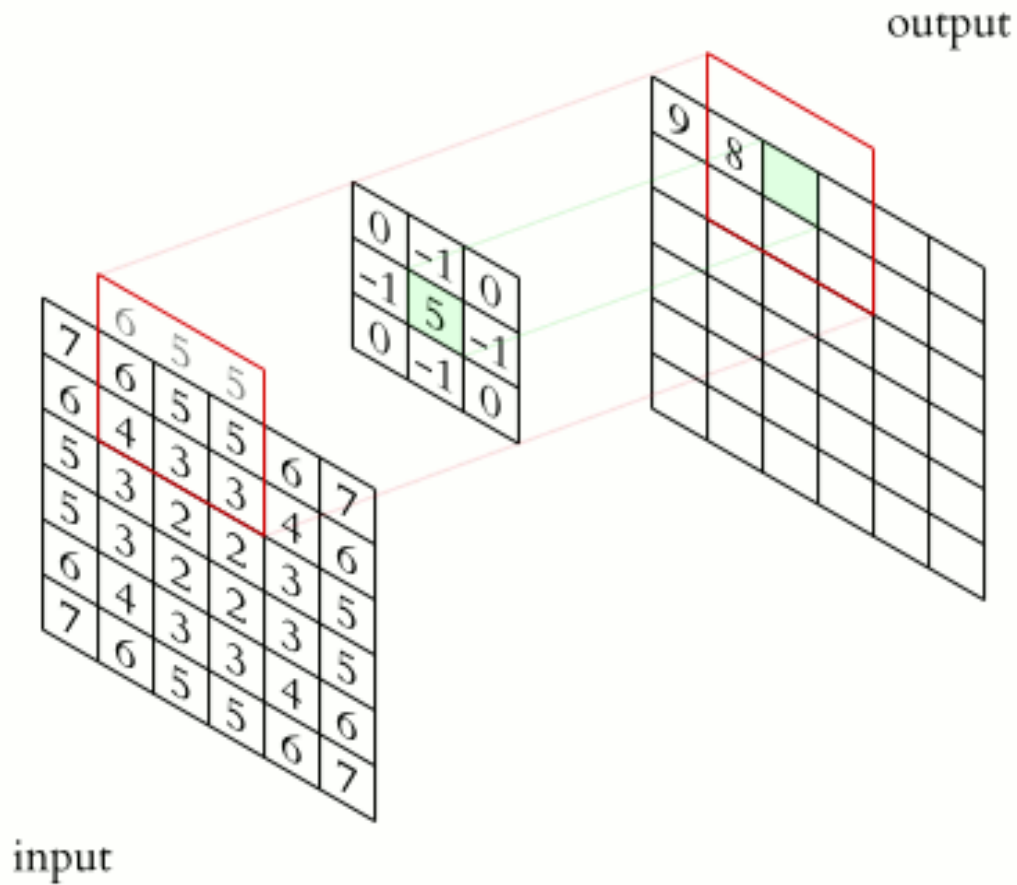
output

input

EDGE DETECTION



EDGE DETECTION



RECAP

naïve version : 500 ms

numpy mul : 250 ms x 2

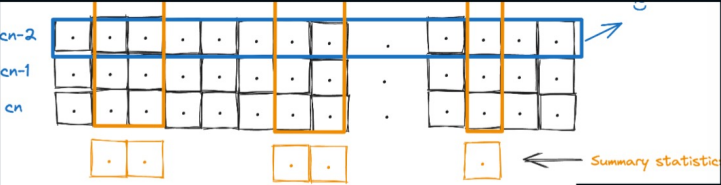
numpy+numba : 50 ms x 10

opencv : 0.5 ms x 1000

And now in mojo ?

AND NOW IN MOJO!

