Explainability-driven model improvement for fault diagnosis trustworthiness

Darian M. Onchis

SMART DIASPORA WORKSHOP

West University of Timișoara
SIMT AI

April 12, 2023

Overview

Overview

Recent papers

Introduction

Deep learning models fulfill the goal of characterizing the condition of structures in an non-invasive manner by accurately classifying accelerometer data.

- The "black-box" nature of the model is a major obstacle to trustworthy operational conditions
- Many techniques have been proposed to explain machine learning models and the model's reasoning (both ante and post-hoc models)
- But these obtained insights are rarely used to improve the model.

Introduction

We shortly overview here two contributions and their trustworthy applications

- AD-HOC: Cognitive neuro-symbolic system LCNR (proposed in 2022)
- POST-HOC: Stability-fit compensation index for LIME (proposed in 2021)

Explainability experimental setup

- To create the dataset, we performed simulations on a prismatic cantilever steel beam with the length L = 1000 mm and the width B = 50 mm, which has the thickness H = 5 mm. The material is a carbon steel with mass density 7850 kg/m^3 that has the Young modulus $2 \times 10^{11} N/mm^2$.
- We created for this beam 582 damage scenarios, which consisted of 3 levels of depth and 194 different positions of the defect. The damage is an open crack with the width of 2 mm between the transverse faces. The three damage levels are as follows: level 1 is represented by the crack depth d1 = 1.25 mm, level 2 by the crack depth d2 = 2.5 mm, and level 3 by the crack depth d3 = 3.75 mm. Because these reduce the cross-section area with 25%, 50%, and 75%,
- By modal analysis, we find the frequencies of the intact beam and those of the beam with generated damage. Complete results are in our Elsevier Mendeley dataset, freely available for scientific research (data.mendeley.com/datasets/rpvh2y2dhv/)

Overview

LCNC/LCNR and Real Logic

We integrate first order logic (Real Logic) and learning based on deep convolutional networks. LCNC/LCNR converts any proposed set of Real Logic formulas into TensorFlow computational graphs, which can be used for deductive reasoning and improved learning, due to the optimization of formulas groundings.

- Constants: are grounded as a tensor of any rank
- Functions: mathematical function or any trainable one
- Quantifiers: $\exists, \forall$
- Connectors: negation, implication, disjunction, conjunction

LCNR system architecture

1. The hybrid system is composed of a deep 1D-CNN and a predicate grounded in Real Logic used to make a regression on beam damage assessment data.
2. The regression problem was solved by using a function F that took the argument of a sequential model with multiple 1D-Conv layers.
3. The axioms in this case are the result of applied aggregation operator $(\forall - pMeanError)$ of distance/similarity of the distance/similarity of F output and target data and the diagonal quantification of input and target data.
4. The learning phase of our model aims to maximize the satisfiability of the proposed formula by optimizing the groundings: $\forall(ltn.diagx, y, eq(F(x), y))$

LCNR for our dataset

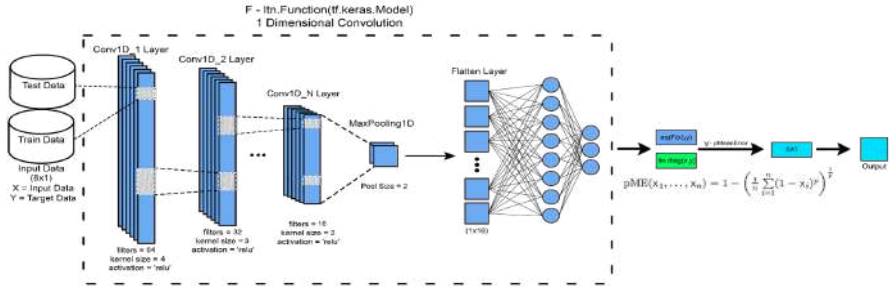


Figure 1: Logic Convolutional Neural Regressor

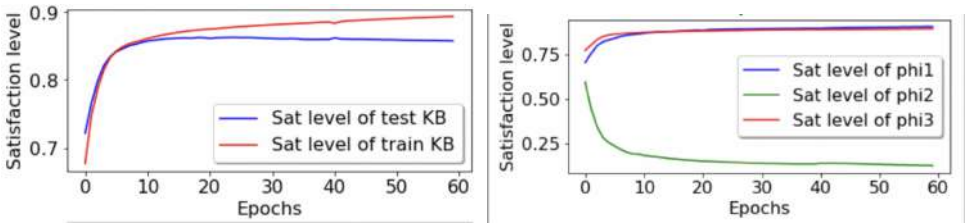
Satisfiability

Satisfiability in LCNR is similar with the one proposed by LTNs and it is a value in [0,1] which denotes the truth value of a proposed Real Logic formula. For example: the satisfaction level of the aggregation of axioms.

- Training prioritizes maximizing the value of satisfiability by adjusting the parameters of proposed formulas. It can be understood as a loss function from Machine/Deep Learning.

Overview

Satisfiability



LIME model and its drawbacks

- LIME model is sampling around the point for which it has to provide the explanations and then it uses the obtained values from the original model to create a new set of data for training an ridged regression.
- Explanations change from one run to another but this is no surprise since this fact is related to the inner workings of the LIME algorithm and the way the perturbation data is generated
- Stability is to be regarded both from a numerical and statistical point of view

SFC index

- The condition number (CN) is characterizing whether the moment matrix of the regression is ill-conditioned or not. For the ridge regression the neighborhood sampling data obtained through the LIME algorithm.
- Furthermore, we have computed the R^2 number, which is to be consider in our case a complementary measure for the global fit of the model. Usually, the R^2 number is between 0 and 1.
- R^2 is often interpreted as the proportion of response variation explained by the regressors in the model.

SFC index

With these two indicators at hand, we define the stability-fit compensation index (SFC) of the model m as a quality indicator of the explanations in the following form:

$$SFC(m) = \left[1/\left(\frac{1}{CN} \times R^2\right) - CN\right] \tag{1}$$

This formula is saying that when we have a large but still within the considered range condition number we can compensate it with a higher fit and vice versa. To be on the safe side, an acceptable SFC will be considered only between 0 and 100. Above this value, we ask for re-sampling, re-scaling and re-computation of the LIME regression.

SFC index

Algorithm 1 Explainable machine-learning AI for condition monitoring of beams

Input: S - Recorded accelerometer signal

Output: $Accuracy, XAI$ - Probabilistic response regarding the damage, Explanations

1: Generate the distributional sets with 10 natural frequencies FS
2: Split the data in training and testing data
3: Train and test the deep learning feedforward network
4: Classify the new FS with the validated $Accuracy$
5: Neighborhood sampling and compute LIME local feature predictions $LValues$
6: Compute SHAP global feature importance $SValues$ combined with $LValues$ to generate the XAI
7: Calculate the SFC index and accept XAI or return to step 5 for re-sampling if SFC>100

Figure 2: LIME and SFC Index

Dataset knowledge transfer and bias correction

- Incremental learning enables artificial agents to learn from sequential data
- We tackle class-incremental learning without memory by adapting prediction bias correction, a method which makes predictions of past and new classes more comparable.
- We introduce a two-step learning process which allows the transfer of bias correction parameters between reference and target datasets. Bias correction is first optimized offline on reference datasets which have an associated validation memory. The obtained correction parameters are then transferred to target datasets, for which no memory is available.
- The second contribution is to introduce a finer modeling of bias correction by learning its parameters per incremental state instead of the usual past vs. new class modeling.

Bias correction method

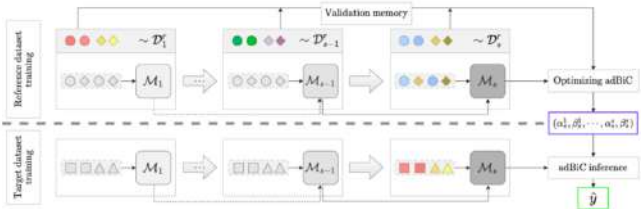


Illustration of *TransIL*, our proposed method, depicting states from 1 to s for a reference and a target dataset. The model $\mathcal{M}$ is updated in each state with data from new classes. States from 1 to $s-1$ are faded to convey the fact that knowledge learned in them is affected by catastrophic forgetting. The class IL process is first launched offline on the reference dataset where *adBiC*, our proposed bias correction layer, is trained using a validation memory which stores samples for past and new classes. Class IL is then applied to the target dataset, but without class samples shared across states since a memory is not allowed in this scenario. The set of optimal parameters of *adBiC* obtained for the reference dataset is transferred to the target dataset. This is the only information shared between the two processes and it has a negligible memory footprint. The transfer of parameters enables the use of bias correction for the target dataset. The final predictions obtained in state s are improved compared to the direct use of $\mathcal{M}_s$ predictions, since the bias in favor of new classes is reduced.

Proposed algorithms

Algorithm 1: Optimization of calibration parameters

inputs : $\mathcal{A}, \mathcal{D}_s^r$ for $s \in [\![1, S]\!]$ ▷ *reference dataset*
randomly initialize $\mathcal{M}_1$;
$\mathcal{M}_1^* \leftarrow \text{train}(\mathcal{A}; \mathcal{M}_1, \mathcal{D}_1^r)$;
for $s = 2 \ldots S$ **do**
  $\mathcal{M}_s^* \leftarrow \text{update}(\mathcal{A}; \mathcal{M}_{s-1}^*, \mathcal{D}_s^r)$;
  $\alpha_s^k \leftarrow 1,\ \beta_s^k \leftarrow 0$ for each $k \in [\![1, s]\!]$;
  foreach $(\mathbf{x}, y) \in \mathcal{D}_s^r$ ▷ *validation set*
  do
    $\mathbf{o_s} \leftarrow \mathcal{M}_s^*(\mathbf{x})$;
    for $k = 1 \ldots s$ **do**
      $\mathbf{o_s^k} \leftarrow adBiC(\boldsymbol{o_s^k}) = \alpha_s^k \boldsymbol{o_s^k} + \beta_s^k \cdot \mathbf{1}$;
    end
    $\mathbf{q_s} \leftarrow \sigma(\mathbf{o_s})$;
    loss $\leftarrow \mathcal{L}(\boldsymbol{q_s}, y)$;
    $(\alpha_s^1, \beta_s^1, \ldots, \alpha_s^s, \beta_s^s) \leftarrow$ optimize(loss) ;
  end
end

Algorithm 2: *adBiC* inference

inputs : $\mathcal{A}$, (α_s^k, β_s^k) averaged on reference datasets for each $s \in [\![1, S]\!]$, $k \in [\![1, s]\!]$
inputs : $\mathcal{D}_s^t$ for $s \in [\![1, S]\!]$ ▷ *target dataset*
randomly initialize $\mathcal{M}_1$;
$\mathcal{M}_1^* \leftarrow \text{train}(\mathcal{A}; \mathcal{M}_1, \mathcal{D}_1^t)$;
for $s = 2 \ldots S$ **do**
  $\mathcal{M}_s^* \leftarrow \text{update}(\mathcal{A}; \mathcal{M}_{s-1}^*, \mathcal{D}_s)$;
  foreach $(\mathbf{x}, y) \in \mathcal{D}_s^t$ ▷ *test set*
  do
    $\mathbf{o_s} \leftarrow \mathcal{M}_s^*(\mathbf{x})$;
    for $k = 1 \ldots s$ **do**
      $\mathbf{o_s^k} \leftarrow adBiC(\boldsymbol{o_s^k}) = \alpha_s^k \boldsymbol{o_s^k} + \beta_s^k \cdot \mathbf{1}$;
    end
    $\mathbf{q_s} \leftarrow \sigma(\mathbf{o_s})$;
    $\hat{y} \leftarrow \underset{y \in [\![1, N_s]\!]}{argmax}(\mathbf{q_s})$; ▷ *inference*
  end
end

Results I

Method	CIFAR-100			IMN-100			BIRDS-100			FOOD-100		
	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$
LwF [21]	53.0	44.0	29.1	53.8	41.1	29.2	53.7	41.8	30.1	42.9	31.8	22.2
w/ BiC	54.0 + 1.0	45.5 + 1.5	30.8 + 1.7	54.7 + 0.9	42.5 + 1.4	31.1 + 1.9	54.6 + 0.9	43.1 + 1.3	31.8 + 1.7	43.4 + 0.5	32.6 + 0.8	23.8 + 1.6
w/ adBiC	54.3 + 1.3	**46.4** + 2.4	**32.3** + 3.2	55.1 + 1.3	43.4 + 2.3	**32.3** + 3.1	55.0 + 1.3	44.0 + 2.2	**32.8** + 2.7	43.5 + 0.6	33.3 + 1.5	**24.7** + 2.5
w/ adBiC + ◎	54.9 + 1.9	47.3 + 3.3	32.6 + 3.5	55.9 + 2.1	44.2 + 3.1	33.1 + 3.9	55.8 + 2.1	44.8 + 3.0	33.3 + 3.2	44.0 + 1.1	34.2 + 2.4	25.3 + 3.1
LUCIR [14]	50.1	33.7	19.5	48.3	30.1	17.7	50.8	31.4	17.9	44.2	26.4	15.5
w/ BiC	52.5 + 2.4	37.1 + 3.4	22.4 + 2.9	54.9 + 6.6	36.8 + 6.7	21.8 + 4.1	56.0 + 5.2	37.7 + 6.3	20.6 + 2.7	49.9 + 5.7	31.5 + 5.1	17.2 + 1.7
w/ adBiC	**54.8** + 4.7	42.2 + 8.5	28.4 + 8.9	**59.0** + 10.7	**46.1** + 16.0	27.3 + 9.6	**58.5** + 7.7	**45.4** + 14.0	27.3 + 9.4	**52.0** + 7.8	**37.1** + 10.7	17.7 + 2.2
w/ adBiC + ◎	55.5 + 5.4	43.6 + 9.9	31.2 + 11.7	59.4 + 11.1	46.6 + 16.5	29.7 + 12.0	59.0 + 8.2	46.0 + 14.6	28.8 + 10.9	52.6 + 8.4	38.2 + 11.8	21.0 + 5.5
SIW [2]	29.9	22.7	14.8	32.6	23.3	15.1	30.6	23.2	14.9	29.4	21.6	14.1
w/ BiC	31.4 + 1.5	22.8 + 0.1	14.7 - 0.1	33.9 + 1.3	22.6 - 0.7	13.9 - 1.2	32.8 + 2.2	22.7 - 0.5	12.8 - 2.1	29.1 - 0.3	20.3 - 1.3	12.1 - 2.0
w/ adBiC	31.7 + 1.8	24.1 + 1.4	15.8 + 1.0	35.1 + 2.5	24.5 + 1.2	15.0 - 0.1	33.0 + 2.4	25.2 + 2.0	15.3 + 0.4	30.9 + 1.5	21.3 - 0.3	14.5 + 0.4
w/ adBiC + ◎	32.8 + 2.9	25.0 + 2.3	16.5 + 1.7	36.4 + 3.8	25.7 + 2.4	16.1 + 1.0	34.4 + 3.8	26.2 + 3.0	16.3 + 1.4	31.5 + 2.1	22.6 + 1.0	15.1 + 1.0
FT+	28.9	22.6	14.5	31.7	23.2	14.6	29.7	23.3	13.5	28.7	21.1	13.3
w/ BiC	30.7 + 1.8	22.5 - 0.1	14.8 + 0.3	33.0 + 1.3	21.9 - 1.3	13.8 - 0.8	32.3 + 2.6	22.5 - 0.8	12.4 - 1.1	28.6 - 0.1	20.6 - 0.5	11.8 - 1.5
w/ adBiC	31.9 + 3.0	23.6 + 1.0	15.0 + 0.5	34.9 + 3.2	23.7 + 0.5	15.7 + 1.1	34.0 + 4.3	25.0 + 1.7	14.2 + 0.7	30.8 + 2.1	22.2 + 1.1	14.2 + 0.9
w/ adBiC + ◎	32.5 + 3.6	24.6 + 2.0	15.9 + 1.4	35.7 + 4.0	24.9 + 1.7	16.2 + 1.6	34.5 + 4.8	25.7 + 2.4	15.4 + 1.9	31.3 + 2.6	22.7 + 1.6	14.5 + 1.2
Joint		72.7			75.5			80.9			71.03	

Table 1: Average top-1 incremental accuracy using $S = \{5, 10, 20\}$ states. Results are presented for each method without parameter transfer and with *BiC* and *adBiC* transfer. The gain (green) and loss (red) in accuracy obtained with parameter transfer are provided for each configuration. *Joint* is an upper bound obtained using a standard training with all data available. ◎ denotes a choice of the reference dataset by oracle, in which the best reference dataset for each state is selected for transfer. Best results for each setting (excluding the oracle) are in bold. A graphical view of this table is in the supplementary material.

Results II

Method	CIFAR-100			IMN-100			BIRDS-100			FOOD-100		
	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$	$S=5$	$S=10$	$S=20$
LwF [21]	41.3	33.3	23.3	45.6	33.5	23.8	44.6	34.0	23.2	29.5	23.3	17.3
w/ adBiC	42.1 + 0.8	34.8 + 1.5	25.0 + 1.7	46.7 + 1.1	35.3 + 1.8	25.6 + 1.8	45.5 + 0.9	35.4 + 1.4	25.2 + 2.0	29.9 + 0.4	24.3 + 1.0	18.7 + 1.4
LUCIR [14]	43.5	27.8	16.6	42.9	27.6	17.0	45.2	27.8	16.0	37.9	22.7	13.9
w/ adBiC	48.3 + 4.8	38.5 + 10.7	25.3 + 8.7	54.1 + 11.2	42.4 + 14.8	23.2 + 6.2	52.8 + 7.6	40.9 + 13.1	25.6 + 9.6	45.7 + 7.8	32.6 + 9.9	19.8 + 5.9
SIW [2]	31.7	21.6	13.7	32.1	22.7	14.4	29.7	22.8	14.1	28.4	18.7	13.5
w/ adBiC	33.7 + 2.0	22.5 + 0.9	14.0 + 0.3	35.0 + 2.9	22.6 - 0.1	12.2 - 2.2	32.1 + 2.4	23.7 + 0.9	13.5 - 0.6	29.9 + 1.5	16.9 - 1.8	13.3 - 0.2
FT+	30.4	21.5	12.9	31.2	22.2	12.0	29.2	22.8	12.2	27.4	18.2	11.6
w/ adBiC	32.0 + 1.6	21.4 - 0.1	13.4 + 0.5	34.8 + 3.6	21.2 - 1.0	13.7 + 1.7	31.9 + 2.7	23.0 + 0.2	13.6 + 1.4	28.8 + 1.4	16.2 - 2.0	12.2 + 0.6

