PANNA

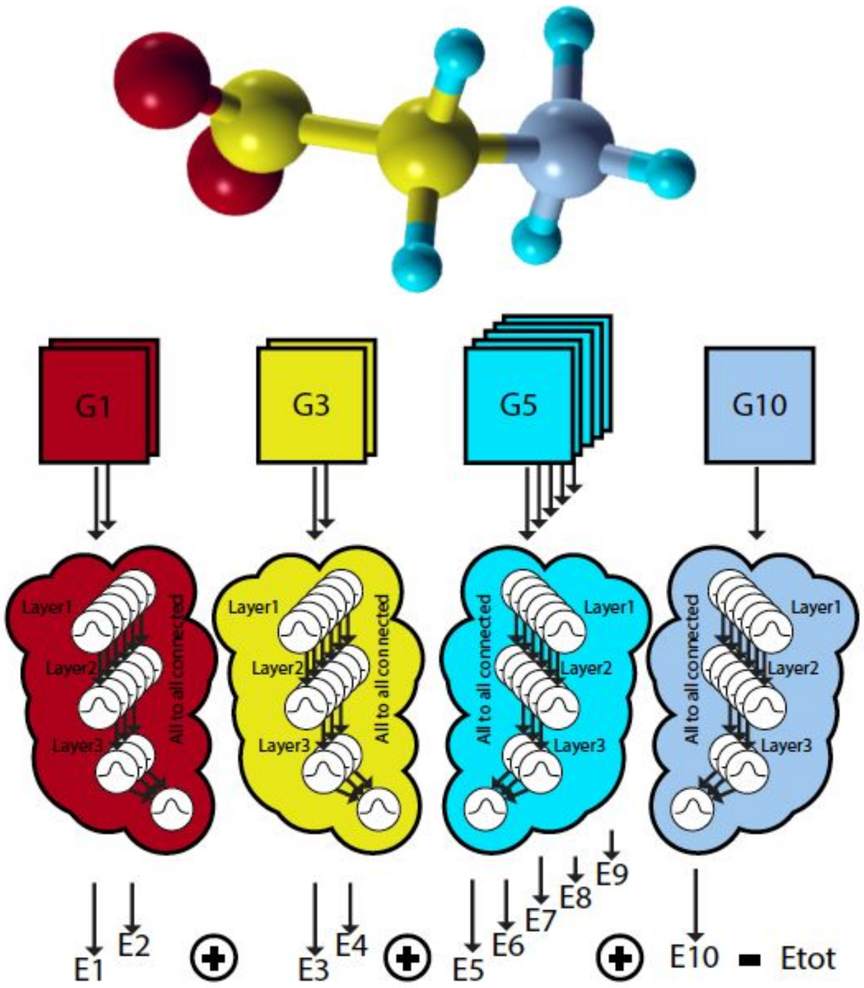
Properties from Artificial Neural Network Architectures

Ruggero Lot, Franco Pellegrini, Yusuf Shaidu, Emine Küçükbenli
SISSA, Trieste, Italy

SISSA
40!

NN4MS

- Atomic environment descriptors
- Fully connected feedforward NN
- Atomic energies ➔ Total energy
- Derivatives ➔ Forces

