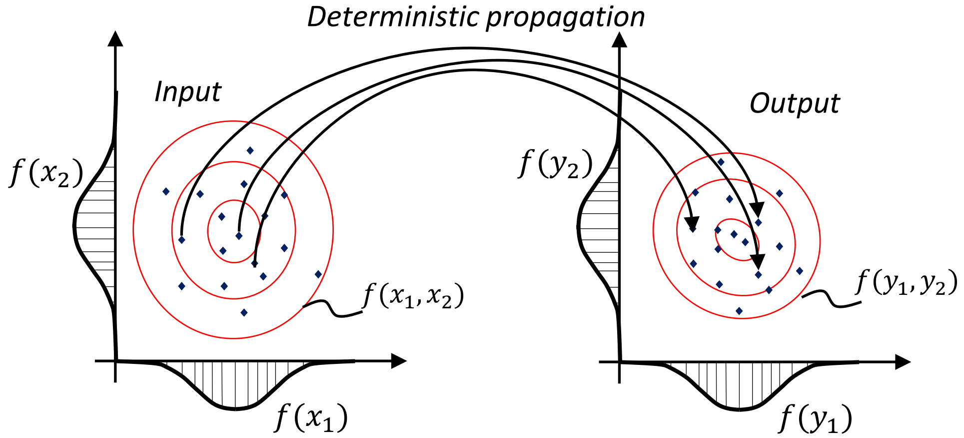
A Sobolev trained neural network surrogate with residual weighting scheme for computational mechanics

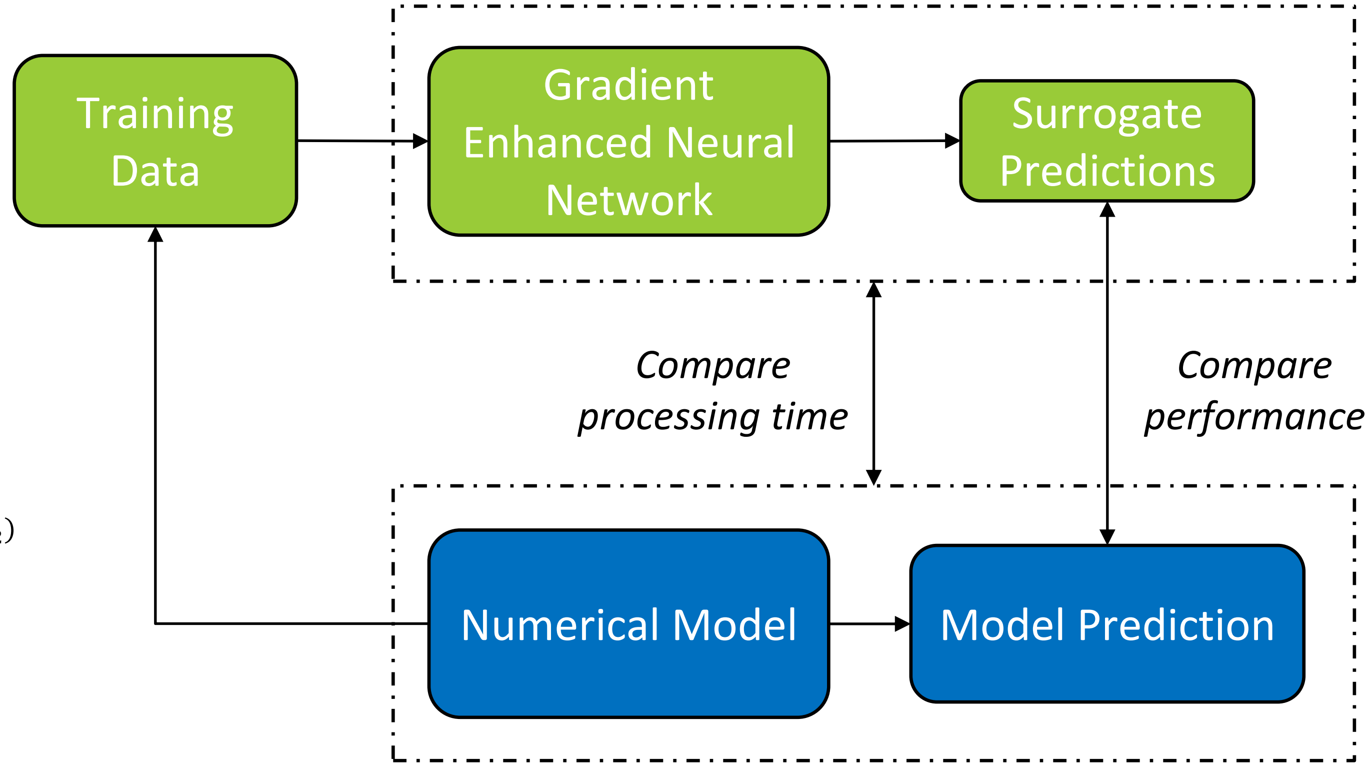
The financial support provided under project-ID 278867966, TRR188 by the Deutsche Forschungsgemeinschaft (DFG) is gratefully acknowledged.

M.G.R. Faes
F.-J. Barthold
M.A. Valdebenito
J. Liedmann

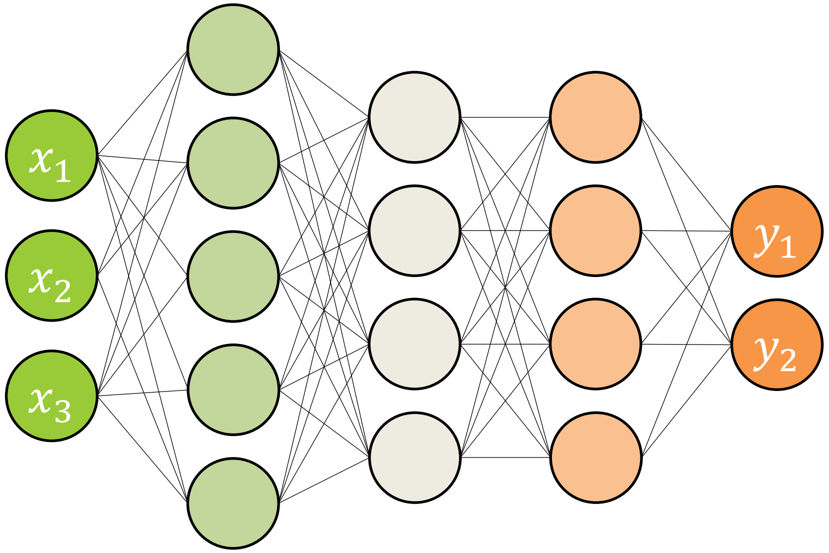
Presented by *Ali Kilicsoy*,
Chair for Reliability Engineering at TU Dortmund

General process





Base neural network



Model parameters θ and data x_i, y_i

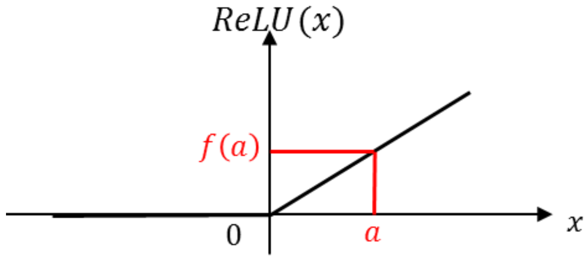
$$\theta = \left(W^{[l]}, b^{[l]}\right)_{l=1}^{L}, (x_i, y_i)_{i=1}^{m}$$

