Automated Feature Extraction with Machine Learning and Image Processing

Prof. Dr. Stefan Bosse

University of Siegen - Dept. Maschinenbau
University of Koblenz - Dept. Computer Science

Image Analysis with CNN

CNNs are a useful class of models for both supervised and unsupervised learning paradigms.

- The CNN learns to map a given image to its corresponding category by detecting a number of abstract feature representations, ranging from simple to more complex ones.

- These discriminative features are then used within the network to predict the correct category of an input image.

Applications of Convolutional Neural Networks

1. Classification of entire images
2. Detection of objects (partial segments of an image)
3. Detection and classification of objects
4. Regression of a numerical target variable
5. Anomaly Detection (non-classified)

Layers of CNN

1. Pre-processing
2. Convolutional Layer
3. Pooling Layer
4. Fully-connected Neural Node Layer
5. Softmax Layer
6. (Transposed Convolutional Layers)

Pre-processing

Mean-subtraction

[Khan, A Guide to CNN for Computer Vision, 218]

The input patches (belonging to both train and test sets) are zero- centered by subtracting the mean computed on the entire training set. Given N training images, each denoted by x ∈ R $^{h \times w \times c}$,we can denote the mean-subtraction step as follows:

$$\hat{x}_0 = \hat{x} - \overline{x}, \overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Normalization

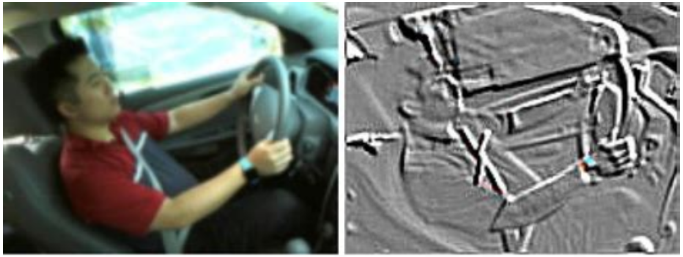
The input data (belonging to both train and test sets) is divided with the standard deviation of each input dimension (pixels in the case of an image) calculated on the training set to normalize the standard deviation to a unit value. It can be represented as follows:

$$\hat{x}_n = \frac{x_0}{\sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2}{N-1}}}$$

PCA Whitening

The aim of PCA whitening is to reduce the correlations between different data dimensions by independently normalizing them.

- This approach starts with the zero-centered data and calculates the covariance matrix which encodes the correlation between data dimensions.
- This covariance matrix is then decomposed via the Singular Value Decomposition (SVD) algorithm and the data is decorrelated by projecting it onto the eigenvectors found via SVD.
- Afterward, each dimension is divided by its corresponding eigenvalue to normalize all the respective dimensions in the data space.

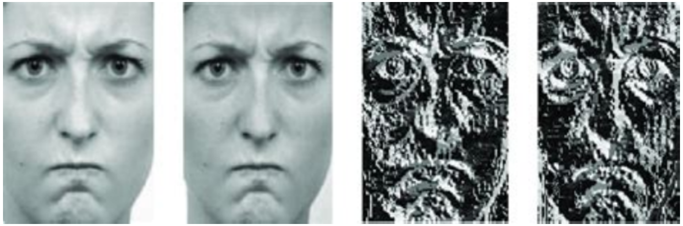


[Xuli Rao et al., 2021]

Local Contrast Normalization

This normalization scheme gets its motivation from neuroscience. As the name depicts, this approach normalizes the local contrast of the feature maps to obtain more prominent features.

- It first generates a local neighborhood for each pixel, e.g., for a unit radius eight neighboring pixels are selected.
- Afterward, the pixel is zero-centered with the mean calculated using its own and neighboring pixel values.
- Sim-ilarly, the pixel is also normalized with a standard deviation of its own and neighboring pixel values (only if the standard deviation is greater than one).
- The resulting pixel value is used for further computations.



[Sumeet Saurav et al, 2021]

Convolutional Layer

- In contrast to kernel-based filtering operations using commonly 3 × 3 two-dimensional filters, convolution can be performed here with any kernel size and dimension.
- In contrast to kernel-based filtering operations, the kernel parameters (weights) are not pre-determined. They are evolved during the ML training process.

