An artificial intelligence framework for vehicle crashworthiness design

by

Timofei Liusko

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master Of Applied Science

in

Systems Design Engineering

Waterloo, Ontario, Canada, 2021

AUTHOR'S DECLARATION

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

STATEMENT OF CONTRIBUTIONS

This work was a part of research, conducted in the Computational Mechanics Research Group. Initial idea of the AI framework, as well as inspiration of it's basic architecture comes from that work, co-authored by my supervisor Professor Kaan Inal, Dr. Christopher Kohar – a post-doctoral fellow, Daniel Connolly - a PhD researcher, and myself. Part of this work was published as "Using Artificial Intelligence to Aid Vehicle Lightweighting in Crashworthiness with Aluminum" on 5 November of 2020 at the 17th International Conference on Aluminium Alloys 2020 (ICAA17). After that, I worked on training optimization mostly alone. Professor Inal supervised all of this work, provided necessary resources, contributed to the review and editing of the manuscript, and handled the project administration and funding acquisition. Dr. Kohar closely guided this project, provided invaluable advice, provided guidance in the original drafting of the manuscript, reviewed and edited the manuscripts to prepare it for publication, guided the conceptualization, aided in the development of the methodology and analysis, created some of the data visualizations, and provided the LS-Dyna simulations material which were used as dataset in this research.

ABSTRACT

Numerical crash test simulations are crucial for vehicle safety design. In the automotive industry, frameworks based on finite element methods are most common as they are precise and reliable. A few of the setbacks are simulation time and computation resources required for simulation. This thesis presents an artificial intelligence framework that utilizes recurrent neural networks to reduce the time and computational resources required to predict axial crash tests on the LS-DYNA models of thin-walled UWR4-like aluminum extrusion profiles. In addition, the work provides an overview of several data preprocessing techniques aiming to improve framework training time; ensembling of neural networks for the framework is explored as an addition to data preprocessing to improve framework performance. The thesis includes a detailed description of the data used and the machine learning models utilized in the framework. Three different sampling techniques are compared to reduce the time required to train the framework – two variants of random sampling and importance sampling; model ensembling is explored to improve accuracy on framework trained on data samples. Experiments show that the artificial intelligence framework reduces the time required to obtain one simulation of an axial crash test by the factor of 270, with a tradeoff of accuracy. Additional experiments on data preprocessing and model ensembling show that the training time of the framework could be reduced from 111 hours to 37 minutes for a single sample or 3 hours for a models ensemble with an additional cost of accuracy.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Kaan Inal for giving me the opportunity to be part of the Computational Mechanics Research Group (CMRG) at the University of Waterloo and for his leadership and support.

This work was a part of a wider research, conducted at the CMRG. I would like to thank Dr. Christopher Kohar and Daniel Connolly for their help and sharing knowledge during this project. Special thanks to Dr. Kohar for his guidance in this project and for asking tricky questions, pushing project quality. Without his help, this project wouldn't be possible.

Thank you to all of the members of the CMRG, especially Olga Ibragimova for recommending and introducing me to the group and endless support she gave.

Contents

List of Figures

List of Tables

1 Introduction

By the beginning of the 21st century, the problems of irreversible climatic changes caused by human activities have riveted the close attention of the entire world community. One of the points of concern of the ecologists is the ever-increasing emission of greenhouse gas (GHG), which, according to several studies, may destructively impact the world economics and climate [1, 2]. Thus, various government policies have aimed to reduce the amount of GHG produced by different economic sectors. In particular, such mandate exists towards the automotive industry, as, according to the Intergovernmental Panel on Climate Change (IPCC) report [3], dated by 2014, the automotive industry overall contributes about 14.6% to the total GHG emission. A common practice among car manufacturers is structural vehicle lightweighting, which betters the fuel economy and diminishes the operation cost, thereby reducing the amount of GHG produced by the vehicle [4]. However, vehicle lightweighting is always coupled with passenger safety issues, which significantly complicates the task from an engineering point of view, and forces vehicle manufacturers to invest money into the research in vehicle design optimization.

From the point of view of consumer safety, optimal vehicle design should be targeted to mitigate a *crash pulse*, which emerges in the event of a vehicle collision and propagates through the vehicle. Characteristics of the crash pulse transferred to the passenger during the crash event directly impact the chance to get a severe injury [5]. The modern vehicle possesses complex structury with thousands of structural constituents, so the mitigation of crash pulse should be a result of the interaction of the parts as a whole, aimed to either dissipate the energy of the pulse with its deformation or to divert the energy away from the passenger [6]. Therefore, automakers need to investigate how these innovative lightweight designs will affect the complete vehicle response during a collision.

The whole process of designing the structure of a vehicle component may somewhat be considered as an optimization problem of discovering the best set of options in the *design parameter space*, delivering a maximum to the *objective function*, which states the way to estimate the quality of a design. In some cases, the process of structural optimization may be reduced directly to the constrained optimization problem [7, 8]. In general, it is not possible neither to express the design space as a cartesian product of vector spaces nor to formulate the optimization problem, which delivers the best design to the vehicle. Therefore the only plausible algorithm for finding the best design is the iterative-based or "trial and error" design

loop. The iterative design process can be considered a robust optimization algorithm, similar to the random walk in design parameter space. Although researchers have attempted to construct a quantitative analysis of some aspects of the iterative design process [9, 10] based on Markov chains, concepts suggested in the scope of these works are not flexible enough to account for peculiarities of a process in the field of vehicle design.

To estimate the quality of a chosen vehicle design in real life, engineers study various quantitative characteristics of several crash tests, suggested by various standard-making organizations, such as The United States National Highway Traffic Safety Administration (NHTSA). In practice, designers also carry out tests for individual vehicle components, subjecting them to load conditions similar to real-world situations. However, building prototypes for each alternative design of a single member to run them into the crash test procedure cannot be viewed as an excellent industrial practice. Those tests, being destructive by nature, suffer both from high development costs and high time costs. Thus, at the moment, engineers rely on various computer-aided engineering (CAE) tools to run experiments virtually, modelling different crash scenarios before using vehicle prototypes in the actual experimental crashworthiness setups. The widespread instrument for conducting virtual experiments is the finite-element (FE) simulation, based on industrial numerical solvers, such as LS-DYNA [11]. FE analysis possesses a lower cost than conducting crashworthiness tests and gives the ability to explore various parameter configurations before assembling the actual prototype. However, mathematical problems being solved to run such simulations still require significant computational resources, which are highly non-linear.

Inspired by the recent advancements in machine learning (ML) applications to numerical methods, researchers have suggested a computationally practical approach allowing to run approximations FE simulations in mere seconds. As numerical experiments are usually aimed to discover particular characteristics of the processes under investigation, scientists came up with the idea to predict these characteristics directly from the parameters, describing the process [12, 13, 14]; thus, reducing the problems to the statement, typical for supervised learning. As supervised ML algorithms require training datasets to operate, researchers constructed them by running FE simulations, varying the parameters mentioned above. Pioneering work in applying this approach to numerical simulations of crash tests is authored by Kohar et al. [6]. In the scope of that work, data generated by numerical simulation of the frontal crash test of a pickup truck is used to train a neural network to predict the time-series response of the occupant crash-

pulse. However, the studies mentioned above only consider the possibility of solving similar problems using machine learning algorithms, omitting their computational cost, which is the defining parameter of the industrial applicability of an algorithm.

