Tutorial Machine Learning

Hyperelasticity I & II

Analytical potential and load types

model implementation

Transversely isotropic hyperelastic potential:

$$W(\mathbf{F}) = 8I_1 + 10J^2 - 56log(J) + 0.2(I_4^2 + I_5^2) - 44 \tag{1}$$

Invariants:

$$I_1 = tr(\mathbf{C}), \quad J = def\mathbf{F}, \quad I_4 = tr(\mathbf{CG_{ti}}), \quad I_5 = tr(Cof(\mathbf{CG_{ti}})) \tag{2}$$

Piola-Kirchhoff-stress as potential $\rightarrow$ GradientTape:

$$\mathbf{P} = \frac{\partial W(\mathbf{F})}{\partial \mathbf{F}} \tag{3}$$

Training with different weighting strategy

weighting based on Frobenius norm

Training with small stress values can prevent the model from learning.
We want to assign a greater weight to small values.

Using the Frobenius norm, we compute:

$$w = \frac{1}{\#(D)} \sum_j \|\mathbf{P}^j\| \tag{4}$$

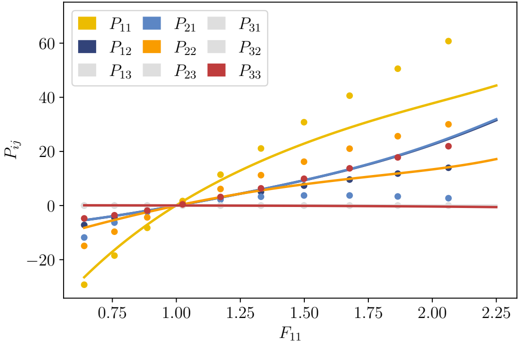
$\Rightarrow$ Calibration showed no significant change in training with/without weighting strategy.

If not mentioned differently, all models trained with 3 layers and 16 nodes for 10,000 epochs.

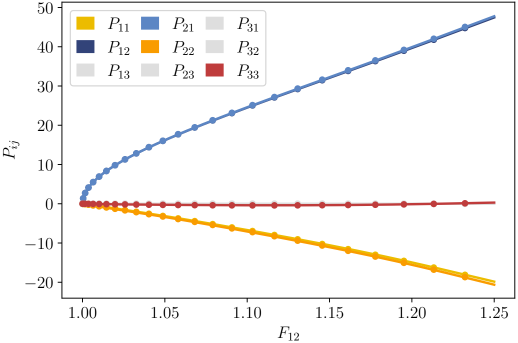
FFNN naive approach

training on right Cauchy-Green tensor

Training with **C** as input, **P** as output:



(a) Mixed load



(b) Shear load

Invariant based implementation

model implementation

We will implement a neural network which takes a vector of invariants:

$$\mathcal{I} = (I_1, J, -J, I_4, I_5) \tag{5}$$

We further restrict the first layer to be polyconvex: (convex for each invariant)

$$\mathbf{a} \otimes \mathbf{b} : \frac{\partial^2 W}{\partial \mathbf{F}^2} : \mathbf{a} \otimes \mathbf{b} \geq 0 \tag{6}$$

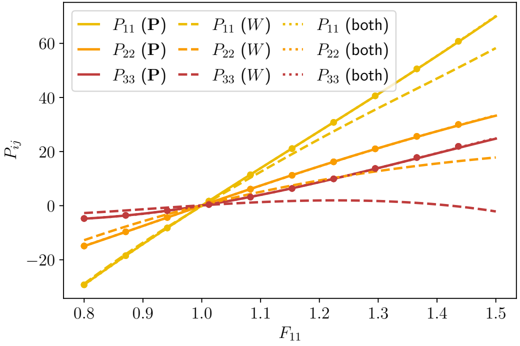
This is matched if W is a function as seen in eq. 7 below:

$$W = W(\mathbf{F}, cof(\mathbf{F}), J) \tag{7}$$

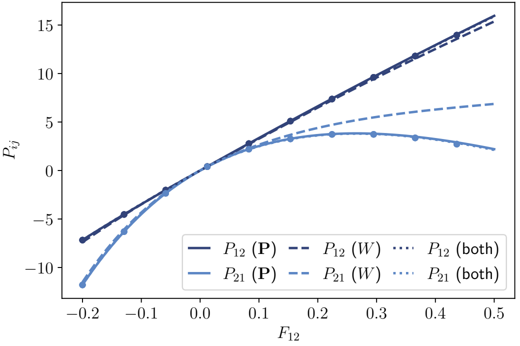
Invariant based implementation

model calibration, complex load data

Interpolation of load paths:



(a) Mixed load, volumetric components

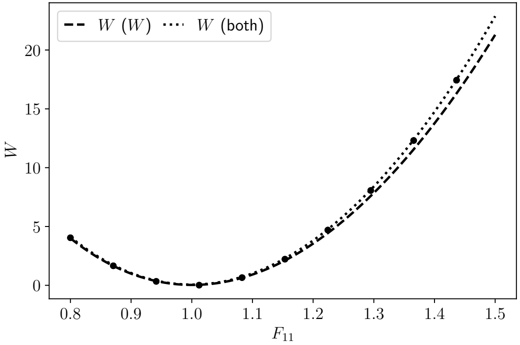


(b) Mixed load, shear components

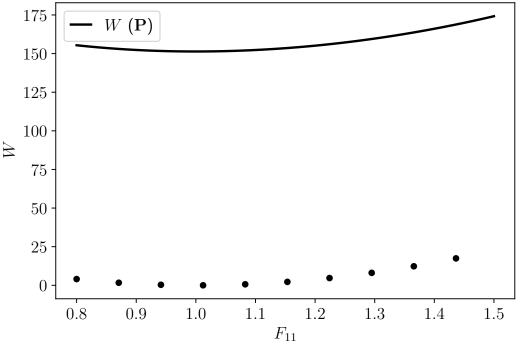
Invariant based implementation

training on complex load data

TECHNISCHE UNIVERSITÄT DARMSTADT



(a) Mixed load

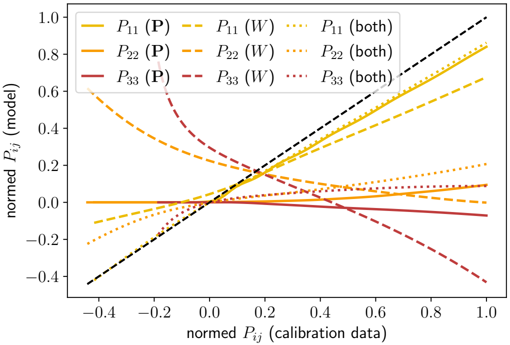


(b) Mixed load

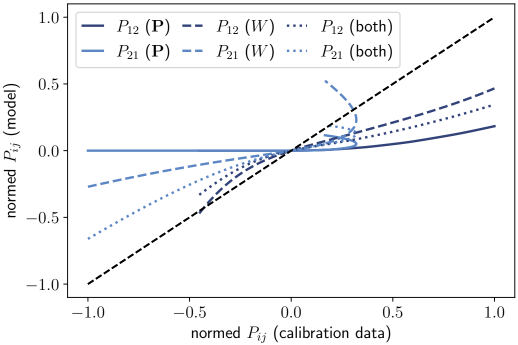
Invariant based implementation

model calibration, uniaxial load data

Extrapolation on complex load states:



(a) Uniaxial training, volumetric components



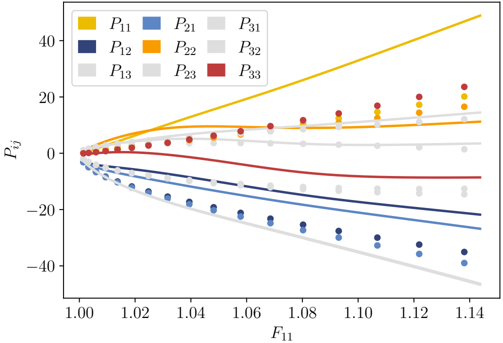
(b) Uniaxial training, shear components

Concentric sampled deformation gradient

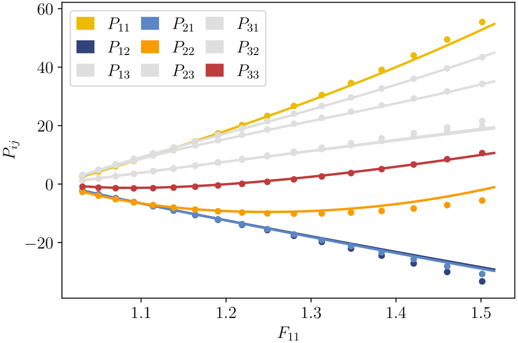
naive training

TECHNISCHE UNIVERSITÄT DARMSTADT

The naive approach is still useful, if the training data is carefully selected:



(a) Concentric sampled training data, 10 samples

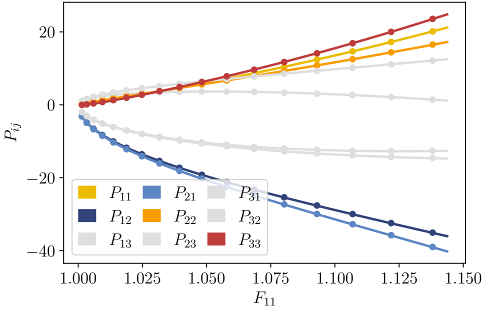


(b) Concentric sampled training data, 90 samples

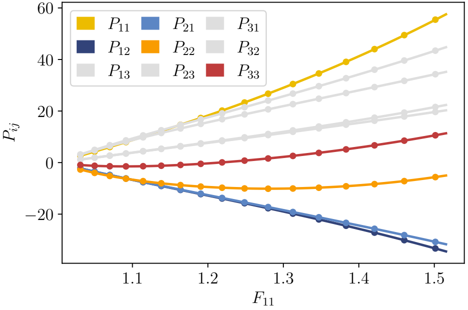
Concentric sampled deformation gradient

invariant training

Small sample size gives small deviation for invariant training:



(a) Concentric sampled training data, 10 samples



(b) Concentric sampled training data, 90 samples

Invariant training for cubic anisotropy

invariant based model

TECHNISCHE UNIVERSITÄT DARMSTADT

Recall the implementation for eq. 5, we change the invariants to:

$$\mathcal{I} = (I_1, I_2, J, -J, I_7, I_{11}) \tag{8}$$

which describes cubic anisotropic material.
We further denote I_7, I_{11} with:

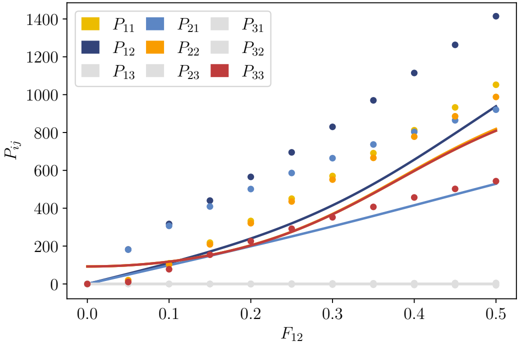
$$I_7 = \mathbf{C} : \mathbb{G}_{cub} : \mathbf{C}; \qquad I_{11} = Cof(\mathbf{C}) : \mathbb{G}_{cub} : Cof(\mathbf{C}) \tag{9}$$

Where $\mathbb{G}_{cub}$ is defined as the fourth order structural tensor:

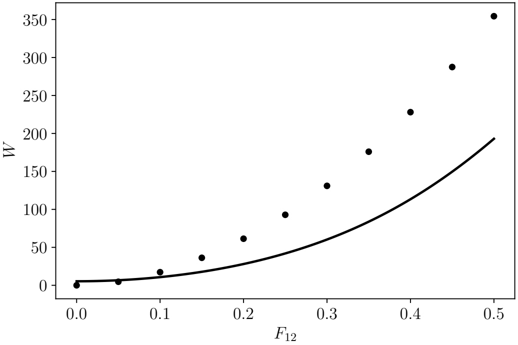
$$\mathbb{G}_{cub} = \sum_{i=1}^{3} \mathbf{e_i} \otimes \mathbf{e_i} \otimes \mathbf{e_i} \otimes \mathbf{e_i} \tag{10}$$

Invariant model for cubic anisotropy

adaption for cubic anisotropic lattice material



(a) Shear load interpolation



(b) Deformation energy

Deformation gradient based NN

observer objective training

For the training data, we use:

$$W(\mathbf{F}) = \mathcal{P}(\mathbf{F}, \mathit{Cof}\,\mathbf{F}, \mathit{det}\,\mathbf{F}) \tag{11}$$

$\Rightarrow \mathcal{P}$ is convex in its arguments.

