A brief history and wild speculation about the future of

julia

Jeff Bezanson
Alan Edelman
Stefan Karpinski
Viral B. Shah

ancient history

Subject: Julia

From: Viral Shah <vshah@interactivesupercomputing.com>
To: Jeff Bezanson <jbezanson@interactivesupercomputing.com>
Cc: Stefan Karpinski <sgk@cs.ucsb.edu>
Date: Thu, Aug 20, 2009 at 12:08 AM

Hey,

Do you have julia up on a website somewhere, or did you take it down? I was just mentioning it to a friend of mine - Stefan Karpinski, and your thoughts of doing a clean language effort. He has thought hard about languages for scientific computing.

-viral

Subject: Julia

From: Viral Shah <vshah@interactivesupercomputing.com>
To: Jeff Bezanson <jbezanson@interactivesupercomputing.com>
Cc: Stefan Karpinski <sgk@cs.ucsb.edu>
Date: Thu, Aug 20, 2009 at 12:08 AM

1 of 20,961

Hey,

Do you have julia up on a website somewhere, or did you take it down? I was just mentioning it to a friend of mine - Stefan Karpinski, and your thoughts of doing a clean language effort. He has thought hard about languages for scientific computing.

-viral

From: **Stefan Karpinski** <sgk@cs.ucsb.edu>
Date: Thu, Aug 20, 2009 at 1:40 AM
To: Viral Shah <vshah@interactivesupercomputing.com>
Cc: Jeff Bezanson <jbezanson@interactivesupercomputing.com>

I'm going to rant now so that you know how I feel about this and why I think that scientific computing desperately needs a new programming system.

My basic take on the current state of affairs in scientific computing language issue is that I'm just sick of having to work in 5-6 different languages all the time in order to any serious data analysis. There's lots of great tools, but any one of them can only do somewhere between 20-40% of what I need to do in total.

From: **Stefan Karpinski** <sgk@cs.ucsb.edu>
Date: Thu, Aug 20, 2009 at 1:40 AM
To: Viral Shah <vshah@interactivesupercomputing.com>
Cc: Jeff Bezanson <jbezanson@interactivesupercomputing.com>

I'm going to rant now so that you know how I feel about this and why I think that scientific computing desperately needs a new programming system.

My basic take on the current state of affairs in scientific computing language issue is that I'm just sick of having to work in 5-6 different languages all the time in order to any serious data analysis. There's lots of great tools, but any one of them can only do somewhere between 20-40% of what I need to do in total.

From: **Jeff Bezanson** <jbezanson@interactivesupercomputing.com>
Date: Thu, Aug 20, 2009 at 8:04 PM
To: Stefan Karpinski <sgk@cs.ucsb.edu>
Cc: Viral Shah <vshah@interactivesupercomputing.com>

Nice to meet you Stefan. Good rant! Glad to know somebody else has thoughts like this; I was starting to think I was crazy.

…

Right now I'm thinking about starting a project to make the fast, simple, and clean open system for scientific computing that the world ought to have. I don't know if it would be based on Julia; I wouldn't mind starting again (that way people wouldn't have to grapple with my code base!!) As I was telling Viral I'd like to take a month or so to take the first steps and see if things look promising. Anybody know of any good tools for generating native code with Scheme?

From: **Jeff Bezanson** <jbezanson@interactivesupercomputing.com>
Date: Thu, Aug 20, 2009 at 8:04 PM
To: Stefan Karpinski <sgk@cs.ucsb.edu>
Cc: Viral Shah <vshah@interactivesupercomputing.com>

Nice to meet you Stefan. Good rant! Glad to know somebody else has thoughts like this; I was starting to think I was crazy.

...

Right now I'm thinking about starting a project to make the fast, simple, and clean open system for scientific computing that the world ought to have. I don't know if it would be based on Julia; I wouldn't mind starting again (that way people wouldn't have to grapple with my code base!!) As I was telling Viral I'd like to take a month or so to take the first steps and see if things look promising. Anybody know of any good tools for generating native code with Scheme?

Subject: object orientation

From: Jeff Bezanson <jeffbez@comcast.net>
Date: Fri, Sep 11, 2009 at 4:05 AM
To: <julia-math@googlegroups.com>

My point of view is that I don't want the language based on single dispatch (2.addTo(3)), class and interface declarations add too much verbosity in most languages, multiple inheritance is too complicated, and inheritance itself is highly suspect and good uses of it are much rarer than people think.

Subject: object orientation

From: Jeff Bezanson <jeffbez@comcast.net>
Date: Fri, Sep 11, 2009 at 4:05 AM
To: <julia-math@googlegroups.com>

My point of view is that I don't want the language based on single dispatch (2.addTo(3)), class and interface declarations add too much verbosity in most languages, multiple inheritance is too complicated, and inheritance itself is highly suspect and good uses of it are much rarer than people think.

From: **Stefan Karpinski** <stefan.karpinski@gmail.com>
Date: Fri, Sep 11, 2009 at 6:40 AM
Subject: Re: object orientation
To: <julia-math@googlegroups.com>

Single vs. multiple dispatch. In general, I feel that multiple dispatch is overkill and that single dispatch is a good thing for simplicity and understandability. In the case of arithmetic and mathematical functions, however, single dispatch fails miserably. Since that is by far the vast majority of what we're dealing with, I think I'm for multiple dispatch in Julia.

From: **Stefan Karpinski** <stefan.karpinski@gmail.com>
Date: Fri, Sep 11, 2009 at 6:40 AM
Subject: Re: object orientation
To: <julia-math@googlegroups.com>

Single vs. multiple dispatch. In general, I feel that multiple dispatch is overkill and that single dispatch is a good thing for simplicity and understandability. In the case of arithmetic and mathematical functions, however, single dispatch fails miserably. Since that is by far the vast majority of what we're dealing with, I think I'm for multiple dispatch in Julia.

From: **Stefan Karpinski** <stefan.karpinski@gmail.com>
Date: Fri, Sep 11, 2009 at 6:40 AM
Subject: Re: object orientation
To: <julia-math@googlegroups.com>

Single vs. multiple dispatch. In general, I feel that multiple dispatch is overkill and that single dispatch is a good thing for simplicity and understandability. In the case of arithmetic and mathematical functions, however, single dispatch fails miserably. Since that is by far the vast majority of what we're dealing with, I think I'm for multiple dispatch in Julia.

From: **Stefan Karpinski** <stefan.karpinski@gmail.com>
Date: Fri, Sep 11, 2009 at 6:40 AM
Subject: Re: object orientation
To: <julia-math@googlegroups.com>

Single vs. multiple dispatch. In general, I feel that multiple dispatch is overkill and that single dispatch is a good thing for simplicity and understandability. In the case of arithmetic and mathematical functions, however, single dispatch fails miserably. Since that is by far the vast majority of what we're dealing with, I think I'm for multiple dispatch in Julia.

From: **Viral Shah** <viral@mayin.org>
Date: Fri, Sep 11, 2009 at 7:06 AM
Subject: Re: object orientation
To: <julia-math@googlegroups.com>

Yes, I think multiple dispatch would be good to have, if nothing else to simplify the implementation of the runtime system. BTW, if you do have named parameters, how does single dispatch work?

From: **Viral Shah** <viral@mayin.org>

Date: Fri, Sep 11, 2009 at 7:06 AM

Subject: Re: object orientation

To: <julia-math@googlegroups.com>

Yes, I think multiple dispatch would be good to have, if nothing else to simplify the implementation of the runtime system. BTW, if you do have named parameters, how does single dispatch work?

From: **Jeff Bezanson** <jeffbez@comcast.net>
Date: Sat, Sep 12, 2009 at 1:26 AM
To: <julia-math@googlegroups.com>

What if we allowed inheritance of all types, but methods were compiled for every exact type signature with respect to builtin types? In other words, if we've already compiled f(int32), we don't invoke that specialization on a my_int argument but rather compile a new version just for my_int. This way type inference can assume any inferred builtin types are exact so it never needs to do dynamic dispatch on them. The only overhead is lots of extra compilation, but that's a reasonable price to ask for something like this.

From: **Jeff Bezanson** <jeffbez@comcast.net>
Date: Sat, Sep 12, 2009 at 1:26 AM
To: <julia-math@googlegroups.com>

What if we allowed inheritance of all types, but methods were compiled for every exact type signature with respect to builtin types? In other words, if we've already compiled f(int32), we don't invoke that specialization on a my_int argument but rather compile a new version just for my_int. This way type inference can assume any inferred builtin types are exact so it never needs to do dynamic dispatch on them. The only overhead is lots of extra compilation, but that's a reasonable price to ask for something like this.

From: **Jeff Bezanson** <jeffbez@comcast.net>

Date: Sat, Sep 12, 2009 at 1:26 AM

To: <julia-math@googlegroups.com>

What if we allowed inheritance of all types, but methods were compiled for every exact type signature with respect to builtin types? In other words, if we've already compiled f(int32), we don't invoke that specialization on a my_int argument but rather compile a new version just for my_int. This way type inference can assume any inferred builtin types are exact so it never needs to do dynamic dispatch on them. The only overhead is lots of extra compilation, but that's a reasonable price to ask for something like this.

time passes...

the jeff, the jeff

julia ×

Stefan Karpinski <stefar Thu, Apr 1, 2010, 4:50 AM

to Julia

the jeff is on fire.

Reply

Forward

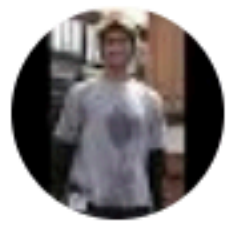
what was he up to?

what was he up to?

demo...

blog post julia × julia/dev ×

blog post julia × julia/dev ×



Viral Shah viral@mayin.c Sat, Feb 18, 2012, 3:41 AM

to julia-dev

I just posted our first blog post on reddit:

http://www.reddit.com/r/ProgrammingLanguages/comments/putpq/why_we_created_julia_a_new_language_and_a_fresh/

•••

blog post

julia · julia/dev

Viral Shah viral@mayin.c Sat, Feb 18, 2012, 3:41 AM

to julia-dev

I just posted our first blog post on reddit:

http://www.reddit.com/r/ProgrammingLanguages/comments/putpq/why_we_created_julia_a_new_language_and_a_fresh/

Stefan Karpinski <stefa Sat, Feb 18, 2012, 5:50 AM

to julia-dev

Um. Perhaps telling everyone else first would have been in order.

blog post julia julia/dev

Viral Shah viral@mayin.c Sat, Feb 18, 2012, 3:41 AM

to julia-dev

I just posted our first blog post on reddit:

http://www.reddit.com/r/ProgrammingLanguages/comments/putpq/why_we_created_julia_a_new_language_and_a_fresh/

Stefan Karpinski <stefa Sat, Feb 18, 2012, 5:50 AM

to julia-dev

Um. Perhaps telling everyone else first would have been in order.

Jeff Bezanson <jeff.bez Sat, Feb 18, 2012, 6:02 AM

to julia-dev

Yes, seems to me this should be up to the author of the post. Were we even finished editing?

modern history

Feb 2012: Julia 0.0

Feb 2012: Julia 0.0

LLVM code gen

Feb 2012: Julia 0.0

LLVM code gen

ccall

Feb 2012: Julia 0.0

LLVM code gen

ccall

save system images

Feb 2012: Julia 0.0

LLVM code gen

ccall

save system images

remotecall

Feb 2012: Julia 0.0

LLVM code gen

ccall

save system images

remotecall

a manual

Keno Fischer kenof@stanf Tue, May 1, 2012, 1:47 AM

to julia-dev

Hello everybody,

I think Julia has reached a size where it might be reasonable to start thinking about having some sort of CI testing and automated binary package building in place (especially once we support Windows builds from master). I am writing this email to collect some ideas regarding and discuss some various different approaches to this.

Keno Fischer kenof@stanf Tue, May 1, 2012, 1:47 AM

to julia-dev

Hello everybody,

I think Julia has reached a size where it might be reasonable to start thinking about having some sort of CI testing and automated binary package building in place (especially once we support Windows builds from master). I am writing this email to collect some ideas regarding and discuss some various different approaches to this.

Feb 2013: Julia 0.1

namespaces/modules

libuv & Windows port

package manager

cfunction

Feb 2013: Julia 0.1

namespaces/modules

Keno Fischer

libuv & Windows port

package manager

cfunction

Feb 2013: Julia 0.1

namespaces/modules

Keno Fischer

libuv & Windows port

Jameson Nash

package manager

cfunction

Feb 2013: Julia 0.1

namespaces/modules

Keno Fischer libuv & Windows port **Jameson Nash**

package manager

cfunction **Jameson**

Nov 2013: Julia 0.2

immutable struct types

keyword arguments

optional arguments

profiler

Nov 2013: Julia 0.2

immutable struct types

Mike Nolta

keyword arguments

optional arguments

profiler

Nov 2013: Julia 0.2

immutable struct types

Mike Nolta keyword arguments

optional arguments

Tim Holy profiler

Aug 2014: Julia 0.3

native REPL

value-based numerical hashing

quality, stability & longevity

Aug 2014: Julia 0.3

Keno

native REPL

value-based numerical hashing

quality, stability & longevity

Aug 2014: Julia 0.3

Keno native REPL

value-based numerical hashing

Tony Kelman quality, stability & longevity

Oct 2015: Julia 0.4

tuples with struct layout

generated functions

documentation system

precompiled modules

generational GC

Oct 2015: Julia 0.4

tuples with struct layout

Keno &
Tim Holy

generated functions

documentation system

precompiled modules

generational GC

Oct 2015: Julia 0.4

tuples with struct layout

Keno & Tim Holy

generated functions

documentation system

Michael Hatherly

precompiled modules

generational GC

Oct 2015: Julia 0.4

tuples with struct layout

Keno & Tim Holy

generated functions

documentation system

Michael Hatherly

Jameson

precompiled modules

generational GC

Oct 2015: Julia 0.4

tuples with struct layout

Keno & Tim Holy

generated functions

documentation system

Michael Hatherly

Jameson

precompiled modules

generational GC

Yichao Yu & Oscar Blumberg

Sep 2016: Julia 0.5

great function overhaul

generator expressions

fused broadcasting syntax

85% test coverage

Sep 2016: Julia 0.5

great function overhaul

generator expressions

Steve Johnson fused broadcasting syntax

85% test coverage

Sep 2016: Julia 0.5

great function overhaul

generator expressions

Steve Johnson

fused broadcasting syntax

85% test coverage

Katie Hyatt

Jun 2017: Julia 0.6

the infamous #265

triangular dispatch

deleting weird string types

we took vector transposes very seriously

Jun 2017: Julia 0.6

Jameson the infamous #265

triangular dispatch

deleting weird string types

we took vector transposes
very seriously

Jun 2017: Julia 0.6

Jameson

the infamous #265

triangular dispatch

deleting weird string types

we took vector transposes very seriously

Jiahao & Andy Ferris

the present

Aug 2018: Julia 1.0

Pkg3

new iteration protocol

new optimizer

named tuples

strings that can hold arbitrary data

Aug 2018: Julia 1.0

Pkg3

Keno new iteration protocol

new optimizer

named tuples

strings that can hold arbitrary data

Aug 2018: Julia 1.0

Pkg3

Keno new iteration protocol

Keno new optimizer

named tuples

strings that can hold arbitrary data

Aug 2018: Julia 1.0

fast keyword arguments

find/search APIs

fast unions and arrays

`nothing` and `missing`

Aug 2018: Julia 1.0

fast keyword arguments

Milan Bouchet-Valat

find/search APIs

fast unions and arrays

`nothing` and `missing`

Aug 2018: Julia 1.0

fast keyword arguments

Milan Bouchet-Valat

find/search APIs

fast unions and arrays

Jacob Quinn & Jameson

`nothing` and `missing`

Aug 2018: Julia 1.0

fast keyword arguments

Milan Bouchet-Valat

find/search APIs

fast unions and arrays

Jacob Quinn & Jameson

Jacob & Milan

`nothing` and `missing`

Aug 2018: Julia 1.0

we took matrix transposes seriously

Aug 2018: Julia 1.0

we took matrix transposes seriously

Andreas, Jiahao,
& Andy Ferris

the future

**what can we do
with all this power?**

what <u>can't</u> we do?

solving grand challenges

http://www.engineeringchallenges.org/

What is the connection?

Engineering tools for scientific discovery

multiple dispatch	personalized medicine
generic programming	personalized education
code specialization	economical solar energy
native code	improve urban infrastructure
parallelism	access to clean water

Cataloging the universe

Cataloging the universe

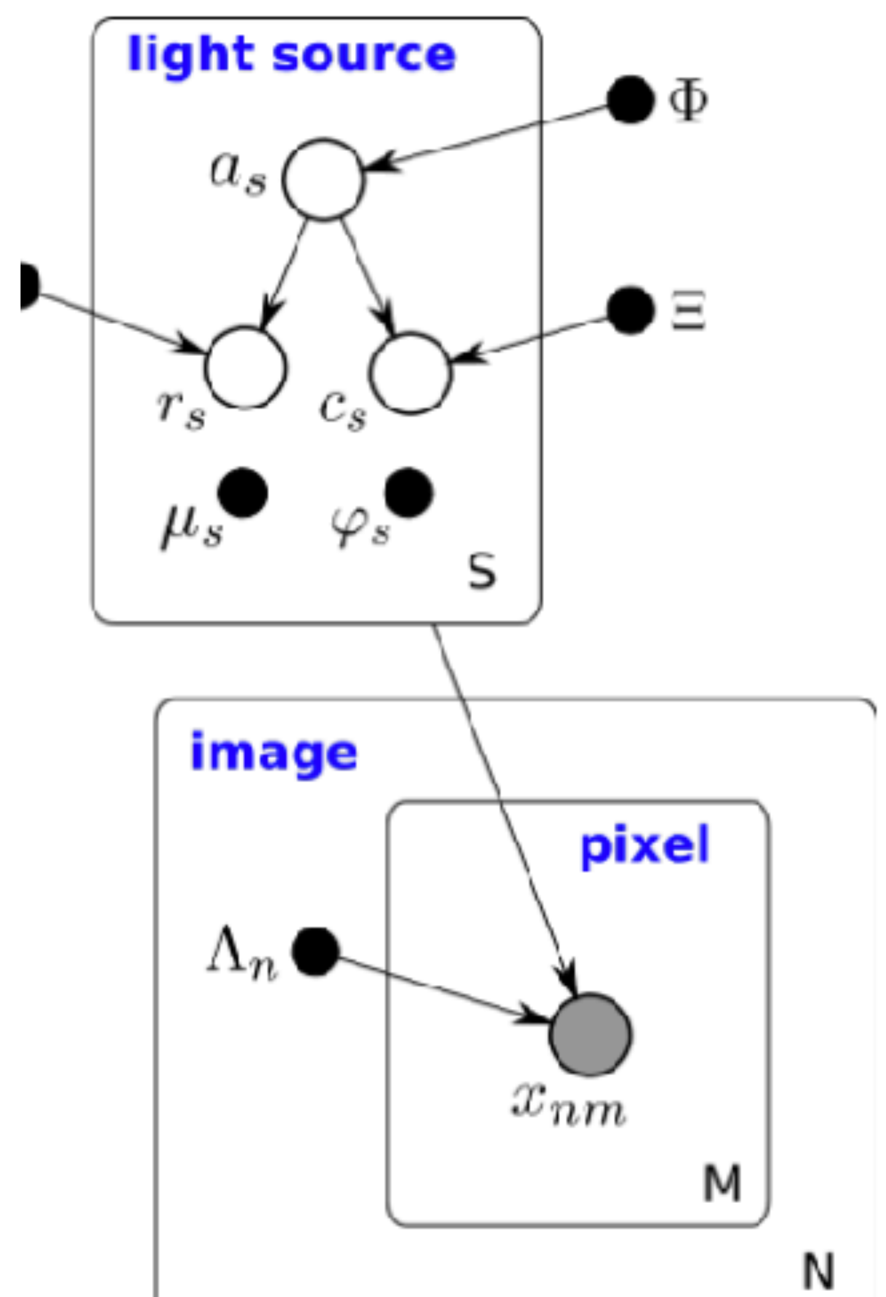
Most light sources are near the detection limit.

Cataloging the universe

Most light sources are near the detection limit.

Cataloging the universe

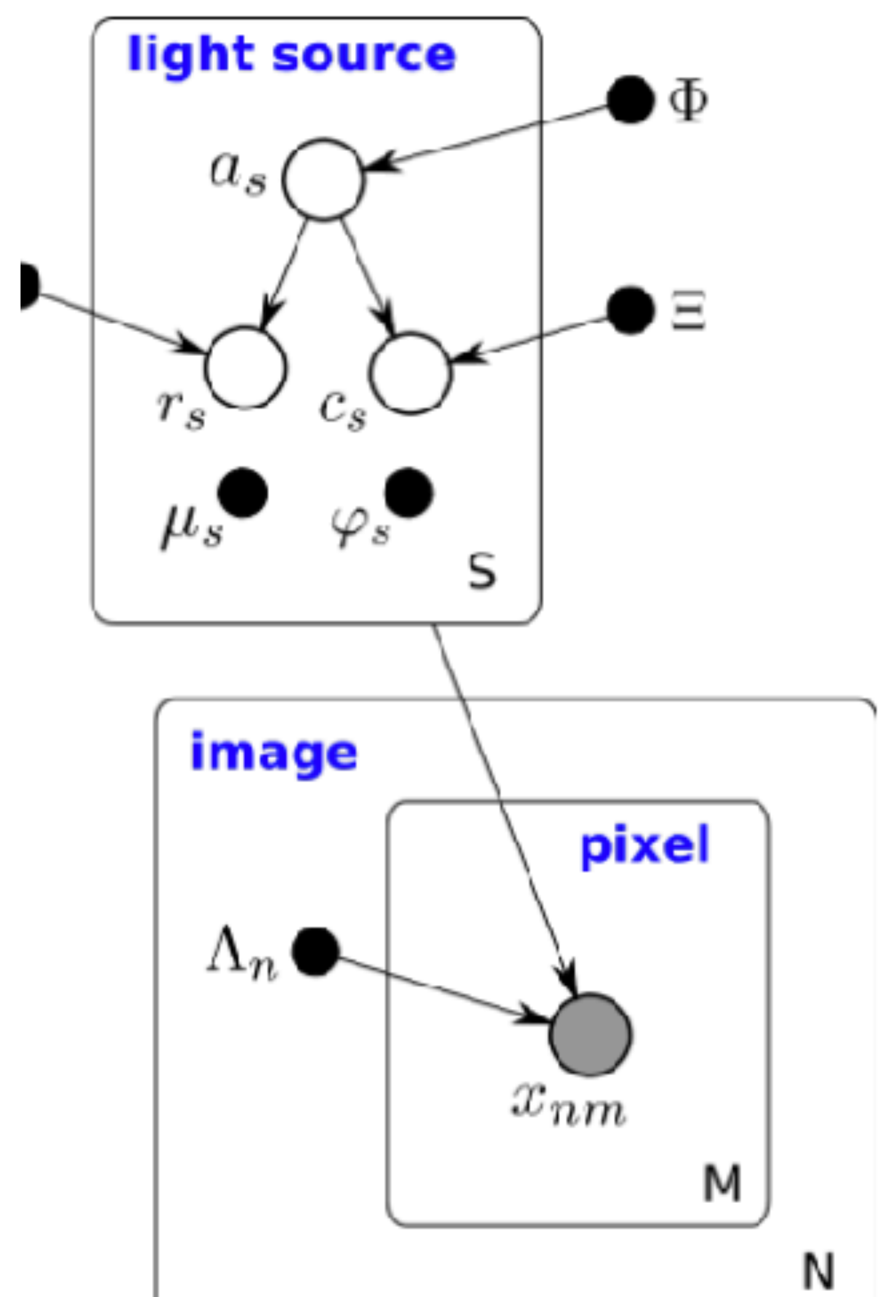
Most light sources are near the detection limit.



Cataloging the universe

Intel Knights Landing, SIMD, Multi-threading

Most light sources are near the detection limit.

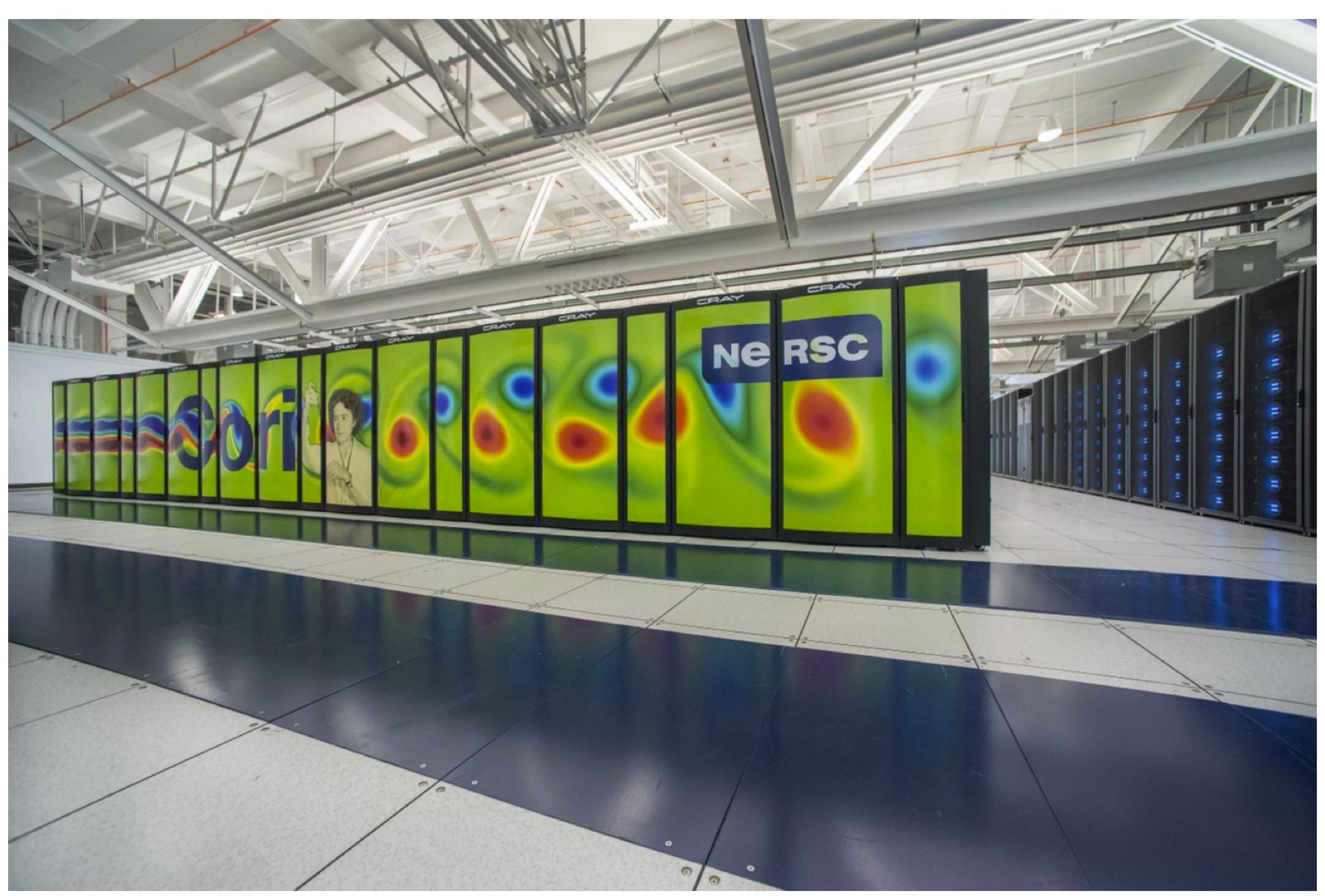


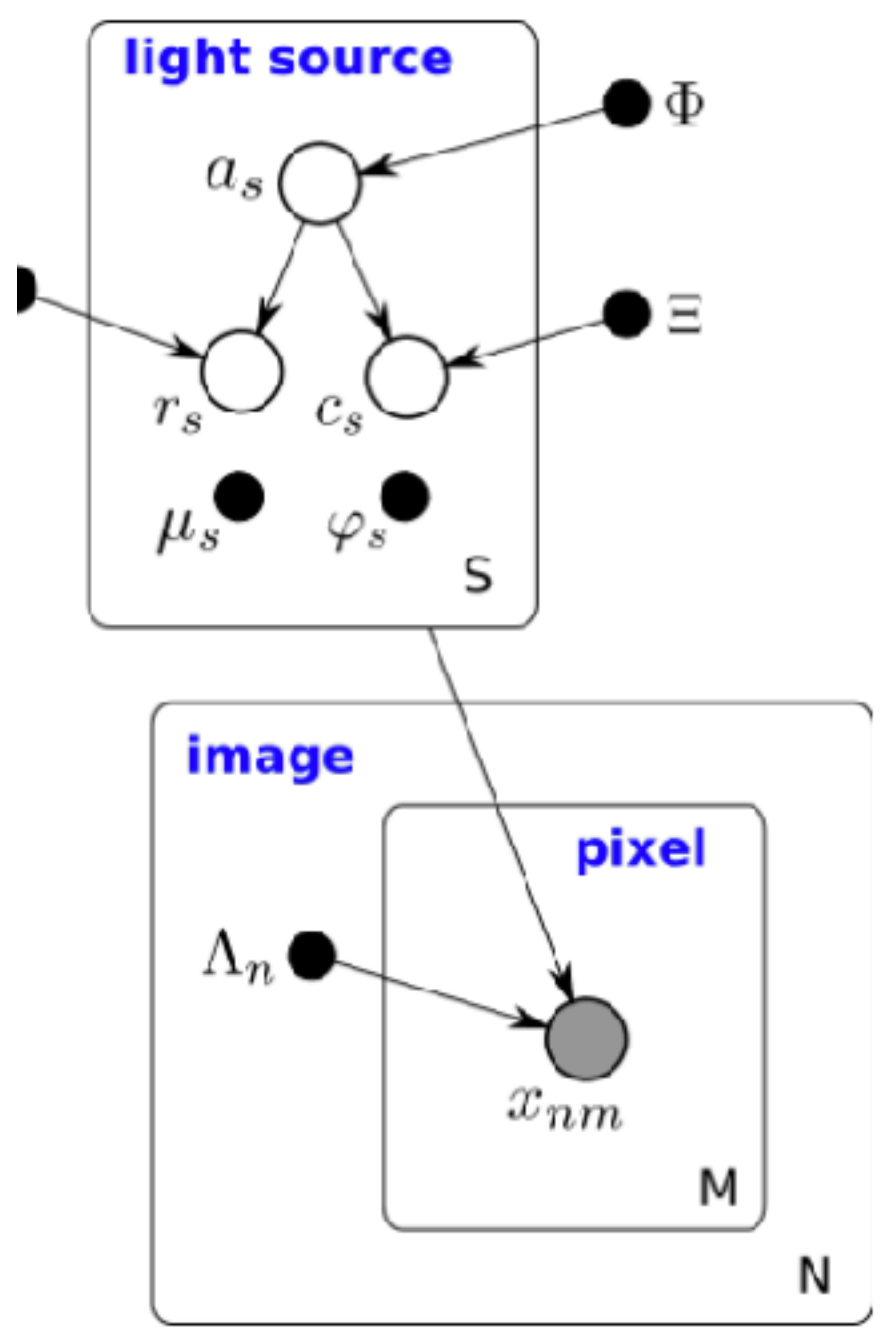
Cataloging the universe

Intel Knights Landing, SIMD, Multi-threading

↓

Most light sources are near the detection limit.





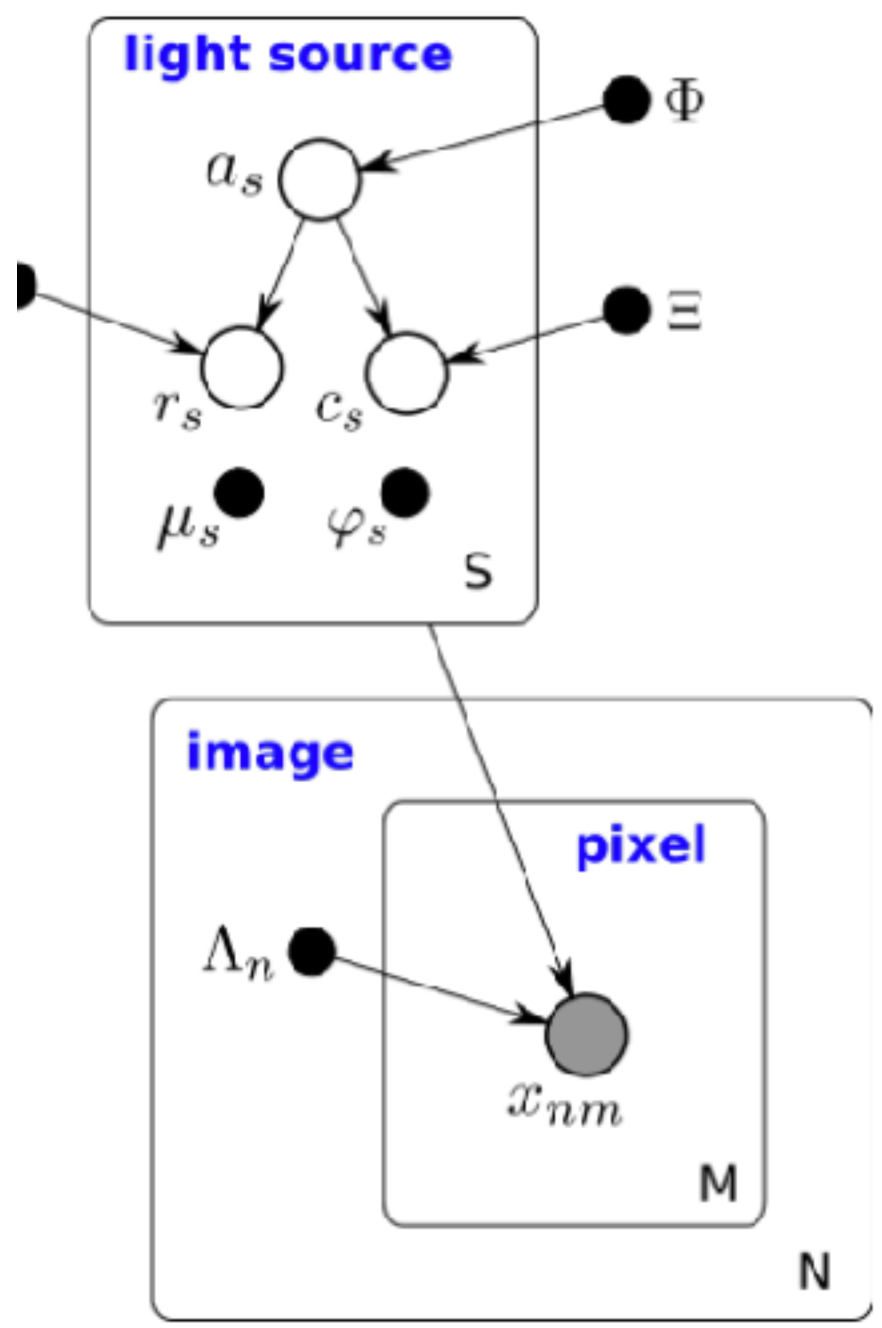
Cataloging the universe

Intel Knights Landing, SIMD, Multi-threading

↓

Multiple dispatch, Generic programming

Most light sources are near the detection limit.



