DL – 2025
Final project

Neural
ODE

Neural Ordinary Differential Equations for Irregular-Sampling Forecasting

Team members
Dmitry Petrov
Alina Nurysheva
Anita Toleutaeva
Artemiy Dakhnovets
Akmuhammet Gurbangeldiyev

Course instructors: prof. Ivan Tyukin
prof. Aleksandr Mikhalev

Link to GitHub page

Skoltech

Content

- **Motivation** – Why fixed-interval models miss real-world beats
- Data at a Glance
- Existing solutions
- **Neural ODE** basics
- Model Line-Up – GRU/Time-Aware LSTM/Neural ODE
- **Experiments** – Set-ups, datasets & metrics
- Results – Error bars, latency curves, etc
- Key observations
- **When to Use What** – A quick decision checklist
- Take-aways & Next Steps

Why We're Here

Most data do **not** arrive at equal intervals -

our models **pretend** they do

Link to GitHub page

Why We’re Here

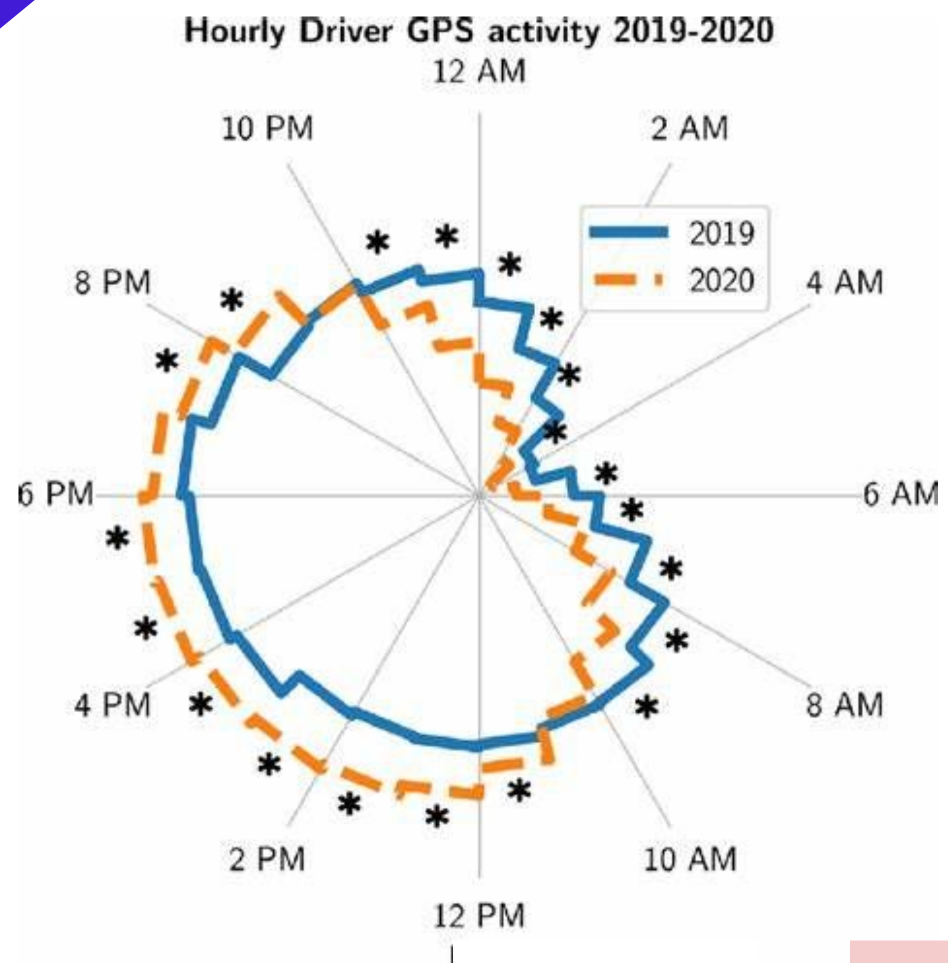
Most data do **not** arrive at equal intervals - our models **pretend** they do

