CHALMERS
UNIVERSITY OF TECHNOLOGY

Vehicle Occupant Kinematics prediction using Machine Learning

A study to understand applications of machine learning in prediction of vehicle occupant kinematics during crash and pre-crash

Master's thesis in Automotive Engineering

SHIVAPRASAD GURRAM

VENKATA SAI KUMAR REDDY

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2021

www.chalmers.se

Master's thesis 2021:70

Vehicle Occupant Kinematics prediction using Machine Learning

A study to understand applications of machine learning in prediction of vehicle occupant kinematics during crash and pre-crash

Shivaprasad Gurram
Venkata Sai Kumar Reddy

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden 2021

Vehicle Occupant Kinematics prediction using Machine Learning
A study to understand applications of machine learning in prediction of vehicle occupant kinematics during crash and pre-crash
SHIVAPRASAD GURRAM
VENKATA SAI KUMAR REDDY

Supervisor: Johan Iraeus, Mechanics and Maritime Sciences, Chalmers
Co-Supervisors: Jobin John, Chalmers
Karl-Johan Larsson, Autoliv
Examiner: Johan Davidsson, Mechanics and Maritime Sciences, Chalmers

Master's Thesis 2021
Department of Mechanics and Maritime Sciences
Division of Vehicle Safety
Injury Prevention
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Hybrid III M50 fast dummy Finite Element model in crash and a representative machine learning prediction of the same.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Vehicle Occupant Kinematics prediction using Machine Learning
A study to understand applications of ML in the prediction of vehicle occupant kinematics during crash and pre-crash
SHIVAPRASAD GURRAM
VENKATA SAI KUMAR REDDY
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

This work presents surrogate models using deep learning and statistical techniques for predictive modeling of occupant kinematics in crash scenarios. Finite Element Methods (FEM) involving discretization of Partial Differential Equations (PDEs) are widely used for vehicle crash and injury analysis involving Finite Element human body models (FE-HBMs). Doing a parametric study of pre-cash and in-crash scenarios involving a complex FE-HBM model depicting a detailed representation of the human anatomy is computationally expensive and time consuming, mainly due to the long pre-crash event. This thesis aims at evaluating machine learning models along with dimensionality reduction tools as the inexpensive surrogates for determining the kinematics of the vehicle occupants.

A FE model of the Hybrid III crash test dummy was used to obtain nodal displacement outputs of the dummy for crash scenarios which in turn are the outputs predicted from the surrogates. These outputs from FE simulations form a large data set in high dimensional format. Performing machine learning with this large data set can become complex. This thesis utilizes one-dimensional convolutional autoencoder (AE-1D) and Principle component analysis (PCA) to obtain a compressed version of the original data. Supervised learning is used to train the surrogates on this low dimensional space with a set of crash parameters acting as inputs. Gaussian process regression (GPR), Feedforward neural networks (FFNN) and Random forest regression (RFR) were used as the machine learning models to build the surrogates. LS-Dyna FE solver was used to generate the sample results for a set of crash parameters. We use in-crash kinematics instead of pre-crash due to the computational limitations in simulating the required samples. For the same reason, a complex FE Human body model such as SAFER or THUMS model was not used.

This thesis provides an understanding of using statistical and neural network based dimensionality reduction methods paired with different ML models on the accuracy of the kinematics predictions. The accuracy of the predictions varies considerably depending on the use of PCA or AE-1D for dimensionality reduction and also the sample size. PCA and AE-1D have similar performance in compressing and decompressing the original data. AE-1D paired with either FFNN or Gaussian process provided the most accurate predictions in most sample sizes.

Keywords: FEM, Human body model, Hybrid III model, SAFER, machine learning, Convolutional Autoencoder, neural networks, Principal component analysis, Gaussian process, surrogate modelling.

Acknowledgements

This master thesis was carried out at the Department of Mechanics and Maritime Sciences along with Autoliv. We thank our supervisors Johan Iraeus along with co-supervisors Jobin John and Karl-Johan Larsson for the continuous support and guidance and our examiner Johan Davidsson. We thank you for all the valuable discussions and the time spent on clearing our doubts throughout. Furthermore, we are thankful to coderefinery and NVIDIA/ENCCS for giving us an opportunity to attend the seminar on AI and software testing. We would like to thank DynaMore Nordic for providing us with an introductory course in LS-DYNA.

Shivaprasad Gurram, Gothenburg, August 2021

Venkata Sai Kumar Reddy, Gothenburg, August 2021

Contents

List of Figures

List of Tables

1
Introduction

Road crash fatalities are considered a major health concern on a global scale as nearly 1.35 million people are killed each year. Additionally, 20-50 million people are non-fatally injured but left with long-term disabilities [1]. Fatalities, in general, are reducing in most developed countries and are on a rise in developing countries. This in part is due to the strict regulations and their enforcement for manufacturers and users in the case of the developed countries and not necessarily the case in the developing countries along with an increase in vehicles on road[2]. It is crucial for automakers to conduct research and development (R&D) in a fast and efficient manner, especially in terms of adhering to safety norms and trying to do their part in making vehicles safer. One of the requirements of building a safe car is to test its crashworthiness[1]. The traditional method of testing this aspect is the physical destructive testing of prototypes. This is a very intensive process in terms of time, cost and complexity to achieve and verify the crashworthiness of vehicles. In recent years, FEM is used as a virtual method to perform and verify this destructive testing of vehicles for crashworthiness before committing to a physical test. Due to the advances in computational capabilities, highly detailed crash simulations reduced the number of physical tests needed to achieve desired crashworthiness[3].

A vehicle crash scenario is split into pre-crash, in-crash and post-crash phases[4]. The pre-crash phase consists of sub-phases, a non-conflict region that defines a time frame before a crash where there is a low to high risk of a possible collision of vehicles and a conflict region in which a collision might occur unless an intervention or corrective action is not taken by the driver. During the conflict phase, safety systems issue warnings of various levels and tries to intervene and avoid a crash. At the end of the conflict phase, after a point of no return, active safety systems try to mitigate injury by activating or priming some passive safety systems like airbags and seat belt pre-tension etc. Once the crash is initiated, all passive systems come into play to minimize injury to the occupants. The post-crash phase is when the crash event has ended and the vehicle is at a halt[4].

A typical crash phase is in the order of 100-150 ms whereas a pre-crash scenario can last upwards of 1-2 seconds. Crash test standards which use Anthropometric Test Devices (ATDs) as human occupant surrogates have them seated in standard driving or passenger positions. In reality, this is not always the case when a vehicle enters a crash i.e., pre-crash phase actions such as swerving, braking etc can change

[1]Crashworthiness is a term used to describe the ability of an automobile to safeguard its occupants in crashes

the occupant position away from the standard driving positions. Understanding this behaviour can be crucial in injury prediction, such as arm position in thorax injury determination in a side impact scenario [25]. Understanding the behaviour of the occupants during pre-crash could also assist in identifying potential occupant positions for the assessment and optimization of critical adaptive or active safety systems [5].

Simulation of the pre-crash phase is a computationally expensive operation. For example, a crash simulation of 200ms utilizing a complex THUMS V3.0 model with complex FE internal organ models [2] took nearly 16h on 4 CPUs of a PC[3]. A pre-crash of 300ms ending with a 50kmph crash combination totalling 500 ms to study the kinematics and injury outcomes took a simulation time of 101 h on 4 CPUs [6]. To build safe and crashworthy vehicles it is important to conduct such studies and FE simulations are a crucial part of the same. It is important to represent the various postures the occupant might end up during the pre-crash phase due to braking, swerving, distractions etc using the FE-human body models (FE-HBMs) along with the various demographics of age, gender and anthropometry [7]. Performing enough simulations to cover all aspects mentioned will be a huge task and any methods to reduce the time to build safe cars is an advantage.

Determining the pre-crash phase kinematic time histories via a machine learning (ML) model based on previous pre-crash simulation data would lead to shorter turnaround times in many occupant safety-related tasks such as safety device designs, optimization of active and passive safety systems etc. The methods that are explained in this paper are aimed at addressing this possibility. This work uses in-crash kinematics to understand and build the ML models as the computational resources and time needed to generate the required samples was limited.

1.1 Background

Many engineering applications make use of mathematical models expressed as parametrized partial differential equations (PDEs). Advances in computational mechanics have allowed the development and use of high-fidelity complex physical and biomechanical human models used in vehicle safety evaluations via discretization procedures such as finite element(FE) methods. The finite element method (FEM) is a popular numerical method used to solve PDEs with complex geometric domains, a multitude of boundary conditions and non-linear material properties etc. It has numerous applications in mechanical engineering, thermal and fluid flow, electromagnetism, biomedical and biomechanical, design of vehicles, products and aircraft etc [26].

One of the applications of the FEM is in the automotive domain, to reduce the

[2] A model with 281,260 elements and 184,242 nodes

[3] FUJITSU PRIMERGY BX922 S2 (3.33 GHz Xeon X5680 processors with 48 GB of installed RAM

development time and cost of intensive destructive physical crash tests. These tests are used to assess vehicle safety characteristics and to meet the government specified minimum requirements. One aim of using FEM is early detection of component failure and also for optimization of component designs. Occupant safety depends on various aspects of the vehicle such as the vehicle structure, interior design, restraint design (airbag, seat belt etc.,) seating position of the occupants etc., all of whose effects can be studied and designs optimized using FEM simulations. FE crash simulations can be parametrized with many variables such as vehicle velocity, angle of impact, material properties of components etc.

The transition to autonomous vehicles puts the driver in the position of a passenger allowing them to be free to do other activities and thus altering their behaviour and reactions from the norm of hand on steering wheel driving posture. They can be more relaxed and unaware of their environment and other vehicles, not react fast enough to avoid crashes etc. These developments introduce new complexities in the way the occupants interact with the safety systems in the vehicle in an accident, prompting more in-depth study [27].

In spite of increasing computational power, complex simulations with high fidelity models are still computationally expensive to solve. Performing these simulations for tasks like optimization, parameter studies, quick design studies etc., will be a tedious and time-consuming process. In some cases, it might be limiting the utility of such simulations in a competitive automotive industry [10][11]. To address this issue Reduced order modelling (ROM) has been extensively developed aiming to reduce the computational resources with controlled loss of accuracy. ROM is a subset of meta modelling which is used to build surrogates for parametric study of a large scale system. The objective of the ROM is to find suitable low dimensional variables of a high dimensional complex system. This is also known as dimensionality reduction. Using the low dimensional variables a complex problem can be solved more efficiently and with a reduced computational burden.

After extracting the reduced base function from ROM, a regression based machine learning approach is used to establish a mapping function between input variables to the low dimension space obtained from ROM. The combination of the ROM and regression methods have been explored for the purpose of reducing the solution time of complex simulations. One study [9] presents a way of using autoencoders to produce a low-dimensional representation of time-histories of a structural FE model and the use of Feed Forward Neural Network (FFNN) to establish a mapping from the input variables to the low dimension representation. Other studies [28][29] show the use of Proper orthogonal decomposition (POD) method paired with neural networks to achieve a similar objective for the study of non-linear Poisson equation and viscous flows, nonlinear structural analysis respectively.

The above mentioned studies form the basic idea of utilizing ROM methods such as PCA and Autoencoder to obtain a low dimensional representation or compressed version of the kinematic time histories. Followed by training an ML model to map

the compressed version of the data with the input variables. This way a compressed version of a predicted result is provided by the ML model based on new input variables which are then decompressed to obtain the predicted time histories.

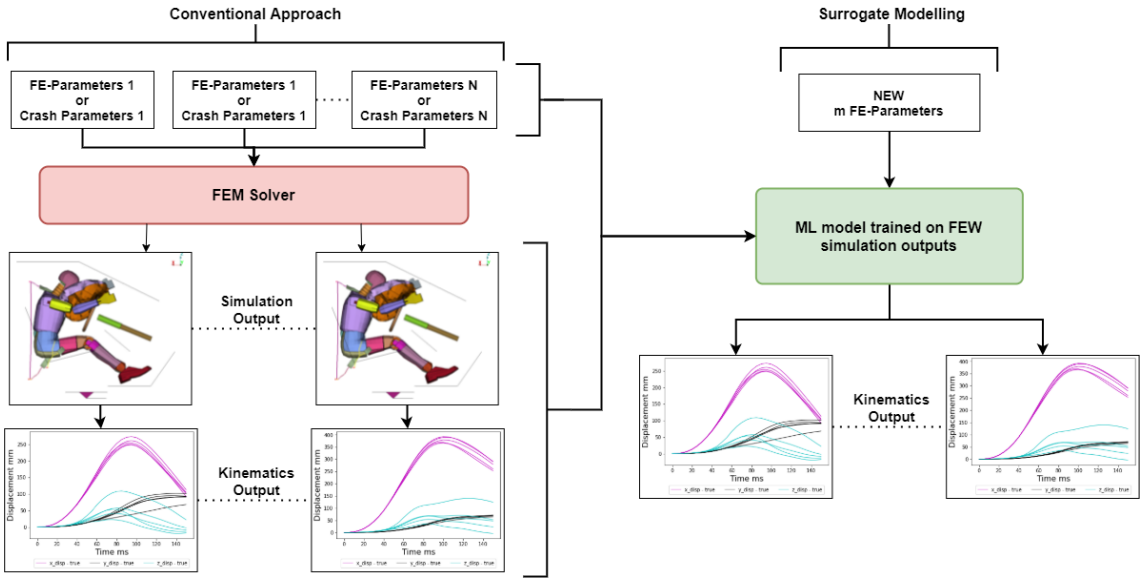


Figure 1.1: Overview of the objective

Figure 1.1 shows the overview of the process involved in the current study. Conventional approaches to various parametric studies, optimization studies etc., involving complex FE models and numerous FEM simulations take a lot of computational resources and time. As shown in the Figure 1.1, the conventional approach involves, individual simulations with varying FE parameters are undertaken with a standard FE solver such as LS-Dyna. The individual simulation outputs are gathered and then studied to arrive at a solution or conclusion on the studied problem. For example, understanding injury mechanics, injury risk due to airbag pressure or vehicle velocity etc. This would require a lot of simulations and in turn a lot of time and resources. By introducing ML models, one needs only a fraction of the computational resources and time to generate an equal number of results as from the conventional approach. This is explained with the surrogate approach as in the same figure. A few FE simulation inputs and output from the conventional approach can be used to train an ML model. This trained ML model can be used to predict the outcome of a number of the simulations without the need of the FEM solver. Any number of simulations outputs can be predicted with the trained model. The time and resources required to obtain enough outputs required for a study using a few FE conventional FE simulations and an ML model are considerably lower than what would be required with the conventional approach alone.

1.2 Objectives

The aim of the thesis is to predict the nodal kinematic responses of a FE human body model using ML models trained on in-crash FE simulation outputs:

- Compare the accuracy of dimensionality reduction methods
- Compare the accuracy of supervised ML algorithms for predicting nodal kinematics from the variables in reduced space obtained with dimensionality reduction.
- Study the effects of the number of samples (FE sims) on the accuracy of the training and prediction

1.3 Limitations

- The thesis work was limited to in-crash kinematics using a Hybrid III M50 fast model instead of a complex FE-Human body model such as SAFER HBM v9 due to computational limitations on hand.
- The project was implemented with the in-crash kinematics obtained with a variation of only 4 major FE simulation parameters, which are vehicle delta velocity, angle of impact, max pull force of the seat-belt pretensioner and airbag leakage scale factor.

2
Theory

In this chapter, brief insights are given on the concepts of FEM, machine learning (ML), methods and concepts of unsupervised dimensionality reduction such as Principal component analysis (PCA), Convolutional Autoencoder (CAE) and supervised machine learning such as Random forest (RF), Feedforward neural networks (FFNN), Gaussian process regression (GPR) model, and basic steps involved in the ML process.

2.1 FEM

The Finite element method (FEM) is a numerical method, has gained increased popularity over the recent years for solving complex real-world Engineering problems. In the FEM, any given domain is divided into a collection of sub-domains (also called finite elements or elements). A collection of these elements is known as an element mesh and these elements are connected to each other at points known as nodes. This is known as discretization. Over each element, a governing equation is approximated to provide the solution and every such equation is assembled for the full domain to provide the solution of the entire problem[40]. Complex real-world problems are solved using the FEM process, one of which is crash simulations and injury biomechanics using FE-HBMs[26].

2.1.1 Finite Element Human Body Models (FE-HBMs)

FE-HBMs are used to study the effects of crashes on the occupant. To understand the injury mechanics and develop safe vehicles, FE-HBMs act as a crucial tool in automotive research and development (R&D). Anthropometric test dummies (ATDs) are a physical model of the human body made of steel and rubber, which are used in destructive crash testing. ATDs use sensors such as accelerometers, force transducers, potentiometers etc to gather measurements for injury prediction. Compared to ATDs, an FE-HBM can be more sensitive in terms of understanding injury outcomes[40].

2.1.1.1 Hybrid III M50 Fast model

The Hybrid III M50 Fast model is a simple representation of the Hybrid III ATD and is a semi-deformable FE model. This model represents a 50^{th} percentile male, 175 cm tall and with a mass of nearly 78 kg. This model is computationally inexpensive compared to a complex FE-HBM and has been used for this thesis work. Figure 2.1

shows the Hybrid III M50 Fast Model used in this project work. The kinematics outputs considered for this thesis is the relative displacements of the Hybrid III M50 Fast model with respect to the vehicle in a crash. The vehicle environment used for these simulations is rigid except for the seat-belt systems and the airbag.

Figure 2.1: Hybrid III M50 Fast model

.

For this thesis, the crash simulations are performed on a parameterized FE model[1] based on variables such as vehicle velocity, angle of impact, seat belt pretension etc. These parameters will be used as input variables for the ML algorithm and will be addressed as FE parameters going further. Table 3.1 shows the parameters used in this thesis. The kinematics output taken from these FE simulations is the nodal displacement, i.e., the x,y,z translational displacement of every node of the Hybrid III model is the kinematic output that is used as the output.