The purpose of this work is to investigate the applicability of a similar framework for predicting the time-series data of shift of the node in the numerical simulation of the axial crash of the thin-walled pipes obtained by extrusion of various aluminum profiles. Such problem statement arises from the series of works authored by Kohar et al., devoted to the development of the UWR-4 extrusion profile [16, 17, 18]. The work, similarly to [6], proposes the baseline, neural network model, employing the LSTM architectural pattern [19] to predict the shift evolution. Such a model is built using the Keras [20] deep learning framework and trained on the whole available data up to convergence. The study additionally suggests a methodology to improve the computational effectiveness of the framework, employing various sampling techniques. Further numerical experiments are represented with training instances of the suggested model on various subsets of the default dataset, obtained with varying the number of nodes sampled according to the designed sampling strategies. The study assesses both the change in the quality of solutions appearing due to applying these sampling methods and the computational effectivity of such procedures; comparison is made for all variations of the training process, using baseline experiment as a benchmark.

The thesis is structured in the following manner: Chapter 2 gives a literature review on the theme, the background information on FE simulations in crashworthiness, machine learning algorithms, and sampling techniques used in work, and outline existing deficiencies in literature. Chapter 3 serves as an outline of the scope and objectives of the work, identifying the gap in the literature it intends to close. Chapter 4 contains a detailed description of the AI framework suggested to solve the stated problem and provides the resulting quality metrics and convergence times. Chapter 5 summarizes sampling techniques applied to the framework for the sake of improving its computational efficiency. Finally, the critical discoveries of this research are presented in Chapter 6, and Chapter 7 gives recommendations on the further and adjacent research directions.

2 Background

2.1 Experimental Crashworthiness

New Car Assessment Program is a United States government-mandated program conducted by National Highway Traffic Safety Administration (NHTSA). This program is designed to provide safety information to the public and to improve occupant safety by providing market incentives for vehicle manufacturers to voluntarily design better crashworthiness into their vehicles [21]. The safety rating of a vehicle prototype is aggregated through a series of experiments, modelling the impact of the different types of collisions on the structure of an automobile - *the experimental crashworthiness testing.*

The benchmark of tests for vehicle crashworthiness is the frontal barrier crash test. This test can be considered a simulation of either a head-on vehicle collision or a vehicle's collision with a stationary wall, with varying angles of crashes, obstacle positions, and the velocities of the collision. The original version of the frontal barrier crash test was designed by NTHSA in 1978 [21]. In this version, vehicles were crashed into a fixed barrier at the speed of 56.3 kilometres per hour(km/h) (35 miles per hour (mph)) while being equipped with Hybrid II adult male dummies [22], situated in driver and front passenger seats. An illustration of the possible setup of this experiment is provided by figure 2.1

Figure 2.1: A possible setup for the NCAP frontal crash test - real-life vehicle (left), and the CAE model(right). Illustration taken from [23]

At the moment, several organizations are simultaneously developing various standards for assessing the quality of automotive products through impact tests. For example, the crashworthiness testing in Europe is governed by local lateral of NCAP, known as Euro-NCAP. Complementary to the federal NCAP program in the USA, vehicle safety assessment through impact experiments are held by the Insurance Institue for Highway Safety (IIHS). In addition to the frontal barrier crash test, each organization provides its reference experiments for vehicle

safety assessment. E.g., two tests conducted in the scope of IIHS policy are depicted by the figure 2.2

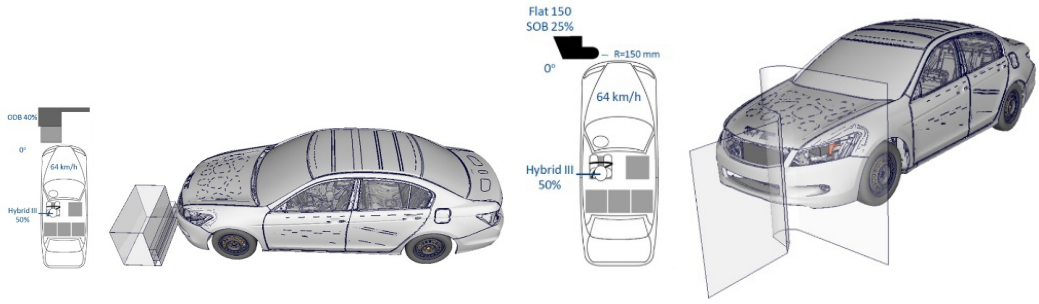


Figure 2.2: IIHS crashworthiness expreriments

Data obtained with crashworthiness tests is invaluable, as the structure of a modern vehicle is becoming more and more complex. However, at the moment, it cannot be said that such experiments are included in the standard pipeline for the design of new vehicles. The primary issue about using a full-scale series of experiments is cost. Crashworthiness testing requires conducting experiments on several prototype vehicles, and it can radically increase the total production cost of vehicles.

2.2 Component Experimental Crashworthinness

Since full-vehicle crash testing is a high-cost procedure, automotive designers additionally employ preliminary evaluation of crashworthiness of individual components included in the vehicle structure. Tubular multi-cell structures with thin walls are of particular interest for such testing due to their wide range of applications in the automotive and aerospace industry as lightweight energy-absorbing structures in crash environments. The energy absorption is the property of a structure or material to reduce the impact force by absorbing it or spreading it over a larger area. A typical example of such structure - an extrusion profile obtained from AA6063 aluminum alloy billet - can be seen in the figure 2.3. Over the past years, energy absorption has been thoroughly studied both theoretically, numerically, and experimentally [24, 25, 26, 27, 28]. Currently, thin-walled tubular structures are widely employed as major energy absorption structures of vehicles, such as front rails, rockets, and pillars.

The setup for such experiments is usually configured to subject the specimen to the loading conditions, akin to that which occurs during the full-scale crashworthiness testing. A summary