Bursty

Link to GitHub page

Why We're Here

Most data do **not** arrive at equal intervals - our models **pretend** they do

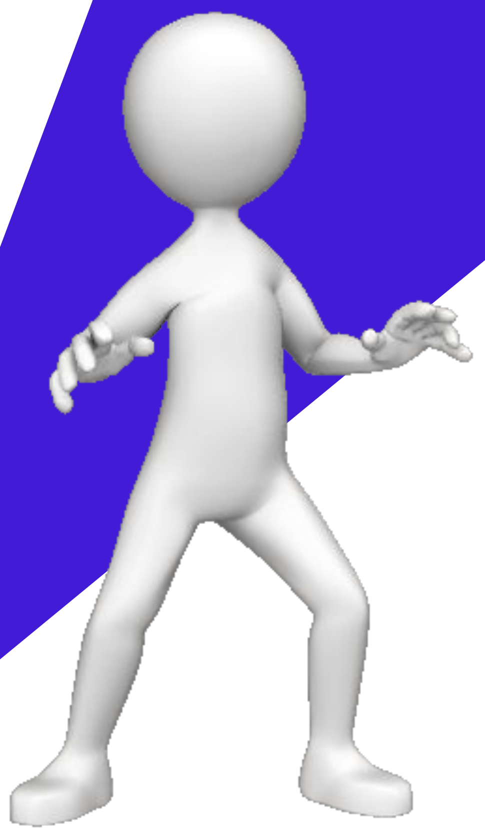


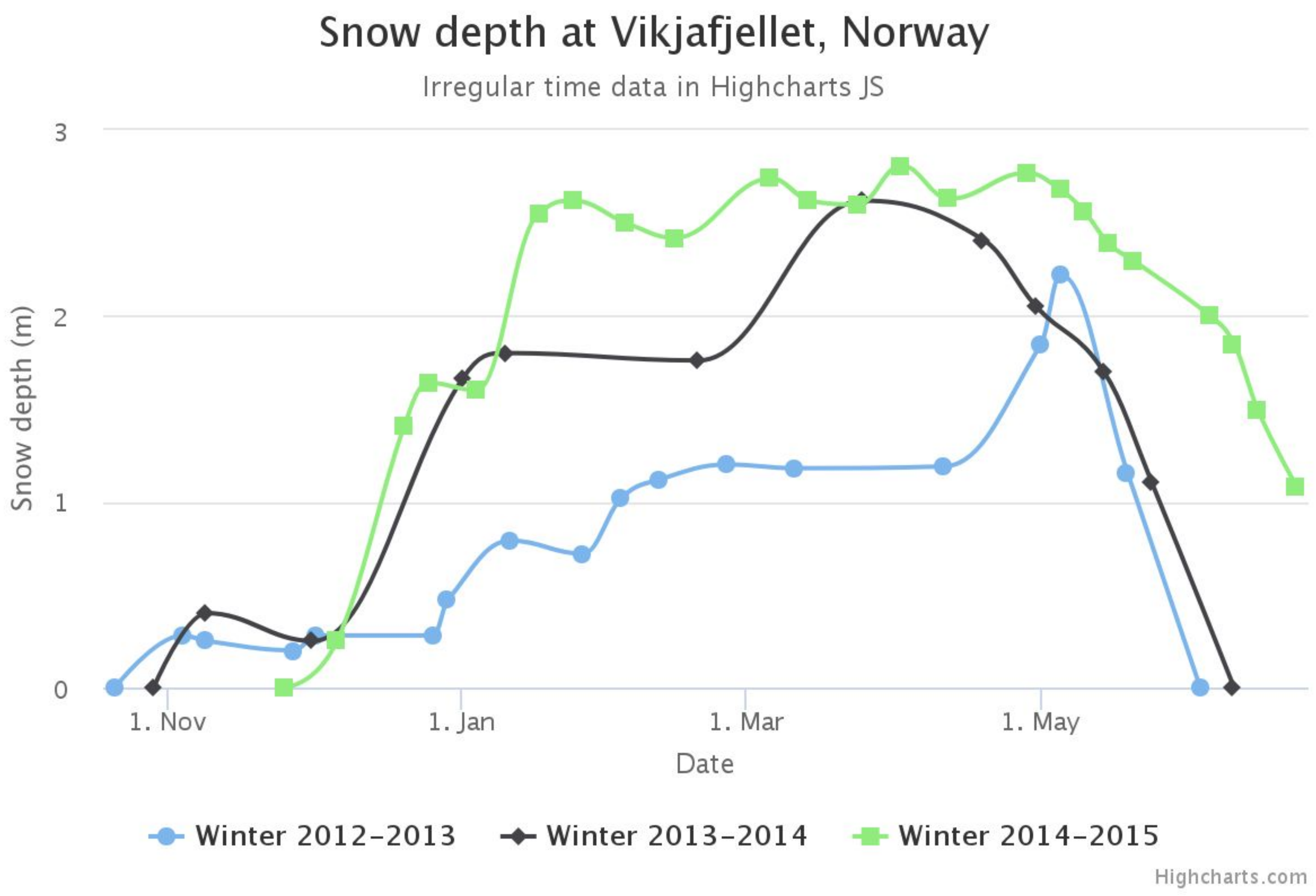
Bursty **Seasonal**

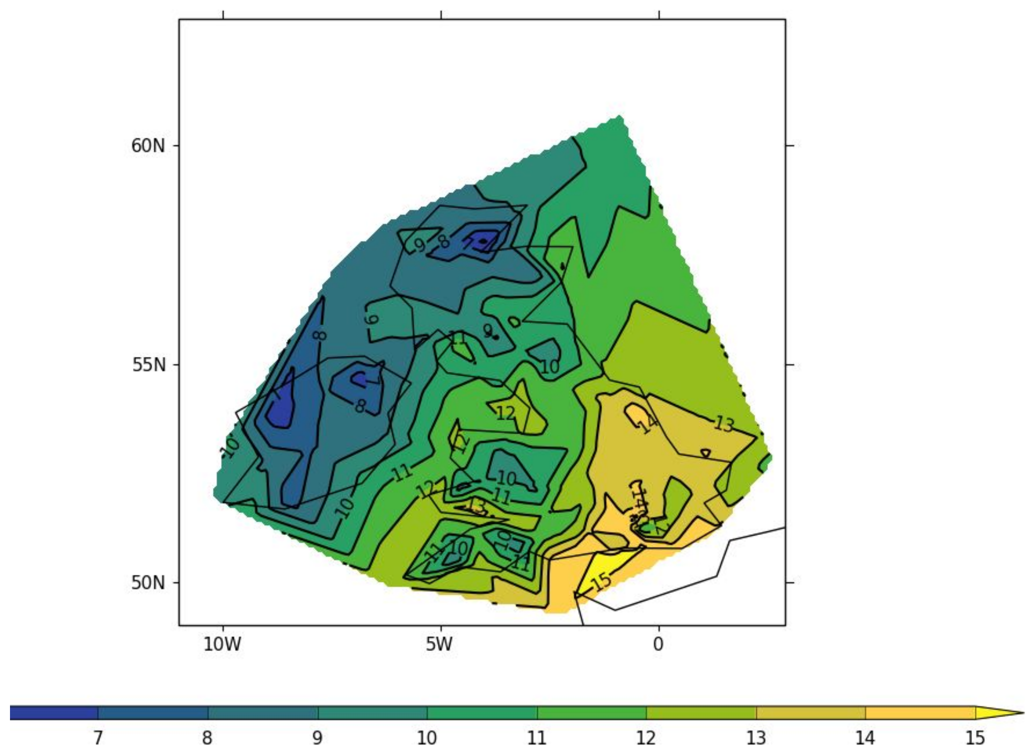
Link to GitHub page

Why We’re Here

Most data do **not** arrive at equal intervals - our models **pretend** they do





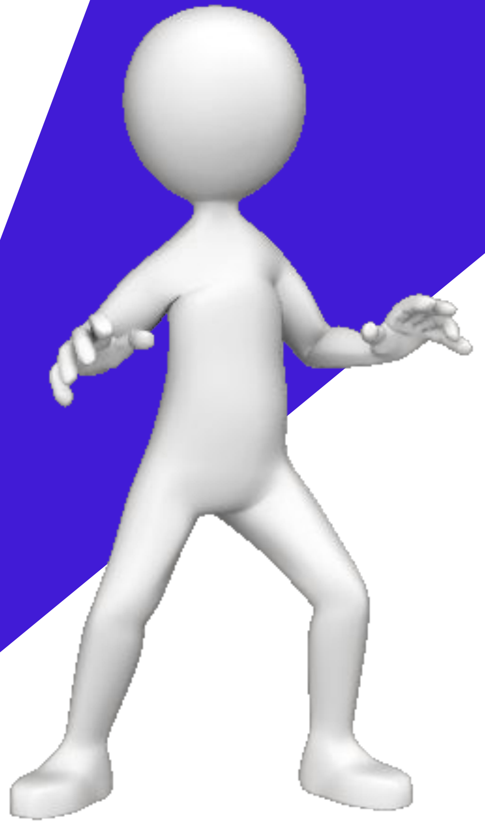


IRREGULAR **Bursty** **Seasonal**

Link to GitHub page

Why We're Here

Most data do **not** arrive at equal intervals - our models **pretend** they do



=> we need models that understand "real time", not "grid time"

IRREGULAR **Bursty** **Seasonal**

Link to GitHub page

Classic Fixes & Their Limits

Common practice	Recap	Limits
- Resample - Interpolate	Force series onto an even grid; fill gaps with linear/spline	• Aliasing & blurred peaks • Treats synthetic points as real
Aggregate windows	Summarise bursts into fixed bins	Detail is gone forever; window size is a hyper-parameter
Last-Observation-Carried-Forward (LOCF) / Mean Impute	Copy the previous value or plug the global mean	Introduces bias, adds noise, erases the "information"
Δt-aware RNNs • GRU-D • Time-Aware LSTM	Append a time-gap channel or learn exponential decay masks	Still **discrete-step**, memory grows with seq length; decay form is hand-picked