2.2 Machine Learning

Machine learning is a subset of Artificial Intelligence (AI) and a discipline in pursuit of building computer systems that can automatically improve themselves with experience and define the laws that govern such learning process. ML models use algorithms to sift through a large input-output database of problems towards a goal of prediction, classification etc. Its applications have been identified and established in the fields of speech recognition, image classification, computer vision, robotics, medicine etc.[12] Machine learning is further categorised based on how the ML algorithm learns to predict accurately. The basic approaches for these are supervised,

[1]Parameterization is the process by which complex physical processes that can not be resolved directly by a numerical model are represented in a simplified or generalized form which can be represented by variables known as parameters [13]

unsupervised, semi-supervised and reinforcement learning.

In this work, Unsupervised Machine learning like convolutional Autoencoder and PCA were used to encode the kinematic time histories to latent (low dimensional) space and then decoded back to original space. With latent space data as the output the ML model is trained with input FE parameter using supervised Machine learning methods. The final goal is to predict a latent space from new FE parameters and then decode the latent space to the full order kinematic time histories.

2.2.1 Supervised Learning

Supervised learning algorithm consists of input data and its targets or known outputs from which a ML algorithm tries to learn and create a mapping function between inputs and targets. The aim is to use the mapping function to get desired output for new input data. It uses patterns to predict the targets and behaviours to determine the outcome of new input data. Methods like classification, regression etc., can utilize such learning models. It is commonly used in applications where past data can be used to predict outcome of future scenarios. some example includes Feed forward neural networks, gaussian regression model, random forest.[23]
In this section we discuss a few of the regression algorithms used for mapping FE parameters and the latent space.

2.2.1.1 Random Forest

Random Forests Regression (RFR) is a classification and regression technique falling under the category of ensemble learning. Ensemble learning is a method of machine learning that makes use of multiple ML algorithms to obtain better predictions than when any of the constituent ML algorithms is used. Bagging and Boosting are two methods used in ensemble learning [30]. RFR is a Bagging technique which uses multiple Decision tree ML algorithms and each tree in the ensemble uses random subset of the training set to make a prediction, an average of which will act as the final prediction.

Decision trees formed with a random subset of the training data are the constituent ML models forming the ensemble for the random forest. There are different types of decision trees, classification trees and regression trees. Classification trees are ones where the outcome of prediction is discreet to the type of data it belongs or a class of outputs and regression trees are one who's outcome is to define a mapping function to the input variables to a continuous output.

In the training part of the RFR, bagging, also known as Bootstrap aggregation is used to pick the samples for training of individual decision trees which form the forest.[32] It reduces high-variance and that each bootstrap data set can be used on its own before being combined and re-sampled[33].

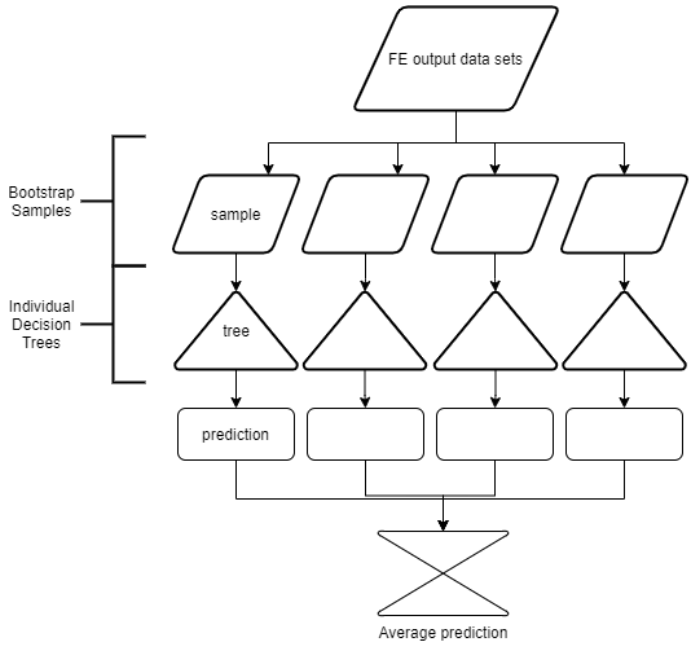


Figure 2.2: Random Forest Regressor workflow

.

2.2.1.2 Feed Forward Neural Networks

The term neural network refers to a network of neurons in the mammalian brain. Inspired by biological phenomena, a neural network is a computational model able to learn from observational data. It consists of a collection of processing units called McCulloch-Pitts neurons[31]: one input layer, one or several hidden layers, one output layer, and the weighted connection between neuron i and neuron j which strengthens the connection between two hidden layers neurons w_{ji}. There are no connections between neurons in the same layer. Information usually travels from inputs to outputs through the hidden layers. Each neuron in the hidden layer is the weighed sum of the previous layer giving a scalar output. Different weights are used to generate the neurons on the subsequent hidden layer. These weights decide the influence of each preceding input neuron or hidden layer on the output. If the output of the weighted sum is greater than the bias for a neuron, that particular neuron will be activated otherwise it remains as zero. Adding a constant bias to the weighted sum helps in retaining the information from a preceding input neuron even when the weighted sum is zero. Thus, helping in increasing the generalization of the neural network. The most widely used activation functions for regression in deep learning are Rectified linear unit (Relu), Exponential linear unit(Elu), and Leaky rectified linear unit(Leaky ReLU). The state of a neuron in a hidden layer is denoted by h_{ji} with bias b_i and weights w_{ji}. similarly the state of neuron in output layer is denoted by O_k with bias B_k and weights W_{jk}. Every neuron in the layer computes as follows[31]

$$h_{ji} = g(\sum_{i=1}^{n} w_{ji}x_i + b_i) \tag{2.1}$$

$$O_k = g(\sum_{k=1}^{n} W_{jk} h_j + B_k) \tag{2.2}$$

During training inputs are applied, weights are updated iteratively by reducing the error between calculated output and the actual output. The mean squared error is used as the loss function to determine the error value. Back-Propagation using chain rule of differential calculus is used to determine the derivative of loss function with respect to weights starting from the output layer and moving back to the input. These derivatives are known as gradients and the neural network is trained by Gradient descent algorithm to arrive at a local minima.

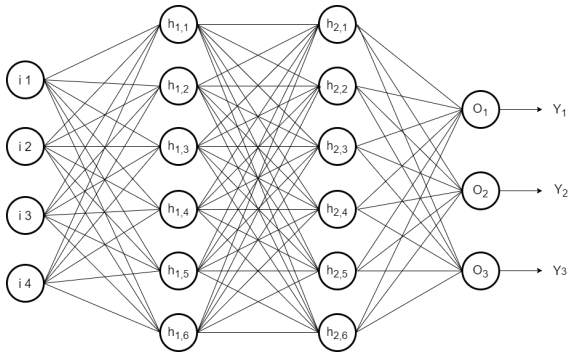


Figure 2.3: A three layer feed forward neural network with four input neurons, two hidden layers each consisting of six neurons, and three output neurons. within each connection, information flows from left to right

.

2.2.1.3 Gaussian Process Regression

A Gaussian process is a collection of random variables with Gaussian distributions defined by mean m(X) and a co-variance kernel K(X,X'), where X is the input domain.

$$f(x) \sim GP(m(X), K(X, X')) \tag{2.3}$$

The objective of the Co-variance matrix is to determine the relationship between any given two data points or the effect of one data point on the other. Thus it is very important to choose a kernel that is able to determine this relationship in the best possible way for the given data set. The basic assumption is that the points with input X which are close are likely to have similar output Y. Gaussian process models the distribution of given data using functions as in equation 2.3 with both input variables and output observations. These functions can then be used to build regression models for predictions.

There are many kernel functions available, some of the kernels we used in this work are Squared exponential(SE), Matern kernel 5/2 and a combination of the two[34]. They contain two hyperparameters one is the standard deviation or variance parameter(σ_f) and the other is length scale(l).

The SE function is exponential and it is infinite time differential which means that the Co-variance function has a mean square derivative of all order. SE function is suitable for modelling smooth functions. SE kernel is expressed as

$$K_{SE} = \sigma_f^2 exp(\frac{-(X - X')^2}{2l^2}) \tag{2.4}$$

where l is the length scale parameter, which describes the reach of influence on neighbours.
The Matern5/2 Kernel is the product of polynomial of order(2) and exponential function, is given by

$$K_{Matern5/2} = (1 + \frac{\sqrt{5}(X - X')}{l} + \frac{5(X - X')^2}{3l^2})exp(\frac{-\sqrt{5}(X - X')}{l}) \tag{2.5}$$

Finding an optimized hyper-parameter for GPR kernels can be done using Monte Carlo tree search, but it is a computationally expensive process. A study on hyperparameter tuning for GPR kernels [36] shows that choice of the kernel has more effect on predictions compared to fine-tuning of the initial distribution of hyperparameters.
In this project the Gaussian process library GPflow is used to implement GPR for the surrogate model used in prediction [37].

2.2.1.4 Hyperparameter Tuning

ML methods are tuned with some unique parameters that control the learning process, these parameters are known as hyperparameters. Hyperparameter tuning is an iterative process of searching for a set of parameters for the learning algorithms to perform at their best for a given data. Doing trial and error for these parameter selections is tedious, hence some of the search algorithms like grid search, random search are used to select these parameters. For these algorithms, discreet values must be specified for each parameter from which the search algorithms pick the most suitable values. In this work, random search algorithm was used for hyperparameter tuning of random forest regressor.

The random search algorithm uses only a limited number of iterations to choose from the combination of available parameter values. This will overlook some combinations as the number of iterations is limited. This method is used to utilize less time in finding probable good combination of hyperparameters.

2.2.2 Unsupervised Learning

Unsupervised machine learning consists of only input data with no targets or labels associated with it. The ML algorithm must figure out what it has been given as data. Its goal would be to find out some patterns, structures within the data. It is widely used in computer vision for image recognition. Some examples which use these techniques are nearest neighbour mapping, k-means clustering, singular value decomposition, Autoencoders etc. [23]

2.2.2.1 Dimensional reduction

High dimensional data refer to data with number of features greater than number of observations. Dimensionality reduction is the process to simplify the understanding of the given data without compromising the information carried within. This can be done either numerically or visually. In this work, number of nodes of the Hybrid III dummy are the features and observations are simulation outputs. Large data sets are becoming increasingly common in wide variety of Engineering and science applications in the digital age. Traditional statistical techniques face challenges to deal with this high dimensional data. Doing regression with high dimensional data is very challenging. However, much of the data is almost similar and can be efficiently brought down to smaller number of variables without loss of much information. Those variables can further be used to visualise the simulation results in a two dimensional or three dimensional way and quickly get some insights about similarities and dissimilarities between simulations, variables, parameters etc[16]. The dimension reduction techniques can be executed in various ways, one of which is by only keeping important variables from original dataset (also called features) and another way is utilizing the redundancy of input data and by finding smaller set of new variables[14]. The most widely used statistical method is principle component analysis, and the machine learning method is Autoencoder which are discussed in the coming sections in detail. Machine learning methods are able to work on high dimensional data with relative ease, but there are some issues that need to be taken care of such as over fitting and computational resource handling. The data which is extracted from the experiments and/or simulations are arranged in a large matrices or array.

For time series data a data matrix is given by 2.6 [8]

$$X_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \tag{2.6}$$

where rows are the number of observations and columns are time steps, also called snapshots (at each time frame). Two of the dimensional reduction techniques widely used in science and engineering are explained below.

2.2.2.2 Principle Component Analysis - PCA

PCA is a technique used to find the dominant features or patterns of large datasets with numerous samples (also known as objects) and variables. It is also characterized as a mathematical model used to reduce the dimensionality without losing much of the variance carried within the low dimension space obtained (i.e., no variance is lost if all principal components are considered). PCA does this by identifying orthogonal directions in the data called principle components which would still largely describe the variance carried by the dataset. Thus a single sample can be expressed in terms of a few 'principle components' instead of a vast array of variables.[17] Nearly any data matrix can be simplified with the help of PCA. It can be used on a well defined

set of samples and variables to build a model of how a system behaves and this in turn can be used for a prediction for new samples. PCA estimates the correlation of the variables by measuring the variance among them. This is used for variable selection.[19]
Assuming the dataset X to be of dimension $m * n$, where m (rows) are the samples and n (columns) are the variables/features. The process of dimension reduction using PCA can be briefly explained in the following steps[20]:

1. Scaling of the dataset - Feature scaling is an important aspect as a precursor for PCA to reduce the variance of features due the the scales of the data in these features. Standardization is a common method of scaling the data which is given as,

$$z = (x - u)/s \tag{2.7}$$

   where, u is the mean of each feature and s is the standard deviation of the feature

2. Construction of the covariance matrix - Identifying the covariance of the features helps in identifying the variance in each feature, with other feature and also identify if the features are correlated. It is of the dimension n*n. For example, covariance of two features x_j and x_k can be given as,

$$\sigma_{jk} = \frac{1}{n}\sum_{i=1}^{n}\left(x_j^{(i)} - \mu_j\right)\left(x_k^{(i)} - \mu_k\right) \tag{2.8}$$

   Where, μ_j and μ_k are the means of features j and k

3. The eigenvectors of the covariance matrix represent the direction of maximum variance and represent the principal components, and the eigenvalues correspond to the magnitude of explainable variance carried by each eigenvector.

4. Explained Variance - To reduce the dimension of our dataset we need to determine which eigenvectors (Principle components) contains the maximum explainable variance. We can do this by calculating the explained variance ratio of the eigenvalues,

$$\frac{\lambda_j}{\sum_{j=1}^{n}\lambda_j} \tag{2.9}$$

   where, λ_j is the eigenvalue of the feature j

   Using the total explainable variance of the principle components, one can choose the number of these that would suffice to define the sample space effectively. If 2 priniple components are sufficient to explain majority of the variance, then W would be a matrix of eigenvectors of dimension $(n * 2)$

5. Feature transformation - To visualize the reduced dimension of the dataset using the choice of principle components, the sample dataset can be projected

onto the PCA subspace (principle components). For examples, consider 2 principle components are enough to explain majority of the variance, the sample data projected onto PCA would be of dimensions $(m * 2)$

$$\boldsymbol{X'} = \boldsymbol{XW} \tag{2.10}$$

Where, X is the dataset of dimension $(m * n)$, W is the matrix containing the eigenvectors containing the maximum explainabe variance $(n * p)$, X' is the transformed dataset in PCA space with dimension $(m * p)$

2.2.2.3 Convolutional Autoencoder

Autoencoder is one of the ML dimensionality reduction techniques using neural networks. Here, high dimensional data can be converted to low dimensional representation called latent space or latent vector by extracting meaningful features from the given data. AE uses neural networks to learn features from the data to minimise the error during reconstruction or decompression i.e., when projecting the low dimension space back to the original dimensions. It is widely used for classification, visualization and regression models. Initialising with random weights to the network, autoencoder model can be trained to minimise the loss function between original and reconstructed data. Stochastic gradient descent method is a common method used to minimise this loss function.

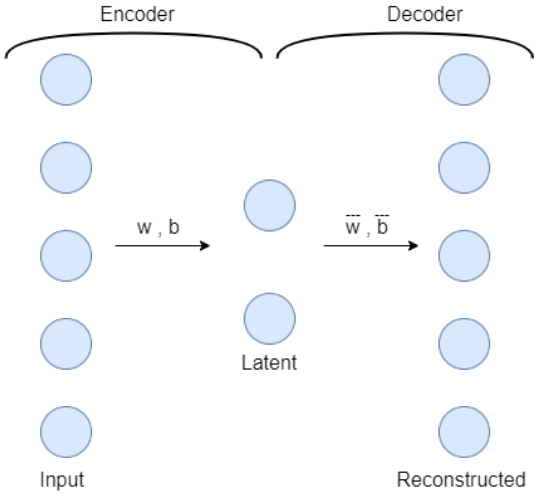


Figure 2.4: A schematic representation of Autoencoder

Fig 2.4 autoencoder with single hidden layers ,$x \in \mathbb{R}^d$ takes as input and send to the hidden layers which can be represented as

$$h = \sigma(Wx + b) \tag{2.11}$$

where W , b and σ are the weighs,bias and activation function of the network.

Although autoencoder is capable of reducing high dimensional data to a lower dimension, it faces challenges when the input matrix is too large. Sudden reduction from a large input layer to a smaller latent space might lead to losing a lot of

information[44]. To solve this issue, a new type of autoencoder emerged known as convolutional autoencoder. Convolutional autoencoder is similar to simple autoencoder in a sense that encodes to latent space and then reconstructed back to original but with the addition of few convolutional and deconvolutional layers, pooling layers in between the input matrix and the flatten layer as shown in Figure 2.5.

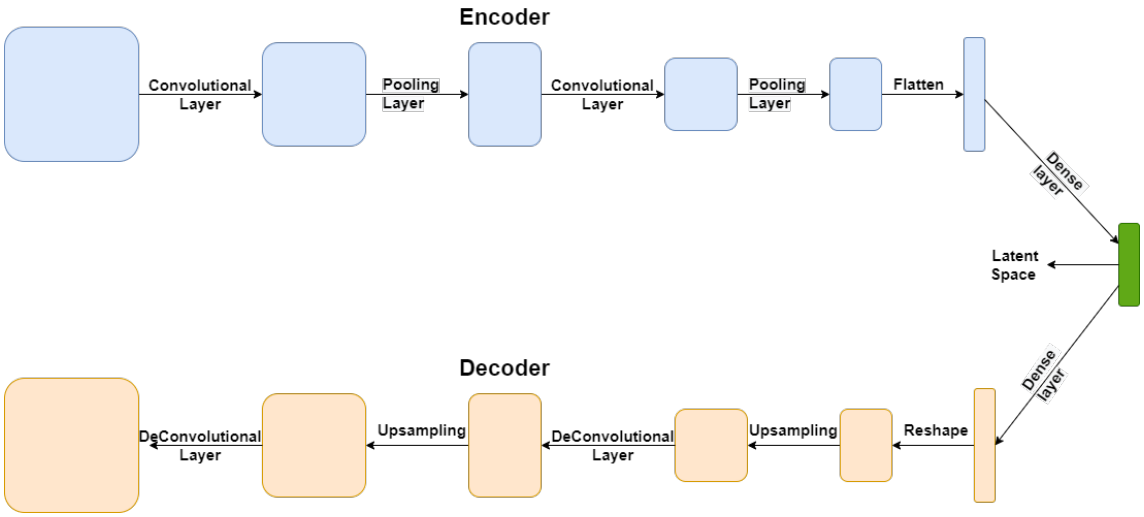


Figure 2.5: A schematic representation of Convolutional Autoencoder

Convolutional Autoencoder-1D takes 2D array of size (m x n) as input and applies a filter of a specified size to get an encoded version of new convolutional layers also called feature maps. This is achieved by applying a filter to the input and then by moving the kernel in the horizontal direction as shown in figure 2.6. The objective of convolution is to extract the most important features from the input matrix and then encoded them to lower dimensional space also called latent space. Pooling is another dimensional reduction layer applied after convolution. The objective of the pooling layer is to take either maximum value or by takes an average of neighbours for each feature maps and represent it in a scalar value. There are no filters applied but strides and padding are applied to the pooling layer.

