ReservoirComputing.jl: An Efficient and Modular Library for Reservoir Computing Models

Francesco Martinuzzi[1,2,3] *, Chris Rackauckas[3,4], Anas Abdelrehim[3], Miguel Mahecha[1,2,5] and Karin Mora[2,5]

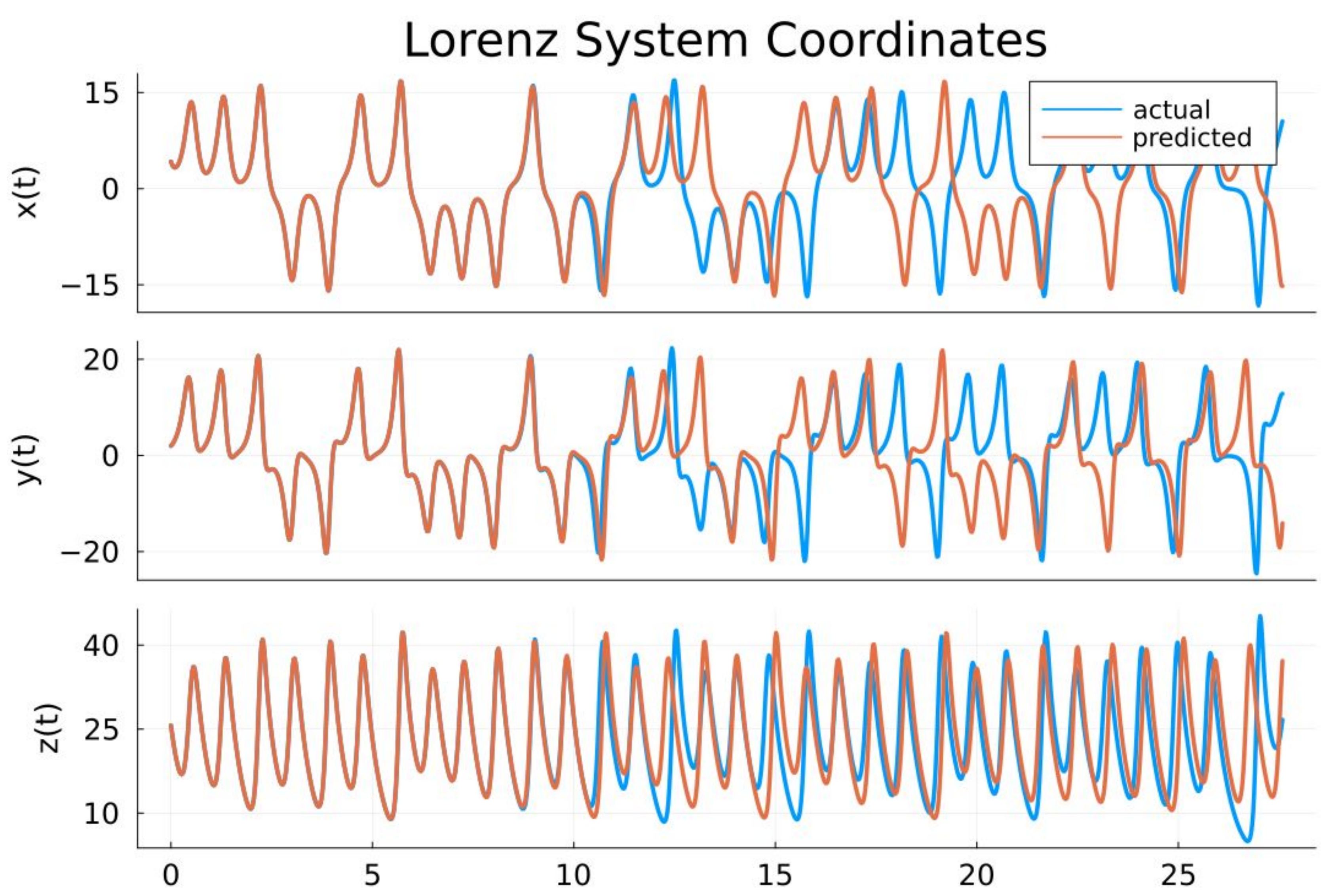
Abstract

Reservoir computing is a powerful modeling technique for sequential data. Proven to be **state of the art** in fields like time series predictions, this family of models is **lacking behind** in terms of software implementations. ReservoirComputing.jl enhances this field by being **intuitive**, **modular**, and **faster** compared to alternative tools.

What is Reservoir Computing?

