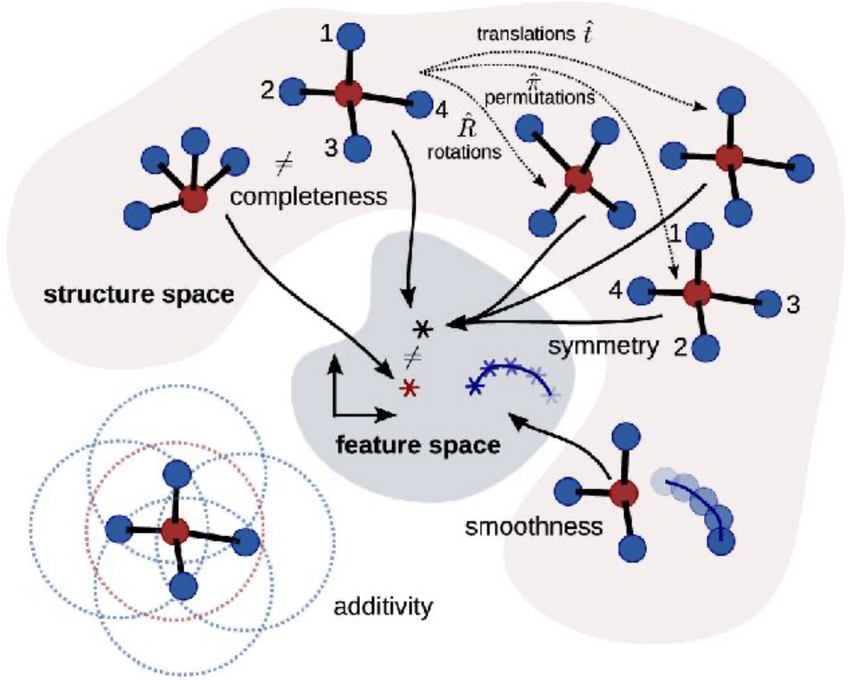
COSMO
COMPUTATIONAL SCIENCE AND MODELING

EPFL



ML-IP 2023

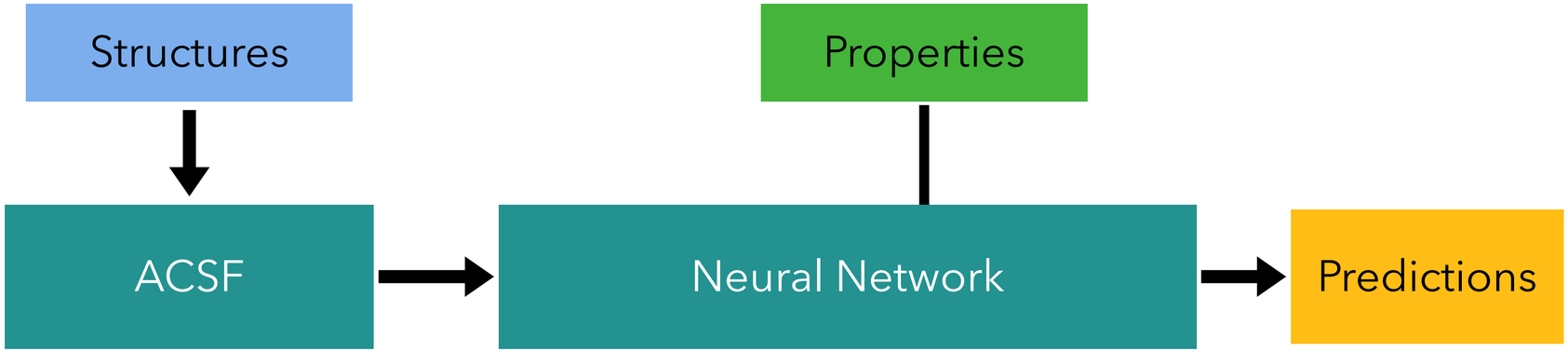
A common language for atomistic machine learning models

Guillaume Fraux

guillaume.fraux@epfl.ch

@Luthaf
@__luthaf__
mas.to/@luthaf

The state of atomistic machine learning



Behler-Parrinello NN

PiNN

SchNET

Snap

n2p2

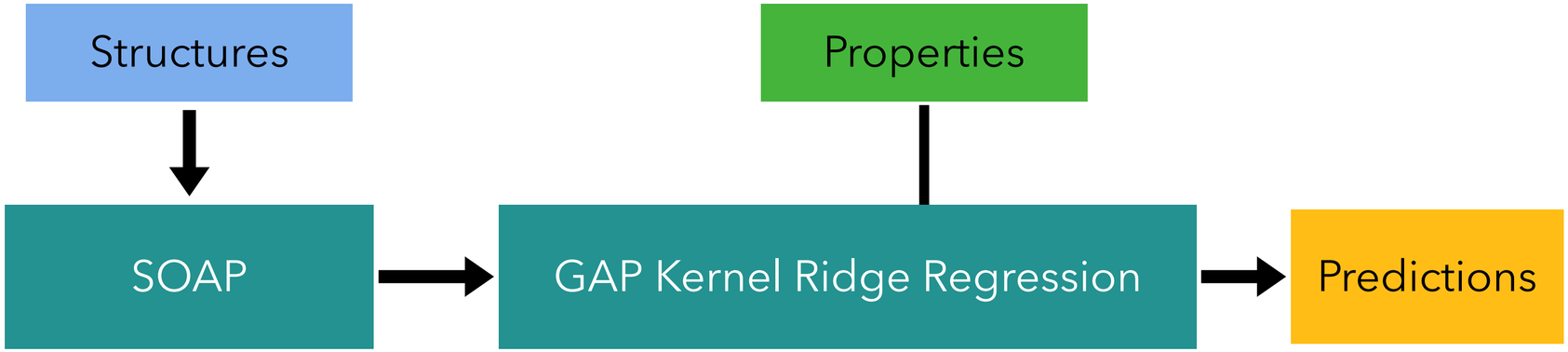
REANN

DeepMD

ANI

ænet

The state of atomistic machine learning



Behler-Parrinello NN

Kernel models

FLARE

PiNN

SchNET

librascal

Snap

n2p2

sGDML

REANN

DeepMD

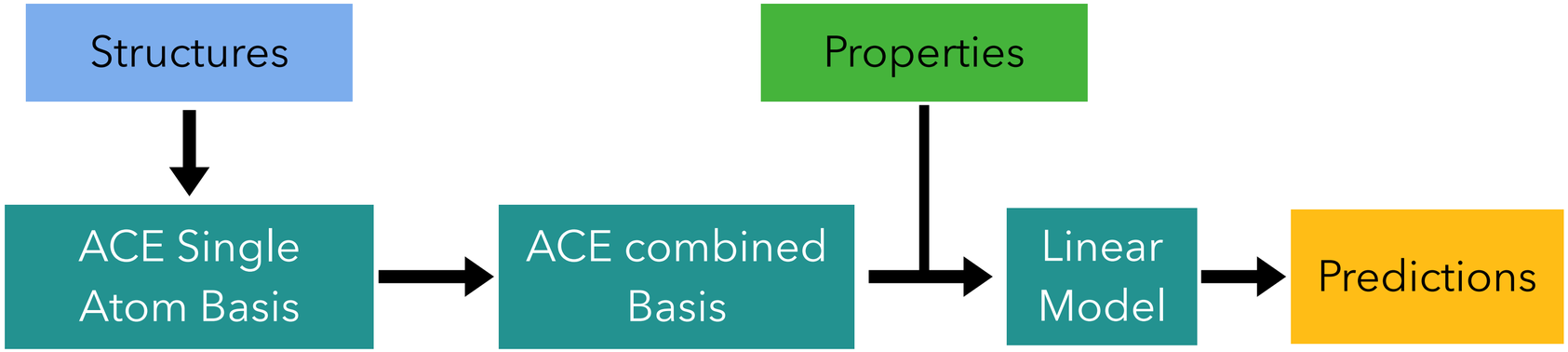
TurboSOAP

ANI

ænet

QUIP

The state of atomistic machine learning



Behler-Parrinello NN

Kernel models

Body-ordered representations

FLARE

PiNN

NICE

SchNET

librascal

Snap

NEP

n2p2

sGDML

REANN

DeepMD

ACEPotentials.jl

TurboSOAP

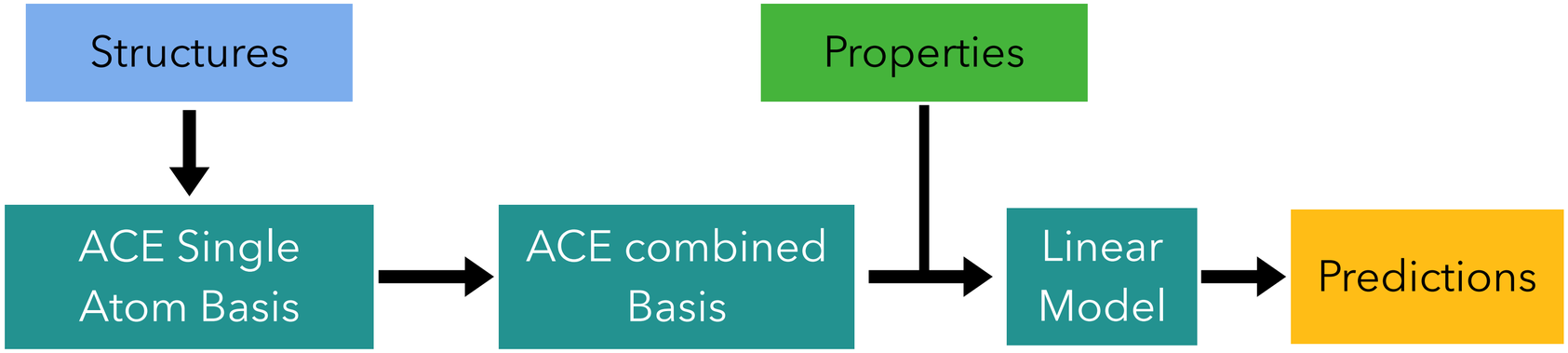
ANI

ænet

PACE

QUIP

The state of atomistic machine learning



Behler-Parrinello NN

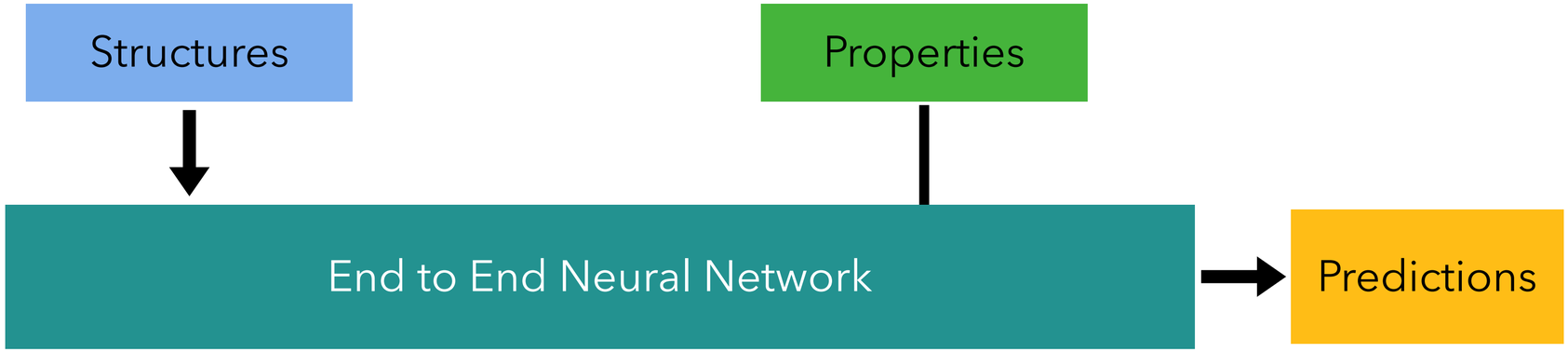
Kernel models

Body-ordered representations

Equivariant representations

FLARE
PiNN
NICE
MLIP / MTP
SchNET
librascal
Snap
NEP
n2p2
Allegro
sGDML
REANN
DeepMD
ACEPotentials.jl
Nequip
MACE
TurboSOAP
ANI
Milad
ænet
PACE
QUIP

The state of atomistic machine learning



Behler-Parrinello NN

Kernel models

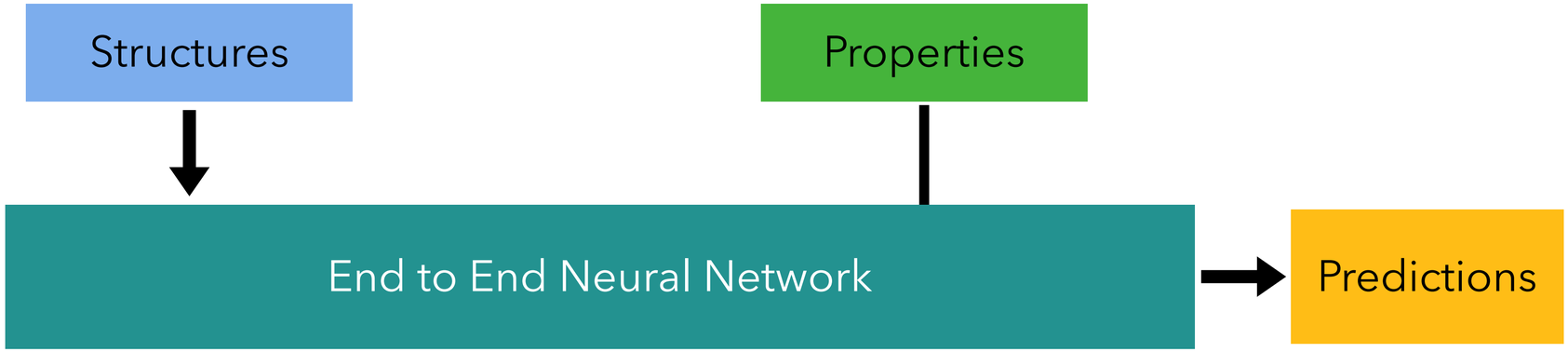
Body-ordered representations

Equivariant representations

End-to-end models

FLARE
PiNN
NICE
MLIP / MTP
SchNET
librascal
Snap
PET
NEP
n2p2
Allegro
sGDML
REANN
Nequip
DeepMD
ACEPotentials.jl
MACE
TurboSOAP
ANI
Milad
ænet
PACE
QUIP

The state of atomistic machine learning



Behler-Parrinello NN

Kernel models

Body-ordered representations

Equivariant representations

End-to-end models

FLARE
PiNN
NICE
MLIP / MTP
SchNET
librascal
Snap
PET
NEP
n2p2
Allegro
sGDML
REANN
DeepMD
ACEPotentials.jl
Nequip
MACE
TurboSOAP
ANI
Milad
ænet
PACE
QUIP

Software for MLIP is fragmented and siloed

A fragmented ecosystem

Software for MLIP is fragmented and siloed

Harder to **combine** ideas

Harder to **compare** models

Lot of **duplicated** effort
(LAMMPS interface, …)

Harder to **experiment** with new representations/models/architectures

A fragmented ecosystem

Software for MLIP is fragmented and siloed

Harder to **combine** ideas

Harder to **compare** models

Lot of **duplicated** effort
(LAMMPS interface, …)

Harder to **experiment** with new representations/models/architectures

What do we need for collaborations and re-use?

A fragmented ecosystem

Software for MLIP is fragmented and siloed

Harder to **combine** ideas

Harder to **compare** models

Lot of **duplicated** effort
(LAMMPS interface, …)

Harder to **experiment** with new representations/models/architectures

What do we need for collaborations and re-use?

Common language to share **data**

Common language to share **code**

A fragmented ecosystem

Software for MLIP is fragmented and siloed

Harder to **combine** ideas

Harder to **compare** models

Lot of **duplicated** effort
(LAMMPS interface, …)

Harder to **experiment** with new representations/models/architectures

N producers

M consumers

What do we need for collaborations and re-use?

Common language to share **data**

Common language to share **code**

A fragmented ecosystem

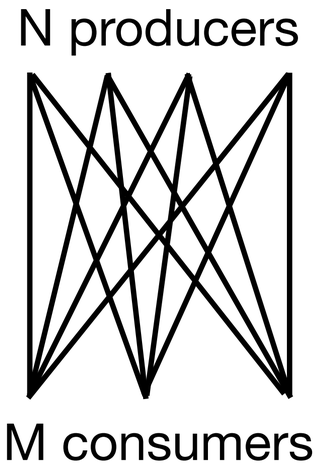
Software for MLIP is fragmented and siloed

Harder to **combine** ideas

Harder to **compare** models

Lot of **duplicated** effort
(LAMMPS interface, …)

Harder to **experiment** with new representations/models/architectures



What do we need for collaborations and re-use?

Common language to share **data**

Common language to share **code**

A fragmented ecosystem

Software for MLIP is fragmented and siloed

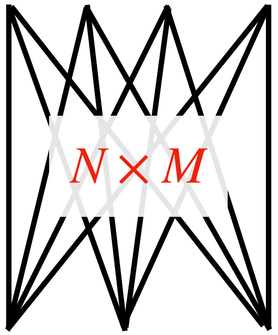
Harder to **combine** ideas

Harder to **compare** models

Lot of **duplicated** effort
(LAMMPS interface, …)

Harder to **experiment** with new representations/models/architectures

N producers



M consumers

What do we need for collaborations and re-use?

Common language to share **data**

Common language to share **code**

A fragmented ecosystem

Software for MLIP is fragmented and siloed

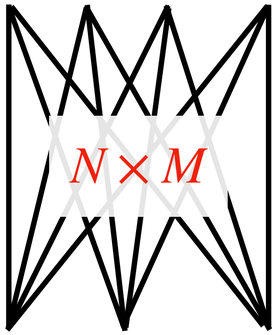
Harder to **combine** ideas

Harder to **compare** models

Lot of **duplicated** effort
(LAMMPS interface, …)

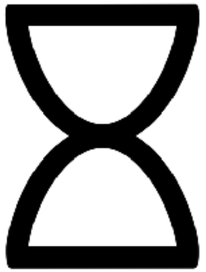
Harder to **experiment** with new representations/models/architectures

N producers



M consumers

N producers



M consumers

What do we need for collaborations and re-use?

Common language to share **data**

Common language to share **code**

A fragmented ecosystem

Software for MLIP is fragmented and siloed

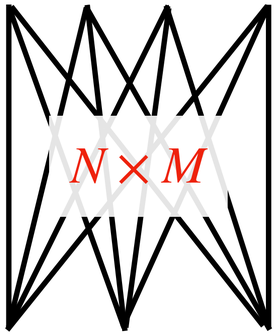
Harder to **combine** ideas

Harder to **compare** models

Lot of **duplicated** effort
(LAMMPS interface, …)

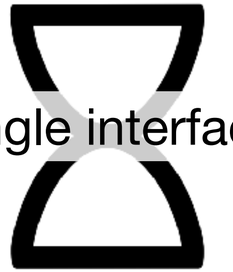
Harder to **experiment** with new representations/models/architectures

N producers



M consumers

N producers



Single interface $N + M$

M consumers

What do we need for collaborations and re-use?