Cataloging the universe

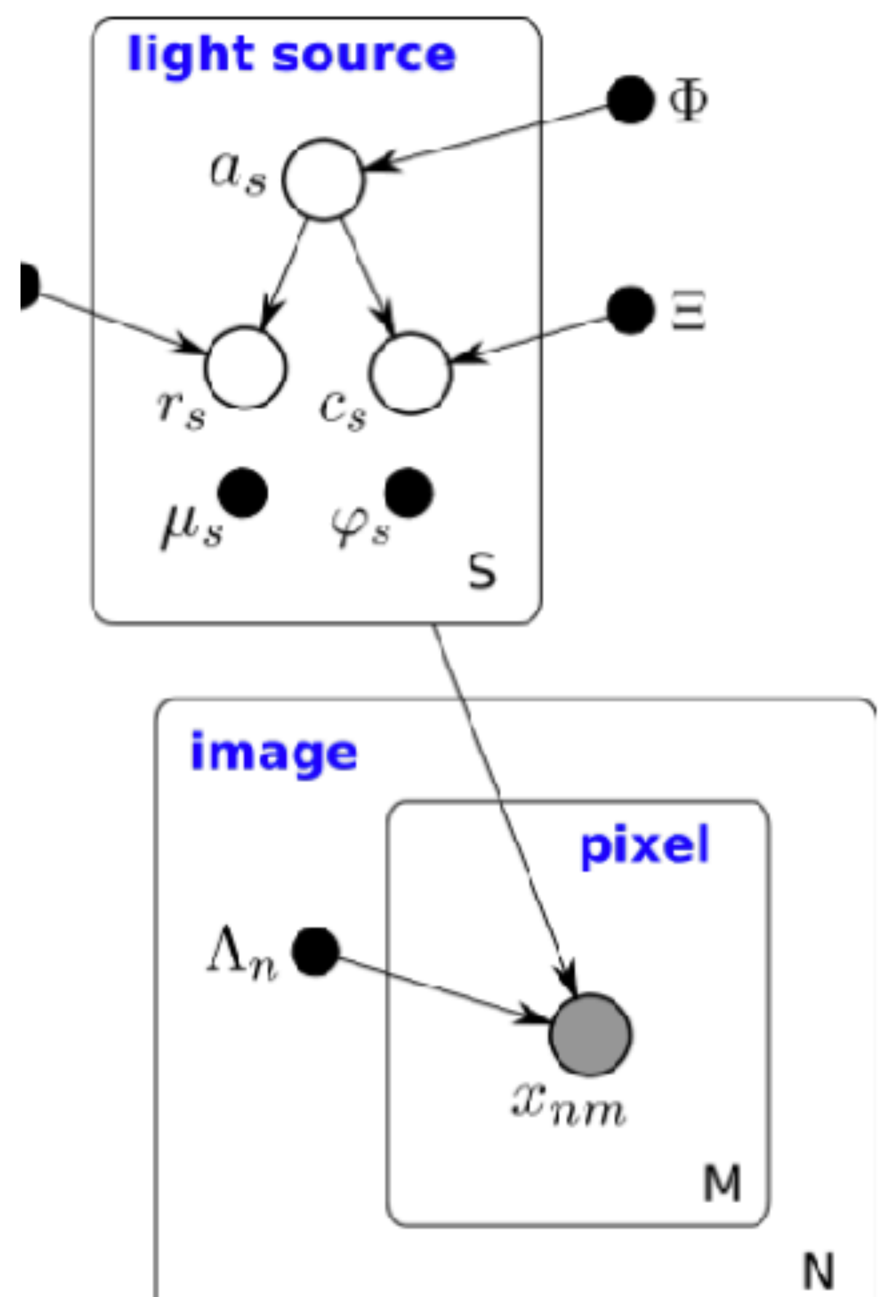
Intel Knights Landing, SIMD, Multi-threading

↓

Multiple dispatch, Generic programming

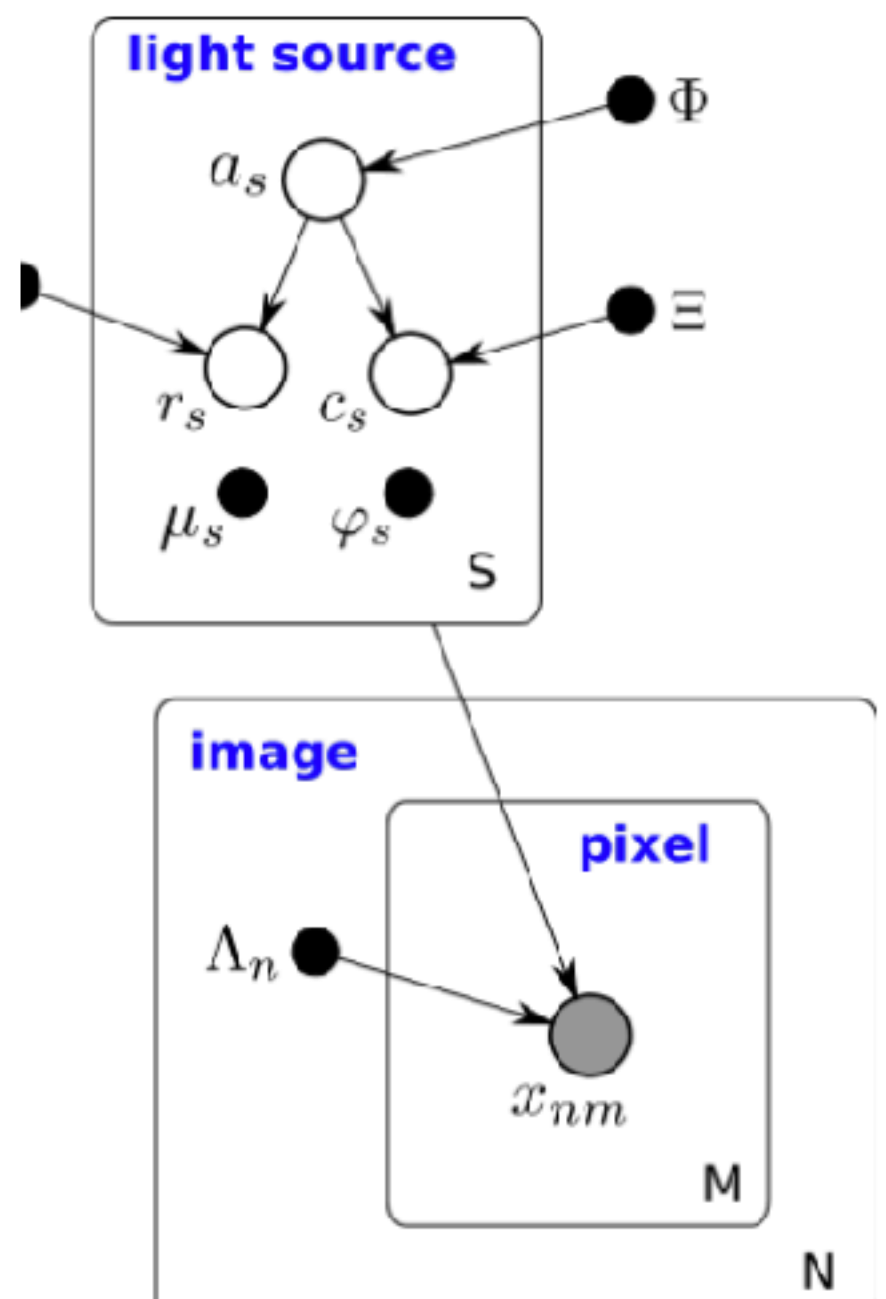
↓

Most light sources are near the detection limit.



Cataloging the universe

Most light sources are near the detection limit.



Intel Knights Landing, SIMD, Multi-threading

↓

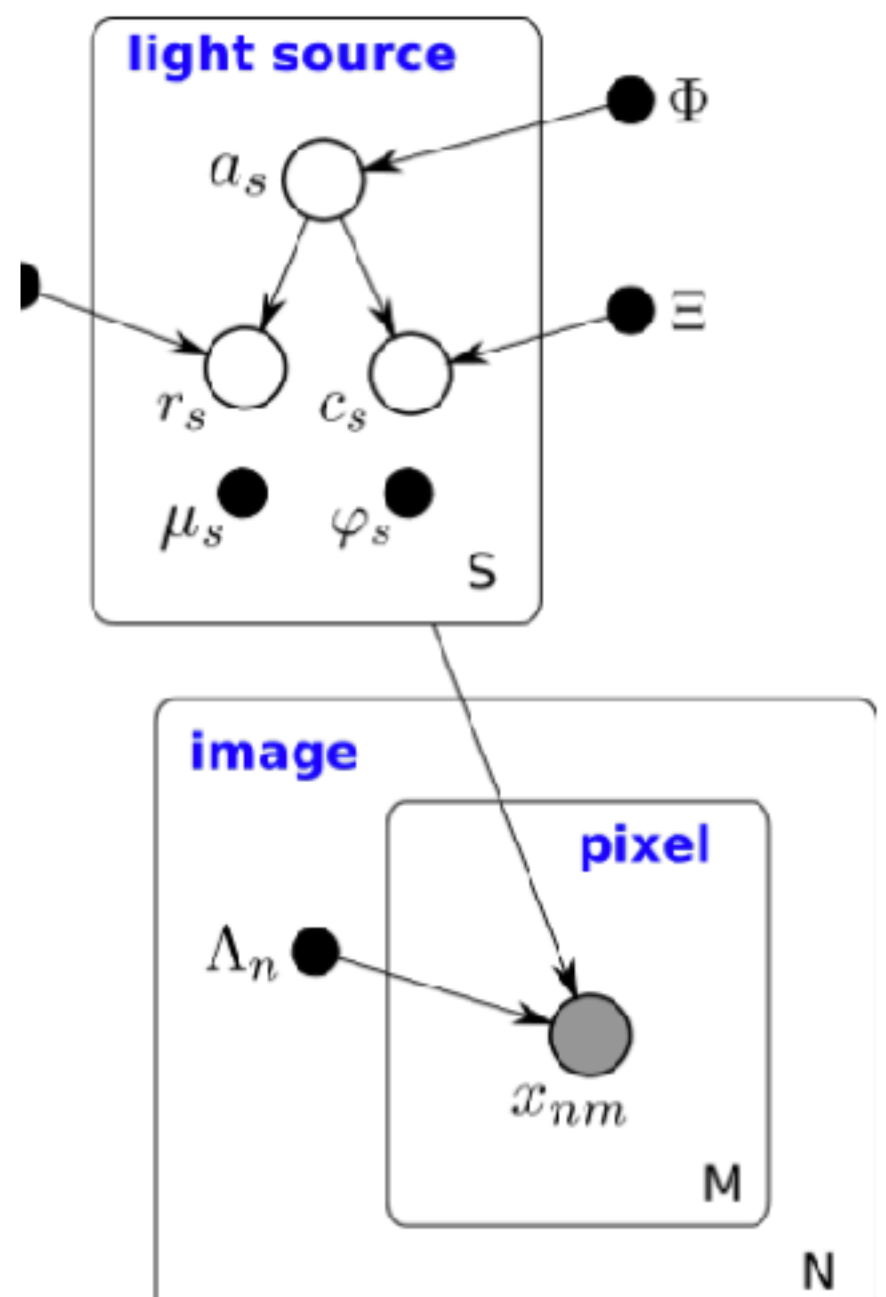
Multiple dispatch, Generic programming

↓

StaticArrays, DataFrames, FITSIO

Cataloging the universe

Most light sources are near the detection limit.



Intel Knights Landing, SIMD, Multi-threading

↓

Multiple dispatch, Generic programming

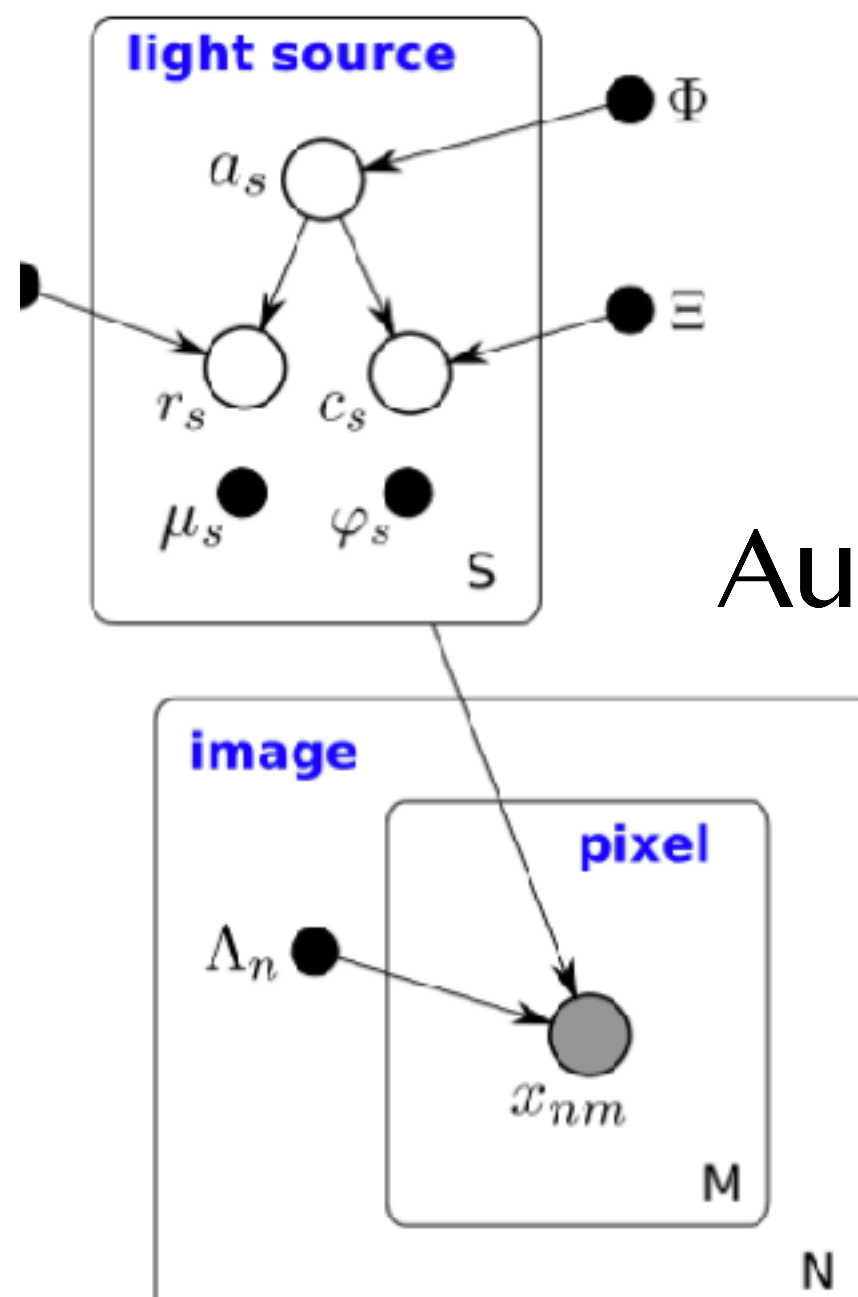
↓

StaticArrays, DataFrames, FITSIO

↓

Cataloging the universe

Most light sources are near the detection limit.



Intel Knights Landing, SIMD, Multi-threading

↓

Multiple dispatch, Generic programming

↓

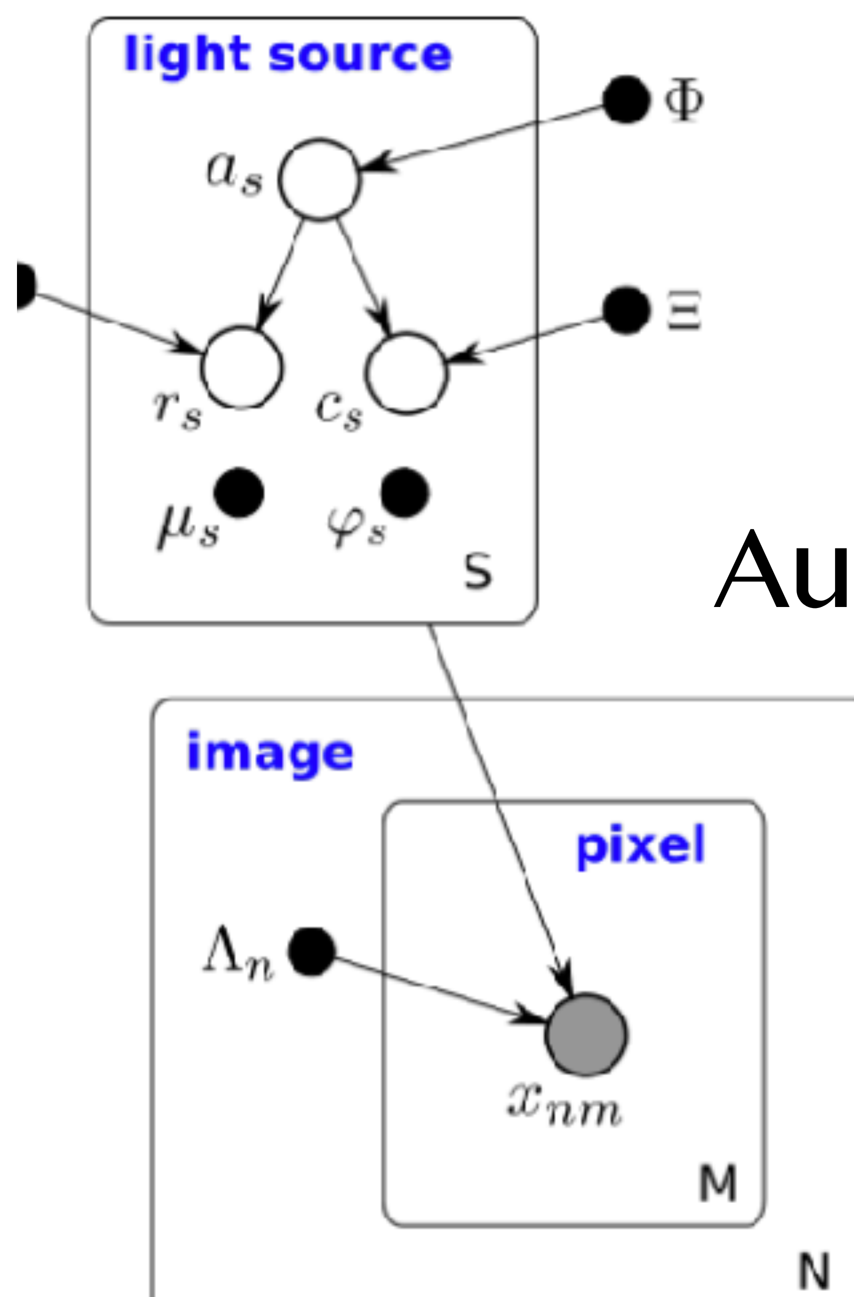
StaticArrays, DataFrames, FITSIO

↓

Automatic Differentiation, Optimization

Cataloging the universe

Most light sources are near the detection limit.



Intel Knights Landing, SIMD, Multi-threading

↓

Multiple dispatch, Generic programming

↓

StaticArrays, DataFrames, FITSIO

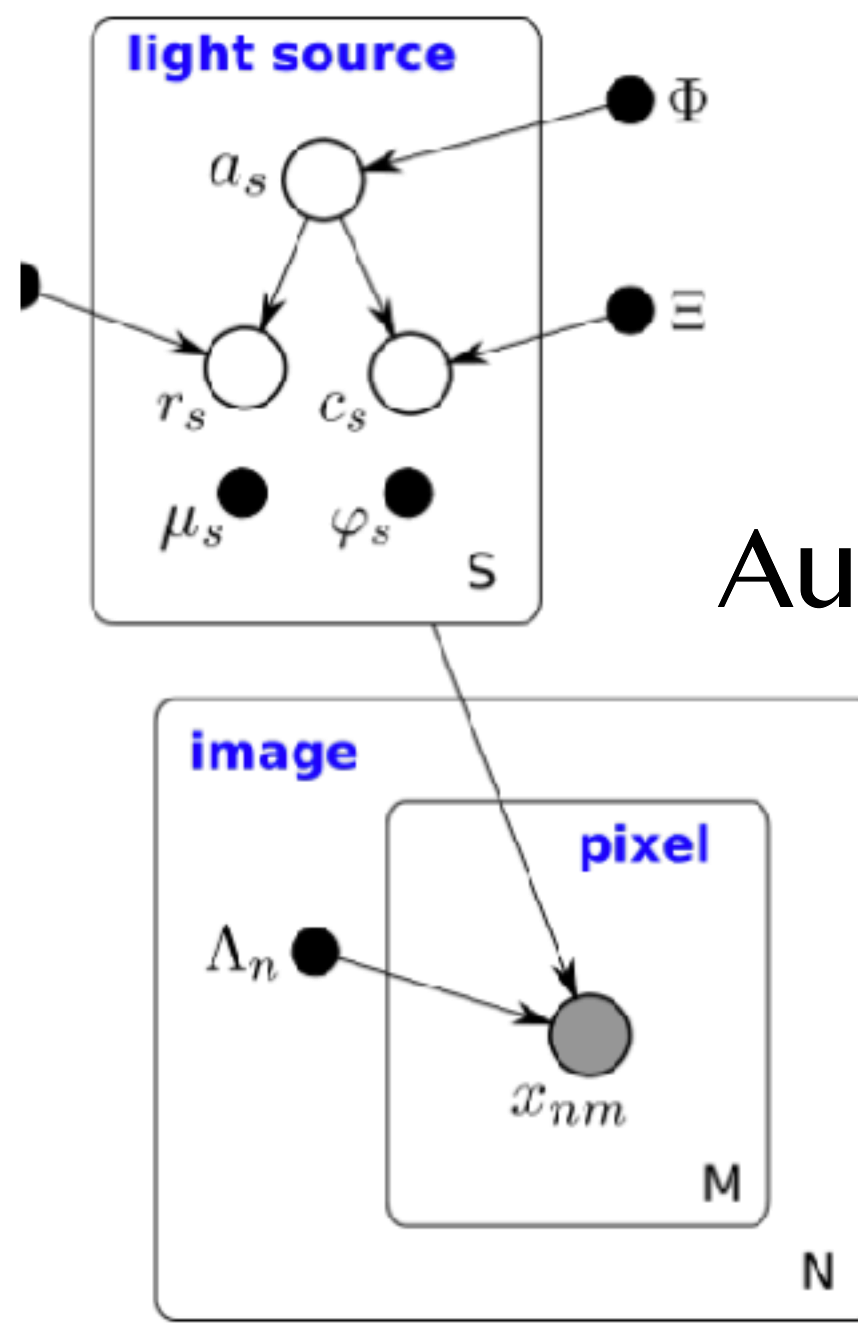
↓

Automatic Differentiation, Optimization

↓

Cataloging the universe

Most light sources are near the detection limit.



Intel Knights Landing, SIMD, Multi-threading

↓

Multiple dispatch, Generic programming

↓

StaticArrays, DataFrames, FITSIO

↓

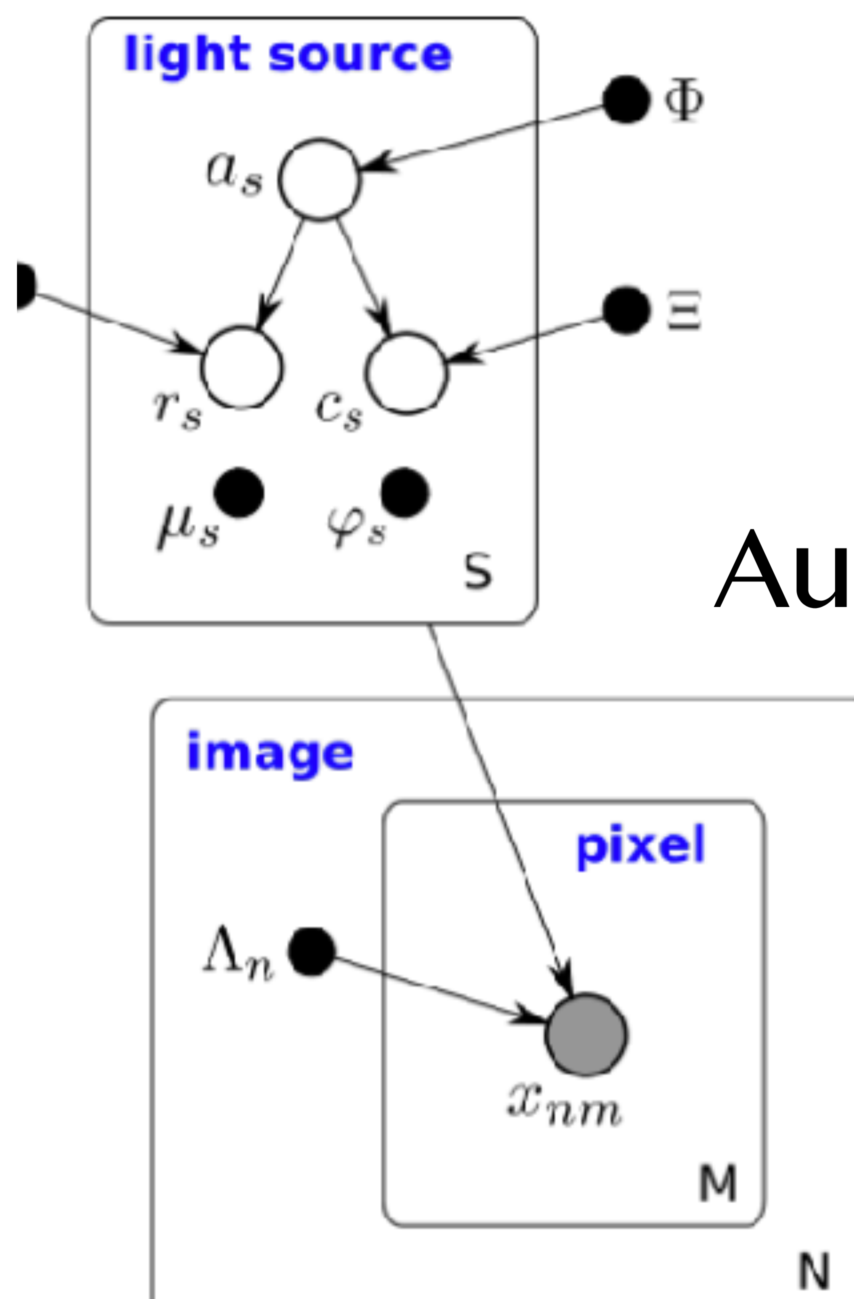
Automatic Differentiation, Optimization

↓

Complete astronomical catalog

Cataloging the universe

Most light sources are near the detection limit.



Intel Knights Landing, SIMD, Multi-threading

↓

Multiple dispatch, Generic programming

↓

StaticArrays, DataFrames, FITSIO

↓

Automatic Differentiation, Optimization

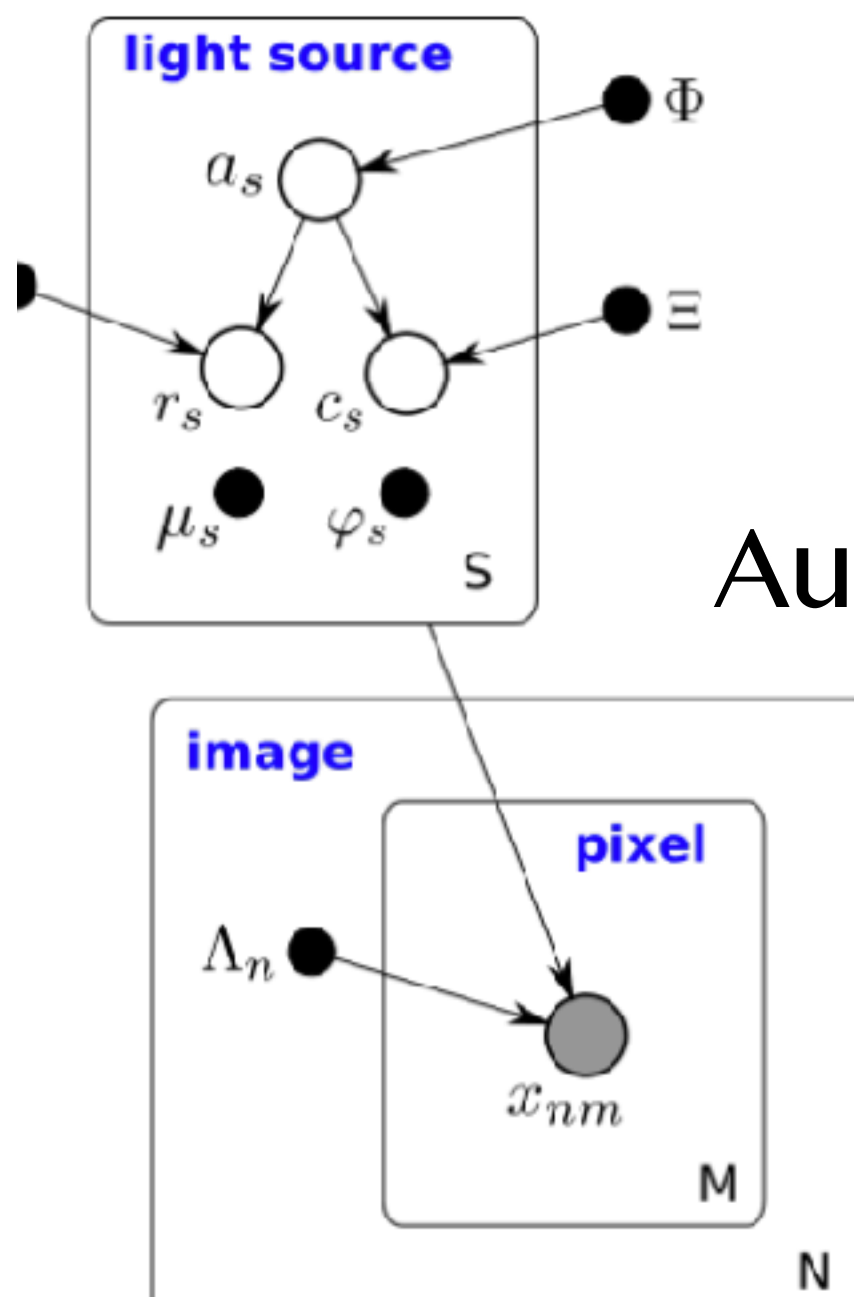
↓

Complete astronomical catalog

↓

Cataloging the universe

Most light sources are near the detection limit.



