Composites Research Network

sampe Conference & Exhibition

THE UNIVERSITY OF BRITISH COLUMBIA
Materials & Manufacturing Research Institute

When Data-efficient Machine Learning Comes to the Rescue: An AI-based Optimization Framework for Advanced Manufacturing

Python tutorial: https://github.com/miladramzy/SAMPE2023_Tutorial

Tutorial Instructors:

Milad Ramezankhani, PhD Candidate
& Abbas S. Milani, PhD, PEng

SAMPE 2023
Seattle Convention Center
Seattle, Washington
April 17-20, 2023

When Data-efficient Machine Learning Comes to the Rescue: An AI-based Optimization Framework for Advanced Manufacturing

https://github.com/miladramzy/SAMPE2023_Tutorial

Tutorial requirements for participants:

1. Personal laptop
   - Software installation is NOT needed as the codes will be run on the cloud.
2. Internet connection
3. Google (Gmail) account to use Colab notebooks
   - https://accounts.google.com/signup/v2/webcreateaccount?flowName=GlifWebSignIn&flowEntry=SignUp
4. Optional: GitHub account
   - https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home

CRN Composites Research Network

THE UNIVERSITY OF BRITISH COLUMBIA

Materials & Manufacturing Research Institute
Experience Innovation with Us

sampe Conference & Exhibition

Content/Agenda

- Introductions to JAX
- Data loading (source and target)
- Create network architectures
- Preprocessing data
- Training the source model
- Transfer learning (knowledge transfer)
- Evaluate models' performance

JAX

- JAX is a Python library designed for high-performance numerical computing, especially machine learning research.

- Its API for numerical functions is based on NumPy, a collection of functions used in scientific computing.

- JAX is NumPy on the CPU, GPU, and TPU, with fast and efficient automatic differentiation for high-performance machine learning research.
- Key features:
    - **Differentiation**: Gradient-based optimization is fundamental to ML. JAX natively supports both forward and reverse mode automatic differentiation of arbitrary numerical functions.
    - **Vectorisation**: JAX provides automatic vectorisation that allows us to automatically transform a function that works for a single input to a function that works for a vector of inputs (batching).
    - **JIT-compilation**: *XLA* is used to *just-in-time (JIT)-compile* and execute JAX programs on GPU and Cloud TPU accelerators.

References:

- https://www.deepmind.com/blog/using-jax-to-accelerate-our-research
- https://github.com/PredictiveIntelligenceLab/TRIPODS_Winter_School_2022
- https://jax.readthedocs.io/en/latest/notebooks/quickstart.html
- https://www.cs.ubc.ca/~fwood/CS340/lectures/AD1.pdf

Jax – Code explained

- Even though JAX has a syntax that is similar to NumPy in most cases, the pseudo random number generation (PRNG) is a notable exception. JAX instead implements an **explicit** PRNG.

```
from jax import random
key = random.PRNGKey(0)
print(random.normal(key, shape=(1,)))
```

- Computing Gradients in JAX: `jax.grad`, which takes a function and returns a new function that computes the gradient of the original function.

```
def square(x):
  return x**2
sqr_grad = grad(square)
```

Data-Efficient Machine Learning

Transfer learning

Case study: Transfer learning composites autoclave processing

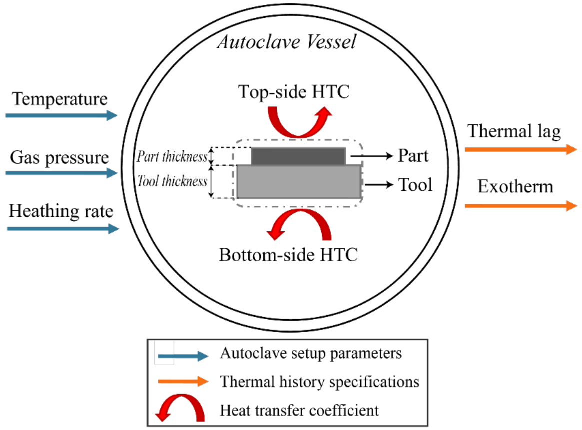
Source material:
7 mm AS4/8552 composite part
(Abundant historical data)

