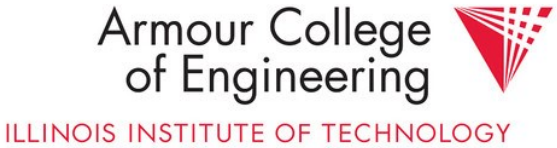

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO




# Application of Deep Learning to Classification and Regression Problems in Mechanics

Inigo Alonso Gago
Master of Engineering Management 2018-19

# Table of Contents

# Figures

# 1. INTRODUCTION

Machine Learning has already become a game-changer in many fields such as Linguistics, Image Processing or Robotics. Becoming the main research topic for the worldwide specialists in these fields.

On the other hand, other fields such as mechanics are just starting to give baby steps on their path to take full advantage of the benefits that Machine Learning could bring.

This work pretends to explore the possibilities of how mechanical engineering problems could benefit from Machine Learning. For this purpose, some of the foundation concepts in Machine Learning and more specifically, in Deep Learning, will be developed. Additionally, an example of how the application of Deep Learning to Mechanics will be provided.

# 2. GOALS

In this section, the goals of the projects are going to be defined.

- Demonstrate how deep learning can be applied to define the shape of defects on material from sensorial input.
- Explain the machine learning concepts underlying this problem

# 3. SCOPE

When dealing with such ambiguous and misleading concepts such as artificial intelligence, machine learning and deep learning it is paramount to define how they are perceived in the following lines.

As usual, an image can do a way better explanatory work than a thousand words.

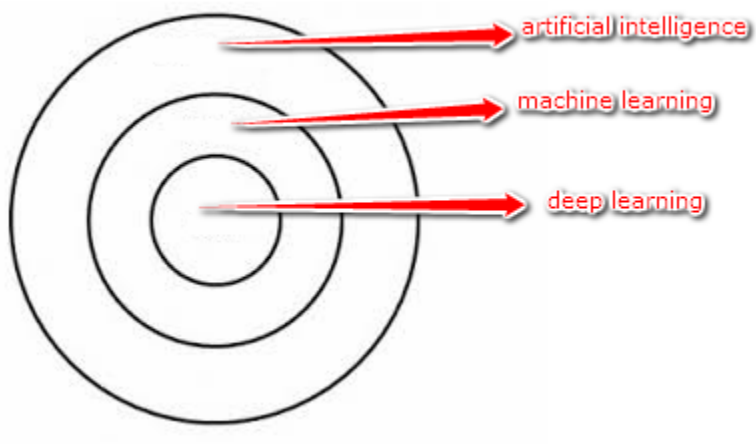



*Figure 1*. Schematic representation of each of the fields.

Deep learning is a subset of Machine Learning, which is also a subset of Artificial Intelligence.

Their respective definitions are presented below:

- Artificial Intelligence:
  It is the field that attempts to solve artificially tasks that are perceived to be unique to human intellect.
- Machine Learning:
  ML intends to enable machines to learn by themselves using the provided data and make accurate predictions. Making predictions is just a part of what human intellect can learn, that is why it is considered to be a subset of AI.
- Deep Learning:
  It is a technique of applying machine learning. This technique consisting of repeatedly stacking machine learning units one after the other.

The present work explores the application of the last of the concepts to problems in mechanics.

In the same way, the tasks that can be achieved through deep learning are 2. Supervised learning and Unsupervised Learning. To define the scope of this project these concepts have to be defined too.

- Supervised Learning:
  Learns to predict from a dataset containing features, but each example is also associated with a label or target. That is, in the case of the example of the popular Boston housing dataset, the price of the houses in Boston is predicted based on a number of features such as the number of bedrooms or the neighbourhood they are located in.

- Unsupervised Learning:
  Experiences a dataset containing many features, to learn useful properties of the structure of this dataset. An example of the unsupervised learning tasks that can be achieved is clustering, which consists of dividing the dataset into clusters of similar examples

The present goal will involve supervised learning as the goal is to predict the shape of the defect.

One last layer of specifications must be defined to understand the scope, supervised learning tasks can provide 2 different types of outputs a classification output or a regression output.

Classification tasks define a category for the input data, whereas the regression tasks provide a numerical output.

In the case of the defect shape detection problem, the shape of the defect will be defined by the coordinates of the points that were utilized to specify the contours of the defect in the original software (FEniCS). Thus, the coordinates constitute a numerical output, a regression problem.

# 4. DEEP LEARNING CONCEPTS

In this section, the most fundamental tools that have been utilized for the defect shape prediction problem, as well as a brief historical introduction to deep learning are defined.