Common language to share **data**

Common language to share **code**

Sharing data for atomistic machine learning

metatensor

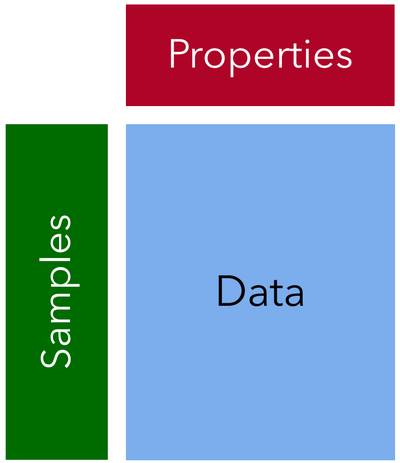
lab-cosmo/metatensor

Sharing data for atomistic machine learning

Store **data** and **metadata**, allow software to share without knowledge of each other

metatensor

lab-cosmo/metatensor

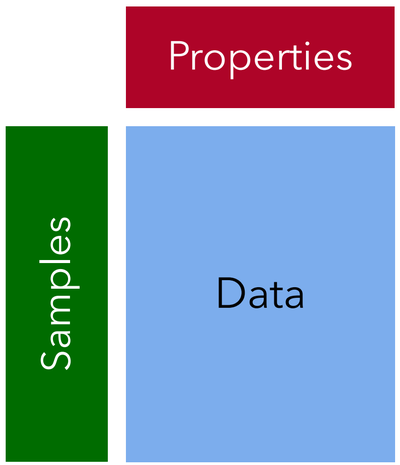


Sharing data for atomistic machine learning

Store **data** and **metadata**, allow software to share without knowledge of each other

metatensor

lab-cosmo/metatensor



Samples

structure	atom
0	0
0	1
0	…
1	0

Properties

l	m	n
0	0	0
0	0	1
0	0	2
	…	
3	-3	3
3	-2	0
	…	

Sharing data for atomistic machine learning

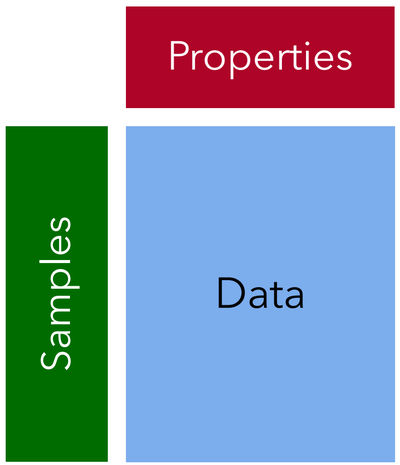
metatensor

Store **data** and **metadata**, allow software to share without knowledge of each other

Store values and **gradients** of these values together

lab-cosmo/metatensor

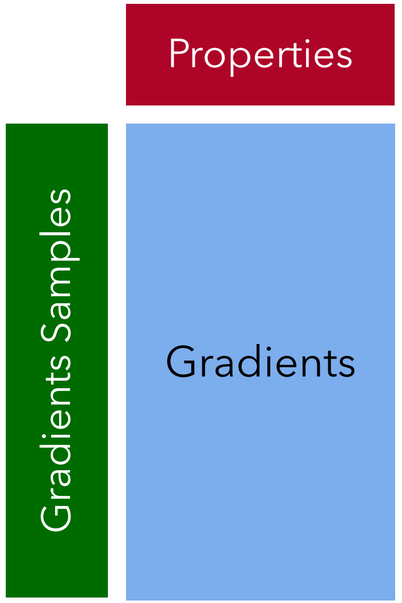
values



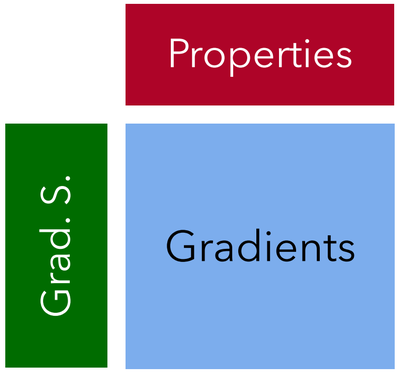
Samples	
structure	atom
0	0
0	1
0	…
1	0

Properties		
l	m	n
0	0	0
0	0	1
0	0	2
	…	
3	-3	3
3	-2	0
	…	

gradients w.r.t.
positions



gradients w.r.t.
parameter



Sharing data for atomistic machine learning

metatensor

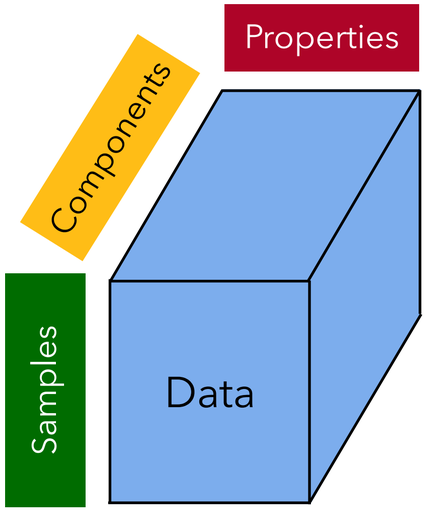
lab-cosmo/metatensor

Store **data** and **metadata**, allow software to share without knowledge of each other

Store values and **gradients** of these values together

Handle vectorial components in the data (equivariance, vector quantities)

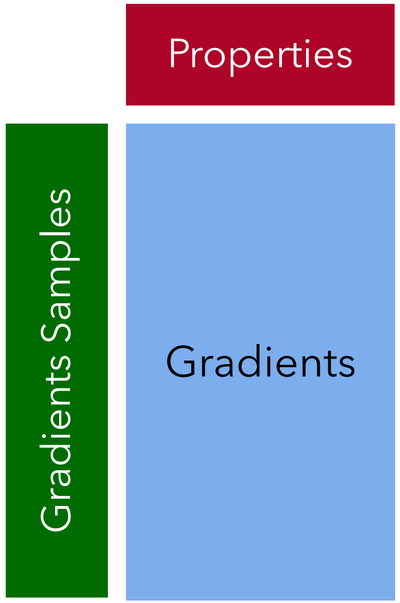
values



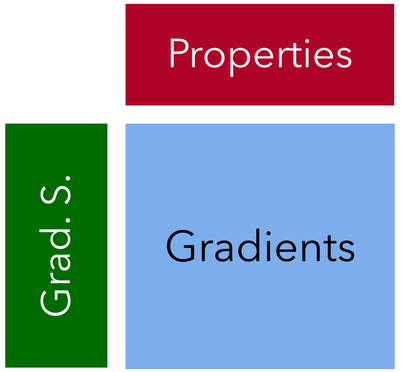
Samples	
structure	atom
0	0
0	1
0	…
1	0

Properties		
l	m	n
0	0	0
0	0	1
0	0	2
	…	
3	-3	3
3	-2	0
	…	

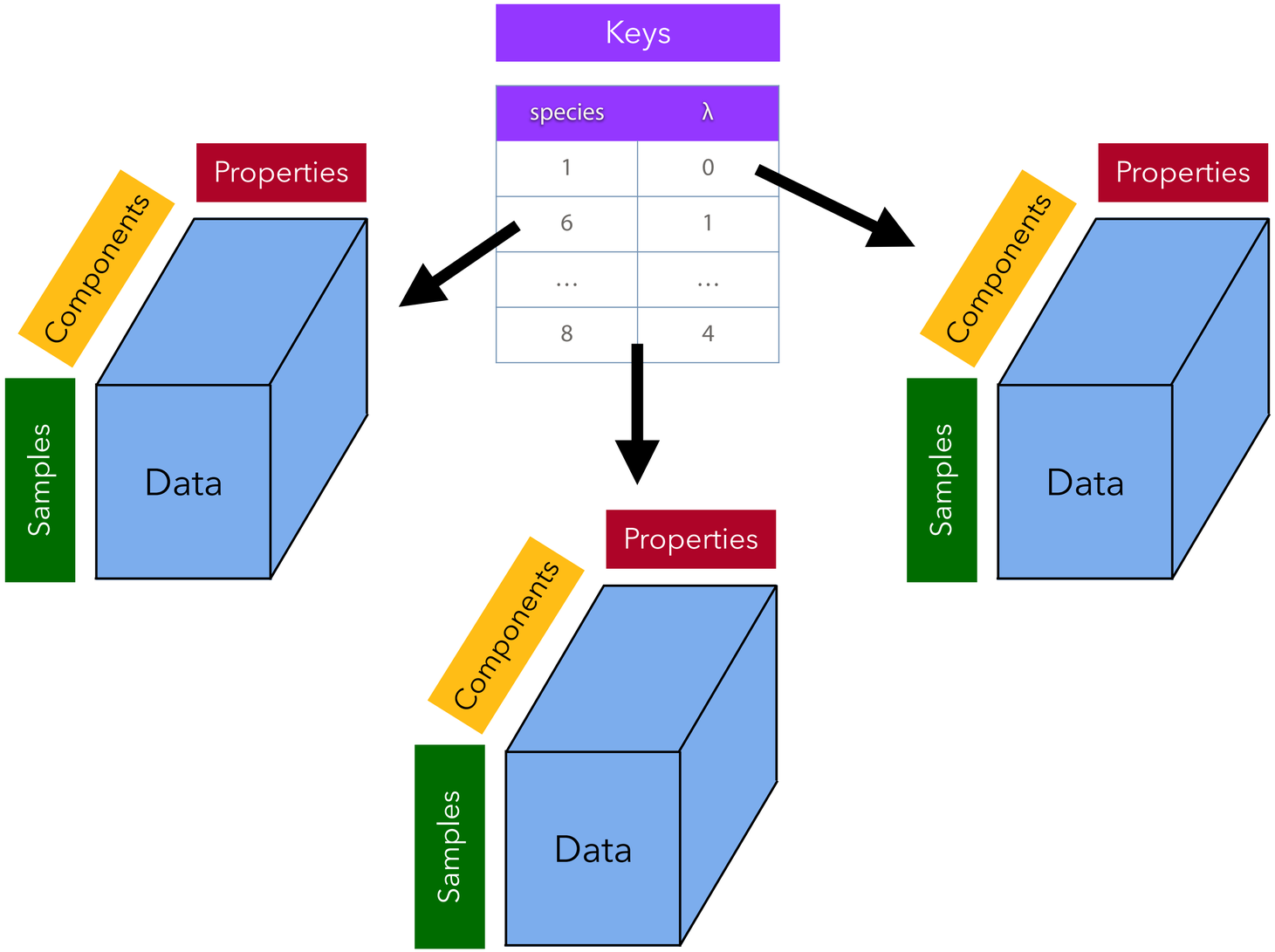
gradients w.r.t.
positions



gradients w.r.t.
parameter

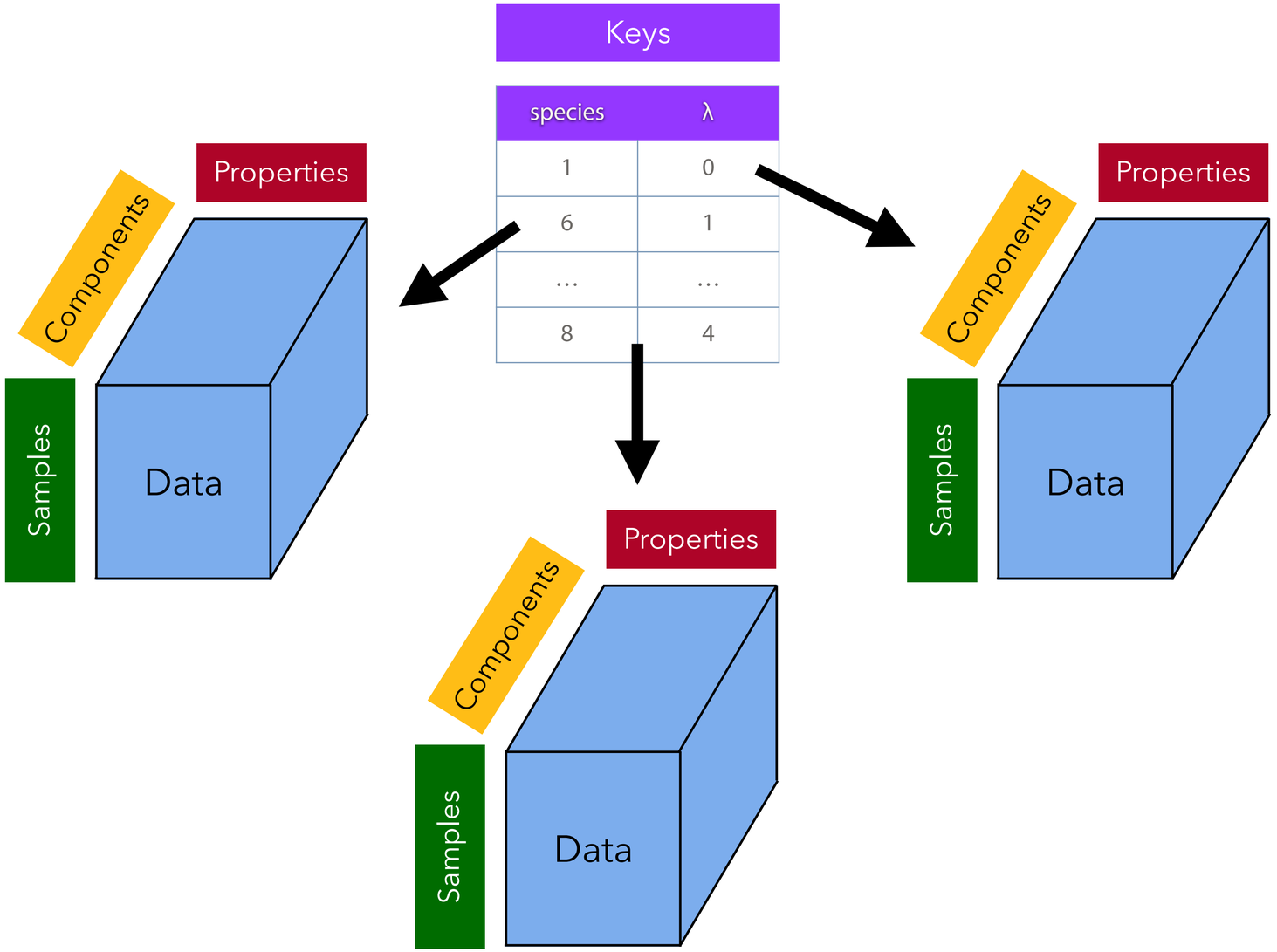


Sharing data for atomistic machine learning



Block sparse data

Sharing data for atomistic machine learning



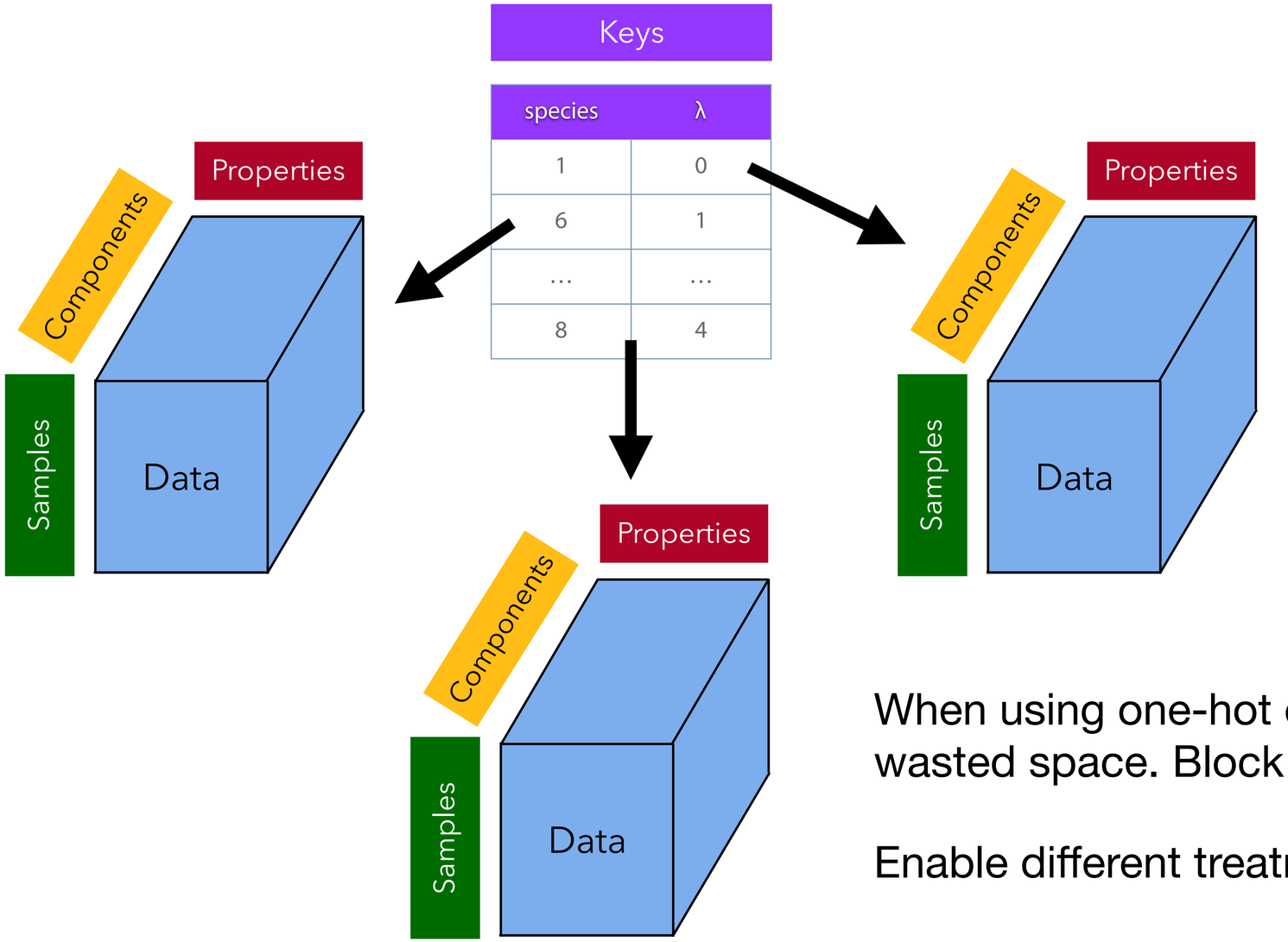
Block sparse data

Equivariant representations

Handle irreducible representations of the group separately

SO(3): keep λ separated, group the corresponding μ

Sharing data for atomistic machine learning



Block sparse data

Equivariant representations

Handle irreducible representations of the group separately

SO(3): keep λ separated, group the corresponding μ

Species handling

When using one-hot encoding of species, potential for a lot of wasted space. Block sparse data only keep the non-zero parts.

Enable different treatment of different species in models

Sharing data across ecosystems

Different researchers use different programming languages, we want to support all of them!

Metatensor is a Rust library, with a **C API**.
This C API can be used from many languages

Sharing data across ecosystems

Exists

Different researchers use different programming languages, we want to support all of them!

Metatensor is a Rust library, with a **C API**.
This C API can be used from many languages

Sharing data across ecosystems

Exists

Planned

Different researchers use different programming languages, we want to support all of them!

Metatensor is a Rust library, with a **C API**.
This C API can be used from many languages

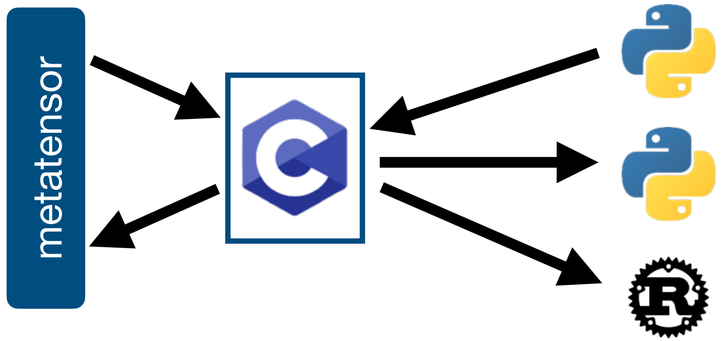
Sharing data across ecosystems

Exists

Planned

Different researchers use different programming languages, we want to support all of them!