Table 2: Average top-1 IL accuracy with 50% of training images for target datasets. Gains are in green, losses are in red.

$S=5$	Raw	$R=1$	$R=2$	$R=3$	$R=4$	$R=5$	$R=6$	$R=7$	$R=8$	$R=9$	$R=10$
	44.19	51.9 ± 0.4	52.0 ± 0.2	52.1 ± 0.2	52.0 ± 0.1	52.1 ± 0.1	52.0 ± 0.1	52.0 ± 0.1	52.0 ± 0.1	52.0 ± 0.1	52.0
$S=10$	Raw	$R=1$	$R=2$	$R=3$	$R=4$	$R=5$	$R=6$	$R=7$	$R=8$	$R=9$	$R=10$
	26.44	36.7 ± 0.7	36.9 ± 0.4	37.2 ± 0.4	37.2 ± 0.3	37.1 ± 0.2	37.0 ± 0.2	37.0 ± 0.1	37.1 ± 0.0	37.1 ± 0.1	37.1
$S=20$	Raw	$R=1$	$R=2$	$R=3$	$R=4$	$R=5$	$R=6$	$R=7$	$R=8$	$R=9$	$R=10$
	15.47	17.6 ± 1.2	17.5 ± 0.7	17.6 ± 0.7	17.8 ± 0.4	17.5 ± 0.3	17.7 ± 0.4	17.8 ± 0.3	17.6 ± 0.2	17.7 ± 0.1	17.7

Table 3: Average top-1 incremental accuracy of *adBiC*-corrected models trained incrementally on FOOD-100 with *LUCIR*, for $S = \{5, 10, 20\}$ states, while varying the number R of reference datasets. For $R \leq 9$, results are averaged across 10 random samplings of the reference datasets (hence the std values). *Raw* is the accuracy of *LUCIR* without bias correction.

Related research in our department

- Many colleagues from the department are involved in related research activities; the AI team is coordinated by Prof. V. Negru
- Daniela Zaharie and team: Multiple Imputation for Biomedical Data using Monte Carlo Dropout Autoencoder
- ...generative autoencoder models allow identifying the latent structure of data and generating new data consistent with this latent structure
- Gabriel Iuhasz and co-authors: Anomaly detection for fault detection in wireless community networks using machine learning
- ...they study anomaly detection techniques to discern hardware failure events in wireless community networks.

Related research in our department

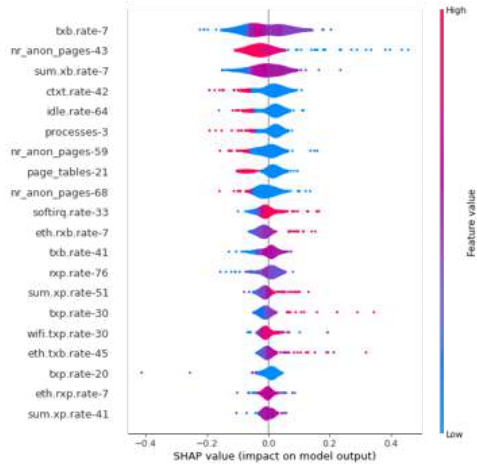


Figure 3: Anomaly detection for fault detection

⟸ Anomaly detection for fault detection in wireless community networks using machine learning

Multiple Imputation for Biomedical Data using Monte Carlo Dropout Autoencoder ⇓

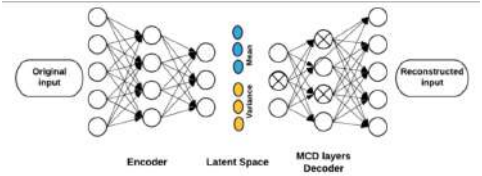


Figure 4: Multiple imputation for biomedical data


THANK YOU!