## 2.1 HISTORY

While the birth of Artificial intelligence could be associated with the creation by IBM of a computer capable of playing the game of checker better than humans in 1952, the design by Frank Rosenblatt of the perceptron in 1957 laid the basis for machine learning as we know it today.

Until the first decade of the 21st century, machine learning remained a very niche topic whose development was stalled and could not be applied to solve any meaningful task.

Nonetheless, 3 major factors have allowed machine learning to become what it has become today:

- Hardware development:
  Early personal computers didn't have the power to train algorithms requiring large amounts of inputs. Supercomputers were highly expensive, out of the range of even small and medium organizations. Thanks to the development of GPUs for the videogame community, a pathway for the training of computationally expensive algorithms was created. Now Nvidia GPUs cluster can match the capabilities that of Supercomputer. Plus, there are GPU cloud services easily accessible to every individual. Implementing the machine learning algorithms on a GPU can speed up the training process by 10x – 100x.

- Data availability:
  The amount of data being collected these days is unprecedented, hardware memory increase and cloud storage have made it possible. There are more sensors than ever before and data to train algorithms is very available through open-source online resources.

- Better algorithms:
  The disposal of so much data has challenged the researchers and helped them getting creative to come up with innovative approaches to speed up and make more efficient the

learning. A clear example of this is what gave birth to one of the most important creations of the field, the Convolutional Neural Networks (CNN). The ImageNet challenge involved dealing with a humongous number of images, an amount for which Multilayer perceptrons were no longer efficient.

The improvements achieved during this decade have led to incredible achievements such as accurate translation of texts, creation of text, images and art or “teaching” robots how to learn.

## 2.2 THE PERCEPTRON

In order to understand how machine learning algorithms work, it needs to be understood how its most basic unit is implemented. As said, the perceptron was created in 1957, with the intention of mimicking the brain functioning with a mathematical algorithm. When it first came out, it was very promising. But over the following years, the performance didn’t exactly reach the expectations. It was studied for many years and the theory was modified and extended in a lot of ways.

The perceptron is a simple model of a neuron. It was the first step towards modelling the human brain and the biological neural network. The links between these nodes show the relationship between the nodes as well as the transmission of data and information.

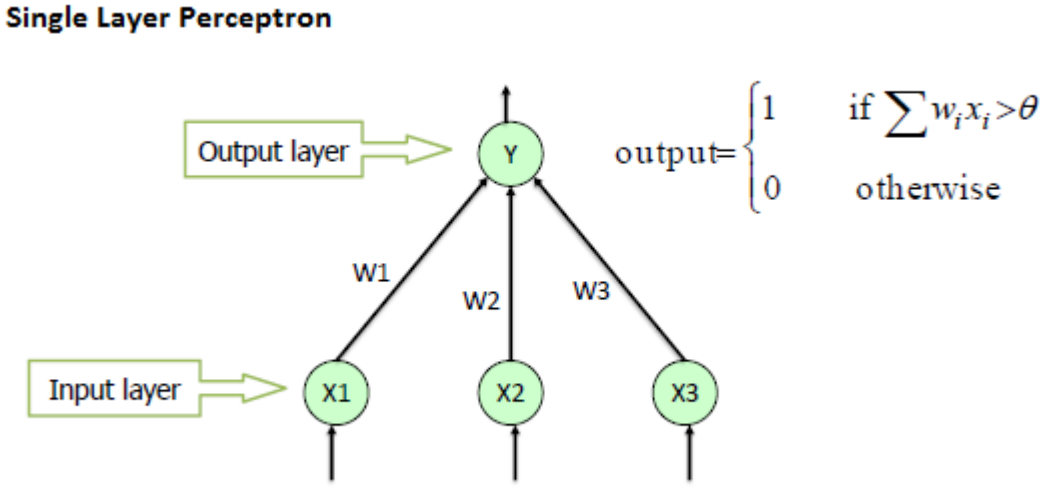



*Figure 2*. Representation of the Single Layer Perceptron

A perceptron has several external input links and one output link. A single layer perceptron (SLP) is a feed-forward network based on a threshold transfer function. SLP is the simplest type of artificial neural networks and can only classify linearly separable cases with a binary target (1, 0).

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > \theta \Rightarrow 1 \quad \text{(Output)}$$

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \leq \theta \Rightarrow 0$$

The weight parameters (w) that model the behaviour of the perceptron are learned and optimized each iteration by the mathematical algorithm known as gradient descent.