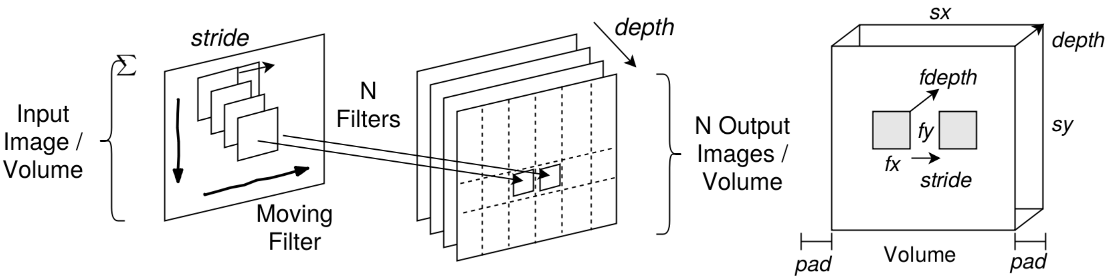


Fig. 1. Convolution with *N* filters applied to one input image (stride: shift of filter position in each dimension)

Padding, Striding, and Dilation

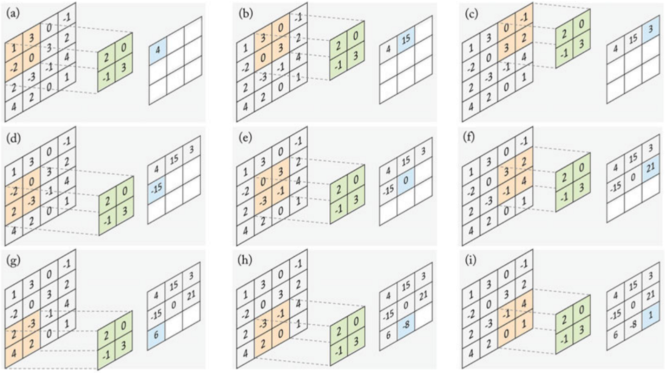


Fig. 2. The filter is slided onto the input feature map to compute the corresponding value in the output feature map. The 2 × 2 filter (shown in green) is multiplied with the same sized region (shown in orange) within a 4 × 4 input feature map and the resulting values are summed up to obtain a corresponding entry (shown in blue) in the output feature map at each convolution step. Filter Image Filter

Padding, Striding, and Dilation

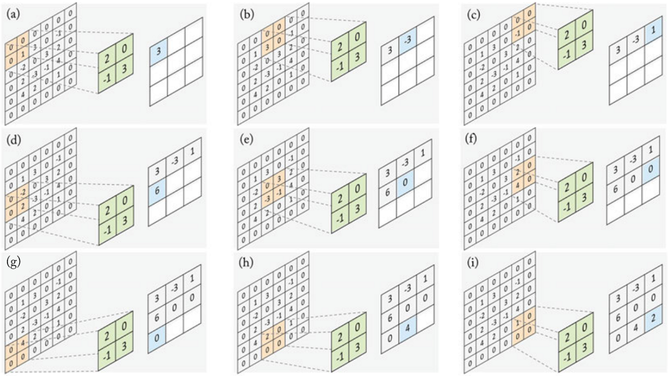


Fig. 3. Convolution layer with a zero padding of 1 and a stride of 2

For a filter with size *f* × *f* pixels, an input feature map with size *h* × *w* pixels, a stride length *s*, and zero-padding of *p*, the output feature dimensions are given by:

$$h_o = \left\lfloor \frac{h - f + s + p}{s} \right\rfloor, w_o = \left\lfloor \frac{w - f + s + p}{s} \right\rfloor$$

Padding

The padding convolutions are usually categorized into three types based on the involvement of zero-padding.

Valid Convolution is the simplest case where no zero-padding is involved. The filter always stays within “valid” positions (i.e., no zero-padded values) in the input feature map and the output size is reduced by f - 1 along the height and the width.

Same Convolution ensures that the output and input feature maps have equal (the “same”) sizes. To achieve this, inputs are zero-padded appropriately. For example, for a stride of 1, the padding is given by $p= \lfloor f/2 \rfloor$. This is why it is also called “half ” convolution.

Full Convolution applies the maximum possible padding to the input feature maps before convolution. The maximum possible padding is the one where at least one valid input value is involved in all convolution cases. Therefore, it is equivalent to padding f - 1 zeros for a filter size f so that at the extreme corners at least one valid value will be included in the convolutions.