Material objectivity is defined with:

$$W(\mathbf{QF}) = W(\mathbf{F}) \quad \forall \mathbf{F} \in GL^{+}, \mathbf{Q} \in SO(3) \tag{12}$$

$$\mathbf{P}(\mathbf{QF}) = \mathbf{QP}(\mathbf{F}) \quad \forall \mathbf{F} \in GL^{+}, \mathbf{Q} \in SO(3) \tag{13}$$

Training on the initial dataset with eq. 11, the objectivity and symmetry are lost in the model.

Deformation gradient based NN

data augmentation

TECHNISCHE UNIVERSITÄT DARMSTADT

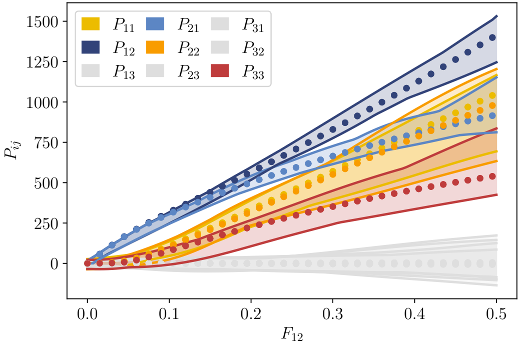
To hold objectivity, we augment the training data to restore objectivity:

$$\tilde{D} = \bigcup_{\substack{\mathbf{Q}_{obj} \in \mathcal{G}_{obj} \\ \mathbf{Q}_{mat} \in \mathcal{G}_{mat}}} \{\mathbf{Q}_{obj}\mathbf{F}\mathbf{Q}_{mat}, W, \mathbf{Q}_{obj}\mathbf{P}\mathbf{Q}_{mat}\} \tag{14}$$

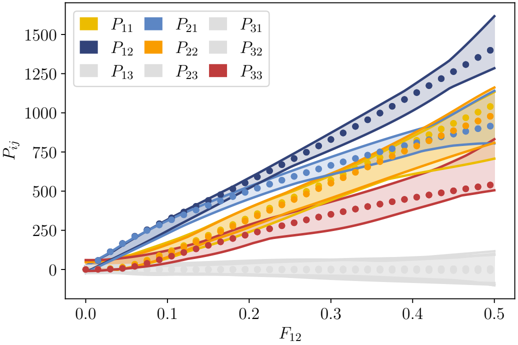
Where $\mathcal{G}_{mat}$ notes the material symmetries and $\mathcal{G}_{obj} \subset SO(3)$ is a set of arbitrary rotation matrices.

Data augmentation

cubic symmetry and shear load



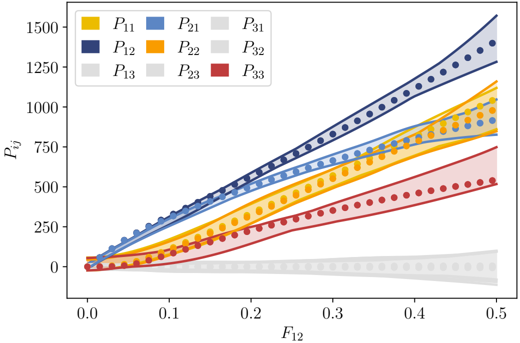
(a) 8 observers



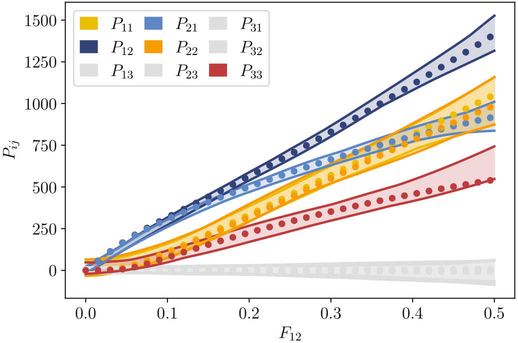
(b) 16 observers

Data augmentation

cubic symmetry and shear load



(a) 32 observers



(b) 64 observers

Resume

- FFNNs can deliver reasonable results if there is enough calibration data
- PINNs provide good models, even if they are trained on fewer data
- Calibration data should cover a wide range of stress states (interpolation instead of extrapolation)
- Better to enforce physical constraints by hard (invariant based model) than in a weak manner (deformation gradient based model)

$\hookrightarrow$ The model and architecture to choose depends on the type of application.