Metatensor is a Rust library, with a **C API**.
This C API can be used from many languages



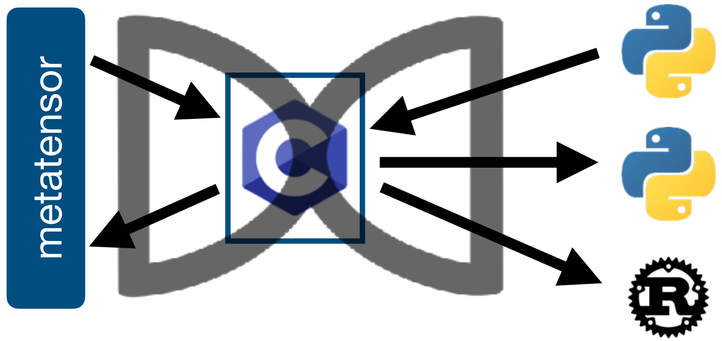
Sharing data across ecosystems

Exists

Planned

Different researchers use different programming languages, we want to support all of them!

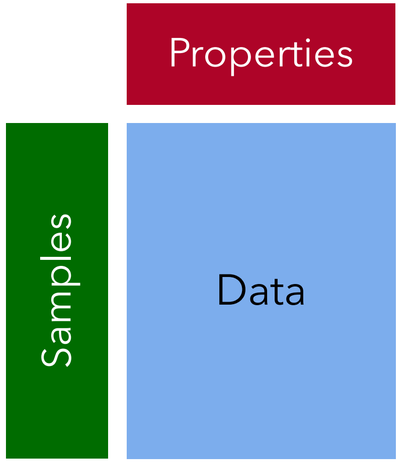
Metatensor is a Rust library, with a **C API**.
This C API can be used from many languages



Sharing data across ecosystems

Exists

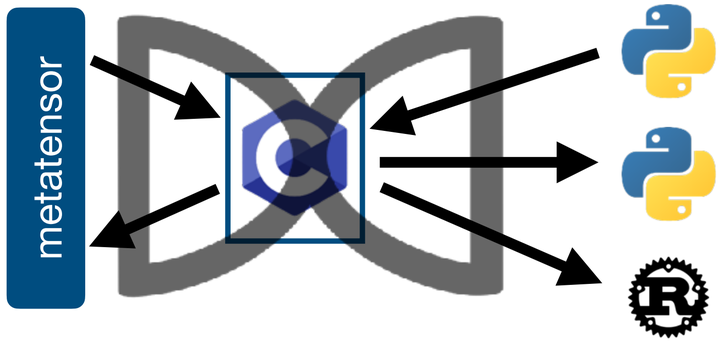
Planned



Different researchers use different programming languages, we want to support all of them!

Even in a single language, multiple ecosystems and data sources

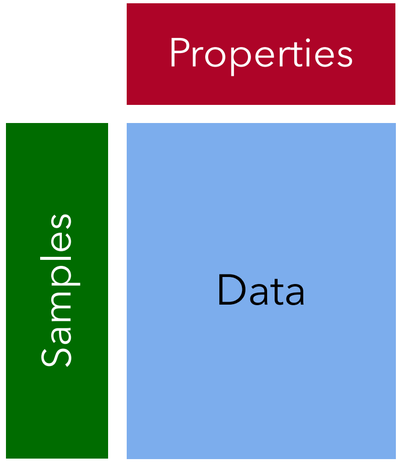
Metatensor is a Rust library, with a **C API**.
This C API can be used from many languages



Sharing data across ecosystems

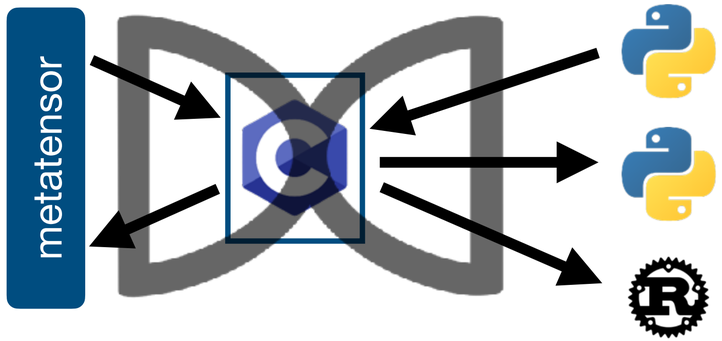
Exists

Planned



Different researchers use different programming languages, we want to support all of them!

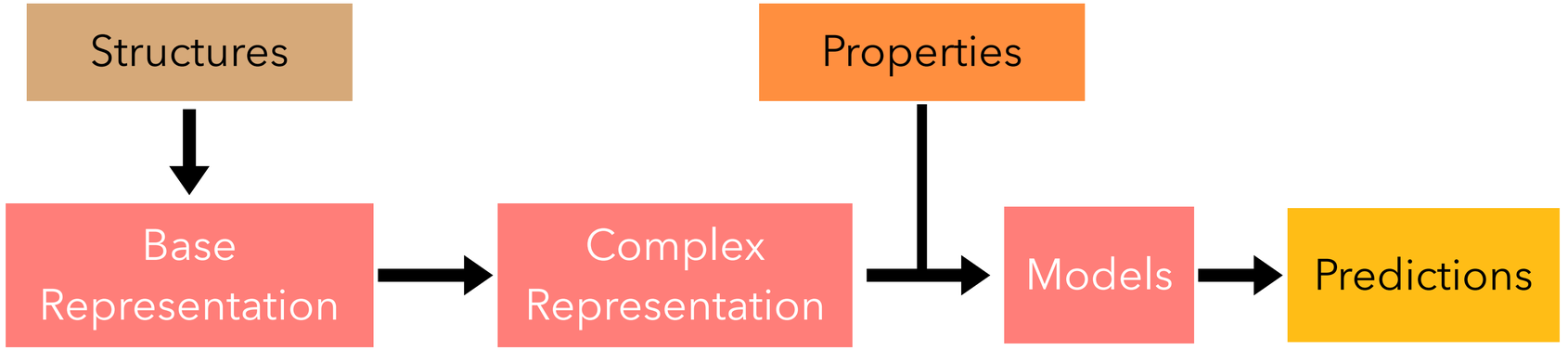
Metatensor is a Rust library, with a **C API**.
This C API can be used from many languages



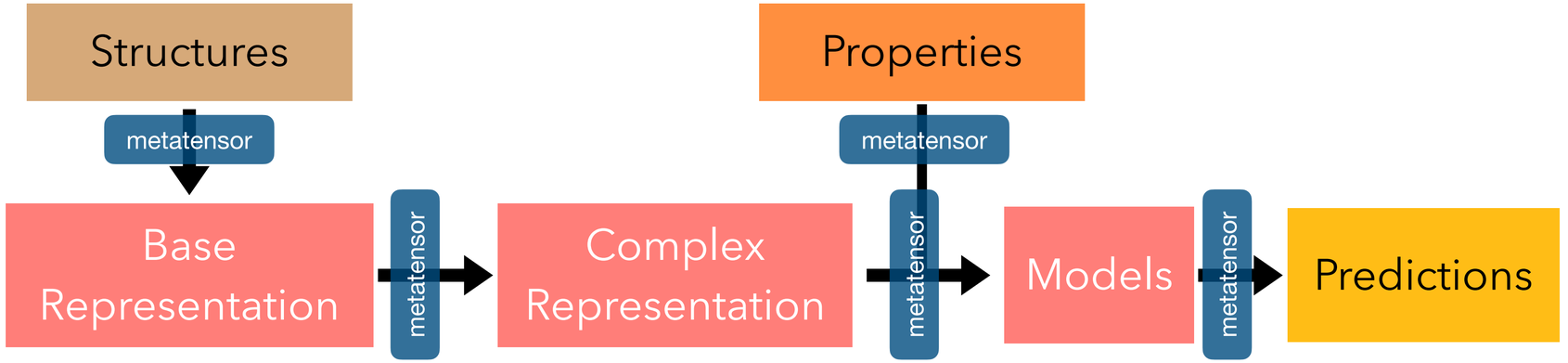
Even in a single language, multiple ecosystems and data sources

Metatensor need minimal information about the data, can manipulate values on GPU, inside torch Tensor (with full autograd support), …

Data sharing in practice



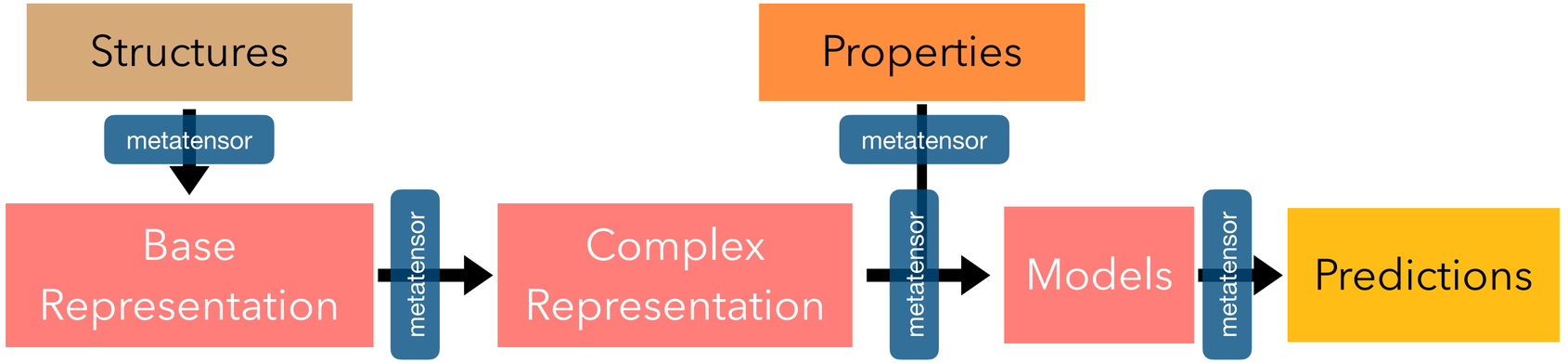
Data sharing in practice

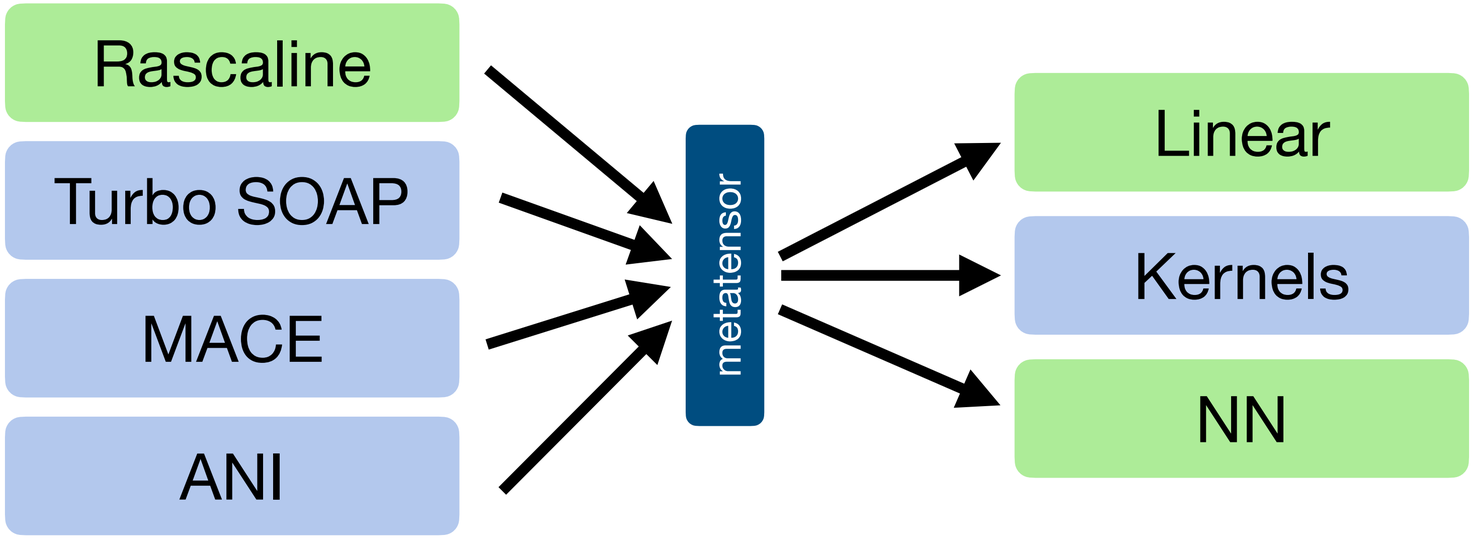


Data sharing in practice

Exists

Planned

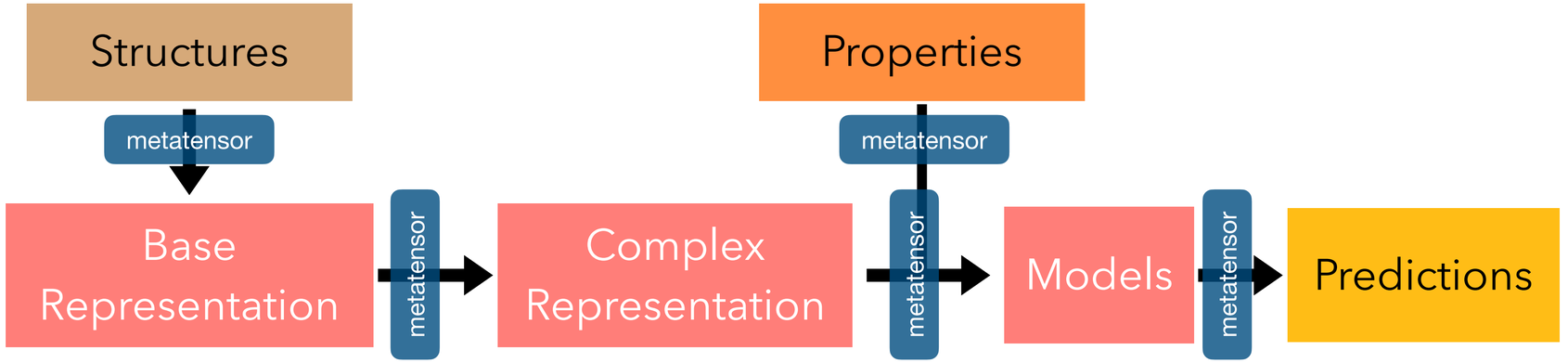


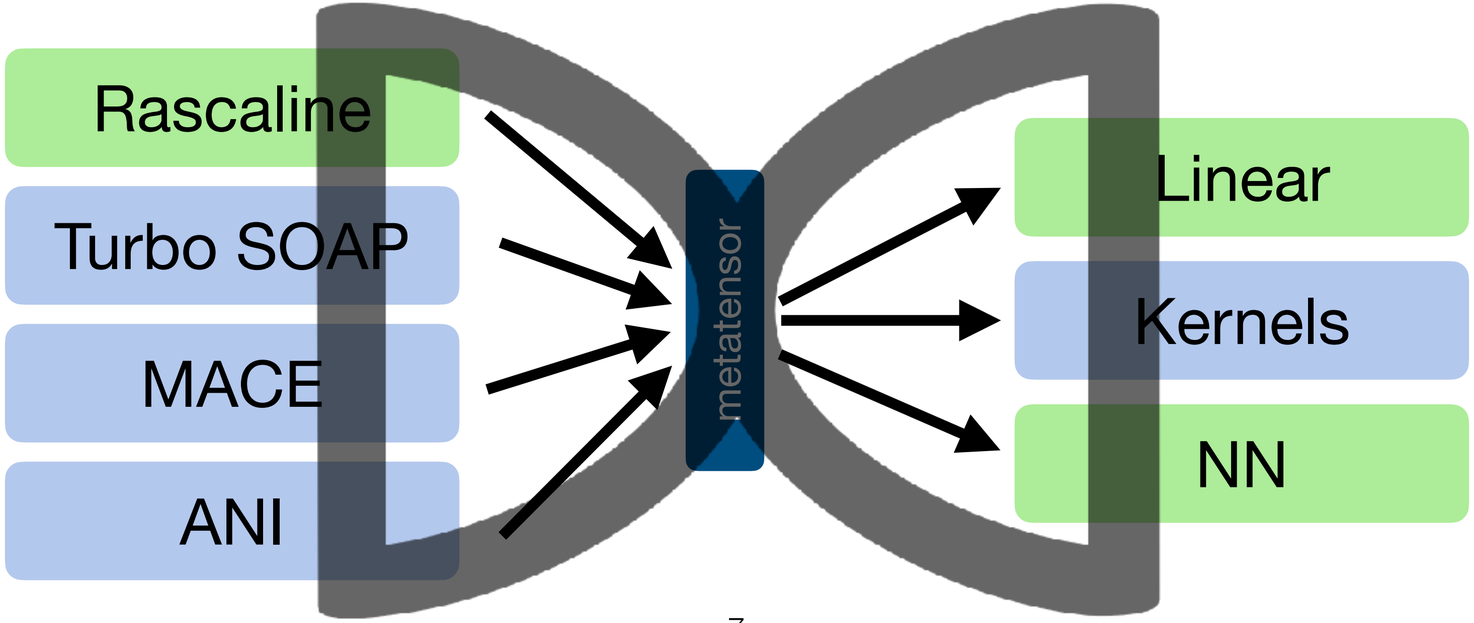


Data sharing in practice

Exists

Planned





Existing metatensor ecosystem

Operations on metatensor data

lab-cosmo/metatensor

Existing metatensor ecosystem

Operations on metatensor data

lab-cosmo/metatensor

Parallel **representations** calculations (SOAP, LODE, …)

Luthaf/rascaline

Existing metatensor ecosystem

Operations on
metatensor data

lab-cosmo/metatensor

Models building blocks
Ridge, Kernels, NN

lab-cosmo/equisolve

Parallel **representations**
calculations (SOAP, LODE, …)

Luthaf/rascaline

Existing metatensor ecosystem

Operations on metatensor data

lab-cosmo/metatensor

Learnable representations

lab-cosmo/torch_spex

serfg/pet

Models building blocks
Ridge, Kernels, NN

lab-cosmo/equisolve

Parallel **representations** calculations (SOAP, LODE, …)

Luthaf/rascaline

Existing metatensor ecosystem

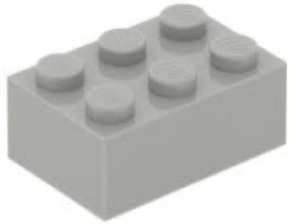
Operations on metatensor data

lab-cosmo/metatensor

Learnable representations

lab-cosmo/torch_spex

serfg/pet



Models building blocks
Ridge, Kernels, NN

lab-cosmo/equisolve

Electronic structure ML

jwa7/rho_learn

lcmd-epfl/Q-stack
Hamiltonian Learning

Parallel **representations** calculations (SOAP, LODE, …)

