Marshall Plan Report

Scientific Machine Learning for Injection Molding Simulation

Wissenschaftliches maschinelles Lernen in der Spritzgusssimulation

Project conducted at the
Computer Science Research Institute,
Sandia National Labratories,
Albuquerque, NM, US

Author	Roxana Pohlmann	*TU Wien*
Advisors	Dr. Rekha Rao	*Sandia National Labratories*
	Assoc. Prof. Nat Trask	*University of Pennsylvania*
	Univ. Prof. Stefanie Elgeti	*TU Wien*

Contents

Acronyms

Acronym	**Phrase**
BN	batch norm
BS	batch size
BVP	boundary value problem
ChebConv	Chebychev Convolution
CNN	convolutional neural network
D	dropout
ELU	exponential linear unit
FEM	finite element method
GAT	Graph Attention Network
GNN	graph neural network
GPR	gaussian process regression
GRF	gaussian random field
LEE	numbers of layers of edge encoder
LNE	numbers of layers of node encoder
LP	numbers of layers of processor
MSE	mean squared error
NHE	number of hidden neurons in edge encoder
NHN	number of hidden neurons in node encoder
NHP	number of hidden neurons in processor
NN	neural network
PDE	partial differential equation
POD	proper orthogonal decomposition
RBF	radial basis functions
ReLU	rectified linear unit

Acronym	**Phrase**
ROM	reduced order model
SVD	singular value decomposition

Symbols

Symbol	**Description**
Continuum Physics	
α	Coefficient of Thermal Expansion
$\mathbf{b}$	Body force
c_1	Material parameter for Neo-Hook
$\mathbf{E}$	Green-Lagrangian strain energy
$\mathbf{F}$	Deformation tensor
$\bar{\mathbf{F}}$	Isochoric part of $\mathbf{F}$
$\mathbf{F}_\theta$	Thermal part of $\mathbf{F}$
g_u	Dirichlet boundary condition
Γ	Boundary of Ω
Ξ	Portion of Spatial Boundary Γ with Essential Boundary Conditions
Γ_D	Boundary of Ω with Dirichlet boundary conditions
Γ_N	Boundary of Ω with Neumann boundary conditions
h_τ	Neumann boundary condition
J	Volumetric change (det $\mathbf{F}$
λ	Material parameter for Neo-Hook
$\boldsymbol{n}$	Outward-Pointing Unit Normal Vector of Spatial Boundary Γ
ν	Poisson ratio
Ω	Reference domain
$\tilde{\Omega}$	Deformed domain
$\mathbf{P}$	First Piola-Kirchhoff tensor

Symbol	**Description**
Ψ	Free energy functional
Ψ_{vol}	Free energy with volumetric change
θ	Initial temperature
Θ	Initial temperature field
θ_{F}	Final temperature
u	Displacement vector
U	Displacement field
x	Spatial coordinate in Ω
$\tilde{x}$	Spatial coordinate in $\tilde{\Omega}$
GNN-based model	
l	Lenghtscale of gaussian random field (GRF)
N_{Geom}	Number of geometries in dataset
Graph	
$\mathbf{A}$	Adjacency matrix
$\mathbf{A}_w$	Adjacency matrix
$\mathbf{D}$	Degree matrix
$\mathcal{E}$	Set of graph edges $\{\mathbf{e}\}$
$\mathbf{e}$	Graph edge
f	Arbitrary signal
f_i	Graph's representation of an arbitrary signal
g	Arbitrary signal
$\hat{g}$	Arbitrary spectral signal
$\mathbf{L}$	Graph Laplacian
Λ	Eigenvalues of $\mathbf{L}$
λ	Eigenvalue of $\mathbf{L}$
$N_{\mathcal{V}}$	Number of graph nodes
Ψ	Eigenvectors of $\mathbf{L}$
$\mathcal{V}$	Set of graph vertices $\{\mathbf{v}\}$
$\mathbf{v}$	Graph node / vertex
e^w	Graph edge weight

Symbol	**Description**
Neural Networks	
$\mathbf{a}$	Edge attention mechanism
α	Collection of neural network (NN) parameters
$\mathbf{b}$	NN bias
d	Input data dimension
d'	Output data dimension
e	edgeImporance
η	Learnable polynomial multipliers
Φ	Data-driven model
γ	Localized spectral graph filter
Γ	Learnable spectral multipliers
h	Hidden features
$\mathcal{L}$	Loss
m	Output data dimension
n	Input data size
$\mathcal{N}$	Node neighborhood
$\oplus$	Aggregation function
$\mathbf{x}$	Node features
Φ	Learnable permutation-invariant function
Υ	Learnable feature transformation
σ	NN activation function
softmax	Softmax function: $\text{softmax}(\mathbf{x}_i) = \exp x_i / \sum_j \exp x_j$
T	Chebychev polynomial
ϑ	Collection of NN parameters
$\mathbf{w}$	NN weights
$\mathbf{W}_a$	Attention weights
$\mathbf{X}$	Input data to NN layer
$\mathbf{y}$	Node output features
$\mathbf{y}$	Target output data
Proper Orthogonal Decomposition	
$\tilde{\mathbf{X}}$	Snapshot matrix of samples projected on $\tilde{\mathbf{U}}$

Symbol	**Description**
$\mathbf{X}$	Snapshot matrix of samples
k	Dimension of reduced problem
$\boldsymbol{\mu}$	Parameters for one sample $\boldsymbol{x}$
$e_p(k)$	Error of projecting the dataset $\mathbf{X}$ to a linear subspace.
$\boldsymbol{\psi}$	Eigenvector in $\mathbf{U}$
$\boldsymbol{\Sigma}$	Diagonal matrix of singular values of $\mathbf{X}$
$\tilde{\boldsymbol{\Sigma}}$	Truncated diagonal matrix of singular values of $\mathbf{X}$
$\mathbf{U}$	Matrix of left eigenvectors
$\tilde{\mathbf{U}}$	Truncated matrix of left eigenvectors
$\mathbf{V}^*$	Matrix of right eigenvectors
$\boldsymbol{x}$	Sample instance, vector in $\mathbf{X}$

POD-GPR model

e	Prediction error between predicted output $\tilde{\mathbf{y}}$ and output $\mathbf{y}$
$\mathcal{W}$	Data for Essential Spatial Boundary Ξ
e_{r}	Relative prediction error
e_{reg}	Regression error
μ	Parameter vector containing the mean and covariance of a bivariate Gaussian
N_{samples}	Number of samples
$\tilde{\mathcal{V}}$	Regressed operator from μ to U
$\mathcal{V}$	Mapping from μ to Θ
$\mathbf{y}$	Output
$\tilde{\mathbf{y}}$	Predicted output

List of Figures

List of Tables

1 Introduction

Injection molding is characterized by injecting a polymer melt into a cavity of a predefined geometry, where the material fills the domain and subsequently solidifies before it is ejected. Inevitably, the process results in an inhomogeneous temperature distribution within the product, causing warpage from residual stresses, resulting in differences between the mold shape and final product. The relationship between the cavity's shape, processing parameters, and the final part geometry is highly nonlinear and surpasses engineering intuition. Thus, partial differential equations (PDEs)-based simulation and numerical design have become an essential tool in process design. While exploring the design space of industrial processes, numerical design optimization requires many function evaluations. Solving the fully-resolved simulation is time-consuming with a large design space; the computational time of design optimization can grow to impractical values. A fast regression model can allow the exploration of a large space at limited, yet sufficient accuracy.

In many fields of engineering, product design typically targets a global objective function. For example, in aerodynamics, optimization efforts often focus on reducing drag, while topology optimization focuses on minimizing compliance. In contrast, in injection molding, the objective is to obtain the desired product shape, making the number of design objectives as vast as the number of products. In addition, the design objectives are very localized. While shrinkage warpage is generally undesirable, local regions of a product may need to conform exactly to the target shape while other regions are subject to larger tolerances. To account for this diversity, as well as for general applicability to all common simulation methods, the focus of this work is on the generation of black-box – i.e., non-intrusive – reduced-order models for the fully-resolved solution of PDEs.

In particular, we examine proper orthogonal decomposition (POD) [4] as a model for a linear combination of basis vectors and graph neural networks (GNNs) [17, 51] as an approach for nonlinear combination. Using plastic injection molding as a use case, we compare the performance, capabilities, and limitations of generating non-intrusive reduced-order models with POD and GNNs. In particular, we focus on their ability to deal with varying simulation settings and geometries, such as mesh types and sizes.

2 Material model

2.1 General continuum mechanics

Let $\mathbf{x} \in \Omega$ denote a spatial coordinate in reference (or initial) configuration of a spatial domain and $\tilde{x} \in \tilde{\Omega}$ denote a deformed mechanical configuration in equilibrium. The deformed spatial domain results from a displacement of the initial spatial domain

$$\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{u}\,. \tag{2.1.1}$$

The deformation tensor $\mathbf{F}$ maps the initial to the deformed configuration

$$\mathbf{F}_{ij} = \frac{\partial \tilde{x}_i}{\partial x_j} = \mathsf{I} + \frac{\partial u_i}{\partial x_j}\,, \tag{2.1.2}$$

where J denotes the volumetric change

$$J = \det \mathbf{F}\,. \tag{2.1.3}$$

The deformation tensor describes deformation on the domain and is the basis for defining strain and stress. The Green-Lagrangian strain tensor $\mathbf{E}$ quantifies the strain, i.e. the deviation from the reference configuration

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}\mathbf{F}^T - \mathsf{I}\right)\,. \tag{2.1.4}$$

The first Piola-Kirchhoff stress tensor $\mathbf{P}$ describes material forces and stresses in the deformed configuration, while the area's domain still corresponds to the initial configuration. Thus, it allows to solve for integral equations in the deformed configuration, while integrating over the reference equations.