Intel Knights Landing, SIMD, Multi-threading

Multiple dispatch, Generic programming

StaticArrays, DataFrames, FITSIO

Automatic Differentiation, Optimization

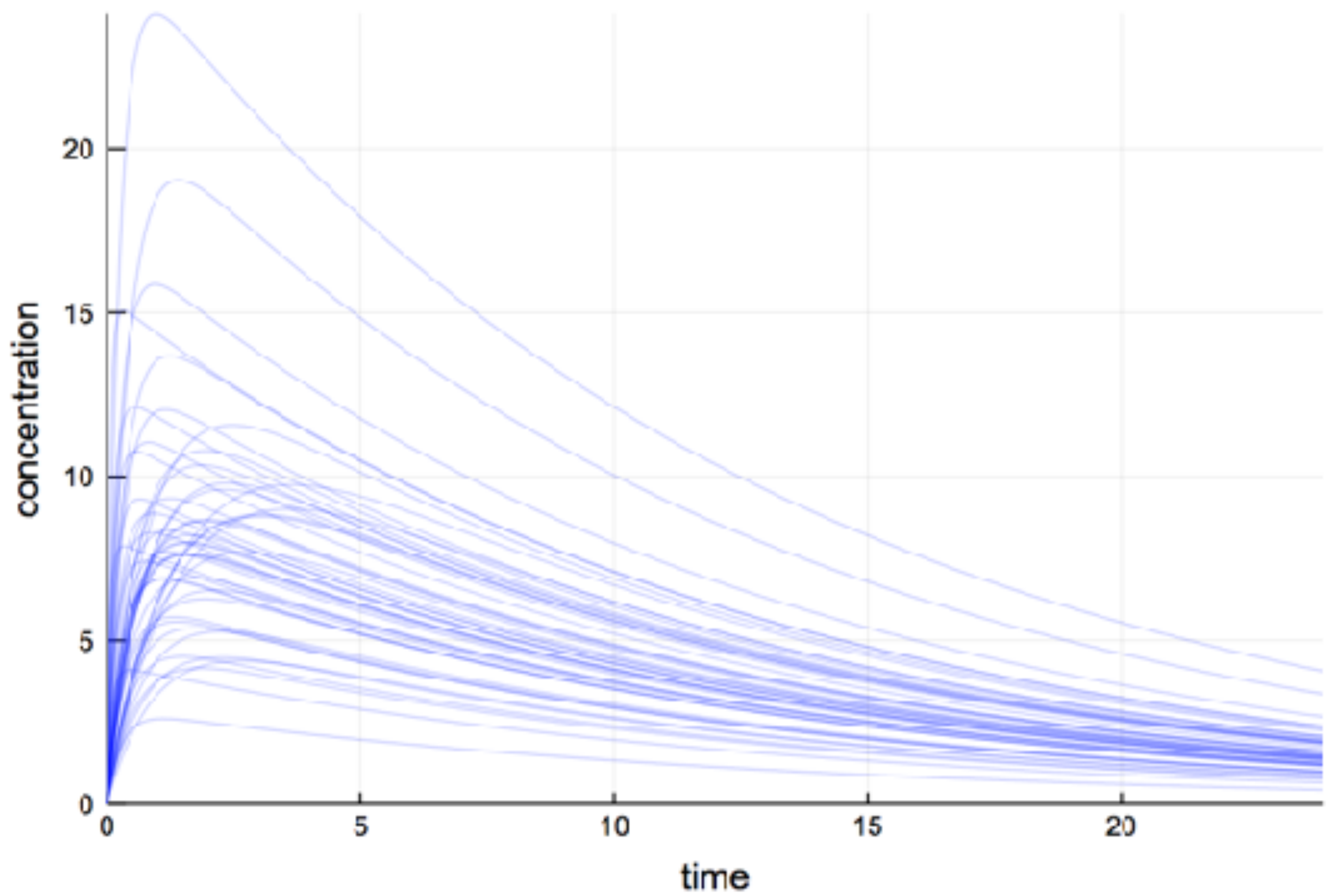
Complete astronomical catalog

Scientific discovery

Personalized medicine

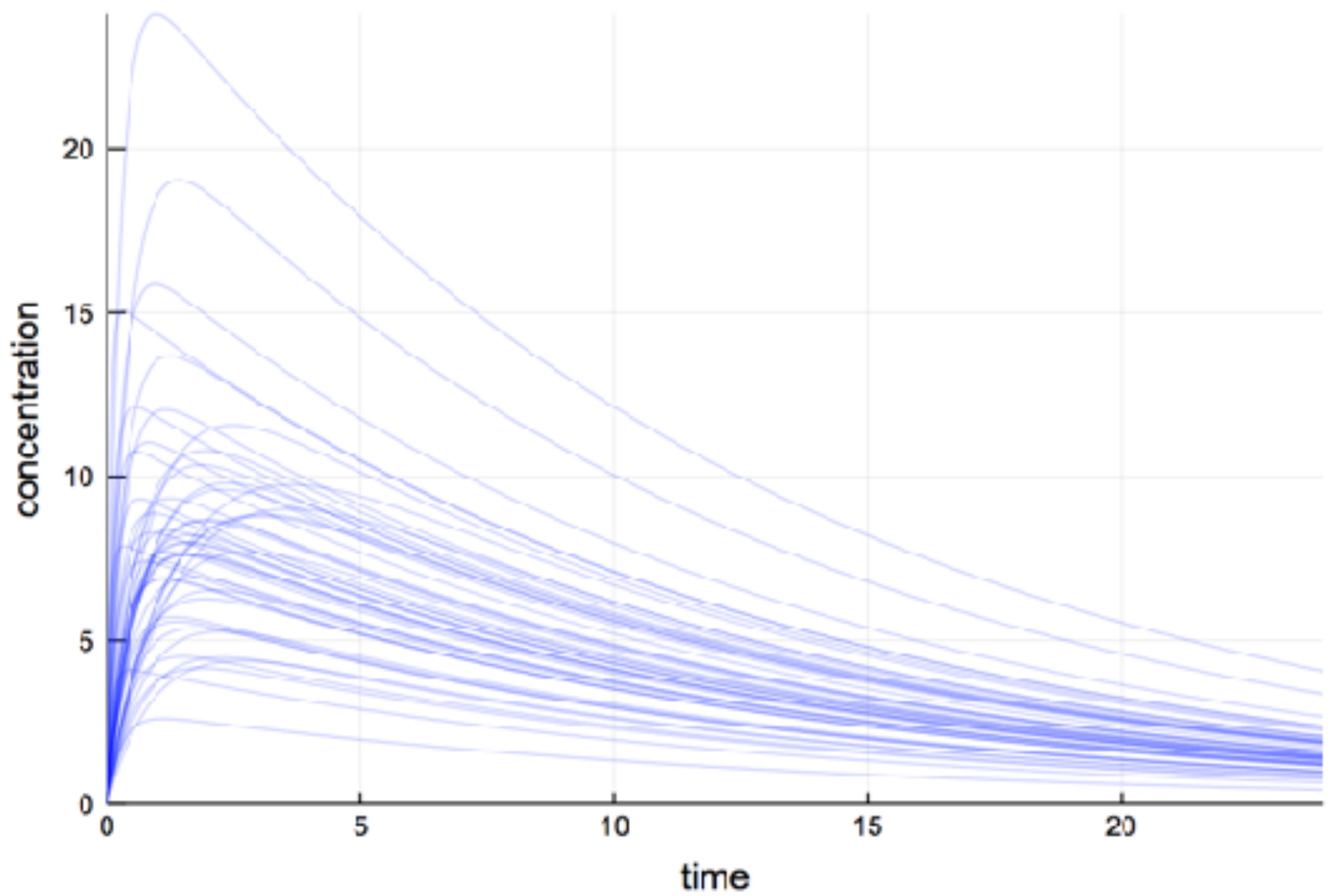
Personalized medicine

Simulation of the random effects from a PK/PD model



Personalized medicine

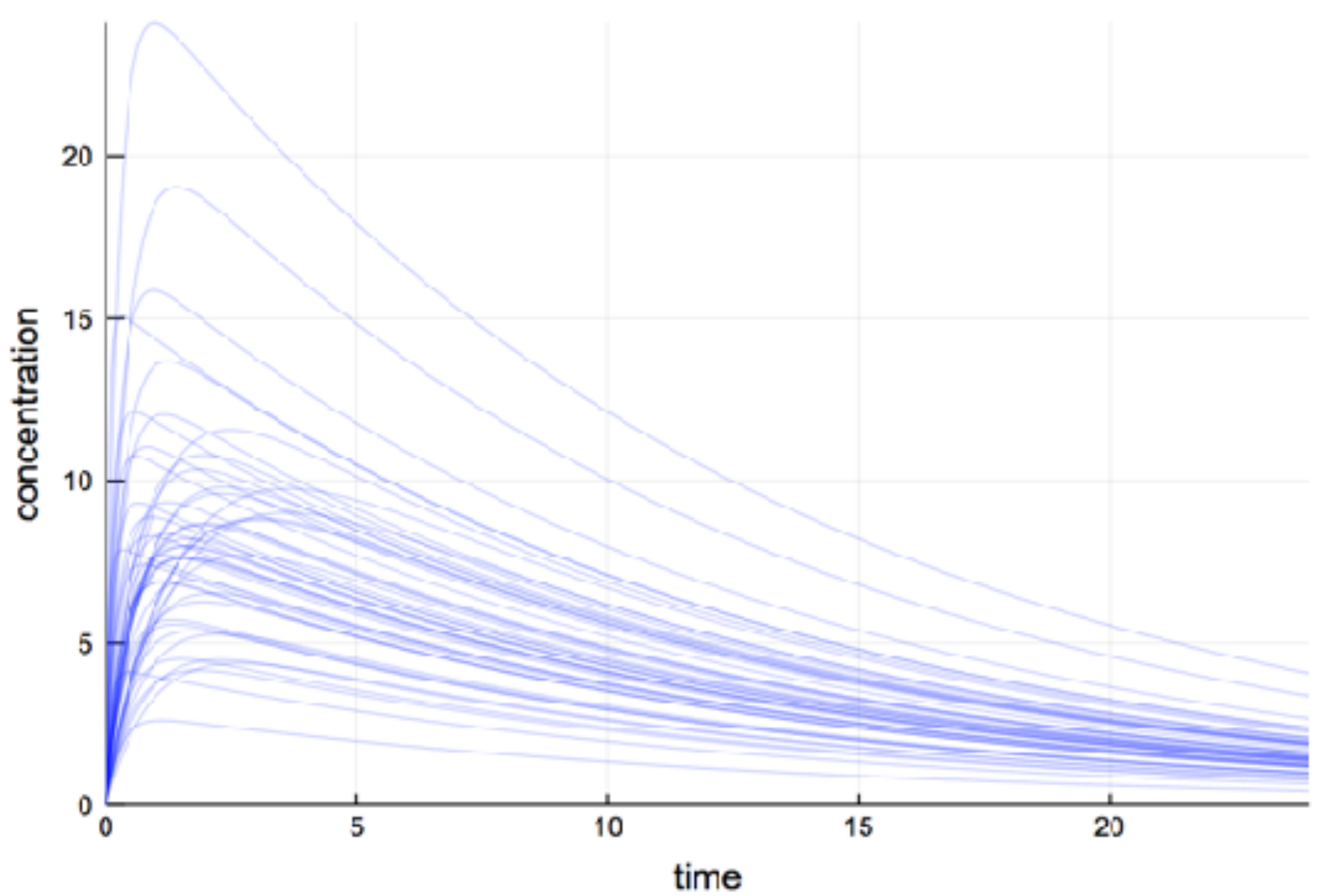
Simulation of the random effects from a PK/PD model