## 2.3 THE MULTILAYER PERCEPTRON

The multilayer perceptron (MLP) could be considered the hello world of deep learning.

A multilayer perceptron (MLP) is a deep, artificial neural network. It is formed by stacking more than one of the perceptrons defined above. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP.

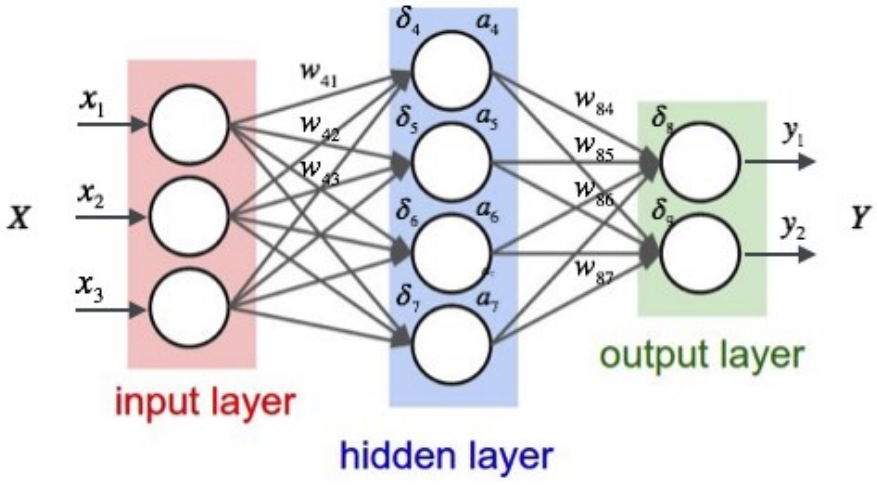



*Figure 3.* Input, hidden and output layers form MLPs.

So as to allow for learning, backpropagation is used to make those weigh and bias (internal inputs that the neurons receive) adjustments relative to the error. This error can be measured in a variety of ways.

A forward and a backward pass are defined in order to explain the behaviour of the mlp networks.

In the forward pass, the signal flow moves from the input layer through the hidden layers to the output layer making predictions, and the decision of the output layer is measured against the ground truth labels. The operations that take place during the forward pass in the neural network above are:

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

In the backward pass, using backpropagation and the chain rule of calculus, partial derivatives of the error function w.r.t. the various weights and biases are back-propagated through the MLP. The parameters of the network are updated based on the gradient descent algorithm.

It is said that the network achieves convergence when the error is limited below a limit, after repeating the forward and backward passes multiple times.

## 2.4 CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNN) were introduced in 2012 through the AlexNet architecture that won the Imagenet challenge. The margin with which this architecture outperformed the previous winner of the Imagenet competition was so significant that everyone immediately realized its potential, the AlexNet had an error rate of just 15.4% while the second-best architecture scored an error of 26.2%.

The CNNs have consistently outperformed densely connected layers consistently since then. The reason being 2 mainly:

- They reduce the number of units in the network (since they are many-to-one mappings). This means, there are fewer parameters to learn which reduces the chance of overfitting as the model would be less complex than a fully connected network.
- They share information in small neighbourhoods. This feature has made them extremely efficient in diverse applications such as image, video and text as the neighbouring inputs (eg pixels, frames, words, etc) usually carry related information.

CNNs basically use an easier way of processing data, without losing information.

The operations taking place in the convolutional neural networks are mainly two: convolution and pooling operations.

## Convolutional Layer

The convolution operation that happens on CNNs is very similar (not exactly the same) to the one taking place on the mathematical operation known as *convolution*. A dot product is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel. This filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

The weights of the filter are translation independent, which makes sense as all the filter has to do is a single task, identify patterns on a domain that shares features.

The most common example to explain their modus operandi is the image processing one, a 2D domain (that can have 1 channel, when it is grayscale or 3, the case of RGB, for example), the value of each pixel in the matrix will range from 0 to 255 – zero indicating black and 255 indicating white.

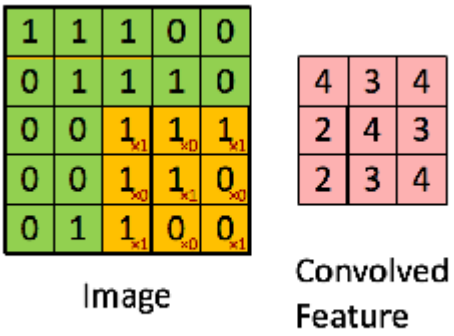



*Figure 4*. Example of the results after the convolutional 3x3 layer.