Receptive Field

⚠ Instead of defining convolutional filters that are equal to the spatial size of the inputs, we define them to be of a significantly smaller size compared to the input images (e.g., in practice 3 × 3, 5 × 5, and 7 × 7 filters are used to process images with sizes such as 110 × 110, 224 × 224, and even larger).

Receptive Field

- This design provides two key benefits: (a) the number of learn-able parameters are greatly reduced when smaller sized kernels are used; and (b) small-sized filters ensure that distinctive patterns are learned from the local regions corresponding to, e.g., different object parts in an image.

- The size (height and width) of the filter which defines the spatial extent of a region, which a filter can modify at each convolution step, is called the “receptive field” of the filter.

  - Note that the receptive field specifically relates to the spatial dimensions of the input image/features. When

Extending the Receptive Field: Delation

In order to enable very deep models with a relatively reduced number of parameters, a successful strategy is to stack many convolution layers with small receptive field.

- However, this limits the spatial context of the learned convolutional filters which only scales linearly with the number of layers. In applications such as segmentation and labeling, which require pixel-wise dense predictions, a desirable characteristic is to aggregate broader contextual information using bigger receptive fields in the convolution layer.

> Dilated convolution is an approach which extends the receptive field size, without increasing the number of parameters.

Extending the Receptive Field: Delation

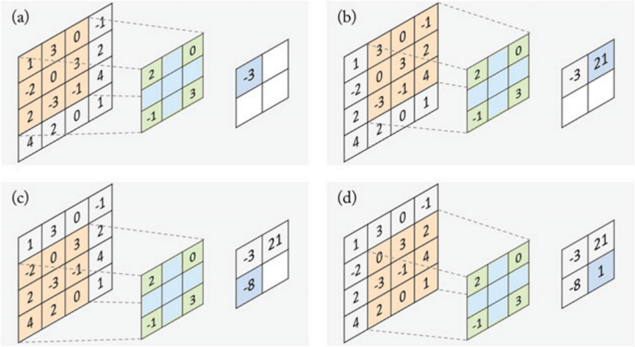


Fig. 4. Convolution with a dilated filter where the dilation factor is $d = 2$

Extending the Receptive Field: Delation

For a filter with size $f \times f$ pixels, an input feature map with size $h \times w$ pixels, a stride length s, zero-padding of p, and dilation d, the output feature dimensions are given by:

$$h_o = \frac{\left\lfloor \left(h - f - \frac{d-1}{f-1} + s + 2p \right) \right\rfloor}{s}), w_o = \frac{\left\lfloor \left(w - f - \frac{d-1}{f-1} + s + 2p \right) \right\rfloor}{s})$$

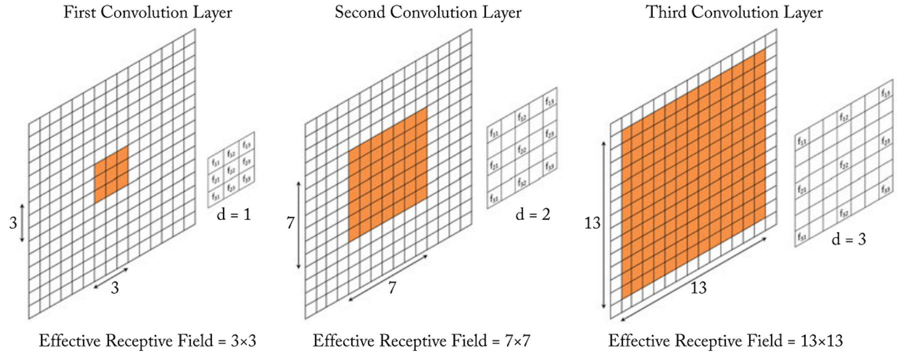

First Convolution Layer
Second Convolution Layer
Third Convolution Layer
3
3
d = 1
7
7
d = 2
13
13
d = 3
f_{11} f_{12} f_{13}
f_{21} f_{22} f_{23}
f_{31} f_{32} f_{33}
Effective Receptive Field = 3×3
Effective Receptive Field = 7×7
Effective Receptive Field = 13×13

Fig. 5. The effective receptive field with respect to the input image is shown in orange at each convolution layer.