2.1.1 Balance equations

Every physical system must at all times obey the conservation of linear momentum and energy. For a stationay system, the balance of linear momentum in the Lagrangian description reads

$$\mathbf{0} = \nabla \cdot \mathbf{P} + \mathbf{b} \quad \text{on}\,\Omega\,, \tag{2.1.5}$$

where $\mathbf{b}$ are body forces. A domain changes its shape only if constraints are applied, which can be a prescribed displacement g_u (a Dirichlet boundary condition), a prescribed traction h_τ (a Neumann boundary condition), or prescribed forces as a body force or a thermally induced force. The conservation of energy (Equation 2.1.5) is valid at every point, leading to the following boundary value problem (BVP):

$$\int_\Omega \operatorname{div} \mathbf{P} + \mathbf{b} \;\mathrm{d}\Omega = \mathbf{0} \tag{2.1.6}$$

$$\int_{\Gamma_\mathrm{D}} (u_i - g_u)\;\mathrm{d}\Gamma_\mathrm{D} = \mathbf{0} \tag{2.1.7}$$

$$\int_{\Gamma_\mathrm{N}} (\mathbf{P}_{ij}\boldsymbol{n}_j - h_\tau)\;\mathrm{d}\Gamma_\mathrm{D} = \mathbf{0}\,. \tag{2.1.8}$$

2.2 Hyperelasticity

Hyperelastic materials have a strain energy function $\Psi(\mathbf{F}^T\mathbf{F})$ that describes the energetic state of a specific configuration. The equilibrium corresponds to a minimum of this energy function. Hyperelastic materials are defined by the property, that the second Piola-Kirchhoff tensor equals the derivative of the strain energy with respect to the deformation tensor

$$\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}}\,. \tag{2.2.1}$$

In addition, the $\mathbf{F}$ has a number of invariants related to indifference to the chosen frame (confer [30], p.233), which can be used to define material laws

$$I_1 = \mathrm{tr}(\mathbf{F}^T\mathbf{F}) \tag{2.2.2}$$

$$I_2 = \frac{1}{2}[\mathrm{tr}(\mathbf{F}^T\mathbf{F})^2 - \mathrm{tr}((\mathbf{F}^T\mathbf{F})^2)] = \mathrm{tr}((\mathbf{F}^T\mathbf{F})^{-1})\det(\mathbf{F}^T\mathbf{F}) \tag{2.2.3}$$

$$I_3 = \det(\mathbf{F}^T\mathbf{F}) = J^2\,. \tag{2.2.4}$$

2.2.1 Neo-Hook

The material law of interest is the compressible *decoupled* Neo-Hook (confer [30] p. 247)

$$\Psi(J, \bar{I}_1, \bar{I}_2) = \Psi_{\mathrm{vol}}(J) + c_1(\bar{I}_1 - 3)\,. \tag{2.2.5}$$

where Ψ refers to the overall strain energy function, Ψ_{vol} to the volumetric contribution and the second term to the isochoric contribution to the strain energy. Choosing the volumetric function proposed by Simo and Miehe [50], the free energy reads

$$\Psi(J, \bar{I}_1, \bar{I}_2) = c_1(\bar{I}_1 - 3) + \frac{1}{4}(J^2 - 1 - 2\ln J)\ \ , \tag{2.2.6}$$

with material parameter c_1. The first Piola-Kirchhoff tensor follows from differentiation of the strain energy function

$$\mathbf{P} = 2c_1(\mathbf{F} - \mathbf{F}^{-T}) + 2\frac{\lambda}{2}(J-1)J\mathbf{F}^{-T}\,. \tag{2.2.7}$$

2.3 Thermally-induced deformation

The thermal deformation can be included in the deformation tensor as a multiplicative split [29, 37]

$$\mathbf{F} = \mathbf{F}_\theta\bar{\mathbf{F}}\,, \tag{2.3.1}$$

where $\mathbf{F}_\theta$ describes only the thermally induced part of the deformation that can be described as

$$\mathbf{F}_\theta = \exp\left(\int_\theta^{\theta_{\mathrm{F}}} \alpha(\hat{\theta})\,\mathrm{d}\hat{\theta}\right)\mathbf{I}\,. \tag{2.3.2}$$

Where $\alpha(\theta)$ is the factor of thermal expansion, Θ is the final temperature and Θ_0 the initial temperature. Assuming constant thermal expansion factor and isotropic material behaviour, the expression can be evaluated to

$$\mathbf{F}_\theta = \exp\left(\theta - \theta_0\right)\mathbf{I}\,. \tag{2.3.3}$$

The first Piola-Kirchhoff tensor for the thermo-mechanical problem follows directly from (2.2.7), where $\mathbf{F}$ includes the thermal contribution.

2.4 Shrinkage-warpage

The shrinkage-warpage is modeled as deformation from an initial inhomogeneous temperature field θ to an equilibrium at constant temperature θ. The solution is derived by solving BVP Equation 2.1.6–2.1.8 with the corresponding thermally-induced stress Equation 2.3.1 with the finite element method (FEM).

3 Data-Driven Methods

3.1 Proper Orthogonal Decomposition

One of the most popular methods for non-intrusive reduced order models is the projection to a linear subspace of a dataset by POD [10]. It follows the assumption that the underlying physics of a spatio-temporal system takes place on a manifold, which is much lower-dimensional than the degrees of freedom in the original dataset. Under this assumption, the system can be transferred to a basis where the first few basis functions carry the most important information. POD is equivalent to applying singular value decomposition (SVD) to aggregated data of a physical system (usually generated by solving a PDE). The **snapshot matrix X** represents solution of the discrete physical system as a collection of snapshots, where each snapshot contains the solution to the system at a specific parameter value $\boldsymbol{\mu}_i$

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_m \\ | & | & & | \end{bmatrix} . \tag{3.1.1}$$

The SVD is uniquely defined for any real or complex-valued matrix $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^* . \tag{3.1.2}$$

The result consists of the square matrix of left eigenvectors $\mathbf{U}$, a diagonal matrix of singular values ordered by magnitude $\boldsymbol{\Sigma}$, and the square matrix of right eigenvectors $\mathbf{V}^*$. The columns of $\mathbf{U}$ can be interpreted as modes of the physical system. If the dataset lies within a linear subspace, the eigenvalues decay exponentially, and the k first eigenvectors of $\mathbf{U}$ form the optimal reduced basis for the dataset

$$\tilde{\mathbf{U}} = \begin{bmatrix} | & | & & | \\ \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \dots & \boldsymbol{\psi}_k \\ | & | & & | \end{bmatrix} . \tag{3.1.3}$$

The reduced representation by this truncation reads

$$\tilde{\mathbf{X}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\mathbf{V}^* . \tag{3.1.4}$$

While the sole application of the SVD retains the original dataset, the dimensional reduction introduces a projection error

$$e_p(k) = \left\| \left(\mathbf{I} - \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T \right) \mathbf{X} \right\| . \tag{3.1.5}$$

As $e_p(k)$ increases with decreasing k, a trade-off must be made between the degrees of freedom in the reduced system and the desired accuracy. One approach to deciding for k is to set a certain reconstruction capability and choose the lowest k that satisfies the requirement.

3.2 Neural Networks

The research effort put into NNs grew rapidly since their advent. Due to the fast research progress and the problem-dependent nature of data-driven models, this report reviews only the very basic ideas. The topic is well explained on numerous websites, blogs and in publicly available books, the author warmly recommends [5, 11, 23]. The following sections start by introducing the fundamental idea behind NN architecture and training, before advancing to CNNs and the GNNs approaches demonstrated (Section 3.2.5 and 3.2.4).

3.2.1 Fully-connected / Vanilla Neural Networks

Biological findings inspired the historically first NN architecture, which consist of fully connected neuron layers, connected by learnable transformations. Each layer maps its input $\mathbf{X}$ of size $n \times d$ to output data $\mathbf{y}$ of size $m \times d'$. The following equations describe the problem for data with one feature, although they easily generalize towards higher dimension. The transformation is applied by (1) multiplying the input with a weight matrix $\mathbf{w}$ with dimension $m \times n$ and adding bias $\mathbf{b}$ of dimension m, and (2) applying a – generally nonlinear – activation function σ. Weights and biases constitute the free parameters of the model, calculating the output

$$\mathbf{y} = \mathbf{w}^T \mathbf{X} + \mathbf{b}\,. \tag{3.2.1}$$

In the early stages of NN research, the logistic or sigmoid function served as activation function, while over time rectified linear unit (ReLU) and the hyperbolic tangent showed outdated sigmoid in most applications [2]. At the time of writing, among a large number of activation functions, ReLU and exponential linear unit (ELU) are the most popular for regression tasks. However, the question of the best activation function remains unanswered at this time and may be problem-specific. Moreover, in general practice the identity is chosen as activation function of the last layer, mapping to the output.

In fact, a NN with only two layers corresponds to a Proper Orthogonal Decomposition basis [40]. **Deep NNS** contain at least one **hidden** layer between input and output. They can nonlinear combinations of the input at the cost of additional computational effort.

Theoretically, NNs with only one hidden layer can approximate continuous function up to any desired accuracy if enough hidden neurons are available [31]. In practice however, computational limitations may prohibit a sufficient number available neurons and, so

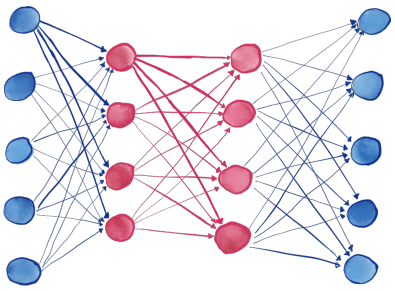


Figure 3.2.1: Fully-connected NN with four layers, of which two are hidden (pink) with five input and output neurons and four hidden neurons.

far, no method is guaranteed to find the optimal parameters for a specific network architecture. In view of limited computational resources deep NNs with fewer neurons per layer and greater depth may outperform shallow networks. They have been shown to reduce datasets to a extraordinary small latent space, du to their nonlinear combination of inputs [32].

The parameters that influence the flexibility of a network consist of the number of layers, the number of neurons per layer, the choice of activation function, and the training method. In the absence of concrete theoretical results, practitioners typically approach these questions through a mixture of expertise and experimentation.

The next section describes the state of the art for retrieving the network weights and biases for a specific architecture.

Training

The capability of NNs to represent relationships in high-dimensional and complex datasets stems from their ability to perform nonlinear mappings. However, this property comes with the drawback of having to find an optimal set of parameters for a high-dimensional nonlinear function, which is realized via numerical optimization of the respective function parameters.