Loss function L

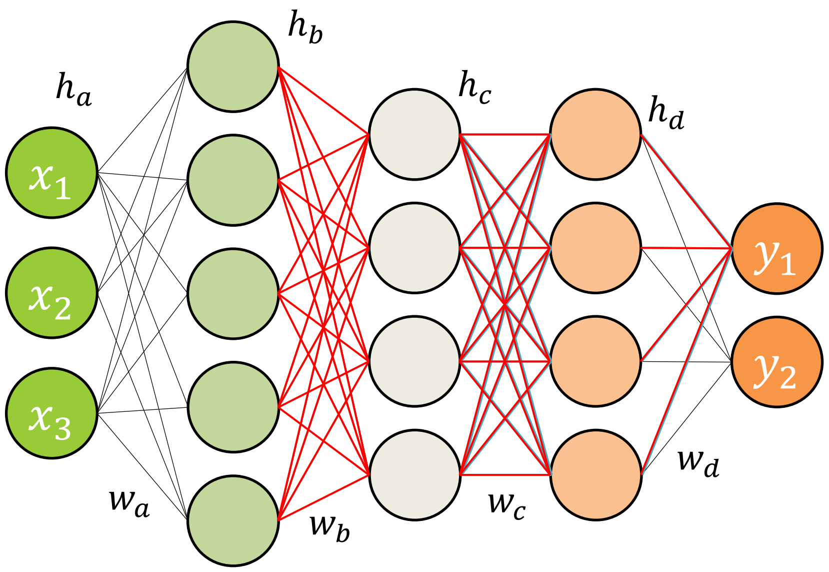
$$L = \frac{1}{2m} \sum_{i=1}^{m} \| N_\theta(x_i) - y_i \|^2$$

$$h^{[l]} = \phi(W^{[l]} h^{[l-1]} + b^{[l]})$$

$$N_\theta(x) = h^{[L]}$$



Base neural network



Model parameters θ and data x_i, y_i

$$\theta = \left(W^{[l]}, b^{[l]}\right)_{l=1}^{L}, (x_i, y_i)_{i=1}^{m}$$

Loss function L

$$L = \frac{1}{2m} \sum_{i=1}^{m} \| N_\theta(x_i) - y_i \|^2$$

$$h^{[l]} = \phi(W^{[l]} h^{[l-1]} + b^{[l]})$$

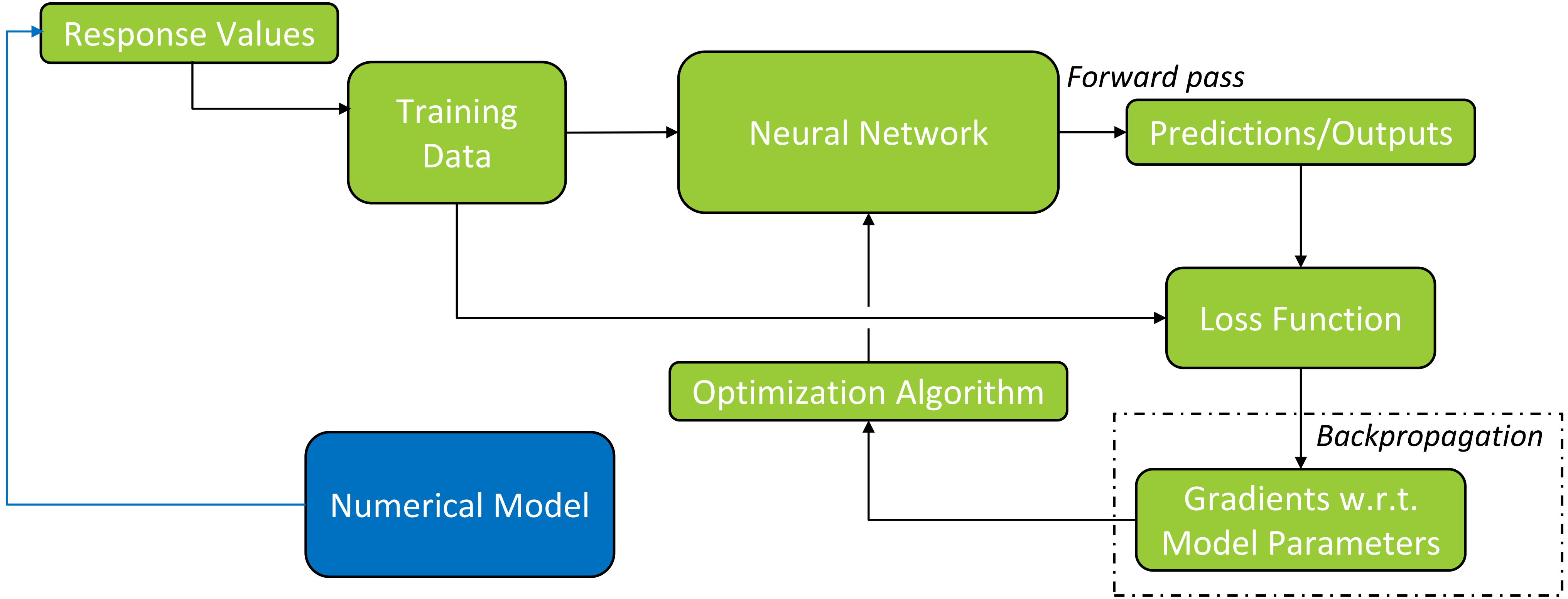
$$N_\theta(x) = h^{[L]}$$

Automatic differentiation per chain rule

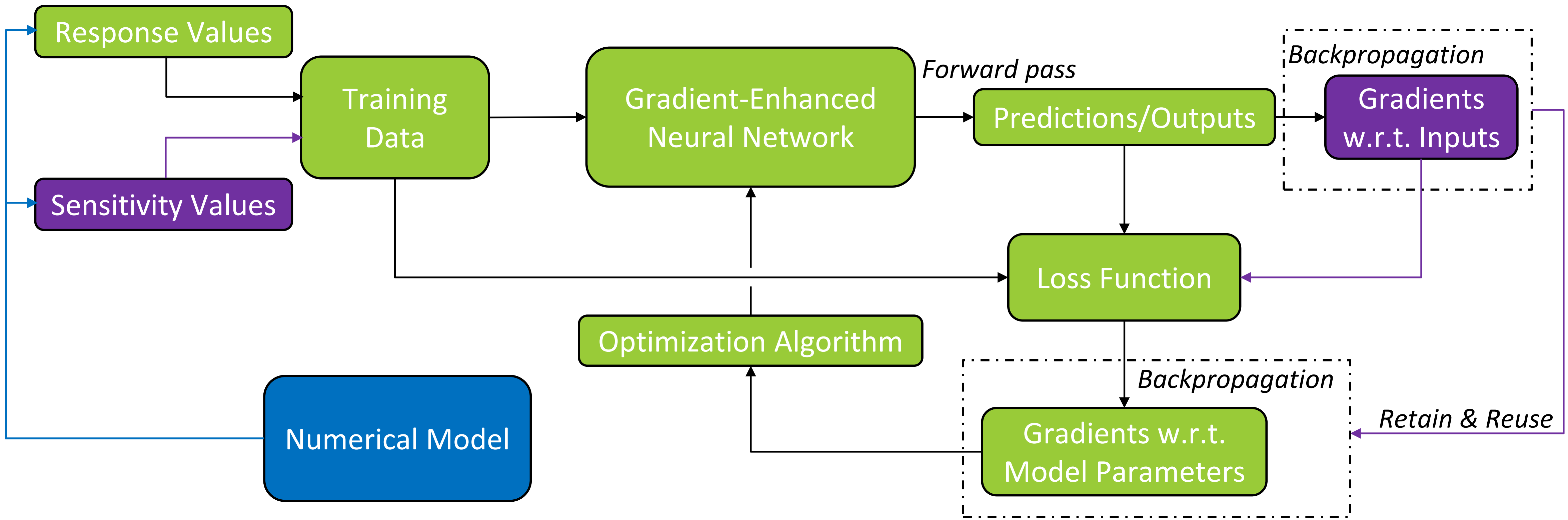
$$\frac{\partial y_1}{\partial w_b} = \frac{\partial y_1}{\partial h_d} \frac{\partial h_d}{\partial h_c} \frac{\partial h_c}{\partial w_b}$$

$$\frac{\partial y_1}{\partial w_a} = \frac{\partial y_1}{\partial h_d} \frac{\partial h_d}{\partial h_c} \frac{\partial h_c}{\partial h_b} \frac{\partial h_b}{\partial w_a}$$