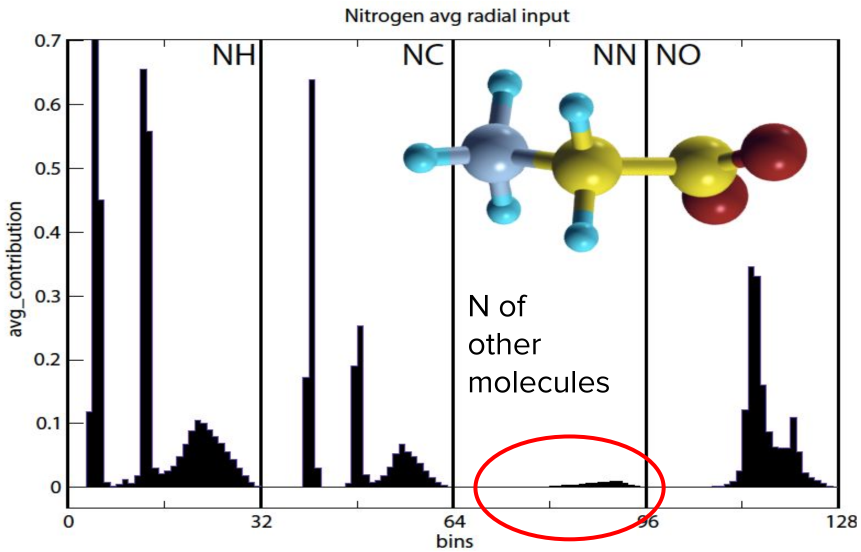


Atomic environment descriptors

(modified) Behler-Parrinello

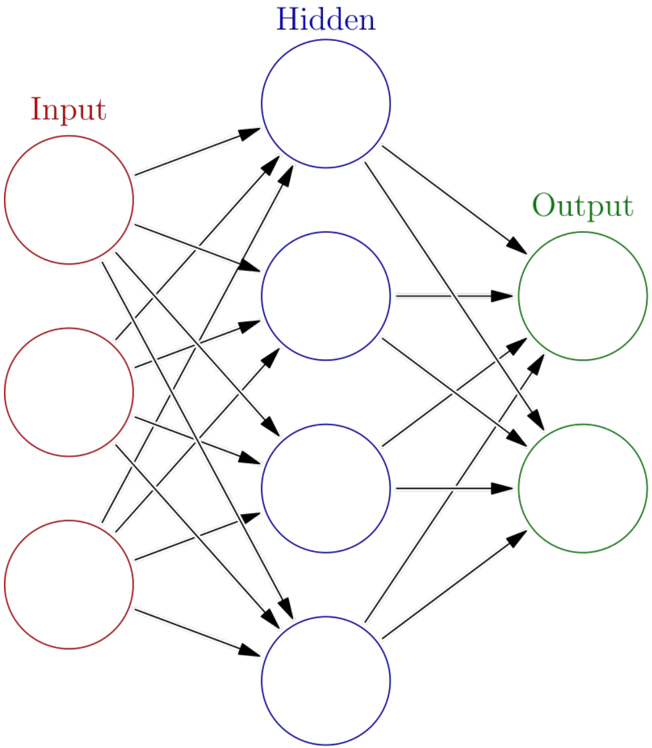
$$G^R_{m,s;i} = \sum_{i \neq j}^{\text{All atoms kind s}} e^{-\eta(r_{ij}-R_m)^2} f_c(r_{ij})$$

$$f_c(r_{ij}) = \begin{cases} 0.5\left[\cos\left(\frac{\pi r_{ij}}{R_c}\right)+1\right] & \text{if } r_{ij} \leq R_c \\ 0 & \text{if } r_{ij} \geq R_c \end{cases}$$



$$G^A_{n,m,s;i} = 2^{1-\xi} \sum_{j,k \neq i}^{\text{All atom of kind s}} (1+\lambda cos(\Theta_{ijk}-\Theta_n))^{\xi} \; e^{-\eta\left(\frac{r_{ij}+r_{ik}}{2}-R_m\right)^2} f_c(r_{ij}) f_c(r_{ik})$$