This optimization requires an objective function (also called loss function) $\mathcal{L}$ to measure the performance of a set of network parameters with the object of minimization

$$\min_{\mathbf{w},\mathbf{b}} \mathcal{L}(\tilde{\mathbf{y}}, \mathbf{y})\,. \tag{3.2.2}$$

Most commonly in regression tasks, the mean squared error (MSE) over all samples

$$\mathrm{MSE}(\tilde{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^{m} (\tilde{\mathbf{y}}_i - \mathbf{y}_i)^2 \tag{3.2.3}$$

quantifies the loss between a prediction $\tilde{\mathbf{y}}$ and the original data $\mathbf{y}$. Random sampling of a normal distribution determines the initial guess of the network parameters [22, 26]. The exact properties of this distribution depend on the activation function, the types of layers, and the networks depth. Gradient-descent numerical optimization algorithms [41] iteratively attempt to find a global optimum of the problem's loss function $\mathcal{L}$ by following the direction of steepest descent of the objective function. Focusing on one layer, the prediction reduces to

$$\boldsymbol{h} = \sigma(\mathbf{w}^T \mathbf{X} + \mathbf{b})\,, \tag{3.2.4}$$

or component-wise

$$h_k = \sigma\left(\sum_{i}^{n} \mathbf{w}_{jk} \mathbf{X}_j + \mathbf{b}_k\right)\,. \tag{3.2.5}$$

Every NN layer applies linear combination of weights and biases before the activation function, thus the analytic derivative with respect to weights and biases follows from applying the derivative chain rule to Equation 3.2.4. The selected loss function quadratically depends on the prediction $\tilde{\mathbf{y}}$, therefore the chain rule enables the computation of the derivative of the loss function with respect to weights and biases. This property allows an efficient calculation of the derivative even for a nonlinear model. An intuitive explanation is to assign a fraction of the residual to each learnable parameter, referred to as **backpropagation** [1, 14]. In practical implementation, a computational graph tracks the recursive combination of input data, weights and biases [19].

In theory, backpropagation and gradient descent can perform their steps on the whole dataset. However, the computation is memory intensive, so today conventional NN training iterates the dataset assuming that a subset of samples approximates the gradient [1], which also allows parallelization.

The vanilla gradient descent algorithm performs one step in the direction of reduced residual (with stochastic or deterministic gradient)

$$\vartheta^{m+1} = \vartheta^m - \alpha \nabla \mathcal{L}(\Phi_{\vartheta^m}) \,. \tag{3.2.6}$$

The primary obstacle in gradient descent optimization is avoiding local minima, which requires careful selection of the learning rate (also known as step size) α and appropriate initialization [22, 26]. If the parameters θ^m correspond to a local minimum of the loss function, no directional derivative reduces the residual and the algorithm does not progress further. The learning rate crucially affects how prone an optimization problem is to local minima, where small value can trap the optimizer in small “sink“, whereas a large value may skip both local and global minima, and may even diverge.

The issues have been addressed by a series of optimization algorithms, mainly by including a momentum term computed as moving average of gradients. The idea is based on rolling balls storing kinetic energy as momentum, which enables them to overcome obstacles, if they collected sufficient momentum [43]. The probably most famous adaptor of the momentum method is the Adam [33] optimizer, which also inspired a number of modified versions [18, 35, 36].

Besides the risk of not performing well on a dataset, NNs can also perform too well – when they overfit to a dataset. **Overfitting** is when a model learns a structure that is not present in the data. It indicates the chosen model is too complex and relates to the variance-bias trade-off (confer [11], Section 2.2.2). The trade-off describes how simple models tend to have bias, but deterministic output for identical input data. Instead more complex models, i.e. with a large number of parameters, accurately approximate the data used for their fitting, however, slight changes in the training data can result in considerably different model parameters. Various regularization methods target the reduction, e.g., randomly dropping entries in the dataset [28], enforcing sparse weight matrices, e.g., by Lasso regularization [46], adding random noise to the input data [53].

To identify the performance of statistical models on datasets with unknown distributions, the dataset is randomly split into a training and a test set. While gradient descent and backpropagation only use the training set, the performance measurement only considers the unknown test set, quantifying the generalization.

Despite the massive amount of research effort put into NNs weight initialization and optimization, neural networks still suffer from an uncertainty whether the optimal parameters for a certain architecture have been reached.

3.2.2 Convolutional Neural Networks

Although this work does not apply CNNs, their success forms the baseline for the development and understanding of graph convolution (Section 3.2.3).

Vanilla neural networks 3.2.1 come with the severe limitations of overfitting to the training data and scaling the number of trainable parameters with the size of the input data. After their invention [34] CNNs fastly overtook the conventional design in image processing.

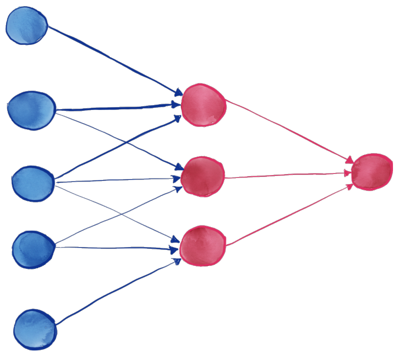


Figure 3.2.2: CNN with five input neurons and two convolutional filters of size three, resulting in one output.

As their name implies, they mimic a mathematical convolution of two signals f, g, which in continuous Euclidean space yields

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,\mathrm{d}\tau\,, \tag{3.2.7}$$

and instead a multiplication on discrete data

$$(f * g)[n] = \sum_{m=-\infty} f[n-m]g[m]\,. \tag{3.2.8}$$

CNN layers learn the second signal g as discrete convolution kernel. The support of this kernel can be chosen and a local support – i.e. smaller than the input data – guarantees local feature extraction. In fact, small kernel sizes, as 3×3 or 4×4, show best generalization capabilities, as local features can be combined to global features. The number of learnable weights for one convolutional filter equals the dimension of the filter and the weights are shared for all inputs. Thus CNN layers decouple the number of weights from the input size and enable processing large data sets on regularly spaced domains. The application of CNNs on **physical data** also yielded great success. A 3×3 convolutional layer on regularly structured physical data equals learning a finite difference stencil for the discrete Laplace operator. Lee and Carlberg demonstrated how CNNs successfully reconstruct a physical manifold for dynamical systems and learn the optimal finite difference stencil for given PDE solutions.

3.2.3 Graph Neural Networks

Graph Neural Networks expand the idea of NNs to spatially unstructured data and were first introduced by Scarselli [47]. The area of geometric deep learning changes rapidly due to a massive amount of research invested. Bronstein and others [7, 8] give a great overview of the state of the art at that time.

This type of network architecture operates on the geometric structure of a graph, i.e. the adjacency matrix defines the relation between vertices. NNs operating on structured data, e.g. image data, are a special case of GNN. One of the key features of this architecture is that the network is invariant to permutations of the vertex order.

After the success story of CNNs on image and physical data, researchers aimed to reproduce convolution on unstructured data. Three main mechanisms of GNN layers with increasing generalization evolved: (a) graph convolution, (b) graph attention, and (c) message passing, where the latter generalizes the former ones.

Introduction to graphs

A (discrete) graph consists of a finite set of vertices (or nodes) $\mathbf{v} \in \mathcal{V}$ connected by a set of edges $\mathbf{e} \in \mathcal{E}$. In a finite weighted graph, additionally, weights e^w are associated to each edge. The following paragraph summarizes the most important features of a graph, while Grady gives a more extensive introduction [24].

The most fundamental quantity to describe a graph is the adjacency matrix $\mathbf{A}$, a matrix of dimensions $N_\mathcal{V} \times N_\mathcal{V}$ taking the values 0

and 1. It indicates a directed edge from vertex A to vertex B with value 1 at matrix position $\mathbf{A}_{\mathrm{AB}}$, the row number corresponding to A and the column number corresponding to B. Values on the diagonal indicate self-loops on vertices. The adjacency matrix notation of an undirected edge consists of two entries, i.e., a directed edge from A to B and additional edge from B to A. If only undirected edges occur in a graph, the adjacency matrix is symmetric and the graph is called undirected. The weighted adjacency matrix $\mathbf{A}_w$ extends the adjacency matrix by containing the edge weight at the respective position instead of the value 1.
The degree matrix $\mathbf{D}$ of dimension $N_{\mathcal{V}} \times N_{\mathcal{V}}$ has only entries on its diagonal equivalent to the number of edges adjacent to a vertex and is calculated as $D_{ii} = \sum_j {e^w}_j$. Figure 3.2.3 displays an exemplary graph with six nodes connected by undirected and directed edges, its adjacency matrix corresponds to

$$\mathbf{A} = \begin{array}{c} \\ \mathrm{A} \\ \mathrm{B} \\ \mathrm{C} \\ \mathrm{D} \\ \mathrm{E} \\ \mathrm{F} \end{array} \begin{pmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} & \mathrm{F} \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} . \tag{3.2.9}$$

Analogous operators to derivatives on continuous functions have been developed, where the graph Laplacian $\mathbf{L}$ expands the Laplacian operator. It directly relates to the adjacency matrix and the degree matrix via

$$\mathbf{L} = \mathbf{D} - \mathbf{A} . \tag{3.2.10}$$

The eigenvectors of the Laplacian form a Fourier basis Ψ, thus representing the Laplacian as two orthogonal matrices and a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_i)$

$$\mathbf{L} = \Psi \Lambda \Psi^T \tag{3.2.11}$$

A graph represents a signal f as a vector of signal values f_i on its $N_{\mathcal{V}}$ nodes.

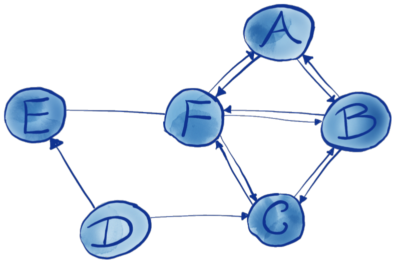


Figure 3.2.3: A graph with six vertices connected by undirected and directed edges.

Spectral graph convolution

The convolution of two signals or functions is defined as the integral of their product after reflecting and shifting one function. Its application allowed great progress in signal processing and CNNs, which motivated researchers to develop extensions to non-regular spaces, such as graphs. In practice, convolutions are calculated after a Fourier transform, as the convolution of two functions f and g is a multiplication of their Fourier transforms in the spectral domain ("Convolutional theorem", see e.g. [15]).

One way to generalize the Fourier convolution to a graph uses the convolutional theorem (confer [7], p.28, Equation (26)). Using the discrete version of an inner product on a graph, the convolution of two signals $(f * g)$ results in a multiplication of the eigenvectors of the Laplacian with the spectral representation of a signal $\hat{g}$

$$(f * g) = \Psi \text{diag}(\hat{g}) \Psi^T f \,. \tag{3.2.12}$$

Thus, convolution on a graph represents a signal in the basis of eigenvectors of the Laplacian of the graph. However, the Laplacian eigenfunctions are not unique, as they are always defined only up to a sign, and for multiple eigenvalues only up to an orthogonal transformation [7].