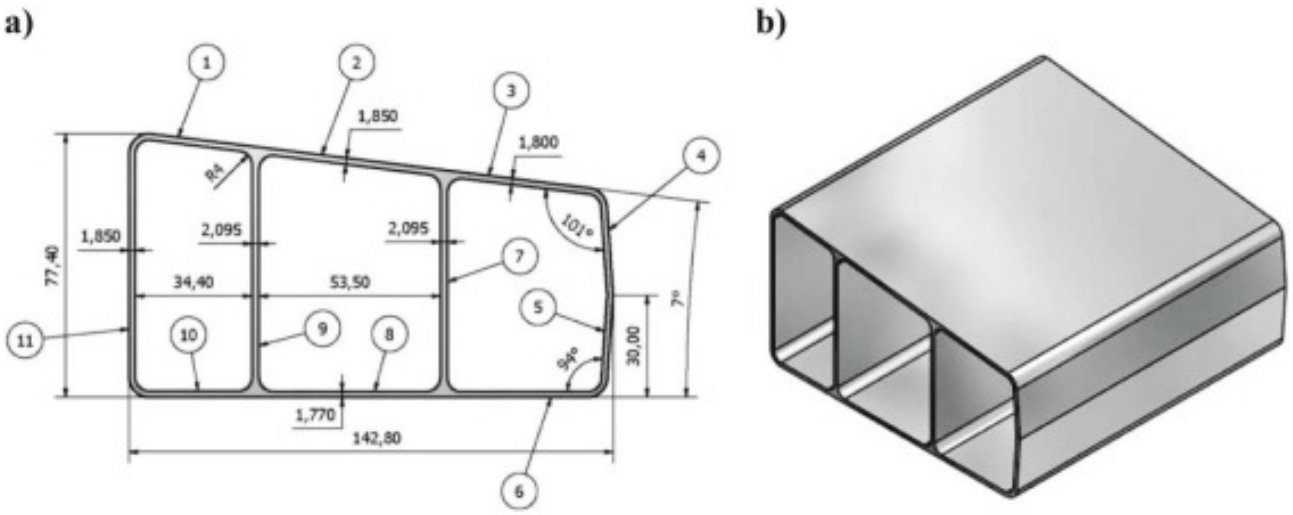
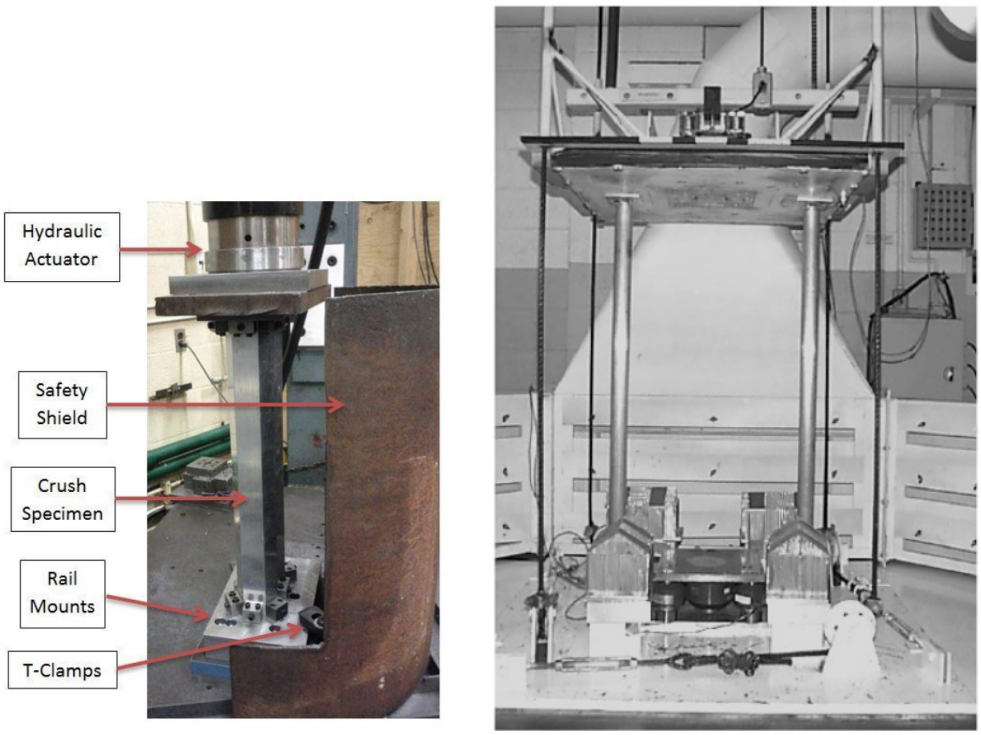


Figure 2.3: Extrusion profile obtained from AA6063 aluminum alloy billet - cross-section (a) and isometric view (b)

of possible loading conditions can be found in [29]. Concerning the tubular structures, setups encountered in the literature include, for example, quasi-static setup with a hydraulic press (see fig.2.4a), where a tube is attached to an immobile stiff plate from the bottom, and the second plate is compressing the tube with a prolonged speed in order to crash it [30]. For dynamic crash testing, a drop tower facility may be used [31]. Such setup suggests raising an impact mass of 80-100 kilograms above the specimen and releasing it to reproduce the velocity of impact similar to real-world collisions. (fig.2.4b). The crashworthiness data used in this work was obtained by modelling experiments in the linear crash sled apparatus located at the University of Waterloo (fig. 2.5). In this setup, instrumented aluminum extrusions were fixated between steel boss structured, and an impact mass of 855 kg was fired along the set of rails using compressed air, reaching the speed of 8 m/s (28 km/h) [15].

2.3 Numerical simulation of crashworthiness

Both full-scale and component-wise crashworthiness test suffers from the destructive nature of the procedure. It makes them poorly applicable for exploring design parameter space, as building new prototypes for minor design variations is unacceptably expensive and time-consuming. Over the past decades, vehicle design has been researched for the cheaper way to perform crashworthiness tests. It eventually resulted in the wide use of CAE techniques, such as numerical modelling, to run virtual crashworthiness experiments. Numerical modelling essentially limits the development cost for the design process only within the maintenance of



(a) (b)

Figure 2.4: a) Hydraulic Press [30] facility for quasi-static and Drop Tower [31] facility for dynamic component crashworthiness testing

computational hardware and software while generally being faster than real-life experiments, thus enabling much faster design space exploration.

The majority of experiments for engineering system design can be reduced to the initial-boundary value problem for a system of partial differential equations (PDEs), serving as a mathematical model for member properties in a fixed finite domain. In particular, FE simulations of crashworthiness usually employ the Cauchy momentum equation as a governing system of PDEs:

$$\ddot{\mathbf{u}} = \frac{1}{\rho}\nabla\sigma + \mathbf{f}_{\text{body}} \tag{2.1}$$

Here $\ddot{\mathbf{u}}$ is the acceleration vector, σ is the Cauchy stress tensor, f is the external body force, and ρ is the density of the medium. In the general case, there is no guarantee that an analytical solution for a given system of PDEs supplemented with initial-boundary conditions exists; thus, formulated problems are usually solved numerically.

The history of modern numerical crashworthiness simulation traces back to the works dedicated to studying energy absorption mechanisms of thin-walled tubular structures. The model of axial collapse of the thin-wall structure was developed by Wierzbicki and Abramowicz in [32,

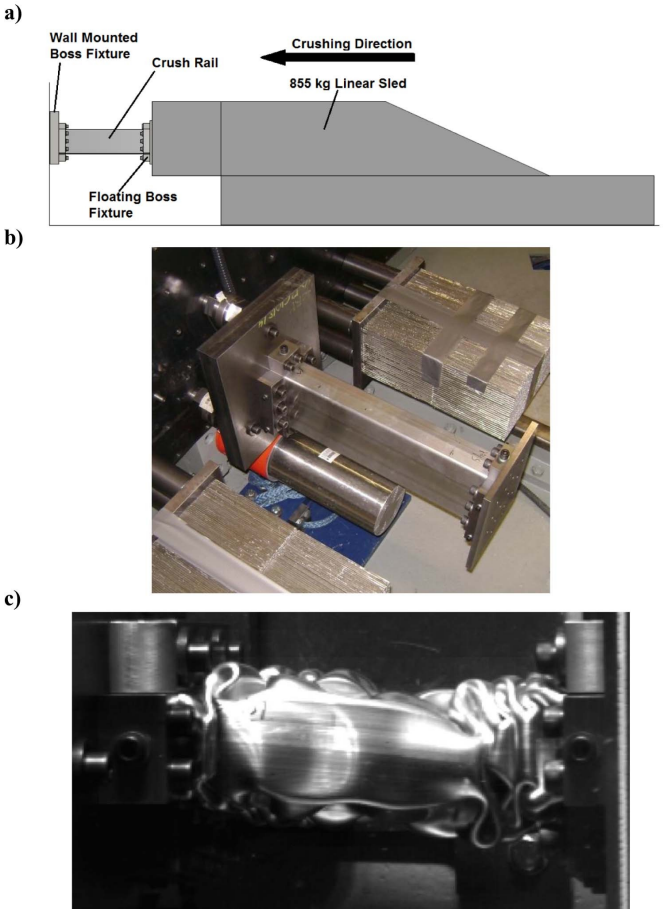


Figure 2.5: (a) Schematic of the sled-track testing apparatus, (b) experimental setup and (c) tube crashing during impact [15]

33, 34]; in the scope of these works a rigid-perfectly plastic material behaviour models were used to comprehend the fundamental mechanics behind the crashing of thin-wall structures. This model was later verified for the application to the quasi-static and dynamic crash experiments for tubes of various shapes by Abramowicz and Jones [35, 36, 37, 38]. Later, Abramowicz replaced the collapse mechanics from rigid-perfectly plastic to rigid-plastic, which resulted in improved accuracy of energy absorption prediction [39]. Following milestones were achieved by Yamashita, Gotoh, and Sawairi [40], and Najafi and Rais-Rohani [41], analytically relating the crashing strength response to the tensile strength of the material and incorporating advanced deformation mechanisms to improve analytical axial crash predictions, respectively.