Modular Get Started Sign In Menu




en-2
en-1
en

Summary statistic

BLOG / DEVELOPER

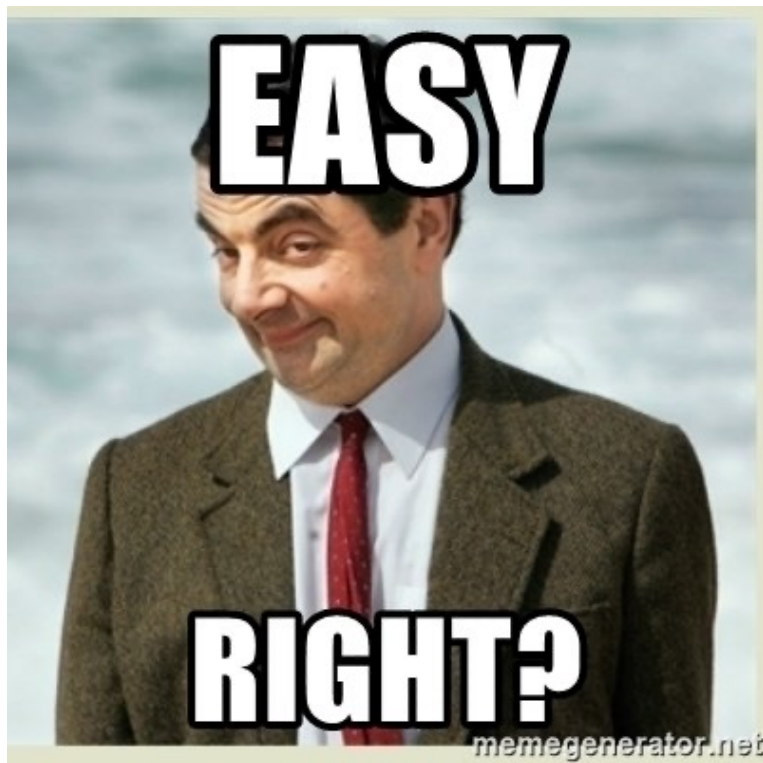
Implementing NumPy style matrix slicing in Mojo 🔥

November 20, 2023



Shashank Prasanna
AI Developer Advocate

<https://www.modular.com/blog/implementing-numpy-style-matrix-slicing-in-mojo>



POWER! UNLIMITED POWER!



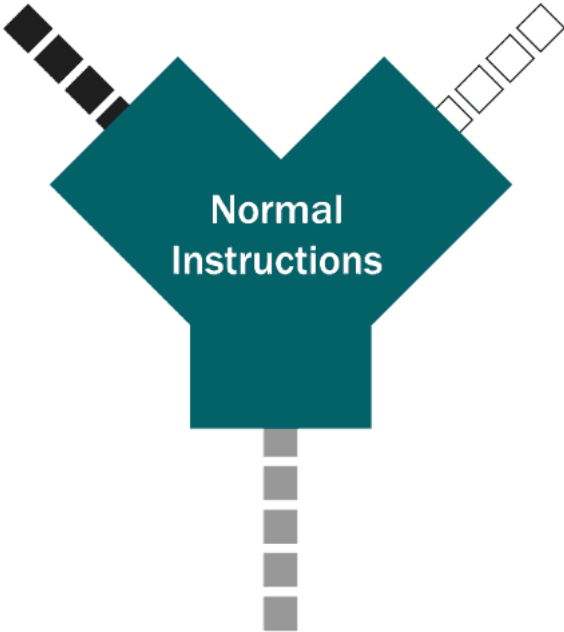
Credits to Georges L. and Corentin R.

“The greatest teacher, failure is”

**IT'S
DEMO TIME!**

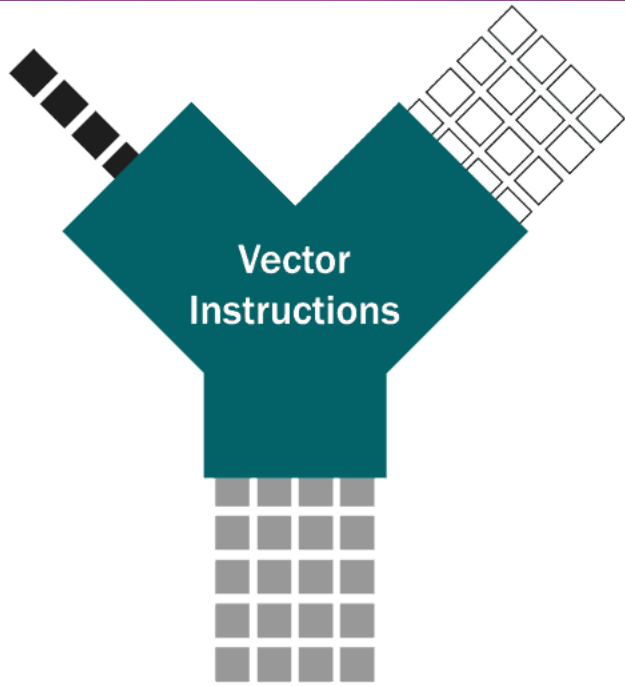
Let's optimize !