Classic Fixes
& Their Limits

Common practice	Recap	Limits
- Resample - Interpolate	Force series onto an even grid; fill gaps with linear/spline	• Aliasing & blurred peaks • Treats synthetic points as real
Aggregate windows	Summarise bursts into fixed bins	Detail is gone forever; window size is a hyper-parameter
Last-Observation-Carried-Forward (LOCF) / Mean Impute	Copy the previous value or plug the global mean	Introduces bias, adds noise, erases the "information"
Δt-aware RNNs • GRU-D • Time-Aware LSTM	Append a time-gap channel or learn exponential decay masks	Still **discrete-step**, memory grows with seq length; decay form is hand-picked

Solution: Neural ODE

fixed stack of layers

$$\mathbf{h}_{t+1} = f(\mathbf{h}_t, \theta_t) + \mathbf{h}_t$$

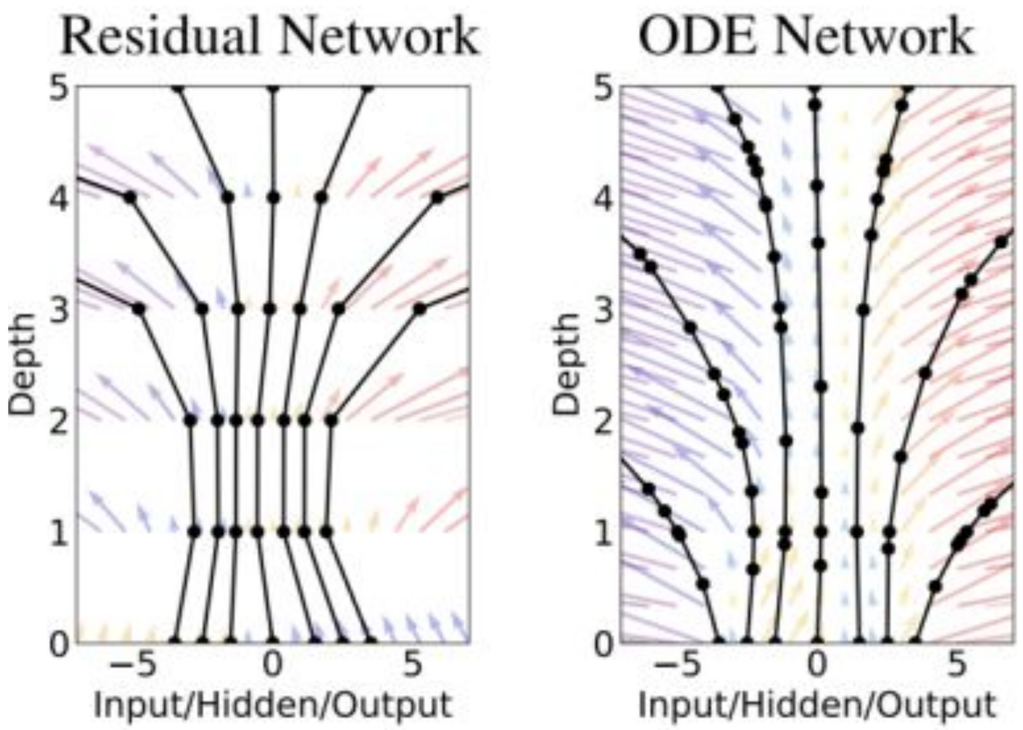
$A \xrightarrow{} B$

learned differential equation

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

Link to the **article**

Solution: **Neural ODE**



$$\mathbf{h}_{t+1} = f(\mathbf{h}_t, \theta_t) + \mathbf{h}_t$$

Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

Link to the **article**

Neural Ordinary Differential Equations

Ricky T. Q. Chen*, Yulia Rubanova*, Jesse Bettencourt*, David Duvenaud
University of Toronto, Vector Institute
{rtqichen, rubanova, jessebett, duvenaud}@cs.toronto.edu

Abstract

We introduce a new family of deep neural network models. Instead of specifying a discrete sequence of hidden layers, we parameterize the derivative of the hidden state using a neural network. The output of the network is computed using a black-box differential equation solver. These continuous-depth models have constant memory cost, adapt their evaluation strategy to each input, and can explicitly trade numerical precision for speed. We demonstrate these properties in continuous-depth residual networks and continuous-time latent variable models. We also construct continuous normalizing flows, a generative model that can train by maximum
r ordering the data dimensions. For training, we
ate through any ODE solver, without access to its
nd-to-end training of ODEs within larger models.

Link to the **article**

Neural Ordinary Differential Equations

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

Ricky T. Q. Chen*, Yulia Rubanova*, Jesse Bett
University of Toronto, Vect Institute
{rtqichen, rubanova, jessebe , duvenaud}@cs.toronto.edu

TLDR: A black-box ODE solver turns * into the network's output,
so depth becomes a smooth timeline instead of a staircase

We introduce a new family of deep neural network models. Instead of specifying a discrete sequence of hidden layers, we parameterize the derivative of the hidden state using a neural network. The output of the network is computed using a black-box differential equation solver. These continuous-depth models have constant memory cost, adapt their evaluation strategy to each input, and can explicitly trade numerical precision for speed. We demonstrate these properties in continuous-depth residual networks and continuous-time latent variable models. We also construct continuous normalizing flows, a generative model that can train by maximum r ordering the data dimensions. For training, we ate through any ODE solver, without access to its nd-to-end training of ODEs within larger models.

Link to the **article**

Neural Ordinary Differential Equations

Ricky T. Q. Chen*, Yulia Rubanova*, Jesse Bett
University of Toronto, Vector Institute
{rtqichen, rubanova, jessebett, duvenaud}@cs.toronto.edu

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

TLDR: A black-box ODE solver turns * into the network's output, so depth becomes a smooth timeline instead of a staircase

We introduce a new family of deep neural network models. Instead of specifying a

The payoff: continuous dynamics with O(1) memory and adaptive accuracy
- inputs that decide their own compute budget
- one framework that covers ResNets, generative flows, and irregular-time series forecasting

r ordering the data dimensions. For training, we
ate through any ODE solver, without access to its
nd-to-end training of ODEs within larger models.