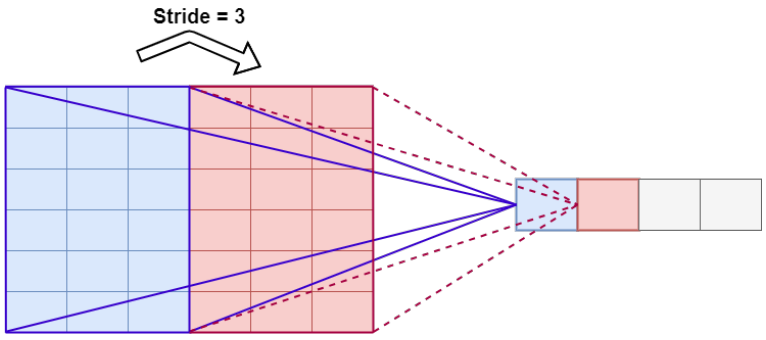


Figure 2.6: A schematic representation of 1D Convolutional Filter with Stride =3

2.3 Model Selection and Evaluation

2.3.1 Train and Test split

To train and evaluate an ML model as discussed in the previous sections, the available data is split into two parts, one for training the ML model and the other to assess the model performance i.e., training data set and testing data set as in Figure 2.7. The test data is not introduced to the ML model during the training process. For this work, the original data is split into 80% of training and 20% of test data.

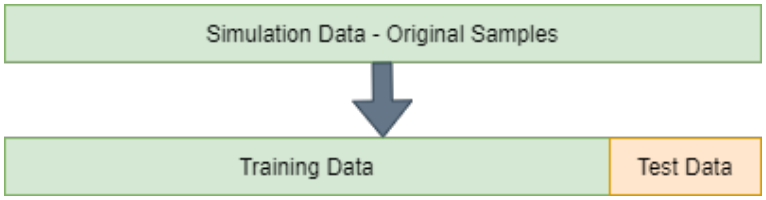


Figure 2.7: Split the simulation data for training and testing

2.3.2 Error Metrics

To evaluate the performance of the ML model, it is necessary to check the accuracy of the predicted values against the true values. The error metrics used in this thesis are;

1. **Mean Squared Error**
   Mean squared error calculates sum of squared difference between true and predicted values.

$$MSE = \frac{1}{n}\sum(Y_{true} - Y_{Predicted})^2 \tag{2.12}$$

2. **Mean Absolute Error**
   Mean Absolute error calculates the mean of absolute difference between true and predicted values.

$$MAE = \frac{1}{n}\sum|Y_{true} - Y_{predicted}| \tag{2.13}$$

3. **Euclidean Distance**
   Euclidean Distance is the distance between two points in space. This metric is chosen to evaluate the performance of the ML model as the input data and the final predictions are the x, y, z translational displacements in 3D space.

$$d = \sqrt{(X_{true} - X_{Pred})^2 + (Y_{true} - Y_{pred})^2 + (Z_{true} - Z_{pred})^2} \tag{2.14}$$

3
Methods

In this chapter, the workflow for building the surrogates is explained. This comprises of data extraction, data preparation or arrangement, dimensionality reduction, regression and model validation. An overview of the methods is given as in Figure 3.1. This was executed for the displacement data of all nodes of the hybrid III dummy as explained in section 2.1.1. The surrogates built using the process are then evaluated for the objectives of comparing the performance of dimensionality reduction methods and the performance of the surrogates based on the number of simulations used in the training process. Three Supervised learning ML methods Feed forward neural networks, Gaussian process regression and Random forest regressor were used to map the FE parameters with the low dimensional space.

In the first part of the thesis, the kinematics data of only 1000 nodes of the hybrid III dummy were used for tuning and testing of different data arrangements, dimensionality reduction methods and surrogates. Once the effectiveness of the data arrangement and the paired surrogate is tested, they were extended for the full dummy comprising of 7441 nodes and evaluated. They are evaluated for sample sizes of 25, 50, 75, 100 FE simulation results, each sample size split 80% for training and 20% for testing purposes as discussed in section 2.3.1.

Both MAE and Mean Euclidean Distance of the mean errors of varying sample sizes can be used to ascertain the performance of PCA and CAE-1D along with the ML models. A few selected nodes are used to determine the empirical closeness of the true and predicted displacements. For a model based on 1000 nodes, a node inside the head and a node on the chest and for the full model with 7441 nodes, a node on the left shoulder, hand and left foot are used to determine this empirical closeness.

3.1 Data extraction from simulations

The data used in this thesis was generated from FE simulations of the Hybrid III dummy of the 50^{th} percentile male in a seated position on the driver side using LS-Dyna (R11.1) solver. Various injury influencing factors in a crash are chosen as varying input variables for the FE simulations, further referred to as FE parameters. These varying FE parameters were chosen based on empirical and engineering judgement and their ranges as in Table 3.1 were generated with uniform distribution and Latin hypercube using 'dynakit' python library [35]. This extracted data is used commonly for both PCA and Autoencoder based surrogates.

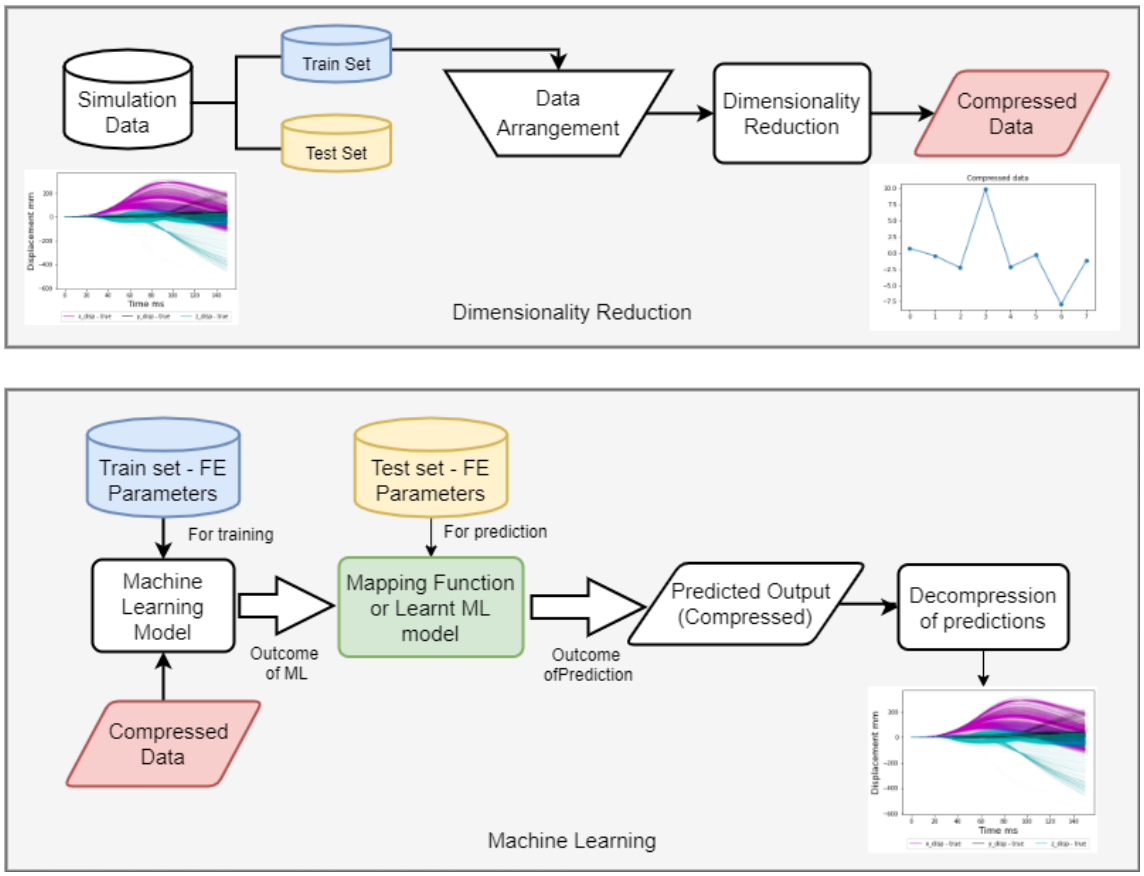


Figure 3.1: Flow of Dimensionality Reduction and Machine Learning

Displacement data from the D3plot result files generated by LS-Dyna was extracted using python library - lasso.dyna by LASSO GmBH. This data is in the global frame of reference. As we are mainly interested in the kinematics of the dummy with respect to the vehicle frame of reference, the data extracted from the D3plots is subtracted with the displacement of a fixed node on the vehicle to obtain the needed relative displacement of the dummy. The extracted data is of a 3D shape $m_{states}, k_{nodes}, xyz_{disp}$ for each simulation. For multiple simulations needed for training, the raw displacement data is arranged in a 4D shape $[n_{sims}, m_{states}, k_{nodes}, xyz_{disp}]$ which is addressed as high dimensional data in this report.

Table 3.1: Parameter ranges generated FE simulations

Parameters	Unit	Minimum	Maximum
Delta Velocity	[km/h]	40	65
Angle of Impact	[degree]	-10	10
Max pull-force of Pretensioner	[kN]	0.99	2.5
Airbag Leakage Scale Factor	[-]	0.5	1.5

Train test split of the data as mentioned in figure 2.7 is used to split the simulation data into a training set and testing set. A constant seed value is used to generate a consistent train test split throughout the study. The training set of data is then further processed as per the necessity of the dimensionality reduction model.

3.2 Surrogate Model 1 - Principal Component Analysis and Regression

In this section, the process of building a surrogate involving PCA as the Reduced Basis (RB) for the ML algorithms is explained.

3.2.1 Data arrangement for PCA

To generate the RB of the original data using PCA some data processing is necessary. The Y data or the observations needed for training the surrogate ML model needs to be of the shape $(n_{samples}, g_{features})$. Hence the high dimensional data needs to be rearranged to this shape. A dimensionality reduction method is used for this purpose. In this section, we discuss the use of PCA to arrive at this goal. [16] suggests some ways of input matrix arrangement for PCA using strain data of a front crash management system and longitudinal member in the crash, one of which is to arrange all the strain data from a FE simulation in a single row and an other is to arrange all the time history data of a state from a simulations in one row and consecutive rows contain the data of different states. A modification of these methods is considered for this study to suit the purpose.

As discussed in subsection 2.2.2.2 PCA is a technique used to determine the dominant characteristics of a given set of observations and ones which can also be used to create a low dimensional space from high dimensional data. The Hybrid III dummy has 7441 nodes and the D3plot has displacement output at 32 states and 3 displacement DoFs (x,y,z displacements).

The method of data representation followed in this study is that the x,y,z displacement of all nodes at individual state are arranged in individual rows. All states of each simulation are appended in new rows resulting in a matrix/array of shape $(n_{sims} \times m_{states}, k_{nodes} \times xyz_{disp})$, for 10 simulations its size is $(320, 22323)$ representing $(n_{samples}, g_{features})$ of the raw data. This method will be further addressed as One State in One Line arrangement (OSioL). This is shown in Figure 3.2.

3.2.2 PCA and Data for Regression

In this section, the process of PCA is discussed for the data representation approach OSioL. PCA is transforming the data in a way that fewer parameters or features describe the variance in the data. Once the raw displacement data is in the format of $(n_{samples}, g_{features})$ and follows an approach as explained in the section 2.2.2.2,

SKlearn PCA library [39] is used to execute PCA and obtain a reduced space of dimension $(n_{samples}, p_{principlecomponents})$. The first few components contain the most amount of variance information of the raw data and is ascertained by the use of explained variance as discussed in section 2.2.2.2 and in Figure A.1. Hence choosing the number of PCs $V = [V_1, V_2, ...V_L]$ is important as the components are sorted based on the amount of information they carry and their significance. The projections of raw data onto these components gives the reduced basis or low dimensional space X'.

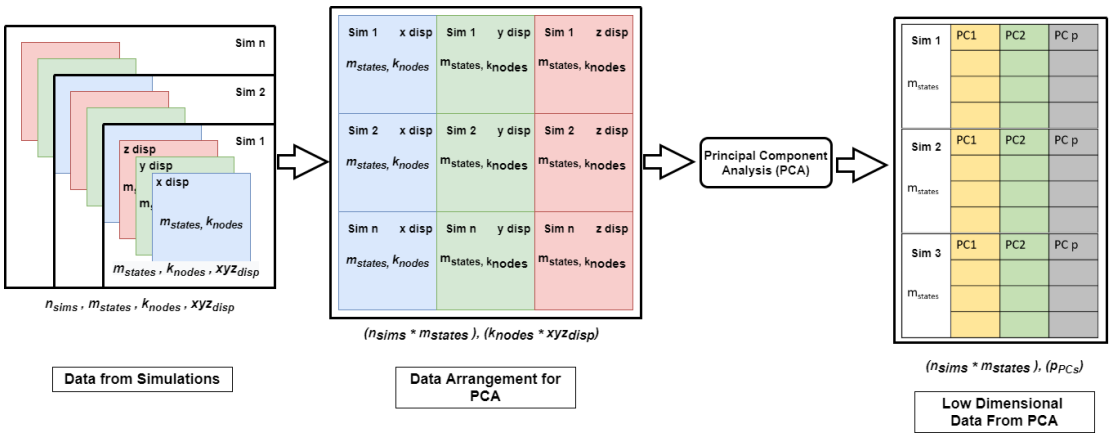


Figure 3.2: Flow of Dimensionality Reduction using PCA

Once the process of PCA is complete and the number of components chosen, the shape of the output from PCA will be $(n_{sims} \times m_{states}, p_{principlecomponents})$. This needs to be arranged in the shape of $(n_{sims}, g_{features})$ to match with the number of FE simulation outputs considered for further process $(n_{sims}, f_{FEparameters})$. To do this, the component values at each time state for one simulation are all arranged in a single row as shown in Figure 3.3

	m_{states}			m_{states}			m_{states}		
Sim 1		PC 1			PC 2			PC 3	
Sim 2									
Sim 3									

Figure 3.3: Arrangement of PCs for ML training

To obtain the low dimensional space from PCA, the number of principal components are determined to capture 99.999% of variance in the training sample data to maintain. This value of variance in terms of percentage was chosen instead of a scalar value as the number of principal components needed to capture a certain amount of variance changes with the sample size. After obtaining the low dimension space from PCA consisting of only a few principal components, it is projected onto the full dimension displacement space. This is then compared with the true displacements to determine the loss due to dimensionality reduction. A similar process is undertaken after predicting the low dimension space from the ML model to obtain the predicted displacements in the original high dimension space.

3.2.3 Regression after PCA

For the data arrangement OSioL the low dimensional subspace obtained from PCA X' are the observations that the surrogate is to train on. X' is of shape $(n_{sims} \times m_{states}, p_{PrincipalComponents})$ and the input FE parameters are of shape $(n_{sims}, j_{fe-parameters})$. For supervised learning, according to section 2.2.1 the dimensions of the input parameters and the output observations must be the same. There is a mismatch in the dimensions of the outcome of PCA and the input FE parameters. To mitigate this, the Principal component values of every state of each simulation is concatenated in rows bringing the shape of low dimension space to $(n_{sims}, m_{states} \times p_{principlecomponents})$. This transformed low dimension space will be used for training and a new low dimensional space of this transformed will be the outcome of the ML prediction for new FE parameters.

For the different ML models used along with PCA, some of the hyperparameters were tuned manually and some were chosen empirically and some kept constant. FFNN was tested with one, two and three hidden layers along with a combination of a different number of neurons in each layer. The number of neurons tested was 64, 128, 256, 512. The best FFNN architecture and neurons combination for each sample size varied and thus individual sample sizes handling a different number of PCs needed a different FFNN architecture. Each sample size was trained for 20000 epochs with a learning rate of 0.0001, and a batch size of 5. Relu activation function was used for all hidden layer neurons during training. Similarly, GPR was initialised with different kernel lengthscale, kernel variance for varied sample sizes of displacement data. GPR is trained over 100 iterations to get an optimised length scale and variance. Gpyflow library was used to train and tune the model. For random forest regression, a random search algorithm as mentioned in section 2.2.1.4 with 100 iterations was used to obtain the best hyperparameters within the number of iterations performed.

Algorithm 1: Regression Model

Input Prepare the training set $D = (inputs(\mu_i), targets(X'_i))$ where i is the number of simulations.

Output Regression Function $f(\mu)$ from D

1. Split the Dataset for train and test in 80% : 20% split
2. 80% split data undergoes PCA
3. Kfold training of train split
4. Recover output $f(\mu^*)$ for a new parameter value μ^*
5. Projecting the reduced order solution to full space
6. Evaluate the recovered full order solution against test samples using error metrics

The training and testing accuracy are evaluated using MAE. Euclidean distance is also used as an additional metric to evaluate the error from the final outcome. Mean Square Error (MSE) is avoided as even one off prediction can cause a huge increase in error value compared to MAE or Euclidean distance. And when a huge model

such as the Hybrid III dummy with 7441 nodes and 32 states is considered, a few off predictions can penalize the error to a great extent. Algorithm 1 gives a brief overview of the regression process.