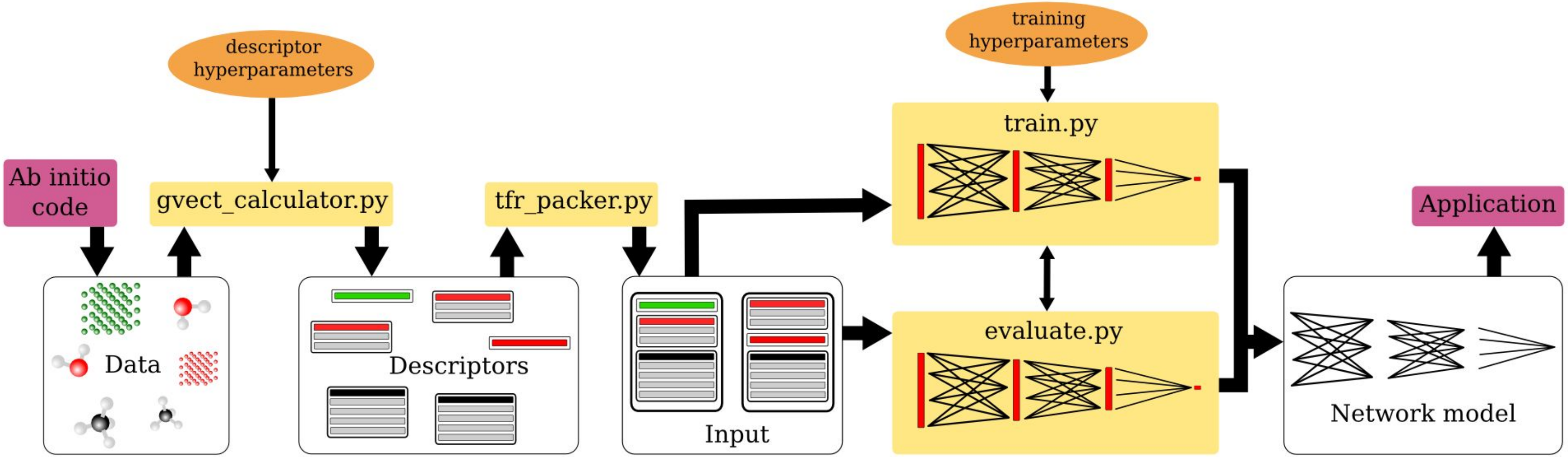
Fully connected feedforward neural networks



$$a_j^{l+1} = g\left(\sum_{i=1}^{n_l} a_i^l W_{ij}^{l+1} + b_j^{l+1}\right)$$

- Simplest deep neural network structure
- Activation function g( )
- Cost $C(\{\bar{O}_i - O_i\})$
- Weights optimized with backpropagation
- Sampling with SGD (➜ Adam)

PANNA pipeline



PANNA features

- Training from simple input files, with TensorFlow efficiency (C/G/TPU) (No need to learn TF, no need to write python scripts)
- Data processing: input parsing (xyz, QE), descriptor (BP, mBP), packaging (TFR)
- Simple definition of architecture: species resolution, trainability, activations (Gaussian, ReLU, RBF), regularizations (L1, L2, clipping), cost functions, learning schedule
- Learning & Predicting forces
- Visualization with Tensorboard
- Molecular dynamics: Export models to LAMMPS, OpenKIM

Sample input

```
[IO_INFORMATION]
data_dir = ./tfr_train
train_dir = ./train
log_frequency = 10
save_checkpoint_steps = 500
```

Files/logging info

```
[DATA_INFORMATION]
atomic_sequence = H, O
output_offset = -13.62, -2041.84
```

System info

```
[TRAINING_PARAMETERS]
batch_size = 50
learning_rate = 0.01
max_steps = 5000
```