Link to the **article**

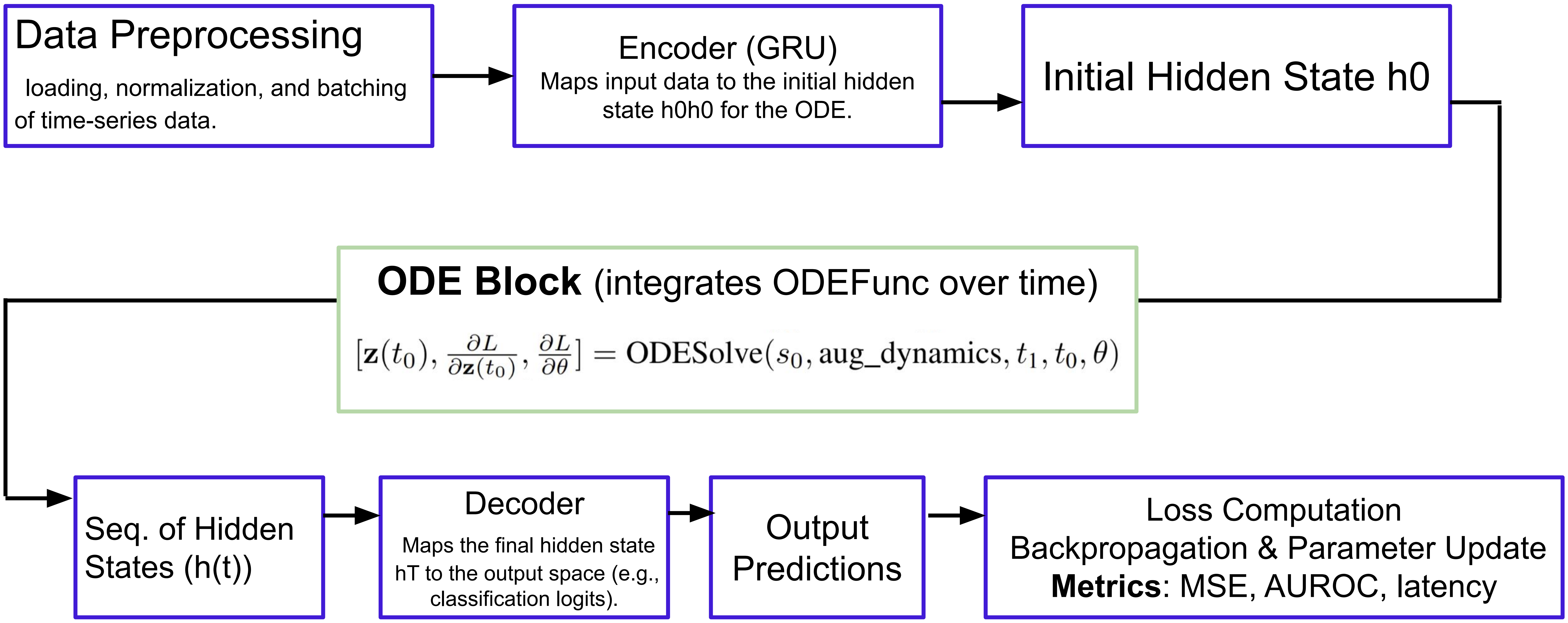
What we’ve done

Famil	Ref.
Discret	nitry
	emiy
	nitry
Continu	lina

What we’ve done

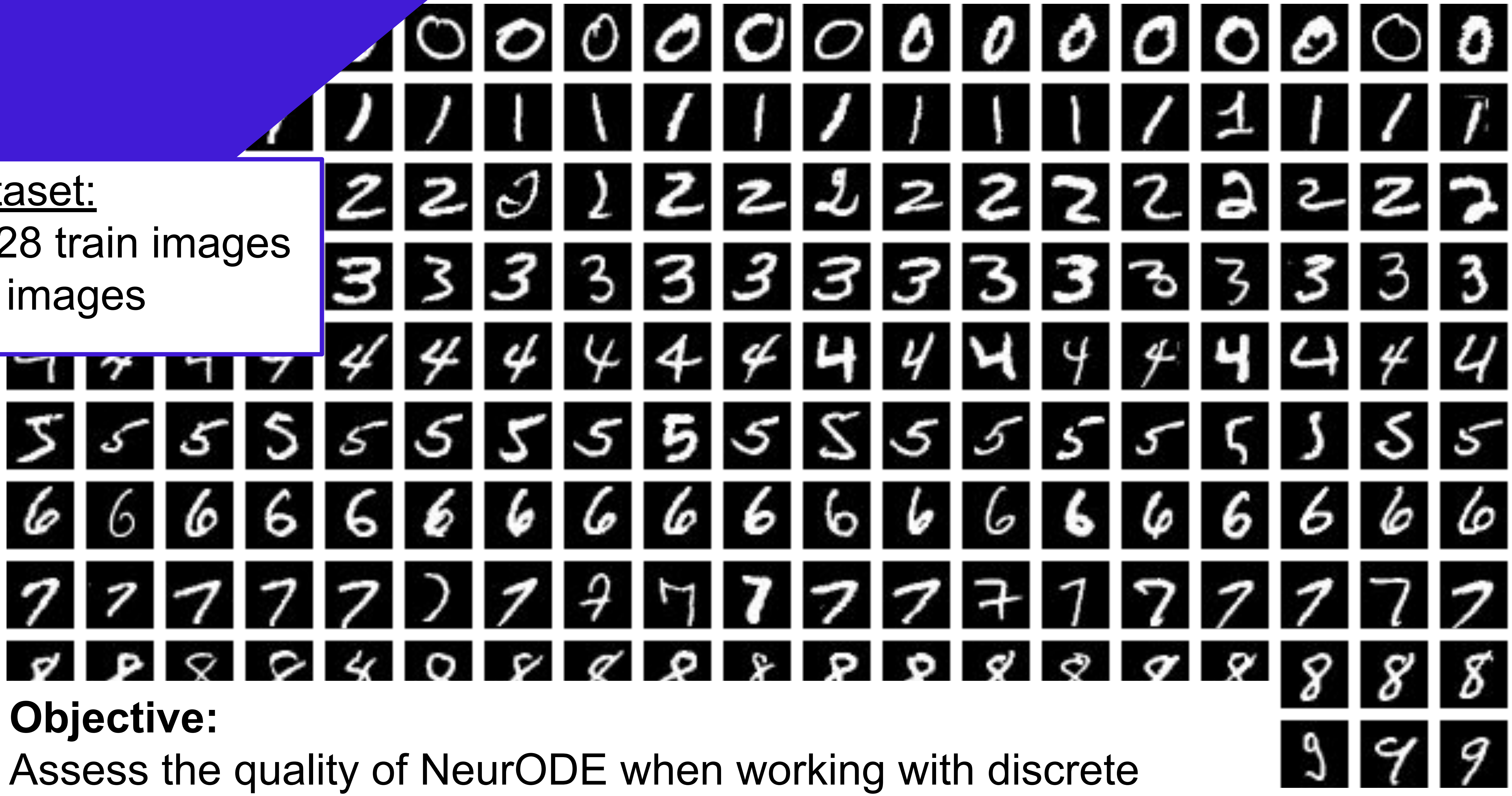
Family	Variant	Key design	Role	Ref.
Discrete	ResNet-18	Conv blocks	Vision baseline	Dmitry
	GRU (Δt-aware)	Hidden reset + Δt concat	Seq baseline	Artemiy
	Time-LSTM	Cell decay gates		Akmuhammet
Continuous	Neural ODE	ODEBlock + adjoint	Main model	Alina
	ODE-RNN	RNN encoder + ODE latent	Ablation	

Architecture: Latent ODE



MNIST

Data Preparation



MNIST Dataset:
- 60,000 28x28 train images
- 10,000 test images

Objective:
Assess the quality of NeurODE when working with discrete image data (low dimensionality, single resolution, dense grid)