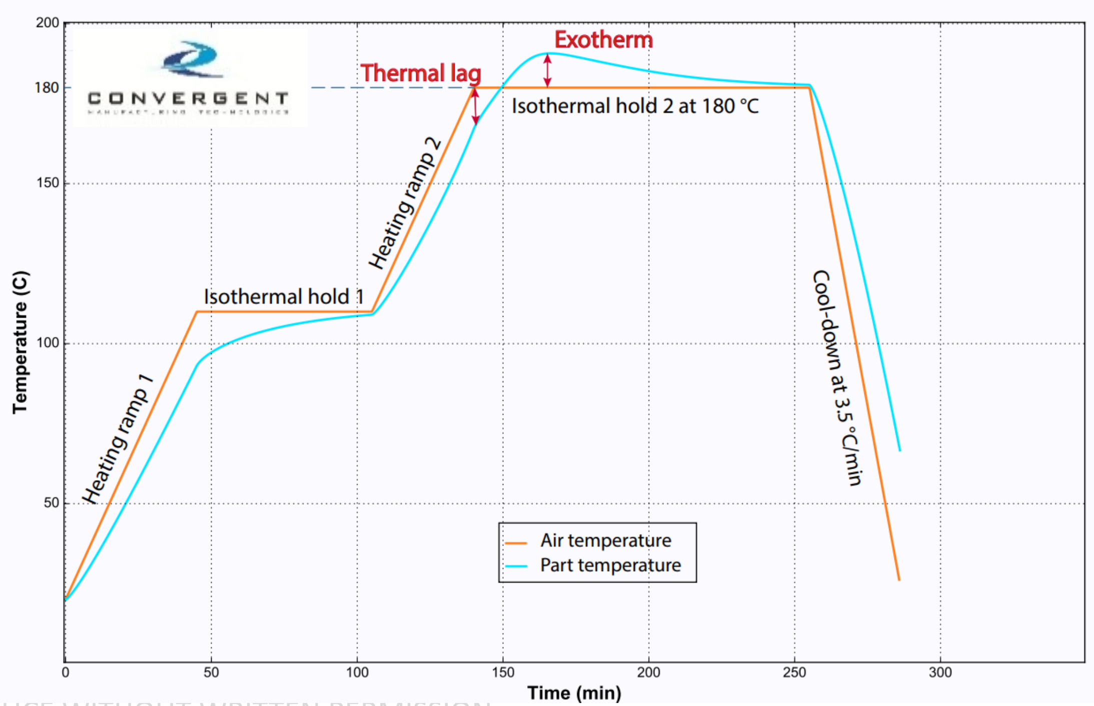
- ->

Target material:
12 mm AS4/8552 composite part
(limited data)

Regression problem:
Prediction of **exotherm**

Input variables	Min	Max
Heat rate – ramp 1 (°C/min)	1	5
Isothermal hold 1 (°C)	105	125
Heat rate – ramp 2 (°C/min)	1	5
Top-side HTC (W/m^{2}K)	10	100
Bottom-side HTC (W/m^{2}K)	10	100