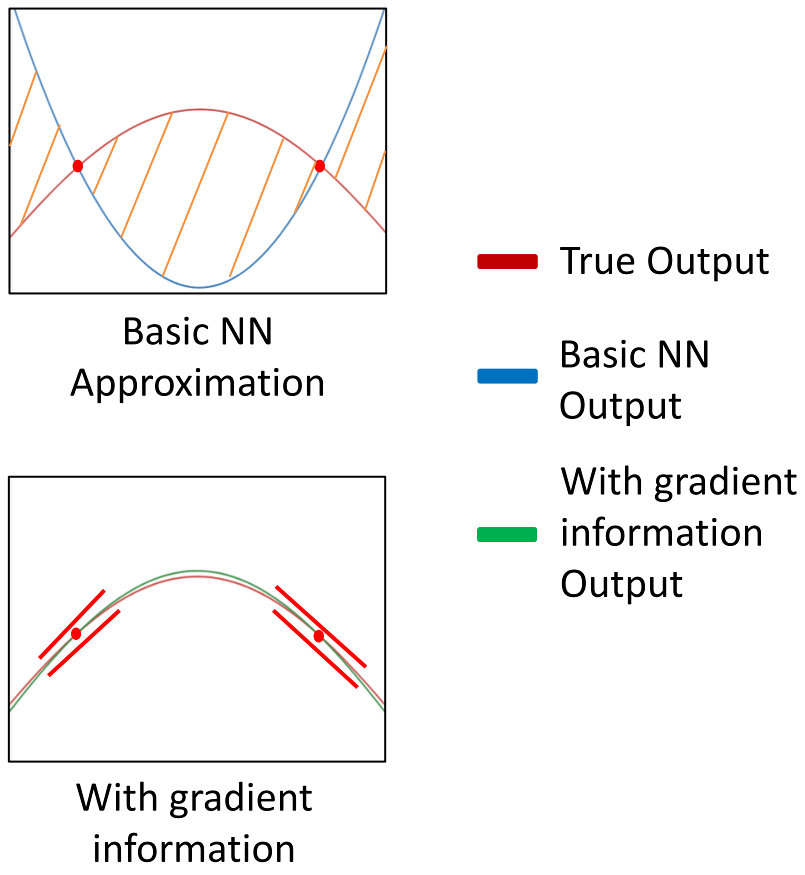
Base neural network

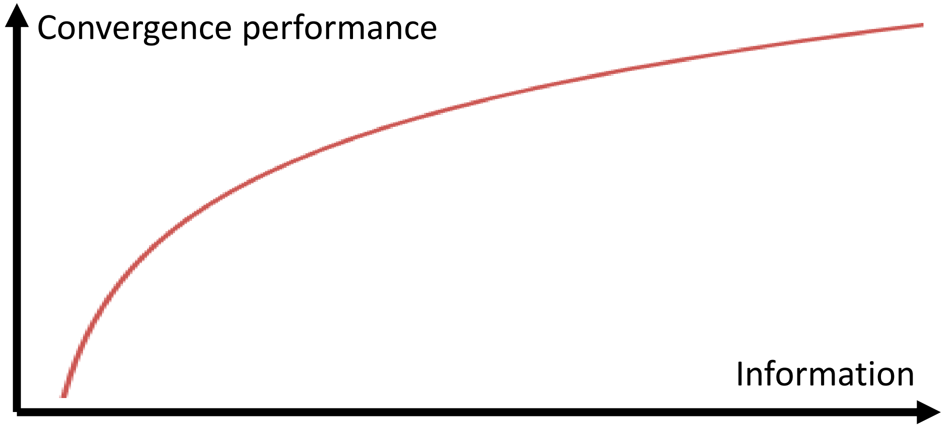


Gradient enhanced neural network



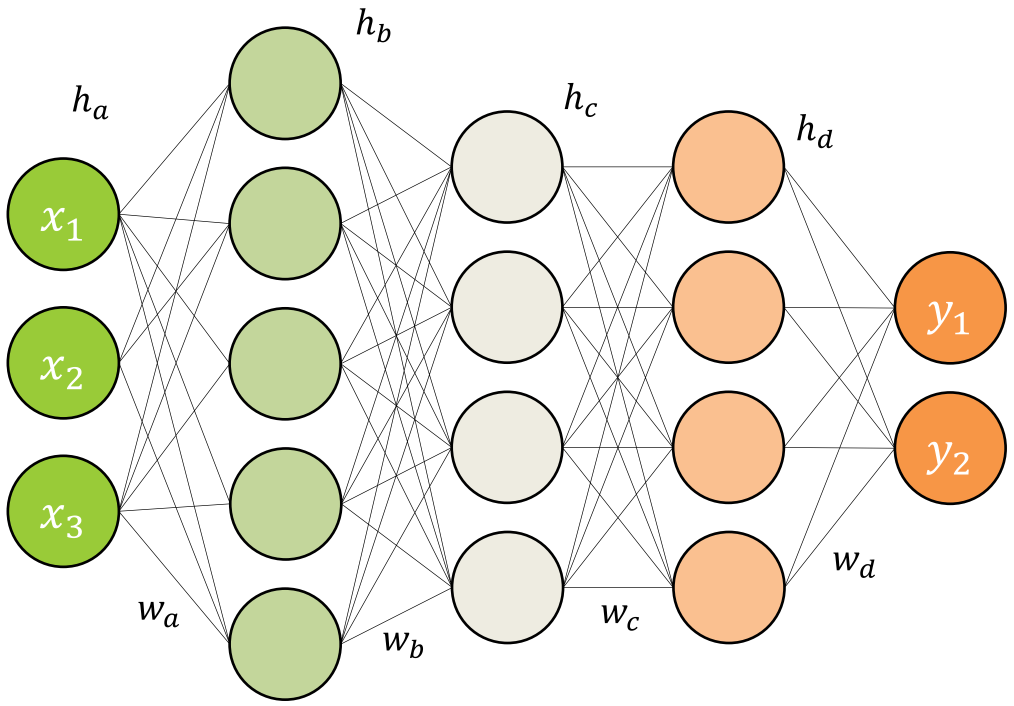
Gradient enhanced neural network





- Sensitivity information decisively influences training convergence
- Sensitivity information is available for low cost (computational, economical, etc.)
- Varying degrees of application

Gradient enhanced neural network



Model parameters θ and data $x_i, y_i, \nabla y_i$

$$\theta = \left(W^{[l]}, b^{[l]}\right)_{l=1}^{L}, (x_i, y_i, \nabla y_i)_{i=1}^{m}$$

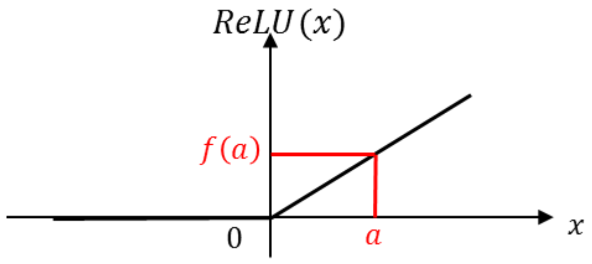
Loss function L

$$L = \frac{1}{2m}\sum_{i=1}^{m}(\|N_\theta(x_i) - y_i\|^2 + \|\nabla N_\theta(x_i) - \nabla y_i\|^2)$$

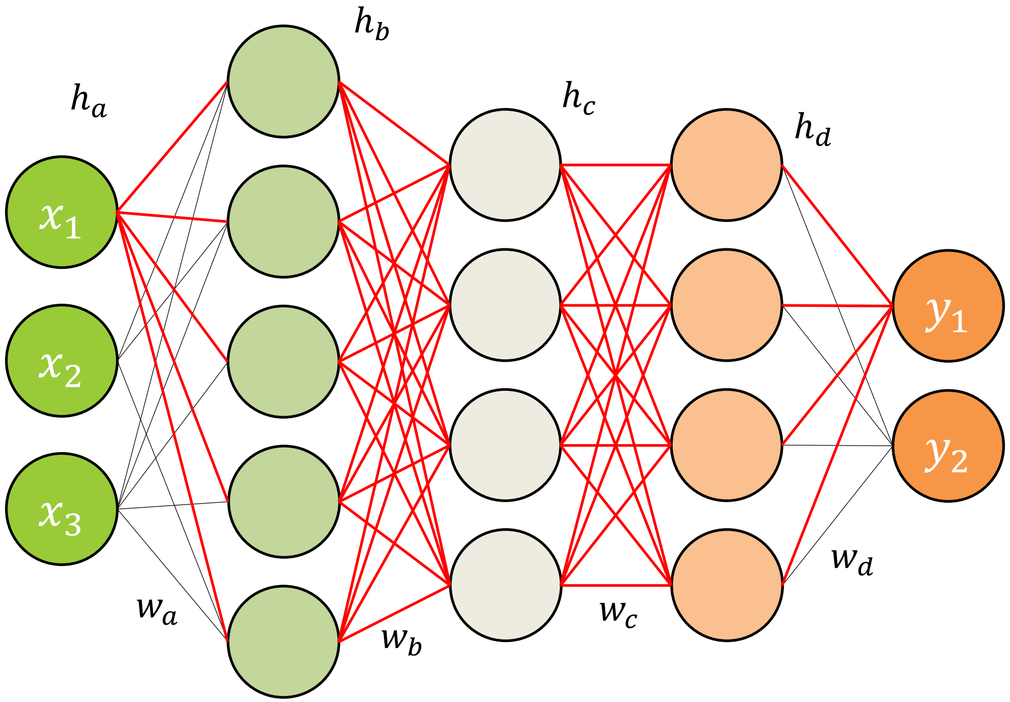
$$h^{[l]} = \phi(W^{[l]}h^{[l-1]} + b^{[l]})$$