The first GNN convolutional layer followed directly from the spectral representation by Bruna [9] as a multiplication of the eigenvectors with learnable spectral multipliers Γ and the signal

$$g_j = \sigma\left(\sum_{j=1}^{q} \mathbf{\Psi}\mathbf{\Gamma}_{j,i}\mathbf{\Psi}^T f_i\right) . \tag{3.2.13}$$

As in Section 3.1, the order of the eigenvectors corresponds to the magnitude of their individual values, and the first eigenvectors relate to the smooth, low-frequency structures of the graphs. In most cases, the first k Laplacian eigenvectors carry the relevant features of an input signal and the remaining eigenvectors can be omitted.

This first definition of graph convolution comes with severe limitations. As aforementioned, the Laplacian eigenfunctions and consequently eigenvectors are not unique and especially the higher-frequency eigenvectors can be unstable, and their computation is expensive. If the graph lies on a manifold, i.e. a non-planar surface within a volume, the result of the convolution depends on the spatial position and does not generalize to new geometries. In addition, the number of learnable parameters for the layer defined in (3.2.13) has the same order of magnitude as the number of input nodes and thus still scales with the graph.

The localized feature extraction property of CNN filters with small support can be translated to GNNs by using the Laplacian as local filter. As a symmetric real-valued matrix, the Laplacian can be decomposed into an orthonormal basis and a real-valued diagonal matrix $\mathbf{L} = \Psi\Lambda\Psi^T$ such that a polynomial on the Laplacian is equivalent to a polynomial on its eigenvalues

$$\mathbf{L}^i = \Psi\Lambda^i\Psi^T . \tag{3.2.14}$$

Since the Laplacian only acts on a 1-hop environment of each node (i.e. within one edge), its i-th power operates maximally on an i-hop environment. A linear combination of Laplace polynomials

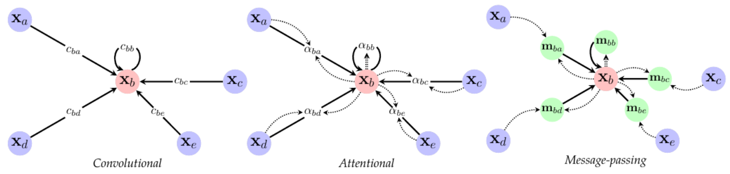


Figure 3.2.4: Main approaches to graph convolution, from [7] p. 78.

preserves the property creating the localized learnable filter γ

$$\gamma(\mathbf{L}) = \Psi\gamma(\Lambda)\Psi^T \tag{3.2.15}$$

$$\gamma(\lambda) = \sum_{j=0}^{r-1} \eta_j \lambda^j \tag{3.2.16}$$

with polynomial coefficients $\eta \in \mathbb{R}^r$.

Graph layers

As mentioned above, the state of the art in (graph) neural networks is currently expanding at a rapid pace due to extensive research. Most popular architectures today follow three main principles: (a) convolution, (b) attention, and (c) message passing [7]. After a brief explanation of these key ideas, the following paragraphs give a detailed explanation of the network layers used in the result section.

Thus, this work limits its explanation to a brief summary of the key ideas, before a detailed explanation of the network layers used in the results section follows.

Spectral Graph Convolution (3.2.3) is directly inspired by the convolutional approach. An aggregation function $\oplus$ collects the features $\mathbf{x}_v$ in the direct neighborhood $\mathcal{N}_u$ of node u, which are possibly transformed by a function Υ and weighted by the edge weight $e^w{}_{uv}$. The aggregation function needs to be permutation-invariant, i.e. does not depend on the order of inputs. Common choices include the summed or mean value. A learnable function Φ computes the output features $\mathbf{y}_u$ at node u from the input node features $\mathbf{x}_u$ and the aggregated neighborhood features.

In the simplest (non-deterministic) case the layer collects the – possibly transformed – inputs from neighboring edges weighted by constant edge weights e^w. A learnable fully-connected layer represents Φ with $\dim(h_u) \times \dim(x_u)$ shared weights, resulting in scalability comparable to CNNs:

$$\mathbf{y}_u = \Phi\left(\mathbf{x}_u, \oplus_{v\in\mathcal{N}_u} {e^w}_{uv} \Upsilon(\mathbf{x}_v)\right) \tag{3.2.17}$$

Graph **attention** networks [6, 51] extend the convolutional approach with a learnable self-attention $\mathbf{a}(\mathbf{x}_u, \mathbf{x}_v)$ mechanism dependent on the node features of the node itself and its neighborhood:

$$\mathbf{y}_u = \Phi\left(\mathbf{x}_u, \oplus_{v\in\mathcal{N}_u} \mathbf{a}(\mathbf{x}_u, \mathbf{x}_v) \Upsilon(\mathbf{x}_v)\right) \tag{3.2.18}$$

The most general notion – message passing [3, 20] – allows arbitrary messages to be passed along edges:

$$\mathbf{y}_u = \Phi\left(\mathbf{x}_u, \oplus_{v\in\mathcal{N}_u} \Upsilon(\mathbf{x}_u, \mathbf{x}_v)\right) . \tag{3.2.19}$$

Thus, the convolutional (Equation 3.2.17) and attentional (Equation 3.2.18) layers represent special cases of message passing (Equation 3.2.19). As the modeling flexibility increases, so does the number of learnable parameters and thus the complexity of successful training.

3.2.4 ChebNet

The Chebyshev graph convolution [17] was among the first adaptations of the computationally efficient filter representation by the graph Laplacian (Section 3.2.3). The Chebychev polynomials gave their name to the method. They form an orthogonal basis with respect to the inner product

$$\int_{-1}^{1} T_i(x)T_j(x)\frac{\mathrm{d}x}{\sqrt{1-y^2}}\,. \tag{3.2.20}$$

They are generated by the recurrence relation

$$T_j(\lambda) = 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda) \tag{3.2.21}$$

$$T_0(\lambda) = 1 \tag{3.2.22}$$

$$T_1(\lambda) = \lambda\,. \tag{3.2.23}$$

Since the relation holds only on the interval $[0, 1]$, the Laplacian eigenvalues need to be rescaled by

$$\tilde{\mathbf{L}} = 2\lambda_n^{-1}\mathbf{L} - \mathsf{I}\,. \tag{3.2.24}$$

In this work, the PyTorch Geometric implementation [19] of Cheb-Conv was used. The motivation for choosing this method was its control over the size of the local filter, by the maximal polynomial exponent.

3.2.5 Graph attention

As already described in Section 3.2.3 GATs extend convolutional networks by a learnable attention mechanism. Thus, the weighting of each node (see Figure 3.2.4, middle) is not constant anymore, but learnable. In contrast to the convolutional approach [54], GATs can represent signals outside the span of the eigenvectors of their Laplacian. The method originally proposed in [51] introduced attention coefficients, which learn the importance of the features of neighboring nodes to the node itself. The initial method suffered from poor scaling, if one node's features exceed the values of the others, which Brody solved by rescaling [6]. With this improvement, the learnable feature importance results to

$$e(\mathbf{x}_i, \mathbf{x}_j) = \sigma\left(\mathbf{a}^T\left(\mathbf{W}_a\mathbf{x}_i, \mathbf{W}_a\mathbf{x}_j\right)\right)\,, \tag{3.2.25}$$

with learnable attention weights $\mathbf{a} \in \mathbb{R}^2 d', \mathbf{W}_a \in \mathbb{R}^{d' \times d}$, where d and d' are the number of features in the input and output, and leaky rELU is proposed as activation function. The output features follow from the aggregation in the neighborhood $\mathcal{N}_i$

$$\mathbf{y}_i = \sigma\left(\mathbf{W}_a, \operatorname{softmax}(e(\mathbf{x}_i, \mathbf{x}_j))\mathbf{x}_j\right) . \tag{3.2.26}$$

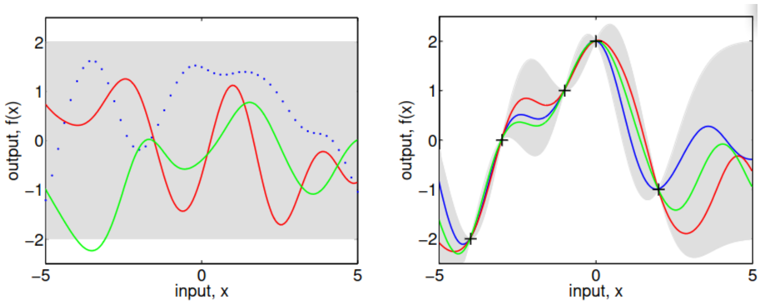
The PyTorch [19] implementation *GATv2Conv* [6] was used.

3.3 Gaussian Process Regression

One line of separation for data-driven models runs between parametric and non-parametric models. While the previous models, Proper Orthogonal Decomposition and NN, have explicit parameters, GPR comes from the group of non-parametric models. The method assumes that the dataset follows an underlying multivariate Gaussian distribution, represented by Gaussian Processes (Figure 3.3.1, left). An extensive explanation can be found in Rasmussen et. al. [45].

The key idea states that samples that are close in the input space, should lead to close predictions in the output. The realization follows by a kernel function, which models the covariance among the different variables and depends only on distance of the input. The predictive model is derived by conditioning the distribution on the input samples (called kriging), such that (1) samples in the training set are exactly predicted, and (2) the predictive output for the test set consists of an estimated mean and standard deviation (Figure 3.3.1, right).

GPR developed to a popular regression method, as it can model even complex functions, requiring few model assumptions and hyperparameters. In addition, the predictive variance made GPRs a popular tool for error estimates.



(a), prior (b), posterior

Figure 3.3.1: The prediction of a GPR with a prior of Standard Gaussian Processes before (left) and after (right) conditioning, from [45], p.15.

In this work, the prior was chosen as a standard Gaussian process with a Gaussian kernel as the covariance model. This kernel predicts an exponential decay of the correlation with increasing input distance. The only parameter of freedom of this kernel is its length scale, i.e., the rate of exponential decay with increasing distance.