3.3 Surrogate Model II - Autoencoder and Regression

In this section surrogates involving Autoencoder(AE) as the Reduced Basis for the ML algorithms are discussed.

3.3.1 Data arrangement for Autoencoder(AE) and Regression

Data extraction from simulations is as explained in section 3.1 and the data used is the same. Displacement matrix received from the simulations is 4 Dimensional and in the form of $(n_{sims}, m_{states}, k_{nodes}, xyz_{disp})$. As Autoencoder takes a 3D array as input, 4D array have been converted to 3D array $(n_{sim}, k_{nodes}Xxyz_{disp}, m_{states})$ i.e., xyz displacements of each simulation are appended in row, their corresponding time states are in columns. This arrangement of data and flow of dimensionality reduction in AE is shown in Figure 3.4. These datasets were normalised to standard range by dividing them with the maximum value of the corresponding simulation dataset. All such simulation data are stacked in the third dimension. In the first phase, 1000 nodes of the hybrid III dummy were used to test different convolution 1D architectures and to optimize the hyperparameters. Once the optimized values and architecture was chosen, it was extended for the full dummy comprising of 7441 nodes and evaluated.

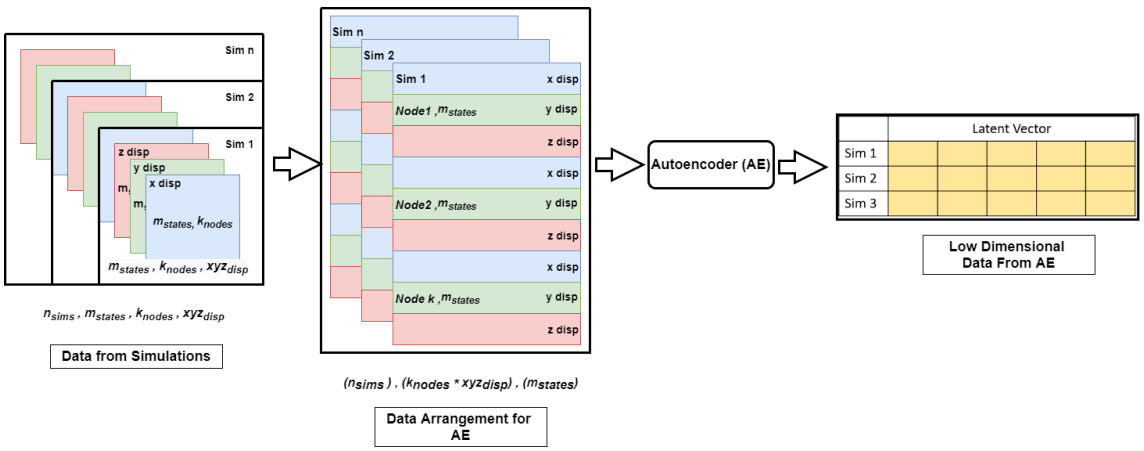


Figure 3.4: Flow of Dimensionality Reduction using AE

3.3.2 Architecture for AE

The architecture for AE is explained in the section 2.2.2.3. The number of feature maps, kernel size and stride are chosen empirically and based on literature [38] [9]. Five architectures were evaluated as shown in table 3.2 and the effect of each architecture is explained in the results section 4.16. All the architectures in table 3.2 are trained with the displacement data sets of simulation sizes 25, 50, 75, 100, and for 1500 epochs with a constant learning rate of 0.0001 and a batch size of 10. An Adam optimizer is used for the minimization of loss, with the loss function being MSE. From these five architectures, the best was selected based on the error of decoded to the true values from FE simulations. MAE and mean Euclidean distance error were calculated for 20% of test samples to help make this choice. The effect of latent space size on the accuracy of decompression is evaluated with four different vector dimensions of 8, 16, 32, 64. The selected Latent space size remains constant while training for all simulation sizes. Further the same process is extended during regression to understand if this same trend repeats during the prediction of test samples. The selected architecture and latent space dimension are further used for the full Hybrid III model with 7441 nodes.

3.3.3 Regression after AE

The AE is trained over the high dimensional displacement matrix to produce a low dimension space known as latent vector. This latent vector along with the FE parameters is used for training various regression models to create a mapping function used in prediction. Algorithm 2 shows an overview of the AE-ML workflow. FE parameters of individual simulations are mapped to the latent vector obtained from the displacement data from the same simulation.

For the different ML models used along with AE, some of the hyperparameters are tuned manually and some are chosen empirically and kept constant. FFNN was tested with one, two and three hidden layers along with a combination of a different number of neurons in each layer. The number of neurons tested was 64, 128, 256. The best FFNN architecture chosen was 2 hidden layers with 64 neurons in each hidden layer. Each sample size was trained for 10000 epochs with a learning rate of 0.0001, and a batch size of 10. Relu activation function was used for both hidden layers during training. These constant hyperparameters were based on the study [38] on structural analysis of a plate using similar timeseries data. Similarly, GPR was initialised with a kernel lengthscale of 1, kernel variance of 0.5 and likelihood variance of 1. GPR is trained over 100 iterations to get a suitable length scale and variance. For random forest regression, a random search with 100 iterations was used to obtain the best hyperparameters within the number of iterations performed.

When test FEM parameters are given to any of the regression models, then the outcome would be a low dimensional space which is then projected to the high dimensional space i.e., displacement data, in the case of FFNN a convolutional decoder is used for this purpose. These predicted displacements are evaluated against the true FEM displacement results using MAE and mean Euclidean distance error

Layers	Arch 1	Arch 2	Arch 3	Arch 4	Arch 5
Input matrix	3000 x 32				
Conv Layer 1	500 x 30	1000 x 30	1000 x 32	1500 x 30	1500 x 32
Average Pooling	–	–	1000 x 16	–	1500 x 16
Conv Layer 2	250 x 28	500 x 28	500 x 16	750 x 28	750 x 16
Average Pooling	250 x 14	500 x 14	500 x 8	750 x 14	750 x 8
Conv Layer 3	125 x 12	250 x 12	250 x 8	375 x 12	375 x 8
Average Pooling	125 x 6	250 x 6	250 x 4	375 x 6	375 x 4
Conv Layer 4	—	125 x 4	125 x 4	150 x 4	150 x 4
Padding	0	0	2	0	2
Latent	8	8	8	8	8
Kernal	3	3	3	3	3
Stride	1	1	1	1	1

Table 3.2: Various Architecture for AE - 1000 Nodes

metrics.

The training of the AE and the FFNN was performed using the GPU version of the Tensorflow framework on google colab utilizing k80 and Tesla T4 GPUs. Computational costs taken for generating simulation and training neural networks is shown in Figure 5.1.

Algorithm 2: AE with Regression Model

Input Arrange the Displacement matrix $D = N_{sim} * d * t$ where d is the degrees of freedom, t is the time states.

Output Regression Function $f(\mu)$

1. Split the AE Dataset for train and test in 80% : 20% split
2. Train AE over displacement matrix. Obtain the low dimensional vector from encoder
3. Train the regression model to establish mapping between low dimensional vector and FEM parameters
4. Recover output $f(\mu^*)$ for a new parameter value μ^*
5. Project the low dimensional vector from step 4 to Convolutional decoder
6. Evaluate the ML predicted against true FEM samples using error metrics

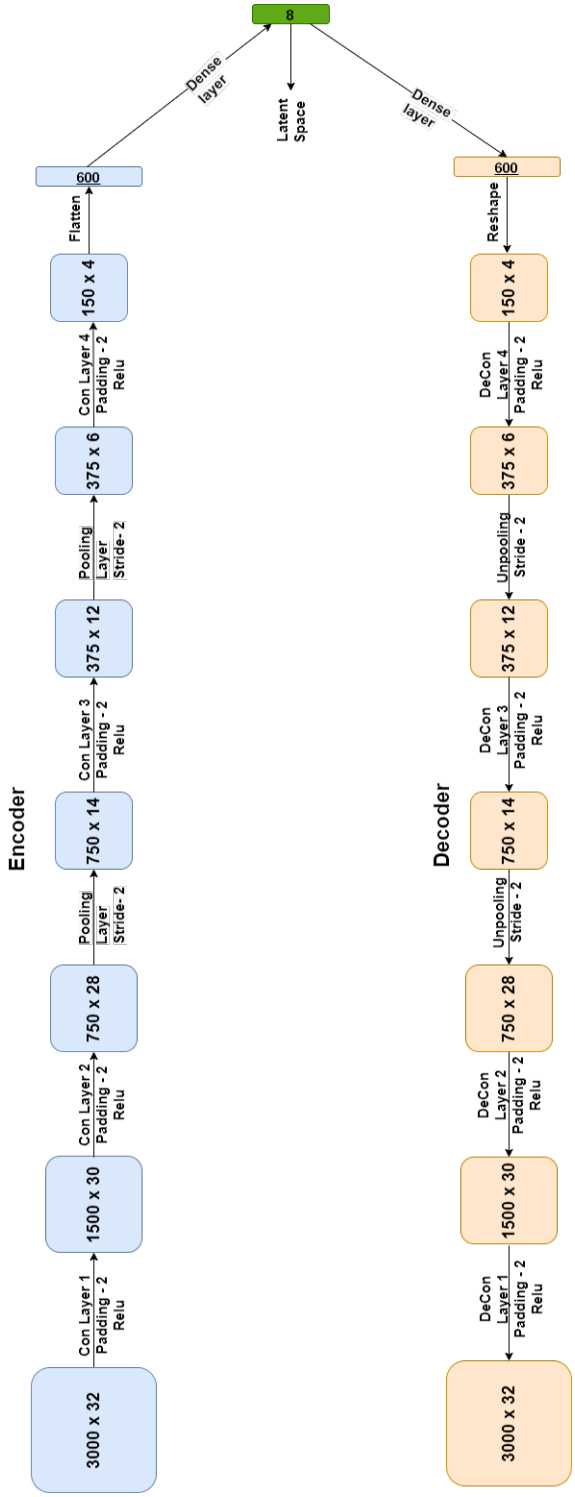


Figure 3.5: AE Architecture 4 - 1000 nodes

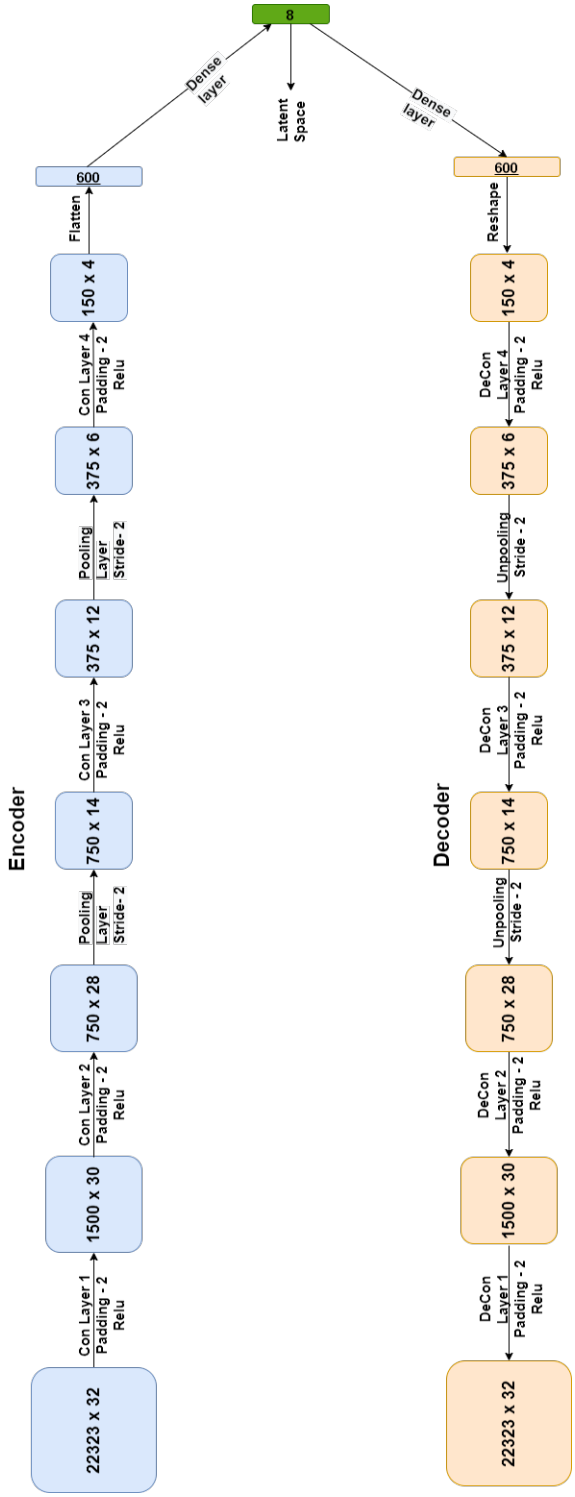


Figure 3.6: AE Architecture 4 - 7441 nodes

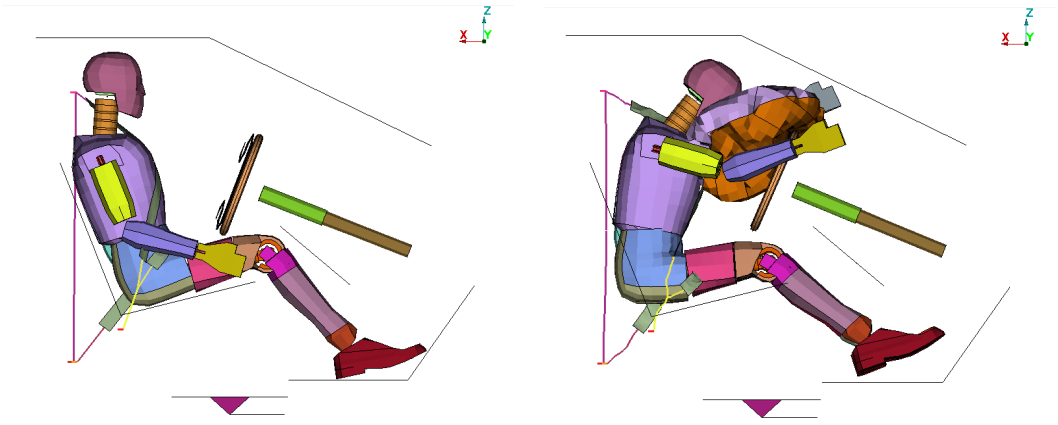
4

Results

In this section, the results corresponding to the above discussed methods i.e., data from FE simulations, the performance of dimensionality reduction methods PCA and AE, the results from the surrogate ML models built on the dimensionality reduction methods are illustrated. For the first part, the results for the full Hybrid III model with 7441 nodes are shown followed by the comparisons with the results for the 1000 node study. The models are evaluated for a simulation size of 25, 50, 75, 100 using MAE and mean Euclidean distance as the metric of evaluation between the true and decompressed or predicted values. The results for the full model are discussed early in the chapter to emphasise the performance of the suggested surrogate models for the full Hybrid III dummy, which is the objective of the thesis. The results for 1000 nodes are shown after the results for the full model, which form the basis on which the surrogates for the full Hybrid III dummy are formed as discussed in section 3.

4.1 FE simulation results

Figure 4.1 shows the model at the beginning of the crash in a seated position and the final position at the end of the 150 ms crash with the airbag deployed. The nodal x y z translational relative displacements of the dummy for a single simulation are as in Figure 4.2.



(a) Dummy at start of crash (b) Dummy at end of crash

Figure 4.1: The position of the dummy at beginning and end of crash

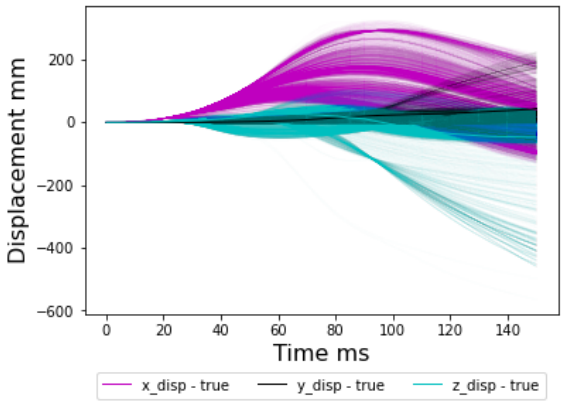


Figure 4.2: All 7441 x y z displacements of a single simulation sample overlaid

4.2 Dimensionality Reduction - Compression and decompression of data

4.2.1 Principal Component Analysis

For PCA, to understand the effects of considering the number of principal components to consider for the low dimension space, Figure 4.3 shows the cumulative variance captured by the PCs obtained from PCA. It can be seen that a few of the PCs can describe the majority of the variance in the simulation data. The blue dashed plot shows a zoomed-in version of the black plot showing the cumulative variance of all the components.

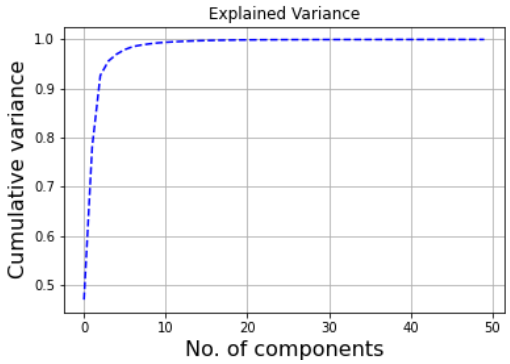


Figure 4.3: Cumulative variance of the first 50 components out of over 1200 PCs.

Figure 4.4 shows the MAE of the true simulation values displacements and the decompressed values from a low dimension space made of varied number of principal components. The error can be considered low for a small number of 10 components but drops significantly when increased, which is expected. The plot ends with an error for 98 PCS which are required to capture 99.999% of the cumulative variance.

This percentage value is carried throughout the process of implementing PCA to obtain the compressed version of the sample data. The effect of the number of PCs on the actual displacement data can be seen in Figure 4.6. Using 10 PCs gives an overall capture of the shape of the displacement curves but does not capture the 0 displacement at 0 ms time state, increasing to 50 PCs helps capture the features of the curves better and also captures the start at 0 ms. Using 98 components captures most of the features of the sample displacement curves with an error of 0.05 mm MAE between the projected displacement data from 98 components and the true displacement data.