Theory

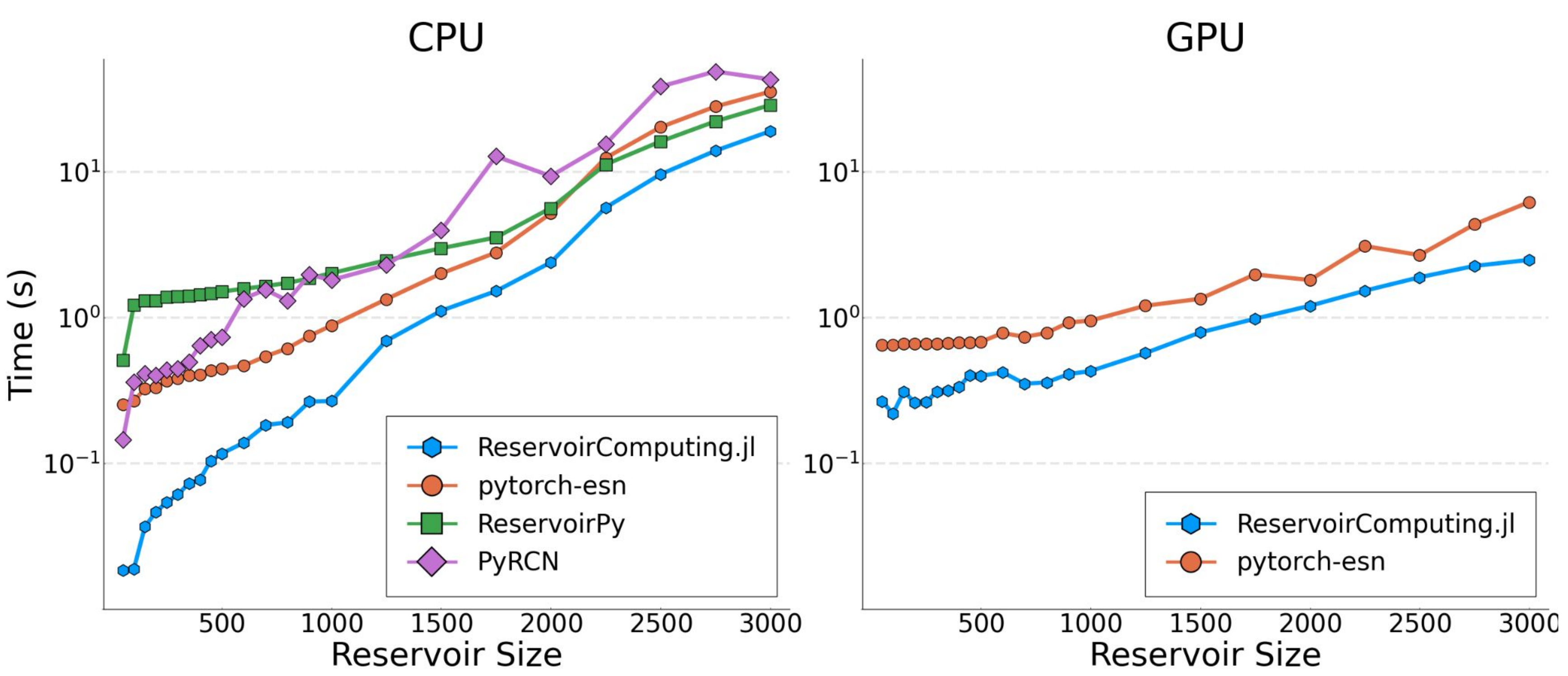
1. A nonlinear system, called reservoir, providing **complex dynamics** is chosen
2. The data is mapped onto the system to **expand** into a higher dimensional space
3. The state evolution of the reservoir is **saved** as the data is fed into it
4. The final states are used for training a readout layer against the desired output using **regression**