GPUs

Personalized medicine

Simulation of the random effects from a PK/PD model

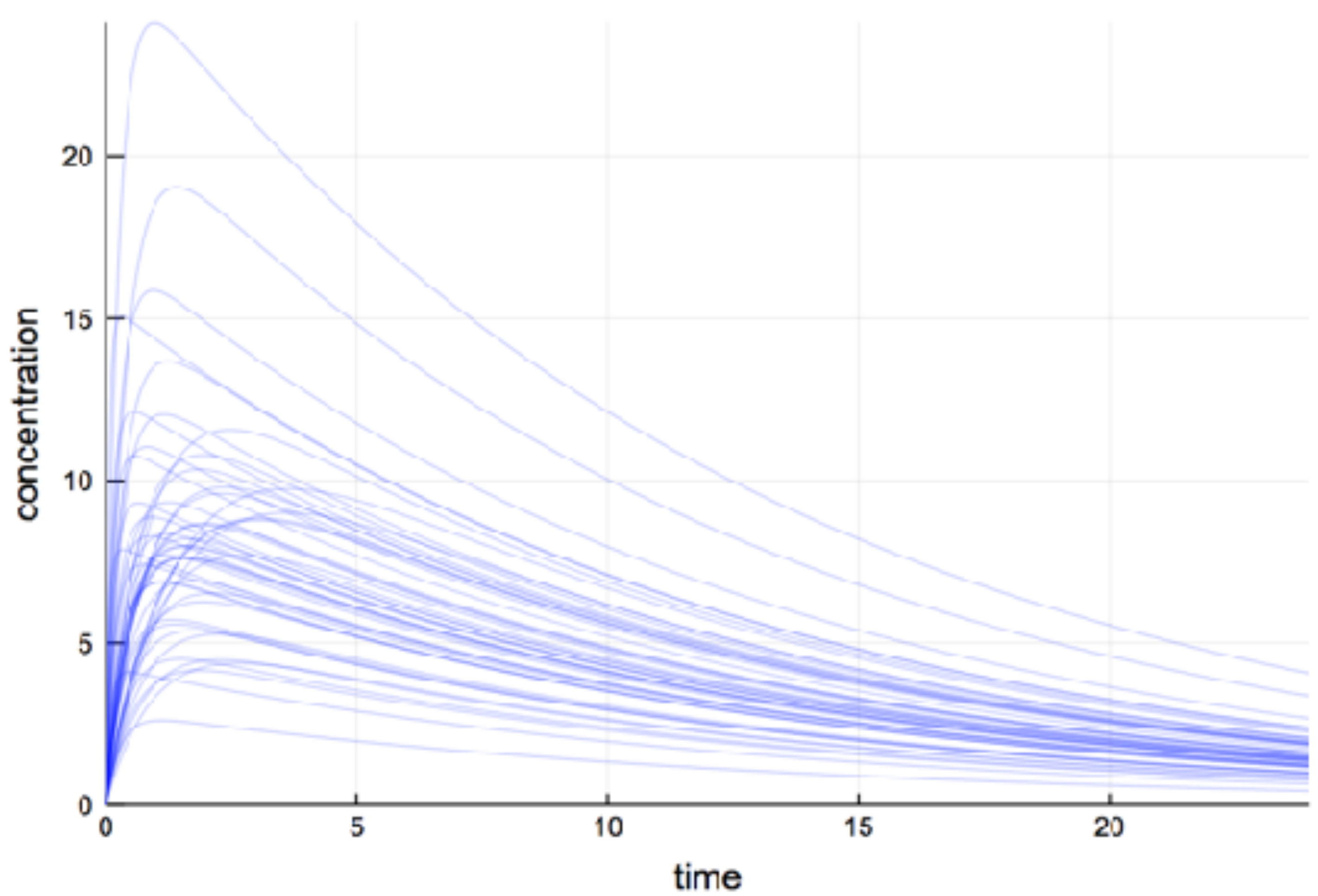


GPUs

GPUs

Personalized medicine

Simulation of the random effects from a PK/PD model

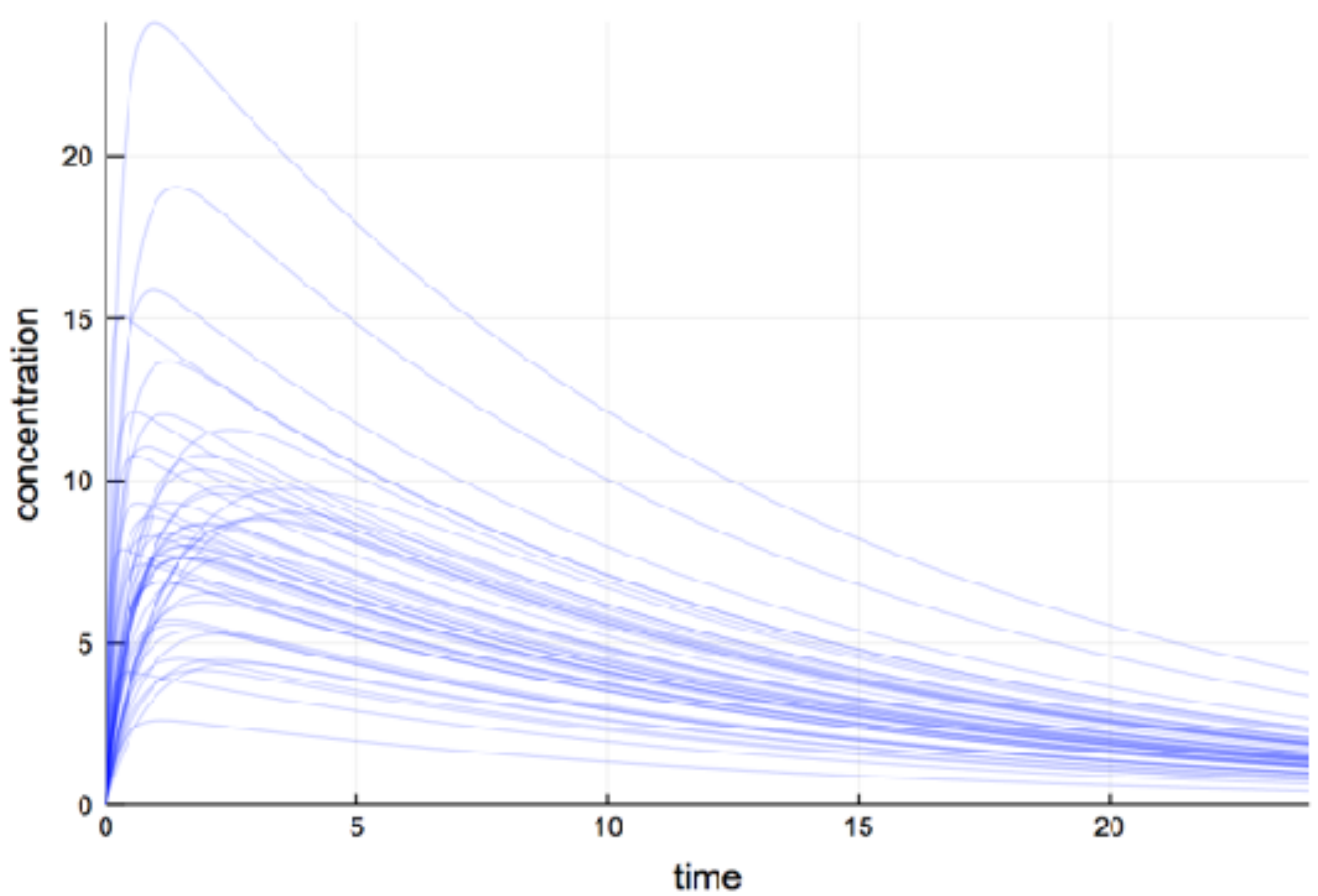


GPUs

GPUs

Personalized medicine

Simulation of the random effects from a PK/PD model



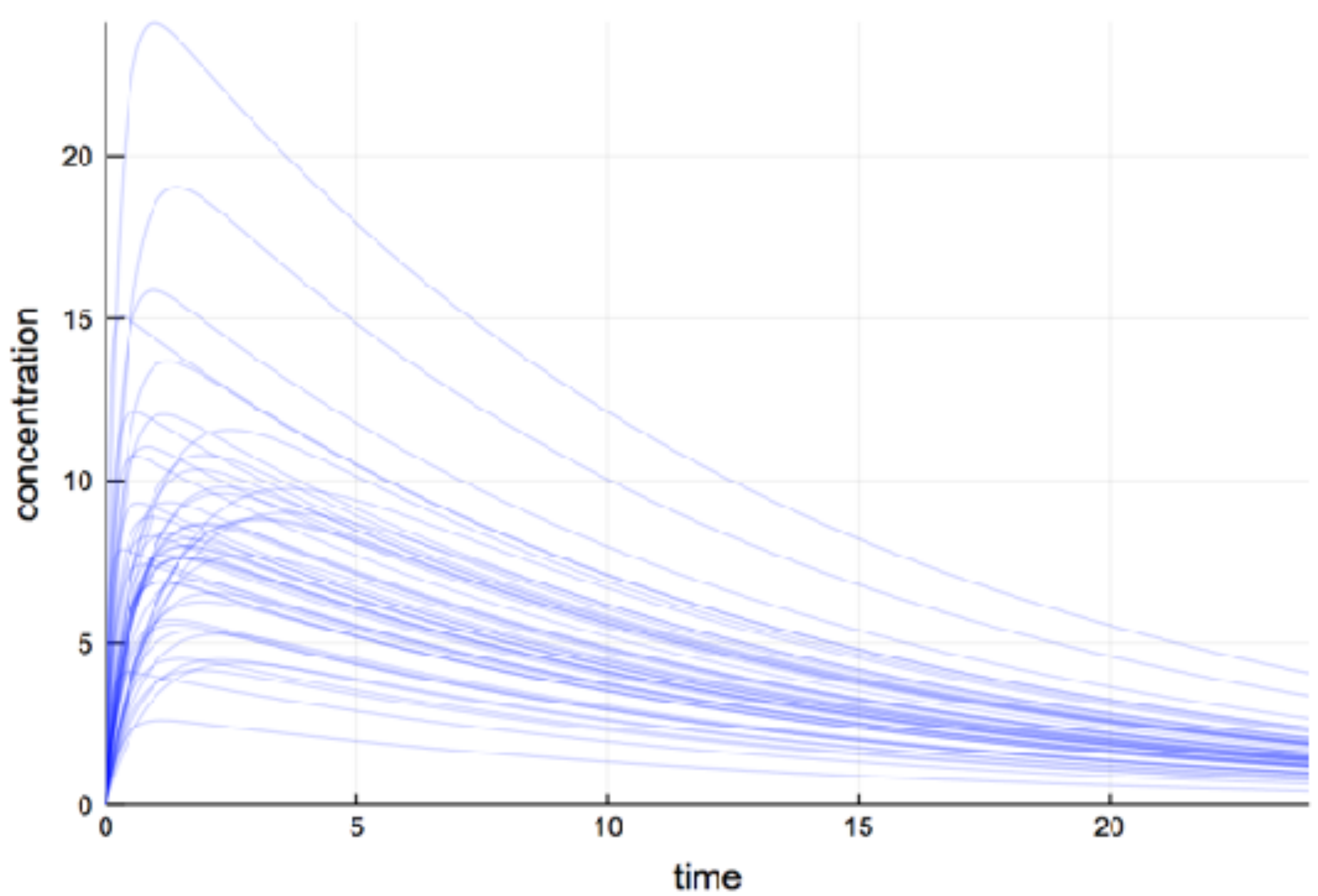
GPUs

GPUs

Multiple dispatch, Generic programming

Personalized medicine

Simulation of the random effects from a PK/PD model



GPUs

GPUs

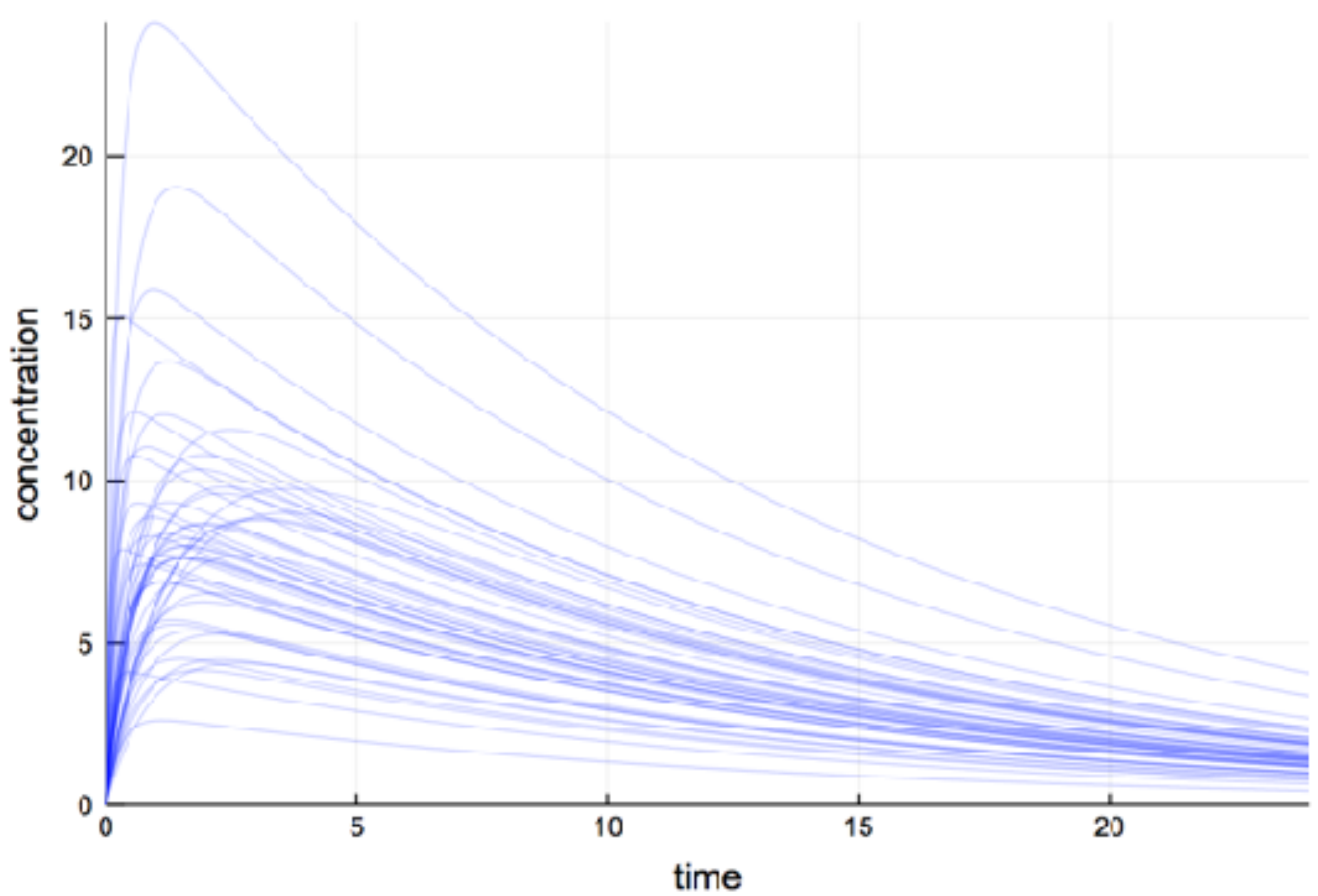
↓

Multiple dispatch, Generic programming

↓

Personalized medicine

Simulation of the random effects from a PK/PD model



GPUs

GPUs

↓

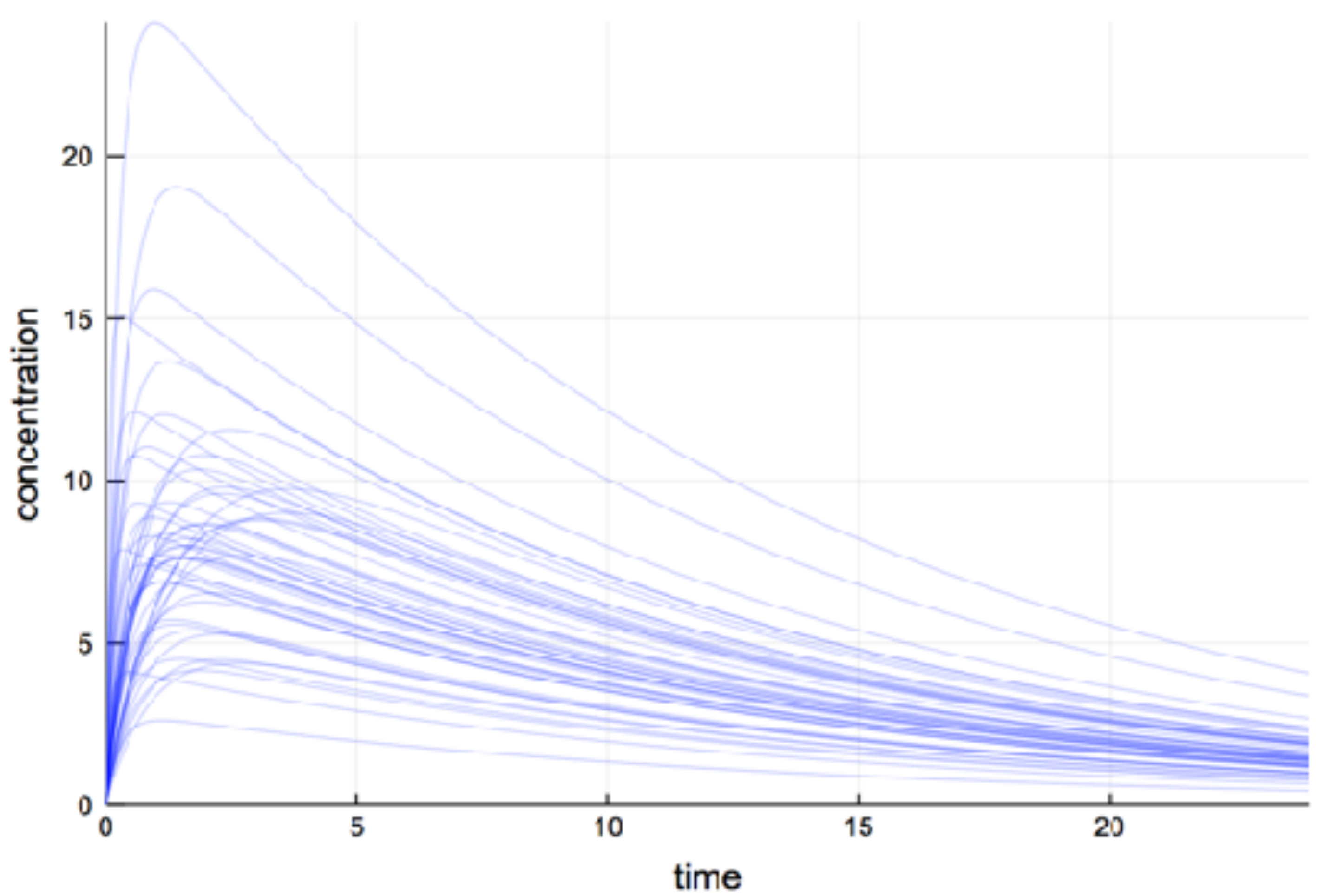
Multiple dispatch, Generic programming

↓

StaticArrays, DiffEq

Personalized medicine

Simulation of the random effects from a PK/PD model



GPUs

GPUs

↓

Multiple dispatch, Generic programming

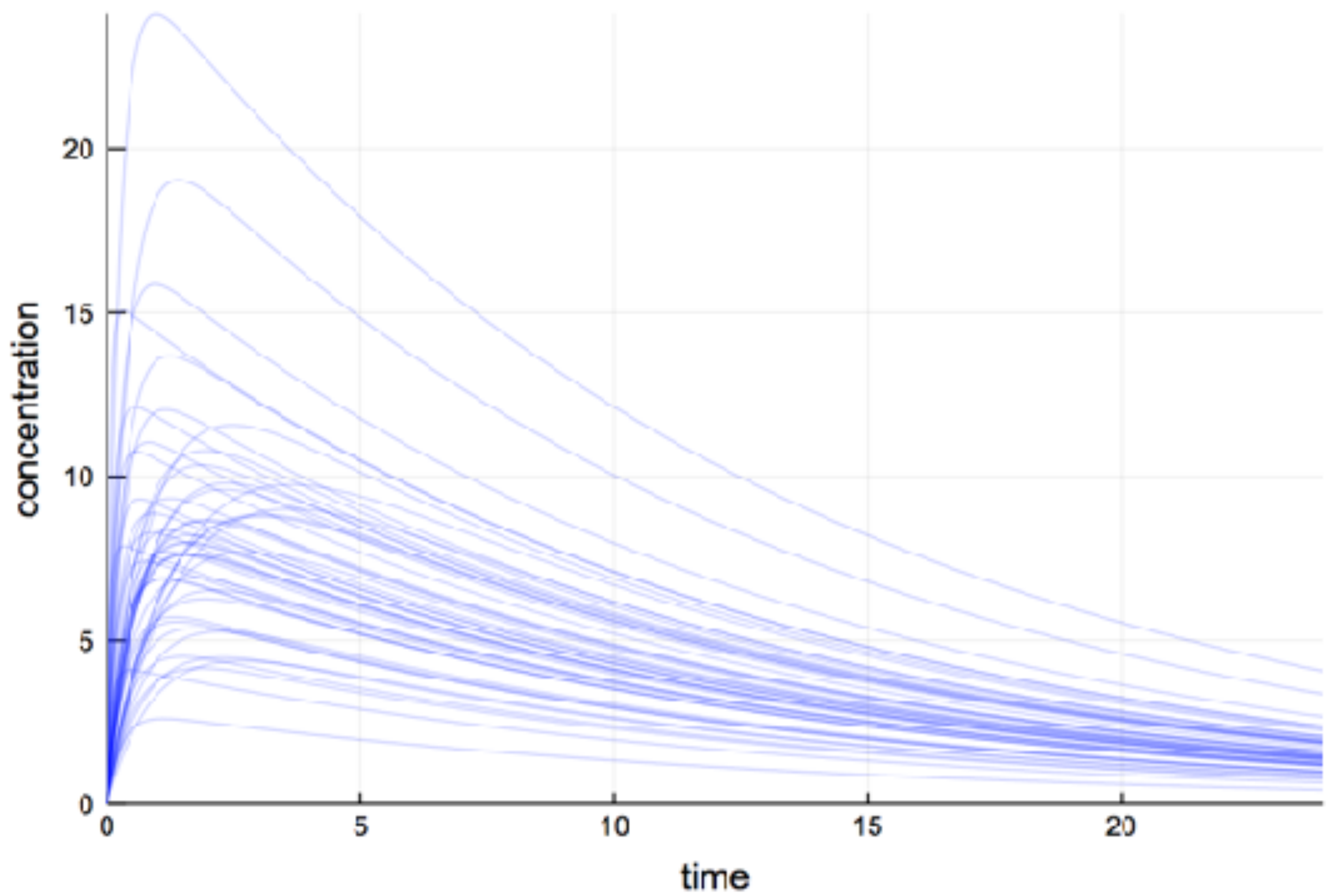
↓

StaticArrays, DiffEq

↓

Personalized medicine

Simulation of the random effects from a PK/PD model



GPUs

GPUs

↓

Multiple dispatch, Generic programming

↓

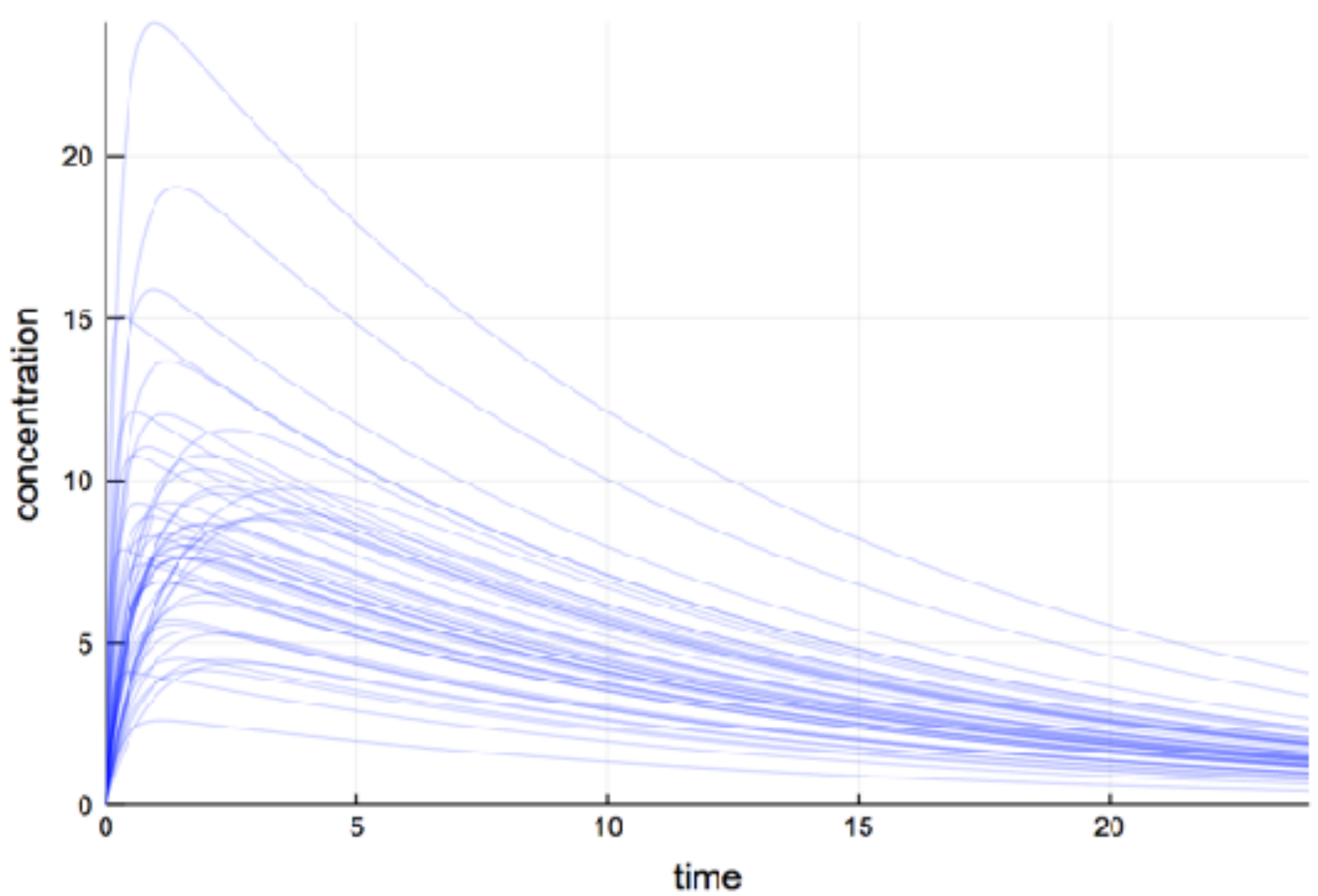
StaticArrays, DiffEq

↓

PharmacoKinetics and PharmacoDynamics

Personalized medicine

Simulation of the random effects from a PK/PD model



GPUs

GPUs

↓

Multiple dispatch, Generic programming

↓

StaticArrays, DiffEq

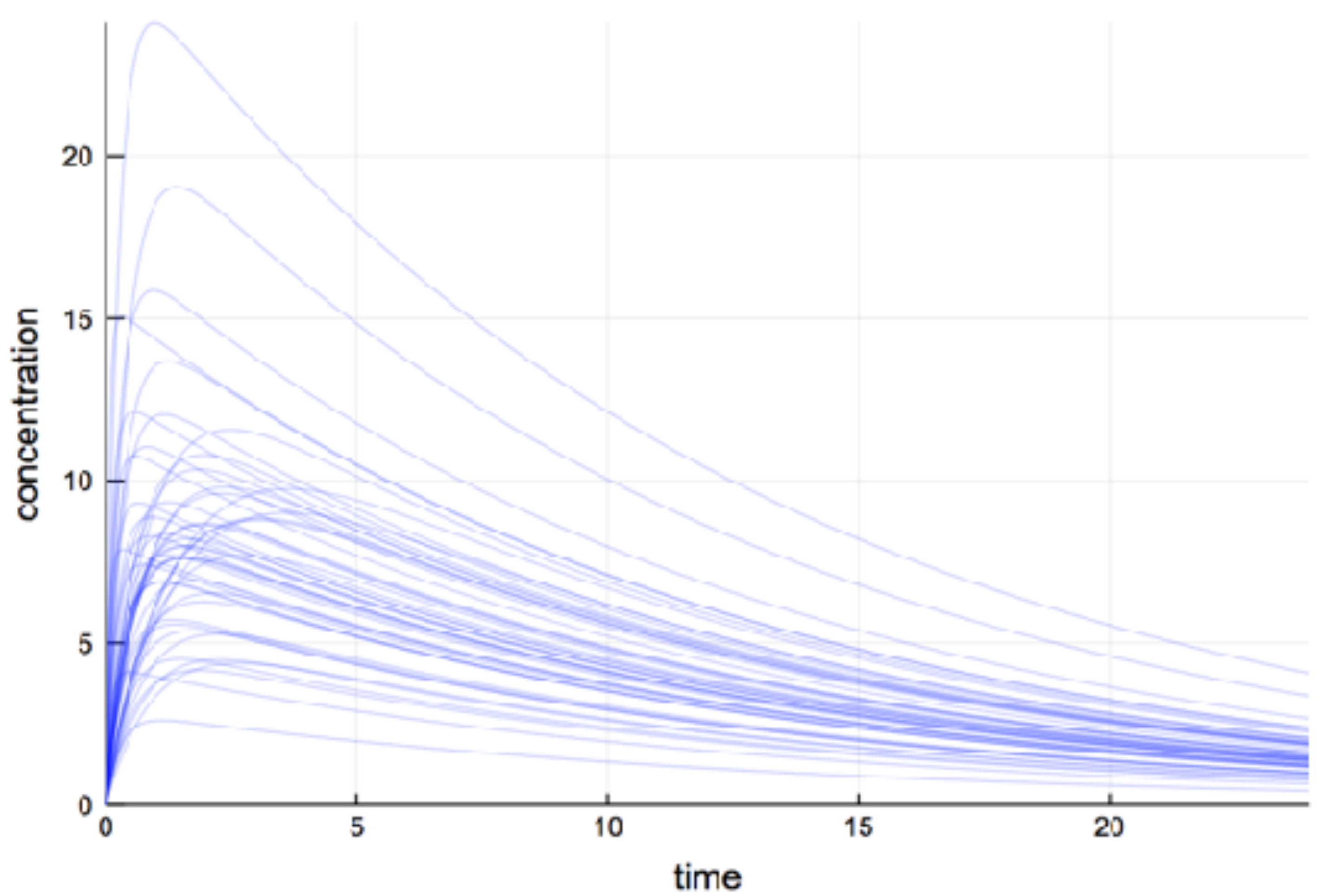
↓

PharmacoKinetics and PharmacoDynamics

↓

Personalized medicine

Simulation of the random effects from a PK/PD model



GPUs

GPUs

↓

Multiple dispatch, Generic programming

↓

StaticArrays, DiffEq

↓

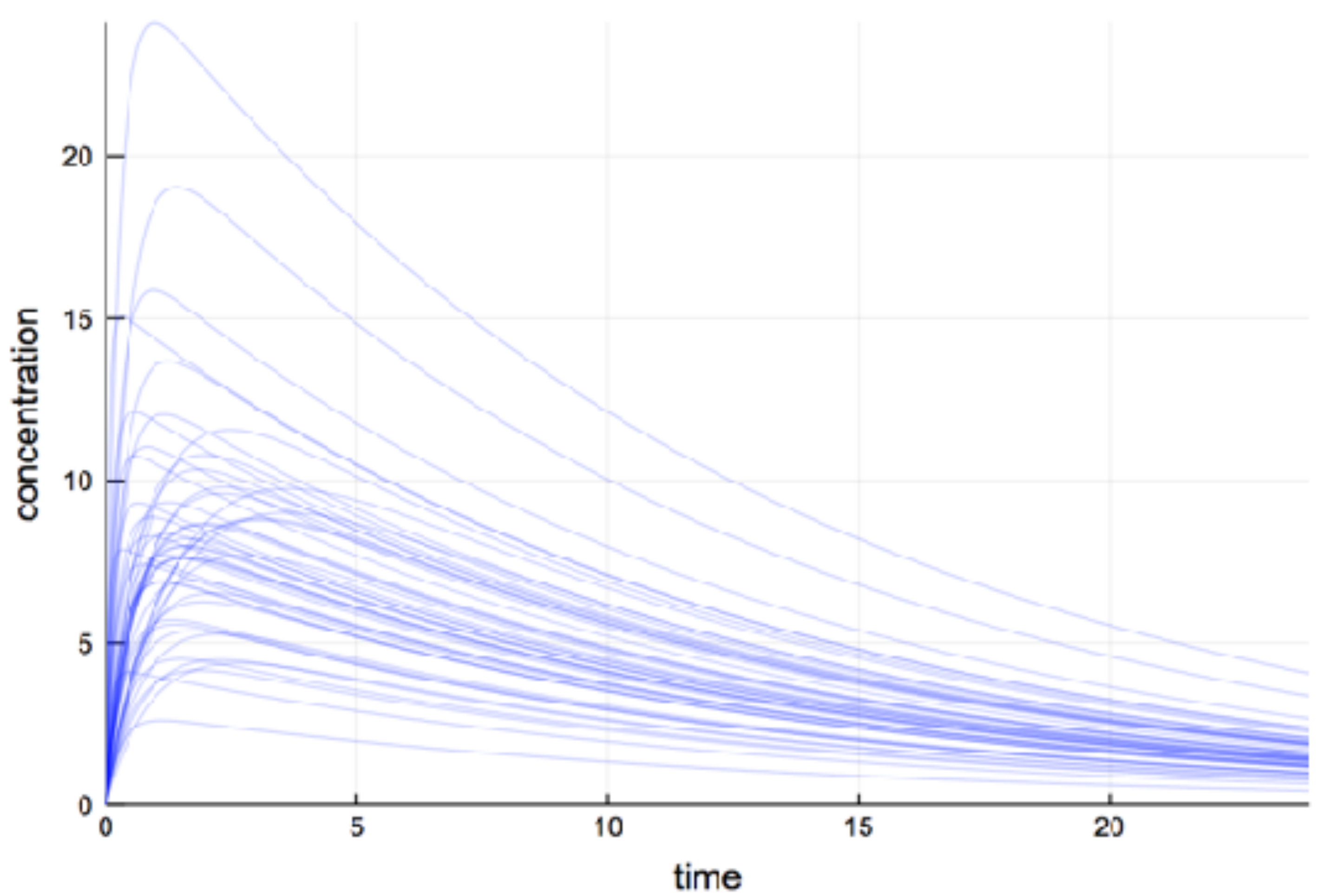
PharmacoKinetics and PharmacoDynamics

↓

Automatic Differentiation, Optimization

Personalized medicine

Simulation of the random effects from a PK/PD model



GPUs

GPUs

↓

Multiple dispatch, Generic programming

↓

StaticArrays, DiffEq

↓

PharmacoKinetics and PharmacoDynamics

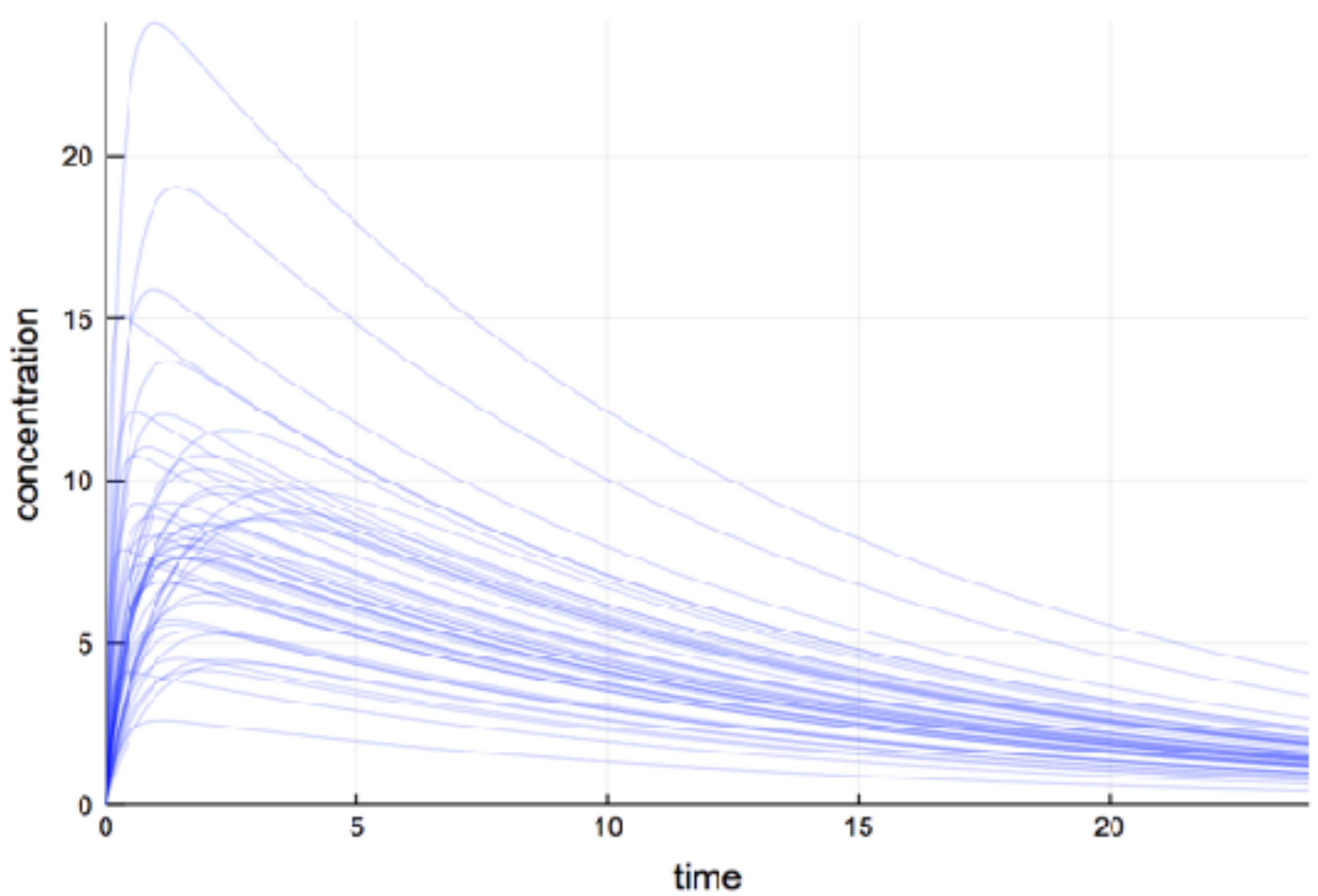
↓

Automatic Differentiation, Optimization

↓

Personalized medicine

Simulation of the random effects from a PK/PD model



GPUs

GPUs

↓

Multiple dispatch, Generic programming

↓

StaticArrays, DiffEq

↓

PharmacoKinetics and PharmacoDynamics

↓

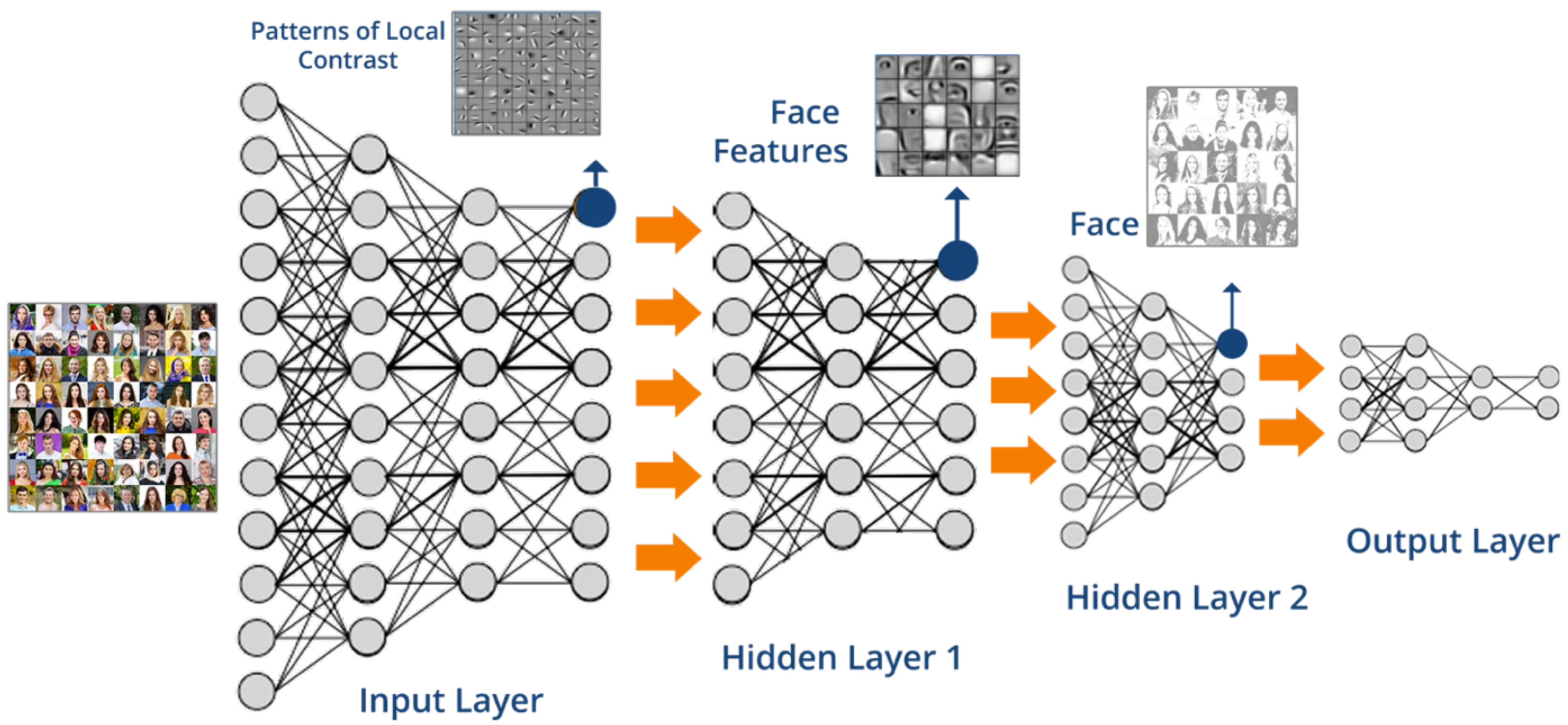
Automatic Differentiation, Optimization

↓

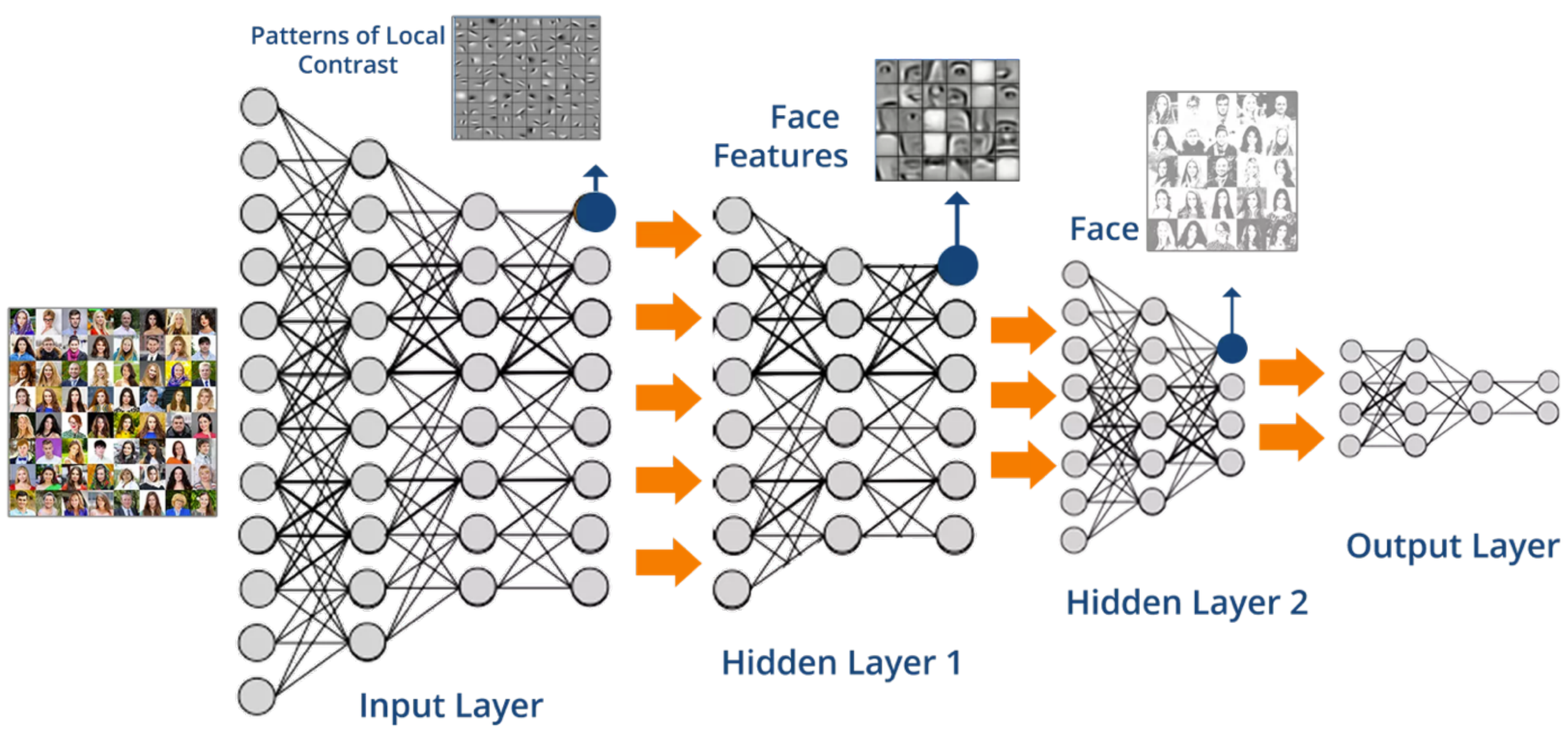
Personalized medicine, Affordable healthcare