$$N_\theta(x) = h^{[L]}$$

$$\nabla N_\theta(x) = \frac{\partial N_\theta(x)}{\partial x}$$



Gradient enhanced neural network

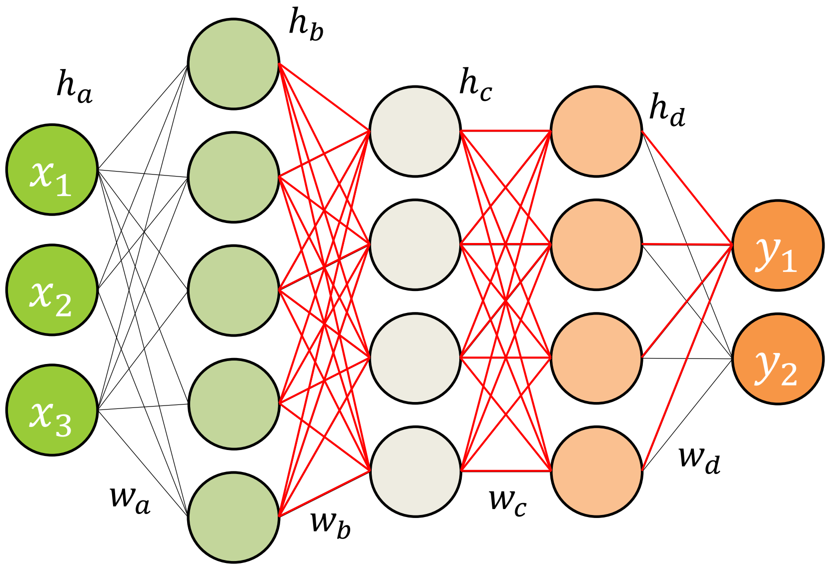


Automatic Differentiation per chain rule

$$\frac{\partial y_1}{\partial x_1} = \frac{\partial y_1}{\partial h_d}\frac{\partial h_d}{\partial h_c}\frac{\partial h_c}{\partial h_b}\frac{\partial h_b}{\partial x_1}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial N_\theta}\frac{\partial N_\theta}{\partial w} + \frac{\partial L}{\partial \nabla N_\theta}\frac{\partial \nabla N_\theta}{\partial w}$$

Previously:



$$\frac{\partial y_1}{\partial w_a} = \frac{\partial y_1}{\partial h_d}\frac{\partial h_d}{\partial h_c}\frac{\partial h_c}{\partial h_b}\frac{\partial h_b}{\partial w_a}$$

→ Reusable gradient terms

Residual weighting

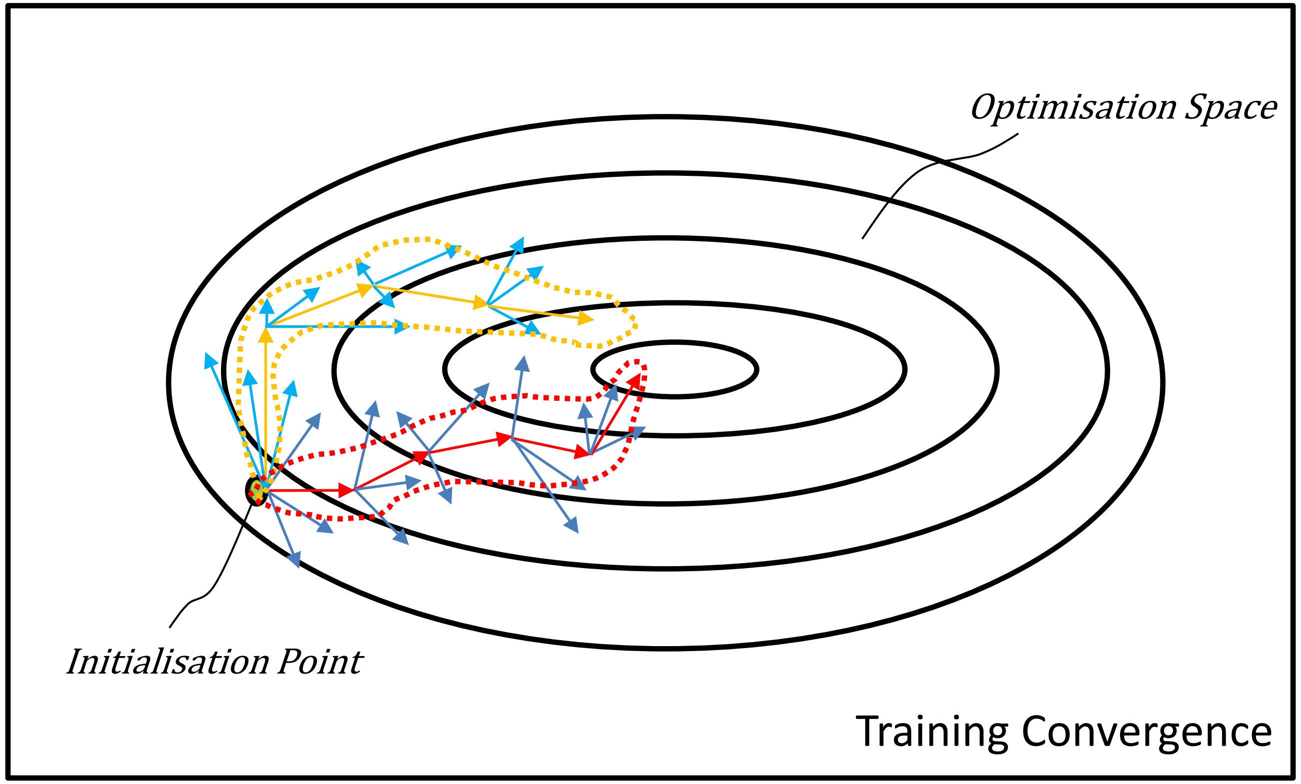
$$L = \frac{1}{2}\sum(\boldsymbol{N_\theta} - \boldsymbol{y})^2 + \frac{1}{2}\sum\sum(\boldsymbol{\nabla N_\theta} - \boldsymbol{\nabla y})^2 \xRightarrow{Gradients} \nabla L = \nabla L_1 + \nabla L_2 + \cdots$$

MSE: $\frac{\partial L}{\partial \theta} = (N_\theta - y)\frac{\partial N_\theta}{\partial \theta}$

- Due MSE definition of loss terms → Magnitude $|L_i|$ acts as a weight for i-th gradient direction
- The less accurate L_i, the greater its gradient factor $|L_i|$
- Varying difficulty → staggered $|L_i|$ → gradient update intuitive to worst L_i, counterintuitive to best L_i

$$\Rightarrow \quad L = \frac{1}{2}\sum \boldsymbol{\lambda_1}(\boldsymbol{N_\theta} - \boldsymbol{y})^2 + \frac{1}{2}\sum\sum \boldsymbol{\lambda_2}(\boldsymbol{\nabla N_\theta} - \boldsymbol{\nabla y})^2$$

Optimal convergence



- Training step split into individual parts of the sum, which are vectors
- Unique composition at each training step
- Optimal convergence path more sophisticated

Weighting targets

Abbreviation	Method	Target
L_{max}	$\max L$	Loss Maximisation
CD_{min}	$\min(1 - \frac{\nabla L \cdot \nabla L_i}{\lvert\nabla L\rvert \cdot \lvert\nabla L_i\rvert})$	Gradient Alignment
SNN	Sobolev Trained	-

$$L = \frac{1}{2}\sum \boldsymbol{\lambda_1}(\boldsymbol{N_\theta} - \boldsymbol{y})^2 + \frac{1}{2}\sum\sum \boldsymbol{\lambda_2}\,(\boldsymbol{\nabla N_\theta} - \boldsymbol{\nabla y})^2$$