4 POD-based model for shrinkage-warpage prediction

4.1 Problem statement

Assume $\mathcal{V}$ maps from a parameter vector $\boldsymbol{\mu} \in \mathbb{R}^n$ to a temperature field Θ on a domain Ω. This temperature field enters a shrinkage-warpage simulation, where the operator $\mathcal{W} : \Theta \to U$ maps the initial temperature field to a displacement field. For a computationally fast prediction, the composed operator $\mathcal{V} \circ \mathcal{W}$ shall be regressed. A dataset of N_{samples} describes the process discretely as

$$X = \{\mu, \mathcal{V} \circ \mathcal{W}(\mu\}_{i=1}^{N_{\text{samples}}} . \tag{4.1.1}$$

This poses the minimization problem for the regressed operator $\tilde{\mathcal{V}}$

$$\min \sum_{i}^{N_{\text{samples}}} \|\tilde{\mathcal{V}} - \mathcal{W} \circ \mathcal{V}(\mu_i)\| . \tag{4.1.2}$$

4.2 Non-intrusive POD-GPR model

As elaborated in Section 3.1, the POD method determines the optimal linear subspace to a dataset. While the coefficients of the reduced basis vectors for samples within the data set result from a multiplication of the eigenvalues and the right eigenvectors, an additional step is required to start the prediction.

The mathematically most rigorous method solves the system of the reduced basis functions and the system matrix [27, 52] and under certain prerequisites even provides an error-bound. However, this approach requires additional knowledge of the system matrix and thus violates the requirement of maintaining non-intrusive, i.e. solely data-driven.

The non-intrusive approach requires a regression model between the input parameters and the coefficients of the basis vectors, and any model connecting the two sets can serve the purpose. The optimal choice depends on the dataset and its complexity. Popular methods include radial basis functionss (RBFs) [12], NNs [23], GPR [45], as applied in [13, 16], and quadratic polynomial regression [48, 49]. The demands on robustness, flexibility, and computational cost determine the optimal regressor choice for a specific dataset.

Considering the strong scaling of NNs with data availability and the limitations of polynomial regression, GPR was selected as regression model for the task at hand. Model generation consists of (1) POD generation where the eigenvectors are truncated up to a prescribed projection error (Equation 3.1.5), (2) regressing the weights in the reduced basis with GPR. The prediction capability of GPR regressed coefficient for new input parameters $\boldsymbol{\mu}_{i>N_{\text{samples}}}$. Following the results of [16], the input data undergoes mean removal. Since the regression target consist of only one quantity, feature scaling becomes obsolete.

A reconstruction performance on the original dataset determines the number of preserved basis vectors k, i.e. a projection error (Equation 3.1.5) of 1% corresponds to a 99% reconstruction of the original dataset. As no clear theory on an optimal regression model could be found, this work places particular focus on the approximation error introduced by projection and regression. Clearly the projection error (Equation 3.1.5) decreases with an increasing number of preserved eigenvectors, however, this also expands the dimension of the output in the regression problem. The author estimates re-

Bounds	mean_x	mean_y	σ_x	σ_y
Min.	0.	0.	0.0568	0.0538
Max.	1.	0.8	0.9958	0.9987

Table 4.1: Geometric bounds for dataset generation.

gression performance will decay with an increasing number of preserved eigenvectors, and poses a trade-off between the projection error and the regression error. This assumptions will be examined by investigating the performance of models with different thresholds on the projection error on the training dataset.

4.3 Dataset

The data consists of the samples described in Equation 4.1.1 on a double T-shaped domain. All samples share the same discretization of the domain without transformations, consisting of 296 vertices connected by triangular elements, and a set of N_{samples} samples has been chosen.

4.3.1 Parametrization of the input

Considering the application of injection molding, different input temperate fields share their maximum close to the flow inlet with smooth transitions to low-temperature regions. Bivariate Gaussian distributions emulate varying inlet positions by shifting their mean and covariance, such that the parameter consists of four parameters describing each field.

Latin hypercube sampling [39] generated $N_{\text{samples}} = 100$ samples within the bounds stated in Table 4.1.

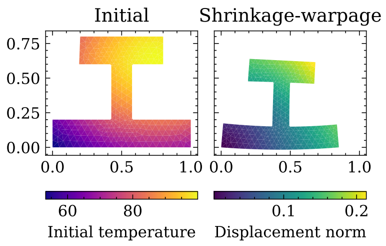


Figure 4.3.1: Sample instance from dataset.

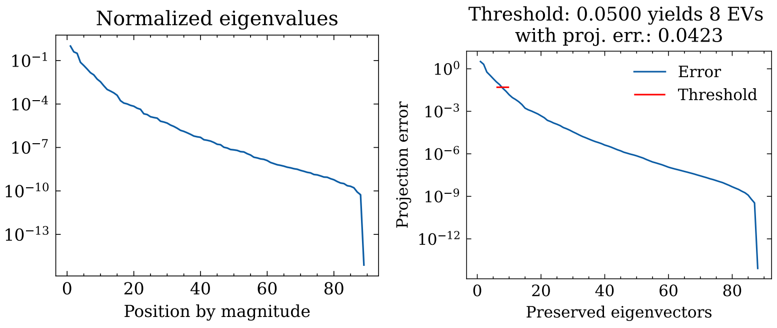
4.3.2 Boundary value problem

A FEM simulation using compressible Neo-Hookean material law (Section 2.2.1) is used to calculate the respective stress field and deformation. Section 4.3.2 states the respective boundary value problem for initial temperature and displacement.

The boundary conditions prevent rigid body movement by fixing the position of the bottom-left corner of the geometry and the vertical degree of freedom of the bottom-right corner. The material parameters were chosen as thermal expansion coefficient $\alpha = 2.5 \cdot 10^{-3}$ and poisson ratio $\nu = 0.25$. Figure 4.3.1 displays a sample from the dataset.

4.4 Results

The first model step consists of applying Proper Orthogonal Decomposition and determining the number of preserved left eigenvectors in the basis. A linear combination of the truncated eigenvectors can only represent the dataset sufficiently, if the eigenvalues' magnitude decays exponentially. Figure 4.4.1a displays the eigenvalues, sorted by magnitude and normalized by the maximal value. In the logarithmic scaling of the y-axis, exponential decay corresponds to a decreasing line, which the decay in Figure 4.4.1a satisfies.



(a) The magnitude of eigenvalues, normalized by 1.

(b) The corresponding projection error for different numbers of preserved eigenvectors.

Figure 4.4.1: The Proper Orthogonal Decomposition yields exponentially decreasing eigenvalues of the dataset (left) and a projection error with similar behaviour (right).

The number of preserved basis vectors is a design choice. In this work, the projection error (Equation 3.1.5) serves as performance measure, as it quantifies the difference between the original dataset and the dataset projection on the selected basis in the L_2 norm. Figure 4.4.1b displays the projection error, behaving analogously to the eigenvalue magnitude. Demanding a 95% reconstruction of the original dataset results in a projection error of 4.23% and 8 basis vectors.

The approximation error for a given sample measures the differences between the prediction $\tilde{\mathbf{y}}$ and the FEM solution $\mathbf{y}$ in the L_2 norm.

$$e = \|\tilde{\mathbf{y}} - \mathbf{y}\|_{L_2} \,, \tag{4.4.1}$$

while normalization by the L_2 norm of the FEM solution delivers the relative approximation error

$$e_{\mathrm{r}} = \frac{\|\tilde{\mathbf{y}} - \mathbf{y}\|_{L_2}}{\|\mathbf{y}\|_{L_2}} \,. \tag{4.4.2}$$

If the reference solution $\mathbf{x}$ to a problem is known, e.g., from a test set, the sample-wise projection error can be calculated as

$$e_p(k) = \left\| \left(\mathbf{I} - \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T \right) \boldsymbol{x} \right\| \,, \tag{4.4.3}$$

and can be interpreted as a lower bound for the approximation error, if there was an ideal regression model. This notion introduces a regression error, as the difference between the approximation error and the projection error

$$e_{\text{reg}} = e - e_{\text{r}} \,. \tag{4.4.4}$$

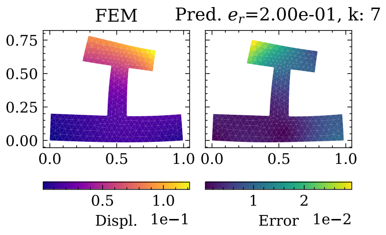
Note that this additive split is an assumption and different relations between the projection and regression error may exist.

Figure 4.4.2 collects the predictions with the largest relative approximation error in the the test set. The relative error ranges between $1.63 - 2.00 \cdot 10^{-1}$ and the number of used basis vectors increases moderately from 7 to 11. The maximal point-wise error appears in the top-left corner for all models.

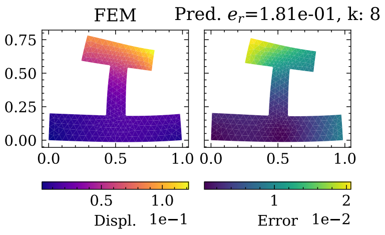
In the exemplary instance, the additional basis vectors seem to enhance the prediction accuracy, although the regression problem maps 4 to 11 instead of only 7 coefficients. Figure 4.4.3 summarizes the effect on all test instances. The blue bars indicate the projection error on the selected basis ($e_p(k)$) and the green bars symbolize the difference to the approximation error e. The overall error lies in the same range for all thresholds sets, while the projection error decreases with the growing basis. This indicates the regression contributes the largest part to the approximation error.

4.5 Discussion and Outlook

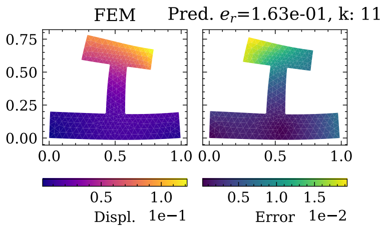
The dataset includes hyperelastic material behaviour and large geometric deviations in the initial conditions exceeding realistic design



(a) $e_p(k) \leq 0.1$.

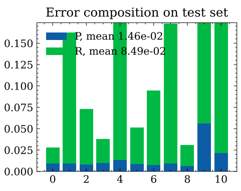


(b) $e_p(k) \leq 0.05$.

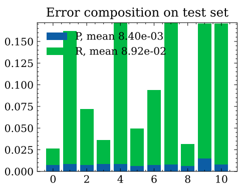


(c) $e_p(k) \leq 0.01$.