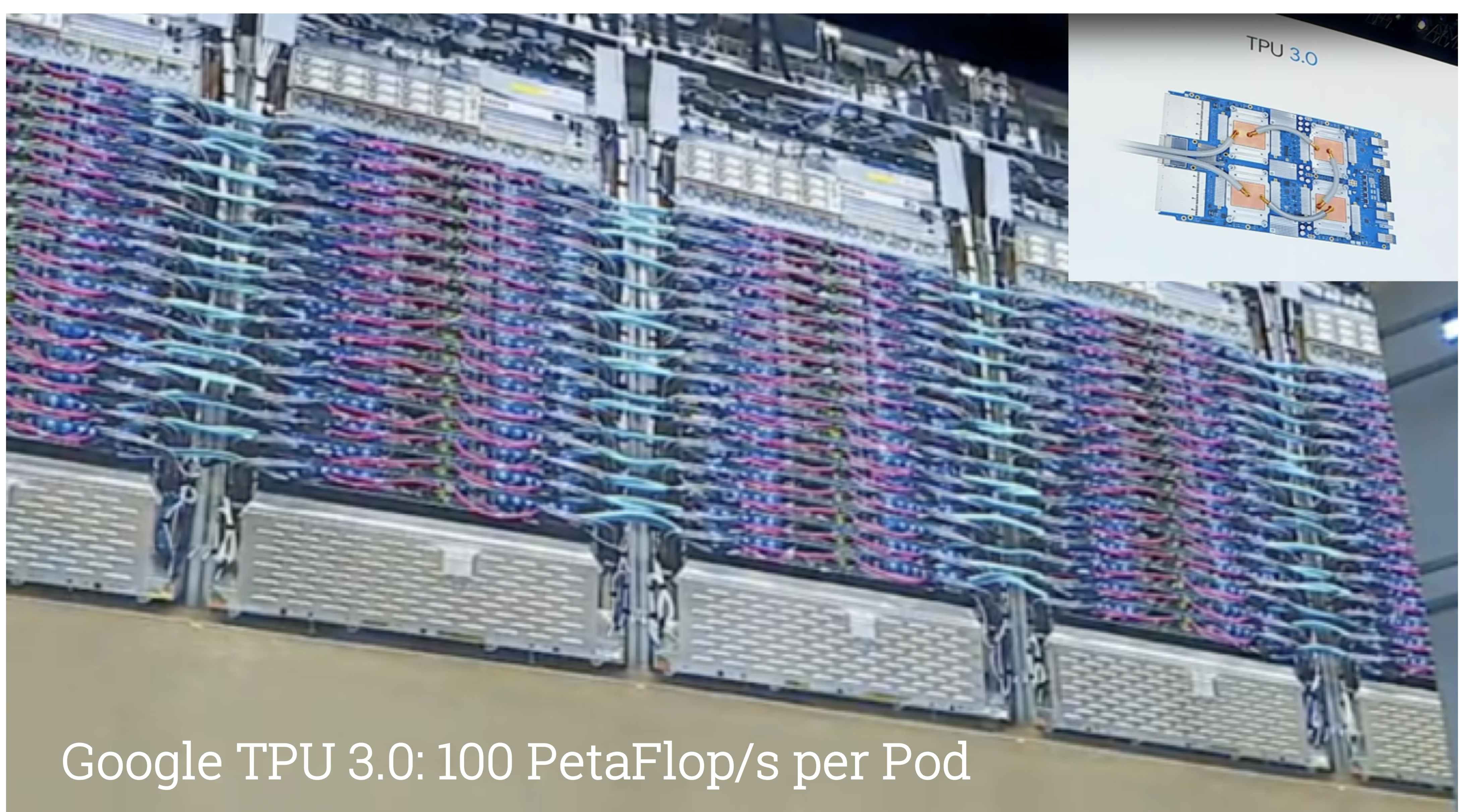
Machine Learning

Machine Learning



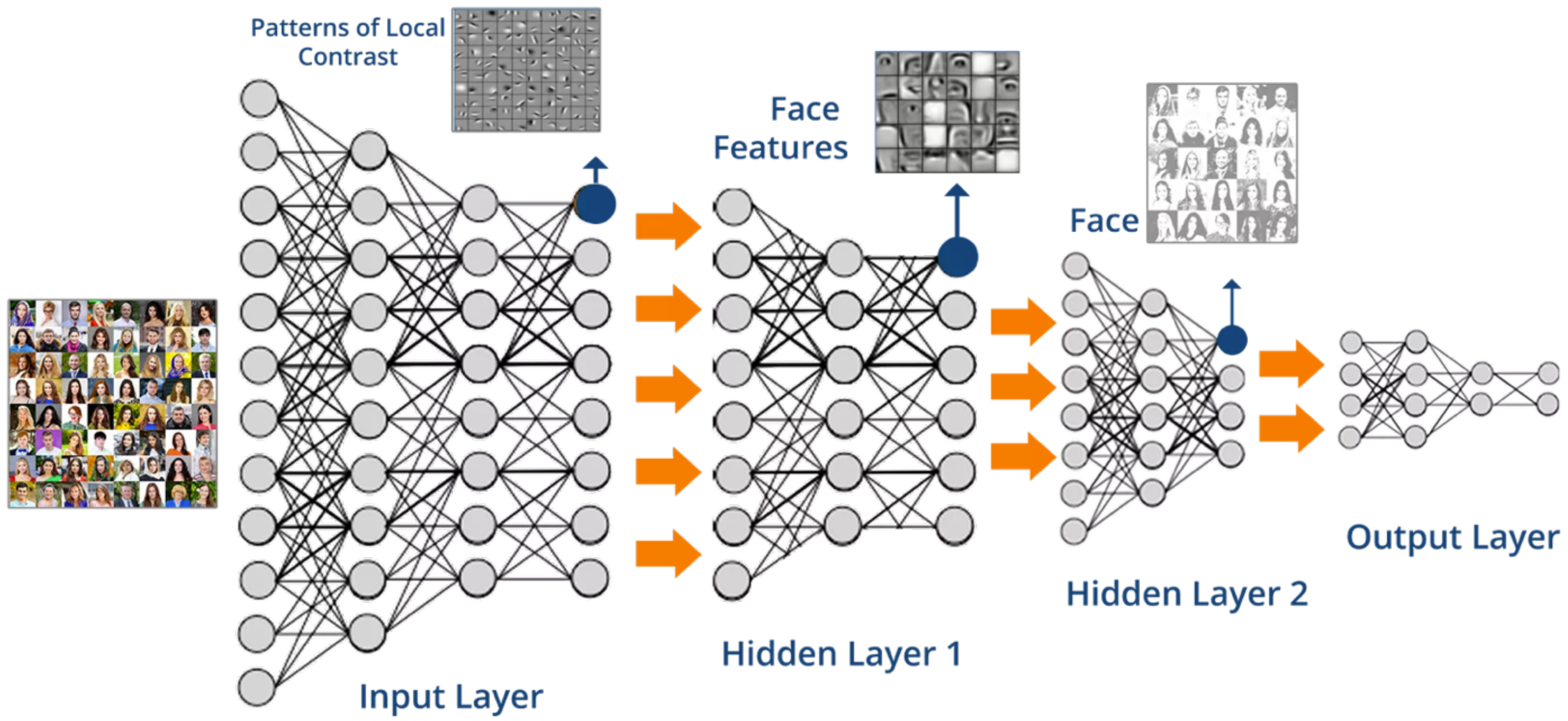
Machine Learning

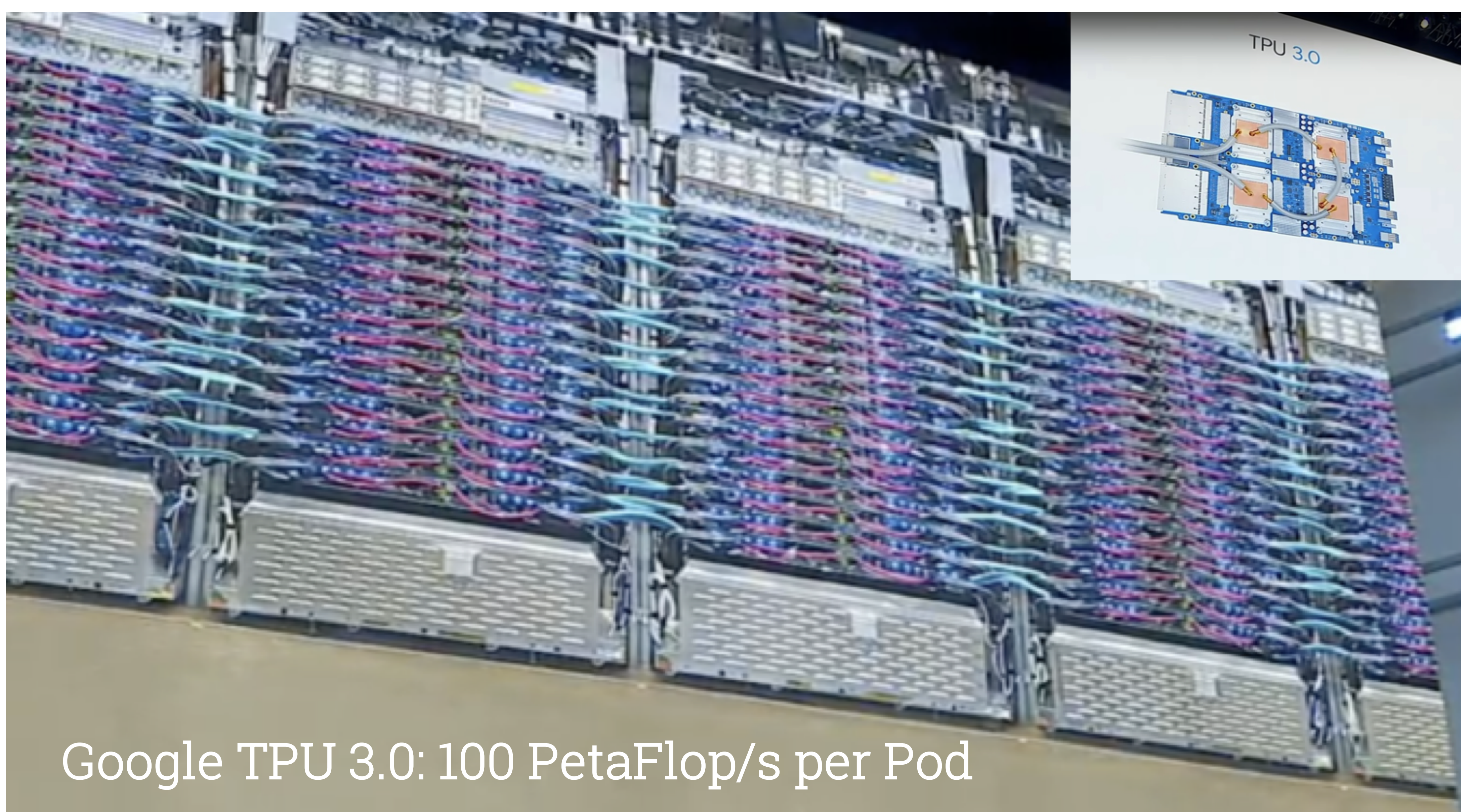




Google TPU 3.0: 100 PetaFlop/s per Pod

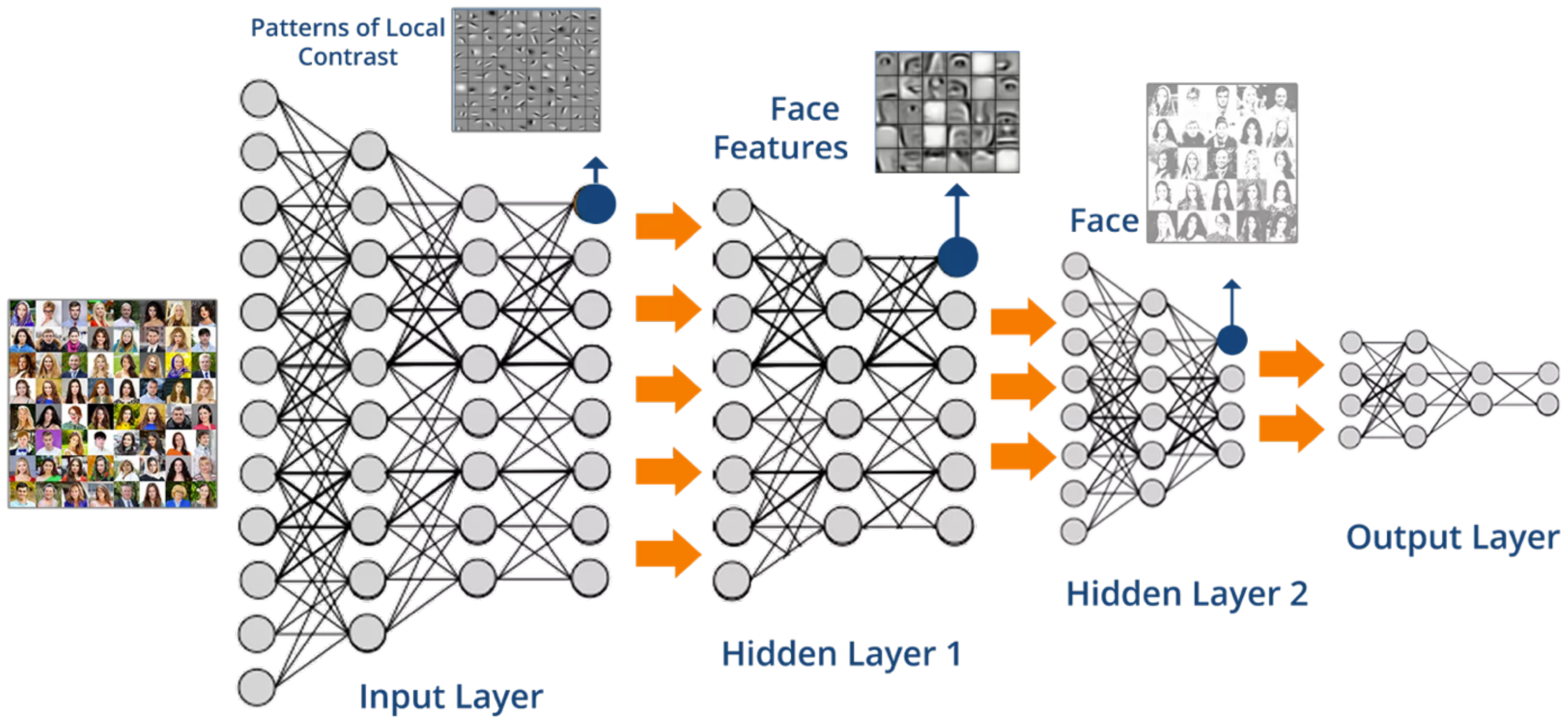
Machine Learning



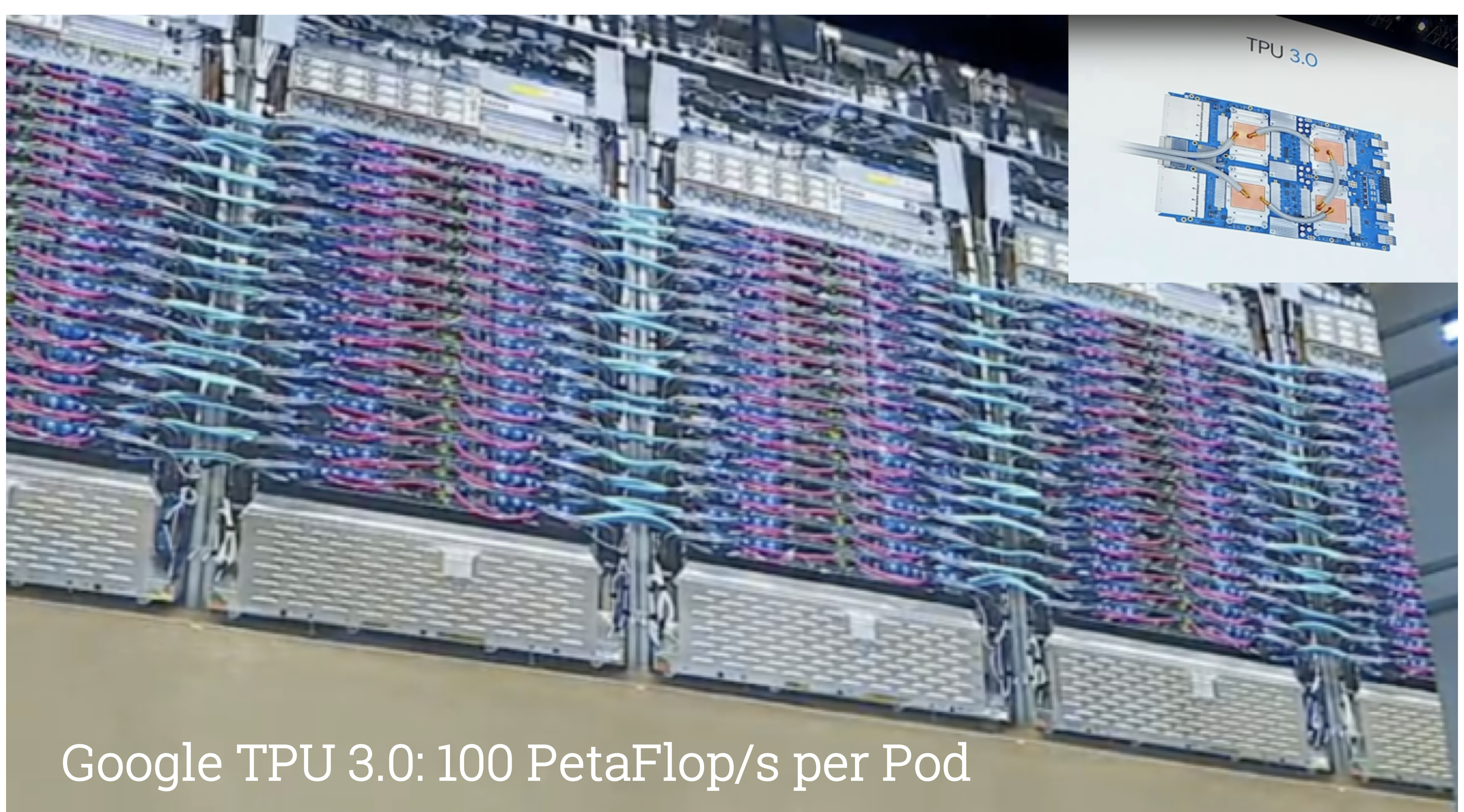


Google TPUs, GraphCores, Nervana

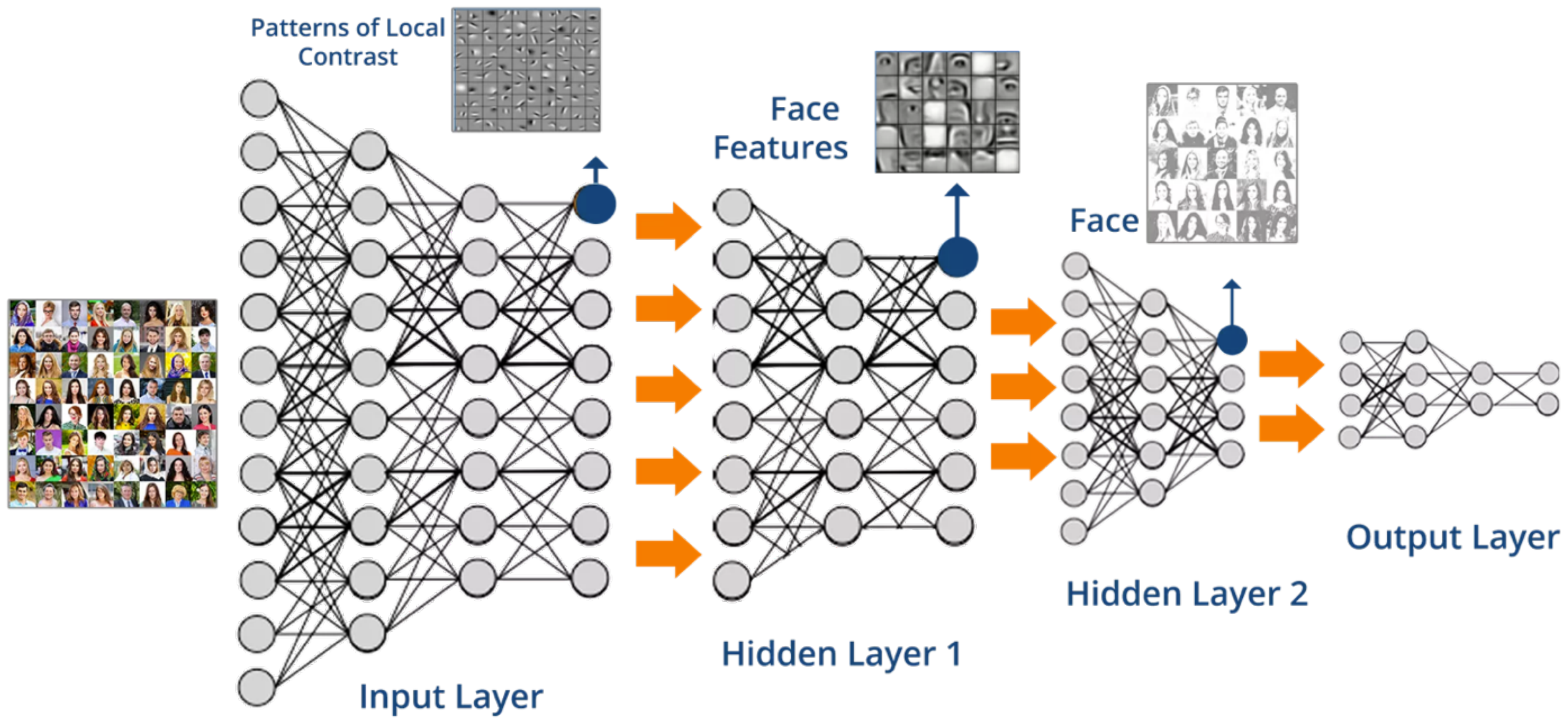
Machine Learning



Google TPUs, GraphCores, Nervana



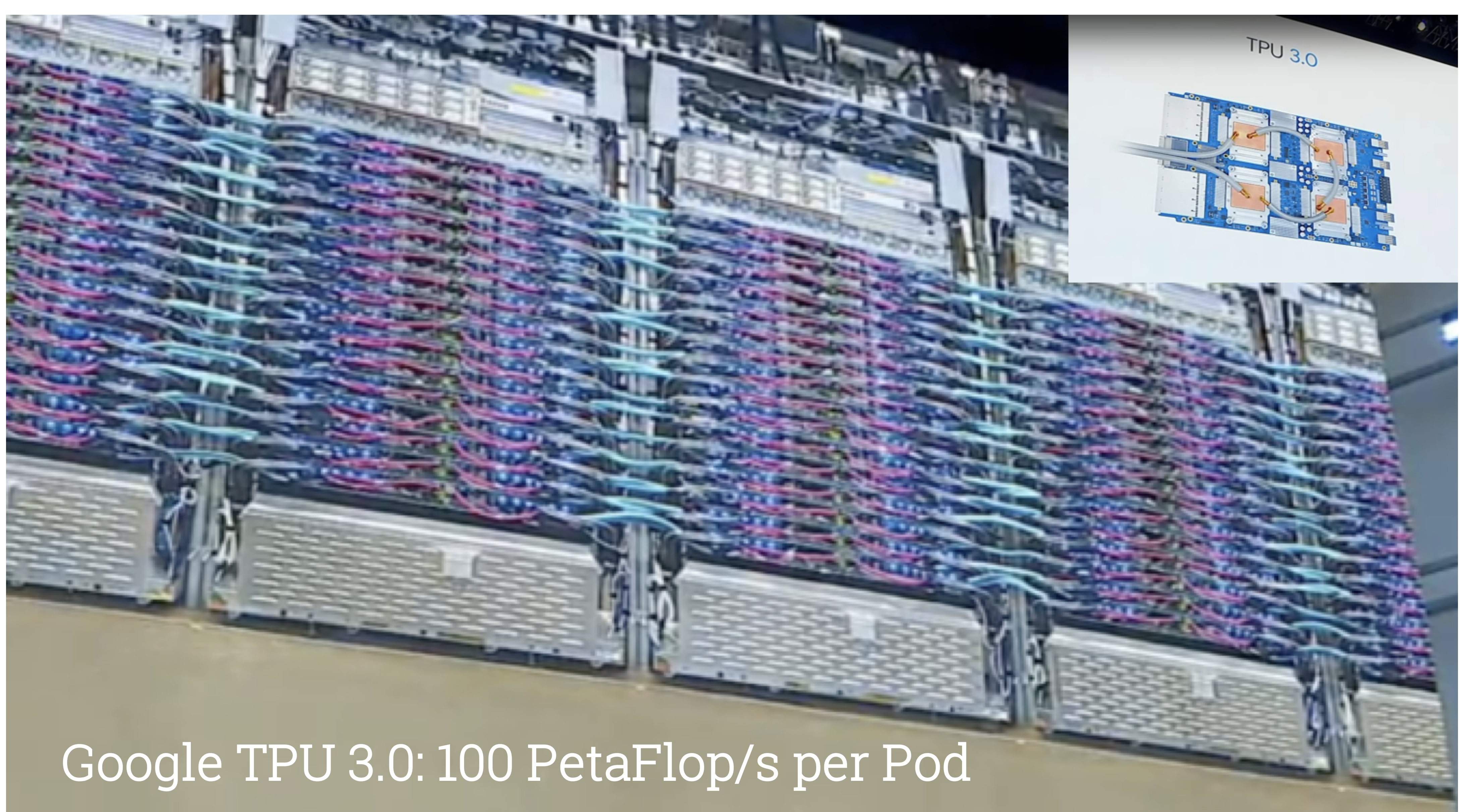
Machine Learning



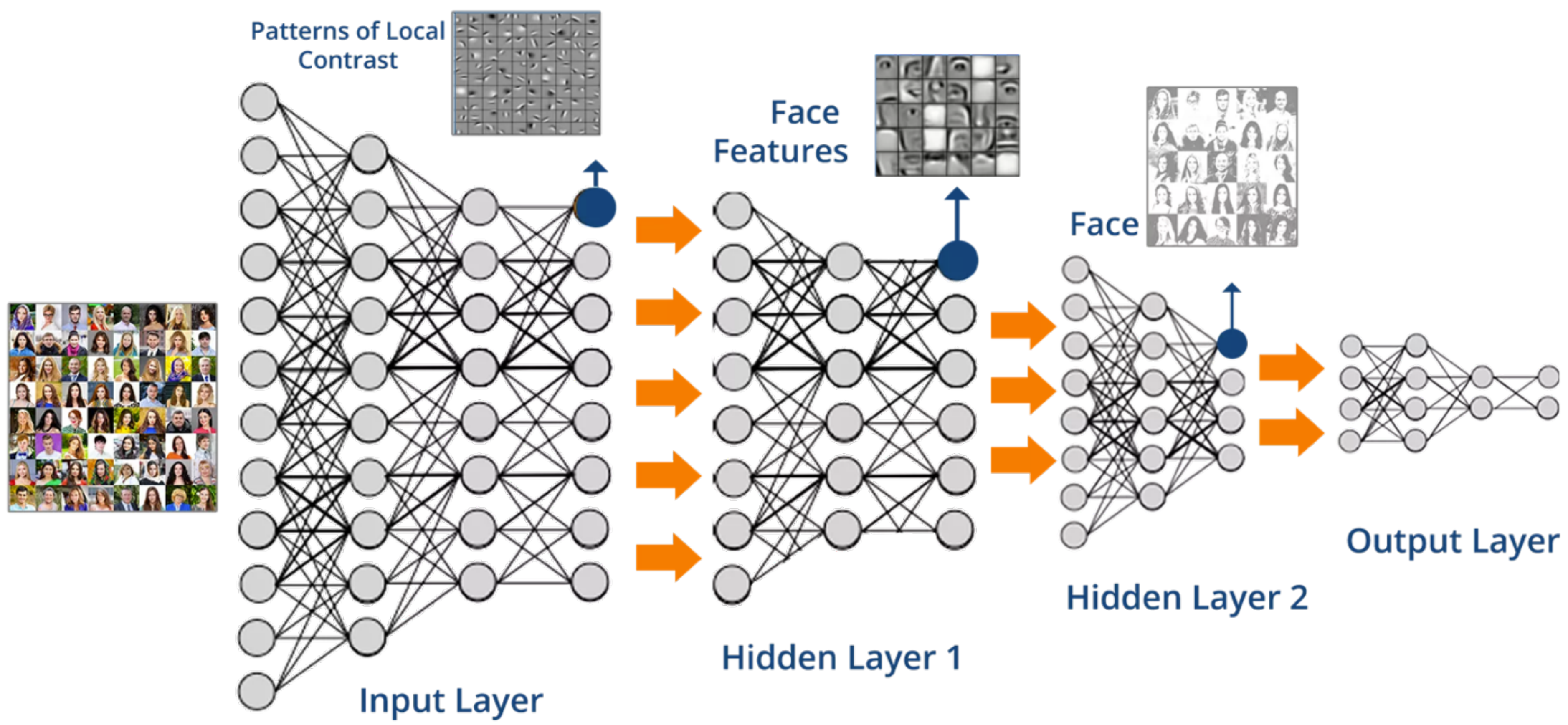
Google TPUs, GraphCores, Nervana

↓

Multiple dispatch, Generic programming



Machine Learning



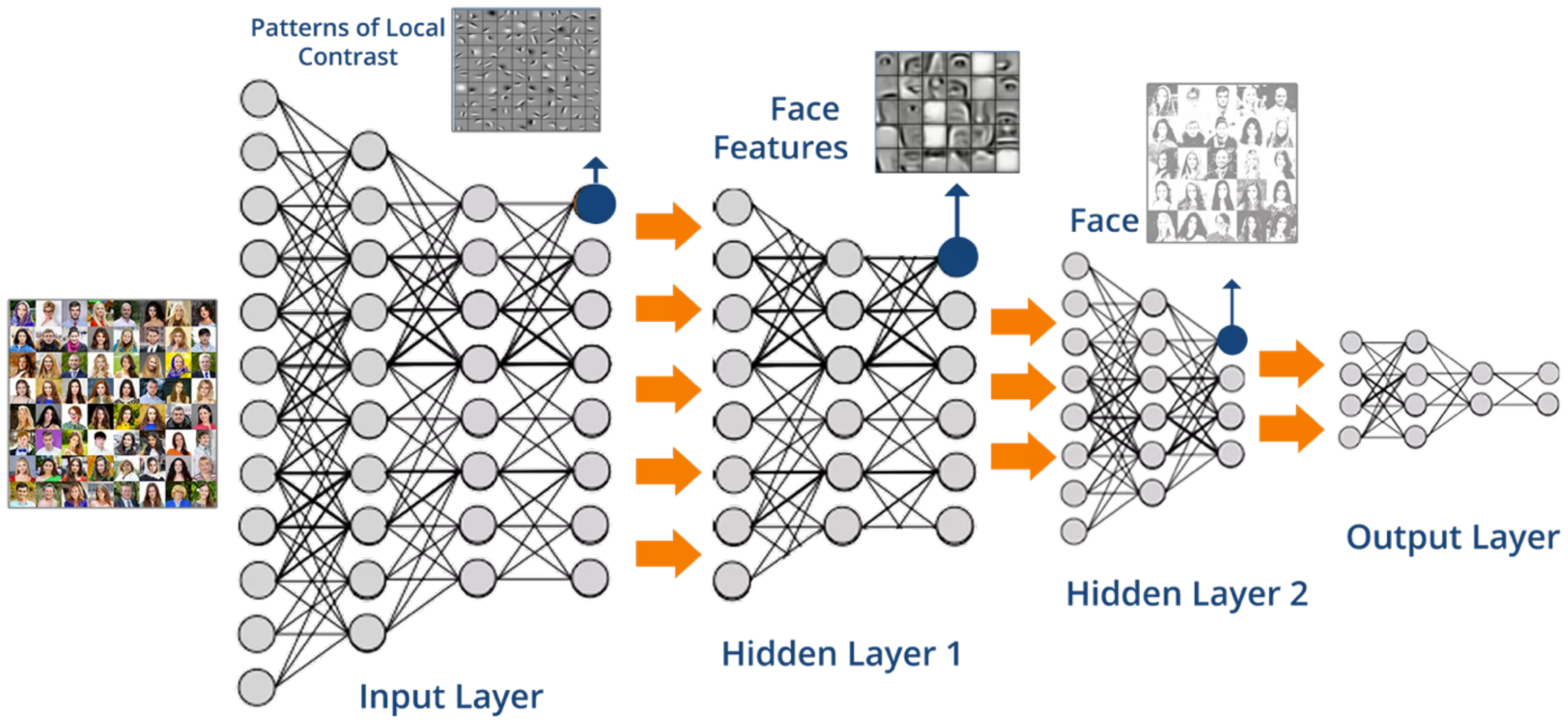
Google TPUs, GraphCores, Nervana

↓

Multiple dispatch, Generic programming

↓

Machine Learning



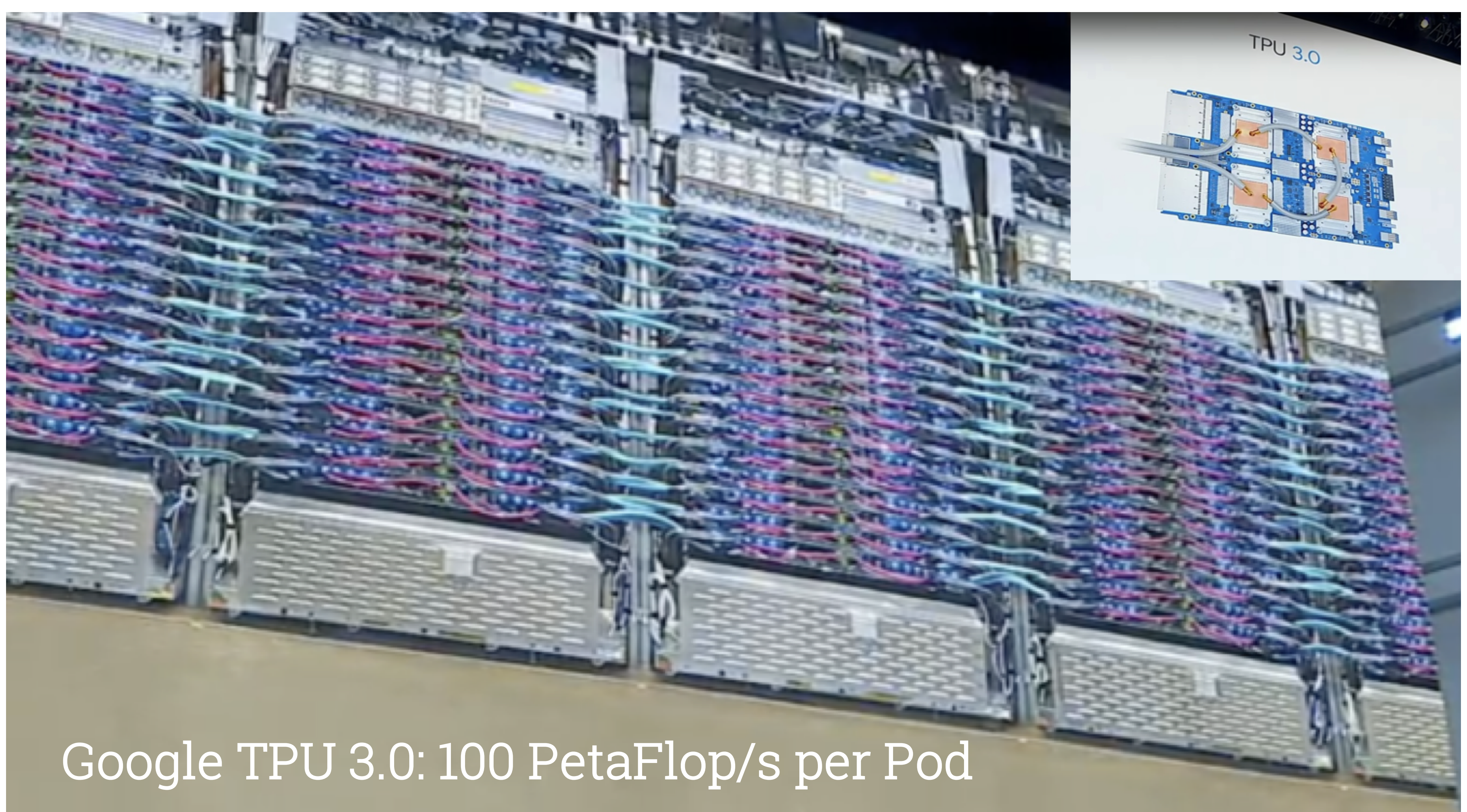
Google TPUs, GraphCores, Nervana

↓

Multiple dispatch, Generic programming

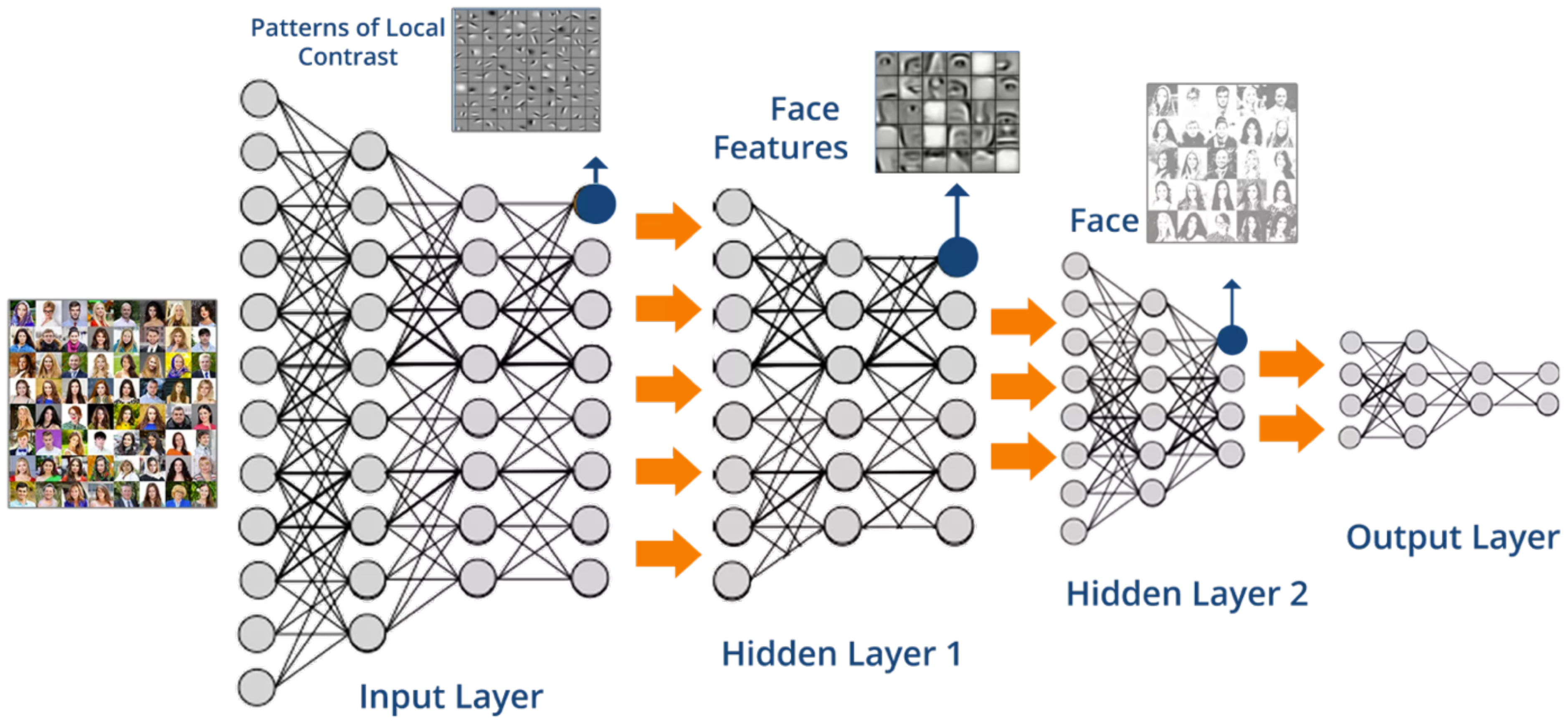
↓

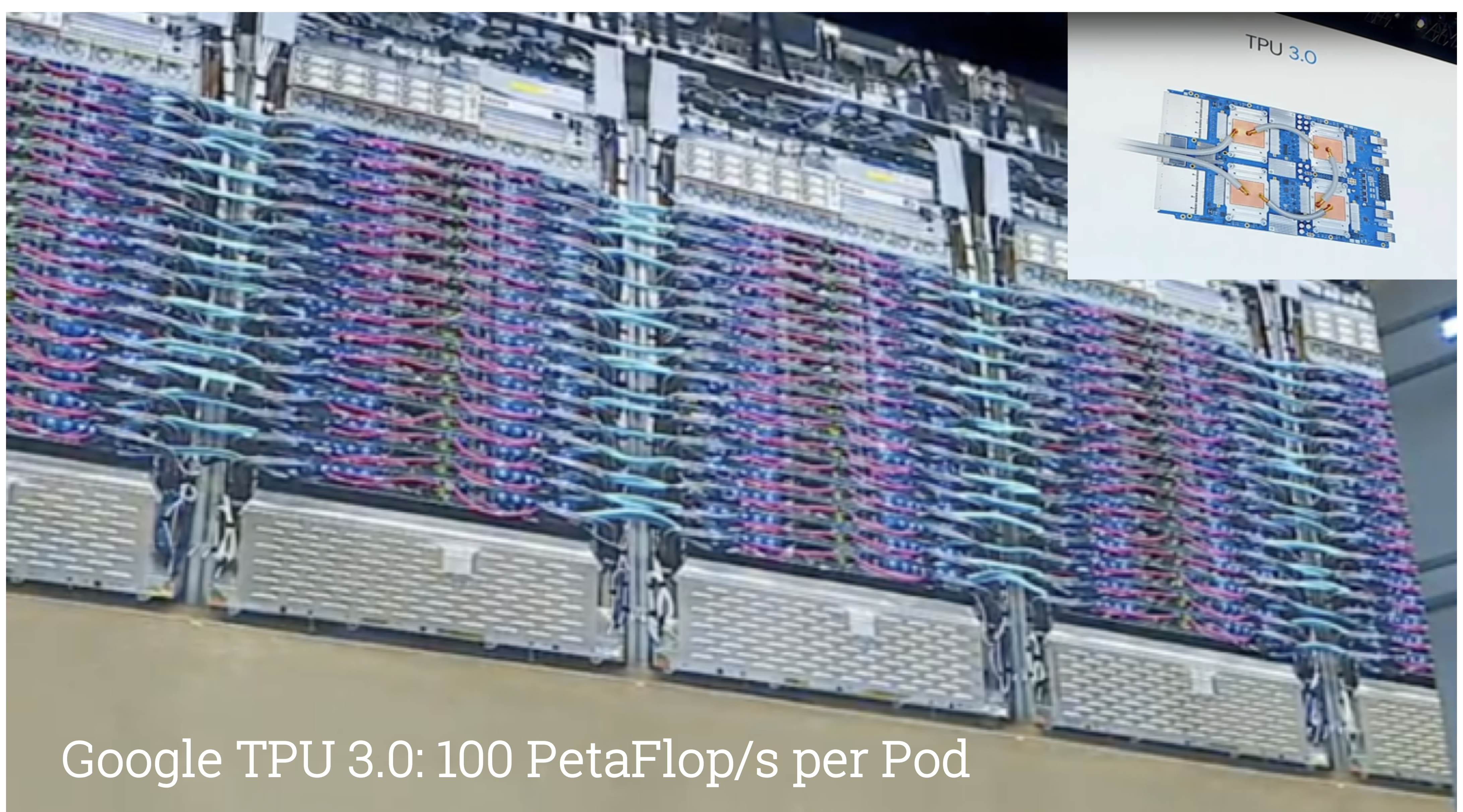
StaticArrays, Flux, Knet



Google TPU 3.0: 100 PetaFlop/s per Pod

Machine Learning





Google TPU 3.0: 100 PetaFlop/s per Pod

Google TPUs, GraphCores, Nervana

↓

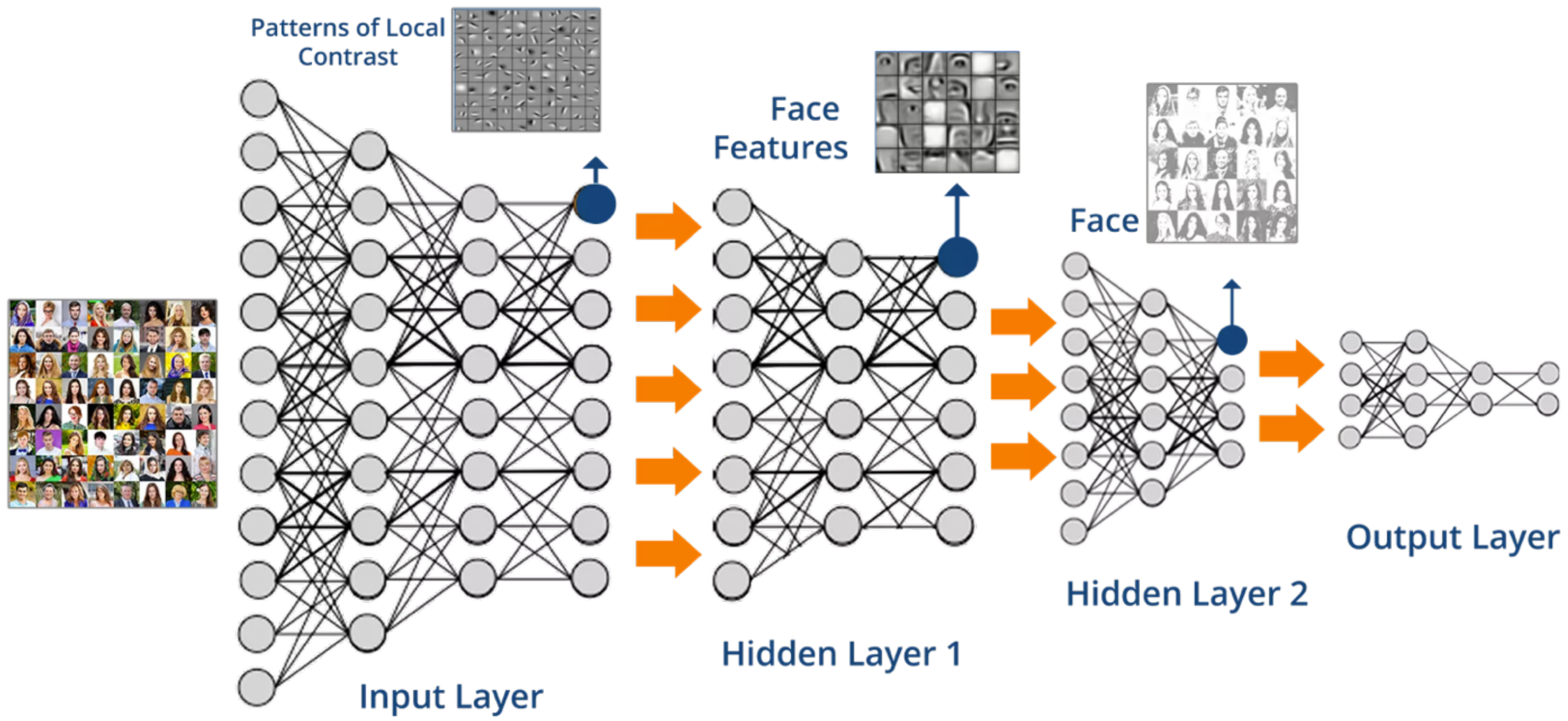
Multiple dispatch, Generic programming