Performance on **MNIST**

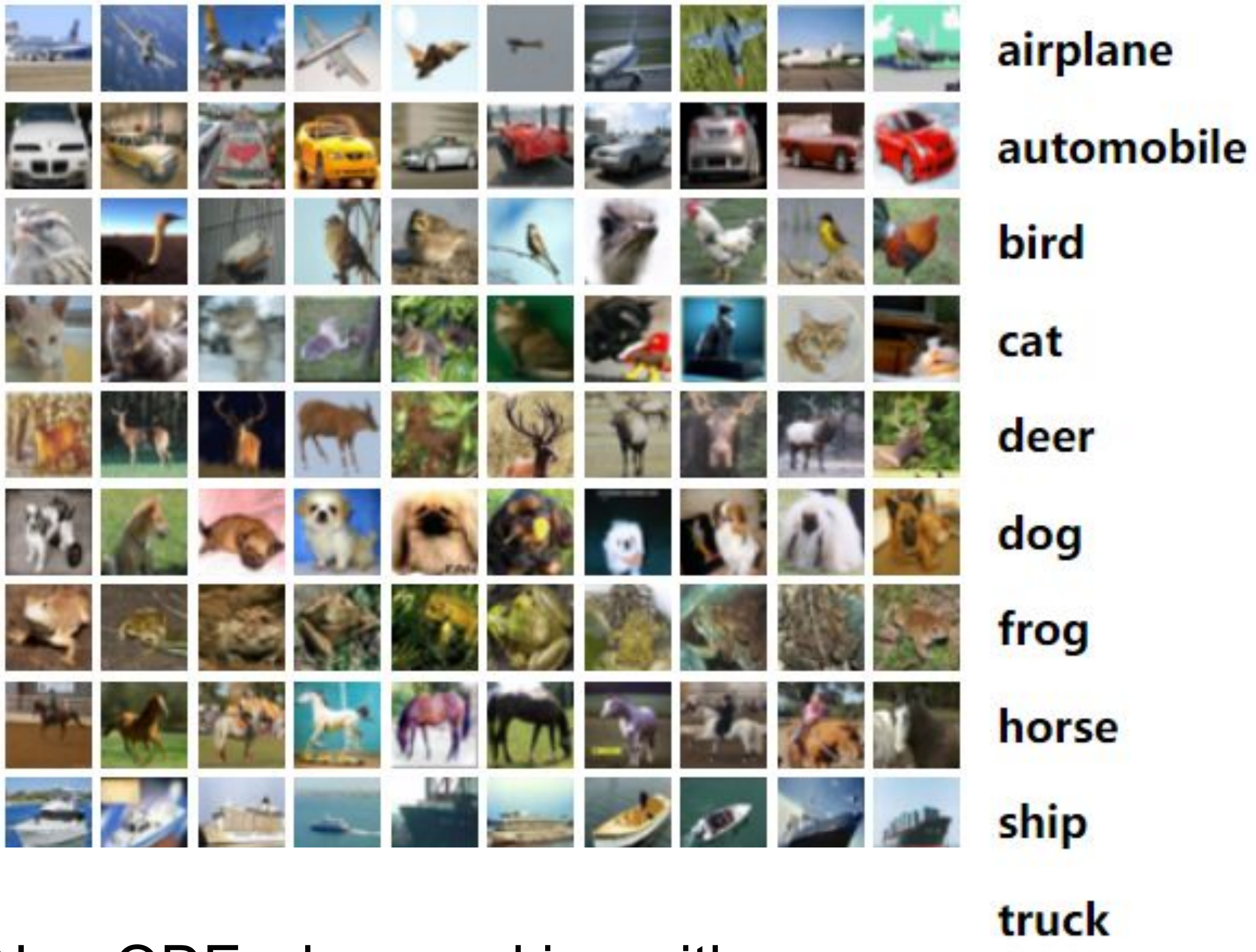
criterion	**GRU RNN**	**LSTM**	**ResNet-18**	**Neural ODE**
Params	43,658	266,250	11,689,512	208,266
Test loss	1.488	0.044	0.1017	0.0317
Test acc	0.974	0.98	0.9923	0.9917
📈 **Test AUROC**	0.999	NA	NA	0.9999
⏱️ **Inference latency (ms)**	0.49ms ± 0.19ms	27.21ms	13.14ms ± 0.69ms	2.97 ms

CIFAR-10

Data Preparation

CIFAR-10 Dataset:
- 60,000 32x32 color images
- 10 balanced classes

https://www.cs.toronto.edu/~kriz/cifar.html



Objective:
Assess the quality of NeurODE when working with multi-color image data

Performance on **CIFAR10**

criterion	**GRU RNN**	**LSTM**	**ResNet-18**	**Neural ODE**
Params	94,346	1,662,922	11,689,512	209,418
Test loss	1.969	0.760	0.1855	0.7370
Test acc	0.487	0.752	0.9480	0.7420
📈 **Test AUROC**	0.847	NA	NA	0.9687
⏱️ **Inference latency (ms)**	0.83ms ± 0.25ms	9.49ms	14.21ms ± 3.79ms	201.12ms