As shown in Figure 5, this reduces the dimensionality of our feature map.

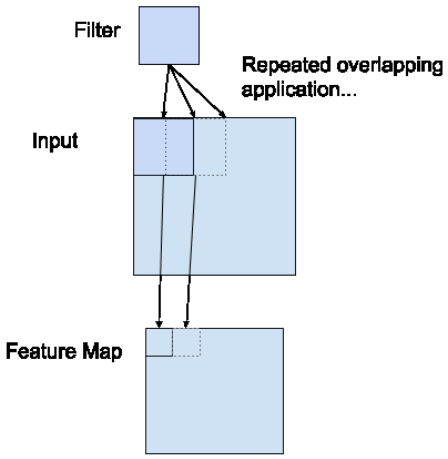



*Figure 5*. The creation of feature maps explained.

In practice, a CNN is able to learn the values of these filters that define the feature map during the training process. Nevertheless, it is still needed to specify parameters such as the number of filters, the filter size, the architecture of the network etc. before the training process.

These are some parameters that need to be specified in this network architectures:

- The number of filters:
  The larger the number of filters we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images.

- Depth:

  Depth corresponds to the number of filters we use for the convolution operation

- Stride:

  Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. The larger the stride, the smaller the produced feature maps.

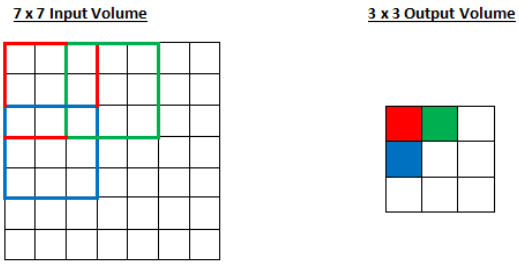



*Figure 6.* Example of a stride of size 2 over a 7x7 domain and a 3x3 kernel.

- Zero-padding:

  This will avoid the consequent dimensionality reduction mentioned above. This is achieved by padding the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. Zero padding allows us to control the size of the feature maps.

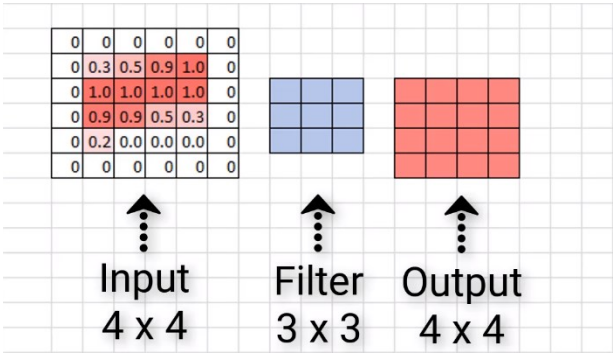



*Figure 7.* Zero Padding over a 4x4 domain.

## Pooling Layer

The aim of this layer is reducing the spatial size of the convolution operation performed. What this does is to decrease the computational power required to process the data through dimensionality reduction. To achieve this dimensionality reduction without losing meaningful information, pooling layers extract dominant features.

The 2 main pooling layer types, depending on how they extract the features are Max Pooling and Average Pooling. Max Pooling provides the maximum value from the analyzed portion, while Average Pooling returns the average of the values in the kernel.

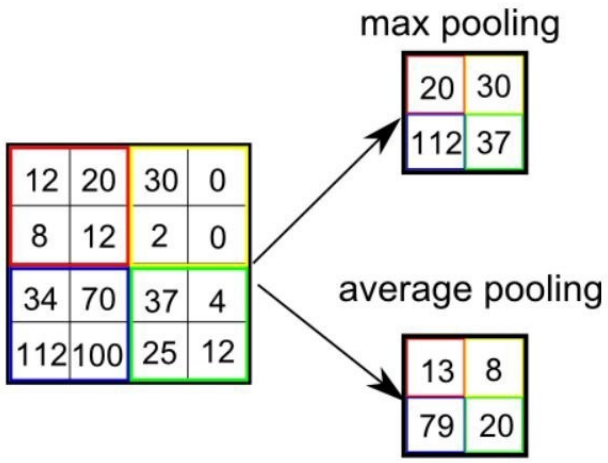



*Figure 8*. Example of the results when Max Pooling vs Average Pooling

## Final Fully Connected Layers

So as to perform the final regression or classification, at least one fully connected layer (FCL) of MLP is introduced. These layers are a cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The FCL will be able to learn non-linear relations that yield accurate output.