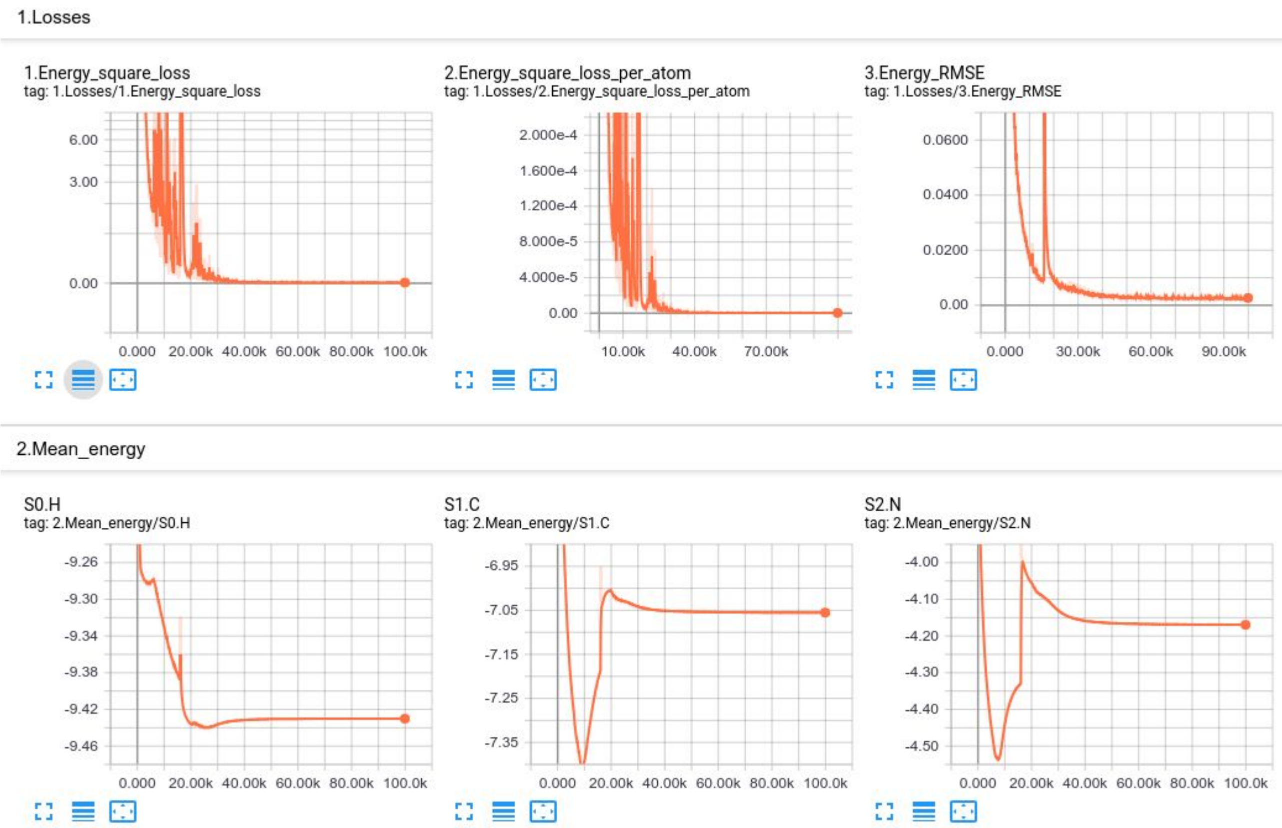
Training hyperparameters

```
[DEFAULT_NETWORK]
g_size = 128
architecture = 64:32:1
trainable = 1:1:1
```

Modular Networks by species:

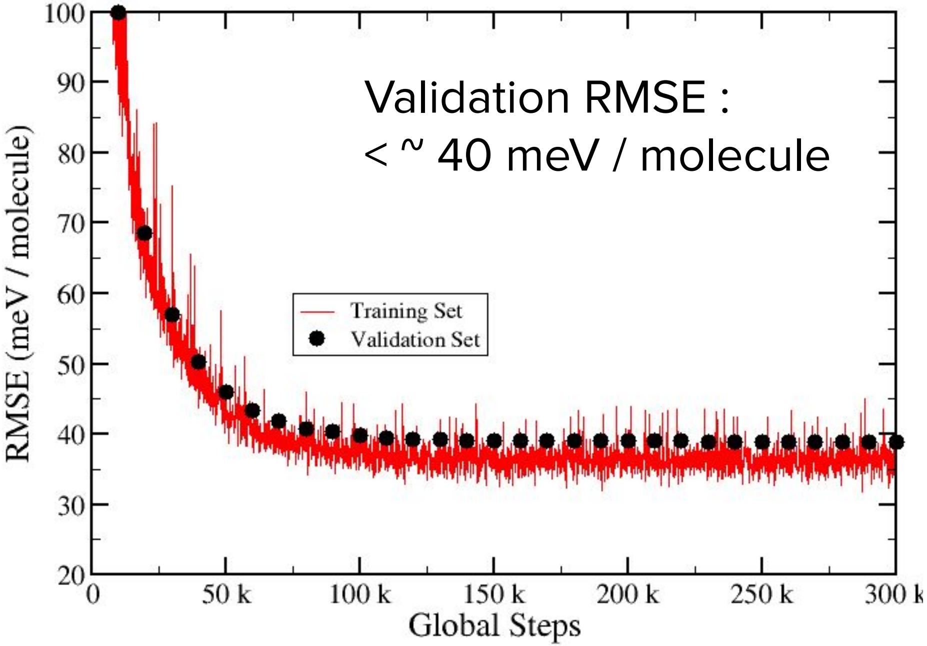
```
[H]
architecture = 128:32:32:1
trainable = 1:1:1:1
activations = 1:1:1:0
```

