Lab Talk #3

February 27, 2023

Ozgur Taylan Turan

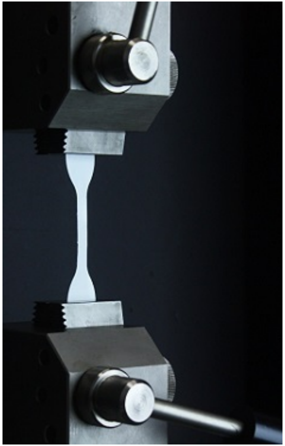
Cooperative Data Driven Modeling[1]

[1] A. Dekhovich, O. T. Turan, J. Yi, and M. A. Bessa (Nov. 2022). *Cooperative Data-Driven Modeling*. arXiv: 2211.12971 [cond-mat]

Introduction

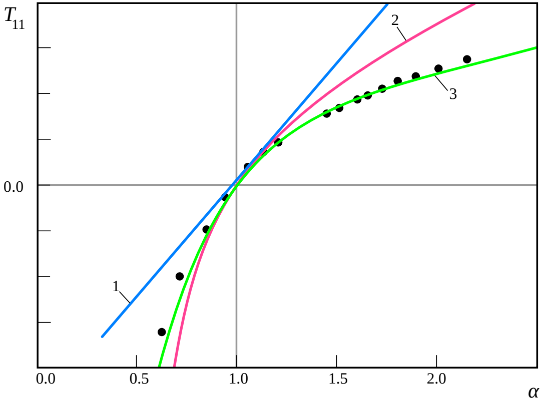
Computational Mechanics (CM)

- Experimental Tests
- Finding a fitting model
- Publish your model without data or the model



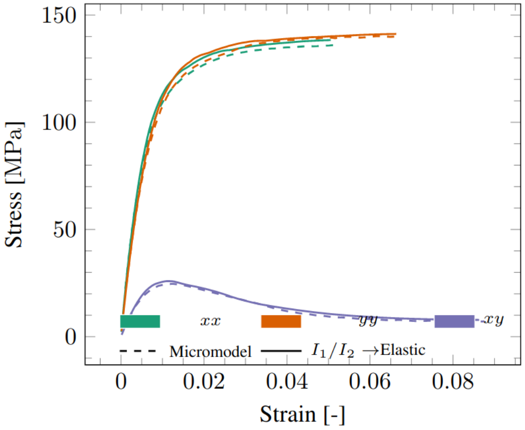
Computational Mechanics (CM)

- Experimental Tests
- Finding a fitting model
- Publish your model without data or the model



Computational Mechanics (CM)

- Experimental Tests
- Finding a fitting model
- Publish your model without data or the model

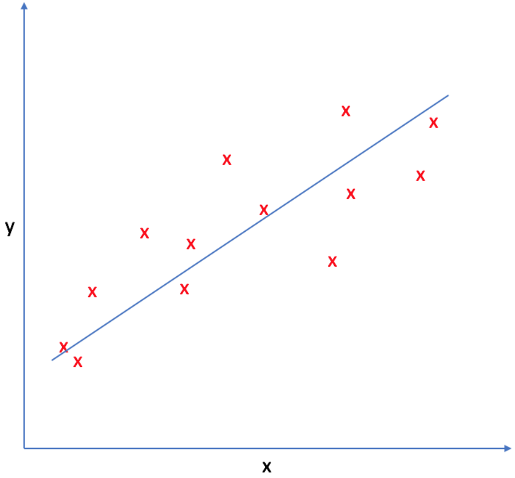


Machine Learning (ML)

- Do the same thing!
- (Mostly) publish data and your model!

Machine Learning (ML)

- Do the same thing!
- (Mostly) publish data and your model!



Intersection of CM & ML

- Open Science
- Sharing Data and Models

Intersection of CM & ML

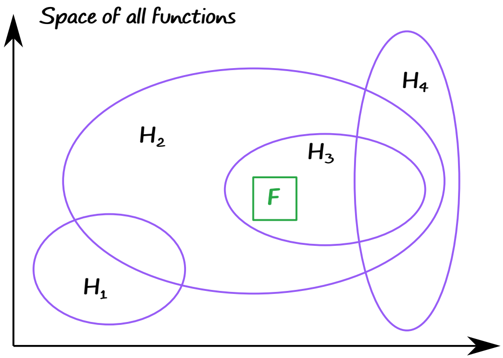
- Open Science
- Sharing Data and Models

Why is this so beneficial?