A.O.M. Kilicsoy, J. Liedmann, M.A. Valdebenito, F.-J. Barthold and M.G.R. Faes, "Sobolev Neural Network With Residual Weighting as a Surrogate in Linear and Non-Linear Mechanics," in IEEE Access, vol. 12, pp. 137144-137161, 2024, doi: 10.1109/ACCESS.2024.3465572

Loss maximization

$\max_{\lambda} L$ subject to $\lambda \geq 1$

- Very straightforward
- Essentially a constant gradient step, but -

→$|L_i|$ dicate Δ_i, decreasing with convergence

Weighting targets

Abbreviation	Method	Target
L_{max}	$\max L$	Loss Maximisation
CD_{min}	$\min(1 - \frac{\nabla L \cdot \nabla L_i}{\lvert\nabla L\rvert \cdot \lvert\nabla L_i\rvert})$	Gradient Alignment
SNN	Sobolev Trained	-

$$L = \frac{1}{2}\sum \boldsymbol{\lambda_1}(\boldsymbol{N_\theta} - \boldsymbol{y})^2 + \frac{1}{2}\sum\sum \boldsymbol{\lambda_2}\,(\boldsymbol{\nabla N_\theta} - \boldsymbol{\nabla y})^2$$

A.O.M. Kilicsoy, J. Liedmann, M.A. Valdebenito, F.-J. Barthold and M.G.R. Faes, "Sobolev Neural Network With Residual Weighting as a Surrogate in Linear and Non-Linear Mechanics," in IEEE Access, vol. 12, pp. 137144-137161, 2024, doi: 10.1109/ACCESS.2024.3465572

Loss gradient alignment

$\min_{\lambda}(1 - \frac{\nabla L \cdot \nabla L_i}{\lvert\nabla L\rvert \cdot \lvert\nabla L_i\rvert})$ subject to $\lambda \geq 1$

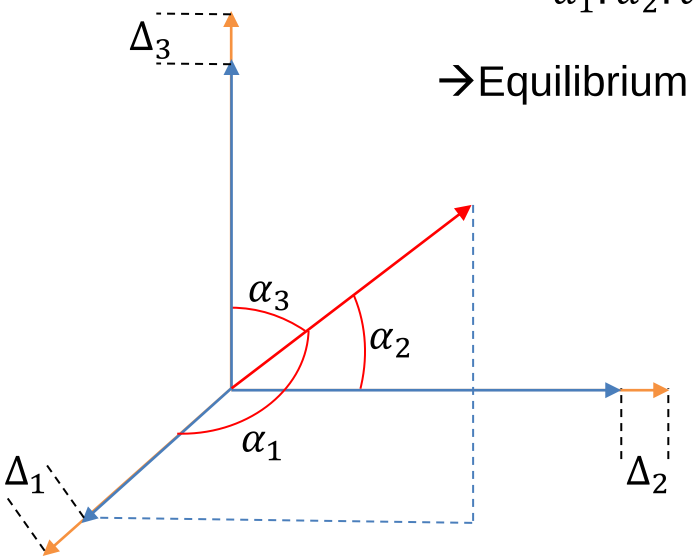
- Ratios $\lvert\nabla L_i\rvert$ dicate α_i, ratios α_i dictate Δ_i
- When a loss target is dominated, its residual weight increases stronger
- Monotonically increasing even with convergence – regularization issue thus clipped

Weighting targets

For one training step:

$\alpha_1 : \alpha_2 : \alpha_3 \quad \Rightarrow \Delta_1 : \Delta_2 : \Delta_3$

→Equilibrium in residual weight change



$$L = \frac{1}{2}\sum \lambda_1 (\boldsymbol{N_\theta} - \boldsymbol{y})^2 + \frac{1}{2}\sum\sum \lambda_2\, (\boldsymbol{\nabla N_\theta} - \boldsymbol{\nabla y})^2$$

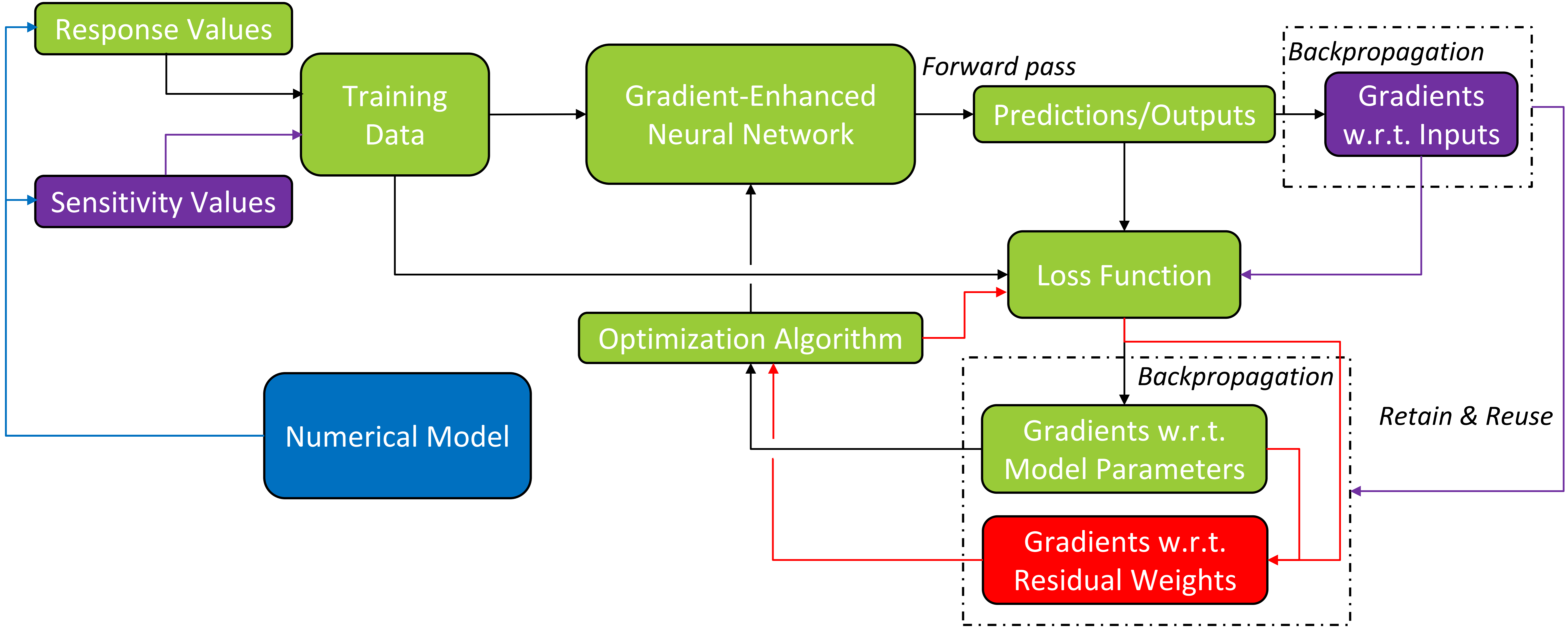
A.O.M. Kilicsoy, J. Liedmann, M.A. Valdebenito, F.-J. Barthold and M.G.R. Faes, "Sobolev Neural Network With Residual Weighting as a Surrogate in Linear and Non-Linear Mechanics," in IEEE Access, vol. 12, pp. 137144-137161, 2024, doi: 10.1109/ACCESS.2024.3465572