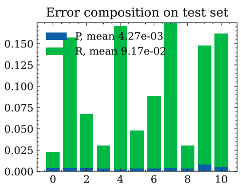
Figure 4.4.2: The samples from the test set with the highest relative approximation error for different number of basis vectors, resulting from different demanded projection accuracy.



(a) $e_p(k) \leq 0.1$.



(b) $e_p(k) \leq 0.05$.



(c) $e_p(k) \leq 0.01$.

Figure 4.4.3: Approximation error on the test set, with the division into error on projection (P) and regression (R) for different projection accuracy.

deviations in injection molding. Despite this variation, POD managed to projection the dataset to a subspace of only roughly 10% of the original degrees of freedom even for a high reconstruction accuracy. The projection error in the test set matches the order of magnitude in the test set, which promises generalization. Thus, a linear subspace seems suitable for approximating the problem.

In contrast, the overall relative approximation error is around 15%. This indicates that the GPR does not satisfactorily recover the relationship between parameters and basis function coefficients. Interestingly, the high regression error does not show sensitivity to the dimension of output variables, i.e., the number of basis vectors. Thus, the GPR might be an unsuitable regression model for the task. As GPR by definition exactly predicts its training data, the author estimates overfitting to the training data and proposes a more deterministic model such as RBF.

5 GNN-based shrinkage-warpage prediction

Under realistic circumstances, a preceding simulation passes the initial temperature field to the shrinkage-warpage calculation, which – in general – does not obey a closed-form parametrization, as used in Section 4.2. A realistic data-driven model for shrinkage-warpage maps from an arbitrary initial temperature field to a displacement field.

In addition, the shape of the product affects the displacement result and design optimization aims at iterating multiple geometric designs (Section 1). Ideally, the data-driven model is applicable to a group of related geometries.

5.1 Problem statement

The aim stated in the former paragraph can be formally described by operator regression. Assume the operator $\mathcal{W} : \Theta \to U$ maps the input temperature field Θ on a domain Ω to the deformation field U. The operator shall be regressed from a collection of training samples of different initial temperatures on a collection of geometric domains $\{\Omega_i\}_{i=1}^{N_{\text{Geom}}}$, leading to the dataset

$$X = \{\theta_n, \Omega_n, \mathcal{W}(\theta_n, \Omega_n)\}_{n-1}^{N_{\text{samples}}} . \tag{5.1.1}$$

We denote the true (unknown) operator as $\mathcal{W}$ and its data-driven reconstruction as $\tilde{\mathcal{W}}$. Operator regression aims at minimizing the least squares error between the output of the regressed operator $\tilde{\mathcal{W}}$ on the given dataset

$$\min \sum_{n=1}^{N} \left\| \mathbf{X}_n - \tilde{\mathcal{W}}(\theta_n, \Omega_n) \right\| . \tag{5.1.2}$$

5.2 Data-driven model

Most reduced order models (ROMs), e.g. the model presented in the last chapter 4.2, extrapolate on a fixed geometric domain, or on a parametrization of the fixed domain with the same number of degrees of freedom. In contrast, GNNs learn the dataset with respect to its edge weights and promise generalization across new graphs. This capability motivated the choice of a GNN as data-driven model for $\tilde{\mathcal{W}}$.

While GNNs achieved great success in learning flow simulations (Section 3.2.5), to the author's knowledge, attempts to regress stationary elasticity are limited to the works of Löetzsch [38] and Gladstone [21]. The influence of the boundary condition on the complete domain is a key feature of elliptic problems. Gladstone [21] finds an unsatisfactory performance of the [42] architecture on their problem and speculate the local message passing fails at transmitting boundary condition information fast enough. As a remedy, they add new edges between random nodes within the domain. Instead Lötzsch et. al. [38] investigate multiple existent GNN architectures and find ChebConv performs best on their electro-magnetic example problem.

Given these results, the following sections investigate the performance of GAT and ChebConv as regression models $\tilde{\mathcal{W}}$. The relation to spectral graph convolution 3.2.3 motivates the choice for ChebConv with approximating global information passing by including up to the fifth exponential of the **L**. In contrast, overall GATs still outperform ChebConv with their additional flexibility.

Following the general state of the art [21, 38, 42], each model consists of

1. A GNN-based processor of one or multiple layers
2. A node encoder from the input features to number of hidden neurons in node encoder (NHN) features
3. A node decoder from NHN encoded node features to the output features
4. Only for GATs: An edge encoder from the edge attributes to number of hidden neurons in edge encoder (NHE) encoded edge features .

The goal of regression is to minimize the least squares error. However, the minimum of the MSE and the least squares error are identical. To avoid unnecessary backpropagation steps, the regression models's loss $\mathcal{L}$ function is selected as

$$\sum_{n=1}^{N_{\text{samples}}} \mathcal{L}(y_n, \tilde{\mathcal{W}}(\theta_n, \Omega_n) = \sum_{n=1}^{N_{\text{samples}}} \text{MSE}\left(y_n, \tilde{\mathcal{W}}(\theta_n, \Omega_n)\right) \ . \quad (5.2.1)$$

The following sections describe the generated dataset and the performance of the selected models.

5.3 Dataset

As the initial temperature field obeys the physical laws of the heat equations, it seems reasonable to assume continuity with respect to the geometric position. To cover a maximal span of initial temperature field, a random model is used for their generation. Samples from a GRF follow a Gaussian distribution while ensuring correlation in a neighborhood of length-scale l. Small values of l yield high-frequency white noise, whereas large values result in a smooth

	Train set	Test set	Total
Number of samples	179	45	224
Number of geometries	23	6	28

Table 5.1: Split of training and test instances.

transition of values. For deformations to occur, l needs to be larger than two element lengths, but smaller than the maximal length of the geometry at hand.

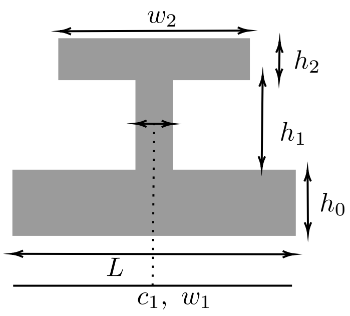
The double T shape from Section 4.2 is used again as an example, this time also adding geometric modifications. The parametrization displayed in Figure 5.3.1a defines an instance by the horizontal length (L) and vertical length (h_0) of the bottom box, the horizontal (w_1) and vertical (h_1) length of the middle box along with its horizontal center of mass (c_1), and the horizontal (w_2) and vertical (h_2) length of the top box. Although not relevant for the shape itself, the characteristic length of the triangular discretization of the shape (Δm) adds another degree of freedom to the data generation.

Latin hypercube sampling [39] generated 28 different geometries, visualized in Figure 5.3.1b within the parameter ranges displayed in Table 5.2. On each geometry, the FEM computed solutions for 8 different initial temperature field generated by GRF with parameter $l = 0.3$. Boundary conditions, material law and parameters correspond to the problem presented in Section 4.2 The resulting 244 samples were divided into training set (80%) and test set (20%) (Table 5.1), where the shapes Ω_i in the training set and test set are distinct.

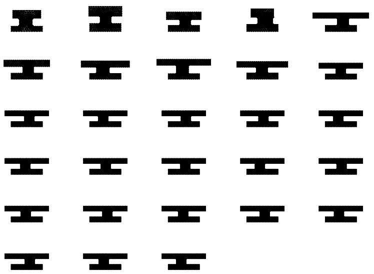
Figure 5.3.2 displays the initial temperature distribution (left) and the resulting deformed shape (right) for one sample instance.

The input data contains the following features for each node

1. Spatial position
2. Value of initial temperature



(a) Geometry parametrization of a double-T shape.



(b) Distinct geometric shapes within the dataset.

Bounds	N_{nodes}	L	w_1	w_2	h_0	h_1	h_2	Δm	c_1
Min.	224	1.0	0.1500	0.3588	0.1500	0.2000	0.1500	0.1000	0.4157
Max.	473	1.0	0.2371	0.8986	0.2402	0.2894	0.2972	0.1988	0.5994

Table 5.2: Geometric bounds for dataset generation, the parameters are represented in Figure 5.3.1a.

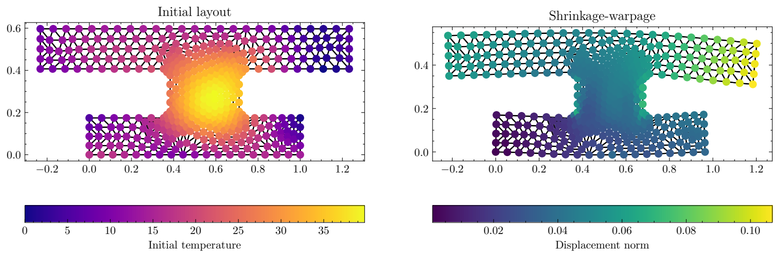


Figure 5.3.2: Sample instance from dataset

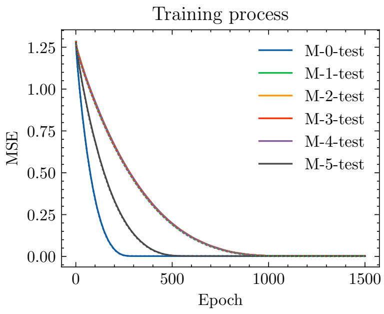
3. Boolean, whether the node lies on the geometric boundary $\partial\Omega_i$
4. Distance to the next node on a geometric boundary (0 if the latter Boolean is True)
5. Boolean, whether the node is subject to a Dirichlet boundary condition
6. Distance to the next node subject to a Dirichlet boundary condition

vertex position, The data is scaled feature-wise to the interval $[0, 1]$, where the bounds are extracted only from the training split. Edge weights (for ChebConv) correspond to the L_2 norm of an edge between two nodes, edge attributes (for GAT) additionally contain the distance vector itself.

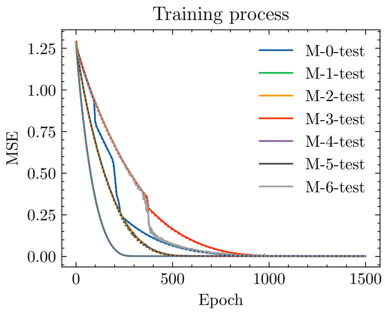
5.4 Results

The performance of NN-based models on a specific dataset may largely vary under the choice of different hyperparameters (see Section 3.2.1). A number of random parameter combinations from predefined options serve as an estimate of the model's sensitivities to its architecture. Table A.1 and A.2 list the choices for GAT and ChebConv, which where were based the findings in [38].