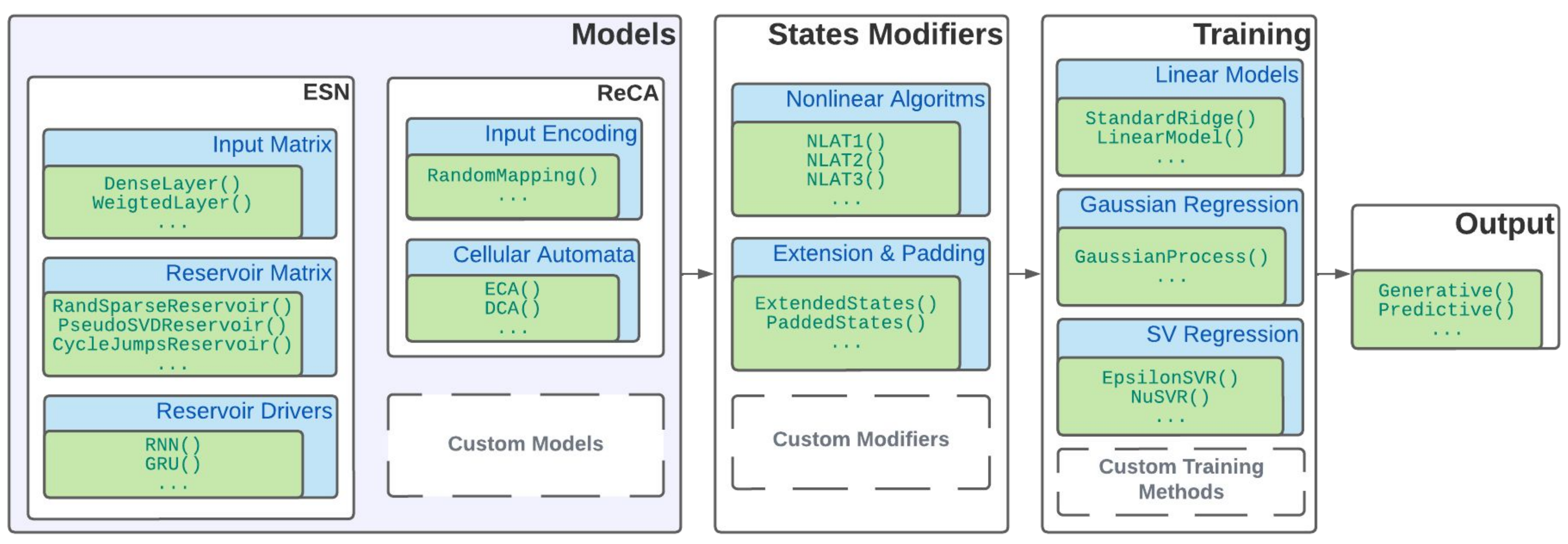


Advantages:

- **Faster** compared to standard deep learning approaches
- Computationally **less expensive**
- No exploding/vanishing gradients
- Suited for prediction of **chaotic** systems

Software Features





Speed

- From **1.5 to 14.3** times faster than the closest alternative for CPUs
- From **1.4 to 3.0** times faster for GPUs

User Friendly

- Multiple **layers** choices
- Multiple **training** choices
- Multiple **prediction** methodologies
- Easy integration of **custom components**

Clear APIs

- High level direct calls available
- More **bespoke** model construction possible

```julia
using ReservoirComputing, OrdinaryDiffEq

# Lorenz system parameters
u0 = [1.0,0.0,0.0]
tspan = (0.0,200.0)
p = [10.0,28.0,8/3]
# Define lorenz system
function lorenz(du,u,p,t)
    du[1] = p[1]*(u[2]-u[1])
    du[2] = u[1]*(p[2]-u[3]) - u[2]
    du[3] = u[1]*u[2] - p[3]*u[3]
end
# Solve and take data
prob = ODEProblem(lorenz, u0, tspan, p)
data = solve(prob, ABM54(), dt=0.02)
train_len = 5000
predict_len = 1250
# One step ahead for generative prediction
input_data = data[:, 300:300+train_len-1]
target_data = data[:, shift+1:shift+train_len]

# Define reservoir builder
res_size = 300
W = RandSparseReservoir(res_size,
                        radius=1.2,
                        sparsity=6/res_size)
# Build Echo State Network
esn = ESN(input_data;
          reservoir = W,
          input_layer = WeightedLayer(),
          nla_type = NLAT2())
# Create the Output Layer
output_layer = train(esn, target_data)
# Predict output
output = esn(Generative(1250), output_layer)
```

Focus

The library is aimed at researchers interested in modeling a wide range of **complex spatio-temporal** data sets, from mathematical models to climate data. The ease of use allows for both **quick explorations** of the algorithm and for more **nuanced constructions**, appealing to both newcomers and expert practitioners.

Jaeger, Herbert. "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note."
Pathak, Jaideep, et al. "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data."

1: Center for Scalable Data Analytics and Artificial Intelligence, Leipzig, Germany
2: Remote Sensing Centre for Earth System Research, Leipzig, Germany
3: Julia Computing, Inc
4: Massachusetts Institute of Technology
5: German Centre for Integrative Biodiversity Research (iDiv)
- **francesco.martinuzzi@uni-leipzig.de**

UNIVERSITÄT LEIPZIG

ScaDS.AI DRESDEN LEIPZIG