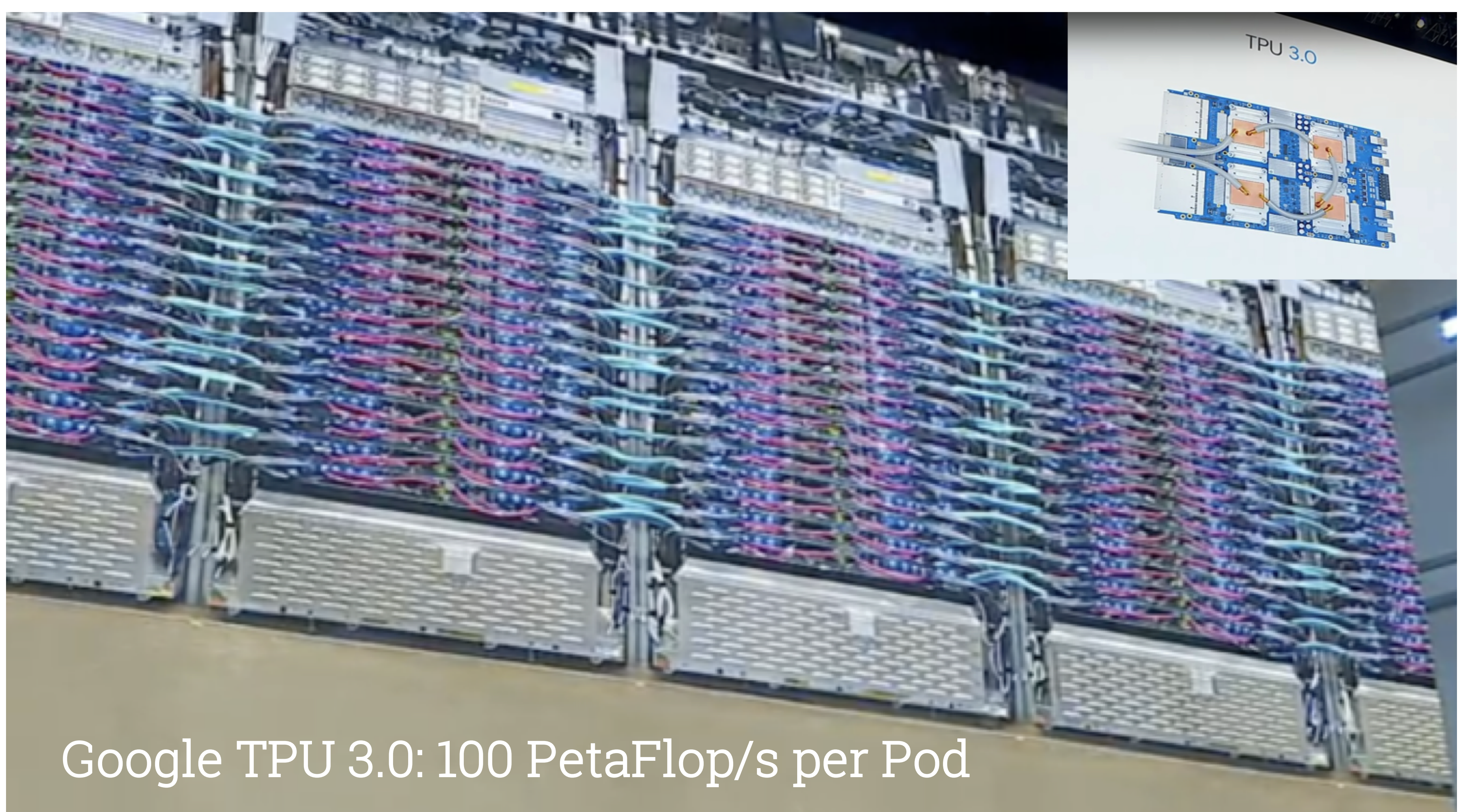
↓

StaticArrays, Flux, Knet

↓

Machine Learning





Google TPUs, GraphCores, Nervana

↓

Multiple dispatch, Generic programming

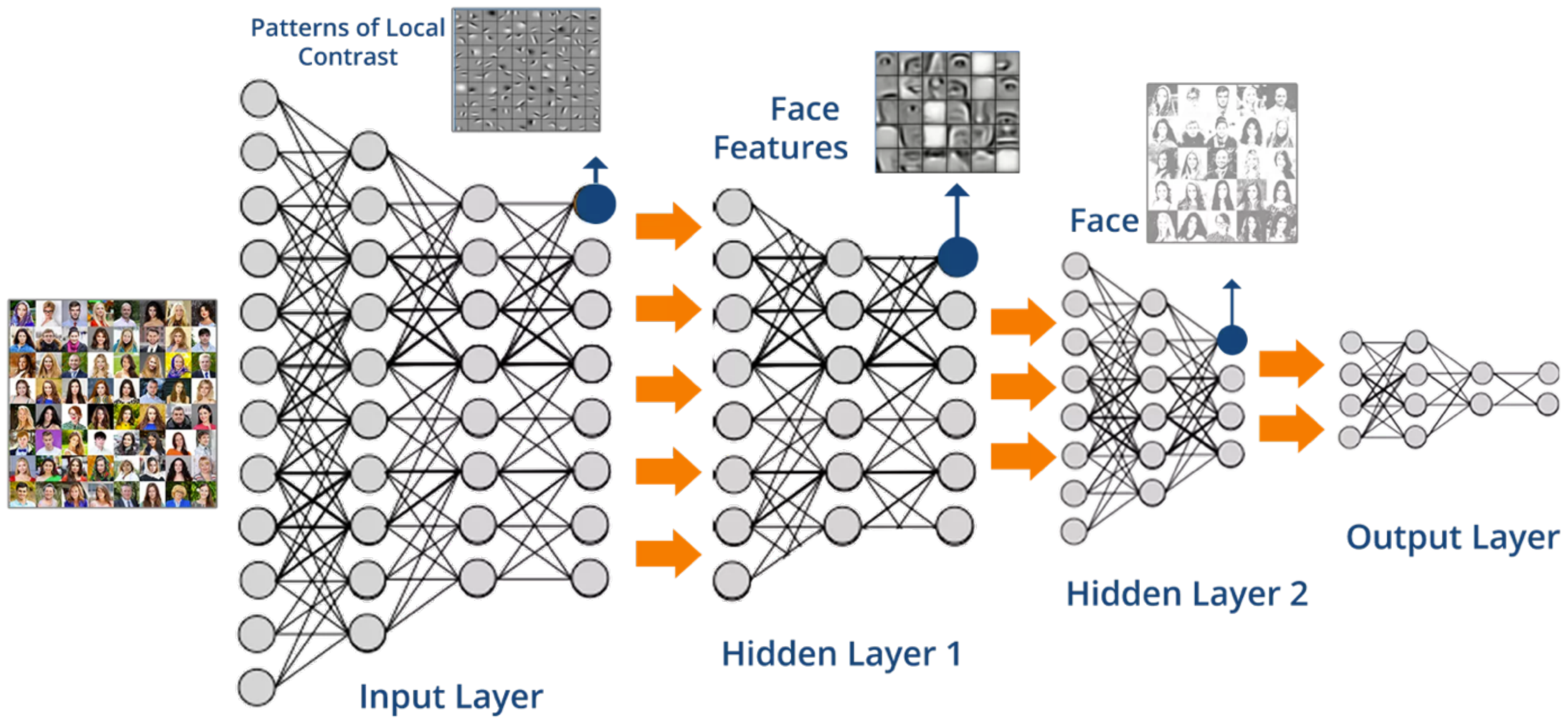
↓

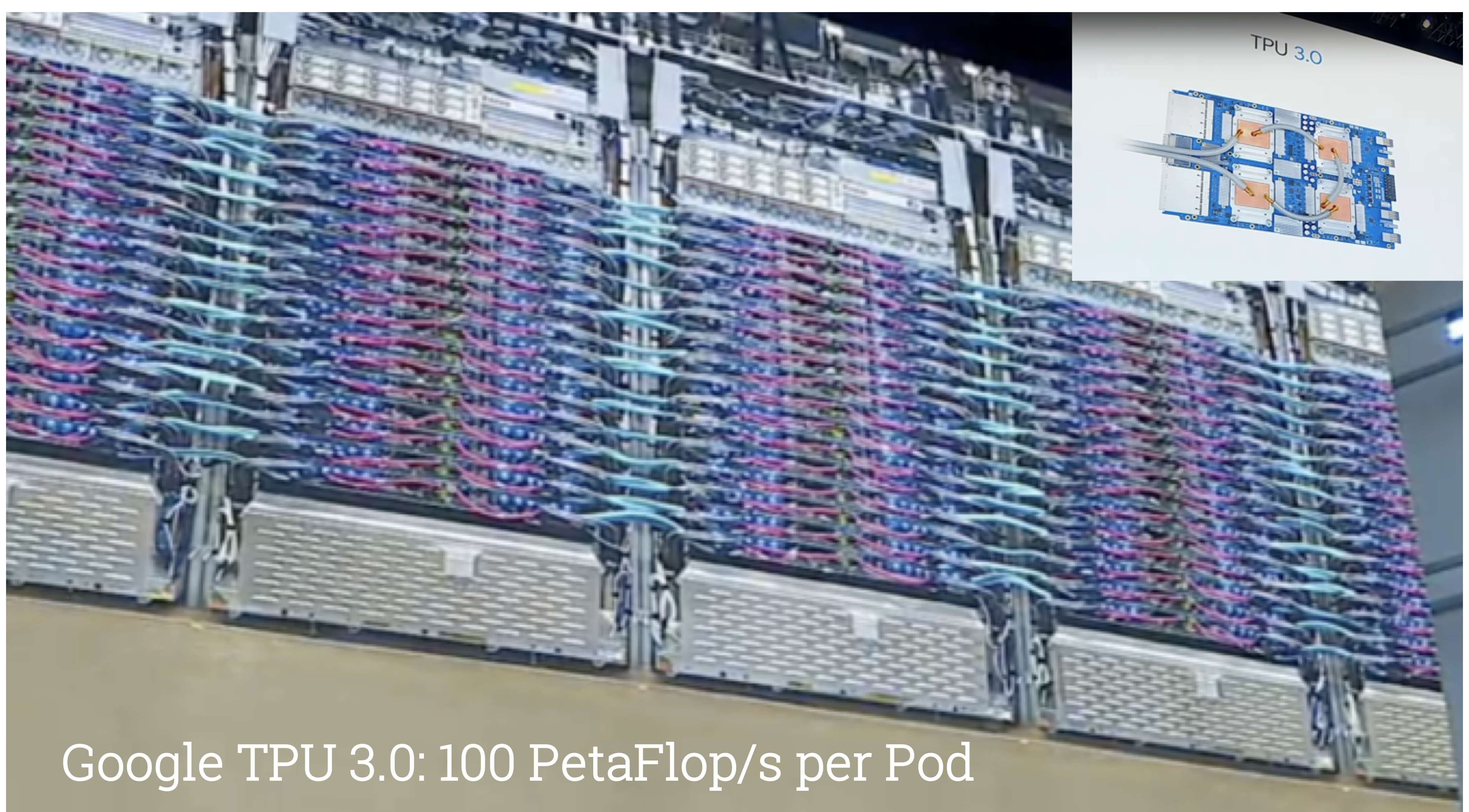
StaticArrays, Flux, Knet

↓

Automatic Differentiation, Optimization

Machine Learning





Google TPU 3.0: 100 PetaFlop/s per Pod

Google TPUs, GraphCores, Nervana

↓

Multiple dispatch, Generic programming

↓

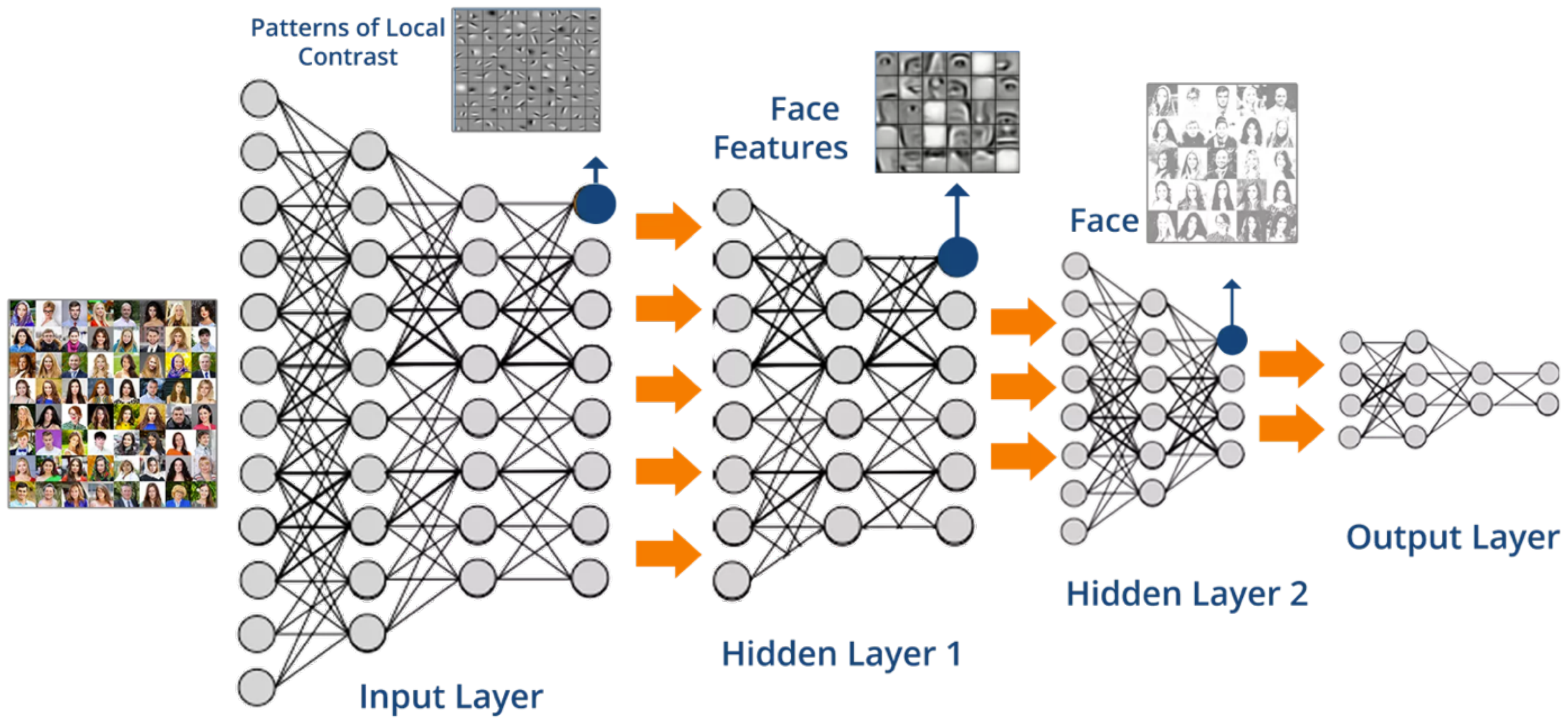
StaticArrays, Flux, Knet

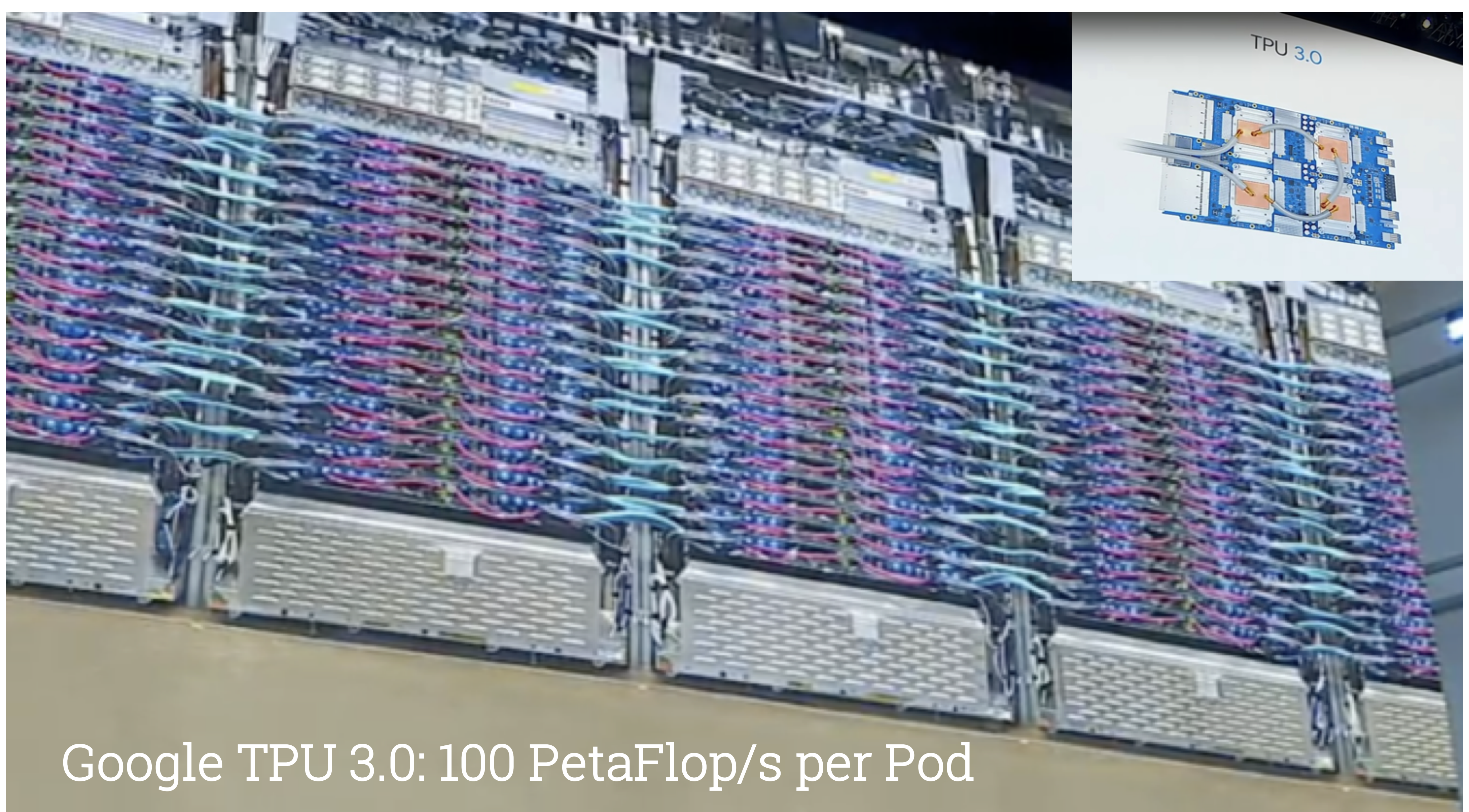
↓

Automatic Differentiation, Optimization

↓

Machine Learning





Google TPU 3.0: 100 PetaFlop/s per Pod

Google TPUs, GraphCores, Nervana

↓

Multiple dispatch, Generic programming

↓

StaticArrays, Flux, Knet

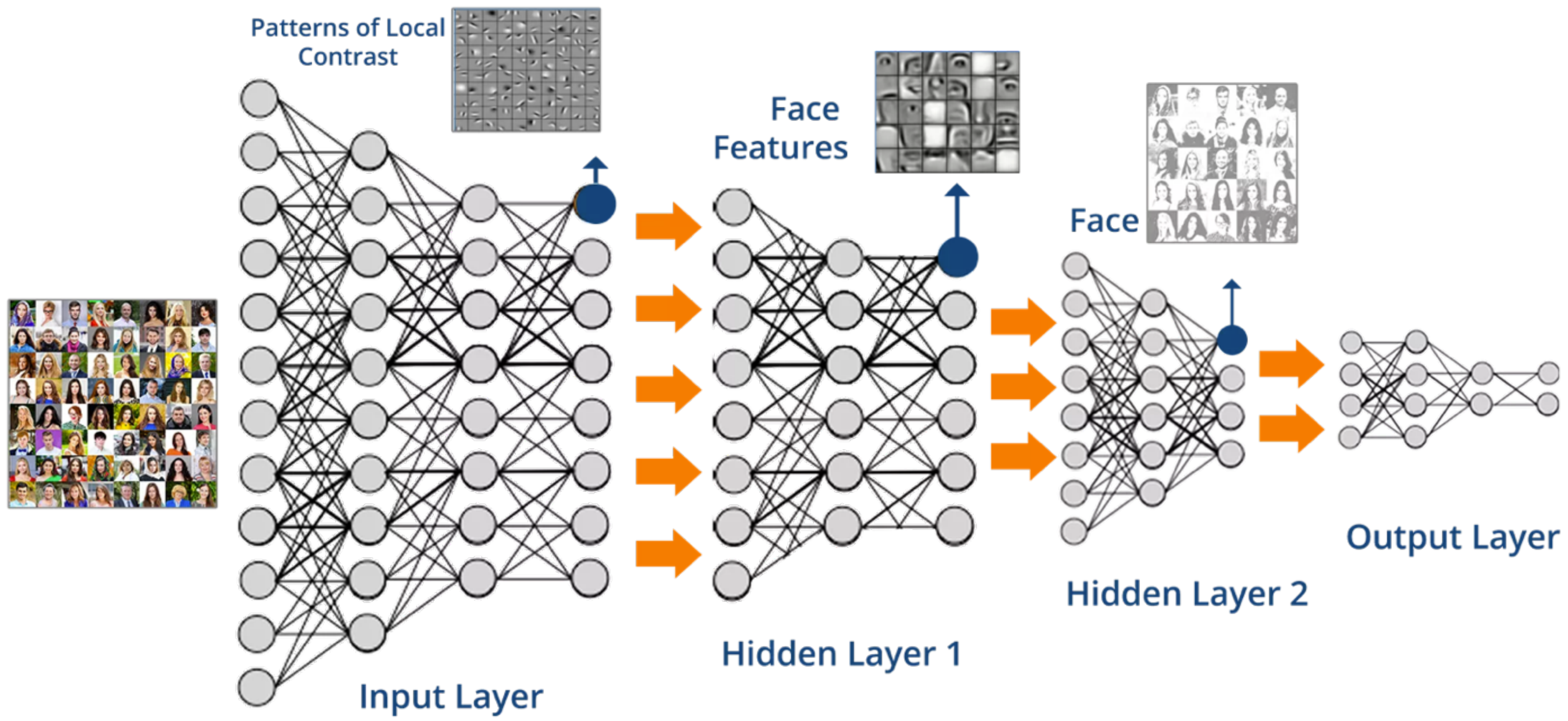
↓

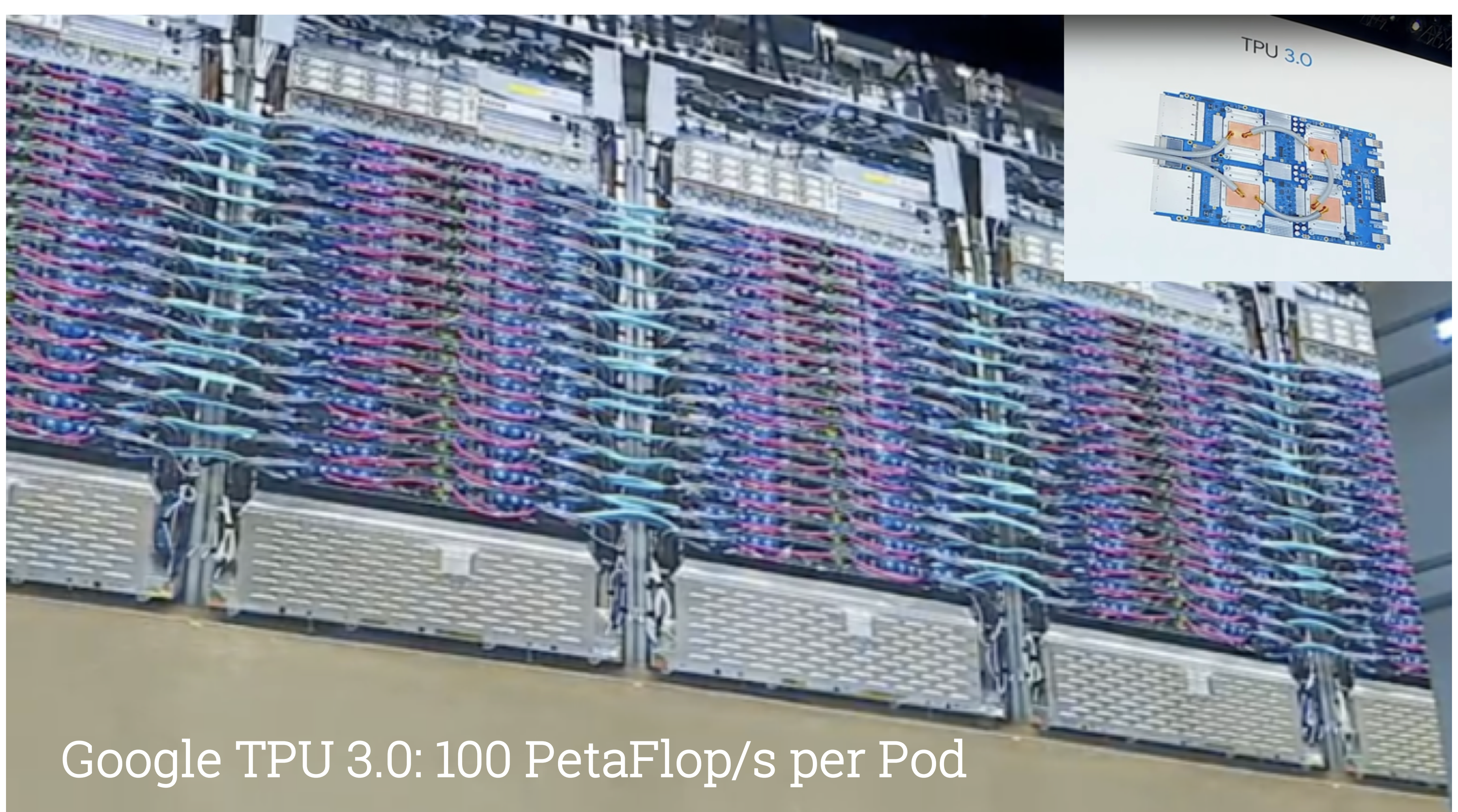
Automatic Differentiation, Optimization

↓

Metalhead

Machine Learning





Google TPUs, GraphCores, Nervana

↓

Multiple dispatch, Generic programming

↓

StaticArrays, Flux, Knet

↓

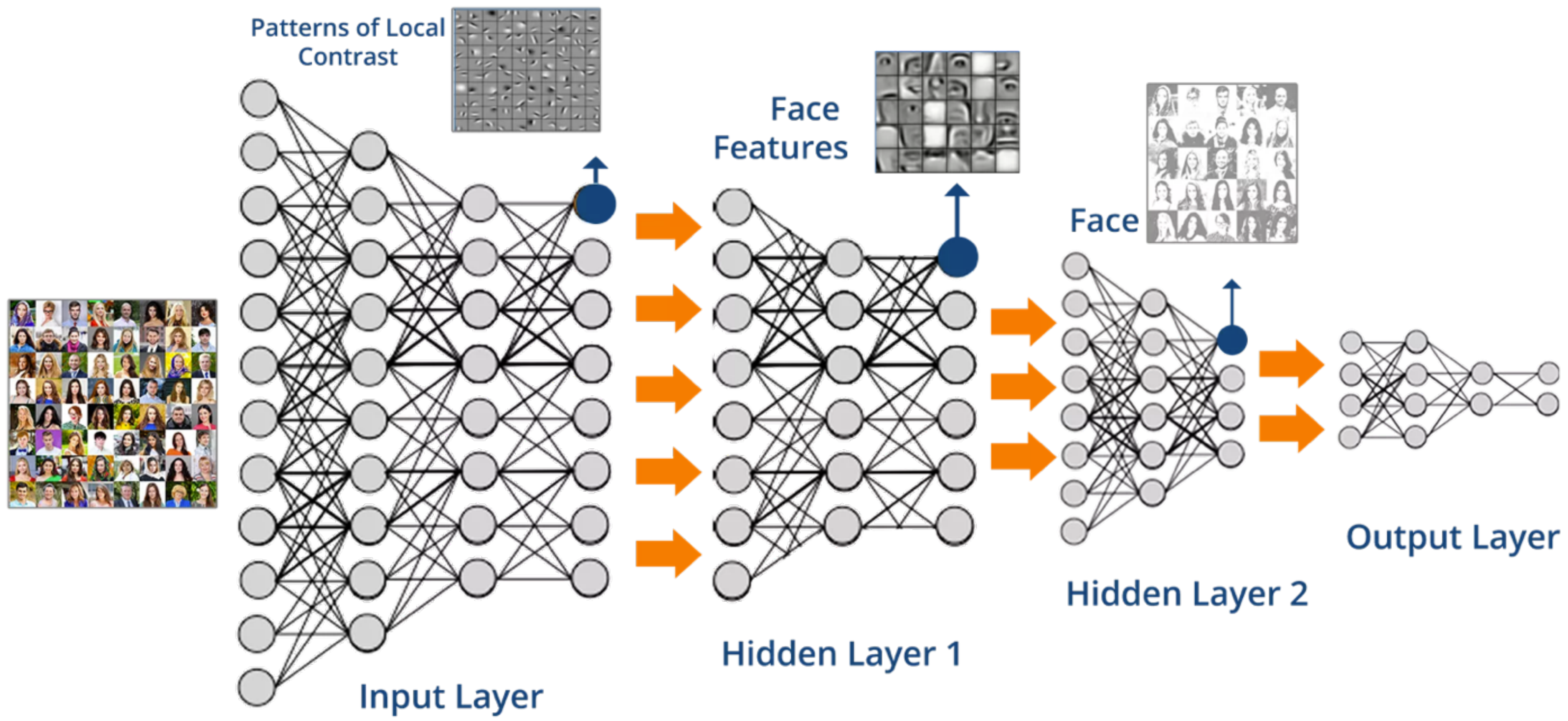
Automatic Differentiation, Optimization

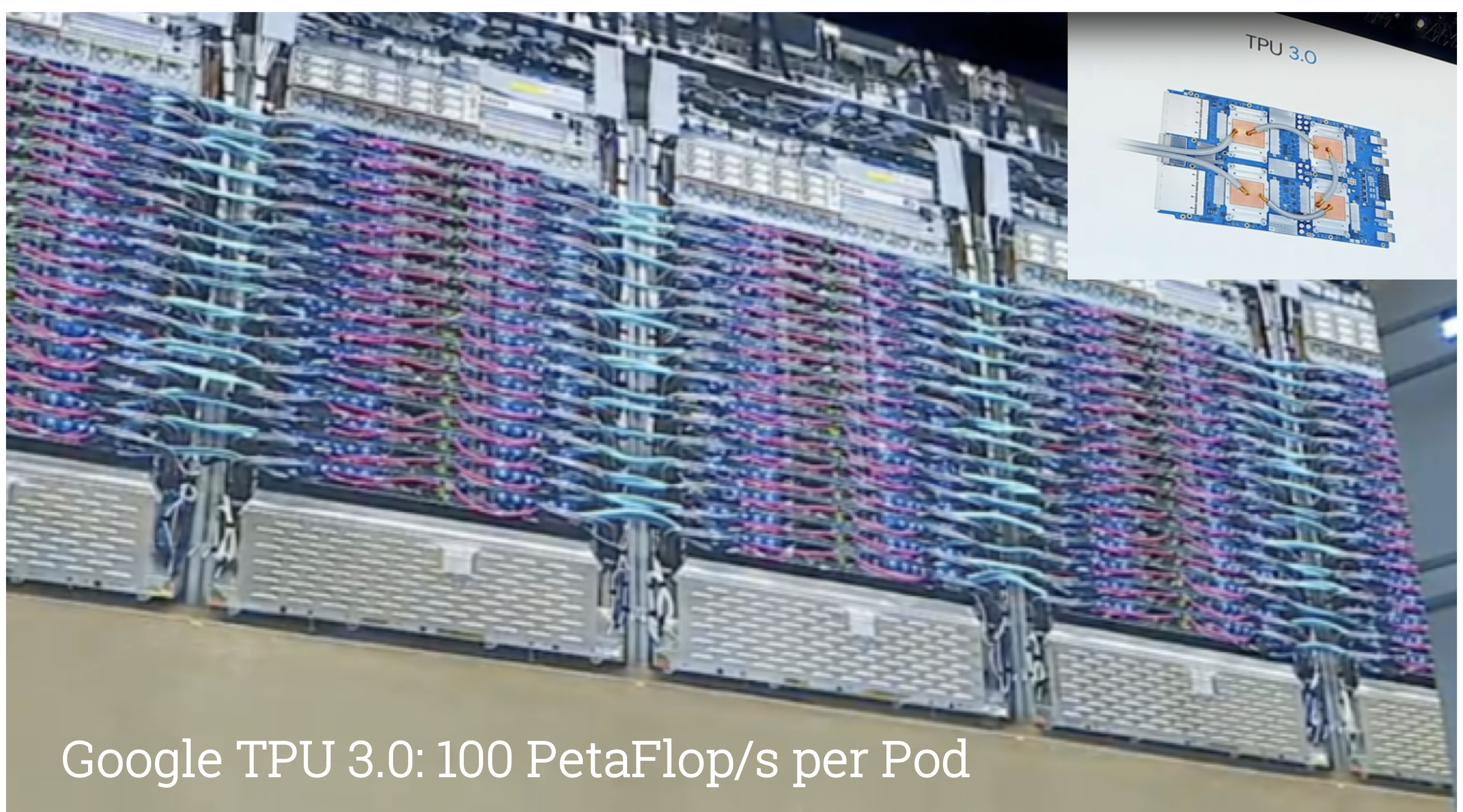
↓

Metalhead

↓

Machine Learning





Google TPUs, GraphCores, Nervana

↓