## Summary

The following figure shows how the components of the CNNs are put together:

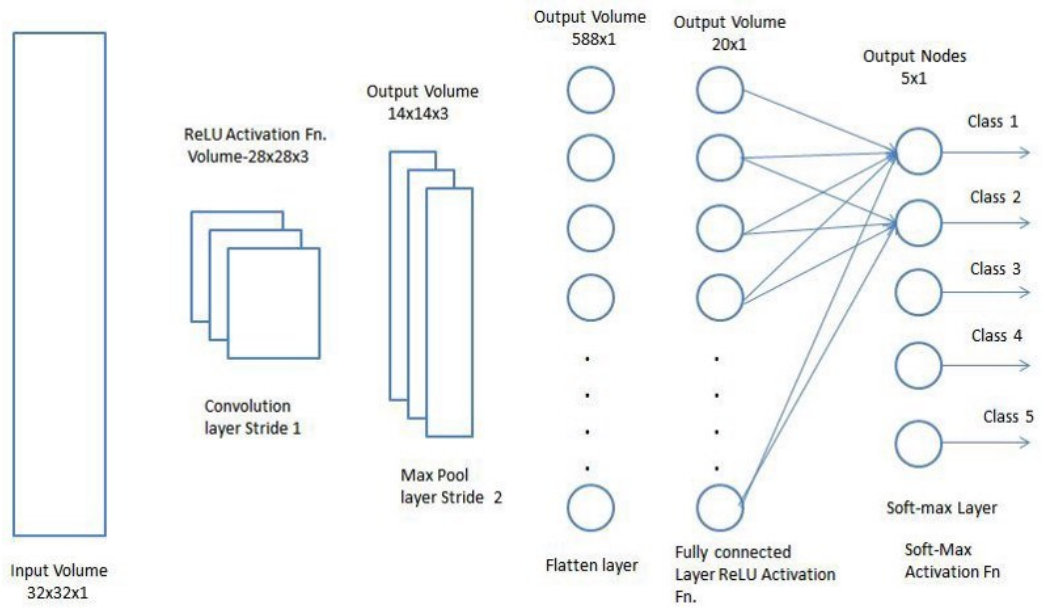



*Figure 9.* Summary of the most common architecture for CNNs.

The input is subjected to the convolutional transformation. This transformation may or may not reduce the dimensionality (in the figure it does), this occurrence will depend on the zero padding. The convolutional layer is generally followed by a pooling layer that reduces the dimensionality extracting the most meaningful results that the convolutional layer has provided.

The set of convolutional layer + pooling layer is repeated over and over again depending on the complexity of the task to be tackled, the larger the complexity, the bigger the number of layers required.

The combination of convolutional and pooling layers is eventually flattened into 1 dimension. This dimension is the input for the sequence of fully connected layers that provide an output.

In the figure, a softmax layer is included in the end. This is required for classification tasks, but as it will be shown below our problem will not need it.

# 5. DEEP LEARNING FOR DEFECT SHAPE DETECTION IN MATERIALS

## 5.1 PROBLEM OVERVIEW

Detecting defects in materials and identifying their particular shapes has remained a challenge for which a variety of engineering tools have been developed. The most common of these methods are Magnetic Particle Testing, Liquid Penetrating Testing, Eddy Current Testing and Eddy Current Array, these approaches have something in common, they are time-consuming and expensive. Especially, compared to the approach based on deep learning presented in the following lines.

## 5.2 PROPOSED SOLUTION

The methodology that will be discussed attempts to identify and detect the shape of the defect by studying how signals evolve after interfering with the defect.

When the signal comes across a defect it will react in a particular way, it will vanish or fluctuate differently depending on the shape of the defect.

This variation will be captured by some sensor and will be post-processed by an Artificial Intelligence that will predict the shape of the defect. The technology is trained and tested on 100,000 examples generated by the simulations performed in the Finite Element Analysis package FEniCS.

Here are a couple of examples of the shapes of the defects generated by FEniCS that will be input to the network:

*Figure 10.* Examples of the actual error shapes.

*Figure 11.* Examples of the actual error shapes.

## 5.3 SIMULATION SET UP

The 100,000 examples are generated on FEniCS following the next instructions:

- A 2D domain is used for the simulations.
- The domain consists of a squared shape.
- Each of the shapes on the square has 8 sensors on it, constituting a total of 32 sensors per simulation.
- The defect is generated with 7 random control points.
- There is a source that generates the signal inside the domain, this source does not change its location for each example.