SISD ARCHITECTURE



- Instructions
- Data
- Results

SIMD ARCHITECTURE



- Instructions
- Data
- Results

ALGORITHM VECTORIZATION

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:  
  var result = Matrix[DType.float32](img.height, img.width)  
  # Loop through each pixel in the image  
  # But skip the outer edges of the image  
  for y in range(1, img.height-1):  
    for x in range(1, img.width-1):  
      # For each pixel, compute the product elements wise  
      var acc: Float32 = 0  
      for k in range(3) :  
        for l in range(3):  
          acc += img[y-1+k, x-1+l] * kernel[k, l]  
      # Normalize the result  
      result[y, x] = min(255, max(0, acc))  
  return result
```

ALGORITHM VECTORIZATION

```
alias nelts = simdwidthof[DType.float32]()
```

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:  
  var result = Matrix[DType.float32](img.height, img.width)  
  # Loop through each pixel in the image  
  # But skip the outer edges of the image  
  for y in range(1, img.height-1):  
    for x in range(1, img.width-1):  
      # For each pixel, compute the product elements wise  
      var acc: Float32 = 0  
      for k in range(3) :  
        for l in range(3):  
          acc += img[y-1+k, x-1+l] * kernel[k, l]  
      # Normalize the result  
      result[y, x] = min(255, max(0, acc))  
  return result
```

ALGORITHM VECTORIZATION

```
alias nelts = simdwidthof[DType.float32]()
```

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:  
  var result = Matrix[DType.float32](img.height, img.width)  
  # Loop through each pixel in the image  
  # But skip the outer edges of the image  
  for y in range(1, img.height-1)  
    for x in range(1, img.width-1, nelts):  
      # For each pixel, compute the product elements wise  
      var acc: Float32 = 0  
      for k in range(3) :  
        for l in range(3):  
          acc += img[y-1+k, x-1+l] * kernel[k, l]  
      # Normalize the result  
      result[y, x] = min(255, max(0, acc))  
  return result
```

ALGORITHM VECTORIZATION

```
alias nelts = simdwidthof[DType.float32]()
```

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:  
  var result = Matrix[DType.float32](img.height, img.width)  
  # Loop through each pixel in the image  
  # But skip the outer edges of the image  
  for y in range(1, img.height-1):  
    for x in range(1, img.width-1, nelts):  
      # For each pixel, compute the product elements wise  
      # var acc: Float32 = 0  
      var acc: SIMD[DType.float32, nelts] = 0  
      for l in range(3):  
        acc += img[y-1+k, x-1+l] * kernel[k, l]  
      # Normalize the result  
      result[y, x] = min(255, max(0, acc))  
  return result
```

ALGORITHM VECTORIZATION

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:  
  var result = Matrix[DType.float32](img.height, img.width)  
  # Loop through each pixel in the image  
  # But skip the outer edges of the image  
  for y in range(1, img.height-1):  
    for x in range(1, img.width-1, nelts):  
      # For each pixel, compute the product elements wise  
      # var acc: Float32 = 0  
      var acc: SIMD[DType.float32, nelts] = 0  
      for k in range(3) :  
        for l in range(3):  
          # acc += img[y-1+k, x-1+l] * kernel[k, l]  
          acc += img.simd_load[nelts](y-1+k, x-1+l) * kernel[k, l]  
          # result[y, x] = min(255, max(0, acc))  
          result.simd_store[nelts](y, x, min(255, max(0, acc)))  
    return result
```

ALGORITHM VECTORIZATION

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:
    var result = Matrix[DType.float32](img.height, img.width)
    # Loop through each pixel in the image
    # But skip the outer edges of the image
    for y in range(1, img.height-1):
        for x in range(1, img.width-1, nelts):
            # For each pixel, compute the product elements wise
            # var acc: Float32 = 0
            var acc: SIMD[DType.float32,nelts] = 0
            for k in range(3) :
                for l in range(3):
                    # acc += img[y-1+k, x-1+l] * kernel[k, l]
                    acc += img.simd_load[nelts](y-1+k, x-1+l) * kernel[k, l]
            # Normalize the result
            # result[y, x] = min(255, max(0, acc))
            result.simd_store[nelts](y, x, min(255, max(0, acc)))
        # Handle remaining elements with scalars.
        for n in range(nelts * (img.width-1 // nelts), img.width-1) :
            var acc: Float32 = 0
            for k in range(3) :
                for l in range(3):
                    acc += img[y-1+k, n-1+l] * kernel[k, l]
            result[y, n] = min(255, max(0, acc))
```