Multiple dispatch, Generic programming

↓

StaticArrays, Flux, Knet

↓

Automatic Differentiation, Optimization

↓

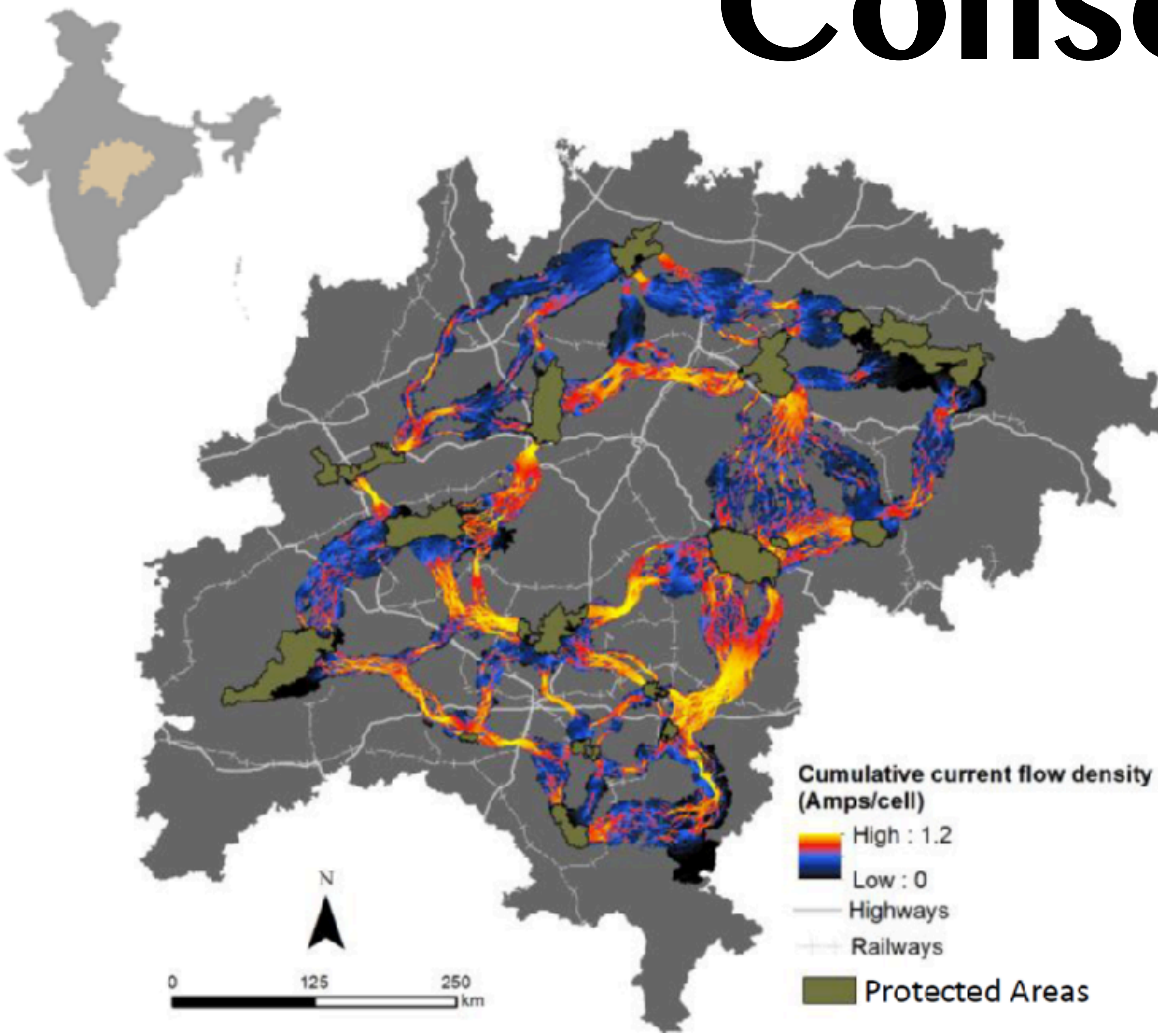
Metalhead

↓

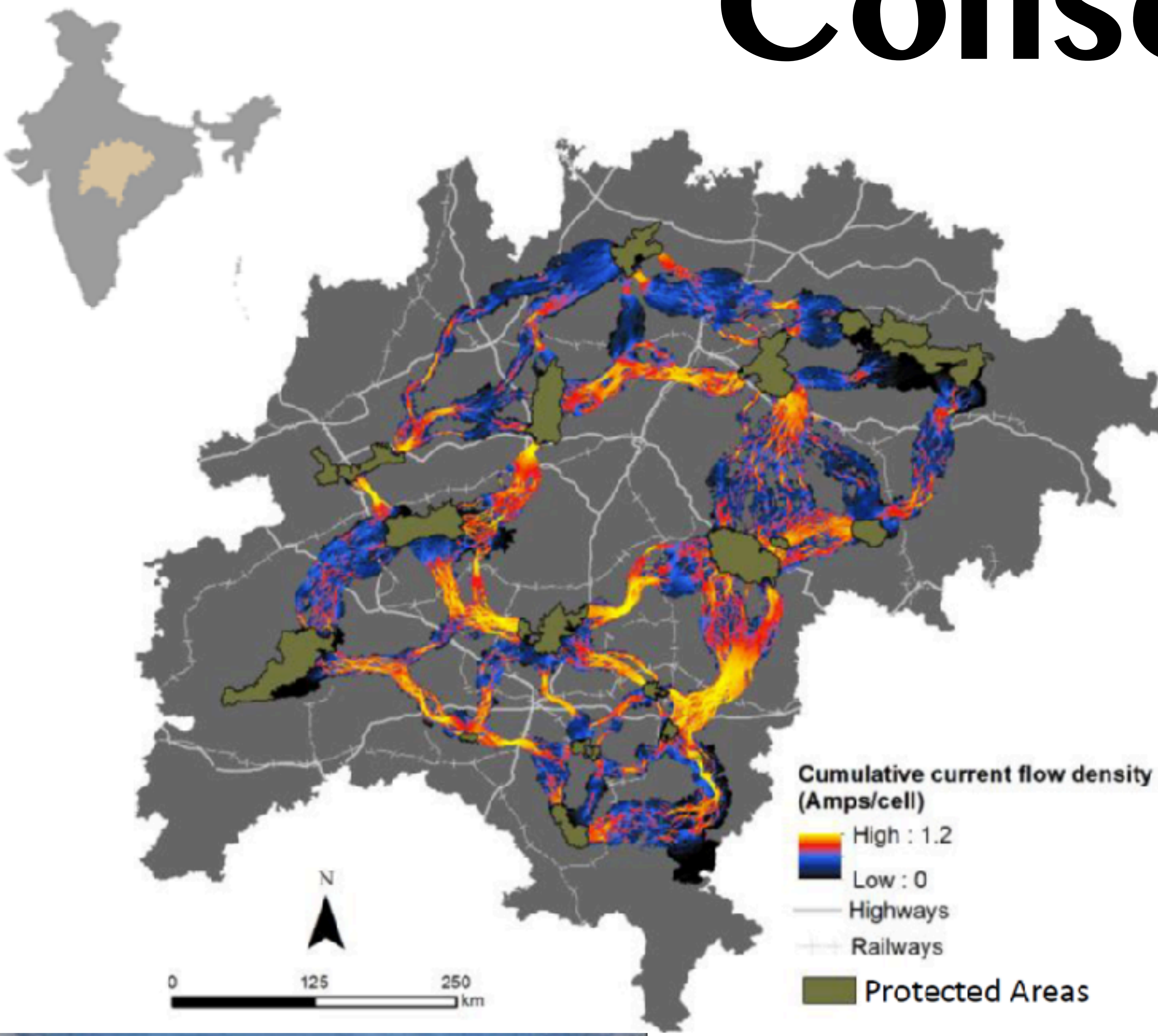
Images, Speech, Text, Autonomy

Conservation

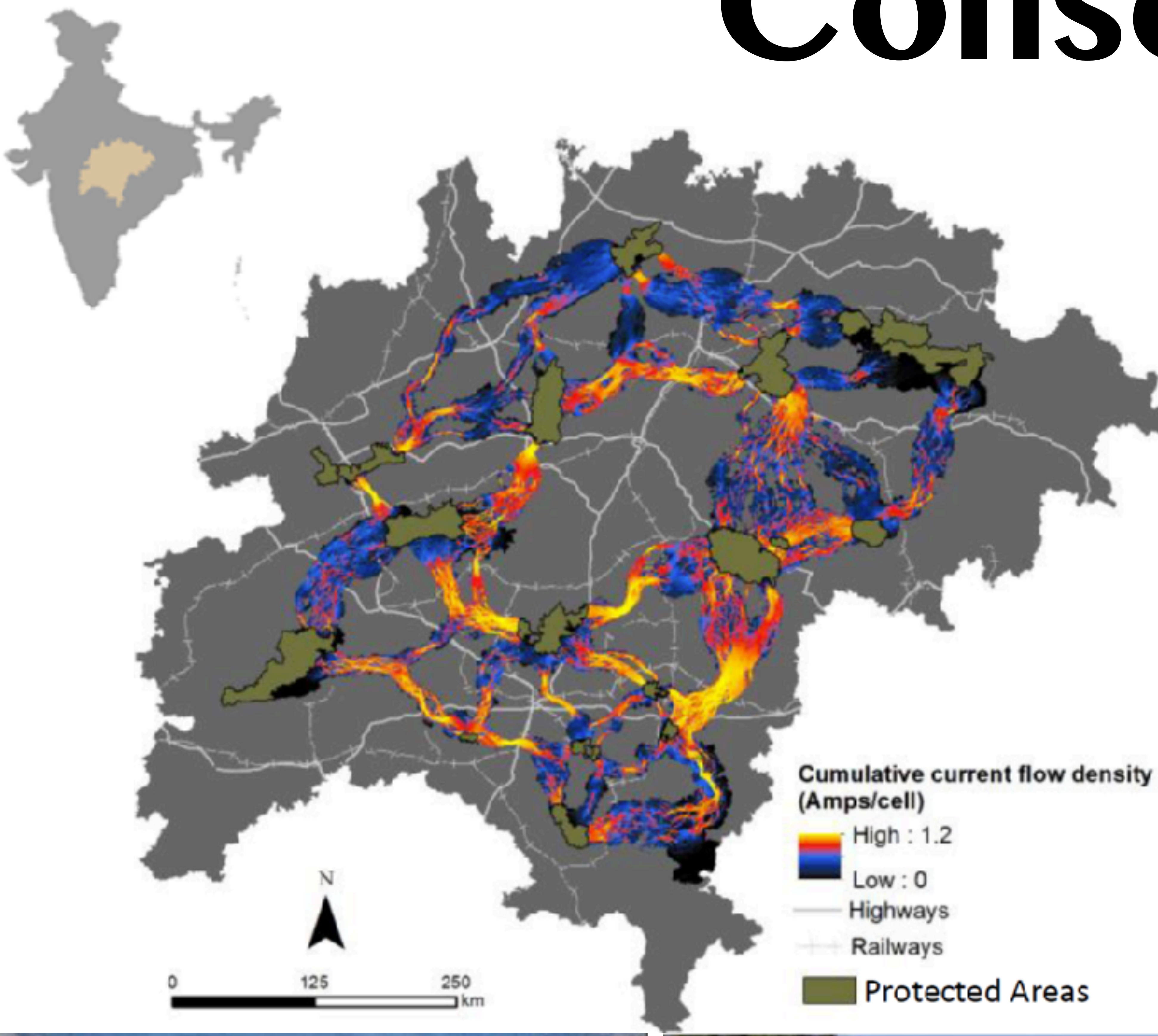
Conservation



Conservation

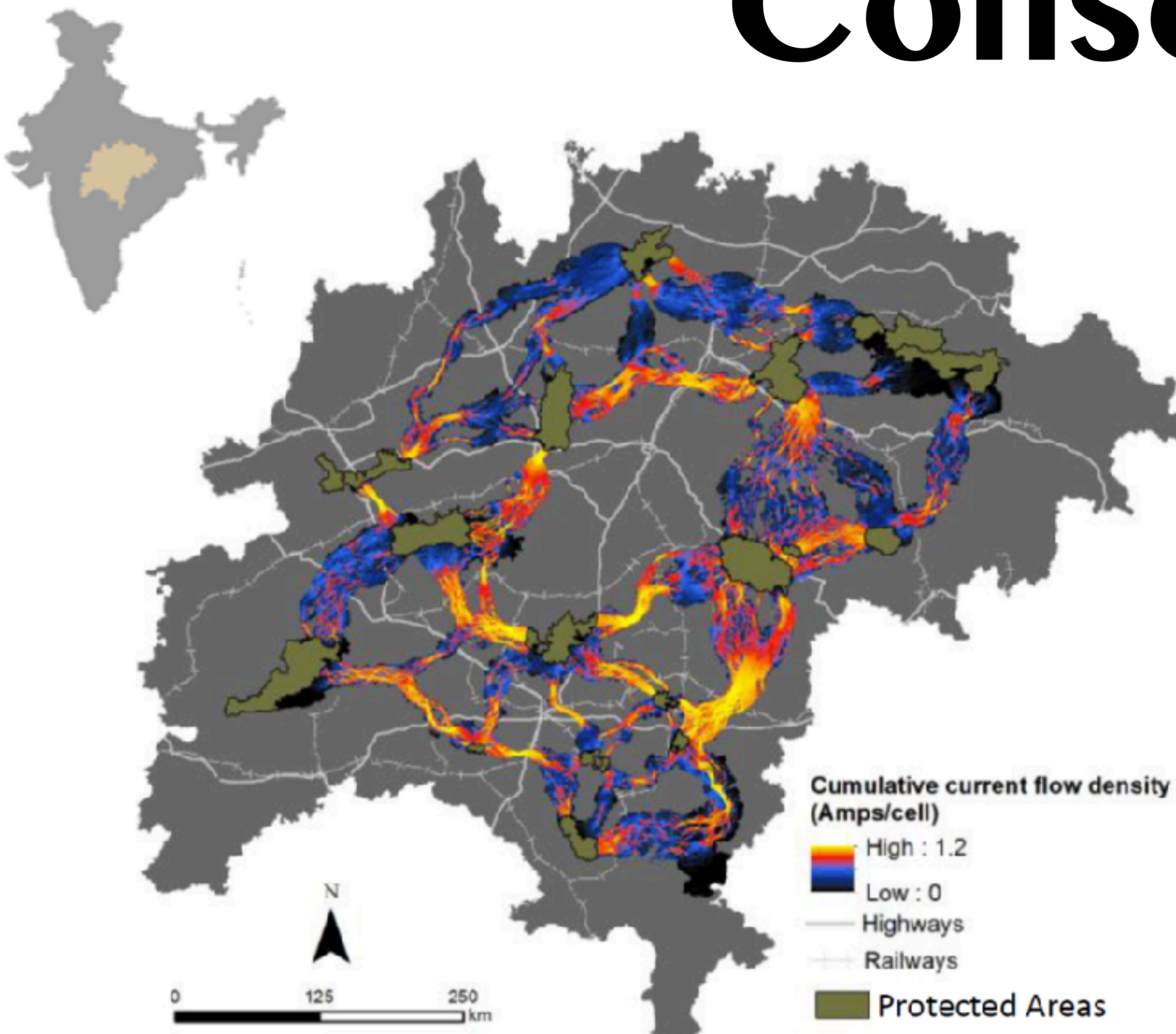


Conservation



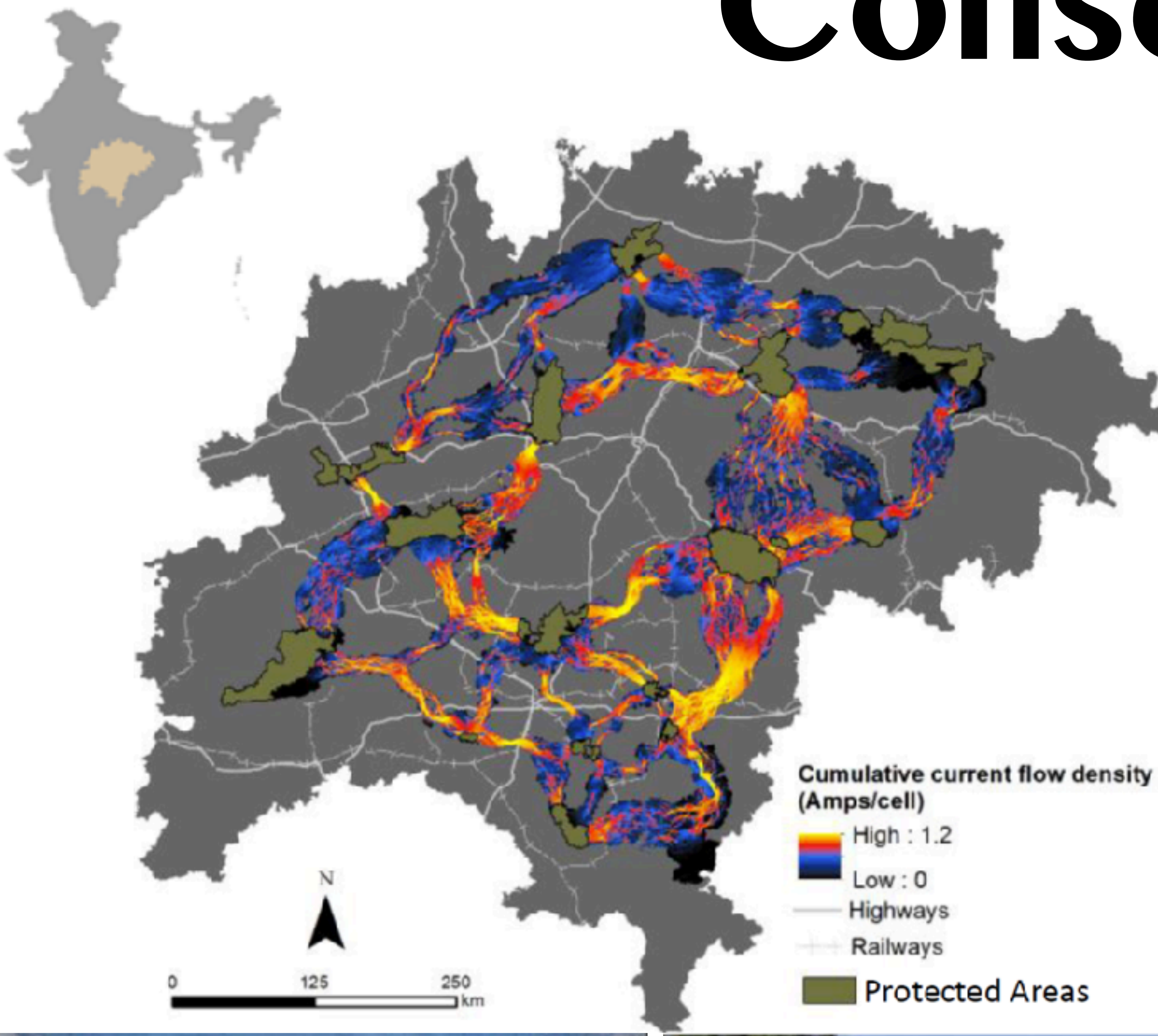
Conservation

CPUs



Conservation

CPUs

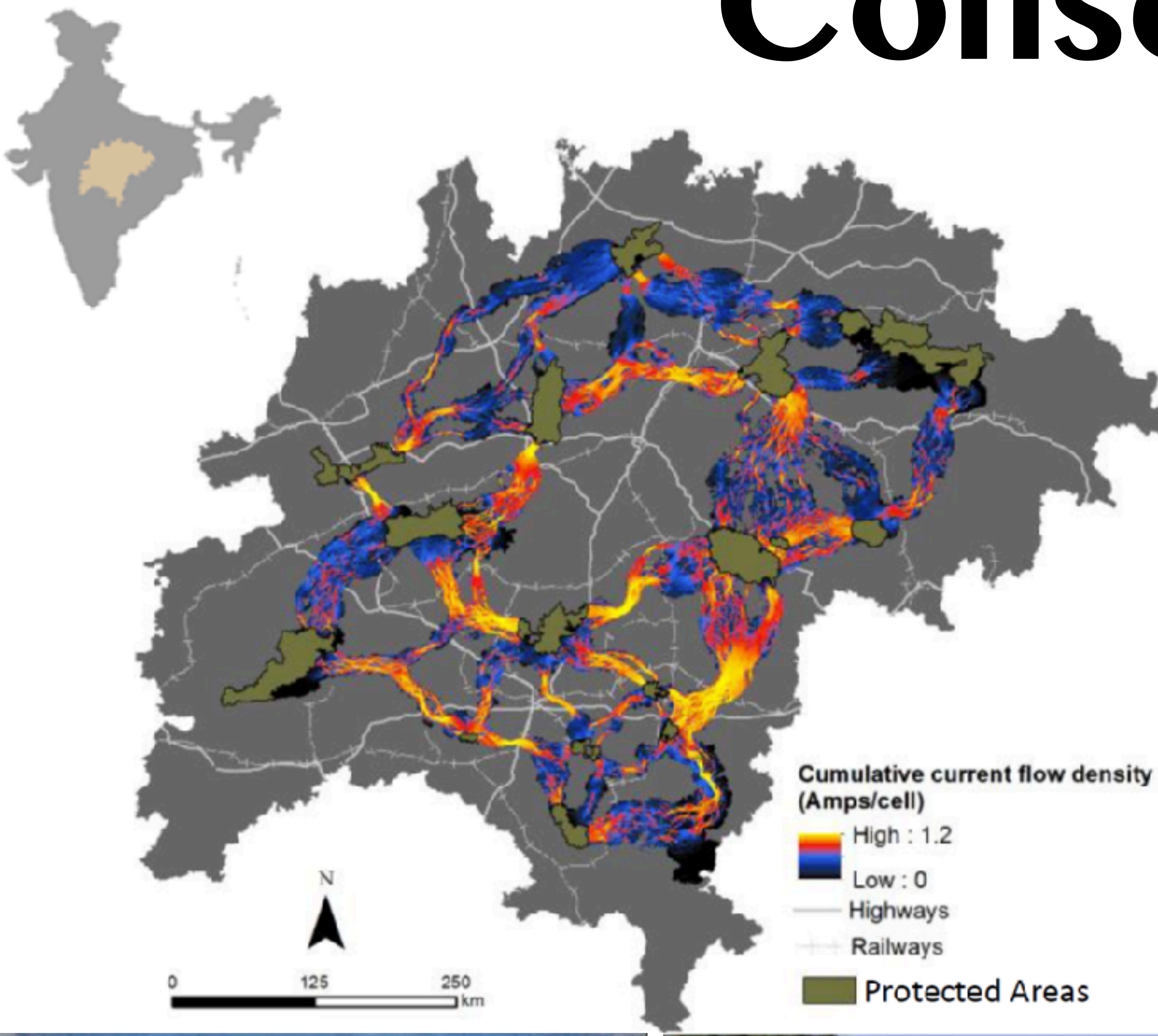


Conservation

CPUs

↓

Multiple Dispatch, Generic programming



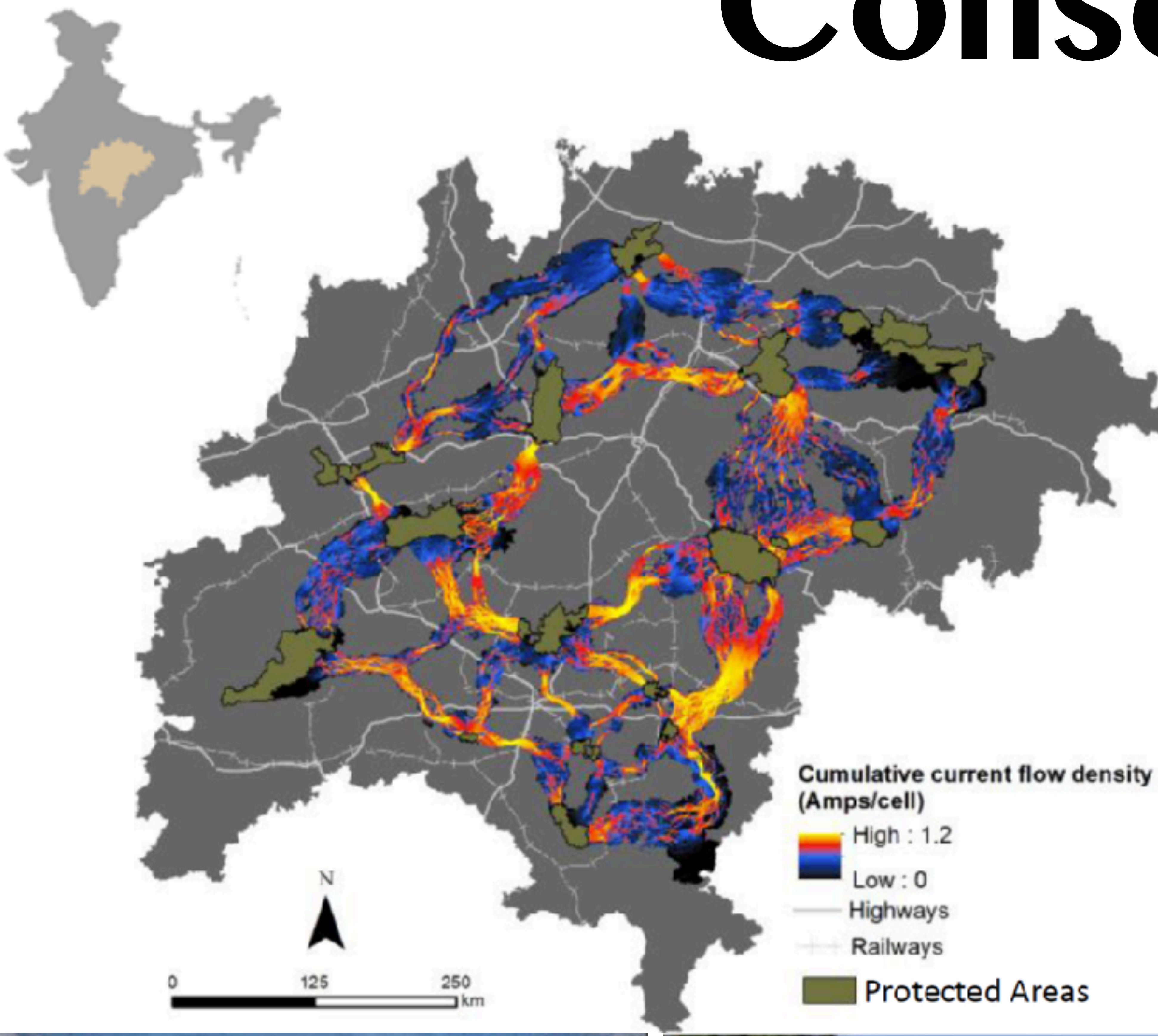
Conservation

CPUs

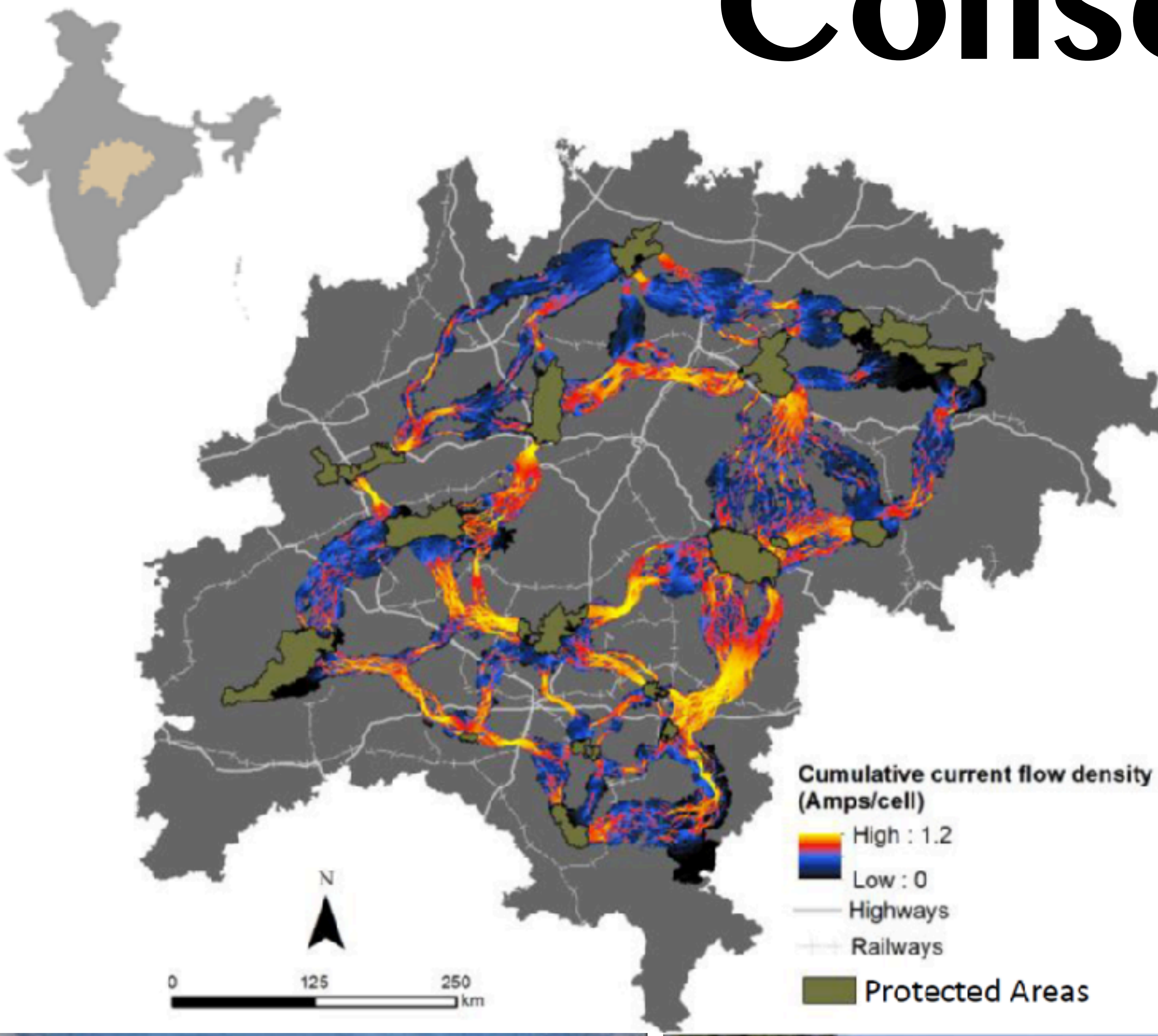
↓

Multiple Dispatch, Generic programming

↓



Conservation



CPUs

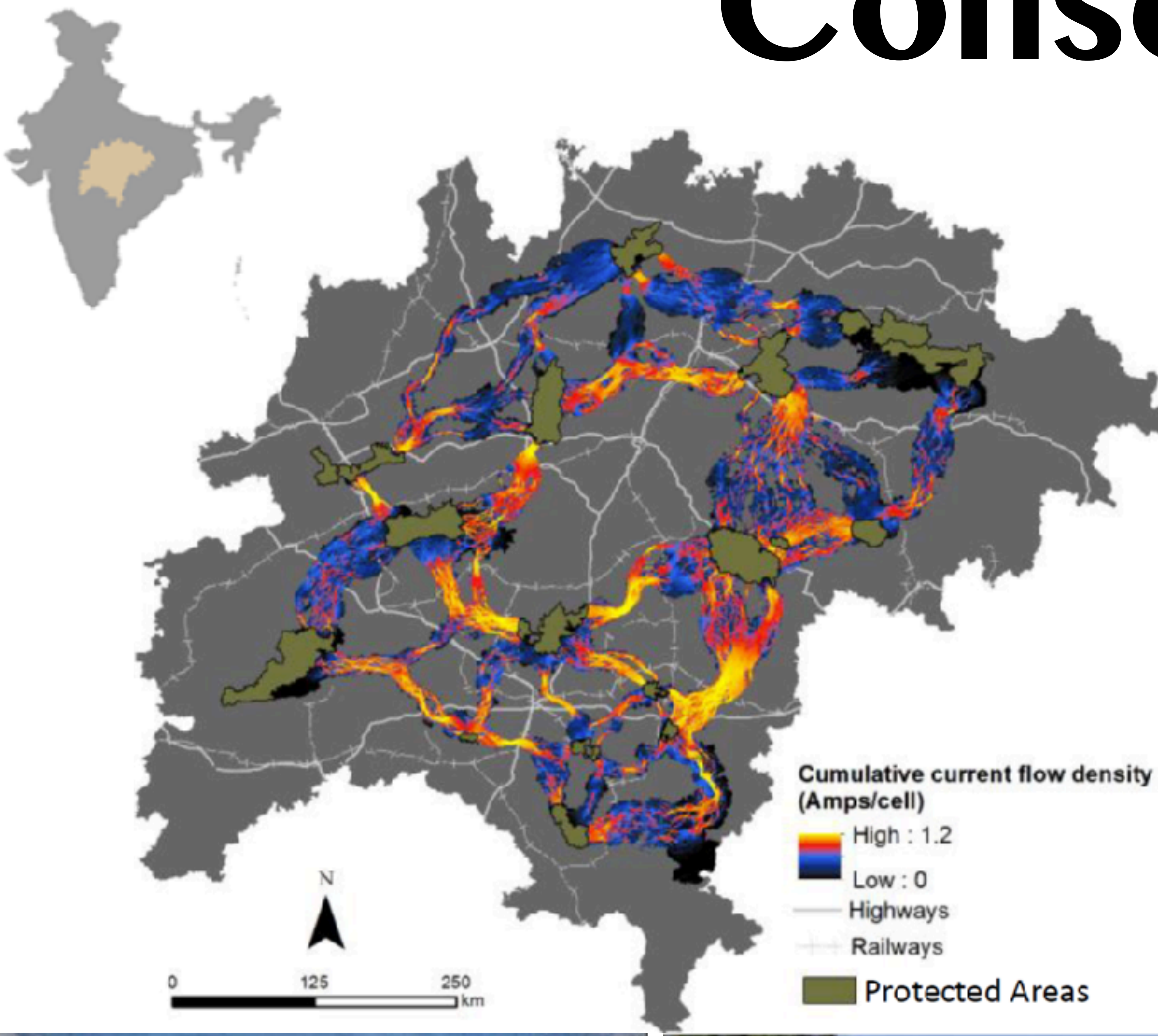
↓

Multiple Dispatch, Generic programming

↓

Sparse Matrices

Conservation



CPUs

↓

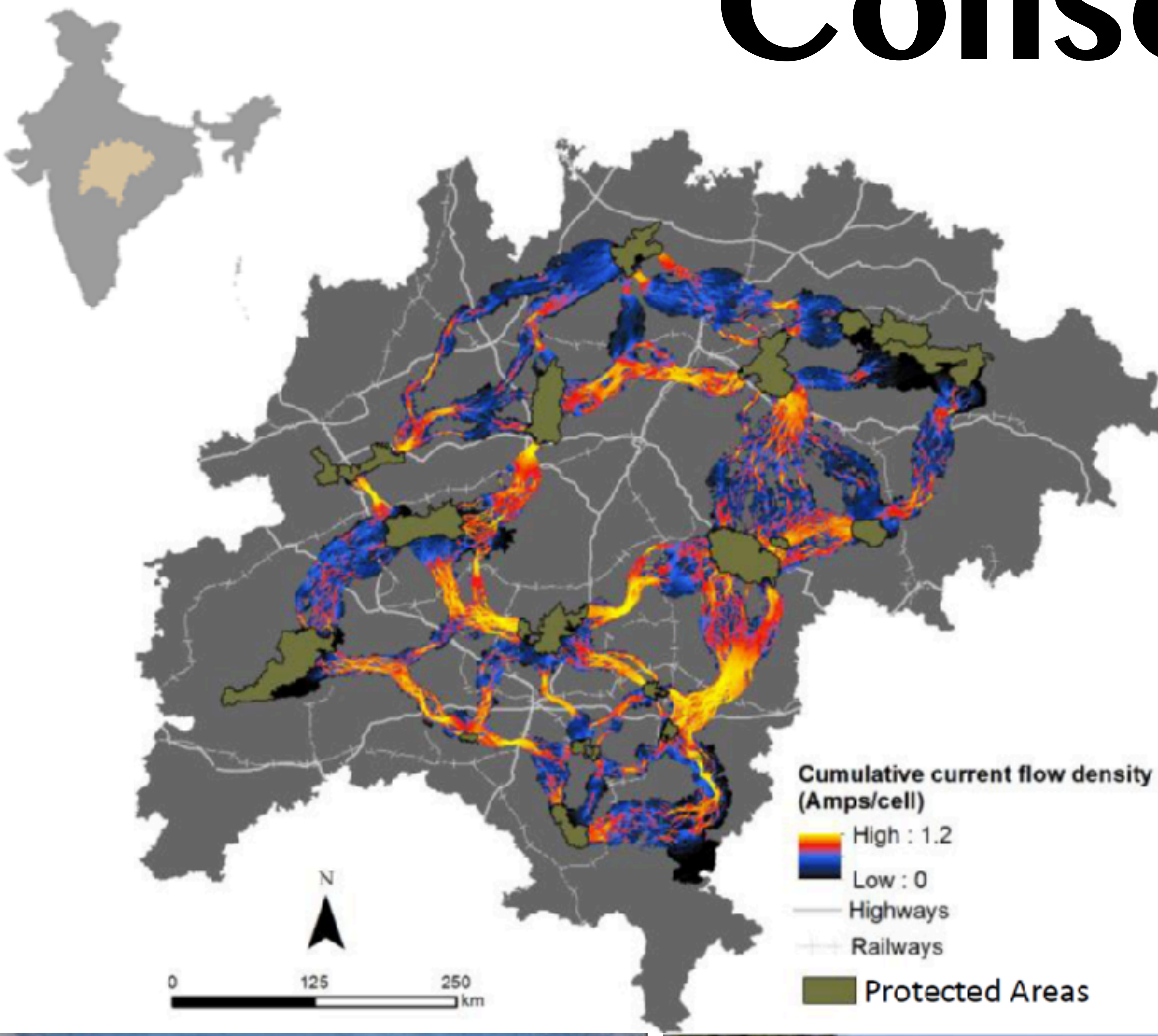
Multiple Dispatch, Generic programming

↓

Sparse Matrices

↓

Conservation



CPUs

↓

Multiple Dispatch, Generic programming

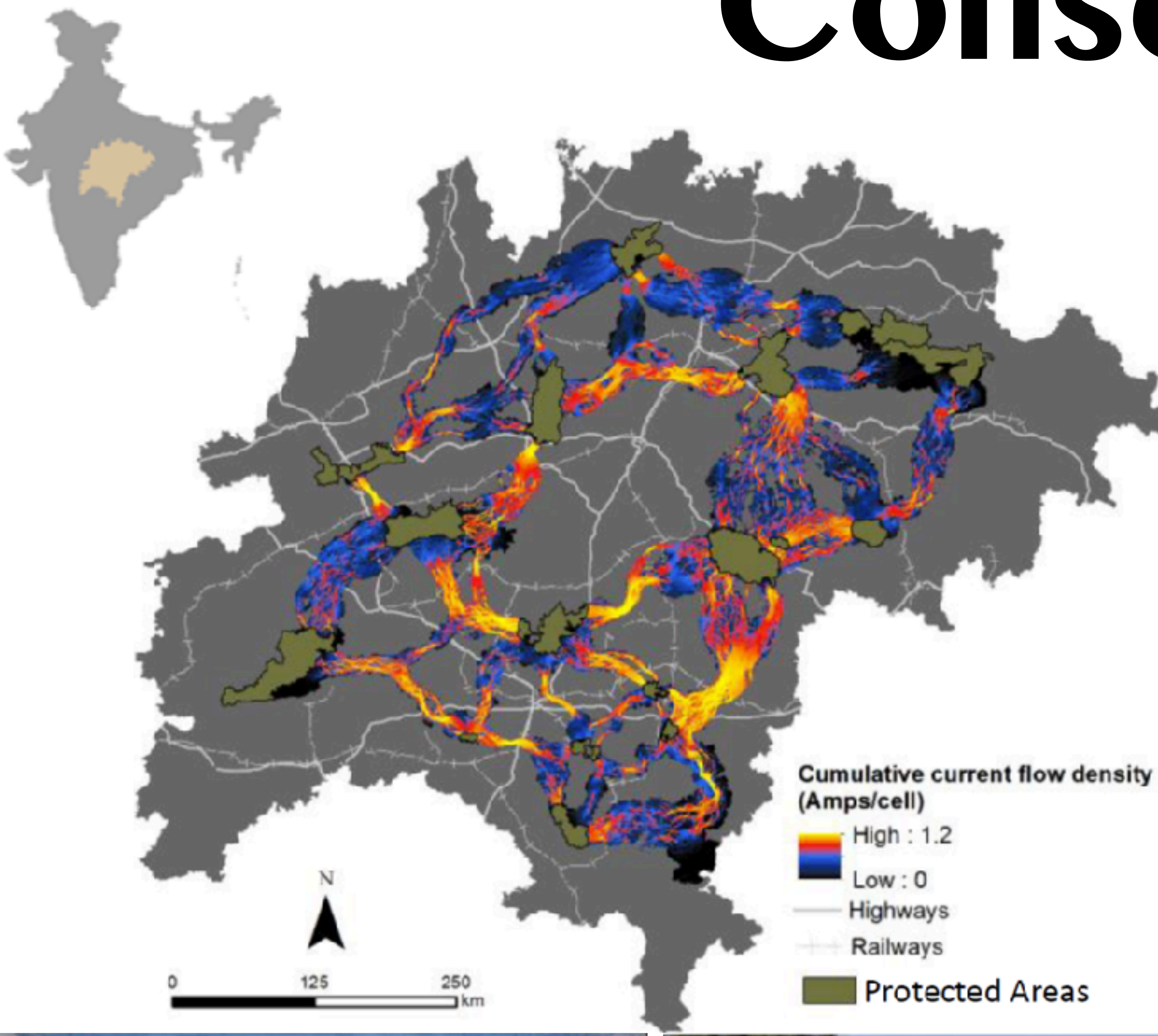
↓

Sparse Matrices

↓

Algebraic Multigrid, Laplacian Solvers

Conservation



CPUs

↓

Multiple Dispatch, Generic programming

↓

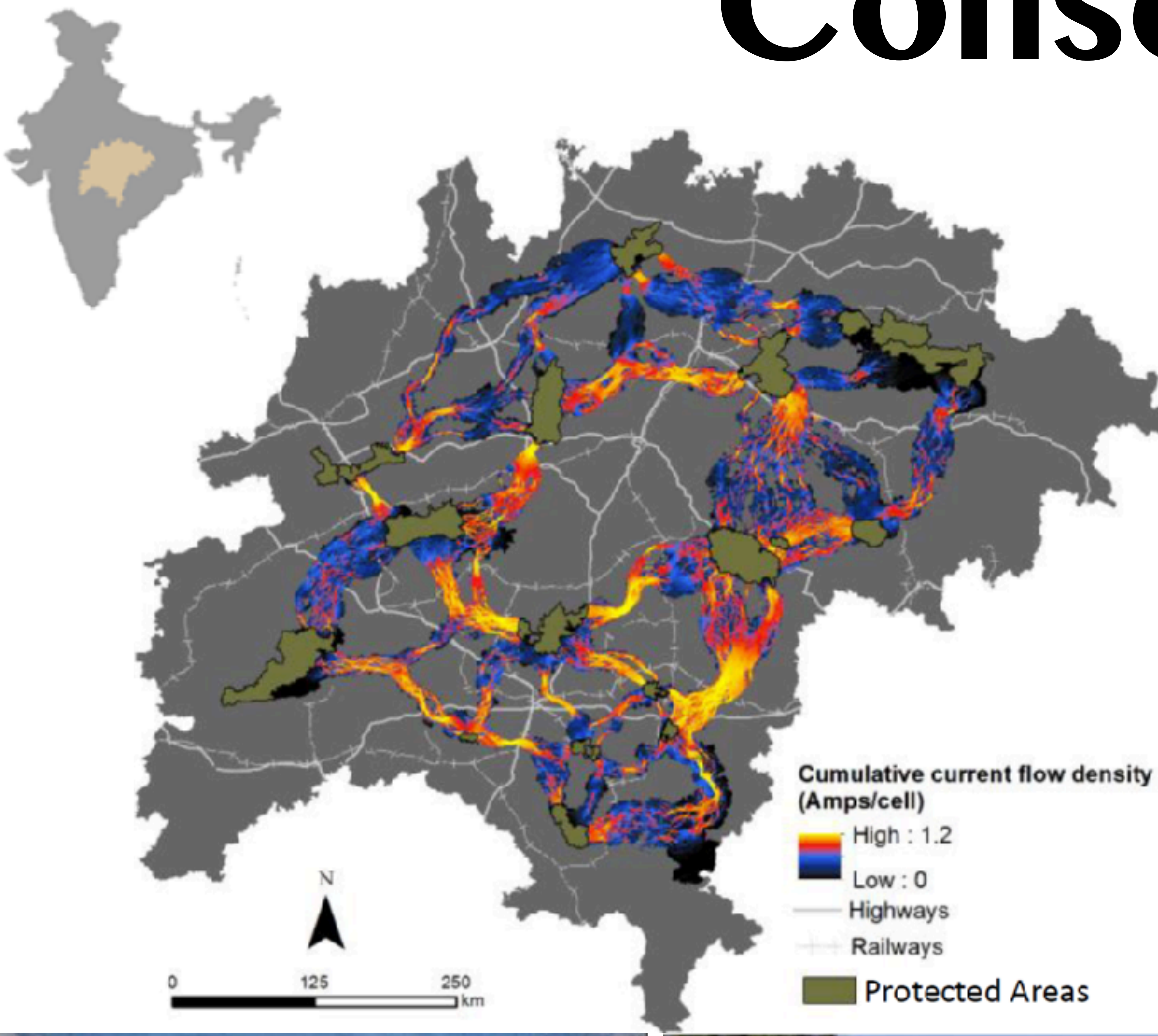
Sparse Matrices