Loss gradient alignment

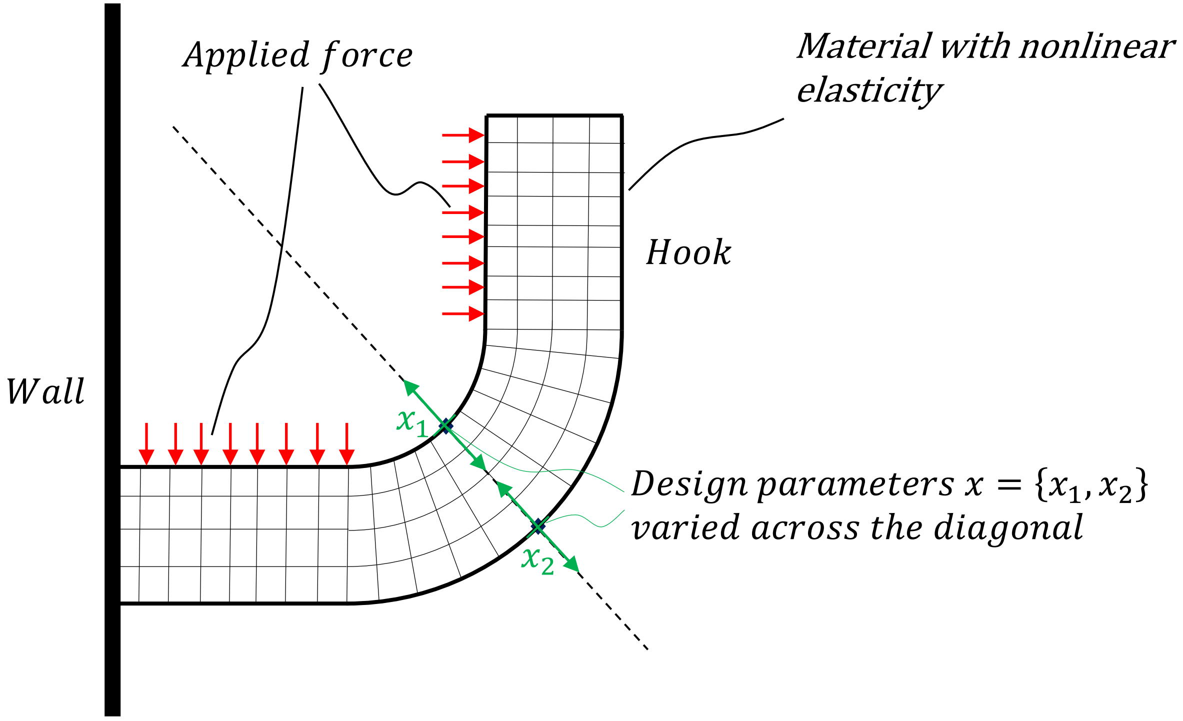
$\min_{\lambda}(1 - \frac{\nabla L \cdot \nabla L_i}{|\nabla L| \cdot |\nabla L_i|})$ subject to $\lambda \geq 1$

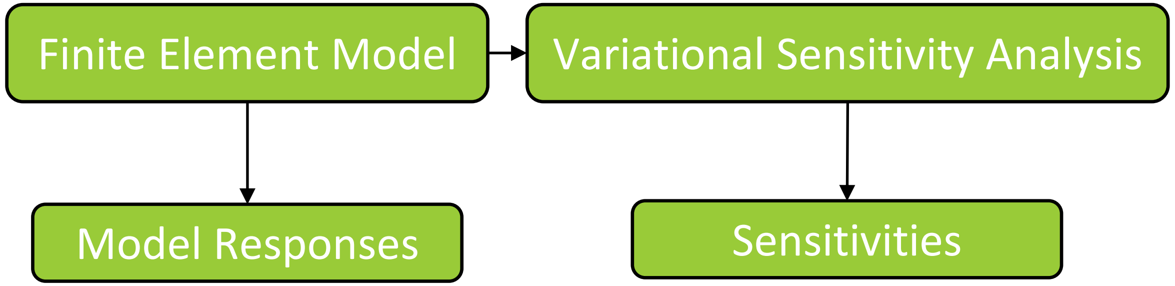
- Ratios $|\nabla L_i|$ dicate α_i, ratios α_i dictate Δ_i
- When a loss target is dominated, its residual weight increases stronger
- Monotonically increasing even with convergence – regularization issue thus clipped

Weighted gradient enhanced neural network



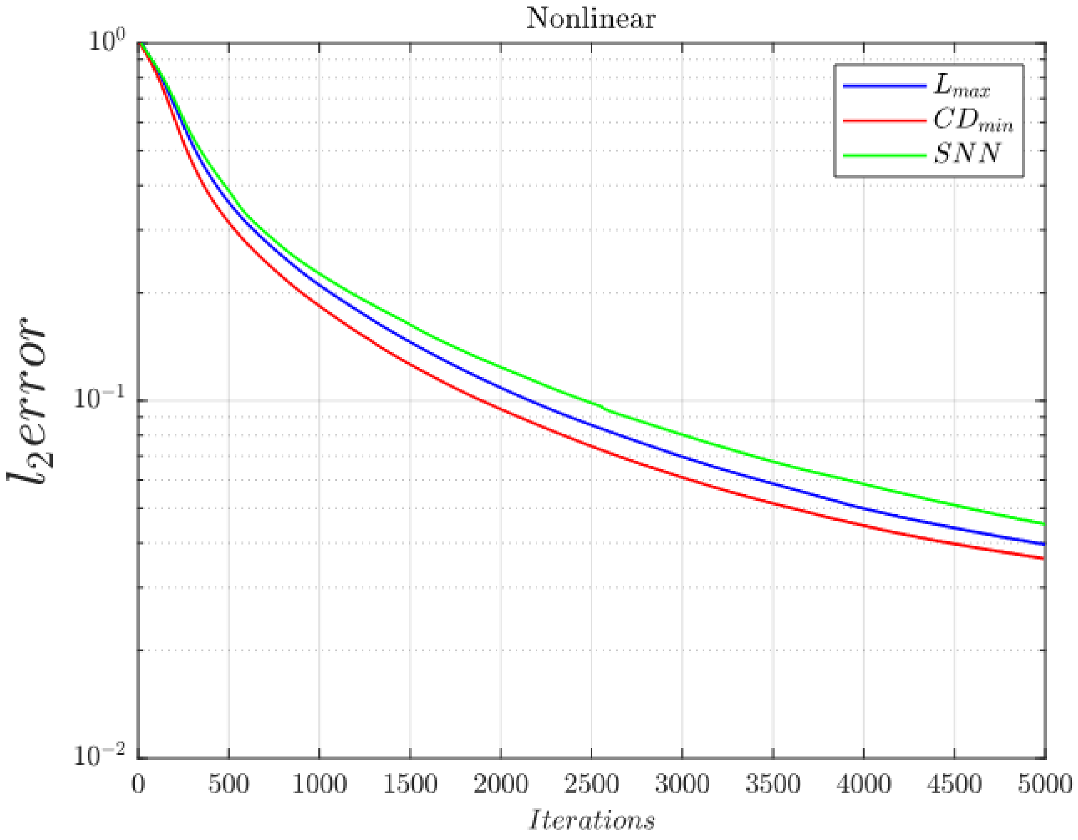
2D Hook with geometric design parameters



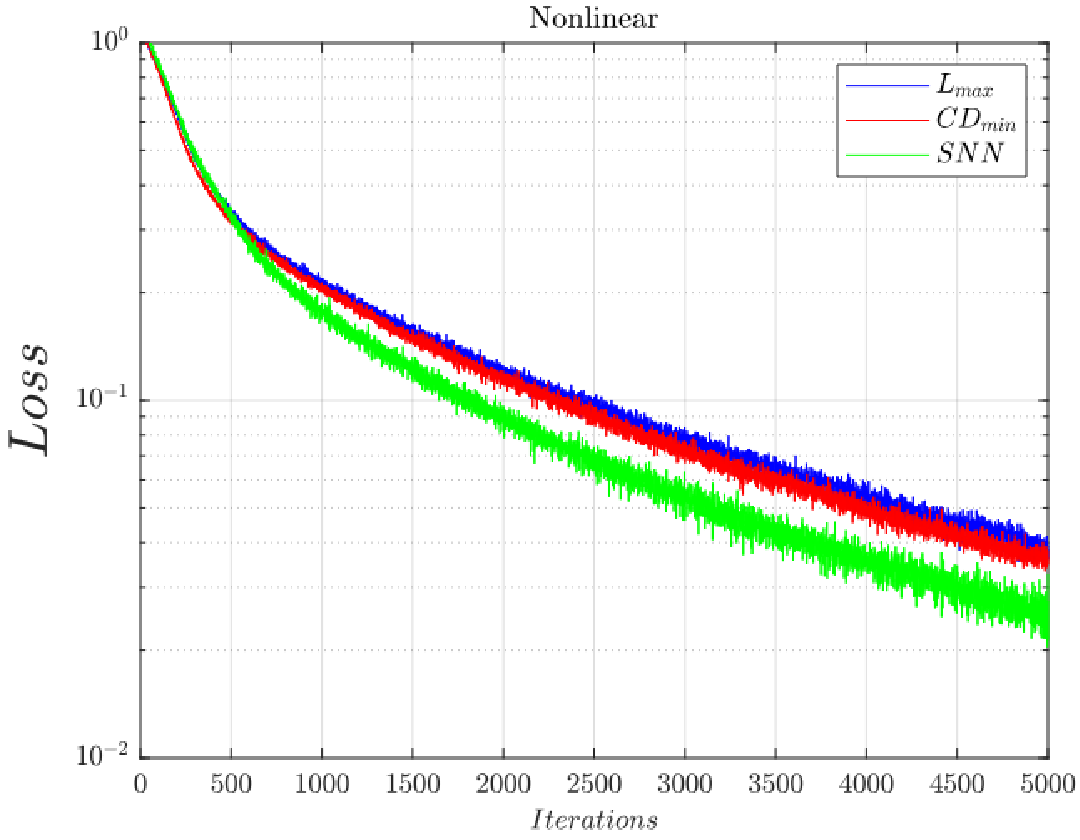


Model Components	Value
Degrees of Freedom per Element	2
Elements	512
Design Parameters	2
Nodes per Element	4
Nonlinear material	St Venant, plane stress
System Behaviour	Static