- Similar tasks
- Similar setups
- Similar behaviours

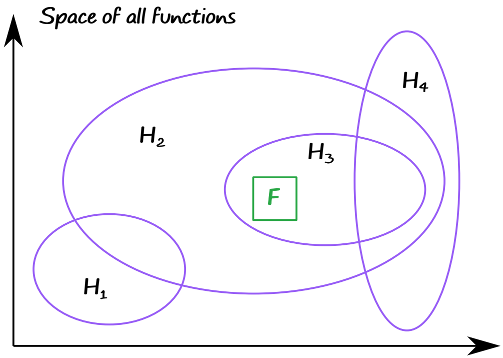
Continual Learning

- Originates back in 90s
- Try to solve stream of tasks
- Exploits similarities between tasks
- Overcome data-scarcity problem

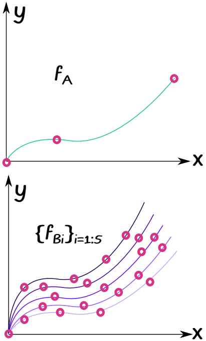


Continual Learning

- Conventional Learning
  $f : \mathcal{X} \to \mathcal{Y}$ with H
- Continual Learning
  $\mathcal{F} := \{f_i : \mathcal{X} \to \mathcal{Y}\}_{i=1}^{M}$ $\mathcal{H} := \{H_i\}_i^M$



AIM

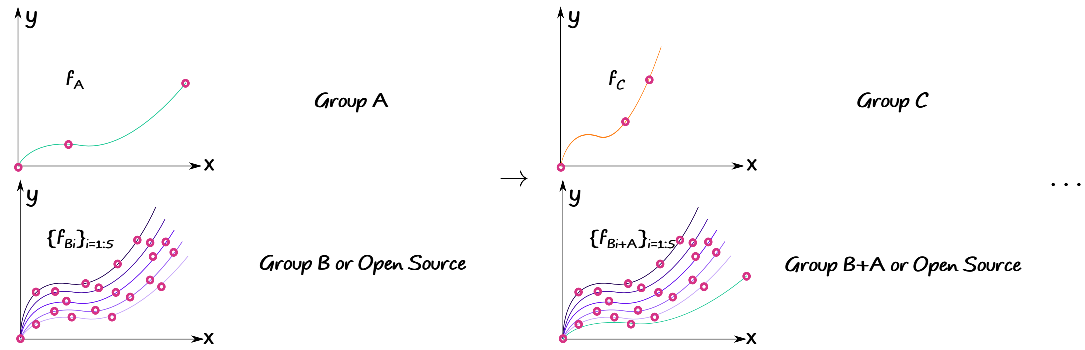


Group A

Group B or Open Source

$\rightarrow \ldots$

AIM


y
x
f_A
Group A
y
x
$\{f_{Bi}\}_{i=1:S}$
Group B or Open Source
→
y
x
f_C
Group C
y
x
$\{f_{Bi+A}\}_{i=1:S}$
Group B+A or Open Source
. . .

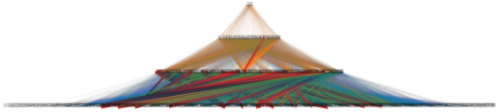
AIM

Currently: Everyone (deep learning enthusiasts) obtain huge amount of data (takes ages to collect), then train their models!

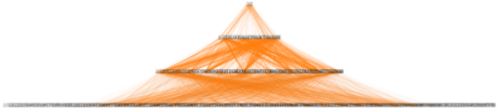
Our Aim: Improve continuous stream of tasks with previous tasks, and show how collaboration might decrease data and parametrization need!

Continual Learning

NNRelief Pruning[1]



(a) LeNet-300-100 pretrained on MNIST



(b) LeNet-300-100 pruned on MNIST

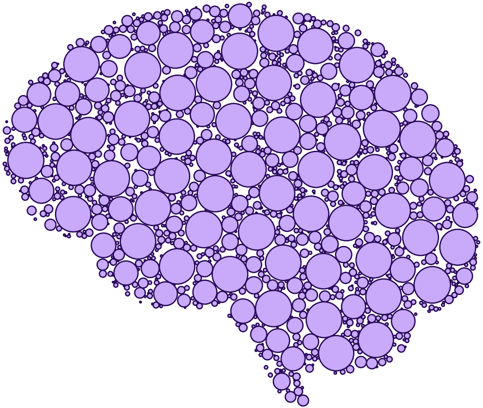
Figure 2: LeNet-300-100 architecture on MNIST before and after pruning, where connections are coloured with respect to importance score: blue (least important) $\rightarrow$ red (most important)