After the emergence of commercial non-linear FE computational program complexes (LS-DYNA [11], PAM-CRASH [42], and ABAQUS [43]), various simulations researchers were able to account for the non-linear influences of contact and elastic-plastic constitutive models, novel to the previous works of axial crash [44, 45, 46]. The conventional definitions of such terms, as crash efficiency and energy absorption, for crashworthiness analysis of axial crash tubes were

introduced by Langseth, Hopperstad, and Hanssen while studying the axial crash of aluminum and steel thin-walled structures using LS-DYNA code [47, 48]. Williams et al. [49] applied similar principles to perform a numerical and experimental analysis of the effects of material anisotropy.

The work [50] by Fyllingena et al. revealed that using more elaborate element formulations, such as solid elements and shell elements, including a linear through-thickness strain distribution, resulted in increasing quality of predicting the force-displacement response, compared to the plane stress shell elements. Kohar et al. [16] studied the effects of elastic-plastic behaviour on the axial crash response of square tubes, varying such statement parameters as yield strength, ultimate tensile strength, hardening rate, and failure strain. This work highlighted that if materials exhibit low hardening capability, there is a tradeoff between the crash efficiency and the energy absorption; however, increasing the material's yield stress boosts its crash efficiency in the alternative case. In [51] Kohar et al. investigated the effects of yield function curvature and anisotropy on the crash response of circular aluminum tubes. The study shows that the prediction of axial crash response is strongly affected by the shape of the yield surface, as altering it has a significant effect on the stress and strain states of the material. Thus, accurate axial crash predictions cannot be carried out by providing only the material anisotropy information. In addition, it was discovered that biaxial balance tension is the most crucial anisotropy parameter for accurate first-order predictions of energy absorption.

As mentioned above, such numerical simulations are often carried out to discover characteristic values describing the behaviour of parts of the vehicle subjected to corresponding loads. For example, Kohar et al. used the following set of metrics to estimate the quality of aluminum rail profile - energy absorption, mean crash force, peak crash force, and crash efficiency. The crash force appears in the collision, the deceleration of colliding structure multiplied by its mass. Crash efficiency is the ratio of stresses that measure how much force travels through an impact protecting material. Further, these metrics may be used to optimize profile geometry directly, using, for example, topology optimization [52], or response surface methodology [53]. This work, in particular, employs curves of node shift evolution in time as characteristics, describing the crash; further, these curves are utilized as target variables to construct a supervised learning algorithm, predicting them directly from the node description (see 4.1). The simulations are held with the LS-DYNA [11] package. LS-DYNA is a nonlinear multi-physics finite element package for evaluating the significant deformation response of structures. The package design

allows a researcher to incorporate various physical details into the crash calculations, e.g., large deformation and rotation, contact with multiple bodies and self-contact, and nonlinear constitutive response. However, the precise crashworthiness evaluation with finite element methods is still challenging, requiring significant computational resources.

2.4 Machine learning

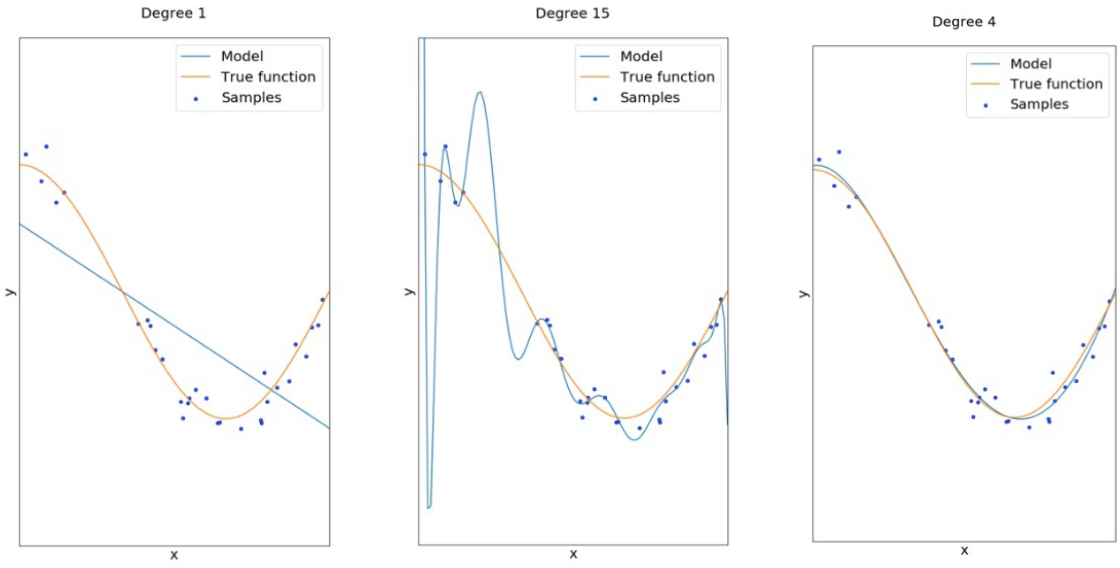
Machine learning is a field of applied mathematics at the intersection of artificial intelligence, mathematical statistics, and optimization, which studies methods for constructing algorithms that can learn from empirical (or precedent) data. In that context, the term ”learning” means discovering how a machine executor can perform tasks without being explicitly programmed to do so. Machine learning approaches are viable when finding the entirely determined sequence of steps to solve the problem is more challenging than helping the machine develop its algorithm. The traditional subfield division of the machine learning approaches is based on the type of "feedback" response available to the machine learning model during the training stage. The three major machine learning subfields are:

- Supervised learning: model is provided with pairs ”example input - desired output”, marked by the ”teacher”, and the goal of the process is to discover the mapping between inputs and outputs.
- Unsupervised learning is then the model has only training inputs, and its goal is to discover the underlying structure of a data. The goal is to either gain a more informative view of data or to discover a representation of data with desired properties (such as lower dimensionality or sparsity) [54]. Representations learned that way are often utilized to improve the performance of models for supervised tasks.
- Reinforcement learning: during the learning process, the model interacts with a dynamic environment, which rewards the model when it performs a specific action in the given state of the environment (such as braking in front of the red traffic lights while simulating the movement of the environment a vehicle on the highway). The objective of the model is to maximize obtained reward during the session of an environment [55]

A core objective of a machine learning model is to generalize from its experience [56]. The machine learning algorithm is considered to generalize well when it accurately performs on

unseen examples after having experienced learning data. The quality of the model is determined by its ability to simultaneously minimize the error on the training data and guarantee the insignificant difference between the training and testing error. From a statistical machine learning perspective, the task of a learner is to create a model of the generally unknown probability distribution over a finite set of samples that enables producing accurate predictions on the new samples [57].