Luthaf/rascaline

Existing metatensor ecosystem

Operations on metatensor data

lab-cosmo/metatensor

Learnable representations

lab-cosmo/torch_spex

serfg/pet

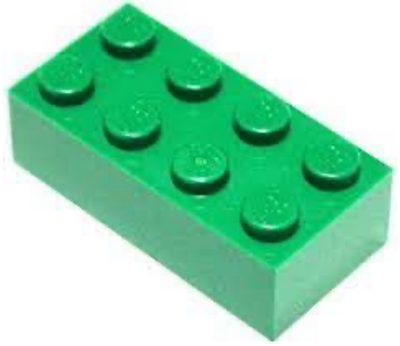
Models building blocks
Ridge, Kernels, NN

lab-cosmo/equisolve

Electronic structure ML

jwa7/rho_learn

lcmd-epfl/Q-stack
Hamiltonian Learning

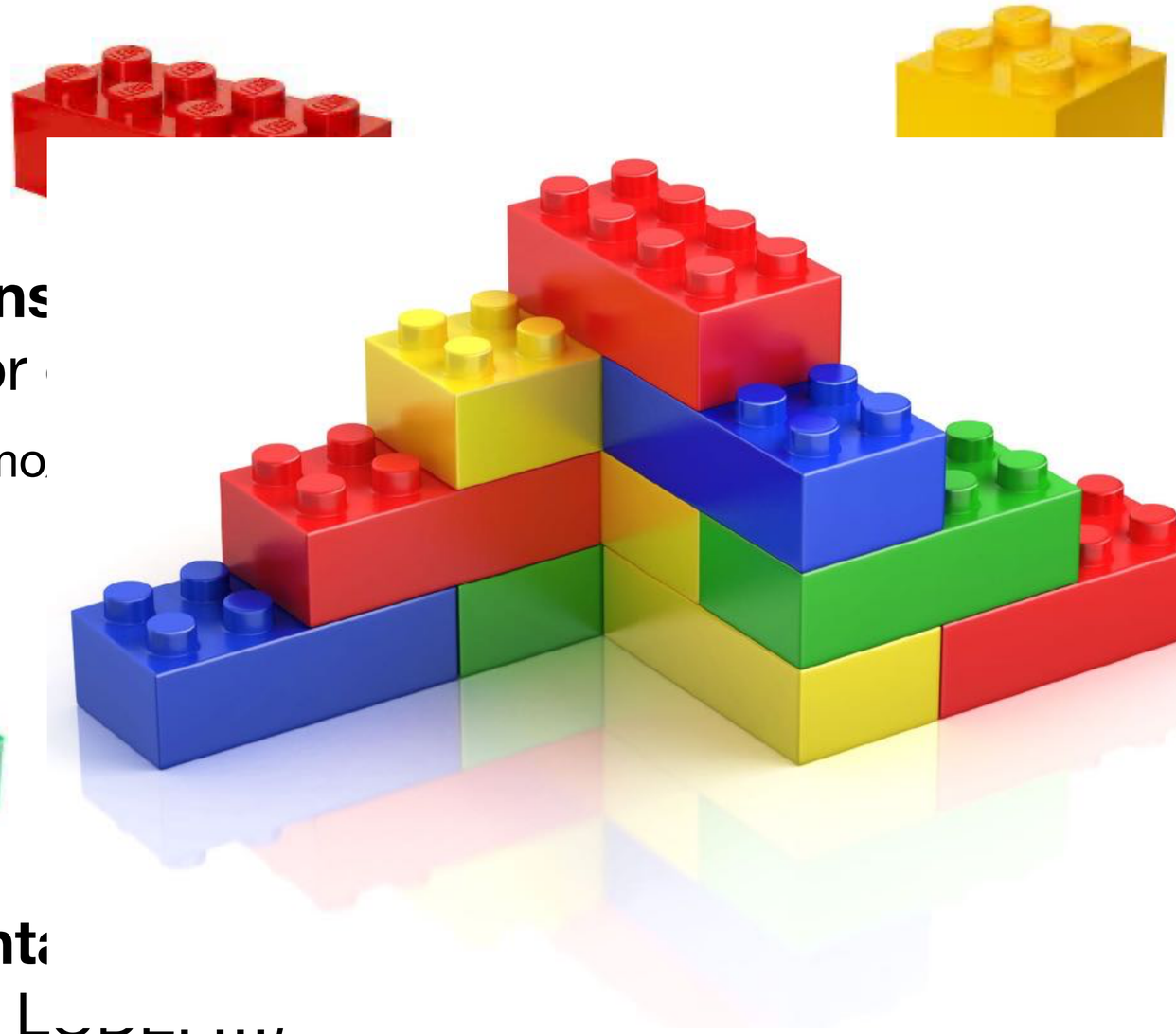


Parallel **representations** calculations (SOAP, LODE, …)

Luthaf/rascaline

Your own software?

Existing metatensor ecosystem



Operations
metatensor
lab-cosmo

ıtions
ɔex

ctronic structure ML
jwa7/rho_learn
lcmd-epfl/Q-stack
Hamiltonian Learning

Parallel **representa**
calculations (SOAP, L

Luthaf/rascaline

Your own software?

Rascaline: representation calculations

Library of representations for atomistic machine learning

Output data as metatensor, can also use metatensor for input

rascaline

Luthaf/rascaline

Rascaline: representation calculations

Library of representations for atomistic machine learning

Output data as metatensor, can also use metatensor for input

rascaline

Luthaf/rascaline

Many different representations

SOAP Pair Spherical Expansion

SOAP Spherical Expansion

SOAP Radial Spectrum

SOAP Power Spectrum

LODE Spherical Expansion

Neighbors Lists

Atomic Composition

Rascaline: representation calculations

Library of representations for atomistic machine learning

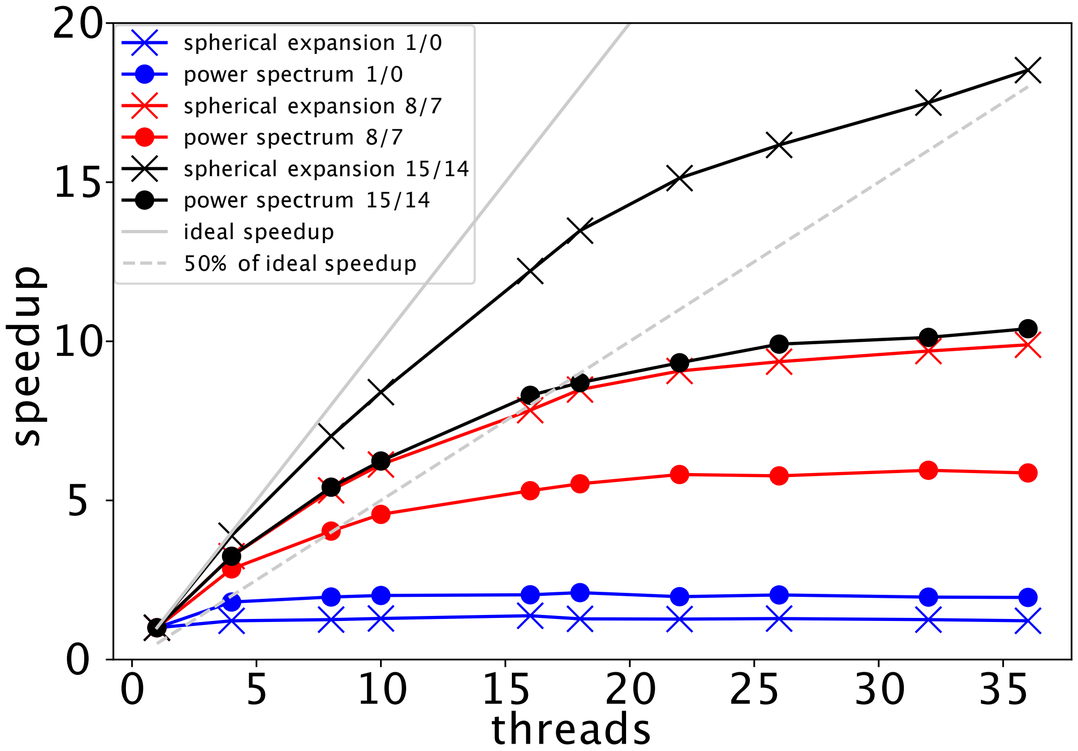
Output data as metatensor, can also use metatensor for input

rascaline

Luthaf/rascaline

Many different representations

Fast — Parallelism



Rascaline: representation calculations

Library of representations for atomistic machine learning

Output data as metatensor, can also use metatensor for input

rascaline

Luthaf/rascaline

Many different representations

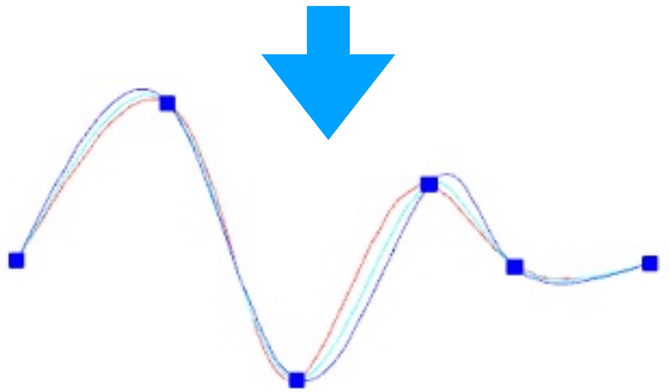
$$I_{nl}(r_{ij}) = \frac{1}{(\pi\sigma^2)^{3/4}} 4\pi e^{-\frac{r_{ij}^2}{2\sigma^2}} \int_0^\infty \mathrm{d}r r^2 R_{nl}(r) e^{-\frac{r^2}{2\sigma^2}} i_l\left(\frac{r r_{ij}}{\sigma^2}\right)$$

Fast — Parallelism

Fast — Radial Basis splining (~3x)

$$I_{nl}(r_{ij}) = \frac{1}{(\pi\sigma^2)^{3/4}} \cdot \pi^{\frac{3}{2}} \frac{\Gamma\left(n_{\text{eff}}\right)}{\Gamma\left(l_{\text{eff}}\right)} \frac{(a r_{ij})^l}{(a+b)^{n_{\text{eff}}}} M\left(n_{\text{eff}}, l_{\text{eff}}, \frac{a^2 r_{ij}^2}{a^2+b^2}\right)$$

Custom Python basis definition



Rascaline: representation calculations

Library of representations for atomistic machine learning

Output data as metatensor, can also use metatensor for input

rascaline

Luthaf/rascaline

Many different representations

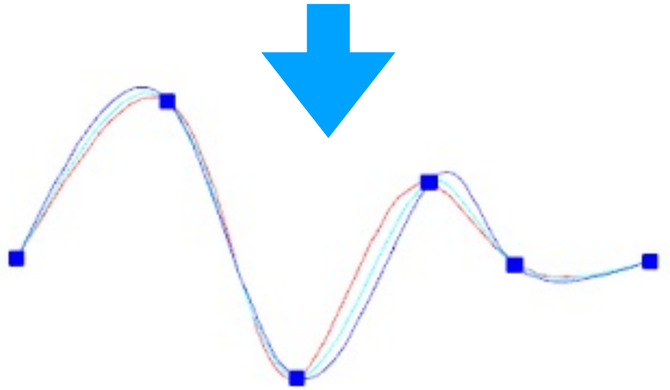
$$I_{nl}(r_{ij}) = \frac{1}{(\pi\sigma^2)^{3/4}} 4\pi e^{-\frac{r_{ij}^2}{2\sigma^2}} \int_0^\infty \mathrm{d}r r^2 R_{nl}(r) e^{-\frac{r^2}{2\sigma^2}} i_l\left(\frac{r r_{ij}}{\sigma^2}\right)$$

Fast — Parallelism

Fast — Radial Basis splining (~3x)

$$I_{nl}(r_{ij}) = \frac{1}{(\pi\sigma^2)^{3/4}} \cdot \pi^{\frac{3}{2}} \frac{\Gamma\left(n_{\text{eff}}\right)}{\Gamma\left(l_{\text{eff}}\right)} \frac{(a r_{ij})^l}{(a+b)^{n_{\text{eff}}}} M\left(n_{\text{eff}}, l_{\text{eff}}, \frac{a^2 r_{ij}^2}{a^2+b^2}\right)$$

Custom Python basis definition



Full integration with PyTorch

Rascaline: representation calculations

Library of representations for atomistic machine learning

Output data as metatensor, can also use metatensor for input

rascaline

Luthaf/rascaline

Many different representations

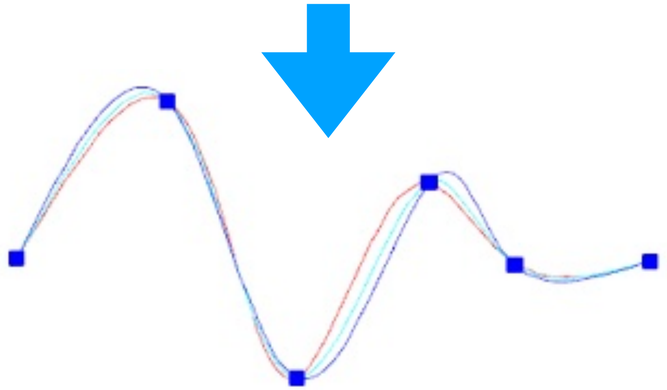
$$I_{nl}(r_{ij}) = \frac{1}{(\pi\sigma^2)^{3/4}} 4\pi e^{-\frac{r_{ij}^2}{2\sigma^2}} \int_0^\infty \mathrm{d}r r^2 R_{nl}(r) e^{-\frac{r^2}{2\sigma^2}} i_l\left(\frac{r r_{ij}}{\sigma^2}\right)$$

Fast — Parallelism

Fast — Radial Basis splining (~3x)

$$I_{nl}(r_{ij}) = \frac{1}{(\pi\sigma^2)^{3/4}} \cdot \pi^{\frac{3}{2}} \frac{\Gamma\left(n_{\text{eff}}\right)}{\Gamma\left(l_{\text{eff}}\right)} \frac{(a r_{ij})^l}{(a+b)^{n_{\text{eff}}}} M\left(n_{\text{eff}}, l_{\text{eff}}, \frac{a^2 r_{ij}^2}{a^2+b^2}\right)$$

Custom Python basis definition

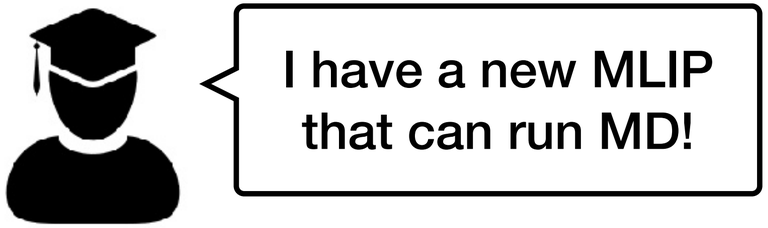


Full integration with PyTorch

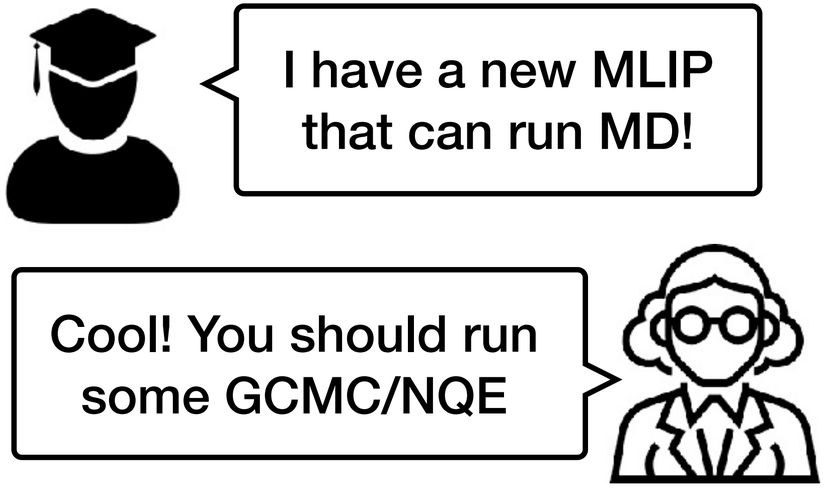
Usable from C/C++/Python/Rust/…

Sharing code for atomistic ML

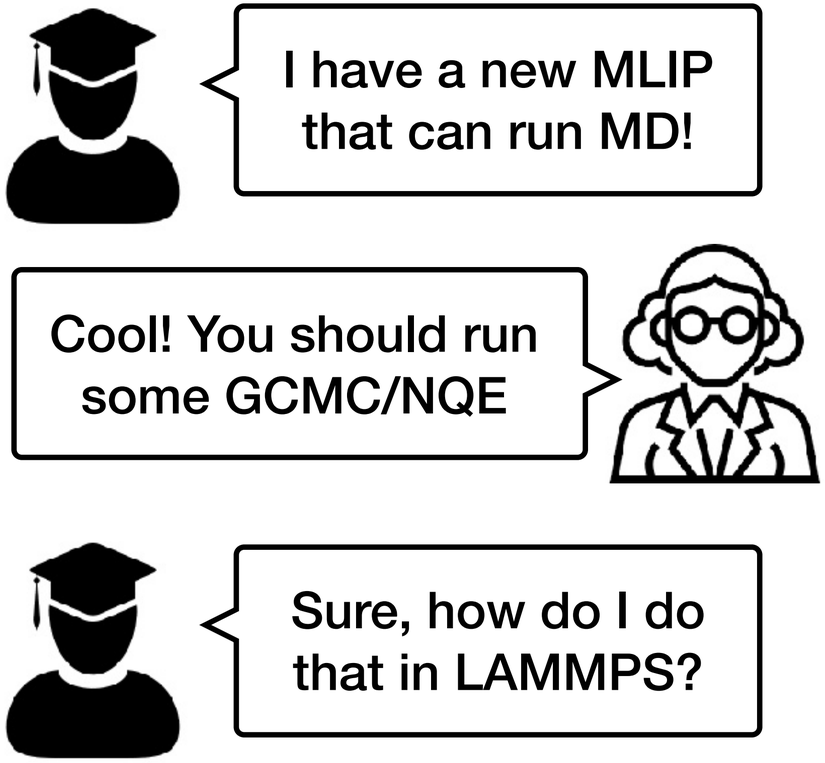
Sharing code for atomistic ML



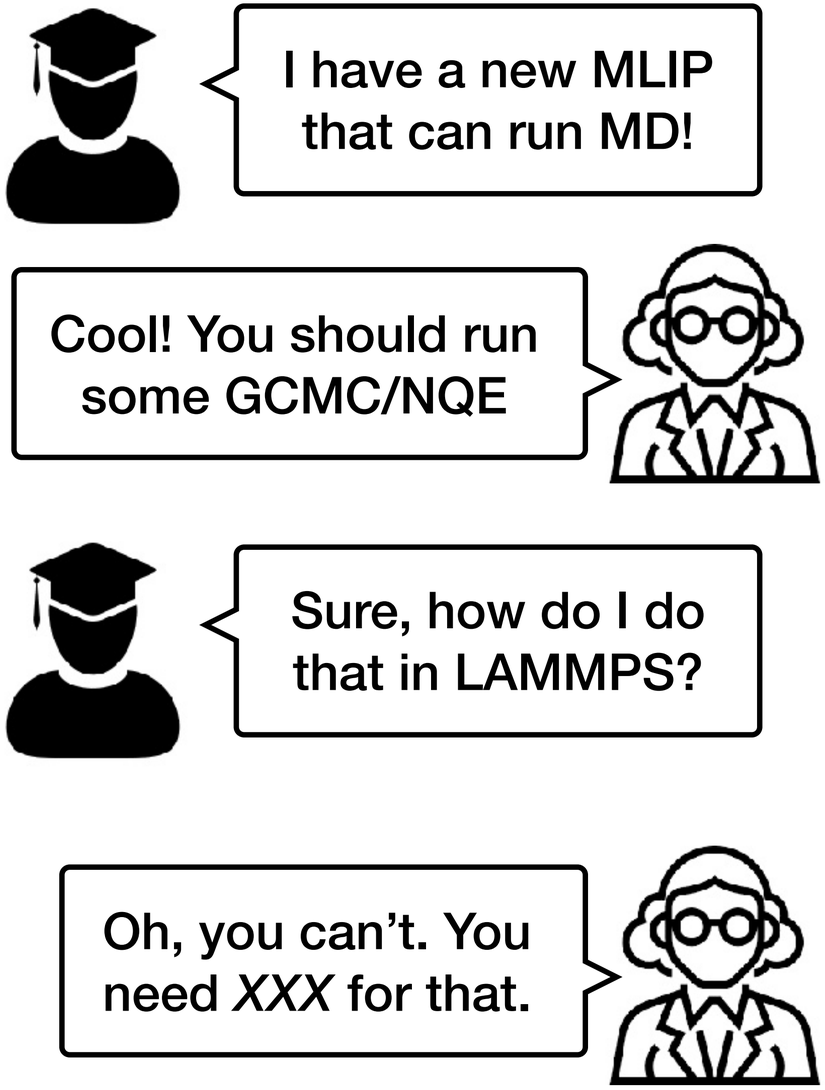
Sharing code for atomistic ML



Sharing code for atomistic ML



Sharing code for atomistic ML



Sharing ~~code~~ for atomistic ML

models

I have a new MLIP that can run MD!