Convolution with R

```
use math,plot
m = matrix(runif(100*100),100,100)
plot(m,auto.scale=TRUE)
k = [|
 1,0,2;
 3,1,-3;
 2,0,-1
|]
m.conv = convolution(m,k,padding=0)
print(summary(m.conv))
plot(m.conv,auto.scale=TRUE)
```

Ex. 1. Convolution operation in R(+)

Nonlinearity

- The weight layers in a CNN (e.g., convolutional and fully connected layers) are often followed by a nonlinear transfer (or a piece-wise linear) function.

- The transfer (or activation) function takes a real-valued input and squashes it within a small range such as [0; 1] or [-1; +1].

    - The application of a nonlinear function after the weight layers is highly important, since it allows a neural network to learn nonlinear mappings.
    - In the absence of nonlinearities, a stacked network of weight layers is equivalent to a linear mapping from the input domain to the output domain.

Sigmoid (logistic)

The sigmoid activation function takes in a real number as its input, and outputs a number in the range of [0,1]. It is defined as:

$$f_{\mathrm{sigm}}(x) = \frac{1}{1 + e^{-x}}$$

Tanh

The tanh activation function implements the hyperbolic tangent function to squash the input values within the range of [□1; 1]. It is represented as follows:

$$f_{\mathrm{tanh}}(x) = \frac{x}{\sqrt{1 + x^2}}$$

Rectifier Linear Unit

The ReLUis a simple activation function which is of a special practical importance because of its quick computation. A ReLU function maps the input to a 0 if it is negative and keeps its value unchanged if it is positive. This can be represented as follows:

$$f_{\mathrm{relu}}(x) = \max(0, x)$$

Noisy RELU

The noisy version of ReLU adds a sample drawn from a Gaussian distribution with mean zero and a variance which depends on the input value (σ(x)) in the positive input. It can be represented as follows:

$$f_{\mathrm{nrelu}}(x) = \max(0, x + \epsilon), \epsilon \in N(0, \sigma(x))$$

Leaky and parametric ReLU

The rectifier function completely switches off the output if the input is negative. A leaky ReLU function does not reduce the output to a zero value, rather it outputs a down-scaled version of the negative input. This function 8and more general with parameter *p*) is represented as:

$$f_{\text{p-relu}}(x) = \begin{cases} x & \text{if} \quad x \geq 0 \\ px & \text{if} \quad x \leq 0 \end{cases}$$

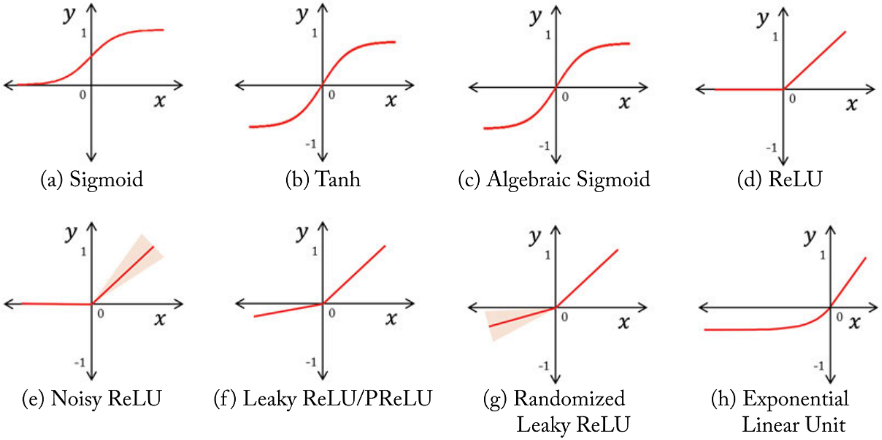


Fig. 6. Different transfer/activation functions applied to product-sums (convolutional or neural network layer)

Pooling Layer

Apooling layer operates on blocks of the input feature map and combines the feature activations. This combination operation is defined by a pooling function such as the average or the max function. Similar to the convolution layer, we need to specify the size of the pooled region and the stride.