The training for each network architecture was performed for 1500 epochs using the Adam optimizer [33] with a constant learning rate of 10^{-4} and $\beta = (0.9, 0.999)$. Figure 5.4.1 displays the test error (solid line) and training error (dotted) for ChebConv (left) and GAT (right) processing units. The final test error with the ChebConv layers ranges from $2.019 - 2.619 \cdot 10^{-3}$ (Table A.2). All models smoothly and monotonically decrease the loss during training, although Figure 5.4.1a reveals how a subset of models converge faster than others. The GAT model yields a lower final loss between $1.887 - 2.194 \cdot 10^{-3}$ (Table A.1) with an equal order of magnitude. The decrease of the loss functions occurs monotonically



(a) ChebConv architecture



(b) GAT architecture

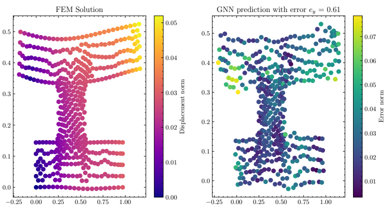
Figure 5.4.1: Evaluation of the test and training error during parameter optimization. The models correspond to Table A.1 and Table A.2. Solid lines display test error and dotted lines training error.

(Figure 5.4.1b), but less smooth than in the previous examples. Importantly, for all tested model, the lines of test error and train error run very close or align, which is an indicator of generalization.

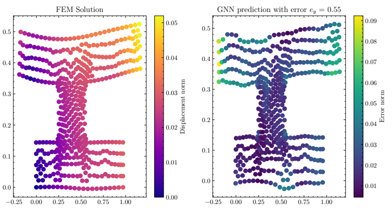
Although the MSE measures the loss during training, ultimately the relative error between the regressed solution and true solution determines the performance

$$e_{\text{rel}} = \frac{||y - \tilde{y}||}{||y||} \,. \tag{5.4.1}$$

Since scaling transformed the data to smaller than one, the relative error compares unfavorably to the MSE. Figures 5.4.2a and 5.4.2b show predictions for the sample sample with the best-performing models from the previous hyperparameter exploration (Table A.1, Table A.2). The relative error in the predictions is 0.61/0.55 with point-wise values up to 0.09. While the ChebConv prediction yields the lower overall error, point error fluctuate among neighboring nodes and does not deliver a smooth deformation field. In contrast,



(a) ChebConv



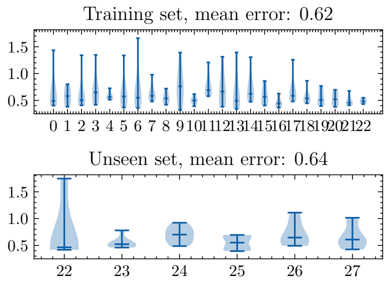
(b) GAT

Figure 5.4.2: A prediction on an unseen test sample with the true solution (left) and prediction (right).

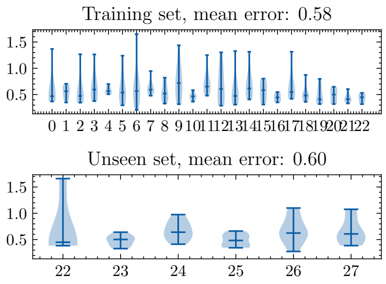
the GAT solution predicts smooth deformations, deviate less form the initial configuration and underestimate this sample.

Generalization across multiple geometries initially motivated the choice of GNNs. For an estimate of its success, we summarize the relative error on each geometic shape individually. Figure 5.4.3 displays the error distribution, its minimum, maximum and median (horizontal line) on each shape in the train set (top) and test set (bottom) with the ChebConv (left) and GAT model (right). Note that the samples in geometry number 22 have been split, such that it appears in both sets. Among the geometries significant outliers to the top are present and the distribution show significant variance within each geometric shape. The median error, however, is roughly constant within the geometries, even the unseen ones. No huge differences between the two prediction models (left and right) are visible.

We caution that although the training process appeared to be convergent, the models were fitted to only 179 sample instances with 224 to 473 nodes with multiple output features per instance.



(a) ChebConv



(b) GAT

Figure 5.4.3: Each violin plot displays the relative error of the sample of one shape as well as their minimum, maximum and median (horizontal lines).

5.5 Discussion and Outlook

During training, the test losses for the different layer architectures reached the same order of magnitude, while the GAT model slightly outperformed the ChebConv model. This contradicts the authors expectations, as the flexibility of the architecture and the edge encoder also increased the number of trainable parameters. A reason for the outperformance could be the availability of the edge attributes, which explicitly provide the spatial relation between two nodes.

It is remarkable how the results do not show signs of overfitting, although one sample instance contains more degrees of freedom than the number of instances passed to training. As the overall prediction error remains intolerably high, the results do not yet allow judgments on the generalization capabilities. However, a resemblance in the error distribution of test set and train set permits optimism. As found in numerous works related to NNs and reported in previous work on physics prediction with GNNs ([21, 38, 42], this kind of model improves with a large amount of training data, which motivates investigations on a larger dataset.

6 Conclusion

Injection molded products come in a wide range of shapes tailored to different applications and design goals. In addition, the behavior of polymer materials varies significantly between crystalline, semi-crystalline, and amorphous forms, as well as within each group. Given the complexity of these physical processes, it's not easy for engineers to determine the optimal shape and process parameters for a desired design. Design optimization algorithms can streamline this process by increasing efficiency and effectiveness.

This work targets the development of data-driven models, which predict the shrinkage-warpage of a product for new initial conditions. Because of the widely varying objectives and materials, the generalization capability of the model was emphasized at the price of more complex models.

The first **POD-based model** (Section 4.2) predicts solutions within a linear subspace of the given dataset with regression. The Proper Orthogonal Decomposition projection reached a reconstruction accuracy of 1% to 10% while reducing the number of degrees of freedom by around 90%. Even though the current model for initial temperature distribution (bivariate Gaussian) is highly simplified, the low projection error in both training and test set indicates a linear combination of eigenvectors can successfully represent the occurring deformations.

While the predicted deformations visually correspond to the reference solution, the regression of coefficients causes a considerable inaccuracy. In the future, this may be circumvented by (1) dropping the non-intrusive character of the model and solving the problem in the reduced basis, (2) investigation of different regression methods between input parameters and basis coefficients.

Overall, the linear base model gave promising results on a limited dataset size for a fixed geometric shape. Proper Orthogonal Decomposition-based models can also account for shape changes that affect only the position of vertices and not the connectivity of the discretization. In this case, the geometric transformation must appear in the input parameters. The limitation to discretization-consistent parameterized geometric modifications motivated the second GNN-based model.

Applying NN architectures allows for complex nonlinear bases with more flexibility, but brings greater risk of overfitting the dataset. In Section 5.2, the performance of NN architectures on a dataset is evaluated. The model inherently allows for (1) arbitrary input fields, and (2) different graphs as input domains. The loss function decreases monotonically during training to the same order of magnitude for all architectures and hyperparameter combinations tested. The overall prediction error e remains unacceptably large at around 0.62. However, it is noteworthy that all models predict the test and training sets with roughly the same error, even for new geometries. Although the training set contains only 144 instances with about 200 vertices each, the results do not indicate overfitting.

Two strategies are proposed to reduce the prediction error: (1) pass more training data, (2) pass additional physical information to the network. Considering the size of the datasets in analogous attempts [21, 38], it seems reasonable to assume that a larger dataset could reduce the approximation error. For the latter choice, physical knowledge can be transferred to the network as a residual term [44]. However, this approach requires the explicit implementation of a constitutive law and the additional need to weight residual and data-induced loss. In contrast, structure-preserving models follow the laws of exterior calculus and can satisfy energy conservation without an explicit constitutive model [25]. In future research, purely data-driven models will be compared with the structure-preserving model.

7 Acknowledgement

My stay in Albuquerque has been a blast, both professionally and on a personal level, thanks to the contribution of so many people to whom I would like to pay tribute.

First of all, I would like to thank my host, Rekha Rao. She made all the organisational efforts to make the stay possible and continuously supported my research and well-being. In the same way, I am very grateful to have had my supervisor Stefanie Elgeti at my side. Not only did she arrange the stay, but she also gave me all the flexibility I needed and was a great help with funding and all organisational matters. Thank you to Nat Trask for hosting me at CSRI and being a great mentor both online and while visiting UPenn. In addition, Dan Bolintineanu was a great support both organisationally and personally.

I received incredible hospitality at CSRI. Special thanks go to Elise Walker, Anthony Gruber, Jonas Actor, Pete Bosler, Eric Cyr and the management. I also felt very much welcomed into the GOMA Group, where Weston Ortiz and Helen Cleaves, in particular, continued to support me as I worked.

In addition to the work in the laboratory, I also had the opportunity to explore the culture of New Mexico. I am very grateful to my climbing crew – Austin, Maggie, Raven, Nick and Marshall – for making me feel at home away from home. In addition, I would like to thank – my roommates James and Jack, Marsaya, Mark, Cleveland and Veena – for the good times we have had and for your constant support.

I am very grateful for this opportunity, and I would like to thank the Sandia National Laboratories, the Marshall Plan Foundation, and the Vienna University of Technology for making this stay possible.

A Appendix

A.1 Hyperparameters

Model	batch norm (BN)	batch size (BS)	dropout (D)	numbers of layers of edge encoder (LEE)	numbers of layers of node encoder (LNE)	numbers of layers of processor (LP)	NHE	NHN	number of hidden neurons in processor (NHP)	$\mathcal{L} \cdot 10^{-3}$
M-1	F	16	0.05	1	1	1	16	64	64	2.250
M-2	T	4	0.05	0	1	3	8	32	32	**2.019**
M-3	T	8	0.01	1	2	2	8	32	32	2.066
M-4	T	16	0.10	0	1	2	8	64	32	2.270
M-5	T	4	0.05	1	1	1	8	32	64	2.025
M-6	F	8	0.05	0	2	1	8	32	32	2.112
M-7	F	16	0.01	1	2	3	8	64	64	2.619

Table A.1: Hyperparameter sweep for GATs framework. The rightmost column indicates the test loss after 1500 training epochs, see glossary for an explanation of the hyperparameter abbreviations.