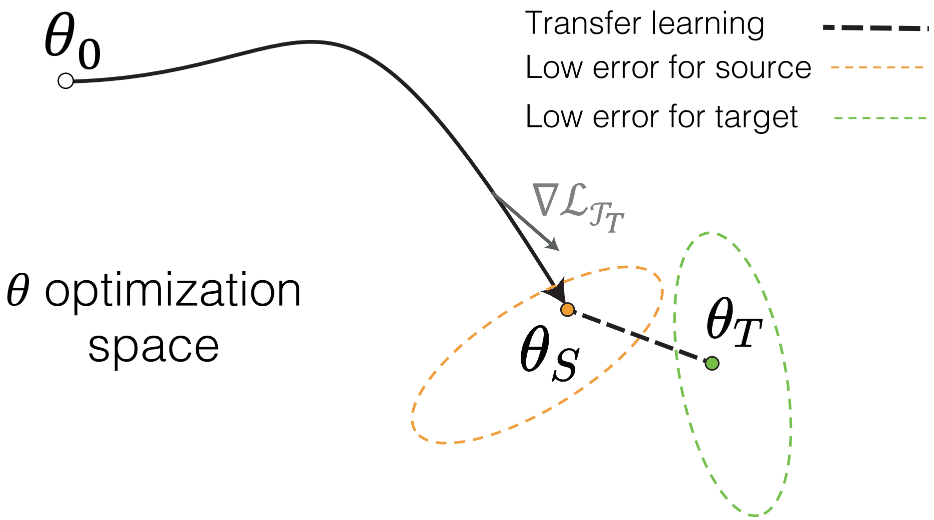
Transfer learning – Neural networks

A neural network model NN is expressed by a parametrized function $f(\theta)$.

Learned (optimized) parameters of the source task $\mathcal{T}_s \rightarrow \theta_s$

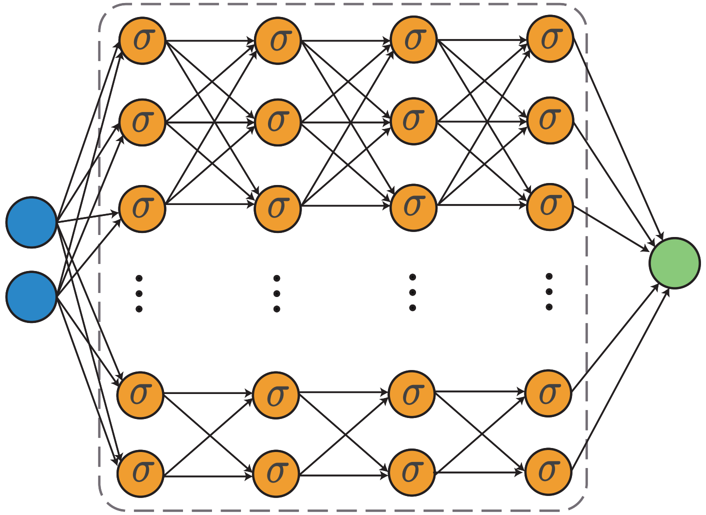
$$\theta_T \leftarrow \theta_S - \alpha \nabla_{\theta_S} \mathcal{L}_{\mathcal{T}_T}(f_\theta)$$

Small changes in the parameters will provide necessary improvements toward learning the target task.



Source task: $\mathcal{T}_s$

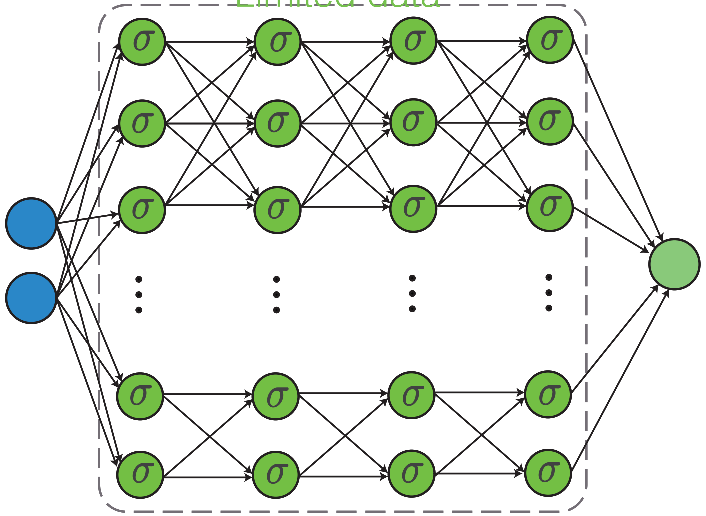
Abundant data



Initialize the target network with the optimized parameters of the source network