Results

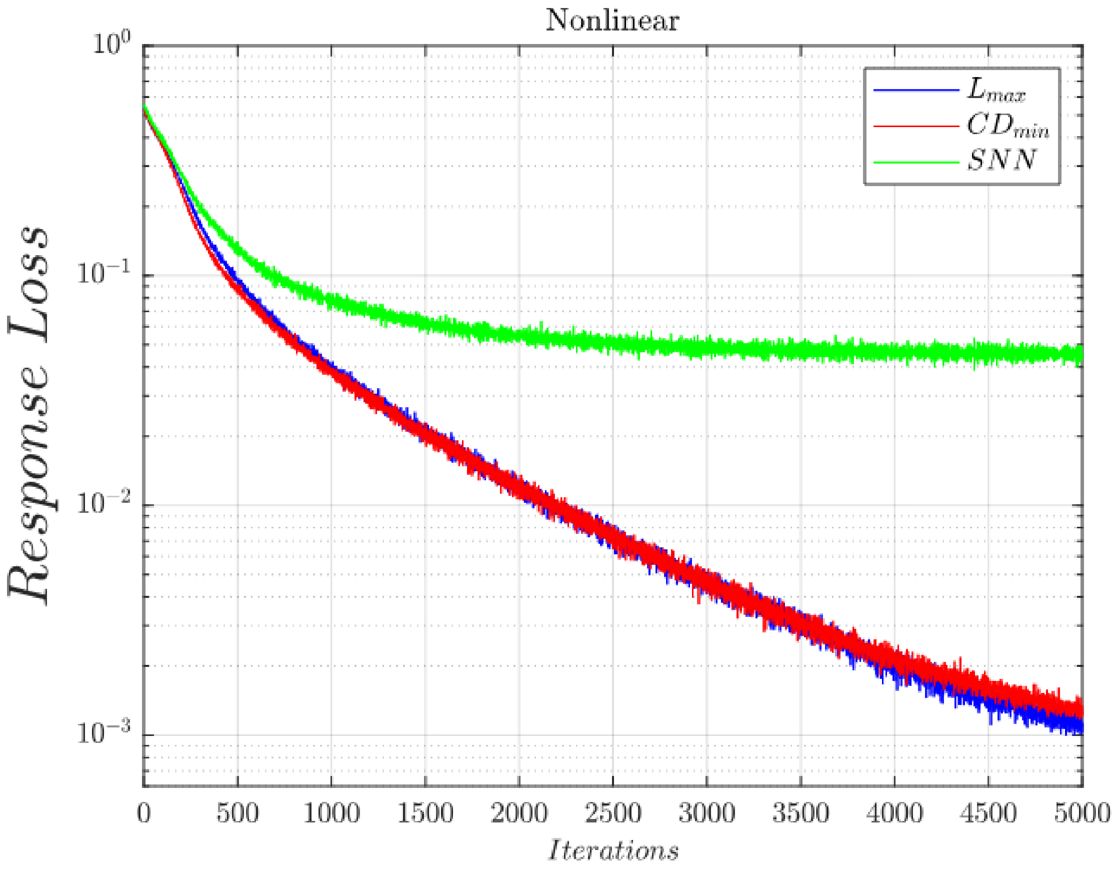


Nonlinear
L_{max}
CD_{min}
SNN
$l_2 error$
10^0
10^{-1}
10^{-2}
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
$Iterations$

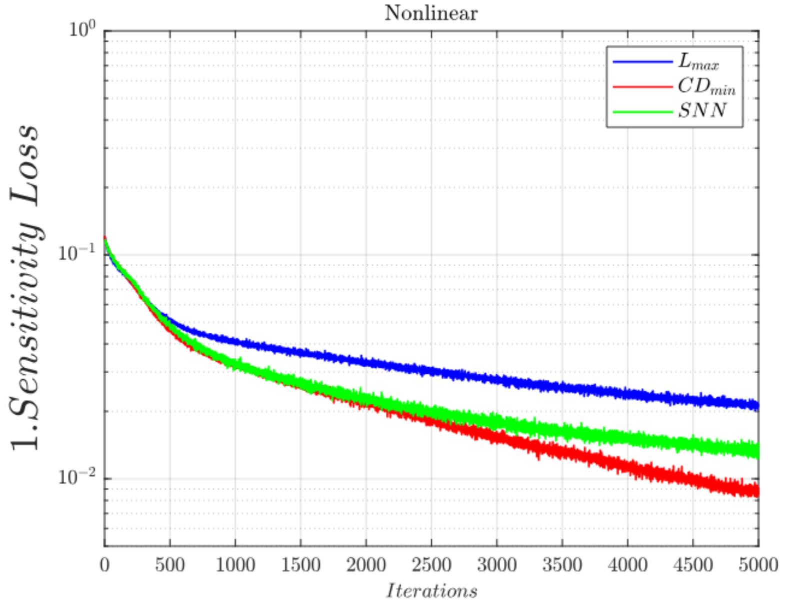


Nonlinear
L_{max}
CD_{min}
SNN
$Loss$
10^0
10^{-1}
10^{-2}
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
$Iterations$