Tensorboard visualization



Tensorboard visualization

Example of training

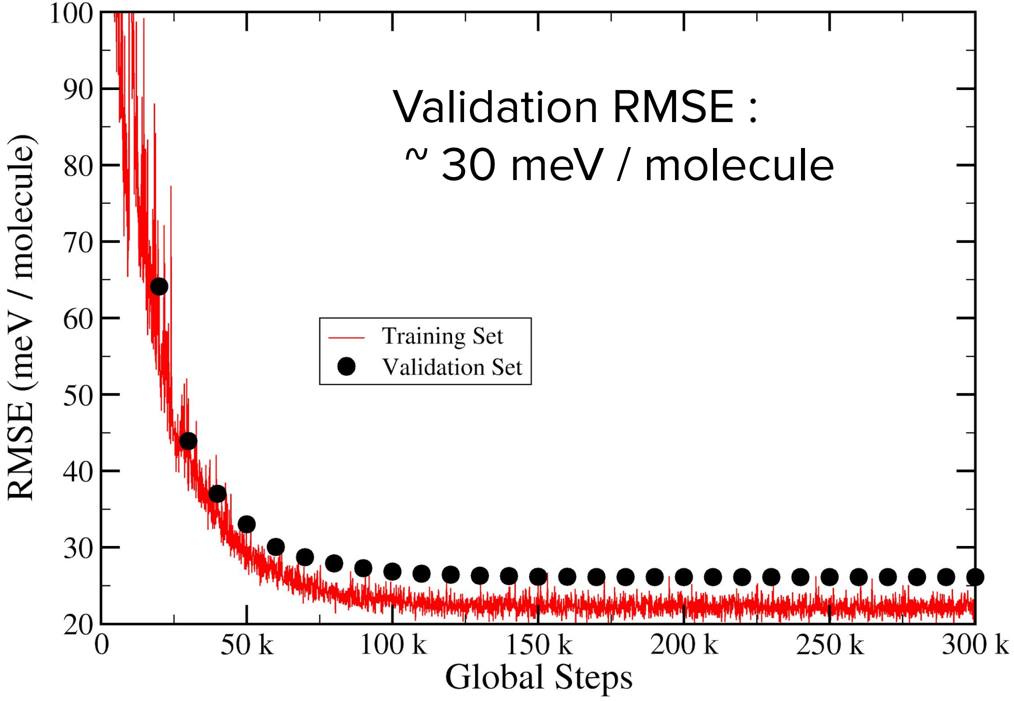


Data: **100k** small molecules ωB97x DFT (ANI* up to 4 heavy atoms)

Architecture: **[384 : 256 : 128 : 64 : 1**] x 4 (~ **500k** p.) Adam Opt., Decaying LR.