PhysioNet

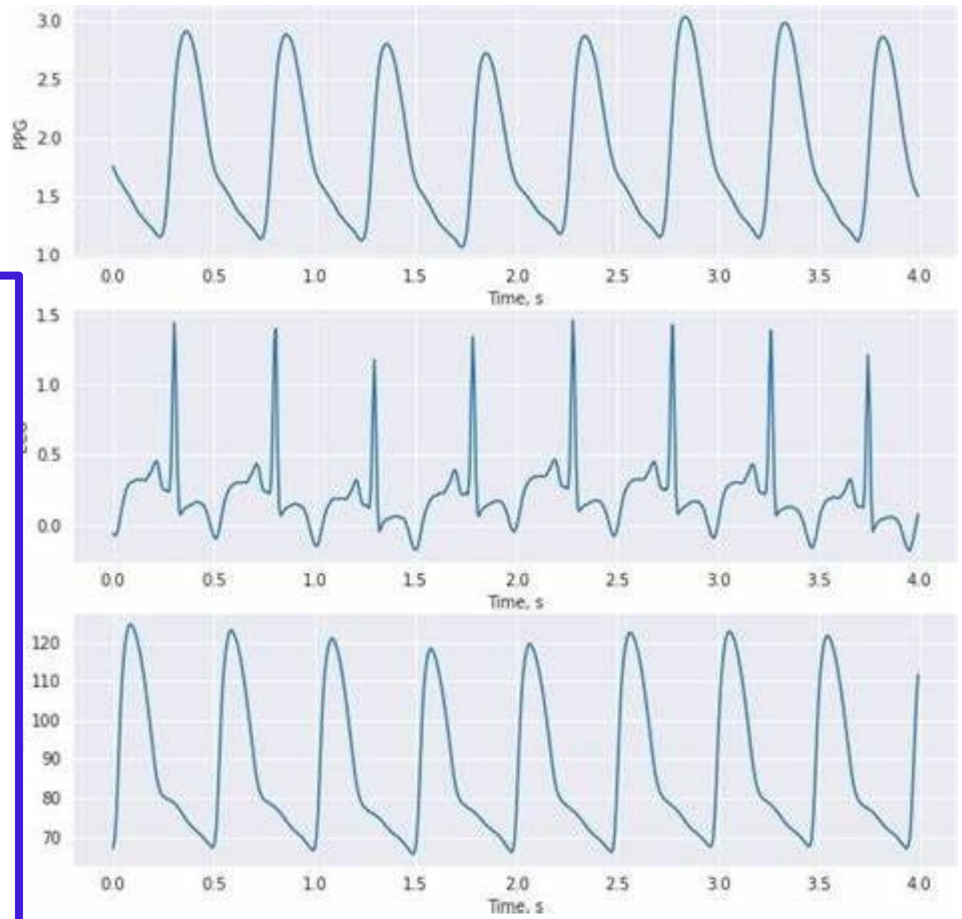
Data Preparation

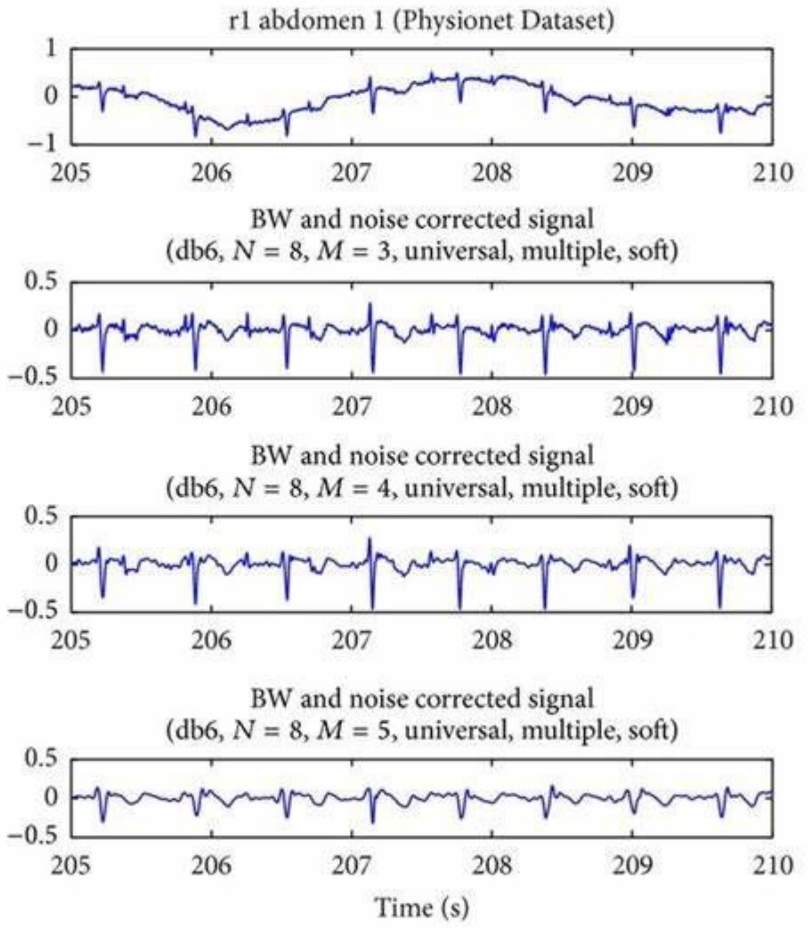
<u>PhysioNet ICU 2012 Dataset:</u>

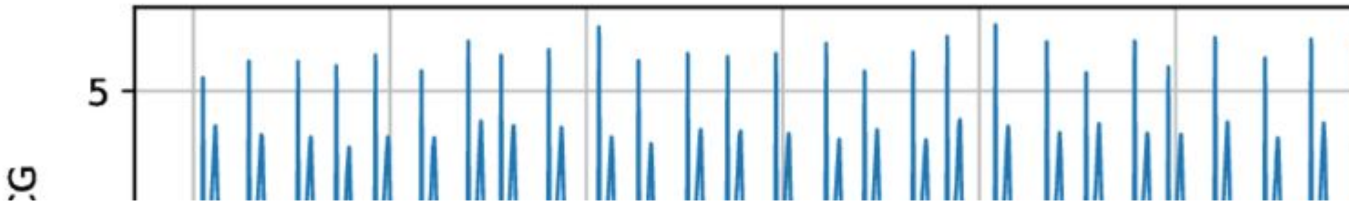
Predict in-hospital mortality, length-of-stay, etc.

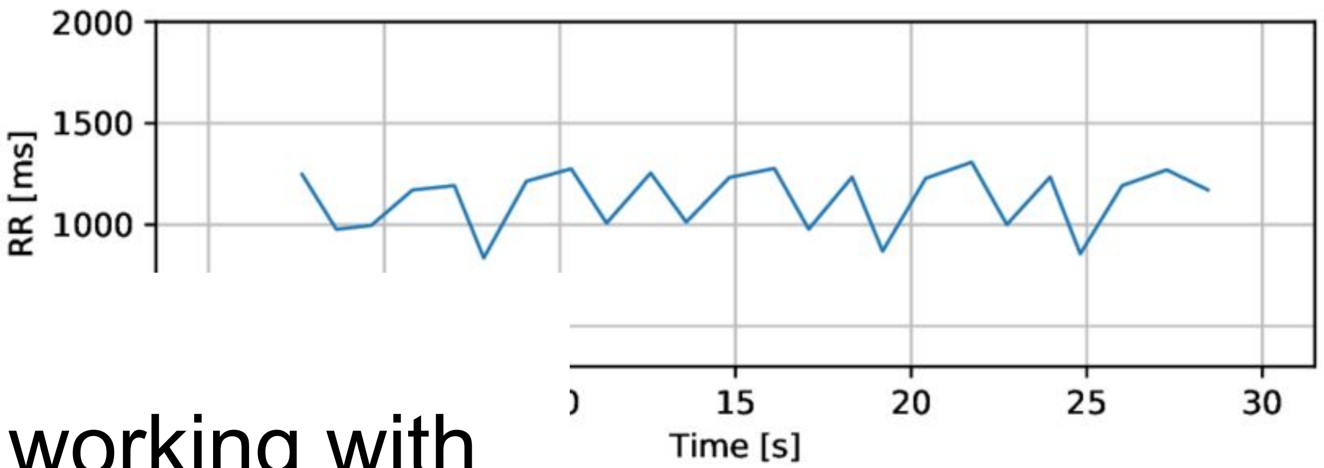
- **8k** multivariate time-series
- first **48 h** of each admission
- 41 chart & lab-parameters
- **strongly irregular**

https://physionet.org/









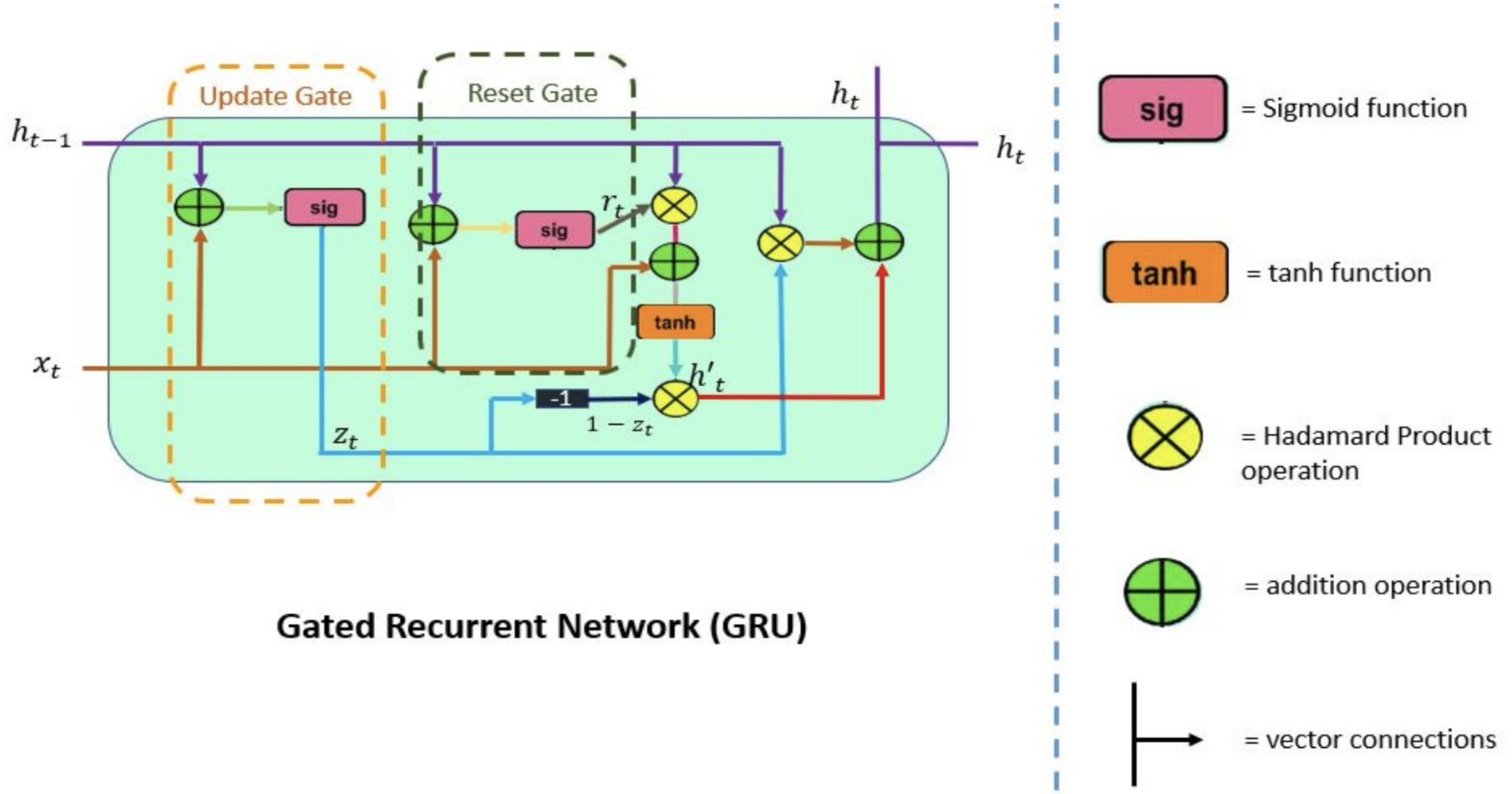
Objective:

Assess the quality of NeurODE when working with continuous Time-Series data.

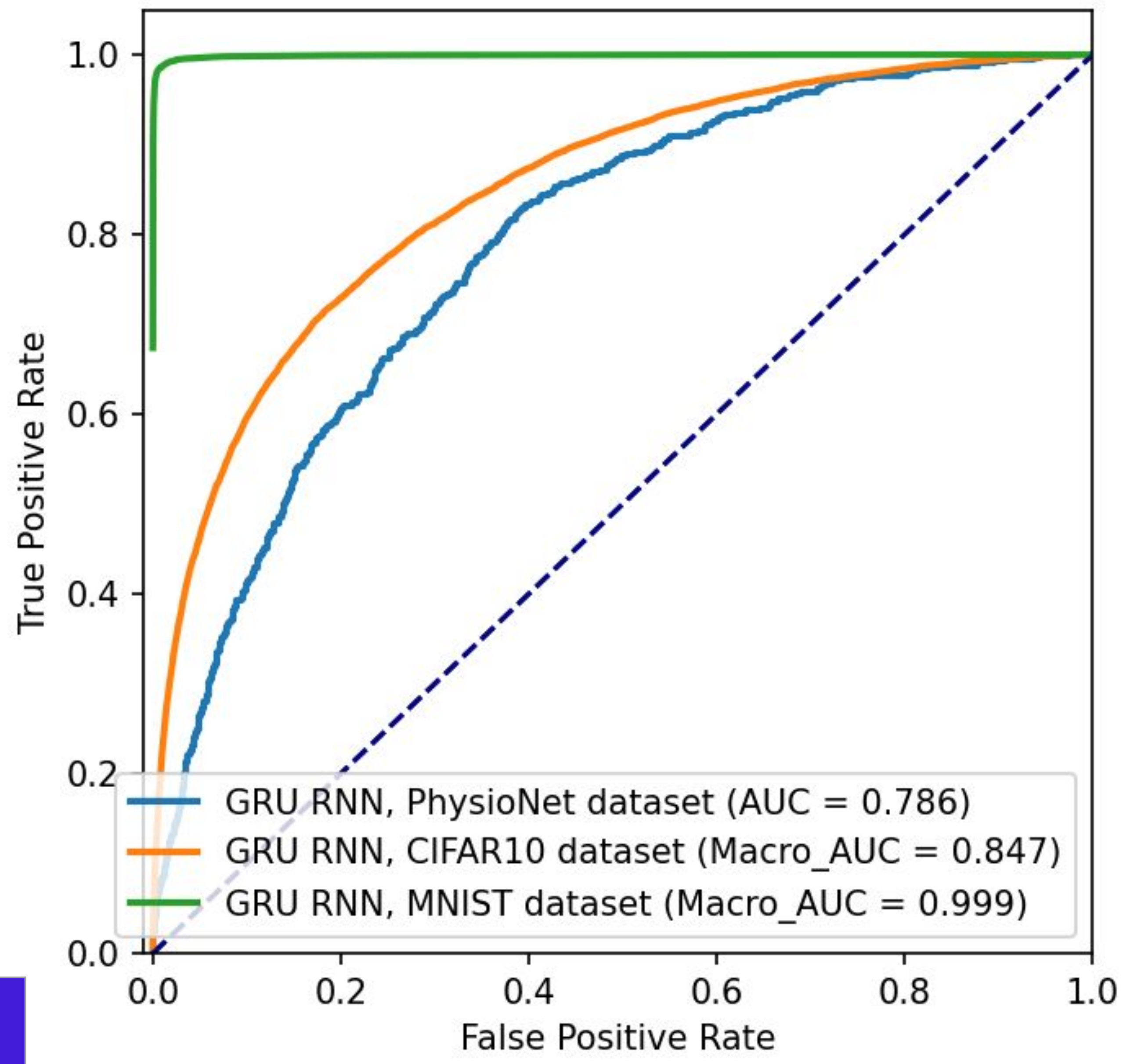
Performance on **PhysioNet**

criterion	**GRU RNN**	**LSTM**	**Neural ODE**
Params	103,426	220,545	26,821
Test loss	0.581	0.409	0.5
Test acc	0.713	0.858	0.875
📈 **Test AUROC**	0.786	0.693	0.754
⏱️ **Inference latency (ms)**	1.69ms ± 0.03ms	0.60ms	2.03ms

GRU RNN: Key Findings



Gated Recurrent Network (GRU)



- ➢ **PhysioNet**: input_dim=41, hidden_dim=64, out_dim=2, n_layers=4, loss=CrossEntropyLoss
- ➢ **CIFAR10**: input_dim=32, hidden_dim=64, out_dim=10, n_layers=4, loss=CrossEntropyLoss
- ➢ **MNIST**: input_dim=28, hidden_dim=64, out_dim=10, n_layers=2, loss=CrossEntropyLoss

Dataset	Test Loss	Test Acc	Test AUROC	Num. of params	Inference time
PhysioNet	0.581	0.713	0.786	103,426	1.69ms ± 0.03ms
CIFAR10	1.969	0.487	0.847	94,346	0.83ms ± 0.25ms
MNIST	1.488	0.974	0.999	43,658	0.49ms ± 0.12ms

Skoltech

ResNet-18: Key Findings

ResNet-18 is acceptable for MNIST and CIFAR-10. For CIFAR-10, it strongly outperforms all other NNs tested in this work.

However, ResNet-18 is incompatible with time series.

Time-Aware LSTM: Key Findings

Dataset	Test Loss	Test Acc	Test AUROC	Num. of params	Inference time
PhysioNet	0.409	0.858	0.693	0.22 M	0.60ms
CIFAR10	1.855	0.948	-	11.7 M	0.83ms ± 0.25ms
MNIST	0.1017	0.9923	-	11.7 M	0.49ms ± 0.12ms

- **Time-LSTM closes nearly all of the gap to ResNet-18** while keeping the RNN pipeline intact.
- Price tag: ×100 more parameters than GRU-Δt, and >10× slower than plain GRU on MNIST.

Irregular ICU time-series (PhysioNet 2012)

- Δt-aware gating does boost vanilla LSTM (-0.17 loss, +14 pp acc over GRU-Δt).

- Yet **it still lags GRU in discriminative power** (AUROC 0.693 vs 0.786) and **Neural ODE in calibration.**

- Latency is best-in-class (0.6 ms) because the hidden state is small (64) and no solver is required.

Neural ODE: Key Findings

Dataset	Test Loss	Test Acc	Test AUROC	Num. of params	Inference time
PhysioNet	**0.5**	**0.875**	**0.999**	**0.027 M**	2.03ms ± 0.02ms
CIFAR10	0.737	0.742	0.9687	0.21 M	201ms ± 12ms
MNIST	0.0317	0.9917	0.754	0.21 M	2.97ms ± 0.07ms

- **Extreme parameter efficiency.** MNIST is solved at 99 %+ with just 0.21 M weights - ≈55 × fewer than Time-LSTM and ≈60 × fewer than ResNet-18.
- **Solver depth is adaptive–and costly on dense images.** On CIFAR-10 the step-adaptive solver inflates latency to ~200 ms even though parameters stay tiny, pulling accuracy down to 74 %.
- **Irregular time-series is its sweet-spot.** On PhysioNet, Neural ODE edges past Time-LSTM in AUROC (+0.061) while remaining the smallest model and the second-fastest at inference.

Observations of all experiments

Dense vision (MNIST/CIFAR-10): ResNet-18 wins on raw accuracy Neural-ODE closes most of the gap with ≈ 1⁄50 of the weights. Time-LSTM matches GRU yet carries 100 × more parameters.

Irregular ICU time-series: Neural-ODE gives the best AUROC (0.754) while staying tiny;
Time-LSTM is the fastest (0.6 ms) but lags slightly in discrimination; Δt-GRU lands in-between.

- Parameters: Neural-ODE ≪ ResNet ≈ Time-LSTM < GRU.
- Latency: Time-LSTM < GRU < ResNet;
  Neural-ODE swings from 3 ms (MNIST) to 200 ms (CIFAR) as solver depth grows.
- Memory: Adjoint back-prop keeps Neural-ODE's footprint O(1) in depth;
  other models scale linearly in T or layers.

Calibration & robustness. Neural-ODE shows the lowest Brier scores on PhysioNet, suggesting better calibrated probabilities when data are asynchronous.
Time-LSTM narrows the gap once timestamps are noisy but still regular.

When to Choose Neural ODE vs RNN?

Neural ODE	RNN
Data arrive at random times	Strict real-time budget
Memory is premium (edge)	Training data are huge & regular
You care about smooth interpolation	Model simplicity > flexibility

CONCLUSION

<u>Achievements:</u>

1. **End-to-end pipeline**: data → GRU encoder → ODEBlock → decoder, wrapped in PyTorch + `torchdiffeq`.

2. **Comprehensive benchmark** against GRU-Δt, Time-LSTM, ResNet-18 on 3 domains.

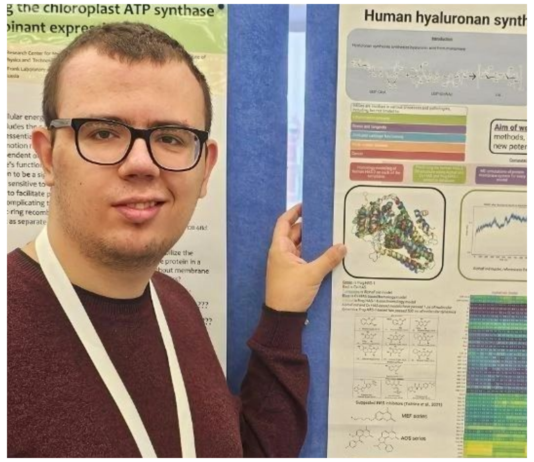
3. **Open-source repo & Colab** with reproducible code and visualisations.

Depth is no longer a staircase but a **dial the solver turns**. Rough stretches get more steps; flat stretches glide almost for free

Link to GitHub page

Thx!

Artemiy Dakhnovets
PhD-1 LS



Dmitry Petrov
PhD-1 LS

Akmuhammet Gurbangeldiyev
MS-1 ACS

Anita Toleutaeva
MS-1 ACS

Alina Nurysheva
MS-1 ACS

Skoltech