- Train your NN for a task of $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Compute the importance scores of the activations
- Prune them with a pre-defined tolerance

[1] A. Dekhovich, D. M. J. Tax, M. H. F. Sluiter, and M. A. Bessa (June 2022b). *Neural Network Relief: A Pruning Algorithm Based on Neural Activity*. arXiv: 2109.10795 [cs]

Continual Prune & Select[1]

- Given a task $f_i : |\mathcal{X} \to \mathcal{Y}$ find sub-network $\mathcal{N}_i$ inside a given NN $\bigcup_{i=1}^{M} \mathcal{N}_i$
- Prune with NNRelief
- Fix the sub-network parameters and repeat the same procedure for f_{i+1}
- In the end you end up with task specific networks



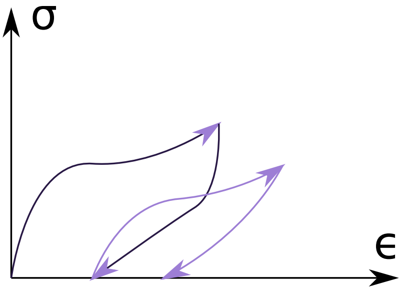
[1] A. Dekhovich, D. M. J. Tax, M. H. F. Sluiter, and M. A. Bessa (Aug. 2022a). *Continual Prune-and-Select: Class-incremental Learning with Specialized Subnetworks*. arXiv: 2208.04952 [cs]

Application

Plasticity

Plastic Constitutive Law:

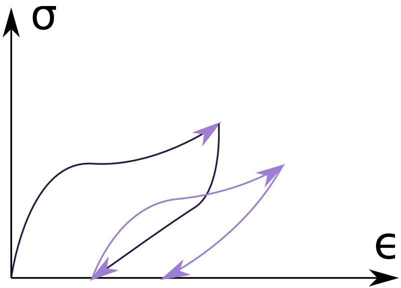
- Path dependent problem
- $\boldsymbol{\sigma} = \mathcal{C}(\epsilon, t, \boldsymbol{\theta})$
- $\sigma \in \mathbb{R}^3$ and $\epsilon \in \mathbb{R}^3$



Plasticity

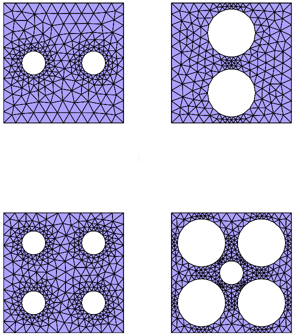
As a continual learning problem:

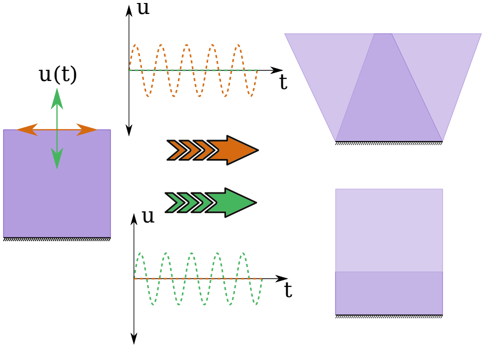
- $\boldsymbol{\theta}$ Determines the different tasks
- Try to learn $\mathcal{C}$
- Given $\{\sigma_{t=1:T}^{\boldsymbol{\theta}_i}, \boldsymbol{\epsilon}_{t=1:T}^{\boldsymbol{\theta}_i}\}_i = 1, M$



Problem

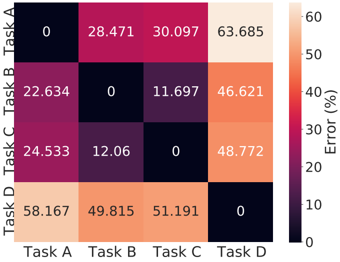
- 4 Tasks with 800 available train paths and 200 test paths (sampled from Gaussian Process)
- Assume the task-ID is known