ALGORITHM VECTORIZATION

```
alias nelts = simdwidthof[DType.float32]()
```

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:
```

```
  var result = Matrix[DType.float32](img.height, img.width)
```

```
  # Loop through each pixel in the image
```

```
  # But skip the outer edges of the image
```

```
  for y in range(1, img.height-1):
```

```
    @parameter
```

```
    fn dot[nelts: Int](x: Int):
```

```
      # For each pixel, compute the product elements wise
```

```
      var acc: SIMD[DType.float32, nelts] = 0
```

```
      for k in range(3) :
```

```
        for l in range(3):
```

```
          acc += img.simd_load[nelts](y-1+k, x-1+l) * kernel[k, l]
```

```
      # Normalize the result
```

```
      return simd_store[nelts](x, acc) * max(0, acc))
```

```
    vectorize[dot, nelts](size=img.width-1)
```

```
  return result
```

ALGORITHM VECTORIZATION

```
alias nelts = simdwidthof[DType.float32]()
```

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:  
  var result = Matrix[DType.float32](img.height, img.width)  
  # Loop through each pixel in the image  
  # But skip the outer edges of the image  
  for y in range(1, img.height-1):  
    @parameter  
    fn dot[nelts: Int](x: Int):  
      # For each pixel, compute the product elements wise  
      var acc: SIMD[DType.float32,nelts] = 0  
      for k in range(3) :  
        for l in range(3):  
          acc += img.simd_load[nelts](y-1+k, x-1+l) * kernel[k, l]  
      # Normalize the result  
      result.simd_store[nelts](y, x, min(255, max(0, acc)))  
    vectorize[dot, nelts](size=img.width-1)  
  
  return result
```

ALGORITHM VECTORIZATION

```
alias nelts = simdwidthof[DType.float32]()
```

```
fn naive(img: Matrix[DType.float32], kernel: Matrix[DType.float32]) -> Matrix[DType.float32]:  
  var result = Matrix[DType.float32](img.height, img.width)  
  # Loop through each pixel in the image  
  # But skip the outer edges of the image  
  for y in range(1, img.height-1):  
    @parameter  
    fn dot[nelts: Int](x: Int):  
      # For each pixel, compute the product elements wise  
      var acc: SIMD[DType.float32,nelts] = 0  
      for k in range(3) :  
        for l in range(3):  
          acc += img.simd_load[nelts](y-1+k, x+l) * kernel[k, l]  
      # Normalize the result  
      result.simd_store[nelts](y, x+1, min(255, max(0, acc)))  
    vectorize[dot, nelts](size=img.width-2)  
  
  return result
```

RECAP

- Far from stable
- Compilation AOT or JIT
- Python friendly but not Python
- Dynamic Python vs Static Mojo
- Python interoperability
- Predictable behavior with semantic ownership
- Low level optimization
- Blazingly fast

Mojo 🔥
The next AI Language?

CONCLUSION

- Python is not yet dead !
But he moves slowly
- This is a great team !
Will they be able to deploy their platform strategy ?
- Will they be able to unite a
community?
To be open-source or not to be

MERCI!



Jean-Luc Tromparent

Principal Engineer @ **hellowork**
group learn, work,
move.

<https://linkedin.com/in/jltromparent>

https://github.com/jiel/laplacian_filters_benchmark

<https://noti.st/jlt/qheMOM>



👉 I need your feedback 👈