*[J.S. Smith et al. Chem Sci 8, 3192 (2017)]

Smaller networks

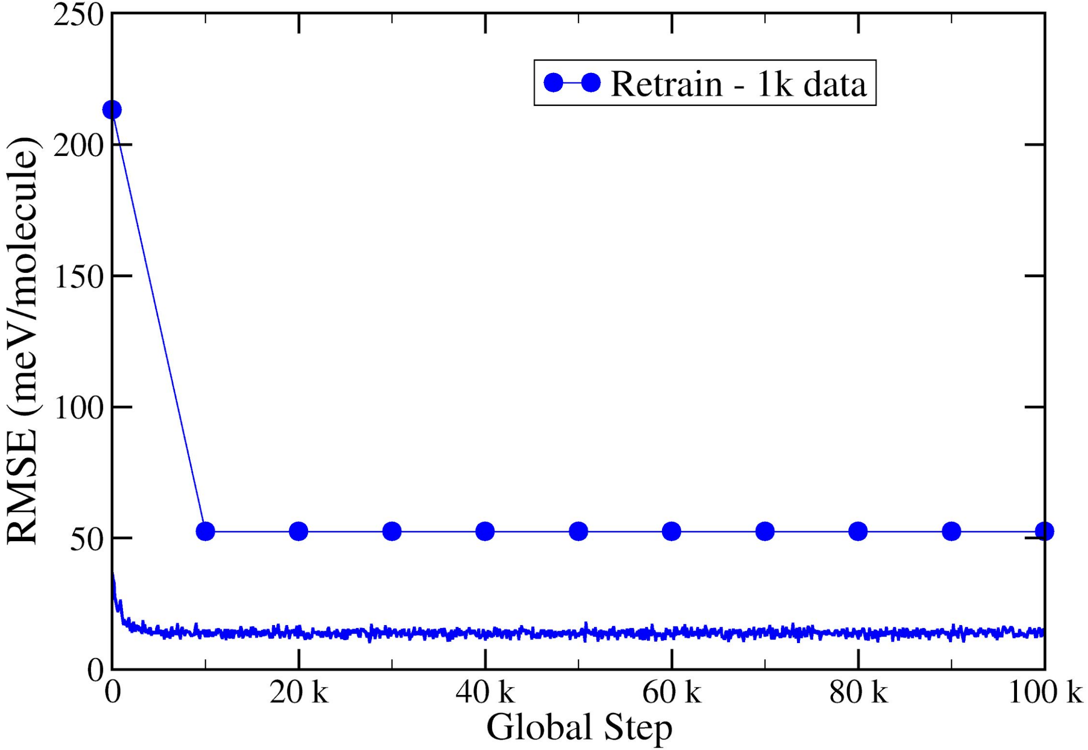


Data: **10k** small molecules ωB97x DFT

Arch: [**192** : **<u>64</u>** : **1**] x 4

(~ **50k** p.) Adam Opt., Decaying LR

Transfer learning



Changing functional (PBE)