On the way to achieving that goal, one usually faces two central challenges: underfitting and overfitting. Overfitting happens then machine learning model trained for too long on the training data or model is too complex for the problem. Any training set is a part of more extensive data distribution; overfitting means that model is adjusted to the details of the training set and its noise to the extent that leads to incapability to generalize to other examples. Underfitting happens when machine learning cannot detect dependencies in the training data, which can happen due to insufficient training time, poor choice of error function or model architecture. One way of balancing that is to monitor when error on validation data ceases to improve while still improving on training data. A simple example illustrating underfitting and overfitting for polynomial regression is given by the figure 2.6. It can be seen that first-degree



(a) (b) (c)

Figure 2.6: Concepts of (a) underfitting, (b) overfitting, (c) statistical fit in application to polynomial regression [58]

polynomial does not have enough statistical capacity to describe the distribution of points. In contrast, a polynomial of degree 15 deviates from the original curve while perfectly matching

training points. At the same time, the fourth-order polynomial approximates the original curve with high precision. The underfitting issue in many cases can be solved by simply increasing the statistical capacity of the model. Overfitting is a complicated problem usually addressed with such methods as cross-validation and regularization.

This work, in particular, solves regression tasks in the scope of the supervised learning approach, as we are aiming to explicitly predict the curves, describing the node shift evolution for the given finite element model of the thin-walled extrusion profile. There exists a multitude of machine learning algorithms capable of working in such paradigm; among them are linear models [59], decision trees [60], AdaBoost [61], and artificial neural networks [62] - superposition of multitude linear models with nonlinear activation functions. We utilize the latter, as neural networks have proved themselves to show good performance even for noisy and highly correlated data [63].

2.5 Supervised learning

The purpose of supervised machine learning algorithms is to discover a particular pattern in the given data via inferring function f, which maps a feature representation of a training sample X into a target variable y. This problem is incorrect from the mathematical point of view, as there are many possible solutions to it. For this reason, the standard approach to obtain a numerical solution to such a problem is to reduce it to the optimization of a chosen loss function (the choice of the latter largely depends on the type of the problem being solved and the target variable type). In addition, to evaluate the performance of a machine learning algorithm, one usually chooses specific metrics, estimating the quality of the solution. In some cases, algorithms can directly optimize the metric, but this statement does not generally hold, as the metric functions may have mathematical properties that are less suitable for optimization.

Supervised machine learning problems with real-valued target variables are called *regression problems*. Such a statement implicitly assumes that target variables lie in the continuous vector space with a defined metrics function. That being said, loss functions for regression tasks measure the distance between the predictions of the model $\hat{f}(x_i)$ on the sample x_i and the target. Probably the most popular loss function for regression problems is the mean squared

error (MSE) loss, given by the formula:

$$\mathrm{MSE}\left((\mathbf{X},\mathbf{Y})\right) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{f}(x_i) - y_i\right)^2 \tag{2.2}$$

Here $\mathcal{D} = (\mathbf{X},\mathbf{Y})$ is the training dataset, where $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ stands for set of feature descriptions of the data, and $\mathbf{Y} = \{y_1, y_2, \ldots, y_n\}$ is the set of target variables. Another widespread loss function is the mean average error (MAE), stated by:

$$\mathrm{MAE}\left((\mathbf{X},\mathbf{Y})\right) = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{f}(x_i) - y_i\right| \tag{2.3}$$

Let us suppose we want to approximate the function $f(x)$ using a regression algorithm, minimizing the expectation over MSE loss on the noisy data $\mathcal{D}$:

$$\mathcal{D} = \{(x_1,t_1),(x_2,t_2),\ldots(x_n,t_n)\}\, t_i = f(x_i) + \varepsilon, \quad \mathbb{E}[\varepsilon] = 0 \tag{2.4}$$

$$\mathbb{E}\left[MSE(\mathcal{D})\right] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[\left(\hat{f}(x_i) - t_i\right)^2\right] \tag{2.5}$$

For each term under the symbol of summation we perform the following transform:

$$\begin{gathered}
\mathbb{E}\left[\left(\hat{f}(x_i) - t_i\right)^2\right] = \mathbb{E}\left[\left(\hat{f}(x_i) - f(x_i) + f(x_i) - t_i\right)^2\right] = \\
\mathbb{E}\left[\left(\hat{f}(x_i) - f(x_i)\right)^2\right] + \mathbb{E}\left[(f(x_i) - t_i)^2\right] + 2\mathbb{E}\left[\left(\hat{f}(x_i) - f(x_i)\right)(f(x_i) - t_i)\right] = \\
\mathbb{E}\left[\varepsilon^2\right] + \mathbb{E}\left[\left(\hat{f}(x_i) - f(x_i)\right)^2\right] + 2\left(\mathbb{E}\left[\hat{f}(x_i)f(x_i)\right] - \mathbb{E}\left[f(x_i)^2\right] - \mathbb{E}\left[\hat{f}(x_i)t_i\right] + \mathbb{E}\left[f(x_i)t_i\right]\right) = \\
\mathbb{E}\left[\varepsilon^2\right] + \mathbb{E}\left[(f(x_i) - t_i)^2\right] + 2\left(\mathbb{E}\left[\hat{f}(x_i)f(x_i)\right] - \mathbb{E}\left[f^2(x_i)\right] - \mathbb{E}\left[\hat{f}(x_i)t_i\right] + \mathbb{E}\left[f(x_i)t_i\right]\right)
\end{gathered} \tag{2.6}$$

Further we have

$$\begin{gathered}
\mathbb{E}\left[\hat{f}(x_i)t\right] = \mathbb{E}\left[\hat{f}(x_i)(f(x_i) + \varepsilon)\right] = \mathbb{E}\left[\hat{f}(x_i)f(x_i)\right] \\
\mathbb{E}\left[f(x_i)t\right] = \mathbb{E}\left[f(x_i)(f(x_i) + \varepsilon)\right] = \mathbb{E}\left[f^2(x_i)\right]
\end{gathered} \tag{2.7}$$

and

$$\mathbb{E}\left[\left(\hat{f}(x_i) - t_i\right)^2\right] = \mathbb{E}\left[\varepsilon^2\right] + \mathbb{E}\left[\left(\hat{f}(x_i) - f(x_i)\right)^2\right] \tag{2.8}$$

Hence, MSE loss can be represented as the sum of the noise variance and the expectation of MSE between the model predictions and the true function. Now we perform the similar trick

with "smart zero", adding $\mathbb{E}\left[\hat{f}(x_i)\right]$ into the second term of (1).

$$\mathbb{E}\left[\left(\hat{f}(x_i) - f(x_i)\right)^2\right] = \mathbb{E}\left[\left(\hat{f}(x_i) - \mathbb{E}\left[\hat{f}(x_i)\right] + \mathbb{E}\left[\hat{f}(x_i)\right] - f(x_i)\right)^2\right] =$$
$$\mathbb{E}\left[\left(\hat{f}(x_i) - \mathbb{E}\left[\hat{f}(x_i)\right]\right)^2\right] + \mathbb{E}\left[\left(\mathbb{E}\left[\hat{f}(x_i)\right] - f(x_i)\right)^2\right] + 2\mathbb{E}\left[\left(\hat{f}(x_i) - \mathbb{E}\left[\hat{f}(x_i)\right]\right)\left(\mathbb{E}\left[\hat{f}(x_i)\right] - f(x_i)\right)\right] \tag{2.9}$$

The third term can be reduced the following way

$$2\mathbb{E}\left[\left(\hat{f}(x_i) - \mathbb{E}\left[\hat{f}(x_i)\right]\right)\left(\mathbb{E}\left[\hat{f}(x_i)\right] - f(x_i)\right)\right] = 2\mathbb{E}\left[\hat{f}(x_i)\mathbb{E}\left[\hat{f}(x_i)\right]\right] -$$
$$\mathbb{E}^2\left[\hat{f}(x_i)\right] - \mathbb{E}\left[\hat{f}(x_i)f(x_i)\right] + \mathbb{E}\left[f(x_i)\mathbb{E}\left[\hat{f}(x_i)\right]\right] = 0 \tag{2.10}$$