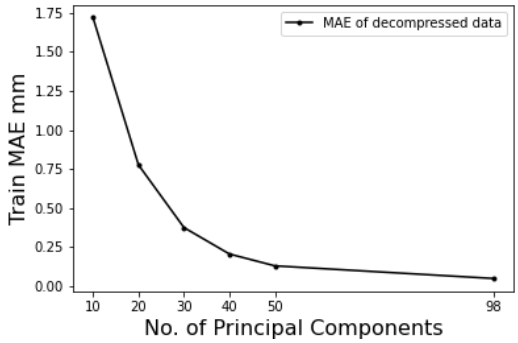


Figure 4.4: MAE of decompressed simulation data based on No. of PCs - sample size of 50 simulations

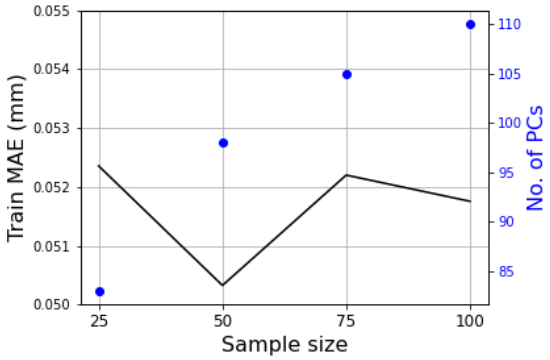
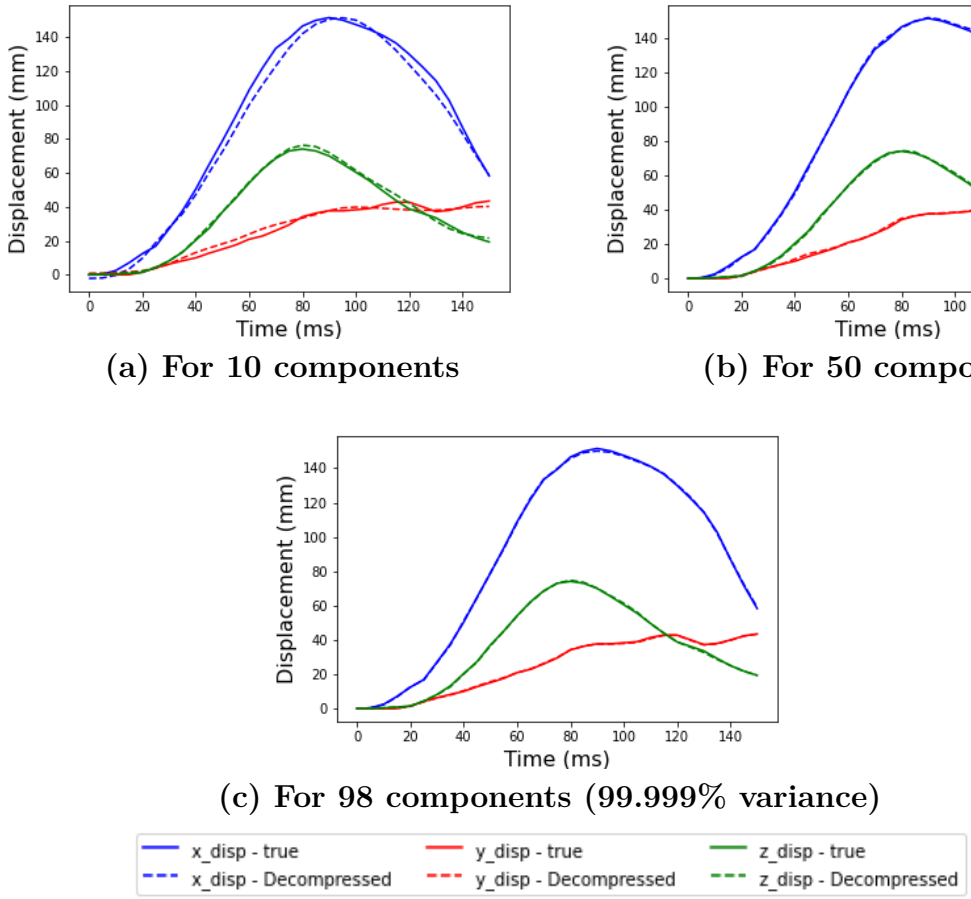


Figure 4.5: MAE of decompressed displacements from PCA on training samples for No. of PCs required to capture 99.999% variance

For varied sample sizes, the required number of principal components needed to capture adequate variance varies. It also varies on the complexity or variation in the data. The plot in Figure 4.5 shows the MAE of decompressed samples from PCA to the true values of displacements and the number of PCs required to capture 99.999% variance for different sample sizes. It can be noted that for varied sample sizes, to capture a certain amount of variance, the number of PCs required is different. Also, though the percentage variance capture is constant the error increases, only by a small margin, this is due to the increase in sample size and the smaller features

of data not being accounted for in the first few PCs that make up the majority of variance in data.



(a) For 10 components

(b) For 50 components

(c) For 98 components (99.999% variance)

Figure 4.6: One trained sample true and decompressed values from PCs for a node on the chest. From a sample size of 50 simulations

4.2.2 Autoencoder

By comparing the effects of different architectures as in Table 3.2 for 1000 nodes, architecture 4 was chosen to use for ML model of full Hybrid III model. The results for these are discussed in section 4.3.1.

Upon the choice of architecture 4, the latent space size is varied to understand its effects on overall error due to dimensionality reduction of the training sample set. Figure 4.7 can be used to investigate these effects using MAE. It is observed that the selection of higher latent dimension reduces the amount of information lost in the decoder but with a very small magnitude. The decrease in error can be attributed to the reduction in loss of data from the flatten layer to the latent vector during the encoding step, as discussed in section 2.2.2.3.

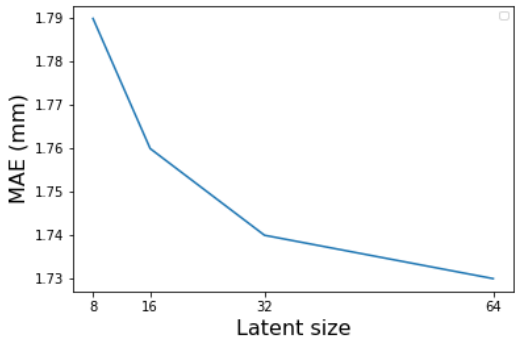
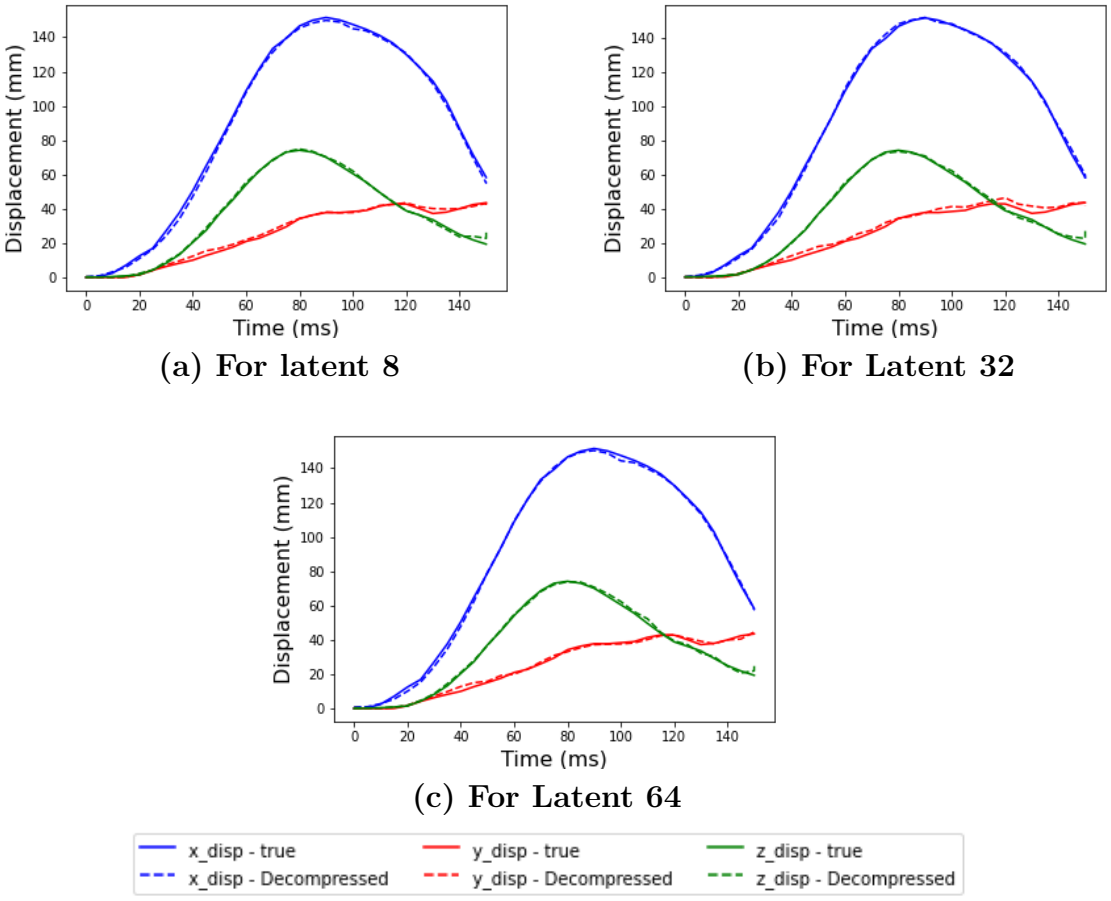


Figure 4.7: Performance of Latent space size

The Figure 4.8 illustrates the true curves with AE decoded curves for varying latent space dimension. It can be seen that by increasing latent size, decoded displacement moves somewhat closer to the true values.



(a) For latent 8

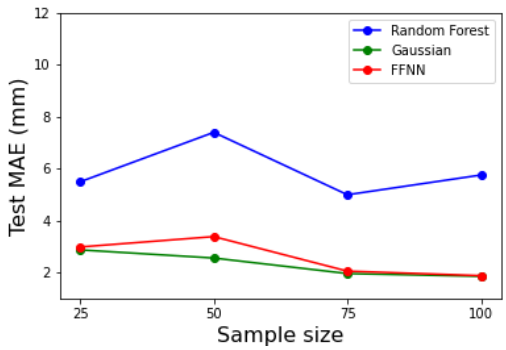
(b) For Latent 32

(c) For Latent 64

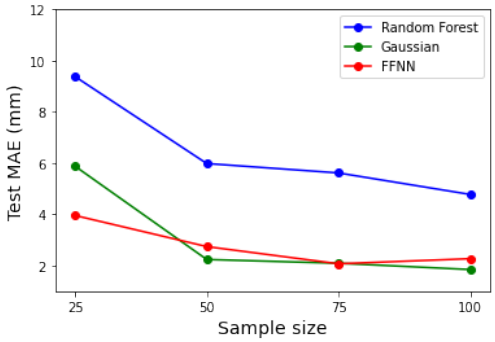
Figure 4.8: One trained sample true and decompressed values from AE for a node on the chest from a sample size of 50 simulations

4.3 Machine Learning

From the results for 7441 nodes, Gaussian process regression and FFNN perform better compared to the tree based Random Forest across all sample sizes as seen in Figures 4.9 and 4.10 using MAE and mean Euclidean distance as the error metrics respectively. This trend remains the same for both PCA and autoencoder based surrogates. The mean test error of predicted displacements of the test set and the true displacement values sees a steady decline from a sample size beyond 50 for FFNN and GPR.

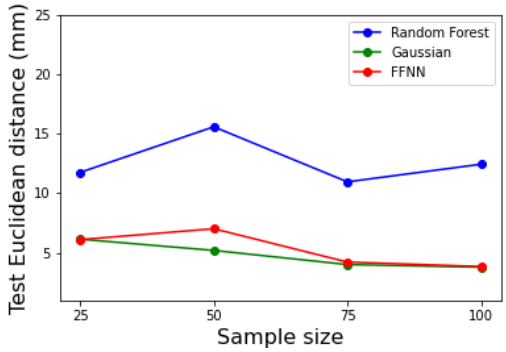


(a) PCA-ML performance

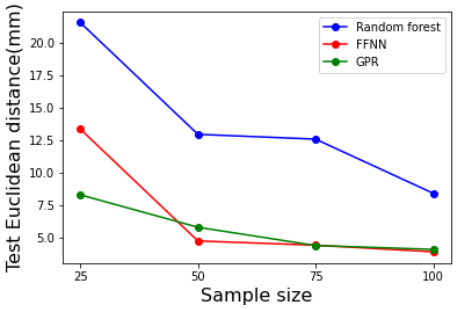


(b) AE-ML performance

Figure 4.9: Test samples MAE from PCA and AE based surrogate model with respect to sample size, For full Hybrid III model



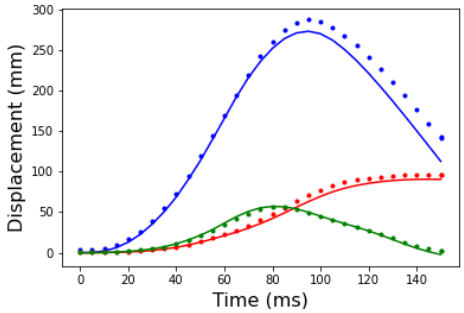
(a) PCA-ML performance



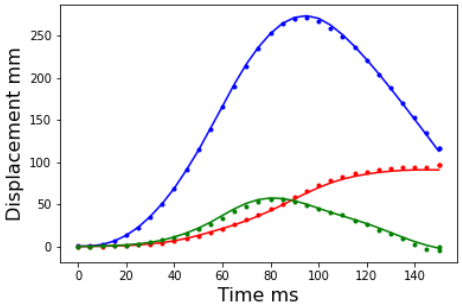
(b) AE-ML performance

Figure 4.10: Test samples mean Euclidean distance from PCA and AE based surrogate models with respect to sample size, For full Hybrid III

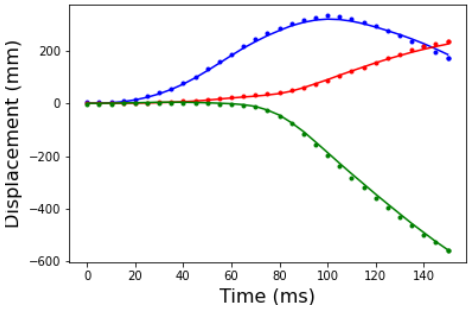
To understand the accuracy of prediction at the nodal level, Figure 4.11 shows the comparison of true and predicted displacement of the choice nodes for one test sample for both PCA and Autoencoder ML based FFNN. This distribution of nodal displacements shows different scales and shapes of the movement of the hybrid III dummy during a crash.



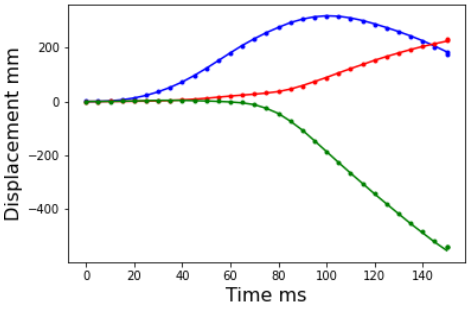
(a) Head Node 1000001 - PCA



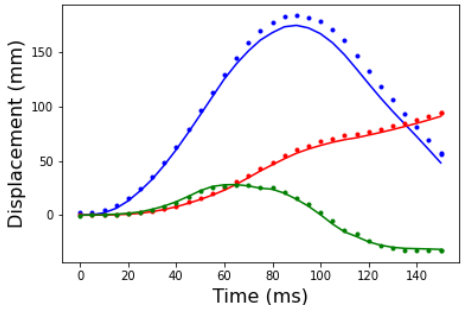
(b) Head Node 1000001 - AE



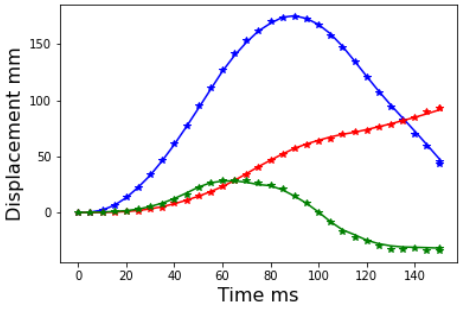
(a) Left fingertip 1002266 - PCA



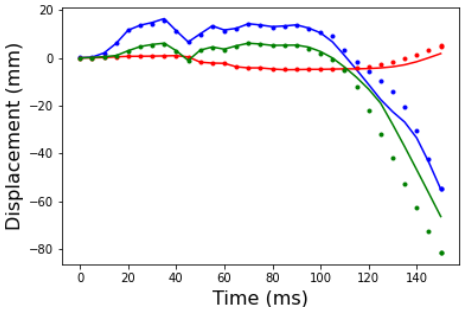
(b) Left fingertip 1002266 - AE



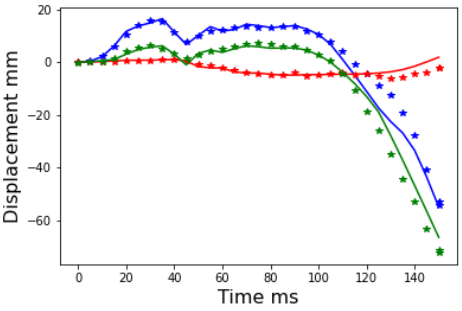
(a) Left shoulder 1002467 - PCA



(b) Left shoulder 1002467 - AE



(a) Left toe 1009569 - PCA



(b) Left toe 1009569 - AE

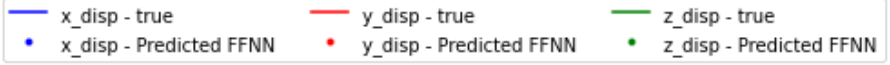
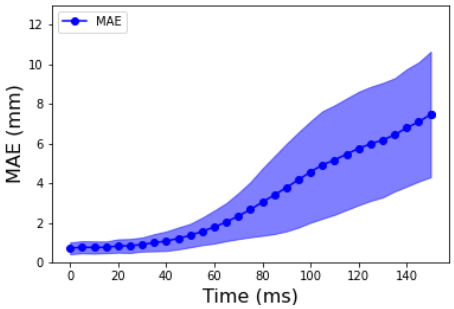


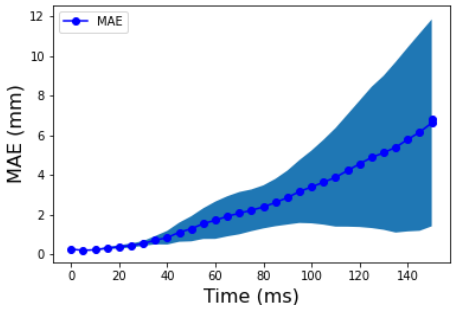
Figure 4.11: One test sample ML prediction versus FEM results of PCA and AE based FFNN for sample size of 50

The predicted displacements obtained by the proposed AE-FFNN are in closer agreement with those computed by the FEM model compared to the ones from the PCA-FFNN model. The closeness of predicted displacements moves farther from the true values towards the end of the plot for PCA-FFNN which is true with other ML models paired with PCA as seen in the Appendix A.