Cool! You should run some GCMC/NQE

Sure, how do I do that in LAMMPS?

Oh, you can't. You need *XXX* for that.

Sharing ~~code~~ for atomistic ML

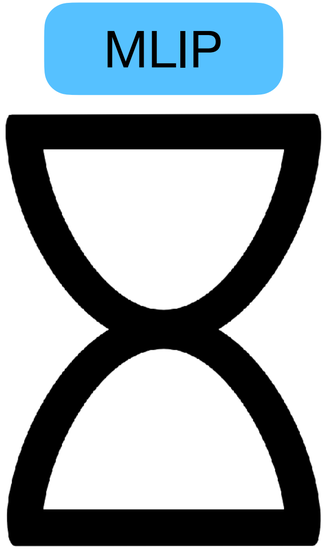
models

I have a new MLIP that can run MD!

Cool! You should run some GCMC/NQE

Sure, how do I do that in LAMMPS?

Oh, you can't. You need *XXX* for that.



Sharing ~~code~~ for atomistic ML

models

I have a new MLIP that can run MD!

Cool! You should run some GCMC/NQE

Sure, how do I do that in LAMMPS?

Oh, you can't. You need *XXX* for that.

Sharing ~~code~~ for atomistic ML

models

I have a new MLIP that can run MD!

Cool! You should run some GCMC/NQE

Sure, how do I do that in LAMMPS?

Oh, you can't. You need *XXX* for that.

Train and export the model **once**

Use it in **many** simulations!

Share with others, who can re-use your trained model

Sharing ~~code~~ models for atomistic ML

I have a new MLIP that can run MD!

Cool! You should run some GCMC/NQE

Sure, how do I do that in LAMMPS?

Oh, you can't. You need *XXX* for that.

Train and export the model **once**

Use it in **many** simulations!

Share with others, who can re-use your trained model

Write **fully custom models** with Python, run inside pure C++/Fortran engines

PyTorch

Metatensor based atomistic ML models

Exists

Planned

1

```
class MySuperModel(torch.nn.Module):
    def forward(self, ...):
        ...
```

Metatensor based atomistic ML models

Exists

Planned

1

```
class MySuperModel(torch.nn.Module):
    def forward(self, ...):
        ...
```

2

```
dataset = create_dataset()
model = train_model(dataset)
```

Metatensor based atomistic ML models

Exists

Planned

1

```
class MySuperModel(torch.nn.Module):
    def forward(self, ...):
        ...
```

2

```
dataset = create_dataset()
model = train_model(dataset)
```

3

```
model.export(capabilities)
```

define required neighbors lists,
the engine will compute them

define possible outputs (energy, dipoles, …)
the engine will request them

Metatensor based atomistic ML models

Exists

Planned

1

```
class MySuperModel(torch.nn.Module):
    def forward(self, ...):
        ...
```

2

```
dataset = create_dataset()
model = train_model(dataset)
```

3

```
model.export(capabilities)
```

define required neighbors lists,
the engine will compute them

define possible outputs (energy, dipoles, …)
the engine will request them

Metatensor based atomistic ML models

Exists

Planned

1

```
class MySuperModel(torch.nn.Module):
    def forward(self, ...):
        ...
```

2

```
dataset = create_dataset()
model = train_model(dataset)
```

3

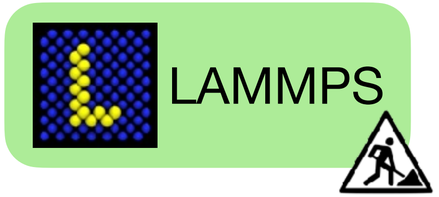
```
model.export(capabilities)
```

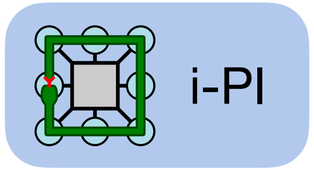
define required neighbors lists,
the engine will compute them

define possible outputs (energy, dipoles, …)
the engine will request them

0

Metatensor-Engine interface, written once





Metatensor based atomistic ML models

Exists

Planned

1

```
class MySuperModel(torch.nn.Module):
    def forward(self, ...):
        ...
```

2

```
dataset = create_dataset()
model = train_model(dataset)
```

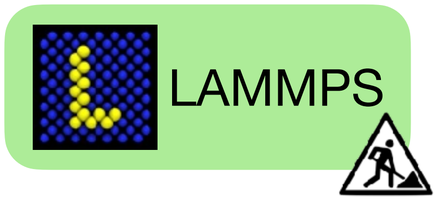
3

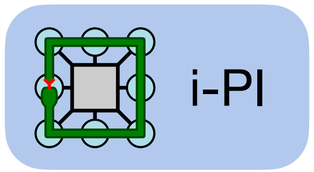
```
model.export(capabilities)
```

define required neighbors lists,
the engine will compute them

define possible outputs (energy, dipoles, …)
the engine will request them

0 Metatensor-Engine interface, written once





1 download pre-trained models

2

```
pair_style metatensor my-model.pt
```

3

```
run simulation && make paper
```

Metatensor based atomistic ML models

Exists

Planned

1

```
class MySuperModel(torch.nn.Module):
    def forward(self, ...):
        ...
```

2

```
dataset = create_dataset()
model = train_model(dataset)
```

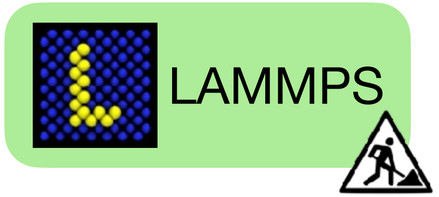
3

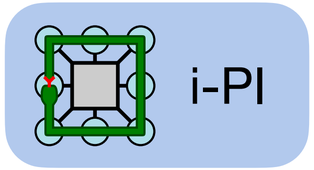
```
model.export(capabilities)
```

define required neighbors lists,
the engine will compute them

define possible outputs (energy, dipoles, …)
the engine will request them

0 Metatensor-Engine interface, written once





1 download pre-trained models

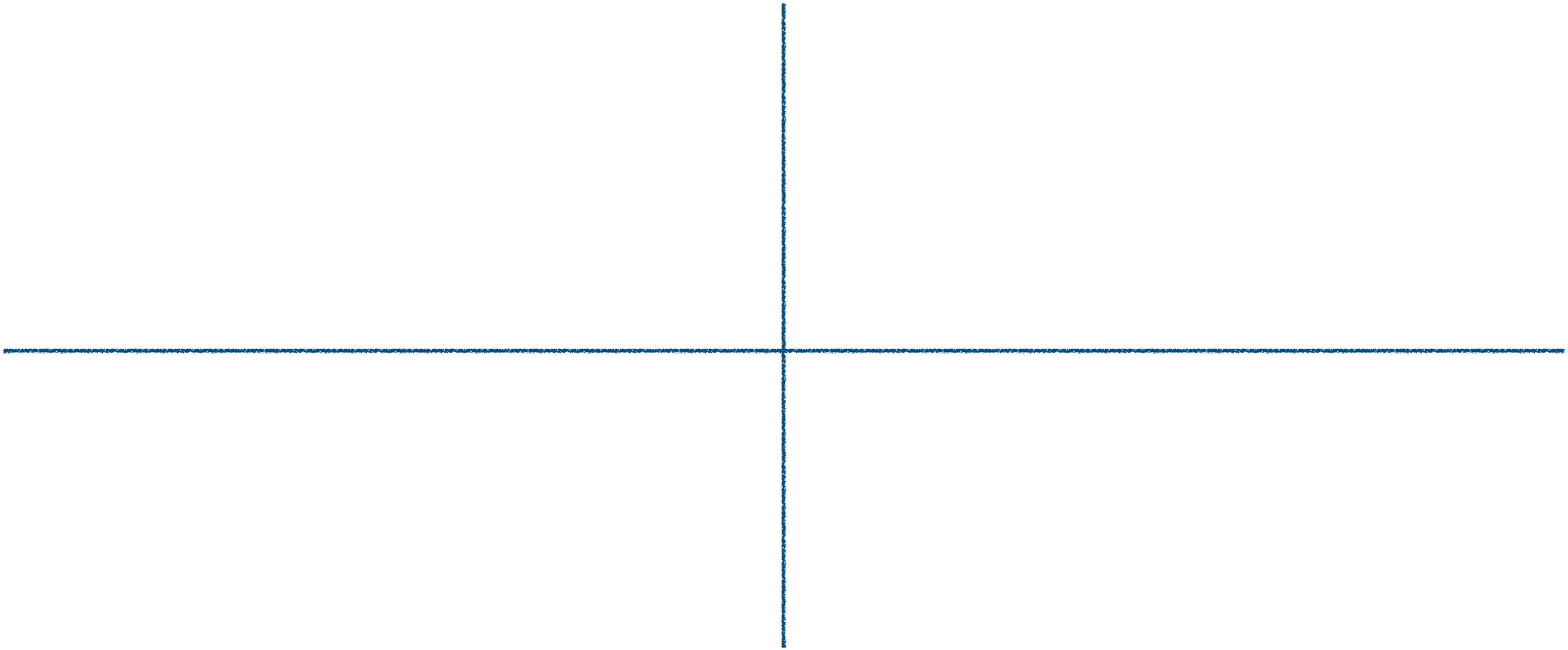
2 `pair_style metatensor my-model.pt`

3 `run simulation && make paper`

Tutorial later today: you'll write your own model and run MD using it in LAMMPS and ASE!

What do I gain?

You are …



What do I gain?

You are …

Developing new models, representations, *etc.*

Make your architecture available to everyone immediately and with fewer efforts

Delegate work on simulation engine interface

What do I gain?

You are …

Developing new models, representations, *etc.*

Make your architecture available to everyone immediately and with fewer efforts

Delegate work on simulation engine interface

Training new models

Easily tweak, re-use and combine existing models

Check your model performance immediately with Python simulations, and then at scale

What do I gain?

You are …

Developing new models, representations, *etc.*

Make your architecture available to everyone immediately and with fewer efforts

Delegate work on simulation engine interface

Training new models

Easily tweak, re-use and combine existing models

Check your model performance immediately with Python simulations, and then at scale

Running simulations for specific applications

Easy re-use and comparison of existing model

Use simulation software you are familiar with

What do I gain?

You are …

Developing new models, representations, *etc.*

Make your architecture available to everyone immediately and with fewer efforts

Delegate work on simulation engine interface

Training new models

Easily tweak, re-use and combine existing models

Check your model performance immediately with Python simulations, and then at scale

Running simulations for specific applications

Easy re-use and comparison of existing model

Use simulation software you are familiar with

Developing a simulation engine

Access the whole space of MLIP at once

Use ML surrogate for more than energy!

How can I adopt metatensor?

metatensor

lab-cosmo/metatensor

```
pip install metatensor
```

```
[dependencies]
metatensor = "0.1"
```

```
find_package(metatensor 0.1)
```

How can I adopt metatensor?

metatensor

lab-cosmo/metatensor

```
pip install metatensor
```

```
find_package(metatensor 0.1)
```

```
[dependencies]
metatensor = "0.1"
```

Surface integration: wrap and unwrap, using metatensor at the boundaries

Very easy to get started **today**! Can be one of many option supported by the code

How can I adopt metatensor?

metatensor

lab-cosmo/metatensor

```
pip install metatensor
```

```
[dependencies]
metatensor = "0.1"
```

```
find_package(metatensor 0.1)
```

Surface integration: wrap and unwrap, using metatensor at the boundaries

Very easy to get started **today**! Can be one of many option supported by the code

Deeper integration: manipulate metatensor data inside your code

No wrapping overhead, metadata helps debugging, helps staying in most memory efficient format

How can I adopt metatensor?

metatensor

lab-cosmo/metatensor

```
pip install metatensor
```

```
[dependencies]
metatensor = "0.1"
```

```
find_package(metatensor 0.1)
```

Surface integration: wrap and unwrap, using metatensor at the boundaries

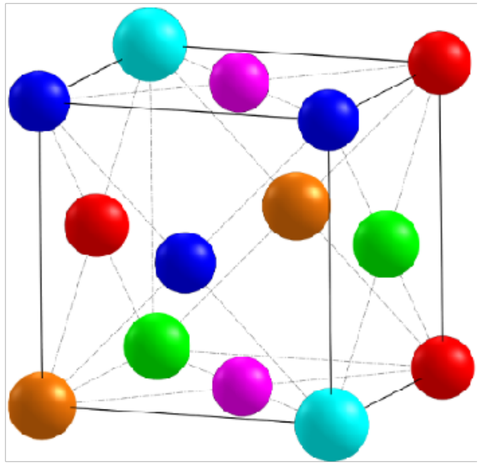
Very easy to get started **today**! Can be one of many option supported by the code

Deeper integration: manipulate metatensor data inside your code

No wrapping overhead, metadata helps debugging, helps staying in most memory efficient format

The code is in active development, we are taking feedback!

Cool science you can do with metatensor



High Entropy Alloys

Huge structure space, lot of complexity, hard to probe experimentally, tuneable properties

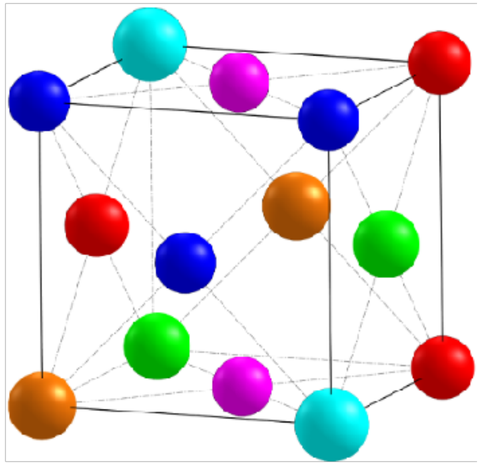
Very good system for atomistic simulations and MLIP

BUT: chemical complexity is hard to handle in MLIP

BASF

Natasha Lopanitsyna
Arslan Mazitov
Maximilian Springer
Sandip De
Michele Ceriotti

Cool science you can do with metatensor



High Entropy Alloys

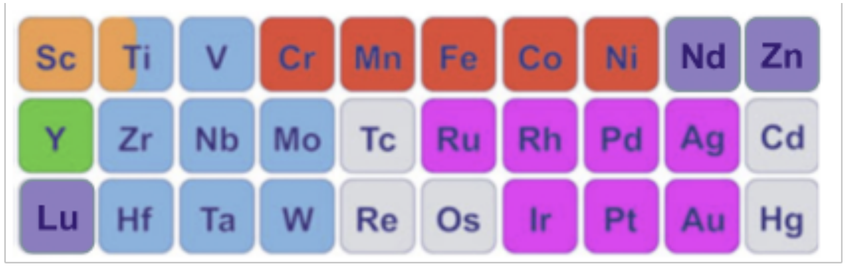
Huge structure space, lot of complexity, hard to probe experimentally, tuneable properties

Very good system for atomistic simulations and MLIP

BUT: chemical complexity is hard to handle in MLIP

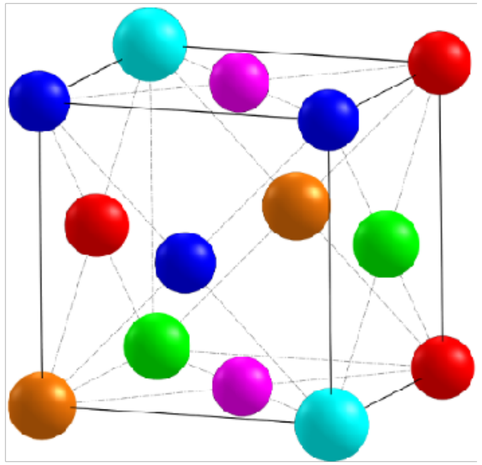
BASF

Natasha Lopanitsyna
Arslan Mazitov
Maximilian Springer
Sandip De
Michele Ceriotti



High Entropy Alloy dataset, 25 elements,
30k distorted bulk structures, DFT energies

Cool science you can do with metatensor



High Entropy Alloys

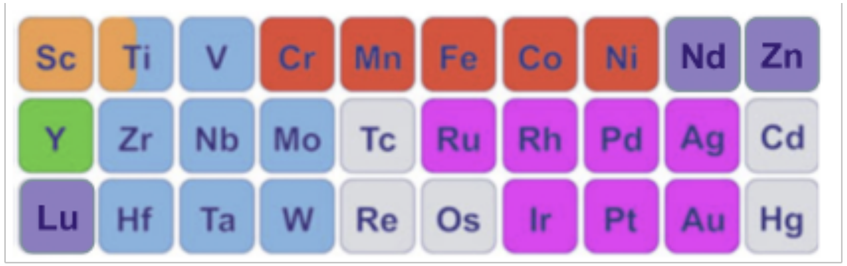
Huge structure space, lot of complexity, hard to probe experimentally, tuneable properties

Very good system for atomistic simulations and MLIP

BUT: chemical complexity is hard to handle in MLIP

BASF

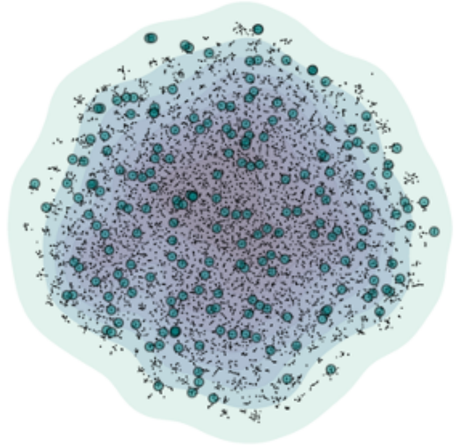
Natasha Lopanitsyna
Arslan Mazitov
Maximilian Springer
Sandip De
Michele Ceriotti



High Entropy Alloy dataset, 25 elements, 30k distorted bulk structures, DFT energies

Generate 1M random bulk structures, with random rattling. Higher energies than MD.

Subselect 30k covering most of the space (FPS) with radial descriptors



Exploiting atomic similarities in SOAP

$$\langle \alpha n \ \alpha' n' \ l \,|\, \rho_i^{\otimes 2} \rangle = \sum_m \langle \alpha n l m \,|\, \rho_i \rangle \otimes \langle \alpha' n' l m \,|\, \rho_i \rangle$$

SOAP power spectrum
(invariant)

Quadratic space scaling with number of species:
10 TiB of storage just for SOAP vectors (sparse!)