- Convolution is pooling with a weighted sum (product sum), poolign applies different mapping functions, e.g., a maximum or relu function.
- The max pooling operation is commonly used, where the maximum activation is chosen from the selected block of values.
    - This window is slided across the input feature maps with a step size defined by the stride

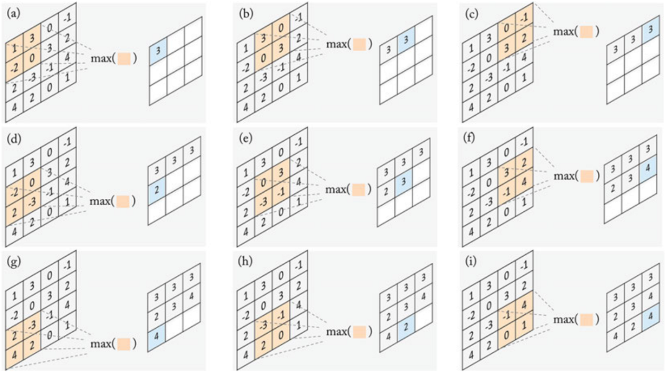


Fig. 7. The operation of max-pooling layer when the size of the pooling region is 2 × 2 and the stride is 1.

(Fully Connected) Neural network Layer

Fully connected layers correspond essentially to convolution layers with filters of size 1×1.

- Each unit in a fully connected layer is densely connected to all the units of the previous layer.
- In a typical CNN, fully-connected layers are usually placed toward the end of the architecture.
  - However, some successful architectures are reported in the literature which use this type of layer at an intermediate location within a CNN.

- Its operation can be represented as a simple matrix multiplication followed by adding a vector of bias terms and applying an element-wise nonlinear function:

$$\vec{y} = f\left(\hat{W}^T \vec{+} \vec{b}\right)$$

with ***W*** as the weights matrix and **b** as the bias vector (offset shift).

- One node of the FNN (***u*** → *v*) can be compuetd by a product sum and the application of the transfer function *f*:

$$v = f\left(\sum_{i=1}^{n} w_i u_i + b\right)$$

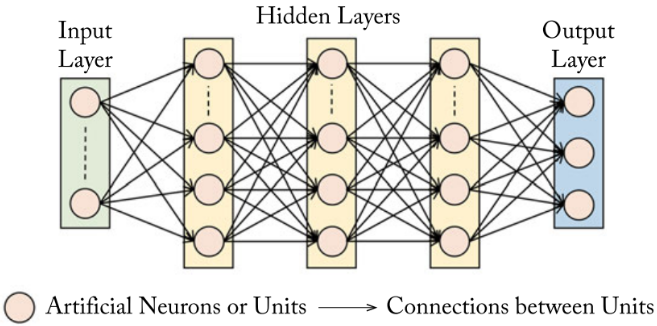


Fig. 8. A Fully-connected Neural Network architecture

Softmax Layer

- Another transfer function usefull for classification that is used in the output layer of multilayer pattern recognition networks is the *softmax* function. This transfer function has the form:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{k=1}^{|z|} e^{z_k}}$$

$$\vec{z} = (z_1, z_2, .., z_n), |z| = n$$

⚠ The outputs of the softmax transfer function can be interpreted as the probabilities associated with each class normalized with all other probabilities. Each output will fall between 0 and 1, and the sum of the outputs will equal 1.

Locality and Invariance

Locality can be defined in terms of adjacency of dimensionality of signals under a special ordering.

- A set of connections are local if they are connected to adjacent dimensions in the ordering of the signal.
  - In the case of images, this corresponds to neighboring pixels.
- One of our aims in looking at locality for images is that we have pixels that are ordered in a sequence and we want to exploit the relationship between pixels in this ordering (e.g., composing objects like pores or cracks).

There are cases, other than images, where this is also used. For instance, in the case of audio signals the ordering is by time. In the case of images, the ordering is naturally the ordering of pixels in the image itself

⚠ A classifier or regression function applied to images should be indenpendent (invariant) to absolute position, rotation, and scaling.

Training Classes

Negative Training

A predictor model is trained on a limited set of well known features, e.g., damages, but including the base-line reference (i.e., none of the damage features). This is basically supervised learning of classifiers or regression models with labelled data.

Positive Training

A generative predicotr model is trained with base-line (reference) data only containing no target features, e.g., damages. The generative model should reconstruct its input data, i.e., it is an Encoder-Decoder architecture compressing the input (e.g., an X-ray image or GUW signal) and finally decompressing the code again to reconstruct the original data (with slight difefrence). If there is a dmaage feature inside the input data, the model is not able to reconstruct the changed data, and an error occurs ⇒ Anomaly Detector. This is basically unsupervised learning.

ROI and Anomaly Detection in Radiography Data

Goal. Detect pores in Aluminum Die casted plates in X-ray radiography data automatically.