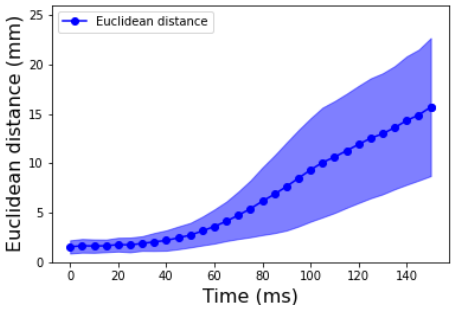
This trend can be observed better in Figure 4.12 which shows the mean and variance of error at every timestep for all nodes of all the test samples. The error is comparatively lower up to 60 ms, beyond which there is an increase in both cases of PCA-FFNN and AE-FFNN. For PCA-FFNN the mean and variance of error at 0 ms is considerably higher when compared to AE-FFNN. Similar results for GPR and RFR are in the Appendix A.



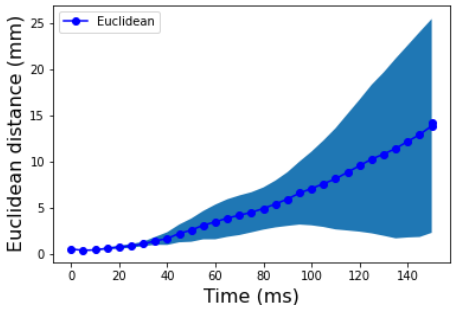
(a) MAE - PCA



(b) MAE - AE



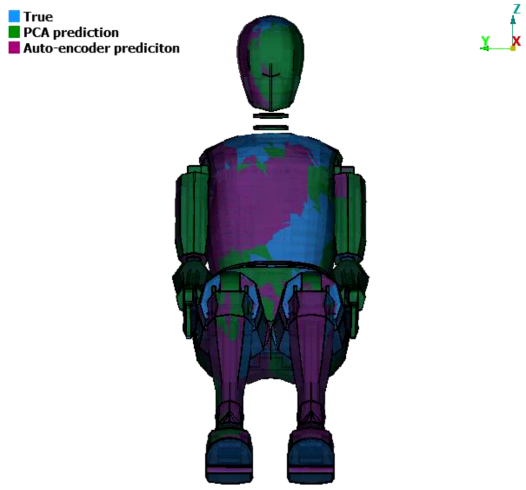
(a) Euclidean distance - PCA



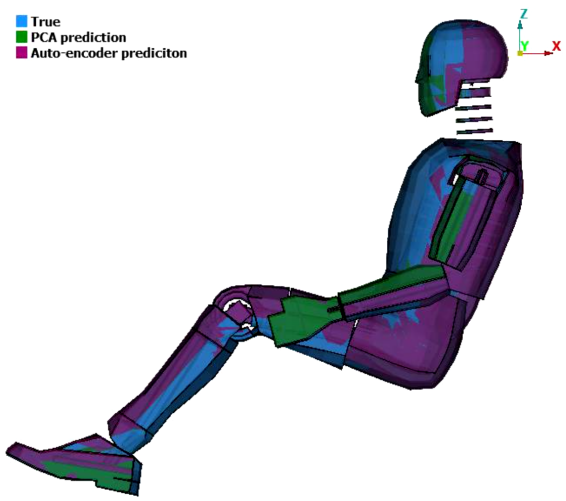
(b) Euclidean distance - AE

Figure 4.12: Mean test samples ML prediction error at every time state for all nodes. For a sample size of 50 - FFNN

The actual positions of the dummy based on the PCA-FFNN and AE-FFNN predictions relative to the true position from FE simulations can be seen in Figure 4.13. These results are from one of the test samples from a total sample size of 50 simulations. It can be noted that at 0 ms both the predicted models are very close to the true position of the dummy. At the end of the crash i.e., at 150 ms, PCA-FFNN outputs are with more error compared to the AE-FFNN prediction. This can be predominantly seen at the extremities of the dummy, i.e., at the hands, head and feet in the side view.

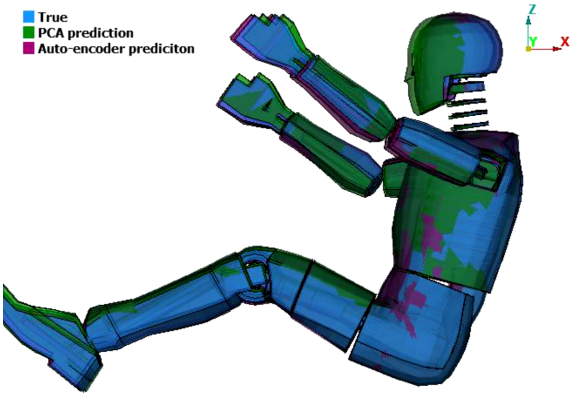


(a) Front view - 0 ms



(b) Side view - 0 ms

(a) Front view - 150 ms



(b) Side view - 150 ms

Figure 4.13: Position of dummy from FE simulation along with PCA-FFNN and AE-FFNN predictions. For a sample size of 50

Figure 4.14 shows the resulting errors of the prediction compared to the true values of the test samples for different latent dimensions. The first observation is that the trend of decreasing error for increasing latent for decoding as in Figure 4.8 is not reflected when the latent dimension is increased for prediction. It becomes apparent that an increasing higher dimensional latent vector representation beyond 32 worsens the prediction error but the magnitude of difference in errors can be considered small.

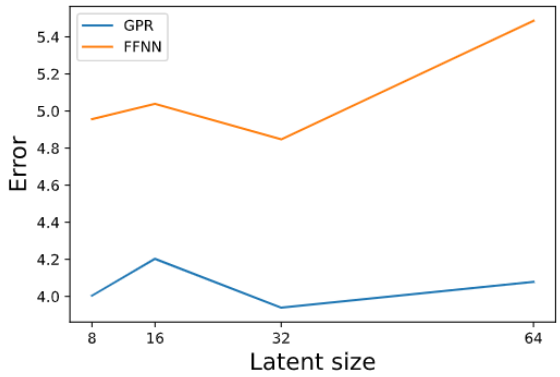


Figure 4.14: Effect of Latent space size on test samples

4.3.1 Results for 1000 nodes

In this section, an overview of the results obtained with 1000 nodes from the Hybrid III dummy is shown. As discussed in section 3 the dimensonality reduction methods and machine learning models were tested and tuned for a 1000 nodes initially. These formed the basis on which the surrogate for the full Hybrid III models were built.

4.3.2 Dimensionality reduction - 1000 nodes

To understand the loss of data due to dimensionality reduction, Figure 4.15 shows the MAE of decoded displacement values from PCs capturing 99.999% variance and true values.

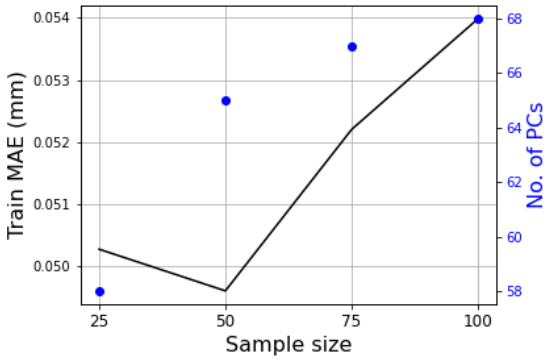
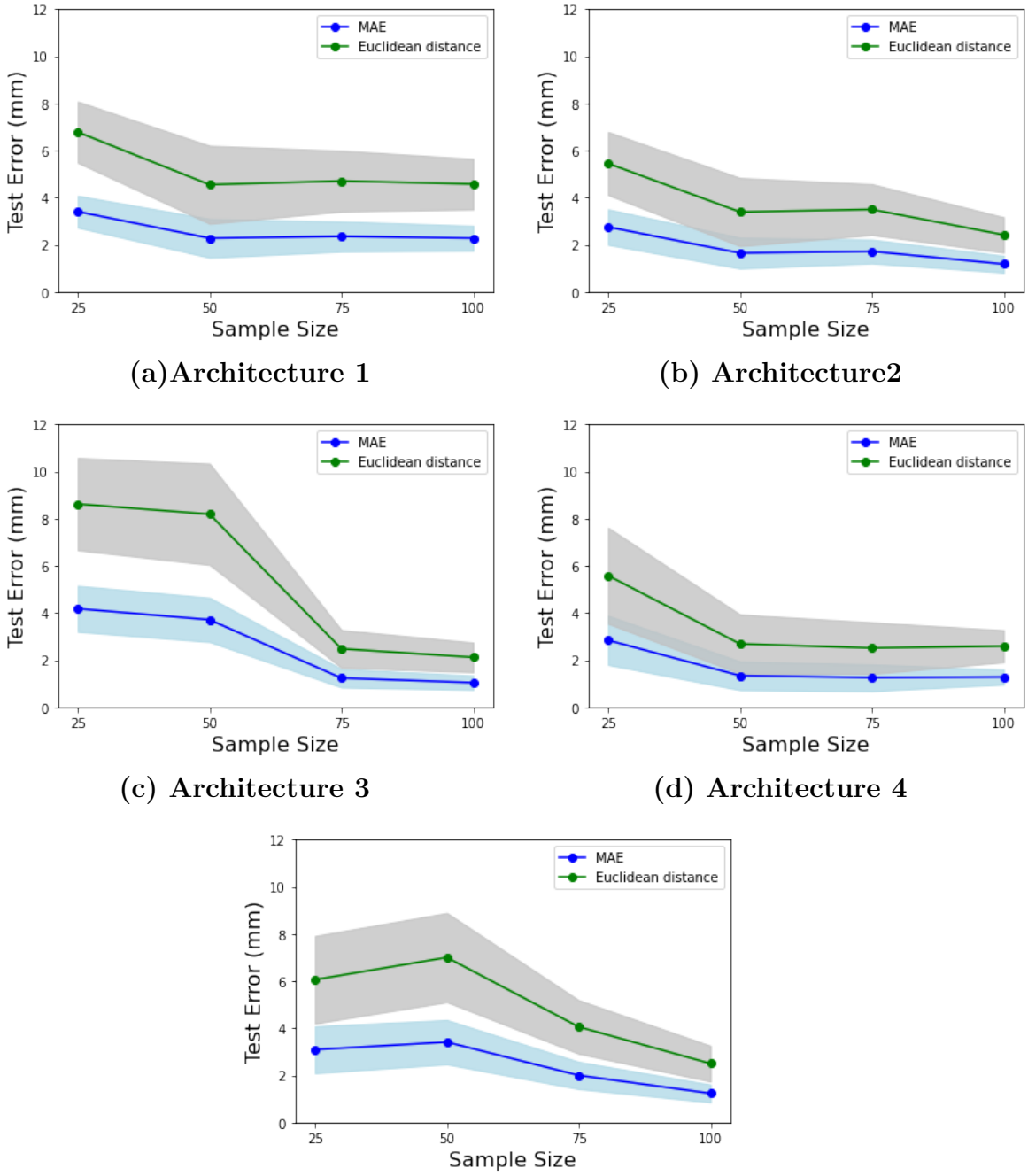


Figure 4.15: MAE of decompressed displacements from PCA on training samples for No. of PCs required to capture 99.999% variance

To decide on the architecture to use for dimensionality reduction, the MAE and mean Euclidean distance error of the true and decoded test samples is given in Figure 4.16. It can be observed that architectures 2 and 4 have better errors i.e., there is lesser loss of data due to the dimensionality reduction compared to the other architectures from a sample size of 50. Another observation that can be made from the plots is the decrease in mean error with increase sample size in nearly all the cases.

Also, further increase of sample size beyond 50, leads to no significant reduction in mean and variance in most cases. After studying these results, architecture 4 is considered as the chosen model as the mean error remains consistent and variance reducing from a sample size of 50 and beyond.



(a)Architecture 1

(b) Architecture2

(c) Architecture 3

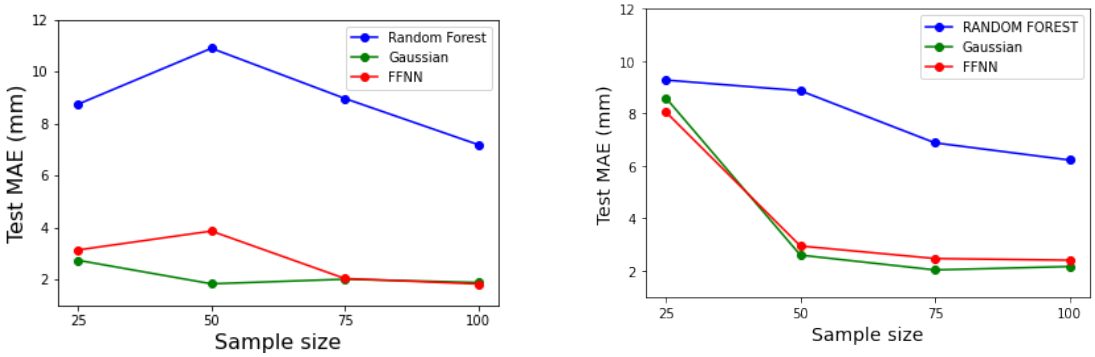
(d) Architecture 4

(e) Architecture 5

Figure 4.16: Error of decoded displacements from CAE-1D against true for test samples

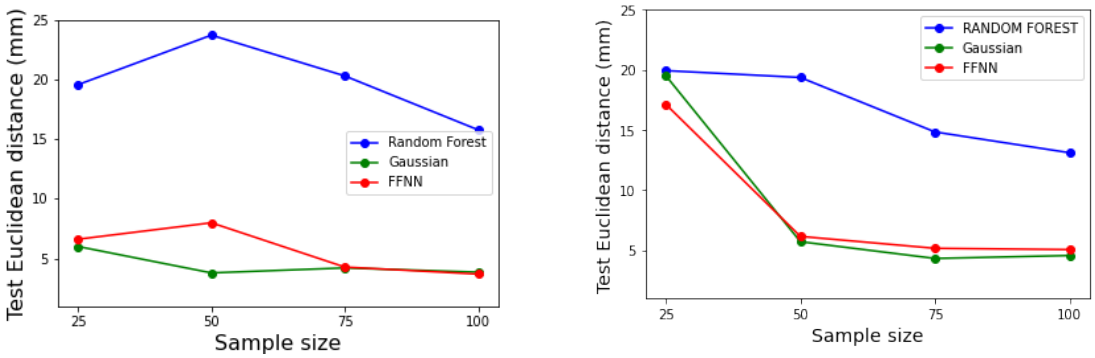
4.3.3 Machine Learning - 1000 nodes

For 1000 nodes, GPR and FFNN perform better for all sample sizes as seen in Figures 4.17 and 4.18. This trend remains the same for both PCA and autoencoder based surrogates. The same is observed for the full Hybrid III results. Additionally, for AE based ML models, there is a steep decline in the error from a sample size of 25 to 50 for both GPR and FFNN.



(a) PCA performance (b) AE performance

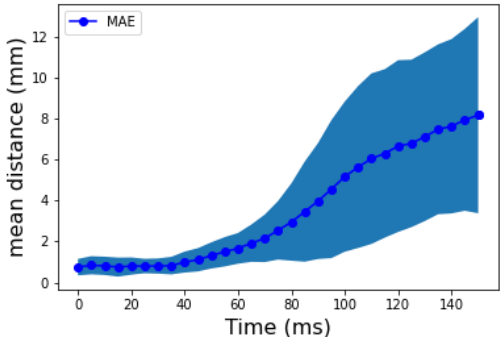
Figure 4.17: For 1000 nodes, MAE of test samples with respect to sample size



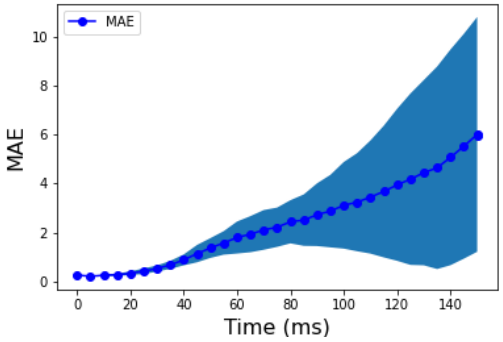
(a) PCA-FFNN performance (b) AE-FFNN performance

Figure 4.18: Mean euclidean distance of test samples from PCA and AE based surrogate models with respect to sample size, For 1000 nodes

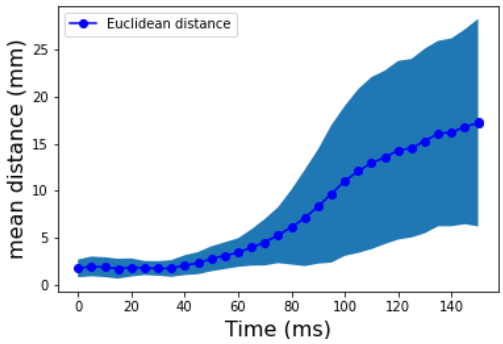
To understand the accuracy of prediction at every timestep, Figure 4.19 shows the MAE and mean Euclidean distance of all nodes at every time state for both PCA and AE with the FFNN model. It can be observed that until about 60 ms the error and variance remain significantly lower compared to after 60 ms. For PCA-FFNN, the mean error and variance are larger at 0 ms compared to the values for AE-FFNN. This trend is similar even for the full model as discussed in the previous section.