Exploiting atomic similarities in SOAP

$$\langle \alpha n \; \alpha' n' \; l \,|\, \rho_i^{\otimes 2} \rangle = \sum_m \langle \alpha n l m \,|\, \rho_i \rangle \otimes \langle \alpha' n' l m \,|\, \rho_i \rangle$$

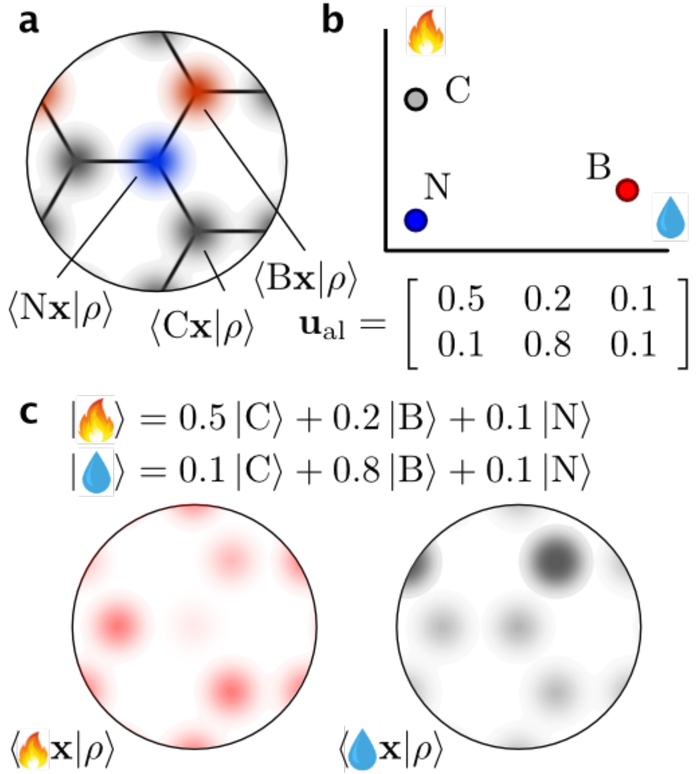
SOAP power spectrum (invariant)

Quadratic space scaling with number of species: 10 TiB of storage just for SOAP vectors (sparse!)

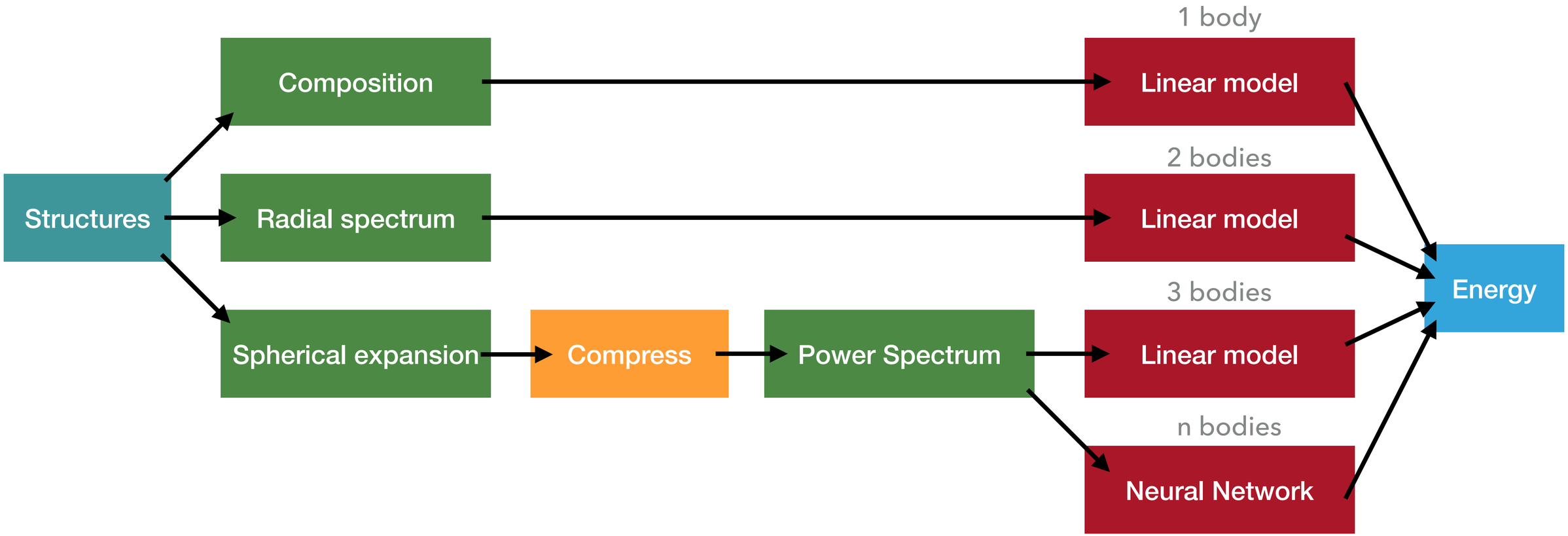
One solution: project real elements onto alchemical "*pseudo-elements*", optimizing the projection matrix in a data driven way

$$\langle \beta n l m \,|\, \rho_i \rangle = U_{\alpha\beta} \, \langle \alpha n l m \,|\, \rho_i \rangle$$

$$\langle \beta n \; \beta' n' \; l \,|\, \rho_i^{\otimes 2} \rangle = \sum_m \langle \beta n l m \,|\, \rho_i \rangle \otimes \langle \beta' n' l m \,|\, \rho_i \rangle$$

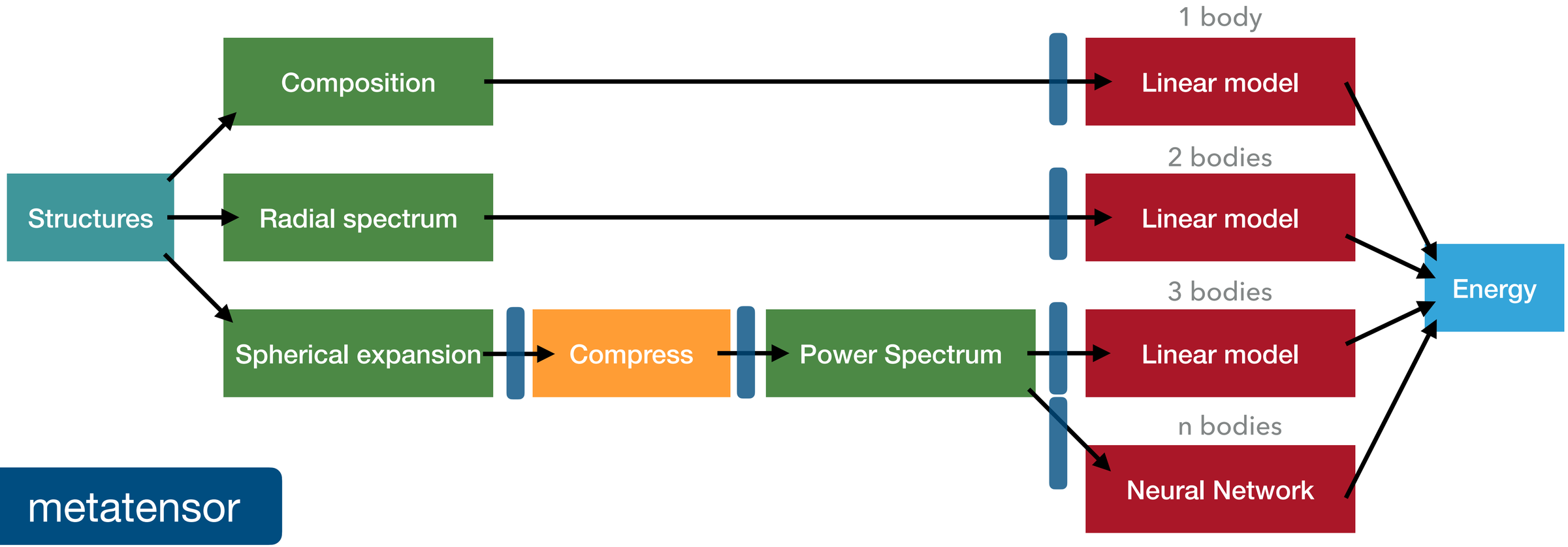


How to do this with metatensor?



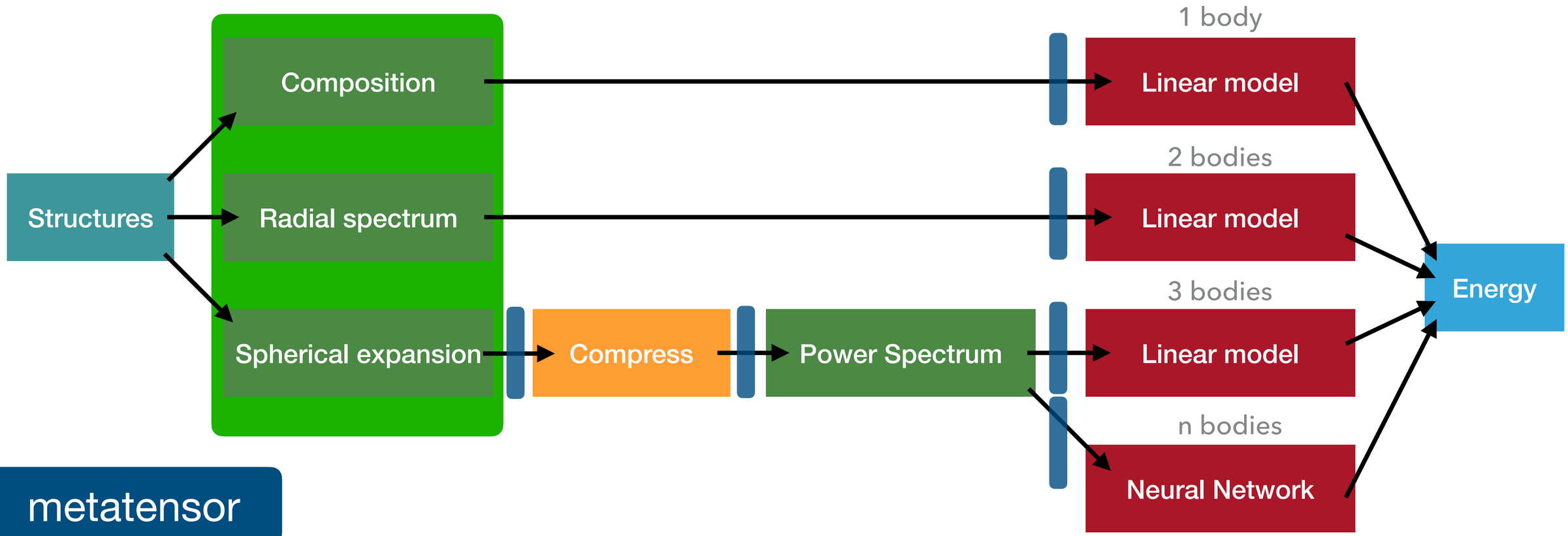
Use Torch integration to learn both compression matrix & models weights simultaneously

How to do this with metatensor?



Use Torch integration to learn both compression matrix & models weights simultaneously

How to do this with metatensor?

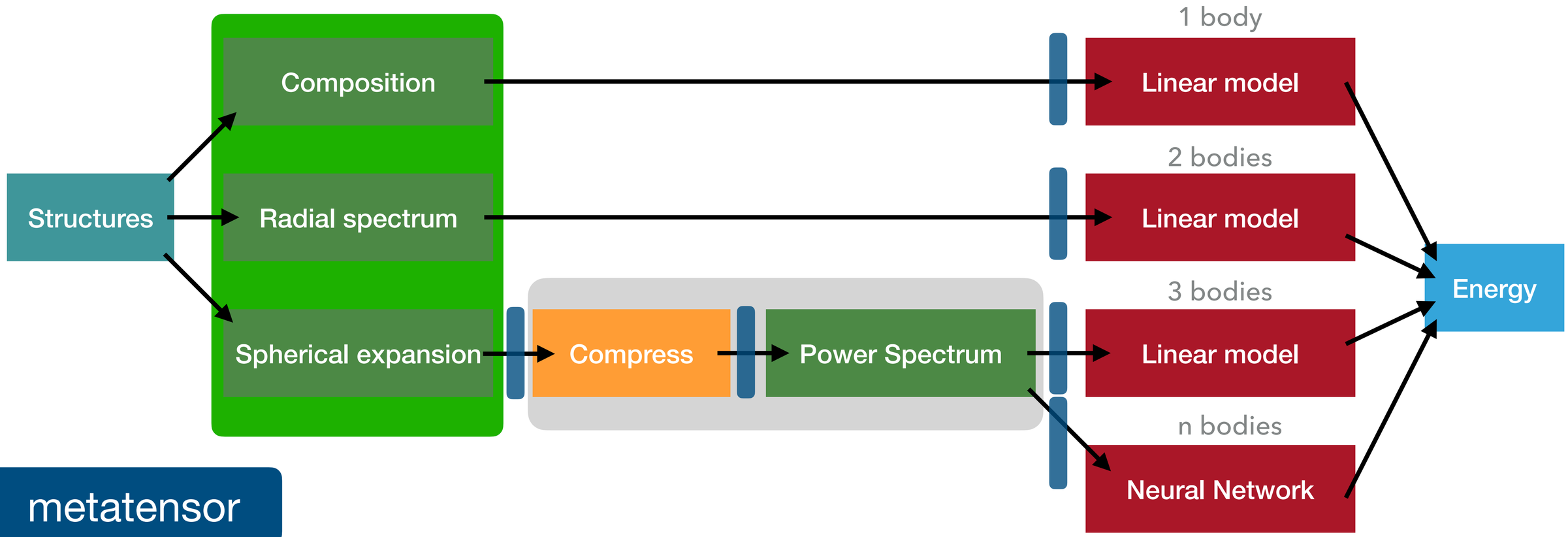


metatensor

rascaline

Use Torch integration to learn both compression matrix & models weights simultaneously

How to do this with metatensor?



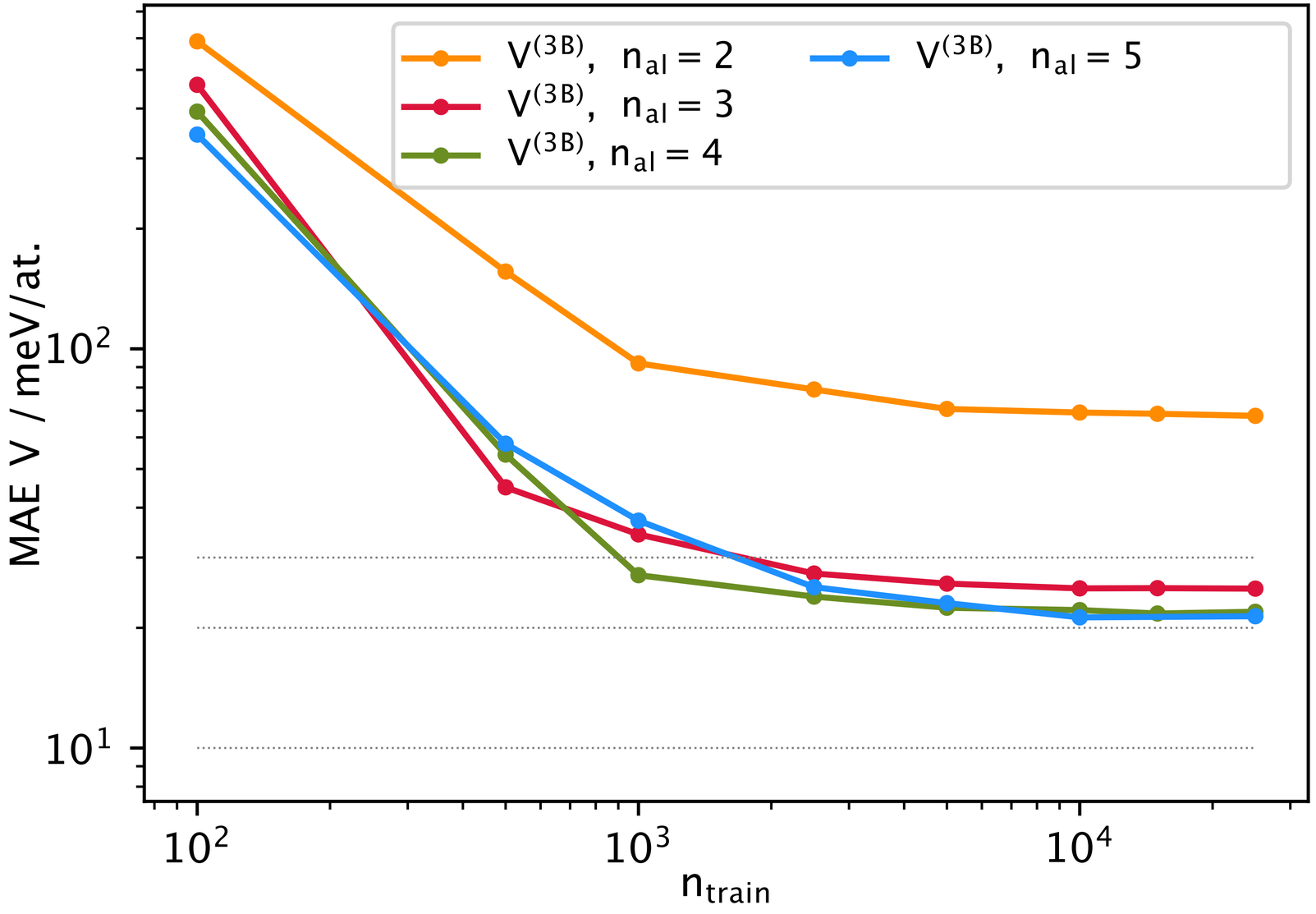
metatensor

rascaline

custom

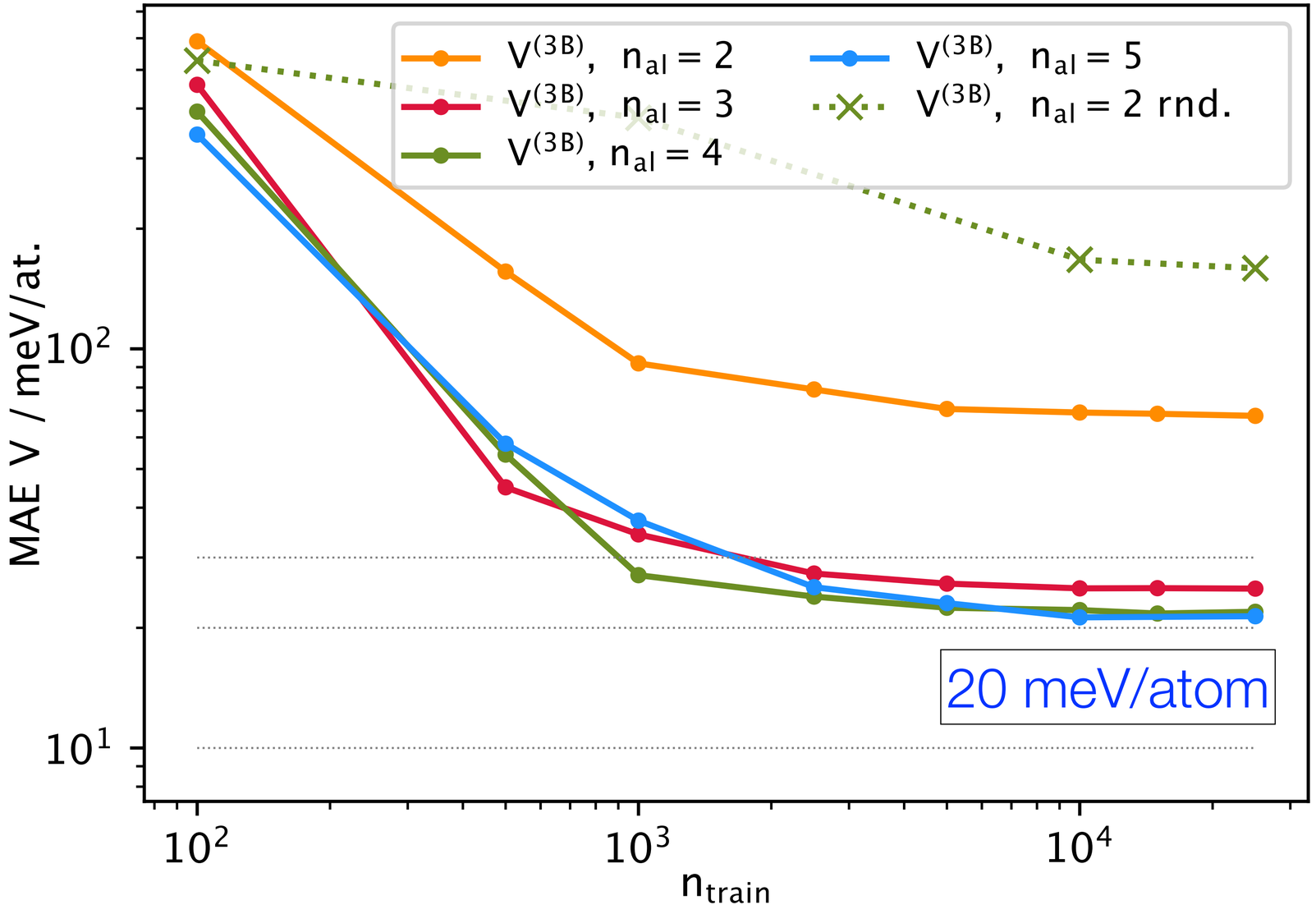
Use Torch integration to learn both compression matrix & models weights simultaneously