System. Industrial X-ray Radiography devices providing different resolutions and X-ray energies, prepared AluDC plates.

Methods and Algorithms. Semantic Pixel Classifier with a simple CNN, DBSCAN pixel clustering, Ellipse Fitting.

Stefan Bosse and Dirk Lehmhus. Automated Detection of hidden Damages and Impurities in Aluminum Die Casting Materials and Fibre-Metal Laminates using Low-quality X-ray Radiography, Synthetic X-ray Data Augmentation by Simulation, and Machine Learning, arXiv:2311.12041 [cs.CV] (2023)

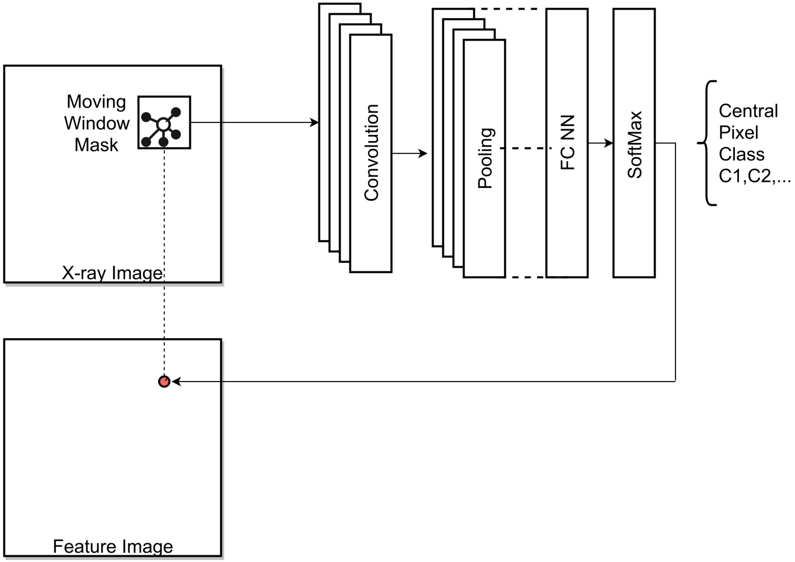
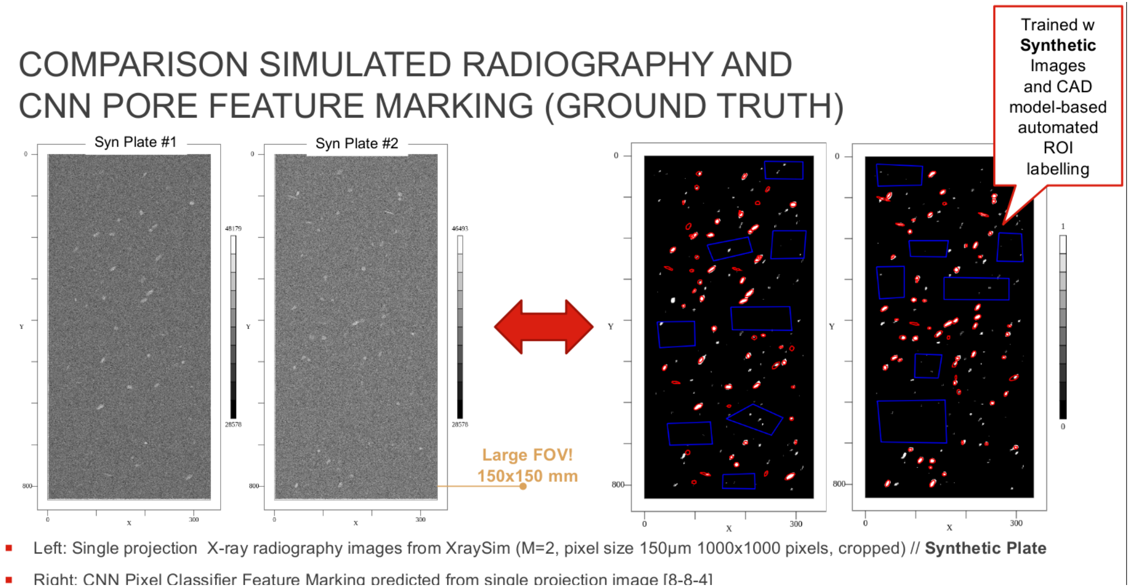


Fig. 9. Pore marking in X-ray images by using a moving window semantic pixel classifier (CNN)