Finally, we get

$$\mathbb{E}\left[\left(\hat{f}(x_i) - t_i\right)^2\right] = \text{Bias}_i + \text{Variance}_i + \mathbb{E}\left[\varepsilon^2\right] \tag{2.11}$$

$$\text{Bias}_i = \mathbb{E}\left[\left(\mathbb{E}\left[\hat{f}(x_i)\right] - f(x_i)\right)^2\right] \tag{2.12}$$

$$\text{Variance}_i = \mathbb{E}\left[\left(\hat{f}(x_i) - \mathbb{E}\left[\hat{f}(x_i)\right]\right)^2\right] \tag{2.13}$$

The equality obtained above is called *bias-variance* decomposition. According to it, the expectation over the MSE loss for a regression model $\hat{f}(\cdot)$ consists of three terms:

- *Bias*, demonstrating whether the model approximates the original function $f(\cdot)$
- *Variance*, yielding the deviations from the mean prediction value
- *Noise*, which cannot be altered by algorithm due to the stochastic nature of the dataset

Thus, in order to minimize MSE loss, we need to minimize both variance and bias of the model, which is a non-trivial problem, as the majority of datasets exhibit a bias-variance tradeoff while varying the parameters of regression algorithms [64]. High bias corresponds to the underfitted model, as its prediction is far from the mapping being reconstructed. In contrast, high variance means that the model is overfitted, as the predictions are far from the expectation over them.

2.6 Artificial Neural Network

An artificial neural network (ANN) is a machine learning algorithm partially inspired by synapses in biological brains. The computation process in this algorithm can be presented

with a directed graph with so-called artificial neurons as its nodes. Neurons in the graph are connected in the manner shown in fig.2.7; to simplify the visualization, they are aggregated into layers of neurons. Artificial neurons produce a real-valued number as an output, passing the inputs through a non-linear transformation known as the "activation function". Each connection

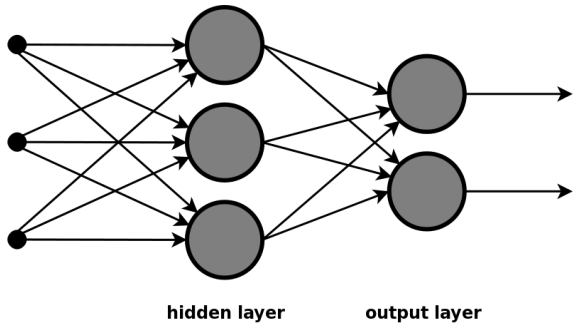


Figure 2.7: An example of feedforward ANN architecture with a single hidden layer

in the neural network has its associated weight. Weights, bound to connections between the two layers, can be naturally represented as a weight matrix $\mathbf{W}_i$. In addition, neurons are usually provided with their bias weights b_i. The following formula describes the calculation of the output of the layer i:

$$\overrightarrow{\text{output}_i} = \sigma\left(\mathbf{W}^i \cdot \overrightarrow{\text{input}_i} + \overrightarrow{b_i}\right) \tag{2.14}$$

Here $\sigma(\cdot)$ is the activation function of choice. In other words, a single layer of the network subjects its input to the superposition of the linear transformation and non-linear activation. ANNs are quintessential machine learning models that may reconstruct even discontinuous dependencies given sufficient data. This statement is a corollary of the universal approximation theorem, proved in the scope of [65]. Consider $I_N = [0,1]^N$ – an N-dimensional real-valued cube, and a parametric family of functions

$$G(Y) = \left\{ g : I_n \times I_n \to \mathbb{R};\ g(x,y) = \sum_{j=1}^{M} \alpha_j \sigma\left(y_j^T x + \theta_j\right) \right\}, \tag{2.15}$$

Here y_j is the j-th weight vector; θ_j, α_j - weight scalars, and $\sigma(x)$ - is a continuous function with following properties

$$\sigma(x) = \begin{cases} 1,\ x \to +\infty \\ 0,\ x \to -\infty \end{cases} \tag{2.16}$$

Then, for arbitrary continuous $f : \mathbb{C}(I_N) \to \mathbb{R}$ exists a two-layered ANN, approximating f with arbitrary precision. High statistical capacity and flexibility are why neural networks

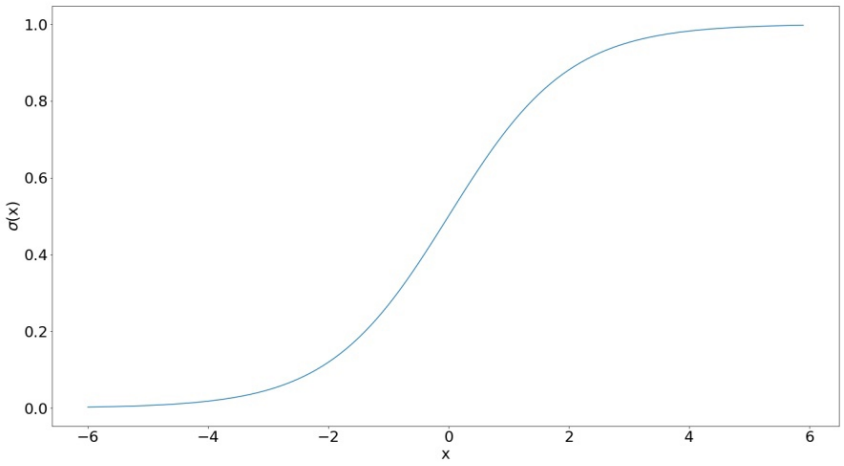


Figure 2.8: Sigmoidal activation function

have found their applications in a huge variety of tasks. One of the major upsides of neural network algorithms is their ability to work with complex, strongly correlated data with little to no preprocessing. Thus, they may be considered a default pick for tasks of natural language processing [66], computer vision [67, 68], and generative modelling [69, 70, 71].

2.6.1 Activation functions

As mentioned before, "activation functions" is the common name for a family of nonlinear functions applied to the outputs of neurons within neural network algorithms. Initially, neural network studies were using functions that satisfy the conditions of the approximation theorem [65], such as sigmoidal unit or hyperbolic tangent. However, later studies showed that this condition is unnecessary for practical applications of neural networks to converge up to good solutions. A classical logistic sigmoid (fig.2.8) is given by the formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.17}$$

Sigmoidal function is a strictly increasing function with two horizontal asymptotes: $y = 1$ at $x = +\infty$, and $y = 0$ at $x = -\infty$. A derivative of the sigmoid unit can be rewritten as:

$$\sigma'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = \sigma(x)(1 - \sigma(x)) \tag{2.18}$$

Therefore, if the value of x is close to the region of sigmoid saturation, the gradient is almost zero-valued. These saturation properties are causing the central problem of sigmoid units - gradient fading. Another problem of training ANNs with sigmoid activations arises from the fact that this function is not zero-centred, leading to oscillations in gradient updates [72].