Validating the architecture



4 pseudo-elements can capture all interactions

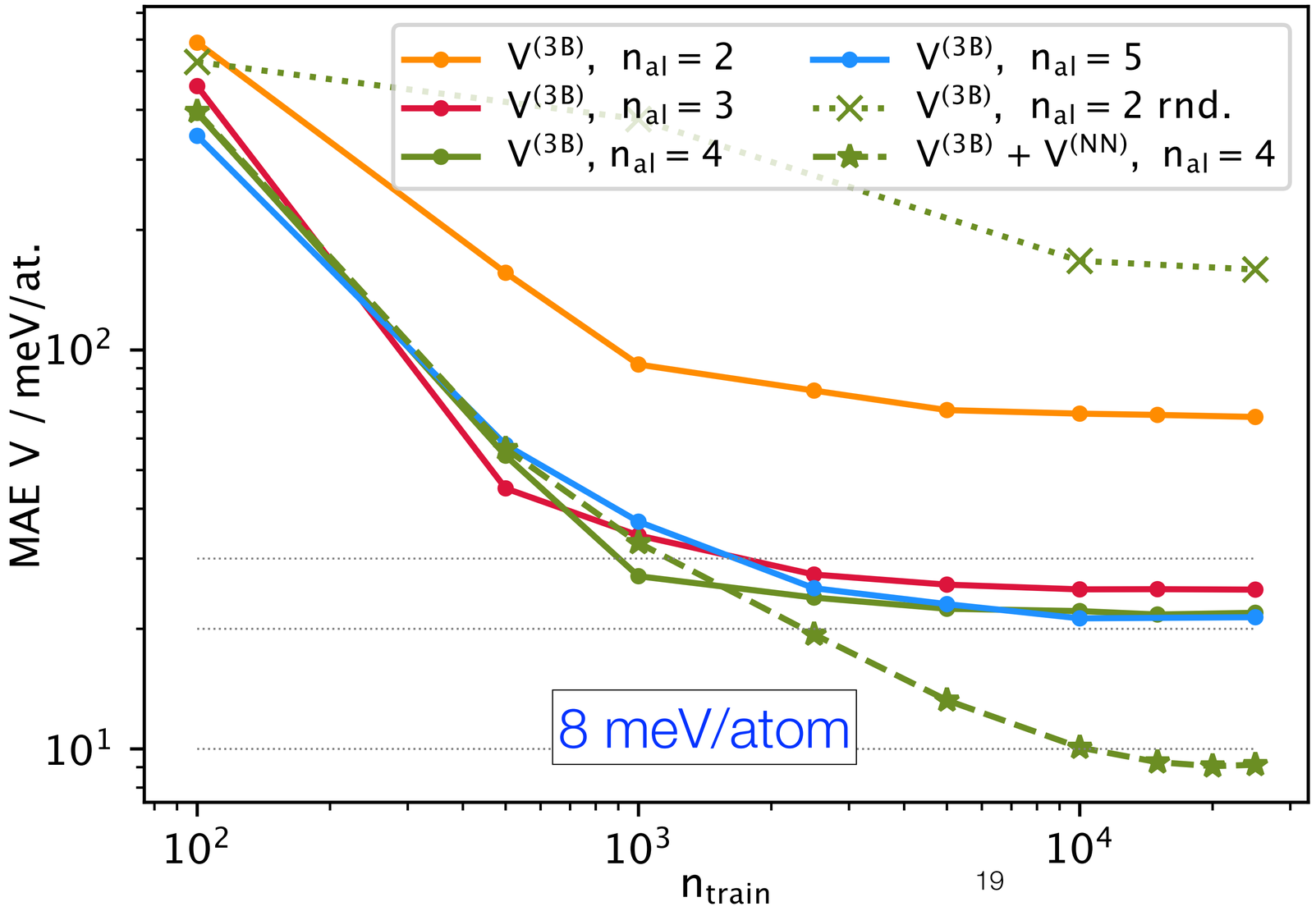
Validating the architecture



4 pseudo-elements can capture all interactions

Large improvement by optimising projection

Validating the architecture

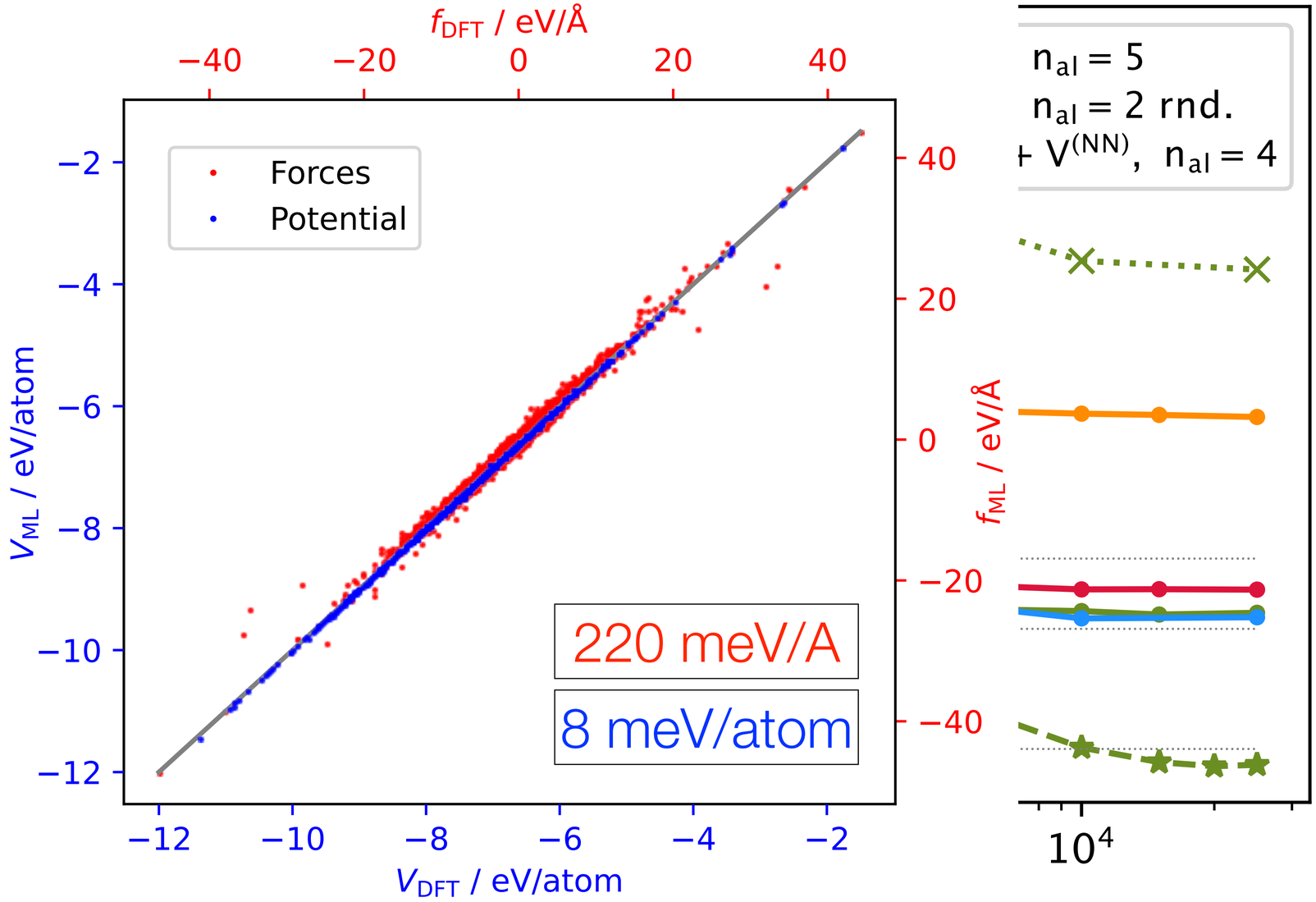


4 pseudo-elements can capture all interactions

Large improvement by optimising projection

Non-linearities to increase body-order

Validating the architecture

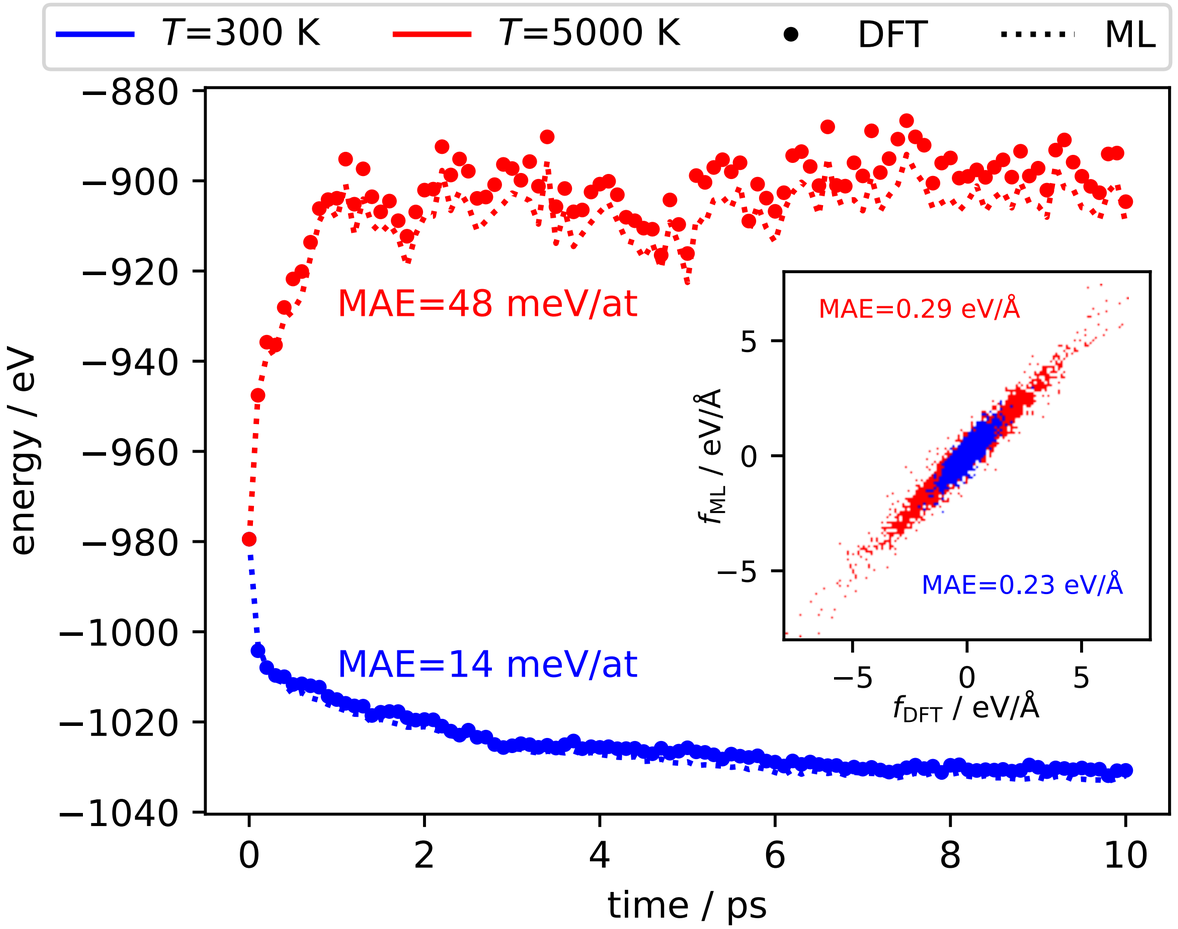


4 pseudo-elements can capture all interactions

Large improvement by optimising projection

Non-linearities to increase body-order

Validating the stability

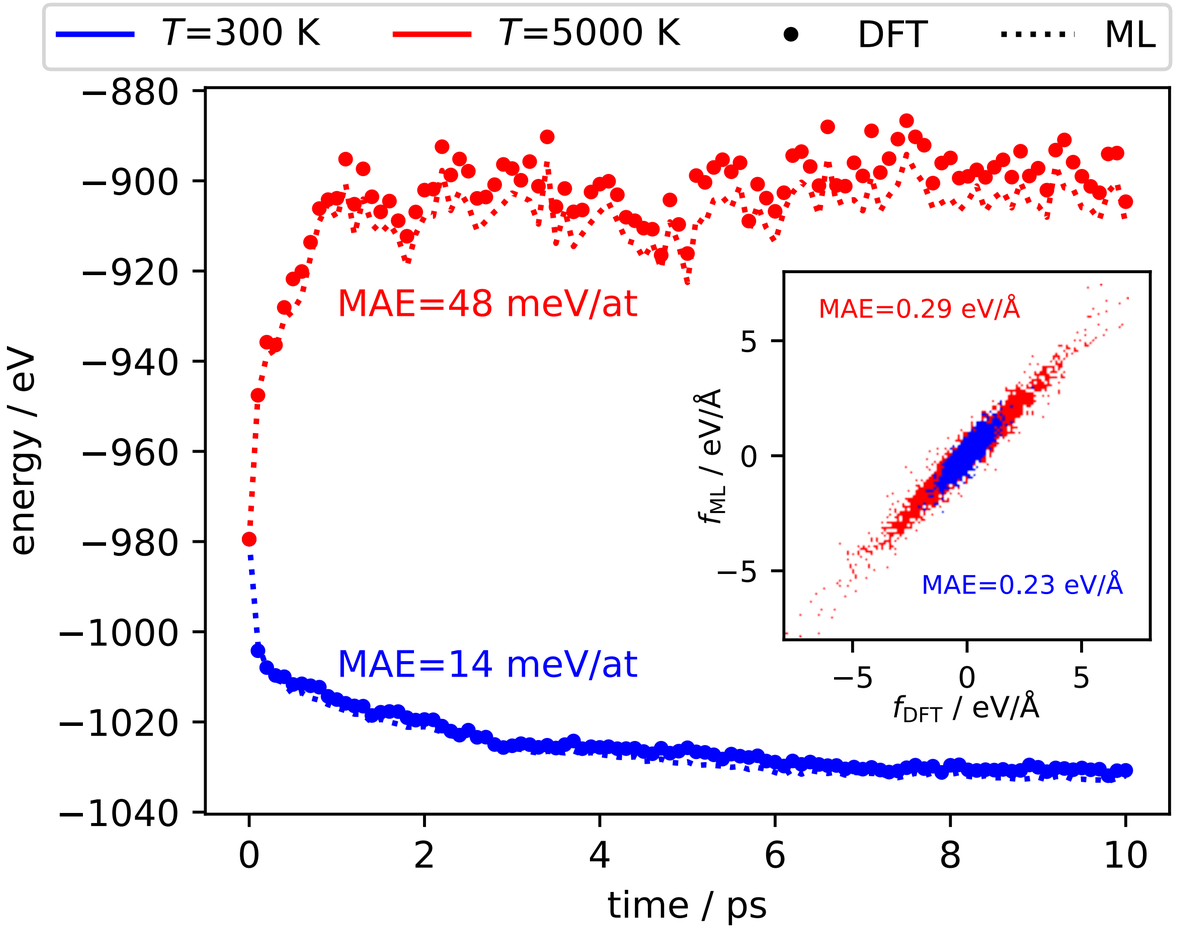


Extremely stable for bulk structures

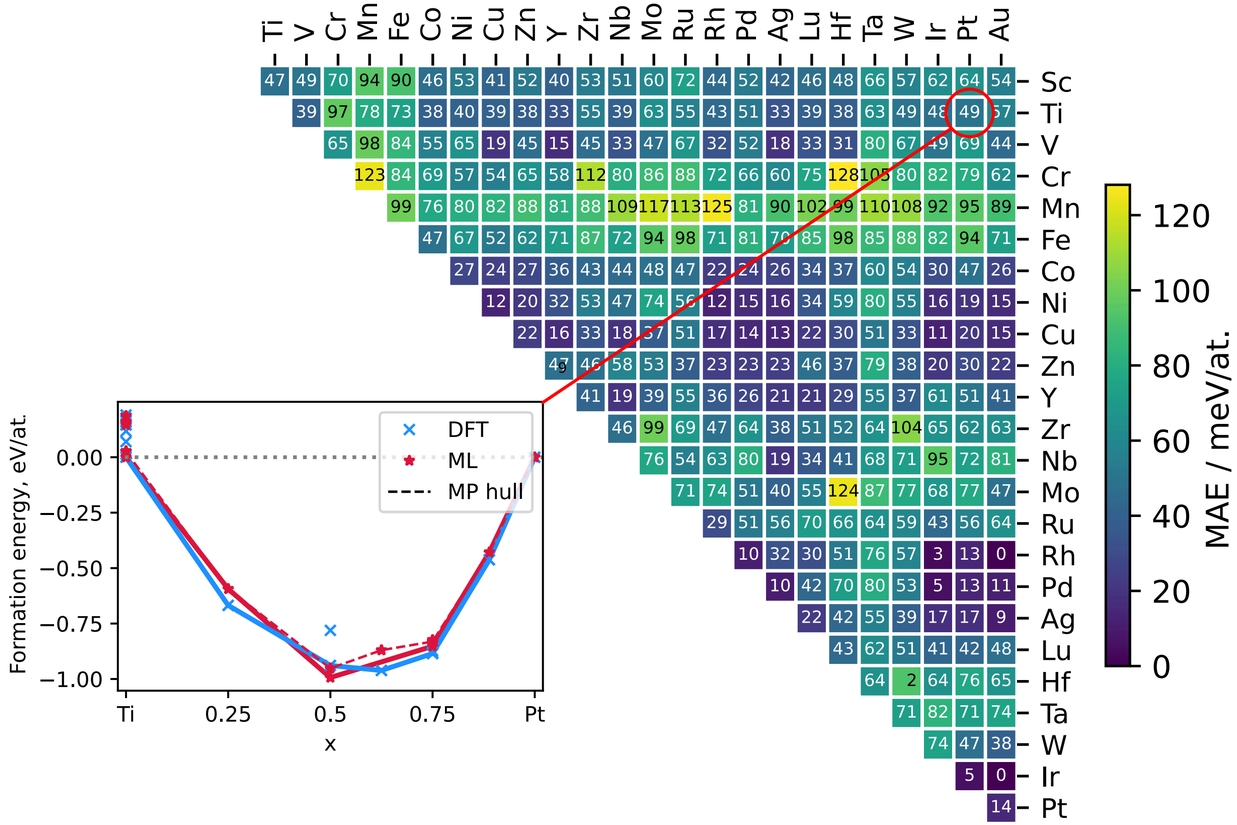
Able to run REMD, melted trajectories

Lopanitsyna, Fraux, Springer, De, Ceriotti, Phys. Rev. Mat. 2023

Validating the stability



Good extrapolation to data outside the training set



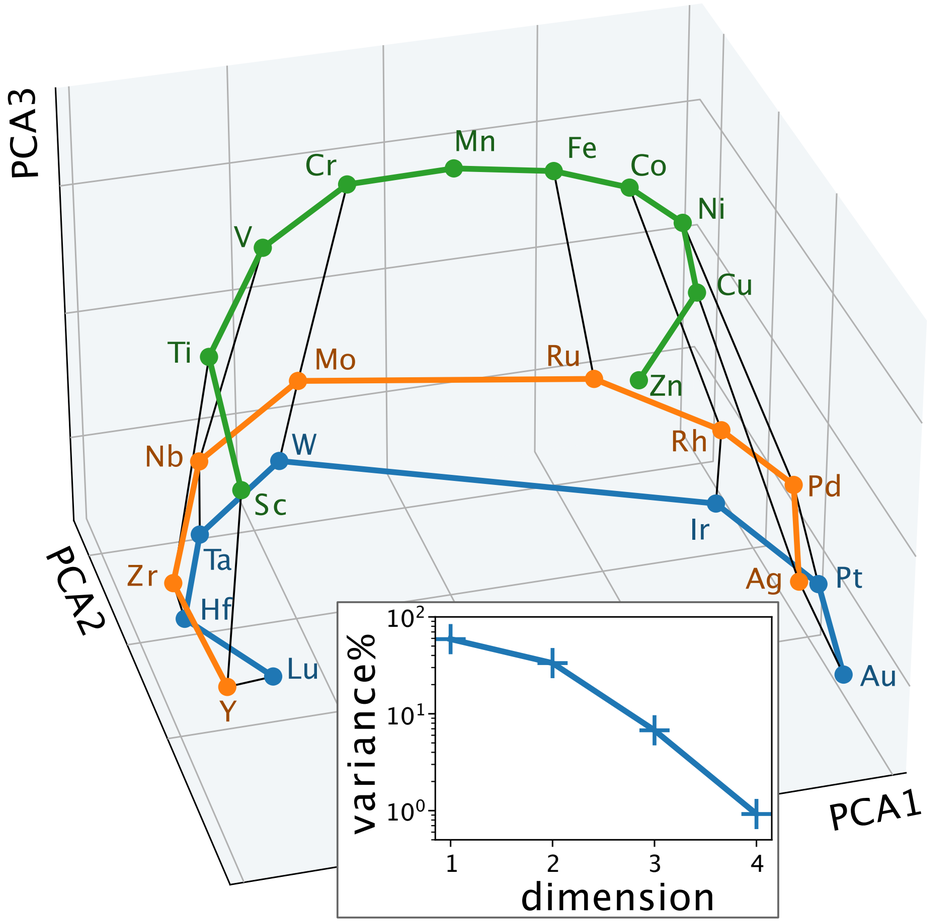
Extremely stable for bulk structures

Able to run REMD, melted trajectories

Lopanitsyna, Fraux, Springer, De, Ceriotti, Phys. Rev. Mat. 2023

Data driven periodic table

4 pseudo-elements are enough to capture all the interactions



$$\langle \beta nlm \,|\, \rho_i \rangle = U_{\alpha\beta} \, \langle \alpha nlm \,|\, \rho_i \rangle$$

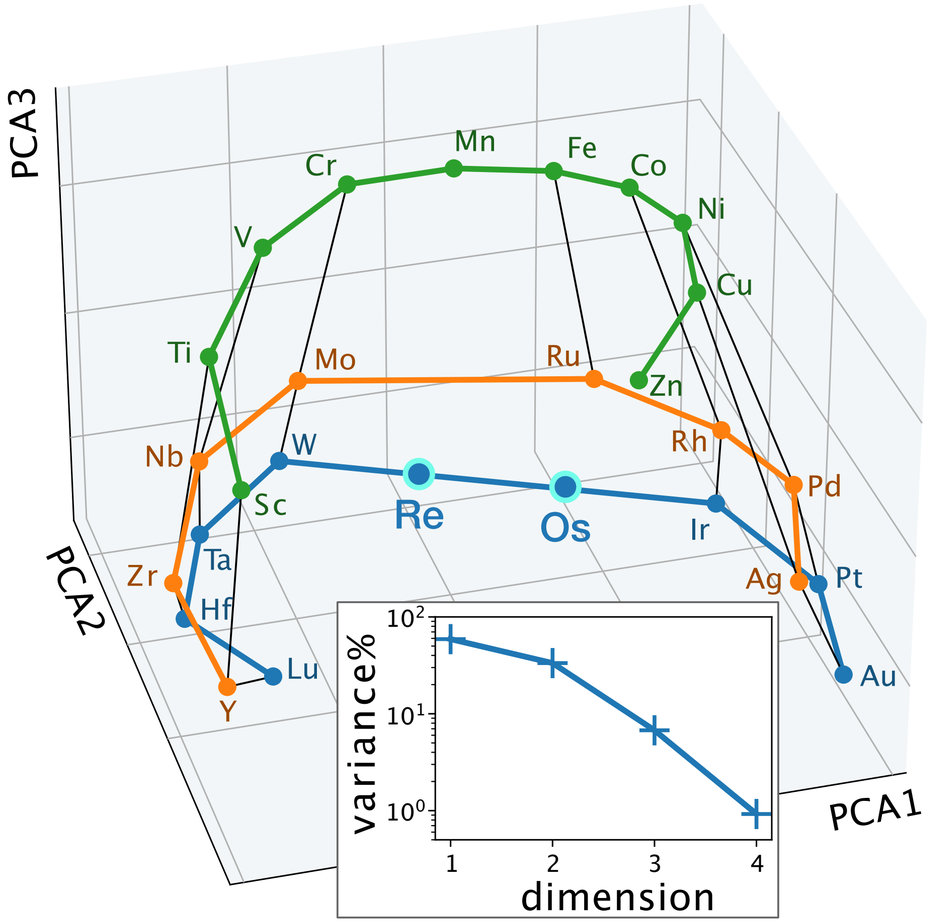
Data-driven periodic table structure

Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Nd	Zn
Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd
Lu	Hf	Ta	W	Re	Os	Ir	Pt	Au	Hg

Lopanitsyna, Fraux, Springer, De, Ceriotti, Phys. Rev. Mat. 2023

Data driven periodic table

4 pseudo-elements are enough to capture all the interactions