ROI and Anomaly Detection in Radiography Data



COMPARISON SIMULATED RADIOGRAPHY AND CNN PORE FEATURE MARKING (GROUND TRUTH)

Trained w **Synthetic** Images and CAD model-based automated ROI labelling

Large FOV! 150x150 mm

- Left: Single projection X-ray radiography images from XraySim (M=2, pixel size 150µm 1000x1000 pixels, cropped) // **Synthetic Plate**
- Right: CNN Pixel Classifier Feature Marking predicted from single projection image [8-8-4]

Fig. 10. Examples of pore marking using a moving window semantic pixel classifier (CNN) and synthetic X-ray image data

ROI and Anomaly Detection in 3D CT Data

Goal. Detect regions of interest in CT data volumes automatically. A ROI bases on anomaly detection and is a candidate for a damages: Breakage, impurity, delamination, cracks.

System. Micro X-ray CT devices providing different resolutions and X-ray energies, prepared composite plates (e.g., GLARE).

Methods and Algorithms. Edge detection using kernel filters and gradient algorithms, Z-profiling slicing the CT volume along z-axis (depth), anomaly marking by LSTM, CNN, and SOM, threshold discrimination.

Chirag Shah, Stefan Bosse, and Axel von Hehl. Taxonomy of Damage Patterns in Composite Materials, Measuring Signals, and Methods for Automated Damage Diagnostics, Materials 15 (MDPI), no. 13 (2022): 4645

Supervised CNN

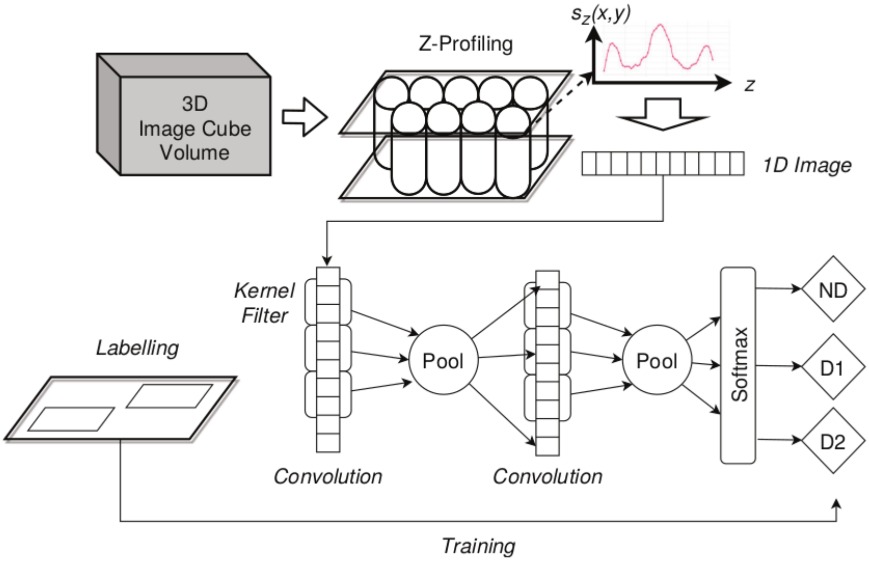


Fig. 11. Z-profile signals as 1D images as input for a CNN damage classifier (ND: No damage class, D1: Damage 1, D2: Damage 2, and so on)