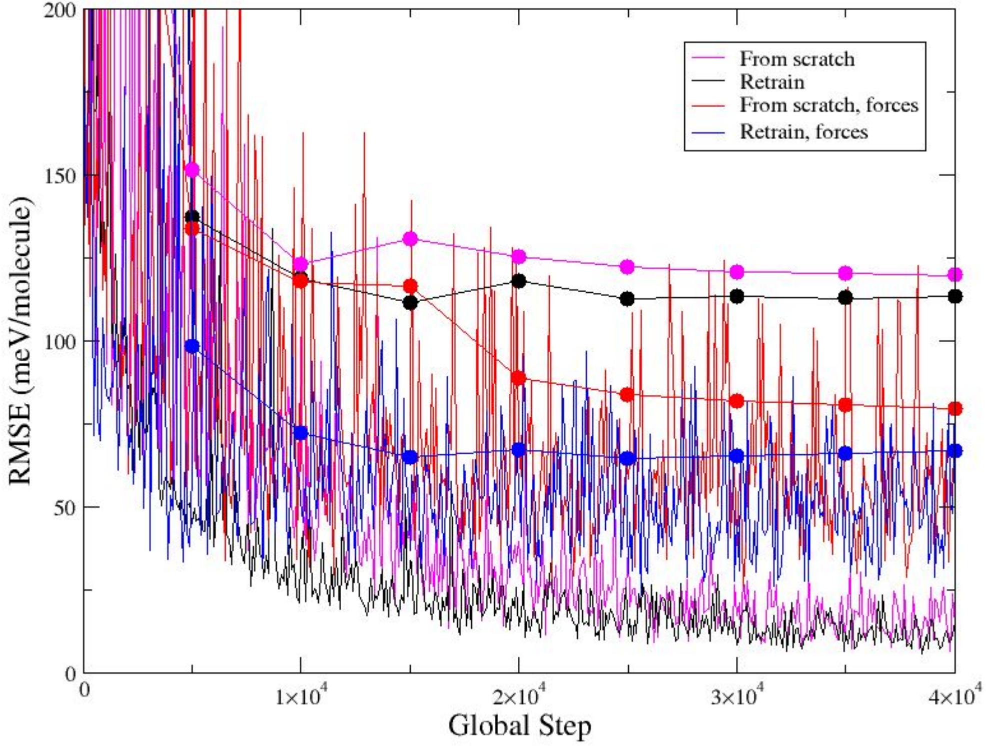
1k data, even smaller network

([**192** : **<u>32</u>** : **1**] x 4

Quickly decaying LR

Same order RMSE in few minutes

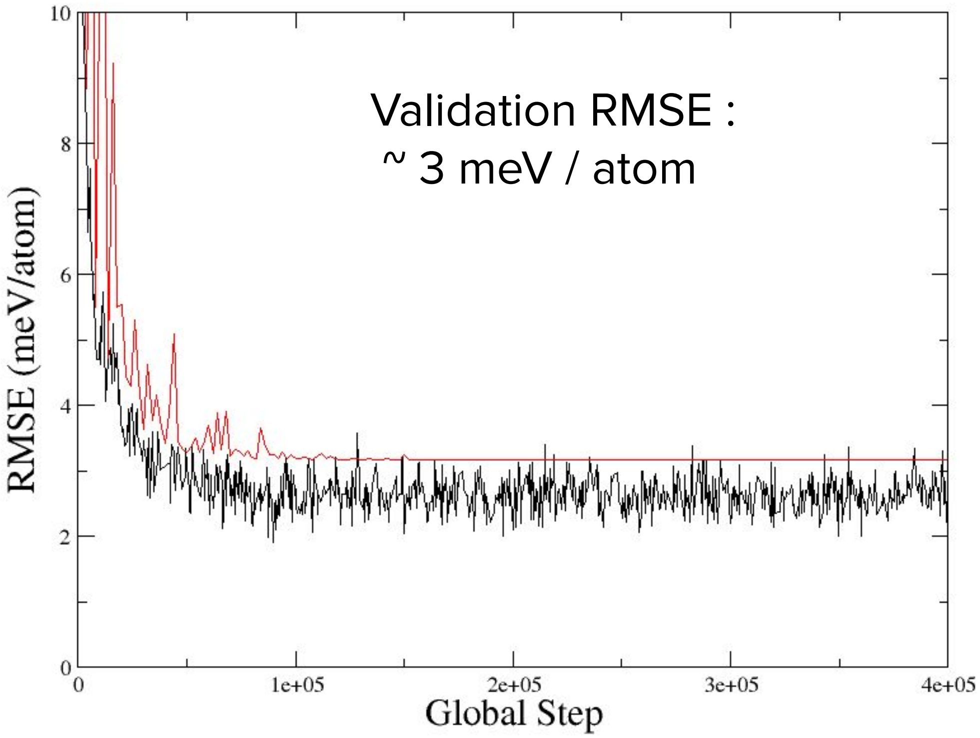
Training with forces



Glycine crystal (1k data)
Same small architecture
Adam, decaying LR

Best results from retrain with forces

Liquid Silicon



Data: ~1500 SW Si cells (~200 atoms) (without forces)
Arch: [**48 : 32 : 16 : 1**]
(~ **2.1k** p.) Adam Opt., Decaying LR

We can also train this with forces:

(G + dG/dR) ➔ Total energy
(200x48)+(200x48x3x200) ➔ 1
5.8M floats -> NN -> 1 float

Thank you!

https://gitlab.com/PANNAdevs/panna