$$\langle \beta nlm \,|\, \rho_i \rangle = U_{\alpha\beta} \, \langle \alpha nlm \,|\, \rho_i \rangle$$

Data-driven periodic table structure

Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Nd	Zn
Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd
Lu	Hf	Ta	W	Re	Os	Ir	Pt	Au	Hg

Able to extrapolate to **new elements**!

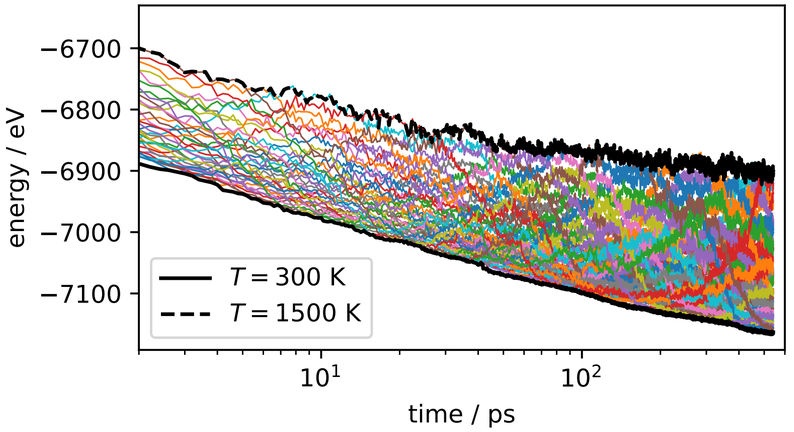
Lopanitsyna, Fraux, Springer, De, Ceriotti, Phys. Rev. Mat. 2023

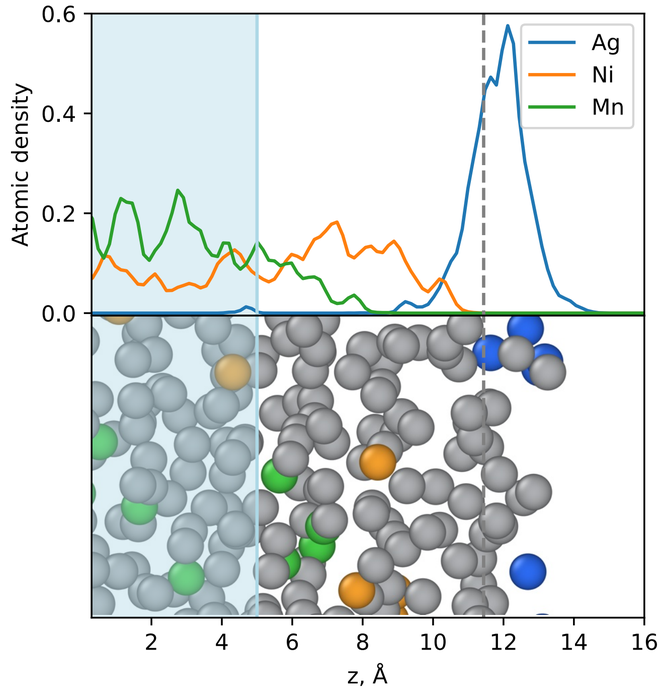
Surfaces segregation

Same element, add surfaces and melted structures

Identical compression weights

Get surfaces propensity from Replica-Exchange MD



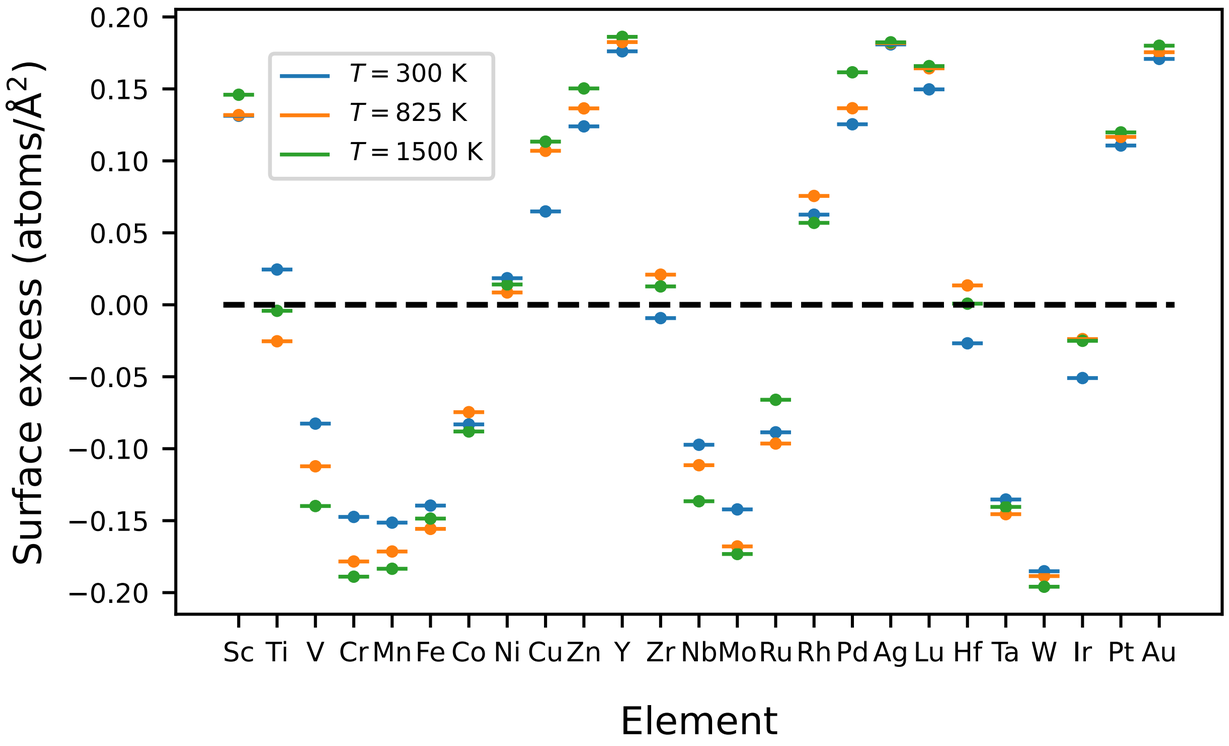


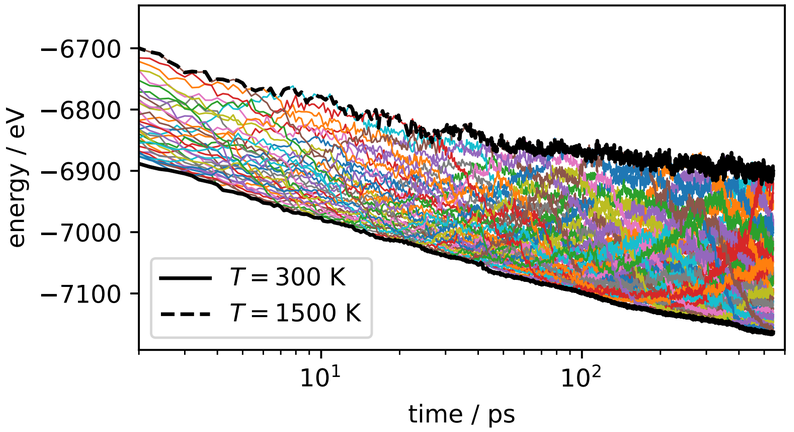
Surfaces segregation

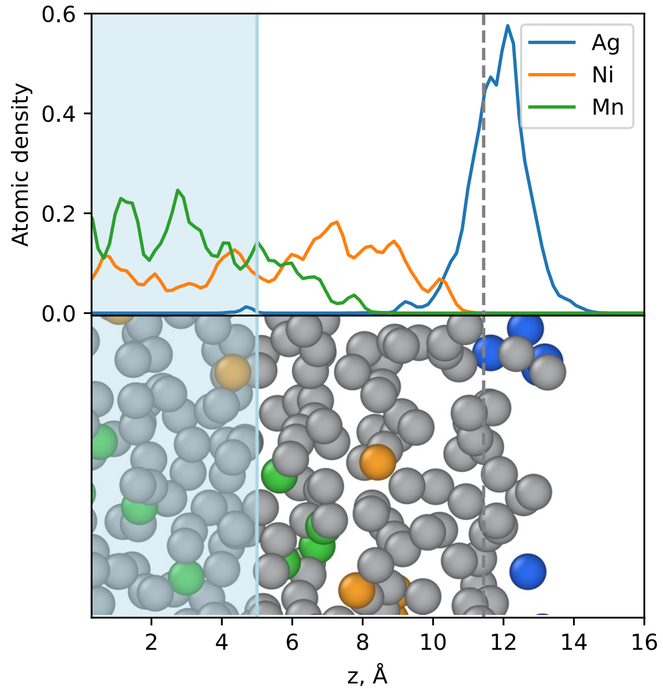
Same element, add surfaces and melted structures

Identical compression weights

Get surfaces propensity from Replica-Exchange MD

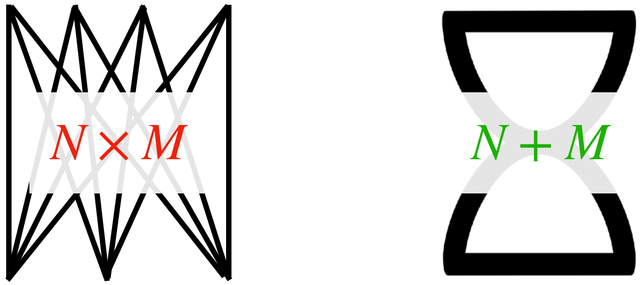




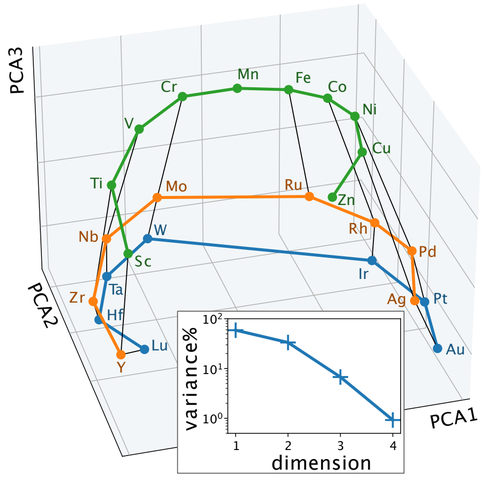


In summary

Sharing **data** for atomistic machine learning



Model for 25 elements,
with learned periodic table



Sharing **code** for atomistic machine learning

Tutorial later today: custom model,
ASE and LAMMPS simulations

@Luthaf

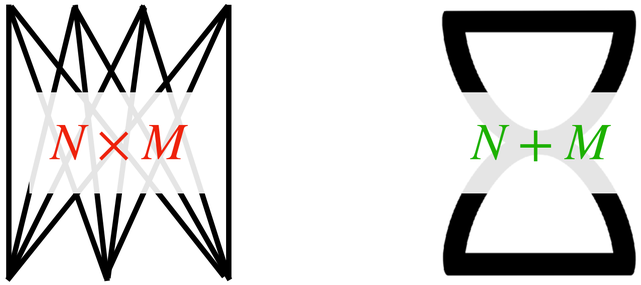
@__luthaf__

mas.to/@luthaf

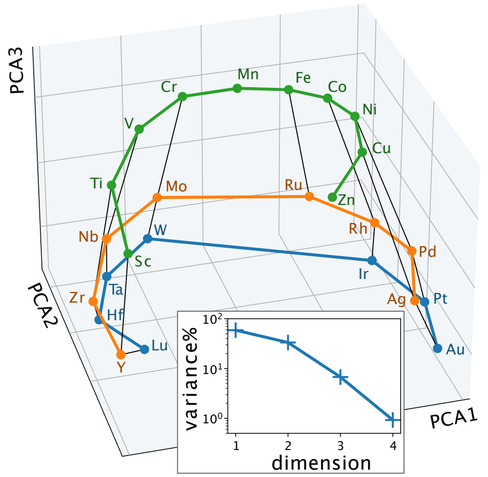
guillaume.fraux@epfl.ch

In summary

Sharing **data** for atomistic machine learning



Model for 25 elements,
with learned periodic table



Sharing **code** for atomistic machine learning

Tutorial later today: custom model,
ASE and LAMMPS simulations

Thank you for your attention!

@Luthaf

@__luthaf__

mas.to/@luthaf

guillaume.fraux@epfl.ch