↓

Algebraic Multigrid, Laplacian Solvers

↓

Conservation



CPUs

↓

Multiple Dispatch, Generic programming

↓

Sparse Matrices

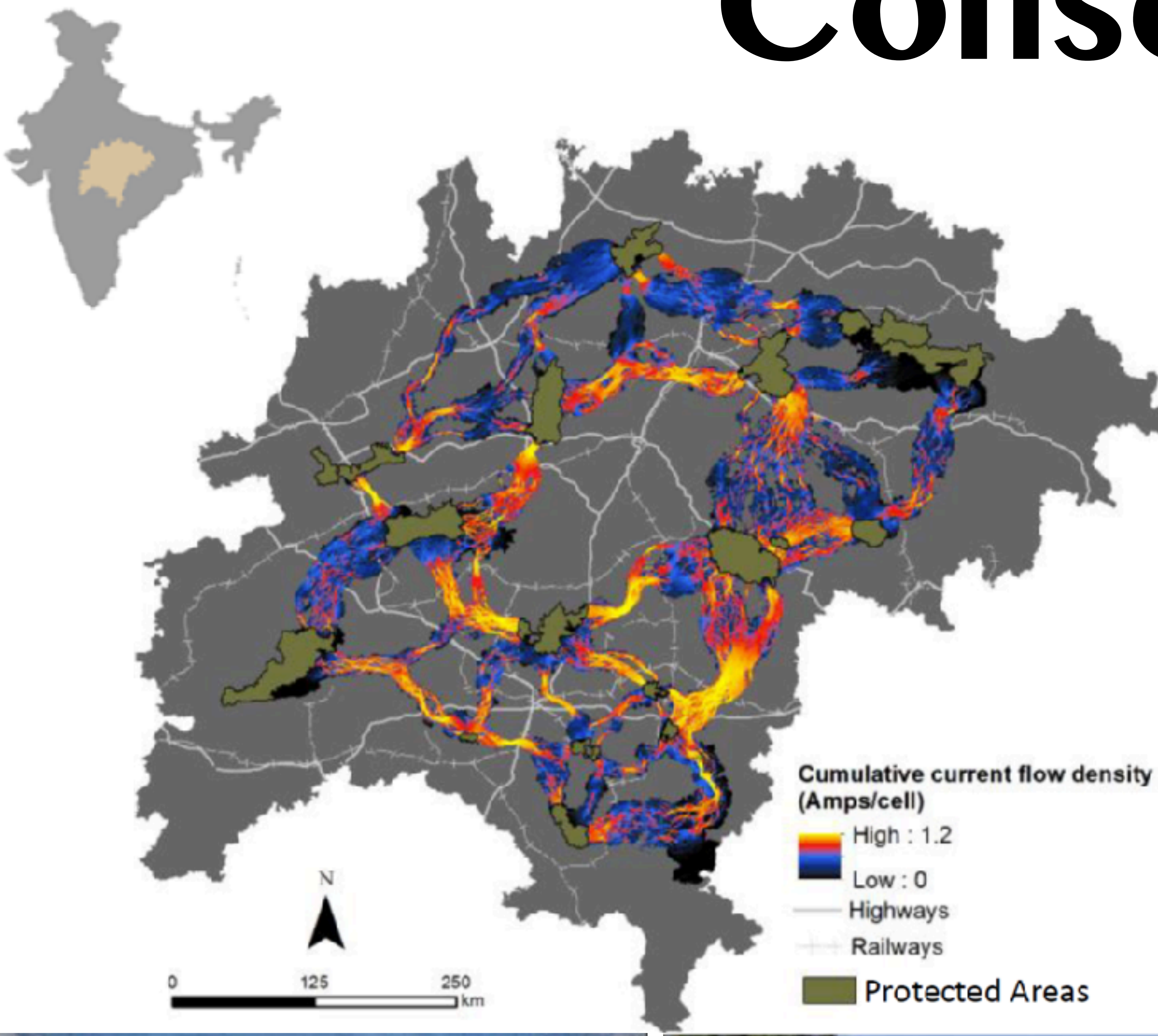
↓

Algebraic Multigrid, Laplacian Solvers

↓

Automatic Differentiation, Optimization

Conservation



CPUs

↓

Multiple Dispatch, Generic programming

↓

Sparse Matrices

↓

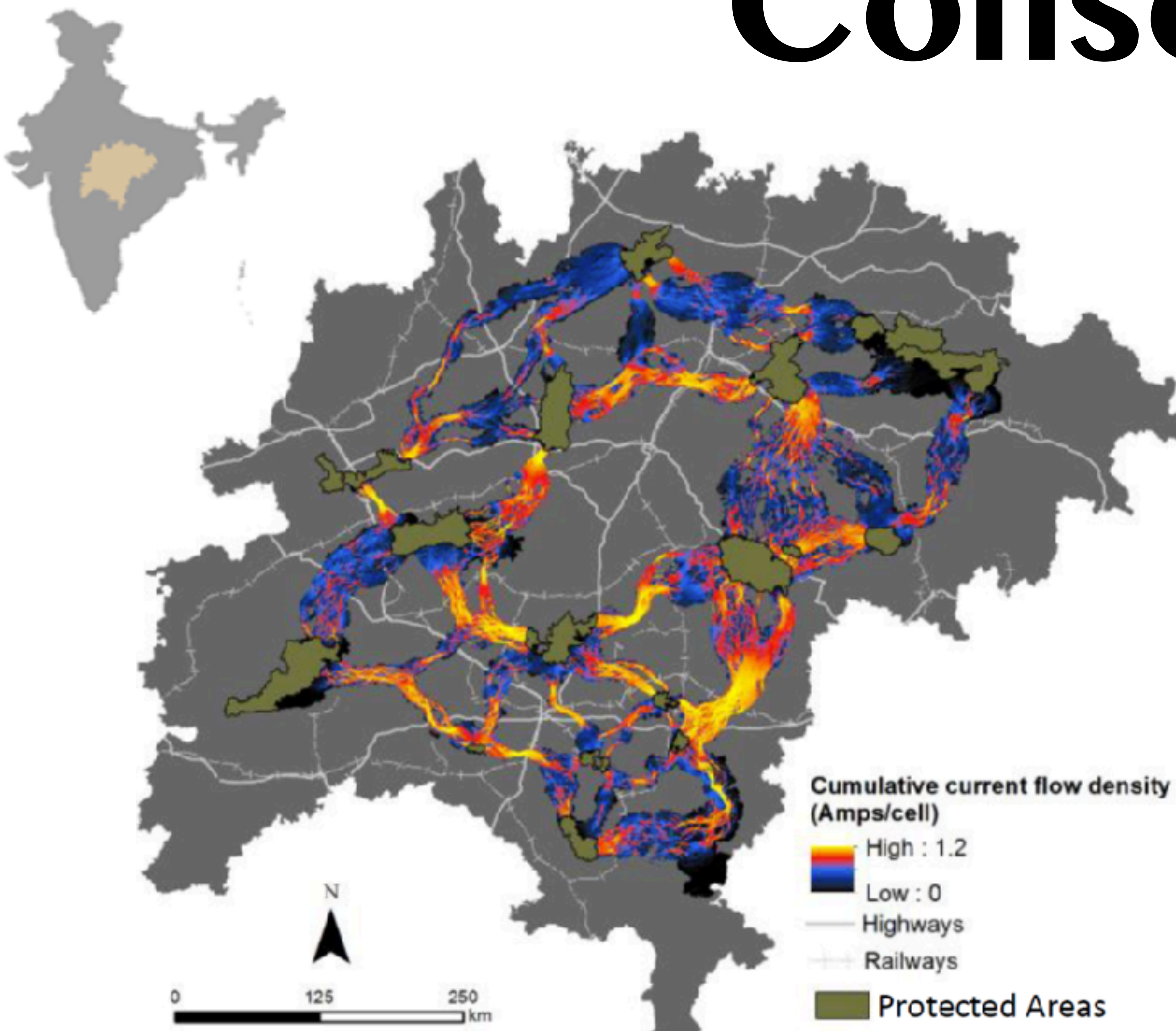
Algebraic Multigrid, Laplacian Solvers

↓

Automatic Differentiation, Optimization

↓

Conservation



CPUs

↓

Multiple Dispatch, Generic programming

↓

Sparse Matrices

↓

Algebraic Multigrid, Laplacian Solvers

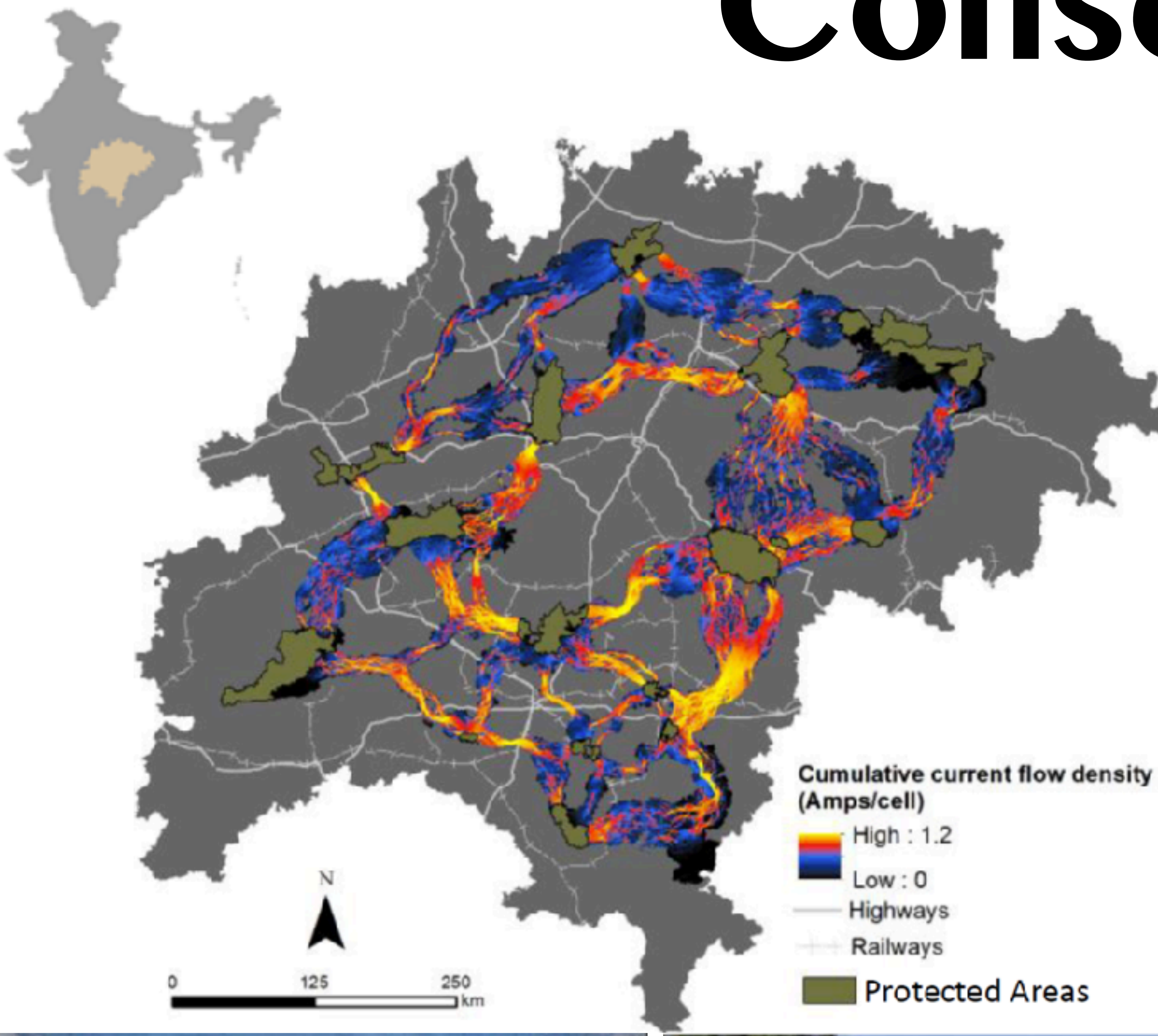
↓

Automatic Differentiation, Optimization

↓

Corridors, Climate change, Fire

Conservation



CPUs

↓

Multiple Dispatch, Generic programming

↓

Sparse Matrices

↓

Algebraic Multigrid, Laplacian Solvers

↓

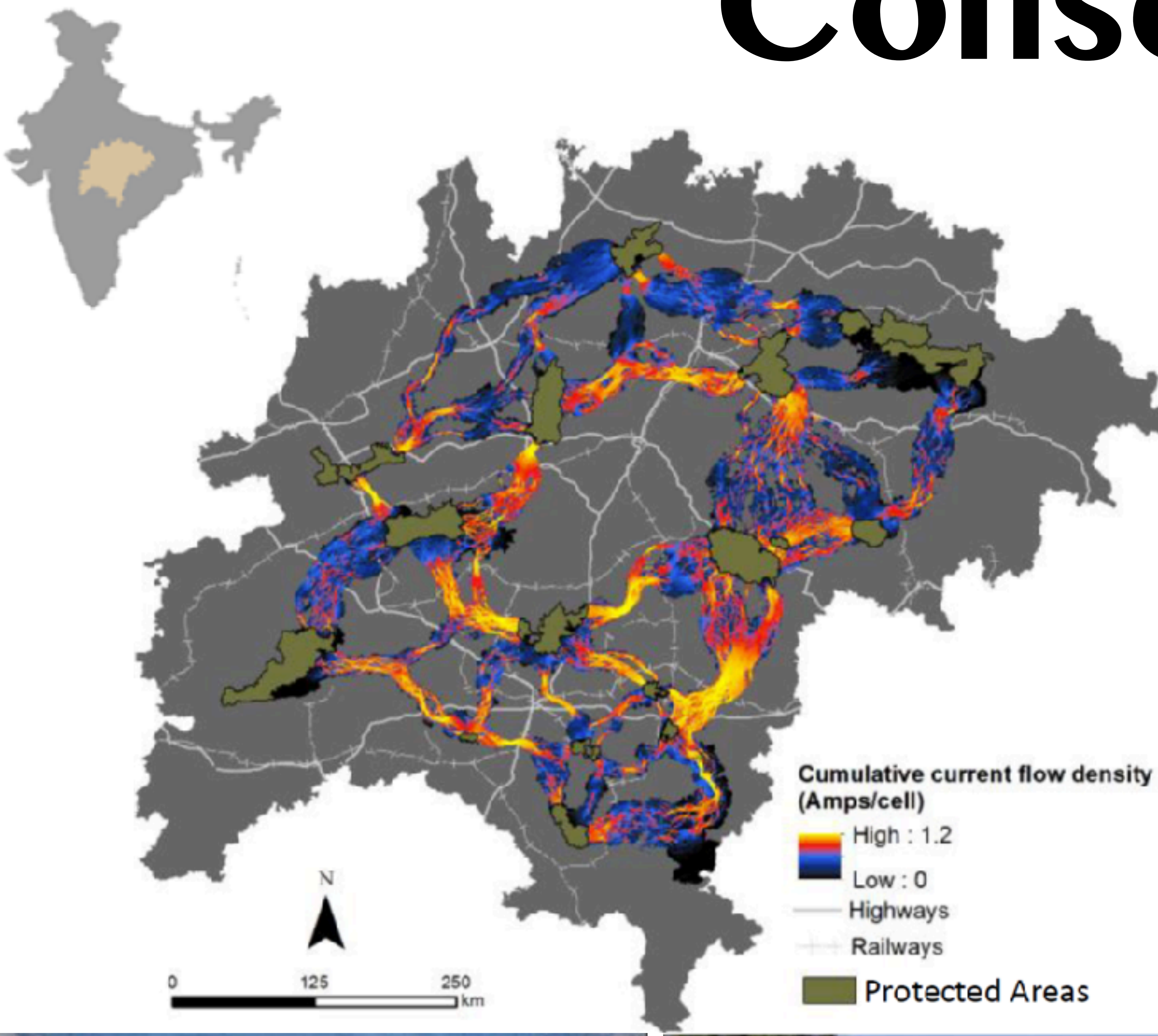
Automatic Differentiation, Optimization

↓

Corridors, Climate change, Fire

↓

Conservation



CPUs

↓

Multiple Dispatch, Generic programming

↓

Sparse Matrices

↓

Algebraic Multigrid, Laplacian Solvers

↓

Automatic Differentiation, Optimization

↓

Corridors, Climate change, Fire

↓

Policy making for conservation

impact through composability & abstractions