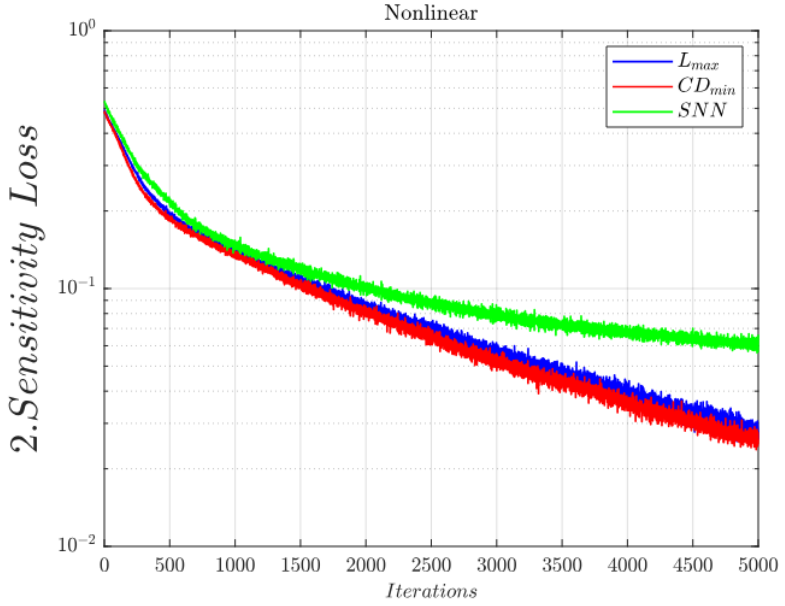
Results



Nonlinear
L_{max}
CD_{min}
SNN
$Response\ Loss$
$Iterations$

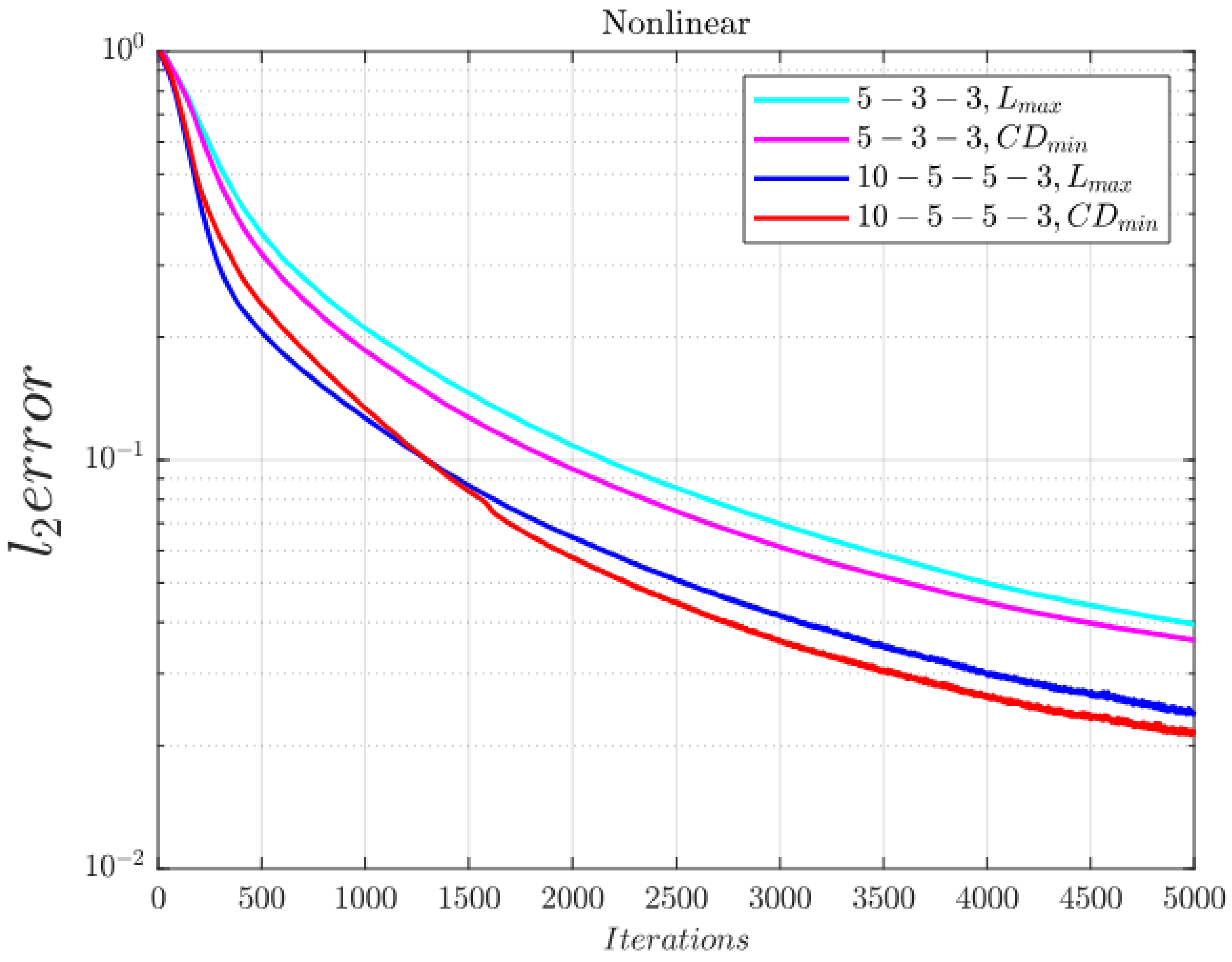


Nonlinear
L_{max}
CD_{min}
SNN
$1.Sensitivity\ Loss$
$Iterations$



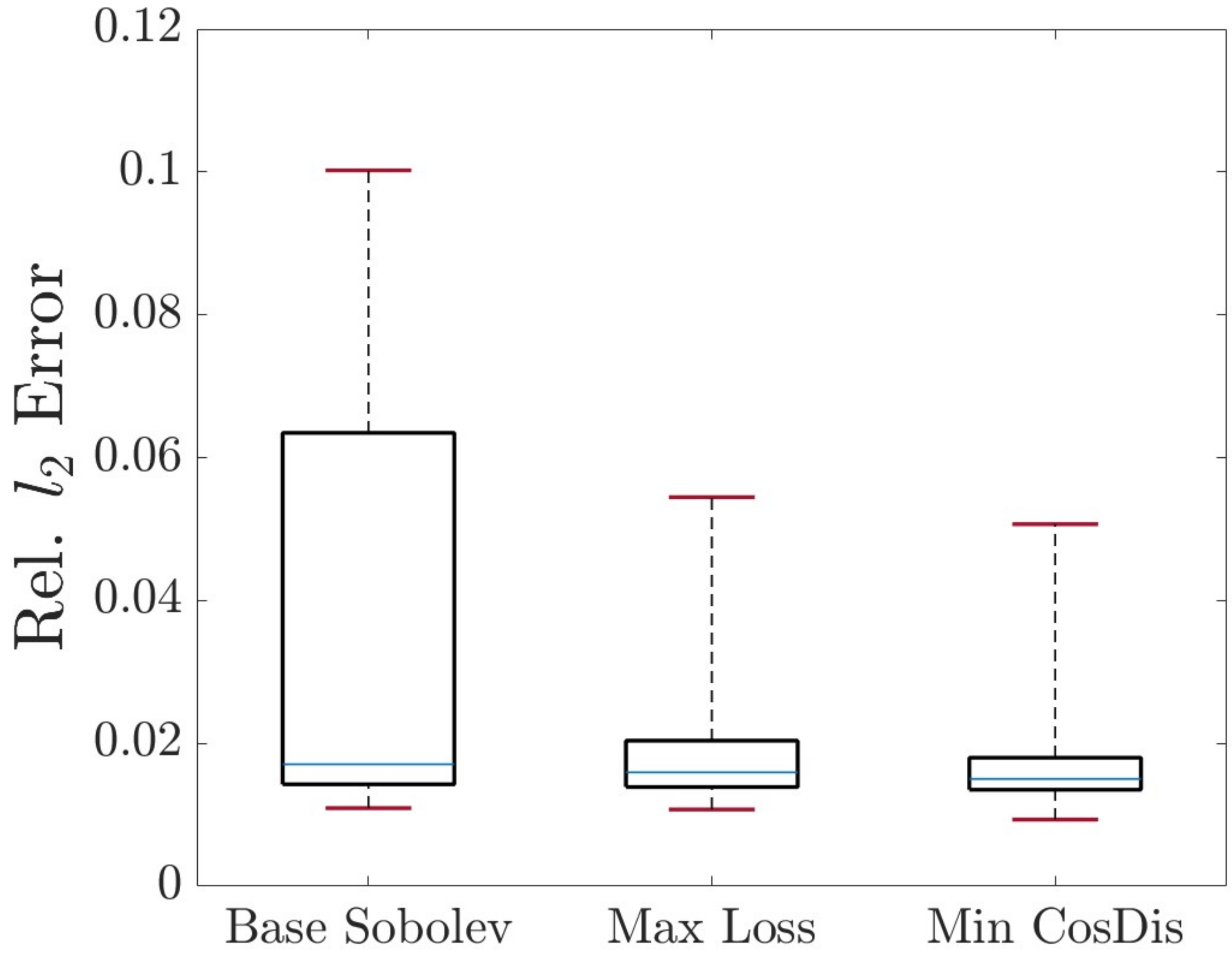
Nonlinear
L_{max}
CD_{min}
SNN
$2.Sensitivity\ Loss$
$Iterations$

Results



Nonlinear
10^0
10^{-1}
10^{-2}
$l_2 error$
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
$Iterations$
$5-3-3, L_{max}$
$5-3-3, CD_{min}$
$10-5-5-3, L_{max}$
$10-5-5-3, CD_{min}$

Results



0.12
0.1
0.08
0.06
0.04
0.02
0
Rel. l_2 Error
Base Sobolev
Max Loss
Min CosDis

Conclusion

- Multiple loss terms lead to staggered training focus
  - potentially counterintuitive at various training steps without priority weighting

- Weighting for improved training convergence
  - dynamic definition to avoid tuning and flexible to current iteration
  - optimization target/function crucial to training convergence
  - → Choice: equilibrium between gradient vectors; no domination by residuals

- Mean accuracy improvement small; however training more robust
  - widening gap with increasing model size

- CD_{min} convergence performs well for all residuals
  - more complex case could show increasing performance gap

Thank you!