Target task: $\mathcal{T}_T$

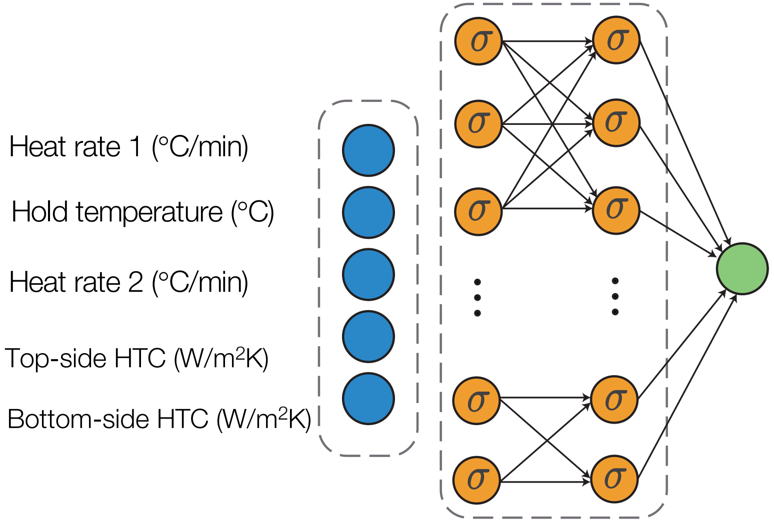
Limited data



Transfer learning – Neural networks

Source task: $\mathcal{T}_s$

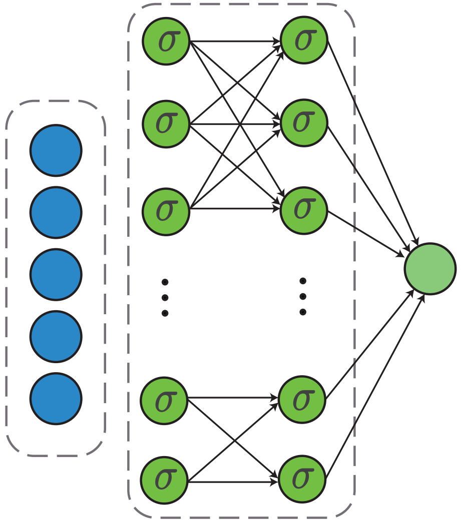
Abundant data



Initialize the target network with the optimized parameters of the source network

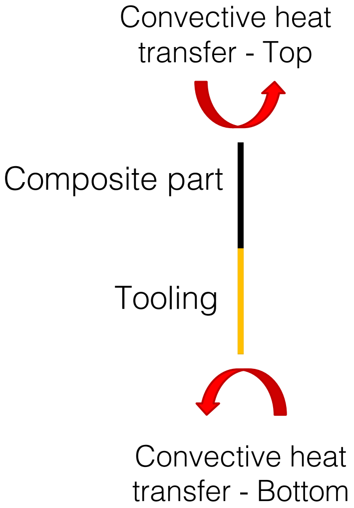
Target task: $\mathcal{T}_T$

Limited data



Source network:
- Input layer size: 5
- Hidden layers: 2 (32 neurons)
- Activation function: relu
- Output layer size: 1

Source data:
- Training set: [656, 5]
- Test set: [219, 5]



Target network:
- Input layer size: 5
- Hidden layers: 2 (32 neurons)
- Activation function: relu
- Output layer size: 1

Target data:
- Training set: [75, 5]
- Test set: [25, 5]

TL procedure

- Data loading (source and target)
- Create network architectures
  - Same architecture for both source and target networks
- Preprocessing data
  - Standardize features by removing the mean and scaling to unit variance
  - Split data into training, validation and test sets
- Training the source model
  - An accurate model can be trained as abundant data is available
- Transfer learning (knowledge transfer)
  - Leverage tasks similarities (same material, different thicknesses) to transfer knowledge
- Evaluate models' performance

Hands-on coding in Python

https://github.com/miladramzy/SAMPE2023_Tutorial

Thank you

https://www.linkedin.com/in/miladramezankhani/

CRN
Composites Research Network

UBC THE UNIVERSITY OF BRITISH COLUMBIA

Materials & Manufacturing
Research Institute
Experience Innovation with Us

sampe
Conference & Exhibition