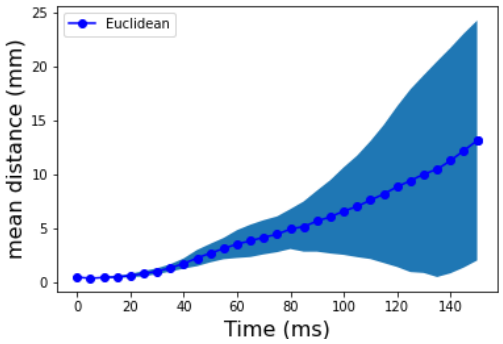
(a) MAE - PCA



(b) MAE - AE



(a) Euclidean distance - PCA



(b) Euclidean distance - AE

Figure 4.19: ML prediction error across every time step for 1000 nodes - FFNN

5
Discussion

Dimensionality reduction methods and ML models were tuned and tested for various cases to achieve good predictions. The effectiveness of dimensionality reduction methods PCA and AE are affected by various factors. For PCA, arrangement of data, number of principal components and for AE, architecture, latent space and other such parameters are these factors. PCA and AE work good at compression and decompression of data, requiring a different size of low dimension space to achieve a similar level of error between the decompressed data and the true values of displacements. For PCA increasing the size of low dimension space by considering more PCs improves the accuracy of decompression. On the contrary for AE, with an increase in latent dimension the error of decompression decreases but with a small magnitude.
Having a larger latent space size to take advantage of the improved accuracy leads to some complexities in the ML process. Since AE uses neural networks to learn from the given data, most of the learning is stored in the encoding-decoding model. The latent space values from AE are the observations used by the ML model to learn from along with the FE parameters. The lesser number of latent space dimension means the easier it is for the ML model to create the mapping function. A larger number of latent space dimension means a larger scope of errors in predicting an equally larger latent output. This can be seen with the increase of latent space, leading to a higher error compared to the smaller latent size and not decreasing as seen with the decompression error. The latent space size effects were checked on a sample size of 50 simulations i.e., a train sample size of 40 simulations. Increasing the number of training samples to check the effect of latent size may differ the outcome of the test error due to having more learning from various training samples.

Choosing an architecture for AE plays a significant role compared to the latent dimension. Both mean and variance of the error of decompression along with the effects of sample size need to be considered while tuning the architectures. The choice of architecture 4 was made as a consistent mean error with a reducing variance for a sample size of 50 and beyond was observed. An architecture with two consecutive convolution layers at the start followed by alternating pooling and convolution layers worked with the data used in this study. The consecutive convolution layers help in learning features from the data better before it is pooled. The same or similar architecture can be used as a baseline to compress data of similar nature.

Increasing the number of nodes on the surrogates built and tuned for 1000 nodes to the full Hybrid III model with 7441 nodes leads to comparable levels of MAE and

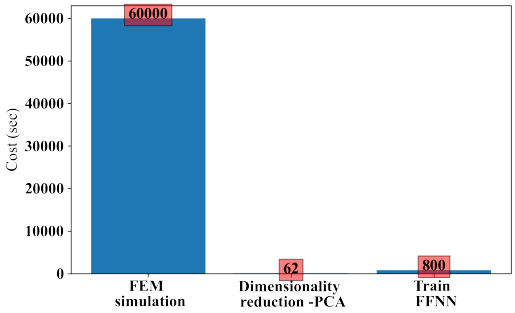
mean Euclidean distance in predicted values of test data as seen in Figures 4.17 and 4.9. This holds true for both PCA and AE paired ML models. GPR and FFNN perform significantly better compared to the tree based RFR in both PCA and AE based surrogates. And from a sample size of 50 simulations and beyond the mean error of test set predictions for both GPR and FFNN has minor variation. When comparing the PCA-ML and AE-ML surrogates, the latter performs considerably better. Tough the MAE and Euclidean distance metric for a sample size of 50 and beyond is comparable, the upper hand of AE-ML can be seen in when the error for all nodes across time is observed. AE-ML fares better at capturing the start of the simulations with least variance and has the least error at 0 ms time and a maximum mean error with larger variance compared to PCA. Contrarily, PCA-ML has a higher mean error and variance at 0 ms and has a slightly higher maximum mean error metric with lesser variance.

Implementing the above discussed method and models for building a surrogate to a full FE-Human Body Model (FE-HBM) such as a SAFER v9 or a THUMS model is feasible, as we saw that scaling the number of nodes from 1000 to 7441 did not lead to significant losses. These could be used as a strong baseline to build the surrogates for the FE-HBM. In this study 4 significant FE parameters were used to build the surrogate. Increasing the number of these parameters will not require significant other modifications as the ML models used to build the mapping function are capable of handling large data sets and input parameters. Another aspect to consider is the resolution of data collected from FE simulations, the current 32 time states for a crash of 150 ms i.e., nearly every 5^{th} ms, is adequate enough for the complexity of the displacements observed with a Hybrid III dummy. For an FE-HBM the resolution may remain considering the smoothness of the curves needed to capture the kinematics adequately. In case of a 1.5 sec precrash simulation, the resolution of data collection can remain. With a larger time frame of the event the number of data outputs gathered in terms of time states will be higher. The dimensionality reduction methods PCA and AE would work with comparable accuracy considering that PCA would require a larger number of PCs to account for 99.999% of variance. AE-1D would work the same as the direction of kernel motion is along with the time state data and thus would probably need minimal change including latent dimension. The computational requirements on the other hand would be more to generate the required number of samples for training and testing. Similarly, the storage for AE models would be larger along with an increase in processing time to generate the low dimensional space.

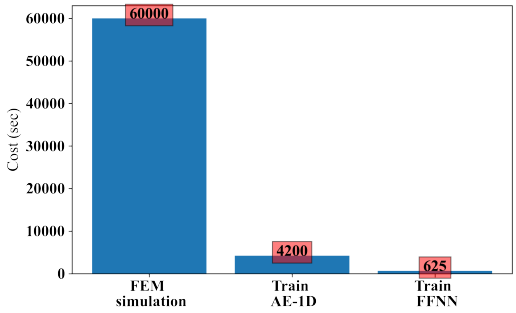
The magnitude of Euclidean distance gives a more 3-dimensional view of the distance between the true nodal displacement values and the predicted values. A peak mean Euclidean distance error metric of 20 mm as seen in the results might be very small considering the movement of the occupant during normal driving. But during a pre-crash event, leading to a crash, it might mean that the safety systems such as belt pre-tensioners, airbags etc., triggering early and lowering their performance or causing undue injury. Also, in an FE simulation involving pre-crash followed by crash, an error in prediction of the dummy model position can result in an increased

injury risk outcome from the crash than what would have been from a crash only scenario. This might lead to faulty calibration of safety systems or over designing of certain aspects. But the magnitude of allowable error from ML models needs to be understood in order to tune the process to obtain the right accuracy of predictions to obtain the right amount of computational resource and time advantage that is sought with such a process. Pre-crash kinematics of the FE human dummy are assumed to be less complex in their kinematics behaviour due to the nature of inputs when compared to intense in-crash scenario. Thus, a model built on intense kinematics behaviour from in-crash simulations such as ones shown in this work should work well for the less intense pre-crash kinematics.

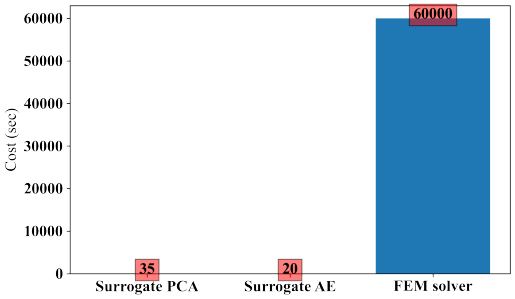
5.0.1 Computational Resource



(a) PCA and simulation cost



(b) AE training and simulation cost



(c) Testing cost

Figure 5.1: Total cost taken for generating, training and testing

To understand the computation costs and the savings that can be made by using an ML surrogte as illustrated in Figure 5.1. The cost of generating 100 simulations in LS Dyna solver is nearly 60000 sec on a PC with a Ryzen 3500 processor, 4 cores and 16 Gb of ram. Dimensionality reduction of 100 simulations using AE-1D architecture 4 was 4300 sec, training the compressed data from a same number of simulations with FFNN took nearly 650 sec on a GPU such as Tesla T4 offered by Google Colaboratory. Dimensionality reduction of 100 simulations with PCA was 62 sec and training the compressed data from PCA with FFNN took nearly 800 sec

on a PC with a Ryzen 3500 processor and 16 Gb of ram.

Once trained the AE model can be saved onto a drive. To predict new unseen test data, the trained model needs to be loaded from the drive. It takes close to 30 sec to predict the displacements of 100 test simulations. This time to predict holds good for GPR, RFR and FFNN ML models. This drastic reduction in computational cost is equivalent to a speed-up factor equal to 99.96%. On the other hand storage space required to save the trained AE model is nearly 2.5 GB for a sample size of 50 simulations compared to above 10 GB to store the d3plots for the same number of simulations.

It is crucial to note that all results obtained for AE-ML models were from using a GPU on Google colaboratory. This is to utilize the advantages of faster processing times of a GPU when using tensorflow for large data sets and in particular ML problems. When the dimensionality reduction was done on a standalone PC with Intel i5 8265u processor with 4 cores, the time taken ranged with an average of 6-8 hours.

6
Conclusion

The aim of the thesis is to predict the nodal kinematic time histories of a FE human body model using surrogate ML models trained on in-crash FE simulations data. A machine learning workflow involving dimensionality reduction methods along with ML models was developed to train, test and evaluate the surrogate model.

Surrogate models were built with different dimensionality methods and they were compared based on the MAE and mean Euclidean distance error of decompressed data from the low dimension space for different sample sizes. Autoencoder was tested for multiple architectures and the chosen one was compared with PCA. For PCA, principal components needed to account for 99.999% of variance in sample data was considered for this comparison.

- PCA and AE are comparable in their ability to compress given data. AE has a significant advantage of being a neural network based dimensionality reduction method wherein it learns from the given data while compressing it. Its disadvantage is that it is highly resource intensive unless a GPU is used for processing.

- PCA has an advantage in that it requires very low computational time and resources but the disadvantage is that there is no learning aspect involved. Hence, the true comparison of these methods can be made when they are paired with the ML models.

- GPR, RFR and FFNN were paired with PCA, AE and evaluated on the unseen test data using MAE and mean Euclidean distance error for varied sample sizes. Considering the error of all nodes of all test samples across time states, AE-ML models had a comparably lower error and variance across time states, especially in the first half of the 150 ms crash.

- PCA-ML had a much higher error and variance overall in this category. GPR and FFNN proved more accurate compared to RFR for a simulation size of 50 and beyond with no much variation in the error. This is true when paired with both PCA and AE.

7
Future Work

Hybrid III M50 Fast model was used in this study considering 4 major FE parameters that influence the kinematics. These parameters were used to both generate the training samples and predict the outcomes from ML models. There can be more FE parameters that can influence the kinematics of the dummy. Using these FE parameters will need more hyperparameter tuning of the ML models. The Hybrid III dummy model used contains 7441 nodes but an FE-HBM such as SAFER v9 or THUMS model have a considerably higher number of nodes and complexities. The current methods and models discussed in this thesis can be considered as a baseline and extended to the FE-HBMs.

The scalability of the models and methods from 1000 nodes to 7441 can be utilized on FE-HBMs by considering a few select nodes from the HBM to build a surrogate and then use morphing tools to set the FE-HBM in desired positions based on the ML predictions. This can help in reducing the hyperparameter tuning and improving model performance.

Convolutional Autoencoder 1D was used in this study to explore its dimensionality reduction performance. Implementing a Convolutional Autoencoder 2D as a dimensionality reduction tool would be an interesting study to undertake. Similarly, alternative ways of data arrangement for PCA can lead to interesting results.

Bibliography

[1] *Road Safety Facts* - Association for safe international road travel. Weblink: https://www.asirt.org/safe-travel/road-safety-facts/

[2] Kopits, Elizabeth; Cropper, Maureen. *Traffic Fatalities and Economic Growth.* 2003.Policy Research Working Paper;No. 3035. World Bank, Washington, DC. © World Bank. https://openknowledge.worldbank.org/handle/10986/18267 License: CC BY 3.0 IGO

[3] Philipp Spethmann, Stefan H. Thomke, Cornelius Herstatt. (2006). *The impact of crash simulation on productivity and problem-solving in automotive R&D.* Technische Universität Hamburg-Harburg

[4] Amir Sobhani1, William Young1, David Logan *Exploring the relationship of conflict characteristics and consequent crash severity.* Australasian Transport Research Forum (ATRF), 34th, 2011, Adelaide, South Australia, Australia

[5] Pronoy. G et al. *A proposal for integrating pre-crash vehicle dynamics into occupant injury protection evaluation of small electric vehicles.* IRCOBI Conference 2015. IRC-15-88

[6] Masami Iwamoto, Yuko Nakahira & Hideyuki Kimpara (2015) *Development and Validation of the Total HUman Model for Safety (THUMS) Toward Further Understanding of Occupant Injury Mechanisms in Precrash and During Crash, Traffic Injury Prevention*, 16:sup1, S36-S48, DOI: 10.1080/15389588.2015.1015000

[7] Brolin K, Östh J, Mendoza-Vazquez M. HUMAN BODY *MODELING FOR APPLIED TRAFFIC SAFETY.* In p. 13–15.

[8] Brunton, S., & Kutz, J. (2019). *Reduced Order Models (ROMs).In Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (pp. 375-402). Cambridge: Cambridge University Press. doi:10.1017/9781108380690.012

[9] Nikolopoulos, S., Kalogeris, I., & Papadopoulos, V. (2021). *Non-intrusive surrogate modeling for parametrized time-dependent PDEs using convolutional autoencoders* arXiv preprint arXiv:2101.05555.

[10] Prasad P, Belwafa JE. *Vehicle Crashworthiness and Occupant Protection* Southfield, Michigan: Automotive Applications Committee American Iron and Steel Institute; 2004.

[11] Roth, Sebastien & Chamoret, Dominique & Imbert, JR & Gomes, Samuel.(2011). *Crash FE Simulation in the Design Process - Theory and Application* 10.5772/23813.

[12] Tom M. Mitchell, *The Discipline of Machine Learning.* July 2006 CMU-ML-06-108

[13] Stensrud, D. (2007). *Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models.* Cambridge: Cambridge University Press. doi:10.1017/CBO9780511812590

[14] Sorzano, C. O. S., Vargas, J., & Montano, A. P. (2014). *A survey of dimensionality reduction techniques.* arXiv preprint arXiv:1403.2877.

[15] Baker RE, Peña JM, Jayamohan J, Jérusalem A. *Mechanistic models versus machine learning, a fight worth fighting for the biological community?*. Biol Lett. 2018;14(5):20170660. doi:10.1098/rsbl.2017.0660

[16] Kracker, D., Garcke, J., Schumacher, A., & Schwanitz, P. Automatic Analysis of Crash Simulations with Dimensionality Reduction Algorithms such as PCA and t-SNE.

[17] Ringnér, M. *What is principal component analysis?*. Nat Biotechnol 26, 303–304 (2008). https://doi.org/10.1038/nbt0308-303

[18] Martínez-Martínez, F., Rupérez-Moreno, M. J., Martínez-Sober, M., Solves-Llorens, J. A., Lorente, D., Serrano-López, A. J., ... & Martín-Guerrero, J. D. (2017). A finite element-based machine learning approach for modeling the mechanical behavior of the breast tissues under compression in real-time. Computers in biology and medicine, 90, 116-124.

[19] KIM ESBENSEN and PAUL GELADI. *Principal Component Analysis* Research Group for Chemometrics, Institute of Chemistry, Umei University, S 901 87 Urned (Sweden). Chemometrics and Intelligent Laboratory Systems, 2 (1987) 37-52

[20] https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad

[21] M.F.M. Alkbir, S.M. Sapuan, A.A. Nuraini, M.R. Ishak, *Fibre properties and crashworthiness parameters of natural fibre-reinforced composite structure: A literature review, Composite Structures,* Volume 148, 2016, Pages 59-73, ISSN 0263-8223, https://doi.org/10.1016/j.compstruct.2016.01.098. (https://www.sciencedirect.com/science/article/pii/S0263822316300034)

[22] K. Morooka, X. Chen, R. Kurazume, S. Uchida, K. Hara, Y. Iwashita, M. Hashizume *Real-time nonlinear FEM with neural network for simulating soft organ model deformation* Medical Image Computing and Computer-Assisted Intervention – MICCAI 2008: 11th International Conference, New York, NY, USA, September 6-10, 2008, Proceedings, Part II (2008), pp. 742-749

[23] `https://www.sas.com/en_us/insights/analytics/machine-learning.html` machine-learning-workings

[24] `https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/`

[25] Donata Gierczycka & Duane Cronin (2021) *Importance of impact boundary conditions and pre-crash arm position for the prediction of thoracic response to pendulum, side sled, and near side vehicle impacts*, Computer Methods in Biomechanics and Biomedical Engineering, DOI: 10.1080/10255842.2021.1900132

[26] Amitha, M. (2012). *Applications of finite elements method (FEM) -An Overview* Conference: "International Conference on Mathematical Sciences, Dec 28-31, 2012"-Nagpur, India. Affiliation: Shivaji Science College, Nagpur. DOI: 10.13140/RG.2.2.36294.42565

[27] Porkolab, Laszlo & Lakatos, Istvan. (2021). *Vehicle occupant safety development with finite element method.* Pollack Periodica. DOI: 10.1556/606.2021.00306.