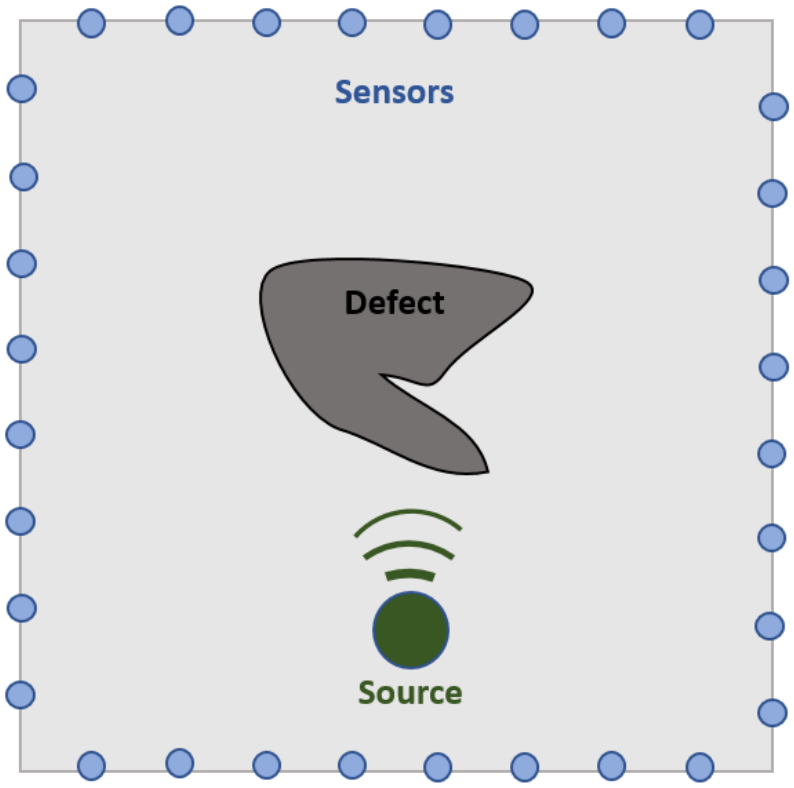



*Figure 12.* Set up of the FEniCS simulations.

## 5.4 DEEP LEARNING ARCHITECTURE

The input of the networks is the signal intensity, captured over 193 timesteps in the 32 sensors, constituting a 193 x 32 input that will contain on each row the data for each timestep.

The output will be the coordinates of the 7 control points that define the spline that characterizes the shape of the defect.

As the goal is to analyze the evolution over each time step, a 1D convolutional network, with a stride of one that analyzes the whole 32 sensor data one by one for every timestep, will do the work. The job done by each 1D CNN is summed up in Figure 13.