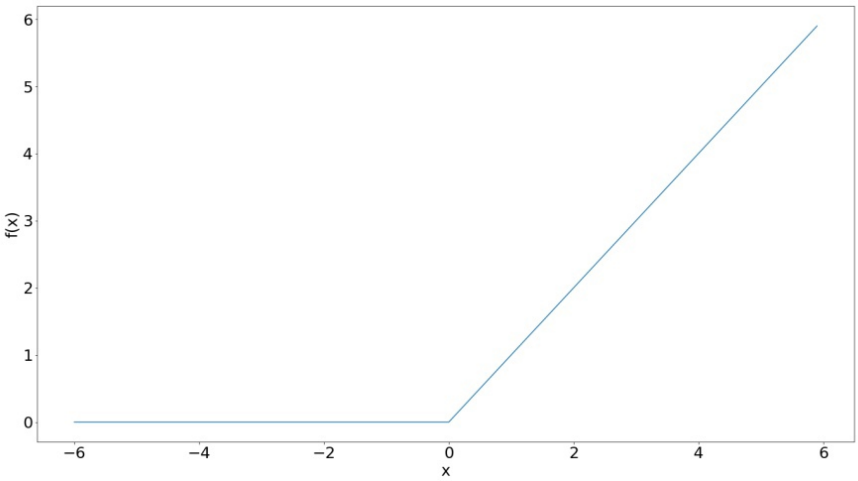


Figure 2.9: ReLU activation function

This work in particular employs a *rectified linear unit* (ReLU), suggested by Nair et al. in [73], as the activation function for the hidden layers of neural network. ReLU is described by the formula (see fig.2.9):

$$f(x) = \max(0, x) \tag{2.19}$$

Because rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimize with gradient-based methods [74], and eliminate the vanishing gradient problem observed for sigmoidal functions. Another advantage of rectified units is their faster computation time, as it does not require computing exponents. However, neural networks with ReLU activations tend to be more prone to overfitting and require additional regularization techniques, which will be discussed further.

2.6.2 Training algorithm

The standard approach to train neural networks with gradient-based optimization is to abuse the chain rule for the computation of derivatives with the so-called backpropagation algorithm [75]. Suppose w_{ij}^k are the weights of the kth layer of the neural network. The execution of the backpropagation algorithm begins from the forward pass, within which the input vector x is fed into the network and propagated through it. During the forward pass, the network stores activations a_j^k (matrix product + bias) for each neuron of each layer during the computations. As a result, we get the value of a loss function E on the vector x. To train the network, we are

required to calculate the following derivative:

$$\frac{\partial E}{\partial w_{ij}^k} \tag{2.20}$$

By using the chain rule w.r.t the activations a_j^k, we obtain

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k}\frac{\partial a_j^k}{\partial w_{ij}^k} \tag{2.21}$$

The second multiplier can be simplified as follows:

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k}\left(\sum_{i=1}^{r_{k-1}} w_{lj} o_j^{k-1}\right) = o_i^{k-1} \tag{2.22}$$

Here r_{k-1} is the number of neurons for previous layer, $o_j^{k-1} = \sigma(a_j^{k-1})$ - sigmoidal function applied to activations from previous layer. Thus, we get

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k} o_i^{k-1} \tag{2.23}$$

To calculate $\frac{\partial E}{\partial a_j^k}$ for hidden layers, we utilize the chain rule

$$\frac{\partial E}{\partial a_j^k} = \sum_{l=1}^{r_{k+1}} \frac{\partial E}{\partial a_l^{k+1}}\frac{\partial a_l^{k+1}}{\partial a_j^k} \tag{2.24}$$

By definition,

$$a_l^{k+1} = \sum_{j=1}^{r_{k+1}} w_{lj}^{k+1}\sigma(a_j^k) \tag{2.25}$$

$$\frac{\partial a_l^{k+1}}{\partial a_j^k} = w_{lj}\sigma'(a_j^k) \tag{2.26}$$

Finally, the rule for gradient computation is given by

$$\frac{\partial E}{\partial w_{ij}^k} = o_i^{k-1} \cdot \sum_{l=1}^{r_{k+1}} \frac{\partial E}{\partial a_l^{k+1}} w_{lj}\sigma'(a_j^k) \tag{2.27}$$

The rule for computation of $\frac{\partial E}{\partial a_j^k}$ is recursive: to calculate gradient at layer $k-1$, it is required to calculate gradient at layer k. To implement the backpropagation algorithm for neural networks, most popular deep learning programming frameworks such as Tensorflow [76] by Google or PyTorch [77] by Facebook store information about the order of operations conducted over the neural network input as a directed computational graph. Backpropagation utilizes automatic differentiation packages.

2.6.3 Optimization

Since the backpropagation algorithm allows us to compute the gradient of loss function w.r.t the parameters of ANN, one commonly employs a gradient-based optimization scheme for ANN training. The correct choice of the optimization scheme dramatically depends on the properties of the loss function. As neural networks with ReLU activations are non-convex and almost everywhere differentiable, the use of gradient descent (GD) is optimal [78] in the sense of asymptotic upper bound estimates. For the sake of computational efficiency, its stochastic modification (SGD), which performs gradient steps on the mini-batches, is usually implemented. However, optimization via SGD often requires precise hyperparameters, such as learning rate, as they significantly impact the optimization process. In practice, people often utilize various modifications of SGD, capable of performing at the same level of quality but more robust in the sense of hyperparameter tuning. This work, in particular, employs the Adam (derived from *adaptive moment estimation*) [79] algorithm, which evaluates the learning rate for each weight individually per the evaluation of first- and second-order moments of the gradient. The following set of equations gives parameter updates for the Adam optimizer:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.28}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.29}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.30}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.31}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.32}$$

Here m_t is the estimate of the mean of the gradient; v_t is the estimate of its uncentered variance; β_1 and β_2 are the rates of exponential decay for the m_t, v_t respectively; α is the learning rate hyperparameter and ϵ is the small constant for numerical stability.

2.6.4 Regularization

ANNs with multiple hidden layers are capable of learning very complicated relationships between inputs and outputs. However, while being trained on small amounts of data, neural networks may consider these complicated relationships as a sampling noise within the training dataset, non-existent in the test data, even if the test samples were drawn from the same

distribution. This problem eventually leads to overfitting, and in order for neural networks to learn the actual structure of the data, they need to be properly regularized. The term regularization in this context means any a priori assumptions about the properties of the function we are reconstructing, thus constraining the set of possible solutions to the given problem [80]. Let us suppose that $\mathcal{D} = \left\{(x_i, y_i);\ i \in \overline{0, N}\right\}$ is the training dataset, and we are trying to reconstruct the mapping f via minimizing the MSE loss. Then, one of the possible ways to regularize the problem is to minimize the loss with the addition of the *regularizing term* $G(f)$:

$$L(\mathcal{D}) = \sum_{i=0}^{N} \left(f(x_i) - y_i\right)^2 + \lambda G(f) \tag{2.33}$$

Here, λ is the hyperparameter, limiting the scale of regularization. Commonly used regularizations include either limiting the norm of the model weights (ridge [81] and LASSO [82]), or limiting the smoothness of the function f (Duchon multidimensional splines [83], gaussian stabilizer [84]). Another popular technique for neural network regularization is *dropout*, which assumes temporal removal of some neurons from the computational graph of the network, along with all its incoming and outgoing connections [85]. The choice of which units to drop is random and is usually specified as a hyperparameter for a given group of neurons. Application of dropout to the neural network essentially means sampling a smaller network from it, consisting only of units that remained untouched during the procedure.