Model	BS	D	LNE	LP	NHN	NHP	$\mathcal{L} \cdot 10^{-3}$
M-1	4	0.01	2	4	32	64	**1.887**
M-2	16	0.01	1	3	64	64	2.160
M-3	16	0.10	2	4	64	32	2.175
M-4	16	0.05	1	3	32	64	2.194
M-5	16	0.10	2	4	64	64	2.200
M-6	8	0.10	2	3	32	32	2.131

Table A.2: Hyperparameter sweep for ChebConv framework. See glossary for label explanation. As the architecture only allows scalar edge weights, edge encoding does not apply.

Bibliography

[1] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4):185–196, June 1993. ISSN 0925-2312. doi: 10.1016/0925-2312(93)90006-O. URL https://www.sciencedirect.com/science/article/pii/092523129390006O.

[2] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding Deep Neural Networks with Rectified Linear Units. *CoRR*, abs/1611.01491, 2016. URL http://arxiv.org/abs/1611.01491. arXiv: 1611.01491.

[3] Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018. URL https://arxiv.org/pdf/1806.01261.pdf.

[4] G Berkooz, P Holmes, and J L Lumley. The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. *Annual Review of Fluid Mechanics*, 25(1):539–575, 1993. doi: 10.1146/annurev.fl.25.010193.002543. URL https://doi.org/10.1146/annurev.fl.25.010193.002543. _eprint: https://doi.org/10.1146/annurev.fl.25.010193.002543.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007. ISBN 0-387-31073-8.

[6] Shaked Brody, Uri Alon, and Eran Yahav. How Attentive are Graph Attention Networks?, 2022. _eprint: 2105.14491.

[7] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, July 2017. ISSN 1558-0792. doi: 10.1109/MSP.2017.2693418. URL https://ieeexplore.ieee.org/document/7974879. Conference Name: IEEE Signal Processing Magazine.

[8] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, 2021. _eprint: 2104.13478.

[9] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally Connected Networks on Graphs. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6203.

[10] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, Cambridge, 2019. doi: 10.1017/9781108380690.

[11] Michael R. Buche and Meredith N. Silberstein. Statistical mechanical constitutive theory of polymer networks: The inextricable links between distribution, behavior, and ensemble. *Physical Review E*, 102(1):012501, July 2020. doi: 10.1103/PhysRevE.102.012501. URL https://link.aps.org/

doi/10.1103/PhysRevE.102.012501. Publisher: American Physical Society.

[12] Martin D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[13] Artūrs Bērziņš, Jan Helmig, Fabian Key, and Stefanie Elgeti. Standardized Non-Intrusive Reduced Order Modeling Using Different Regression Models With Application to Complex Flow Problems, 2021. _eprint: 2006.13706.

[14] Yves Chauvin and David E. Rumelhart, editors. *Backpropagation: Theory, architectures, and applications*. Backpropagation: Theory, architectures, and applications. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US, 1995. ISBN 0-8058-1258-X (Hardcover); 0-8058-1259-8 (Paperback). Pages: x, 561.

[15] Steven B. Damelin and Jr Miller, Willard. *The Mathematics of Signal Processing*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 2011. ISBN 978-1-107-01322-3. doi: 10.1017/CBO9781139003896.

[16] Boukje M. de Gooijer, Jos Havinga, Hubert J. M. Geijselaers, and Anton H. van den Boogaard. Evaluation of POD based surrogate models of fields resulting from nonlinear FEM simulations. *Advanced Modeling and Simulation in Engineering Sciences*, 8(1):25, November 2021. ISSN 2213-7467. doi: 10.1186/s40323-021-00210-8. URL https://doi.org/10.1186/s40323-021-00210-8.

[17] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett,

editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf.

[18] Timothy Dozat. Incorporating Nesterov Momentum into Adam. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2016)*, February 2016. URL https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ.

[19] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric, 2019. _eprint: 1903.02428.

[20] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272. PMLR, July 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html. ISSN: 2640-3498.

[21] Rini Jasmine Gladstone, Helia Rahmani, Vishvas Suryakumar, Hadi Meidani, Marta D'Elia, and Ahmad Zareei. GNN-based physics solver for time-independent PDEs, 2023. _eprint: 2303.15681.

[22] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, March 2010. URL https://proceedings.mlr.press/v9/glorot10a.html. ISSN: 1938-7228.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[24] Leo J. Grady and Jonathan R. Polimeni. *Discrete Calculus.* Springer, London, 2010. ISBN 978-1-84996-289-6 978-1-84996-290-2. doi: 10.1007/978-1-84996-290-2. URL http://link.springer.com/10.1007/978-1-84996-290-2.

[25] Anthony Gruber, Kookjin Lee, and Nathaniel Trask. Reversible and irreversible bracket-based dynamics for deep graph neural networks. In *Procedings of the Thirty-seventh Conference on Neural Information Processing Systems*, November 2023. URL https://openreview.net/forum?id=4SoTUaTK8N.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. URL http://openaccess.thecvf.com/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html.

[27] Jan S Hesthaven, Gianluigi Rozza, and Benjamin Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations.* SpringerBriefs in Mathematics. Springer International Publishing, Cham, 2016. ISBN 978-3-319-22469-5 978-3-319-22470-1. doi: 10.1007/978-3-319-22470-1. URL https://link.springer.com/10.1007/978-3-319-22470-1.

[28] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012. _eprint: 1207.0580.

[29] G. A. Holzapfel and J. C. Simo. Entropy elasticity of isotropic rubber-like solids at finite strains. *Computer Methods in Applied Mechanics and Engineering*, 132(1):17–44, May 1996. ISSN 0045-7825. doi: 10.1016/0045-7825(96)

01001-8. URL https://www.sciencedirect.com/science/article/pii/0045782596010018.

[30] Gerhard A. Holzapfel. *Nonlinear Solid Mechanics: A Continuum Approach for Engineering.* Wiley, April 2000. ISBN 978-0-471-82304-9. Google-Books-ID: _ZkeAQAAIAAJ.

[31] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, January 1990. ISSN 0893-6080. doi: 10.1016/0893-6080(90)90005-6. URL https://www.sciencedirect.com/science/article/pii/0893608090900056.

[32] Kenji Kashima. Nonlinear model reduction by deep autoencoder of noise response data. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5750–5755, December 2016. doi: 10.1109/CDC.2016.7799153. URL https://ieeexplore.ieee.org/document/7799153.

[33] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[34] Yann Lecun. Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels, editors, *Connectionism in perspective.* Elsevier, Zurich, Switzerland, 1989.

[35] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the Variance of the Adaptive Learning Rate and Beyond. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, August 2019.

[36] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[37] S. C. H. Lu and K. S. Pister. Decomposition of deformation and representation of the free energy function for isotropic thermoelastic solids. *International Journal of Solids and Structures*, 11(7):927–934, July 1975. ISSN 0020-7683. doi: 10.1016/0020-7683(75)90015-3. URL https://www.sciencedirect.com/science/article/pii/0020768375900153.

[38] Winfried Lötzsch, Simon Ohler, and Johannes S Otterbach. Learning the Solution Operator of Boundary Value Problems using Graph Neural Networks. *ICML 2022 2nd AI for Science Workshop*, 2022.

[39] M. D. McKay, R. J. Beckman, and W. J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239–245, 1979. ISSN 0040-1706. doi: 10.2307/1268522. URL https://www.jstor.org/stable/1268522. Publisher: [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality].

[40] Michele Milano and Petros Koumoutsakos. Neural Network Modeling for Near Wall Turbulent Flow. *Journal of Computational Physics*, 182(1):1–26, October 2002. ISSN 0021-9991. doi: 10.1006/jcph.2002.7146. URL https://www.sciencedirect.com/science/article/pii/S0021999102971469.

[41] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN 978-0-

387-30303-1. doi: 10.1007/978-0-387-40065-5. URL http://link.springer.com/10.1007/978-0-387-40065-5.

[42] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning Mesh-Based Simulation with Graph Networks, 2021. _eprint: 2010.03409.

[43] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, January 1999. ISSN 0893-6080. doi: 10.1016/S0893-6080(98)00116-6. URL https://www.sciencedirect.com/science/article/pii/S0893608098001166.

[44] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv:1711.10561 [cs, math, stat]*, November 2017. URL http://arxiv.org/abs/1711.10561. arXiv: 1711.10561.

[45] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006. ISBN 0-262-18253-X.

[46] Fadil Santosa and William W. Symes. Linear Inversion of Band-Limited Reflection Seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986. doi: 10.1137/0907087. URL https://doi.org/10.1137/0907087. _eprint: https://doi.org/10.1137/0907087.

[47] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, January 2009. ISSN 1941-0093. doi: 10.1109/TNN.2008.2005605. URL https://ieeexplore.ieee.org/document/4700287. Conference Name: IEEE Transactions on Neural Networks.

[48] Christian Schwarz, Patrick Ackert, and Reinhard Mauermann. Principal component analysis and singular value decomposition used for a numerical sensitivity analysis of a complex drawn part. *The International Journal of Advanced Manufacturing Technology*, 94(5):2255–2265, February 2018. ISSN 1433-3015. doi: 10.1007/s00170-017-0980-z. URL https://doi.org/10.1007/s00170-017-0980-z.

[49] Christian Schwarz, Thomas Kropp, Christian Kraus, and Welf-Guntram Drossel. Optimization of thick sheet clinching tools using principal component analysis. *The International Journal of Advanced Manufacturing Technology*, 106(1):471–479, January 2020. ISSN 1433-3015. doi: 10.1007/s00170-019-04512-5. URL https://doi.org/10.1007/s00170-019-04512-5.

[50] J. C. Simo and C. Miehe. Associative coupled thermoplasticity at finite strains: Formulation, numerical analysis and implementation. *Computer Methods in Applied Mechanics and Engineering*, 98(1):41–104, July 1992. ISSN 0045-7825. doi: 10.1016/0045-7825(92)90170-O. URL https://www.sciencedirect.com/science/article/pii/004578259290170O.

[51] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks, 2018. _eprint: 1710.10903.

[52] K. Veroy and A. T. Patera. Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds. *International Journal for Numerical Methods in Fluids*, 47(8-9):773–788, 2005. ISSN 1097-0363. doi: 10.1002/fld.867. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.867. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.867.

[53] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[54] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6861–6871. PMLR, May 2019. URL https://proceedings.mlr.press/v97/wu19e.html. ISSN: 2640-3498.