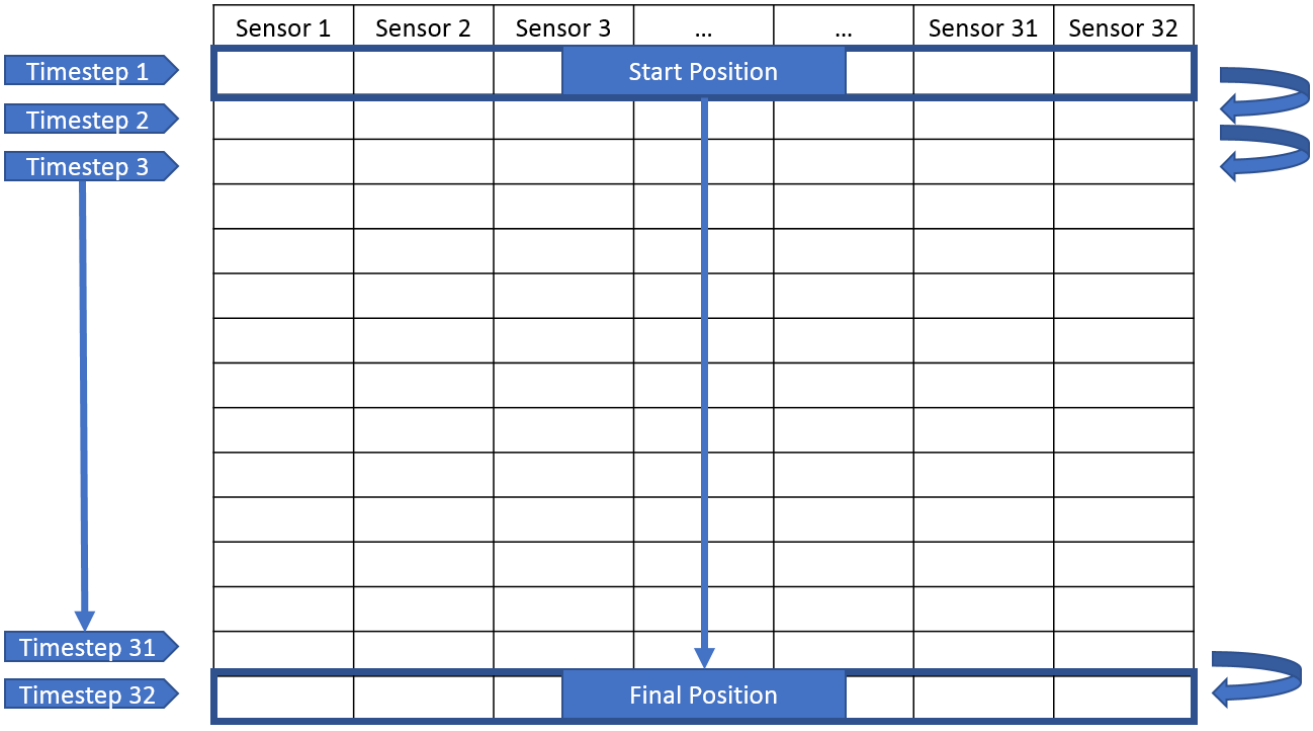



*Figure 13.* 1-dimensional convolution over the time domain.

The with a kernel of size 3, a stride of one and applying the same padding technique (to keep the dimensionality), the convolution will repeat itself a total of 4 times.

The first 2 times are followed by a max-pooling for each 2 data points, the same block structure is repeated after that.

Once the convolution operations are finished the information is flattened to be ready to go through the last layers of MLPs.

A dropout regularization technique is applied after the 2nd and 4th convolutional layers and after every densely connected layer.

The optimizer that it is used is ADAM optimizer, with its hyperparameters set to the values recommended in the original paper, whereas the loss function that is set for optimization is the mean squared error loss function.

$$\sum_{i=1}^{n} \frac{\left(w^T x(i) - y(i)\right)^2}{n}$$

**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Require:** $\theta_0$: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
**while** $\theta_t$ not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
**end while**
**return** $\theta_t$ (Resulting parameters)

*Figure 14.* Equation of the mean squared error and pseudo-code for the ADAM optimizer.

The activation functions are RELUs for every layer except for the last one, where due to the regressive nature of the problem, no activation function is applied, this way an unbounded numerical output is guaranteed.

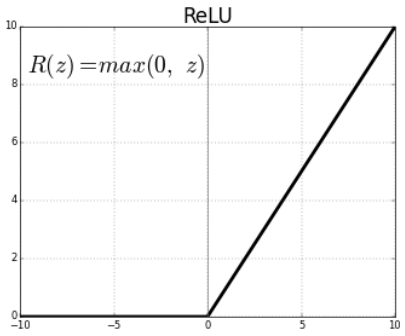



*Figure 15.* RELU activation values.

The summary provided by Keras is the following:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv1d_12 (Conv1D) | (None, 193, 64) | 6208 |
| conv1d_13 (Conv1D) | (None, 193, 64) | 12352 |
| max_pooling1d_6 (MaxPooling1 | (None, 96, 64) | 0 |
| dropout_11 (Dropout) | (None, 96, 64) | 0 |
| conv1d_14 (Conv1D) | (None, 96, 128) | 24704 |
| conv1d_15 (Conv1D) | (None, 96, 128) | 49280 |
| max_pooling1d_7 (MaxPooling1 | (None, 48, 128) | 0 |
| dropout_12 (Dropout) | (None, 48, 128) | 0 |
| flatten_3 (Flatten) | (None, 6144) | 0 |
| dense_9 (Dense) | (None, 1024) | 6292480 |
| dropout_13 (Dropout) | (None, 1024) | 0 |
| dense_10 (Dense) | (None, 512) | 524800 |
| dropout_14 (Dropout) | (None, 512) | 0 |
| dense_11 (Dense) | (None, 512) | 262656 |
| dropout_15 (Dropout) | (None, 512) | 0 |
| dense_12 (Dense) | (None, 14) | 7182 |

Total params: 7,179,662
Trainable params: 7,179,662
Non-trainable params: 0

*Figure 16.* Summary of the architecture provided by Keras.

## 5.5 RESULTS

As it is shown the model succeeds in finding the seven control points that define the shapes of the defects. Been capable of reproducing defect shapes based solely on sensor data and without any visual information.