Supervised CNN

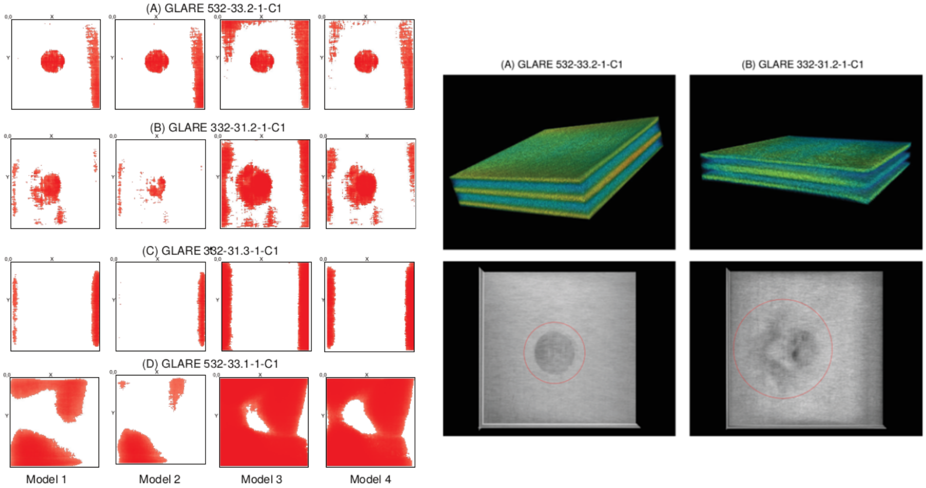


Fig. 12. (Left) Damage feature maps retrieved from four different CNN classifiers and for the specimen A (training and prediction), B, C, and D) (Right) CT image volume and selected x-y slice visualization (A-B) With centred resin defect in the PREG layer

Unsupervised SOM

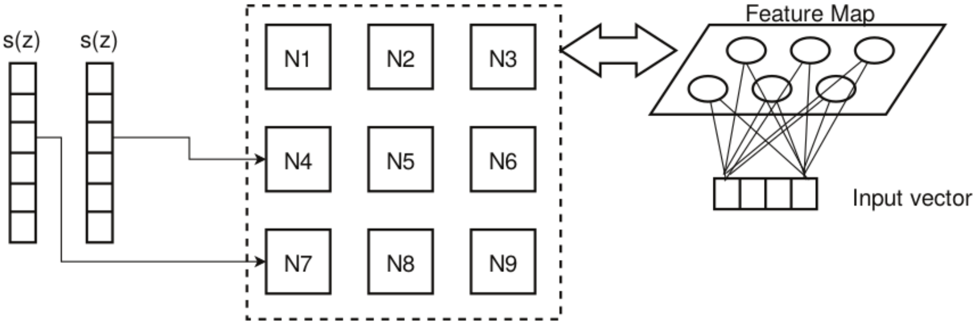


Fig. 13. Principle concept of Self-organising Maps (SOM). The neural node set {n} (squares, left side) represents a feature map {f} (circles, right side)

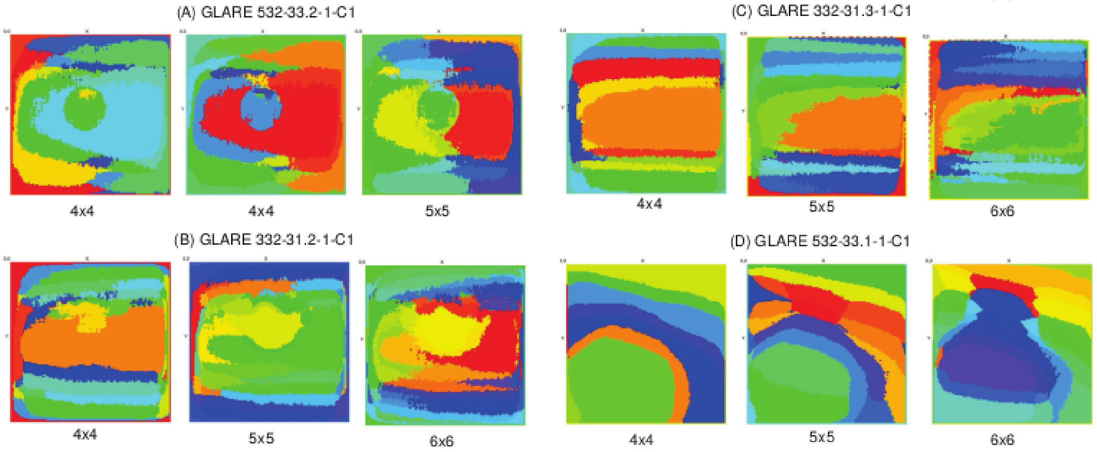


Fig. 14. SOM feature maps of the z-signal volumes for different specimen and with different SOM network sizes (rows × columns); Specimen A: Sharp resin washout, B: fuzzy resin washout; C: base-line; D: large area delamination

Summary

- Further depth reading: A Guide to ConvolutionalNeuralNetworks for Computer Vision, Khan et al., 2018
- CNN consists of different layers: Stacked convolutional layers, pooling layers, fully-connected neural layers, and softmax layers for classification.
- The CNN learns to map a given image to its corresponding category by detecting a number of abstract feature representations, ranging from simple to more complex ones.
- These discriminative features are then used within the network to predict the correct category of an input image.