UzADL method

UzADL

- Unsupervised learning method
- Localize defected part
- Binary classification
- Principal stages:
  - Annonte unlabeled images
  - Train based on obtained images
  - Visualize defective regions

Annonte unlabeled images

- Pre-Processing:
  - Represent images in 4D Tensor
  - $X \in \mathrm{R}^{M \times C \times H \times W}$,
    - M: Total number of images
    - C: Number of channels
    - H: Image height
    - W: Image width
  - This tensor contains all the raw pix el values for the images.
  - Standardization of Pixel Values

Pseudo-Labeling

- Want to label datas in 2 class (Normal – defected)
- Extract Features from the Images
- Construct the Graph Based on Feature Similarity
  - Create a graph of images
- Form the Adjacency Matrix and Degree Matrix
- Compute the Graph Laplacian Matrix
  - Degree matrix – adjacency matrix

$$\begin{pmatrix} M_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & M_K \end{pmatrix}$$

- Obtain the largest EVL and corresponding EVT
- Save into 1D vector
- Apply a threshold to classify On largest Eigenvectors(e.g. 0)
- Assign label to X0(class N) and X1(class A)

Input: Tensor X_{std}; **Output:** X_0, y_0, X_1, y_1;
1: **Let** M_{Adj} be adjacent matrix of X_{std};
2: **Obtain** M_{Deg} based on M_{Adj};
3: **Let** $M_{GL} = M_{Deg} - M_{Adj}$;
4: **Obtain** EVL and EVT of M_{GL};
5: **Stack** largest EVTs into V_{GL};
6: **for** i in $len(V_{GL})$ **do**
7: **Cluster** v_i into c_0 and c_1;
8: **end for**
$X_0 \leftarrow []$, $y_0 \leftarrow []$, $X_1 \leftarrow []$, $y_1 \leftarrow []$
9: **for** j in $len(c_0)$ **do**
10: **Extend** X_0 with $X == c_0[j]$;
11: **Extend** y_0 with 0;
12: **end for**
13: **for** k in $len(c_1)$ **do**
14: **Extend** X_1 with $X == c_1[k]$;
15: **Extend** y_1 with 1;
16: **end for**
17: **Return** X_0, y_0, X_1, y_1.

Training process

- Use pre-trained ResNet-18 model
- Extract features from the images
- After extracting the features, develop a model of several fully connected layers
- input of this model is the feature vector from ResNet-18.
- Use a sigmoid for last layer
- Use weighted binary cross-entropy loss function is
- Use SGD optimizer

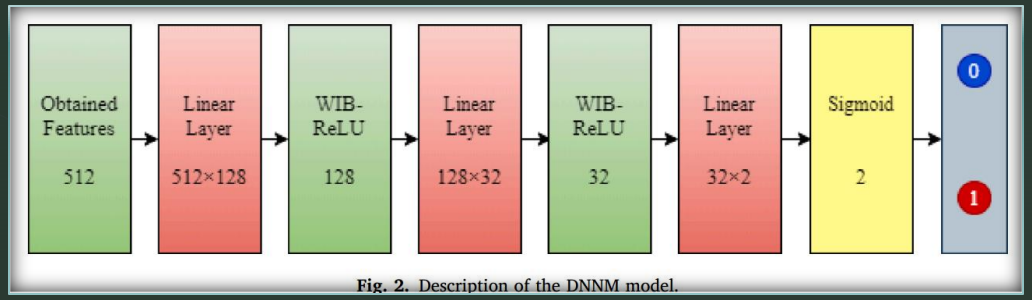
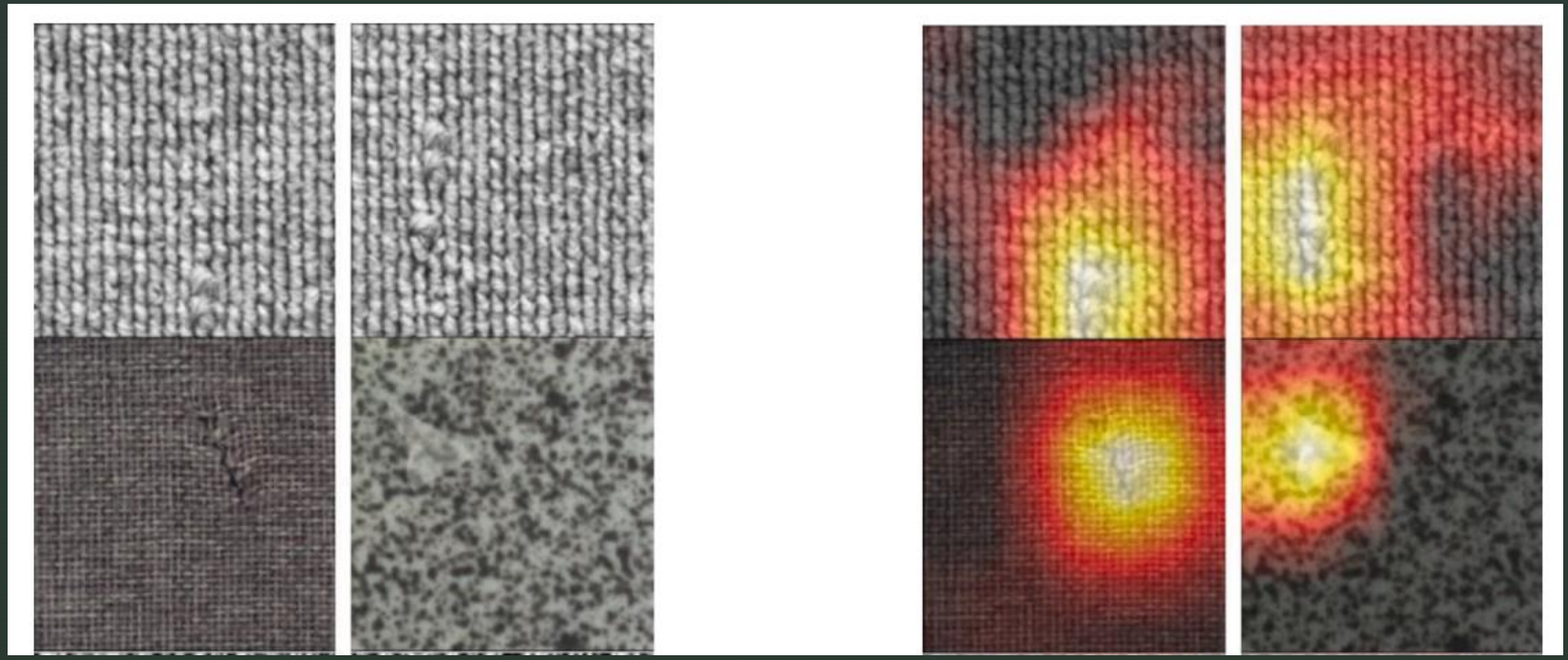


Fig. 2. Description of the DNNM model.

Defect visualization

- if the model predicts an image as defected:
  - Compute significance of each feature map
    - partial derivative of the pre-final layer’s output
  - It measures how sensitive the output is to changes in each feature map.
  - Apply Global Average Pooling On partial derivative
  - Result shows the importance of each feature map in defected class
  - heatmap highlights the regions in the image that contributed the most to the "defective" classification.

Heatmap example



Performance

Dataset Name	Model Name	AS	PS	RS	F1	AUC	Time (s)
NT	SSBD	0.942	0.926	0.978	0.952	0.870	2.034
	LDFC	0.958	0.932	**0.976**	0.954	0.886	1.901
	CTPT	0.937	0.944	0.948	0.946	0.892	1.972
	ADIC	0.832	0.819	0.829	0.824	0.792	4.281
	UNST	0.819	0.822	0.842	0.832	0.809	3.335
	UzADL	**0.984**	**0.971**	0.973	**0.972**	**0.899**	**1.224**
MVAD	SSBD	0.970	0.968	0.960	0.964	0.910	7.920
	LDFC	0.968	0.960	0.940	0.960	0.906	**5.240**
	CTPT	0.940	0.917	0.939	0.928	0.899	8.710
	ADIC	0.858	0.820	0.880	0.850	0.826	14.003
	UNST	0.892	0.865	0.835	0.850	0.843	12.827
	UzADL	**0.990**	**0.992**	**0.990**	**0.991**	**0.913**	7.164
DW	SSBD	0.972	0.921	0.939	0.930	0.892	8.802
	LDFC	0.980	0.899	**0.997**	0.948	0.905	9.124
	CTPT	0.958	0.962	0.938	0.950	0.899	9.120
	ADIC	0.903	0.895	0.891	0.893	0.850	19.573
	UNST	0.902	0.945	0.901	0.923	0.878	17.092
	UzADL	**0.996**	**0.997**	0.996	**0.996**	**0.946**	**5.530**