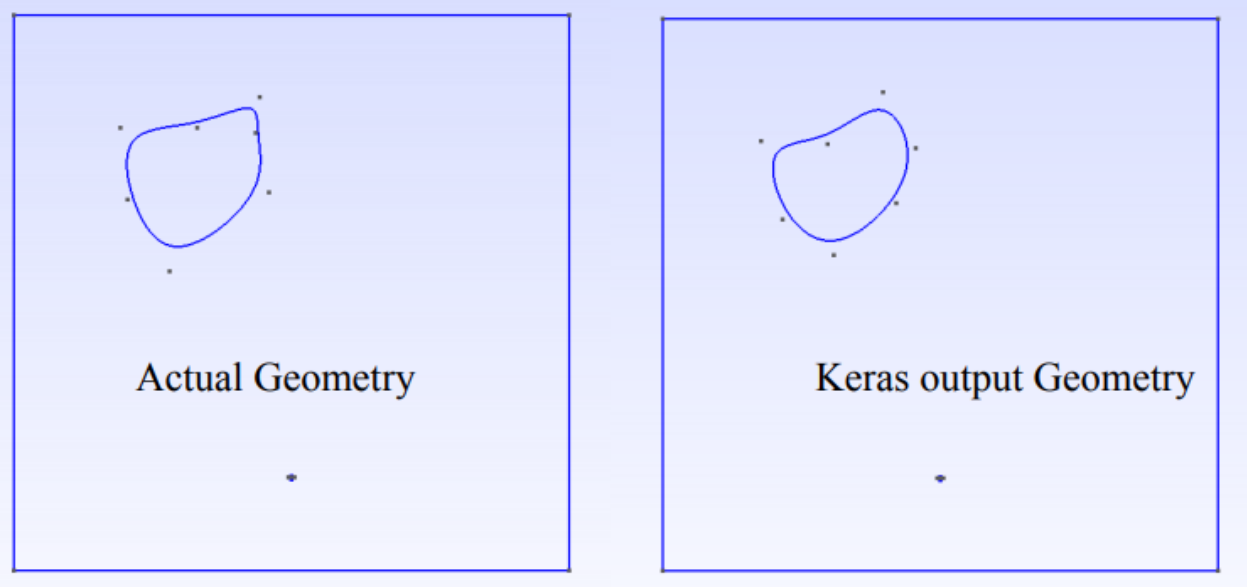



*Figure 17.* Example of the real shape versus the predicted shape.

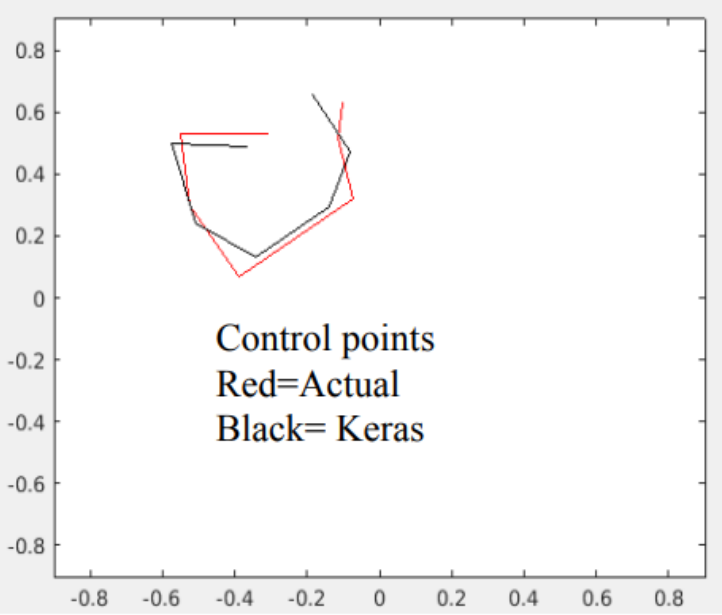



*Figure 18.* Comparison of the location of the original versus the predicted shape control points.

# 6. CONCLUSIONS

While still to be unleashed, Artificial Intelligence has a lot of potential in the field of solid mechanics. A couple of conclusions that may inspire future work are the following:

- Especially in applications involving a large number of data or simulations it could be great for pattern identification.
- The use of machine learning in mechanics goes beyond the possibilities to simplify numerical algorithms.
- Currently, the most relevant information concerning the application of artificial intelligence to mechanics can be found in what is known as *Physics-Informed Neural Networks* a series of successful applications of machine learning to solve some equations in physics.