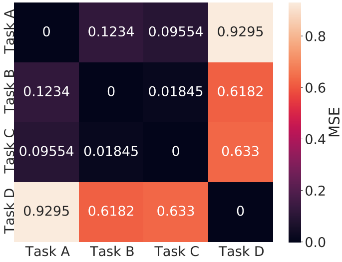




Difference of Tasks

- Same strain paths for each task
- Different strain localization





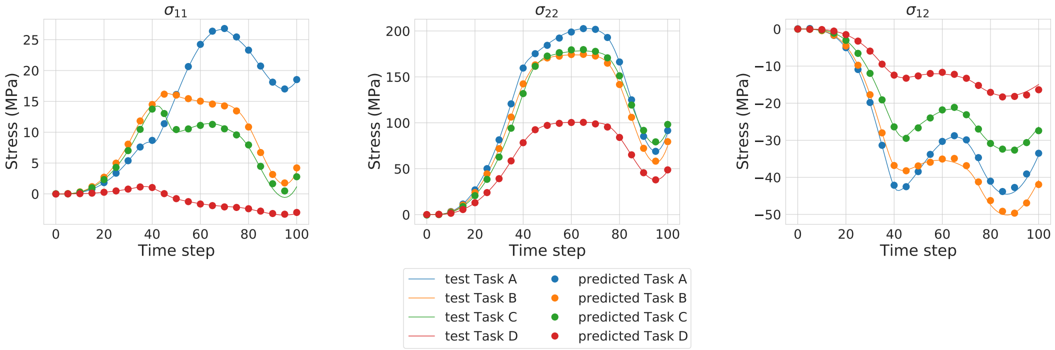
Some Hyper-parameters

- Not the best...
- 2 cells GRU with 128 hidden size

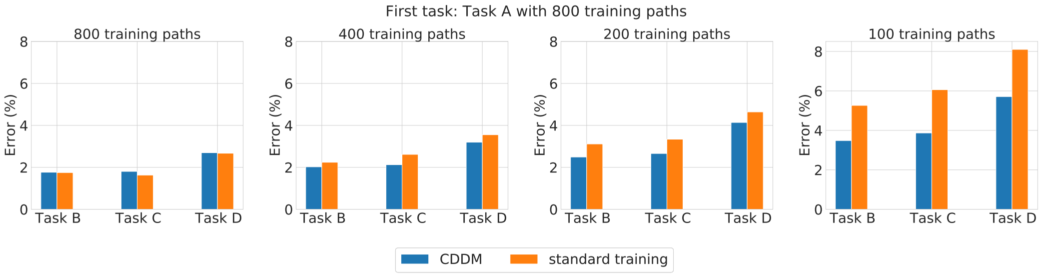
epochs	lr	wd	pruning iter	pruning param α	retrain epochs
1000	0.01	10^{-6}	1	0.95	200

Example

- With 800 training paths for all tasks

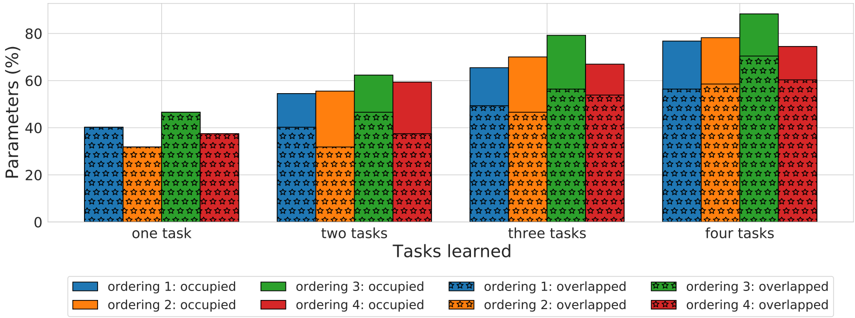


Compare with Standard Learning



Same holds for other orderings as well!

Parameters


Parameters (%)
0
20
40
60
80
one task
two tasks
three tasks
four tasks
Tasks learned
ordering 1: occupied
ordering 2: occupied
ordering 3: occupied
ordering 4: occupied
ordering 1: overlapped
ordering 2: overlapped
ordering 3: overlapped
ordering 4: overlapped

Parameters


Parameters overlapped (%)
100
90
80
70
60
50
40
30
20
10
0
one task
two tasks
three tasks
four tasks
Tasks learned
ordering 1
ordering 2
ordering 3
ordering 4

Conclusions & Future Work

Conclusion

- Less number of parameters compared to standard learning.
- Ordering does not matter.
- Mostly better than standard learning especially in scarce data regime!

However

- Deep stuff is not for me!
- What happens for a much wider task variance?
- This strategy is limited by number of parameters.

Thanks!