[28] J.S.Hesthavena, S.Ubbialia *Non-intrusive reduced order modeling of nonlinear problems using neural networks* https://doi.org/10.1016/j.jcp.2018.02.037

[29] Guo, M., & Hesthaven, J. S. (2018). *Reduced order modeling for nonlinear structural analysis using Gaussian process regression.* Computer methods in applied mechanics and engineering, 341, 807-826.

[30] Opitz,D. ; Maclin, R. (1999). *Popular Ensemble Methods: An Empirical Study* Journal of artificial intelligence research,11: 169-198. DOI: https://doi.org/10.1613/jair.614

[31] Mehlig, B. (2019). Artificial neural networks. arXiv e-prints, arXiv-1901.

[32] Singh K, Xie M. *Bootstrap a statistical method* Unpublished manuscript, Rutgers University, USA. Retrieved from http://www. stat. rutgers. edu/home-/mxie/RCPapers/bootstrap. pdf. 2008:1-4.

[33] Zoghni, Raouf (September 5, 2020). *Bagging (Bootstrap Aggregating), Overview.* The Startup – via Medium.

[34] Edward, C. (2006). Rasmussen and Christopher KI Williams. Gaussian processes for machine learning. MIT Press, 211, 212.

[35] Yash N P, Akhil S, https://libraries.io/pypi/dynakit

[36] Chen, Z., & Wang, B. (2018). How priors of initial hyperparameters affect Gaussian process regression models. Neurocomputing, 275, 1702-1710.

[37] @ARTICLEGPflow2017, author = Matthews, Alexander G. de G. and van der Wilk, Mark and Nickson, Tom and Fujii, Keisuke. and Boukouvalas, Alexis and León-Villagrá, Pablo and Ghahramani, Zoubin and Hensman, James, title = "GPflow: A Gaussian process library using TensorFlow", journal = Journal of Machine Learning Research, year = 2017, month = apr, volume = 18, number = 40, pages = 1-6, url = http://jmlr.org/papers/v18/16-537.html

[38] Yoo, S., Lee, S., Kim, S., Hwang, K. H., Park, J. H., & Kang, N. (2021). Integrating deep learning into CAD/CAE system: generative design and evaluation of 3D conceptual wheel. Structural and Multidisciplinary Optimization, 1-23.

[39] Pedregosa et al., JMLR 12, pp. 2825-2830, 2011 https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[40] Reddy J. N. (2019). Introduction to the finite element method. McGraw-Hill Education.

[41] Saadat, M. N.,& Shuaib, M. (2020). Advancements in Deep Learning Theory and Applications: Perspective in 2020 and beyond. Advances and Applications in Deep Learning, 3.

[42] Jin, L., & Liang, H. (2017, June). Deep learning for underwater image recognition in small sample size situations. In OCEANS 2017-Aberdeen (pp. 1-4). IEEE.

[43] Wu, C. S., Madotto, A., Hosseini-Asl, E., Xiong, C., Socher, R., & Fung, P. (2019). Transferable multi-domain state generator for task-oriented dialogue systems. arXiv preprint arXiv:1905.08743.

[44] Oyedotun, O. K., & Dimililer, K. (2016). Pattern recognition: invariance learning in convolutional auto encoder network. International Journal of Image, Graphics and Signal Processing, 8(3), 19-27.

A

Appendix

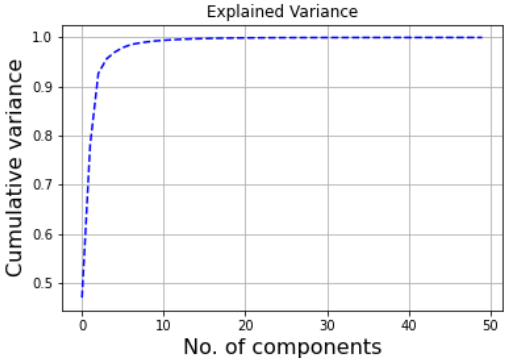
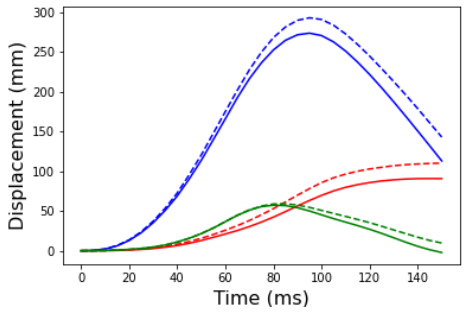
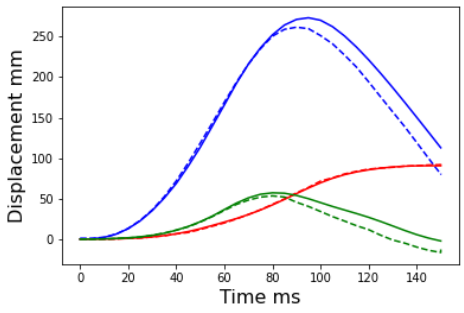


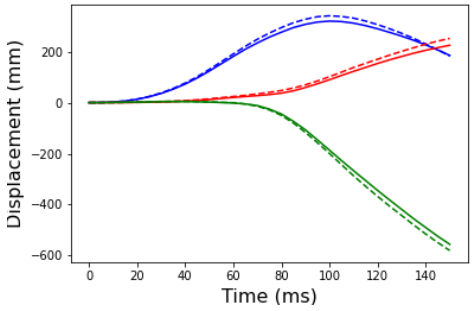
Figure A.1: Cumulative variance of the first 50 components out of over 1200 PCs. Considering 1000 Nodes of the FE Hybrid III dummy



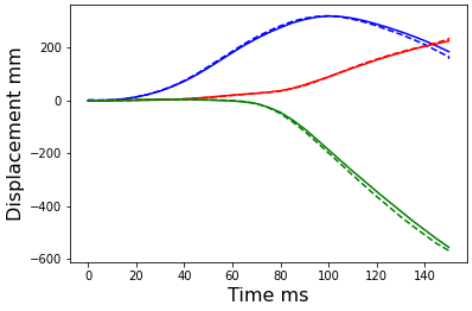
(a) Head Node 1000001 - PCA



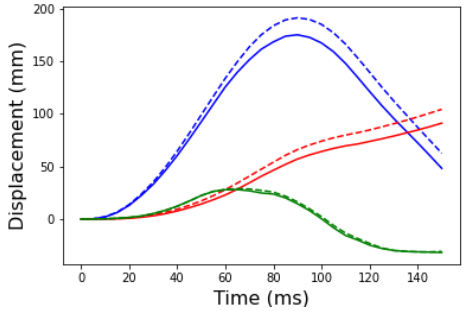
(b) Head Node 1000001 - AE



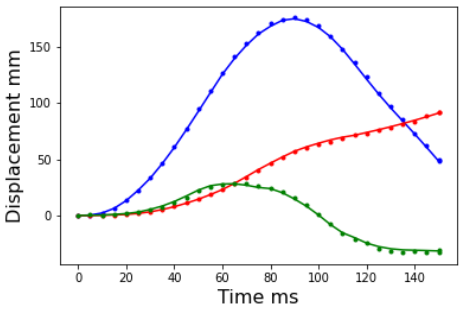
(a) Left fingertip 1002266 - PCA



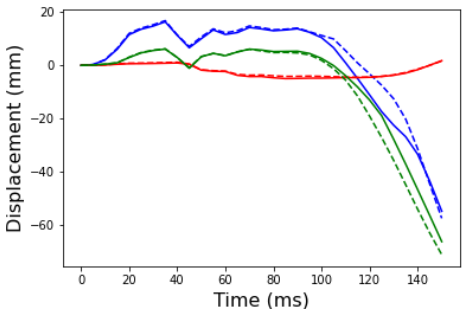
(b) Left fingertip 1002266 - AE



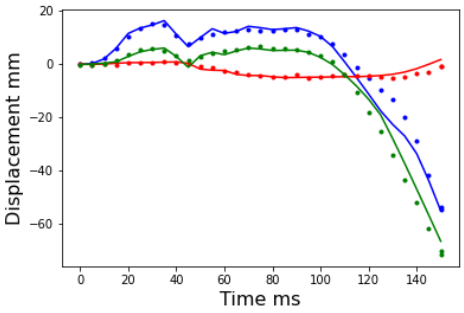
(a) Left shoulder 1002467 - PCA



(b) Left shoulder 1002467 - AE



(a) Left toe 1009569 - PCA



(b) Left toe 1009569 - AE

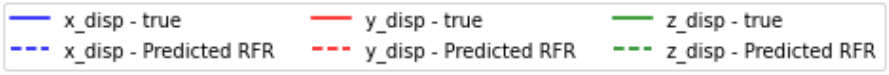
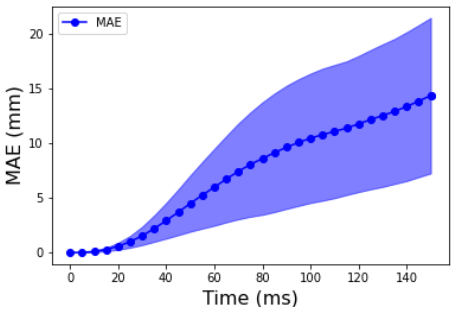
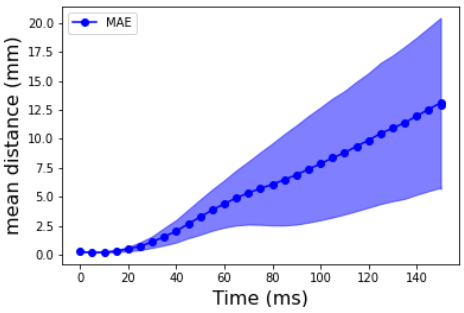


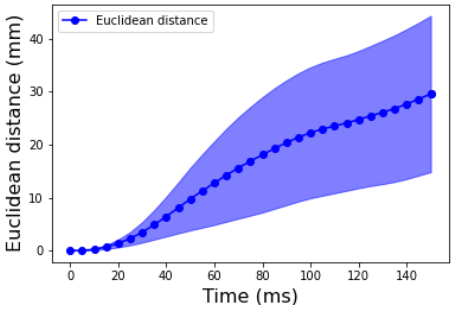
Figure A.2: ML prediction versus FEM results of PCA and AE based Random Forest Regressor for one test sample for sample size of 50



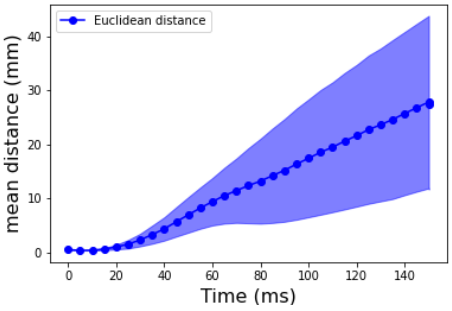
(a) MAE - PCA



(b) MAE - AE

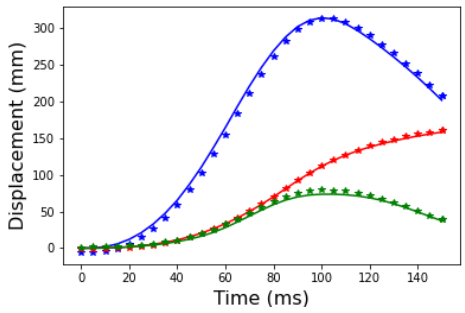


(a) Euclidean distance - PCA

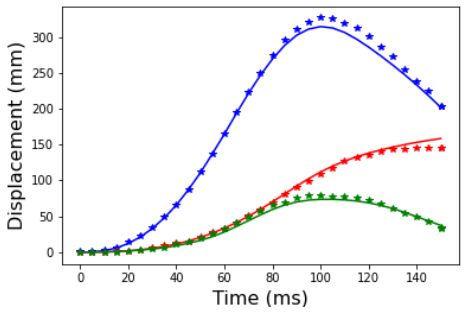


(b) Euclidean distance - AE

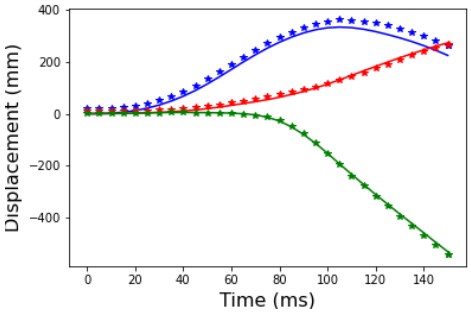
Figure A.3: ML prediction versus FEM results of PCA and AE based Random Forest Regressor for one test sample for sample size of 50



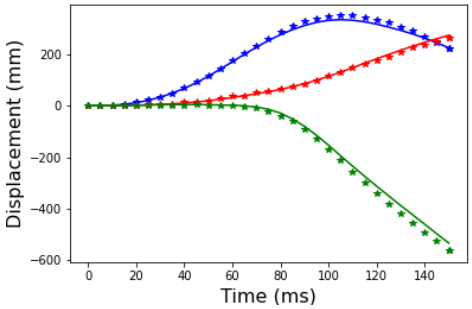
(a) Head Node 1000001 - PCA



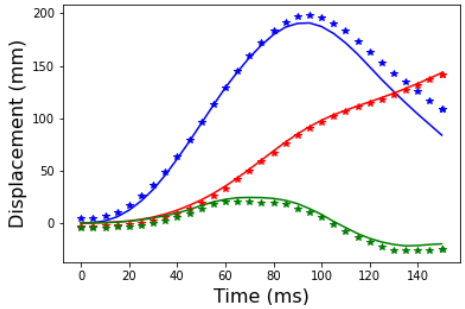
(b) Head Node 1000001 - AE



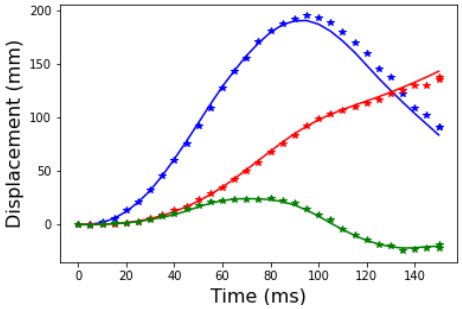
(a) Left fingertip 1002266 - PCA



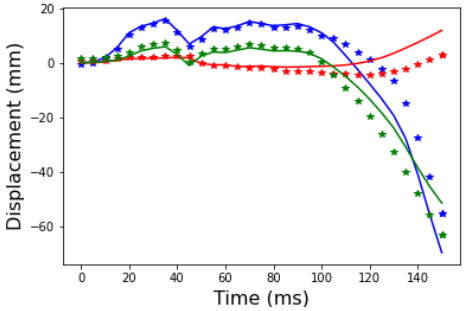
(b) Left fingertip 1002266 - AE



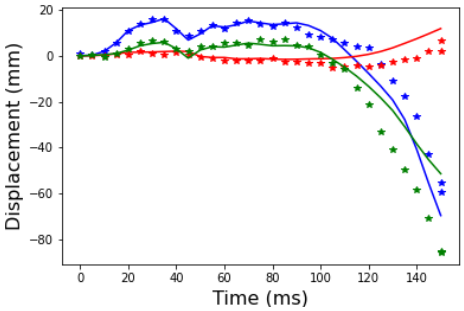
(a) Left shoulder 1002467 - PCA



(b) Left shoulder 1002467 - AE



(a) Left toe 1009569 - PCA



(b) Left toe 1009569 - AE

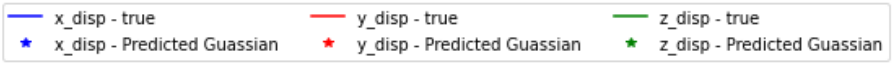
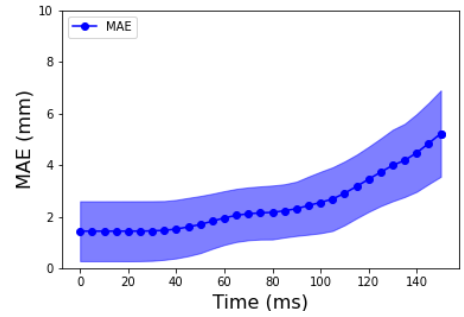
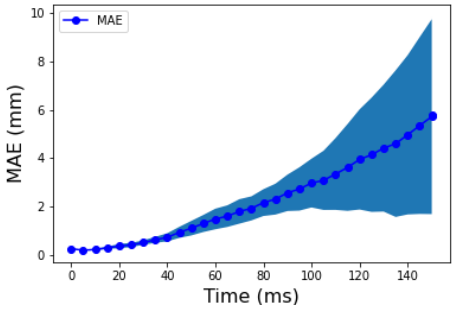


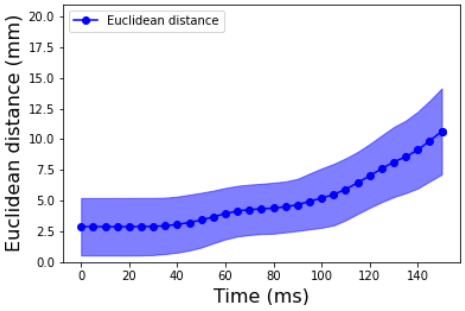
Figure A.4: ML prediction versus FEM results of PCA and AE based Gaussian Process Regressor for one test sample for sample size of 50



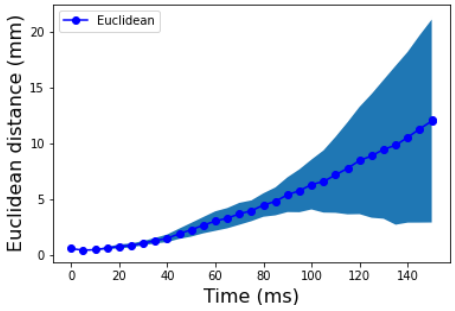
(a) MAE - PCA



(b) MAE - AE



(a) Euclidean distance - PCA



(b) Euclidean distance - AE

Figure A.5: ML prediction versus FEM results of PCA and AE based Gaussian Process Regressor for one test sample for sample size of 50

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY