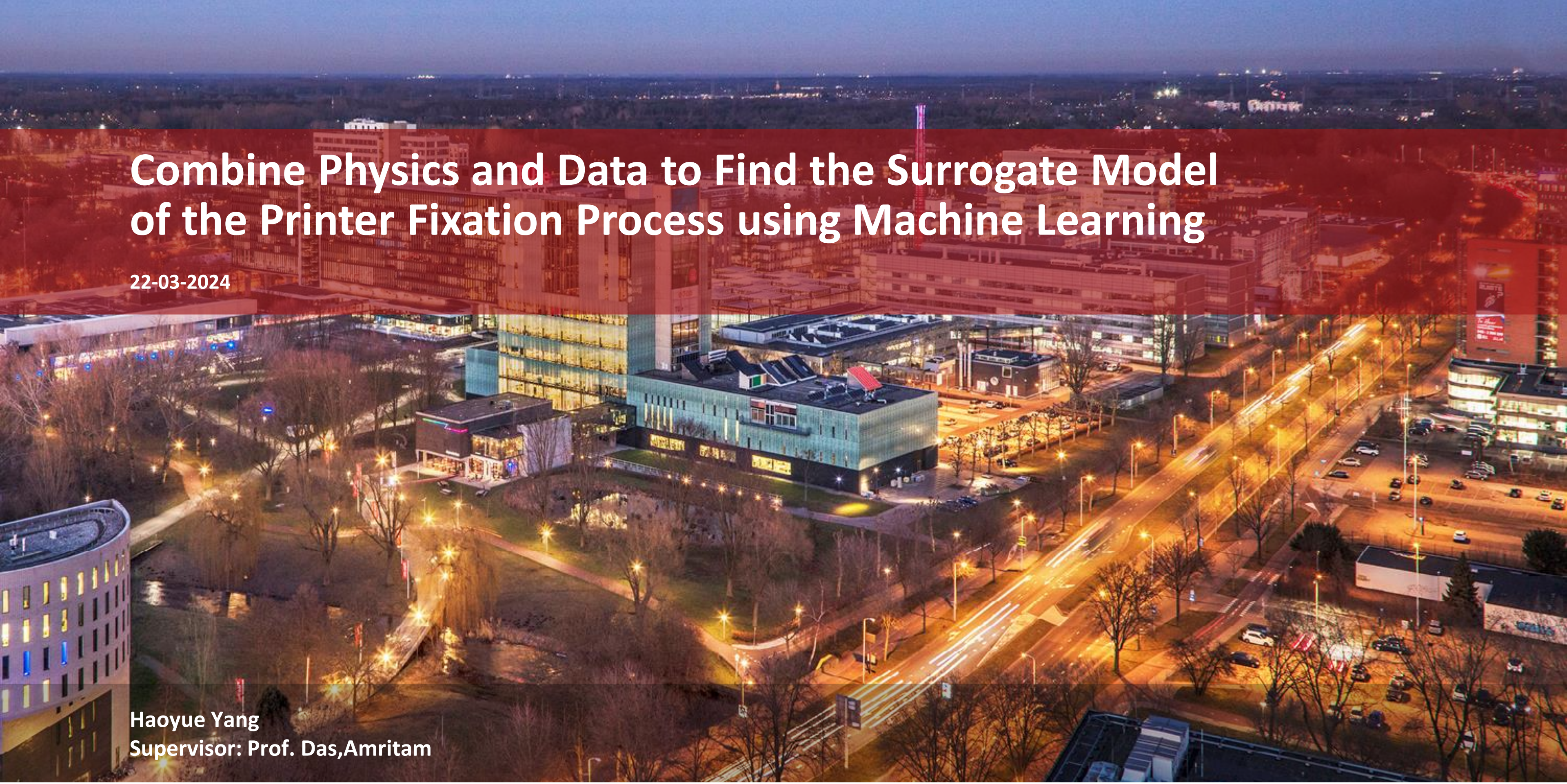


Combine Physics and Data to Find the Surrogate Model of the Printer Fixation Process using Machine Learning

22-03-2024

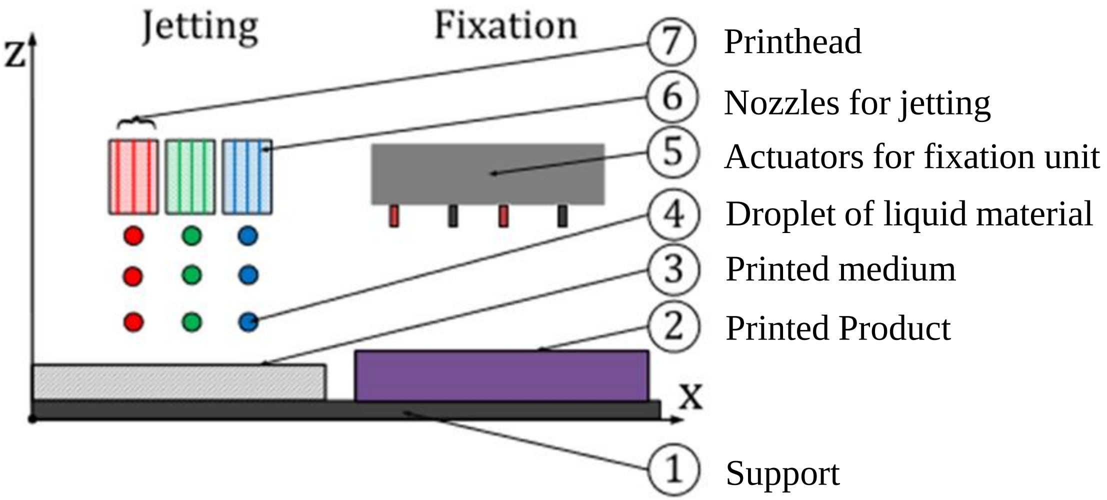
Haoyue Yang
Supervisor: Prof. Das,Amritam

Department, Sub department or Capacity Group

EINDHOVEN UNIVERSITY OF TECHNOLOGY

- Introduction
    - Inkjet Printing and Fixation process
    - Objective of the research
    - Governing Physics of the fixation model
- PINN(Physics Informed Neural Network)
    - Data-driven optimization with PINN
    - Parameter estimation problem with PINN
- Result
    - Dataset introduction
    - Result of PINN with different hyper-parameters
    - Result of PINN estimation on different dataset
- Future Work

Inkjet Printing

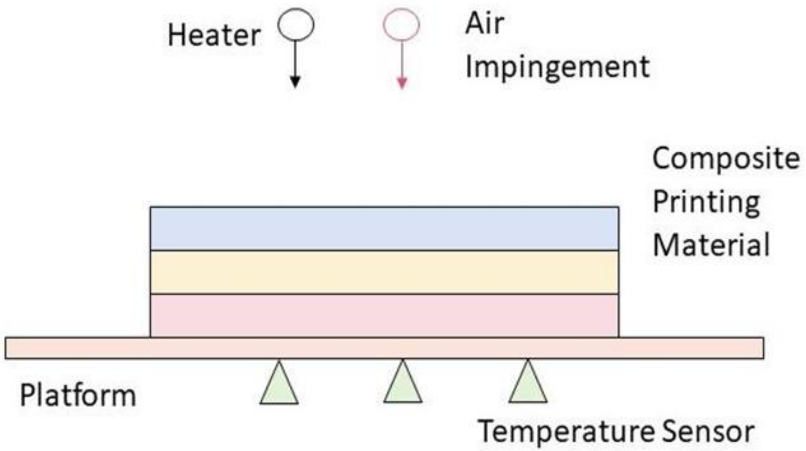


Fixation Process

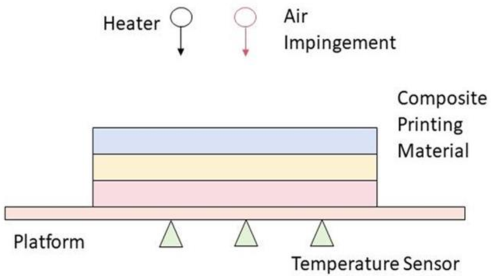
For optimal drying, the moisture content of the printed product is important.

- **No measuring sensor…**
- **Vary over location and time…**

This makes the entire fixation process hard to control.



Governing Physics of the fixation model



- Dynamic space-time varying system
- Dynamic model:

$$\frac{\partial \mathbf{u_i}(x_i,t)}{\partial t} = \overbrace{D_i(x_i)\frac{\partial^2 \mathbf{u_i}(x_i,t)}{\partial x_i^2} + U_i(x_i)\frac{\partial \mathbf{u_i}(x_i,t)}{\partial x_i}}^{\text{PDE}} + K_i(x_i)\mathbf{u_i}(x_i,t) + \overbrace{P_i(x_i)p(t)}^{\text{ODE}}$$

Diffusion coeffient   Transport coeffient   Reaction coeffient

- Boundry Condition

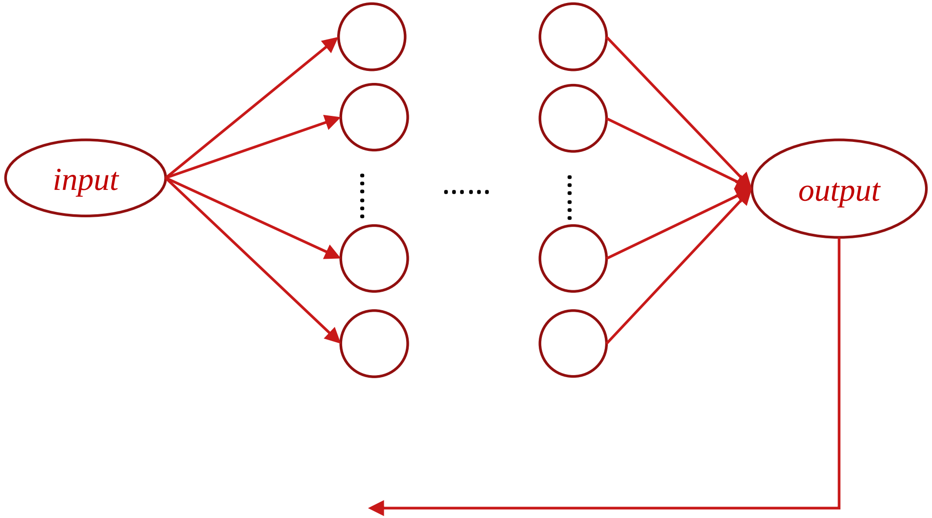
How to optimally combine the physics and data to find the surrogate model of the fixation process?

My solution is using **machine learning** to combine physics and data.

PINN(Physics Informed Neural Network)

PINN(Physics Informed Neural Network)

Normal neural network training is a data-driven optimization problem.



$$u^\star = \operatorname{argmin}_{\mathbf{u}} \frac{1}{N} \sum_{i=1}^{N} L\left(\nu_i, y_i, \mathbf{u}\right)$$

Data-driven

Optimization problem

The prior physics knowledge isn't used in the algorithm…

PINN(Physics Informed Neural Network)

Machine Learning

$$u^\star = \text{argmin}_{\mathbf{u}} \quad \frac{1}{N}\sum_{i=1}^{N} L\left(\nu_i, y_i, \mathbf{u}\right)$$

Data-driven

Optimization problem

Physics Model

$$\begin{cases} \dfrac{\partial v}{\partial t}(t,x) + \mathcal{N}[v](t,x) = 0, \quad (t,x) \in [0,T) \times \Omega, \\ v(\cdot, x)|_{x\in\partial\Omega} = v_{\partial\Omega}, \\ v(0,\cdot) = v_0. \end{cases}$$

Initial condition

Boundary condition

PDE $\Rightarrow$ Model-based

Physics Informed Neural Network

optimization

Measurement error

$$\theta^\star = \text{argmin}_{\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N} \left(u_{\boldsymbol{\theta}}(t_i, x_i) - y_i\right)^2$$

Constraint

$$\text{s. t. } \frac{\partial u_{\boldsymbol{\theta}}}{\partial t}(t,x) + \mathcal{N}[u_{\boldsymbol{\theta}}](t,x) = 0, \quad (t,x) \in [0,T) \times \Omega.$$

PINN is an optimization under constrain problem.

Optimize a PINN

$$\theta^\star = \operatorname{argmin}_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \left(u_{\boldsymbol{\theta}}(t_i, x_i) - y_i\right)^2$$

$$\text{s. t. } \frac{\partial u_{\boldsymbol{\theta}}}{\partial t}(t, x) + \mathcal{N}[u_{\boldsymbol{\theta}}](t, x) = 0, \quad (t, x) \in [0, T) \times \Omega.$$

Optimize a PINN

$$\theta^\star = \text{argmin}_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} (u_{\boldsymbol{\theta}}(t_i, x_i) - y_i)^2$$

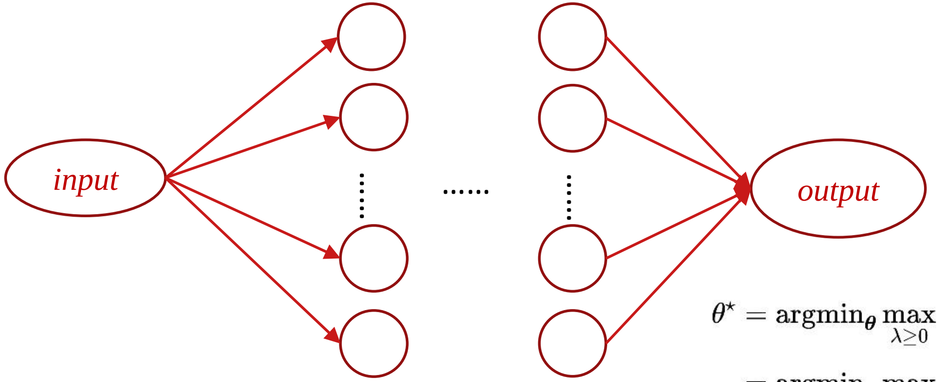
$$\text{s. t.} \iint (\frac{\partial u_{\boldsymbol{\theta}}}{\partial t}(t,x) + \mathcal{N}[u_{\boldsymbol{\theta}}](t,x))^2 dt dx = 0, \quad (t,x) \in [0,T) \times \Omega.$$

Using **Lagrange multiplier:**

$$\theta^\star = \text{argmin}_{\boldsymbol{\theta}} \max_{\lambda \geq 0} \sum_{i=1}^{N} \frac{(u_{\boldsymbol{\theta}}(t_i, x_i) - y_i)^2}{N} + \underbrace{\lambda \iint_{[0,T]\times\Omega} \left(\frac{\partial u_{\boldsymbol{\theta}}}{\partial t}(t,x) + \mathcal{N}[u_{\boldsymbol{\theta}}](t,x) \right)^2 dt dx}_{\text{Physics cost}}$$

PINN(Physics Informed Neural Network)

Different from normal machine learning, the optimization of PINN is:



$$\theta \leftarrow \theta - \alpha_\theta \nabla_\theta \mathcal{L}_\lambda(\theta) \quad \lambda \leftarrow \lambda + \alpha_\lambda \nabla_\lambda \mathcal{L}_\lambda(\theta)$$

(Primal-dual optimization)

$$\begin{aligned}\theta^\star &= \operatorname{argmin}_{\boldsymbol{\theta}} \max_{\lambda \geq 0} \sum_{i=1}^{N} \frac{(u_{\boldsymbol{\theta}}(t_i, x_i) - y_i)^2}{N} + \lambda \frac{1}{N_{\text{phy}}} \sum_{j=1}^{N_{\text{phy}}} \left(\frac{\partial u_{\boldsymbol{\theta}}}{\partial t}(t_i^p, x_i^p) + \mathcal{N}[u_{\boldsymbol{\theta}}](t_i^p, x_i^p) \right)^2 \\ &= \operatorname{argmin}_{\boldsymbol{\theta}} \max_{\lambda \geq 0} \mathcal{L}_\lambda(\theta)\end{aligned}$$

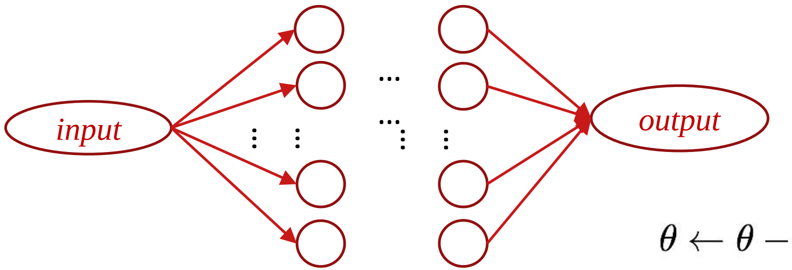
Inverse Problem: Parameter estimation

Parameter estimation is the inverse problem of data-driven optimization. When we lack knowledge of the physics model but have **input dataset** along with corresponding **output**, it's still possible to estimate the parameters in the PDE.

To achieve this, we can **add the unknown parameters into the optimizer as trainable parameters.**

PINN(Physics Informed Neural Network)

This is the principle of parameter estimation:



$$\theta \leftarrow \theta - \alpha_\theta \nabla_\theta \mathcal{L}_\lambda(\theta) \quad \lambda \leftarrow \lambda + \alpha_\lambda \nabla_\lambda \mathcal{L}_\lambda(\theta) \quad \mu \leftarrow \mu - \alpha_\mu \nabla_\mu \mathcal{L}_\lambda(\theta)$$

$$\begin{aligned}\theta^\star &= \operatorname{argmin}_{\boldsymbol{\theta},\mu} \max_{\lambda \geq 0} \sum_{i=1}^{N} \frac{(u_{\boldsymbol{\theta}}(t_i, x_i) - y_i)^2}{N} + \lambda \frac{1}{N_{\text{phy}}} \sum_{j=1}^{N_{\text{phy}}} \left(\frac{\partial u_{\boldsymbol{\theta}}}{\partial t}(t_i^p, x_i^p) + \mathcal{N}_\mu[u_{\boldsymbol{\theta}}](t_i^p, x_i^p) \right)^2 \\ &= \operatorname{argmin}_{\boldsymbol{\theta},\mu} \max_{\lambda \geq 0} \mathcal{L}_\lambda(\theta)\end{aligned}$$

Results Presentation

Dataset Introduction

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} + \beta \frac{\partial u}{\partial x} + \gamma u$$

Use the MATLAB function

sol = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)

to make the dataset. The parameters in PDE, boundary and initial conditions can be defined by yourself.
Define a specific PDE as follow:

$$\frac{\partial u(x,t)}{\partial t} = 3.5 \frac{\partial^2 u(x,t)}{\partial x^2} + 6.5 \frac{\partial u(x,t)}{\partial x} + 8.5 u(x,t) \quad 0 \le x \le 2, 0 \le t \le 1$$

$$u(x,0) = sin(\pi x) \quad x(0,t) = x(2,t) = 0$$

Use the MATLAB function to find the solution, the points in the solution are our dataset.

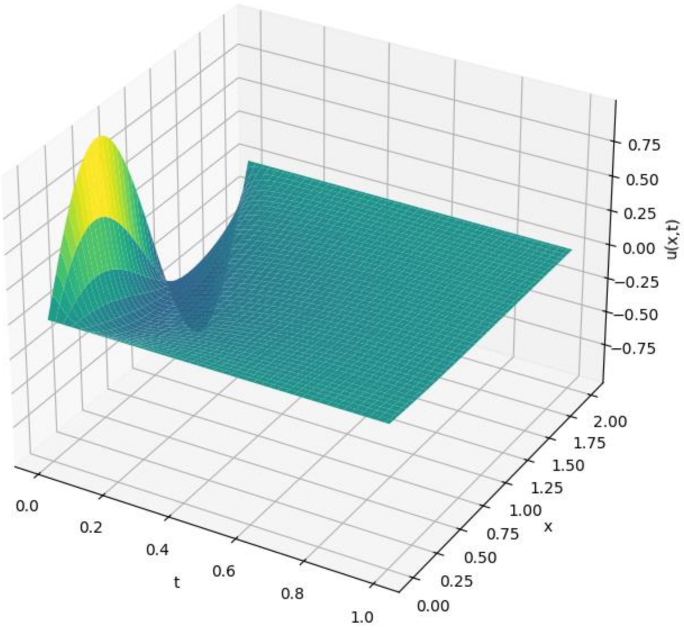
Dataset Introduction

Define a PDE as follow:

$$\frac{\partial u}{\partial t} = 3.5\frac{\partial^2 u}{\partial x^2} + 6.5\frac{\partial u}{\partial x} + 8.5u \qquad 0 \leq x \leq 2, 0 \leq t \leq 1$$

$$u(x,0) = sin(\pi x) \quad x(0,t) = x(2,t) = 0$$

Select 512 x points, and 256 t points. The dataset is visualized as :

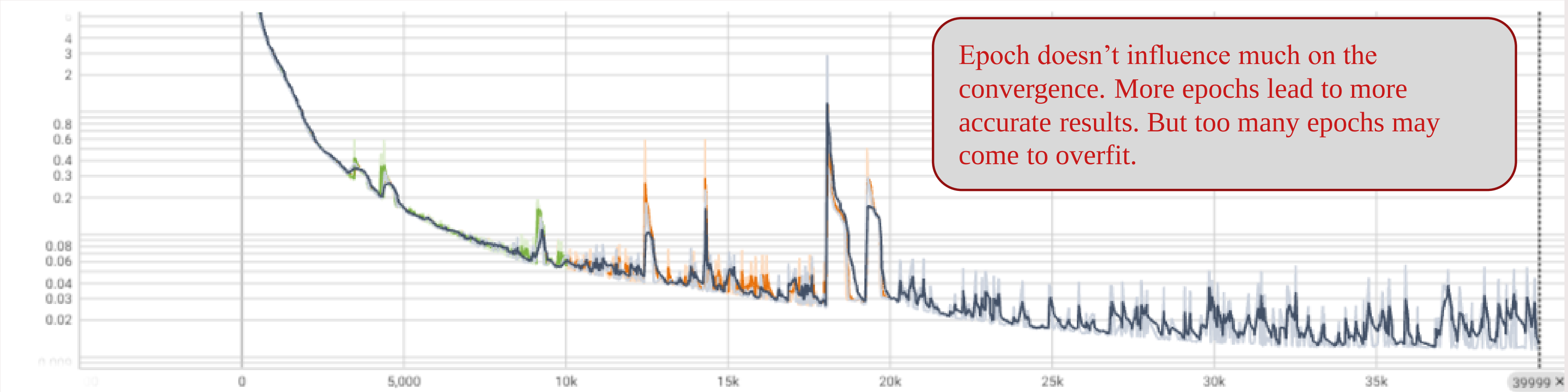


Different Epochs

Dataset size: 3000
Batch size: 1000
Learning rate:0.001
Hidden layers: [32,32]
Activation Function: tanh

α: 3.5   β: 6.5   γ: 8.5

Epoch	α_pred	β_pred	γ_pred	loss
10000	3.3250	6.4638	7.8527	0.0514
20000	3.4084	6.5854	8.1606	0.0298
40000	**3.4521**	**6.6641**	**8.3377**	**0.0115**



Epoch doesn’t influence much on the convergence. More epochs lead to more accurate results. But too many epochs may come to overfit.

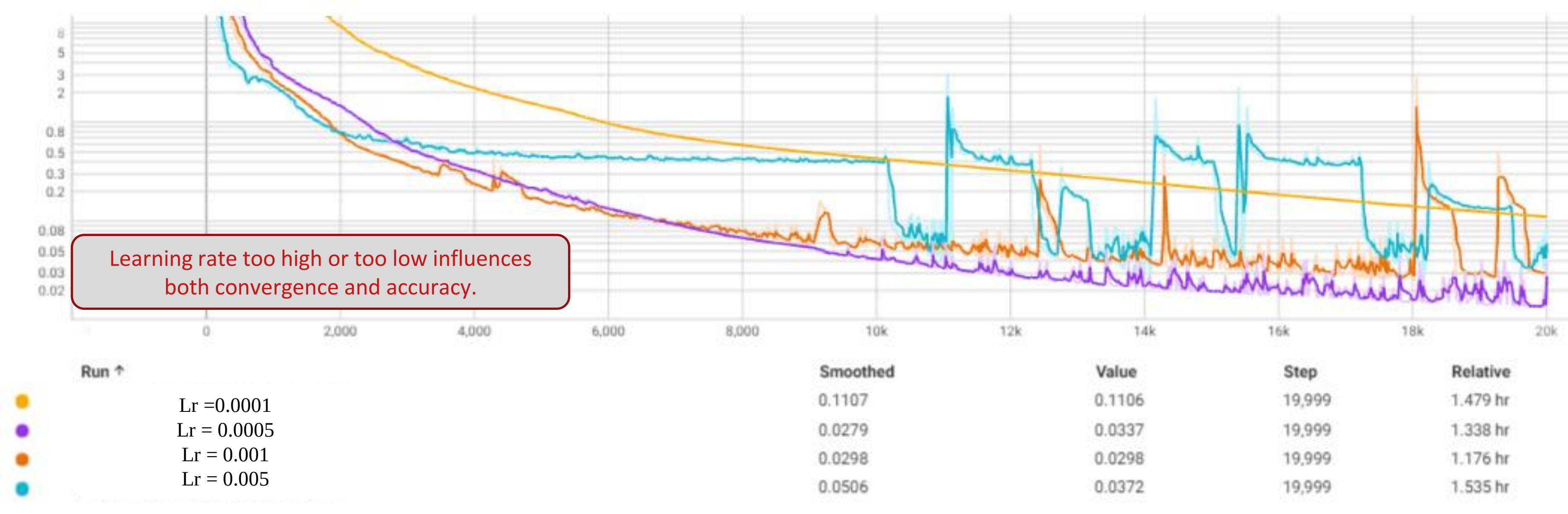
Run ↑	Smoothed	Value	Step	Relative
Epoch=10000	0.0551	0.0514	9,999	39.62 min
Epoch=20000	0.0298	0.0298	19,999	1.176 hr
Epoch=40000	0.0128	0.0115	39,999	2.714 hr

Different Learning Rate

Dataset size: 3000
Batch size: 1000
Epoch: 20000
Hidden layers: [32,32]
Activation Function: tanh

α: 3.5 β: 6.5 γ: 8.5

Lr	α_pred	β_pred	γ_pred	loss
0.0001	2.8040	5.9089	8.1714	0.1107
0.0005	**3.4508**	**6.5293**	**8.3082**	**0.0279**
0.001	3.4084	6.5854	8.1606	0.0298
0.005	3.3871	6.5714	8.0326	0.0506
0.01	3.1979	6.5162	7.4019	0.0951



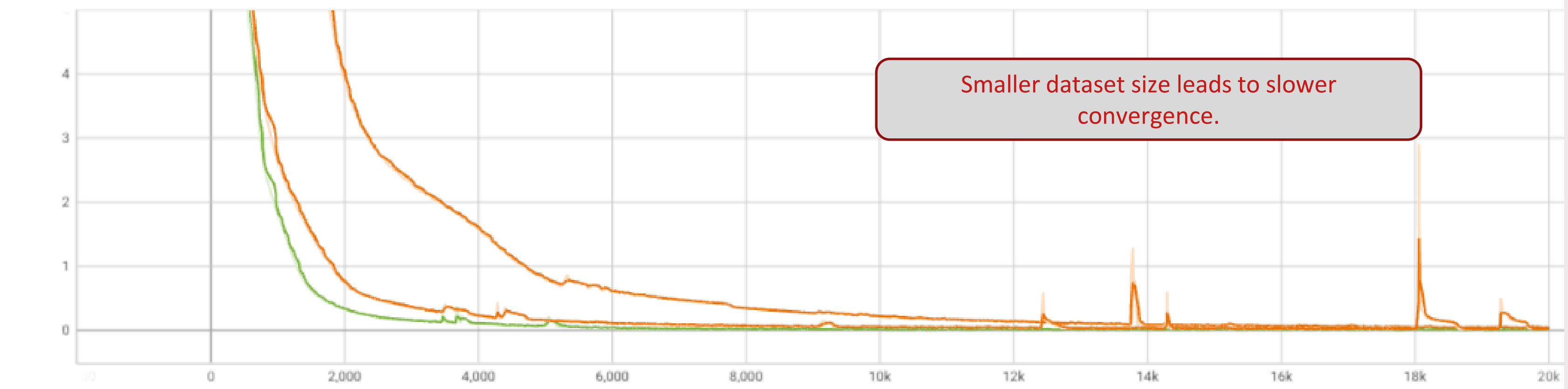
Learning rate too high or too low influences both convergence and accuracy.

Different Dataset Size

Batch size: 1000
Learning rate:0.001
Epoch: 20000
Hidden layers: [32,32]
Activation Function: tanh

α：3.5　β：6.5　γ：8.5

Data size	α_pred	β_pred	γ_pred	loss
1000	3.3487	6.6316	8.0147	0.0376
3000	3.4084	**6.5854**	8.1606	0.0298
6000	**3.4393**	6.5986	**8.2980**	**0.0162**



Run ↑	Smoothed	Value	Step	Relative
Data size = 1000	0.0417	0.0376	19,999	28.06 min
Data size = 3000	0.0298	0.0298	19,999	1.176 hr
Data size = 6000	0.0192	0.0162	19,999	2.416 hr

Test the network on different PDE

Dataset size: 3000
Batch size: 1000
Epoch: 40000

Learning rate:0.0005
Hidden layers: [32,32]
Activation Function: Tanh

	α	α_pred	β	β_pred	γ	γ_pred	loss
I	3.5	3.4521	6.7	6.6641	8.5	8.3377	0.0115
II	5	4.8468	-3.5	-3.3422	-12	-12.2361	0.0168
III	5	4.8190	7	6.6506	-9	-9.3474	0.0069

Future Work

- Tune the hyperparameters or find some other ways to make the results of the network more accurate. And test the network on the data from fixation process.
- Extend the model to the coupled PDE-ODE part, and the model has a spatial coefficient in the parameters.
- After identifying the parameters in the real setup, find a way to predict the change of temperature and moisture in the fixation process in a future state.

Q&A