2.7 Recurrent Neural Networks

A recurrent Neural Network (RNN) is an architectural pattern capable of handling sequential data [75]. Neural networks based on this architecture have found application in many subject areas - language modelling [86], speech recognition [87], audio generation [88] and etc. The whole concept behind the vanilla RNNs comes from the equation, describing the state $s(t)$ of a parameterized dynamical system evolving in time.

$$s^{(t)} = f(s^{(t-1)}; \theta) \tag{2.34}$$

Here θ is the vector of system parameters. Let us suppose that we are working with a discrete-time system, evolving during T timesteps in total. In this case, such equation describes a computational graph, which can be unfolded by applying the definition T times:

$$s^{(T)} = f(s^{(T-1)}; \theta) = f(f(s^{(T-2)}; \theta); \theta) = \dots \tag{2.35}$$

A classical RNN is defined by an equation of a dynamical system, describing the evolution of its state $h^{(t)}$ driven by an external time-dependent signal $x^{(t)}$ (see fig. 2.10):

$$h^{(t)} = f(h^{(t-1)}; x^{(t)}; \theta) \tag{2.36}$$

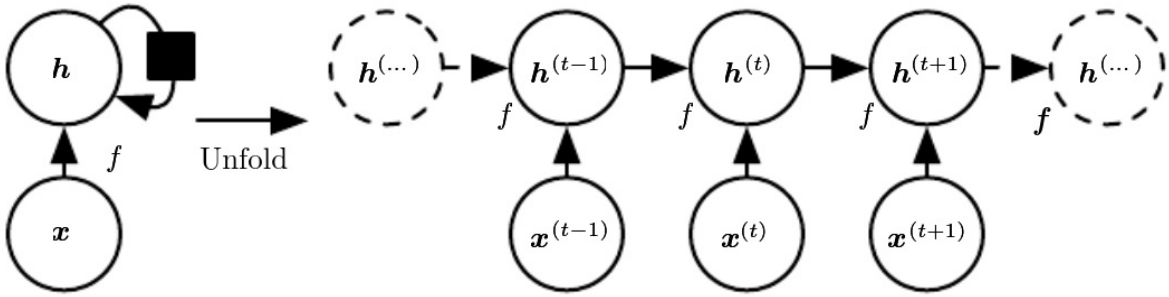


Figure 2.10: RNN schematic - circut diagram and unfolded graph [74].

RNNs are usually trained to predict the next sequence element, given the sequence as a whole. In that case the hidden state of neural network may be considered as a lossy summary [74] of a sequence, mapping the whole sequence $\mathbf{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(T)})$ into a single vector of a constant length $h^{(t)}$. A computational graph for calculating the loss of a simple neural network is yielded by the figure 2.11. The following set of update equations describes forward propagation through this graph for a problem with discrete target variables:

$$\begin{aligned} a^{(t)} &= b + Wh^{(t-1)} + Ux^{(t)} \\ h^{(t)} &= \tanh(a^{(t)}) \\ o^{(t)} &= c + Vh^{(t)} \\ \hat{y}^{(t)} &= \text{softmax}(o^{(t)}) \end{aligned} \tag{2.37}$$

Here, the network parameters are weight matrices W, U, V and the corresponding biases b, c. The total loss for a given sequence $\mathbf{x}$ will be a sum for losses for each element of the sequence. It should be noted that the complexity for both the forward and the backward pass of the algorithm by the length of the sequence is $\mathrm{O}(T)$, and the computation of this graph principally cannot be parallelized.

RNNs may be trained with backpropagation-through-time algorithm (BPTT) [89], which is essentially an application of classical backpropagation algorithm to the unfolded graph of the RNN. Training neural networks with BPTT may be difficult due to exploding or fading

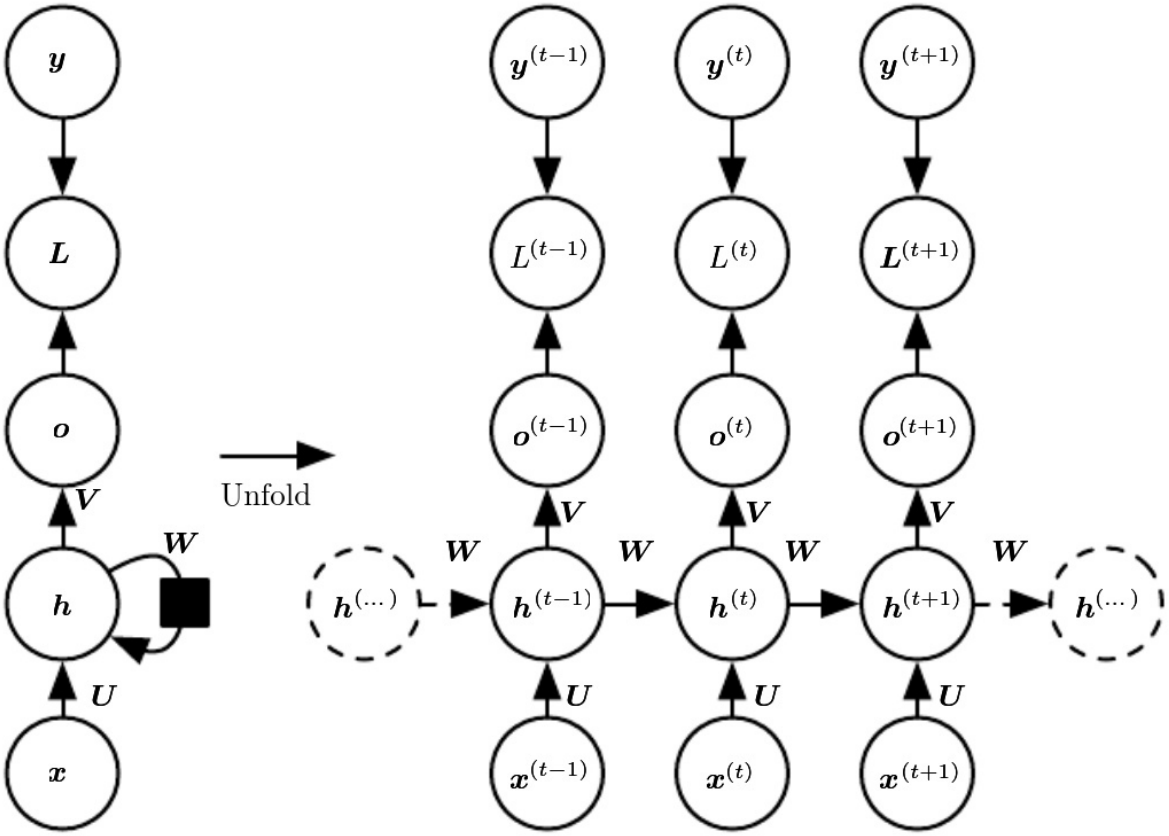


Figure 2.11: The computational graph to calculate the loss function for RNN training [74].

gradients [90]; common approaches to overcome these difficulties is to either clip the norm of the gradient [91] or to employ the truncated version of BPTT [92]. In the work [93], authors proposed a way to regularize RNN with the dropout technique mentioned above.

2.8 LSTM

In comparison to vanilla RNN, data flow through long short term memory (LSTM) [19] neural networks is controlled by two 'gates' - *input gate* i^t and *forget gate* f^t. This gate aims to determine how much information about the current element of the sequence can be used to update the hidden states. Moreover, LSTM utilizes a second hidden state - a memory cell vector c^t. These modifications allow LSTM to more efficiently capture long-term dependencies in the sequence-like data [94]. In the runtime, hidden state and cell vectors are updated according to

the following set of equations.

$$
\begin{aligned}
i^t &= \sigma\left(W^i x^t + V^i h^{t-1} + b^i\right) \\
f^t &= \sigma\left(W^f x^t + V^f h^{t-1} + b^f\right) \\
o^t &= \sigma\left(W^o x^t + V^o h^{t-1} + b^o\right) \\
\tilde{C}_t &= \tanh\left(W^{\tilde{C}} x^t + V^{\tilde{C}} h^{t-1} + b^{\tilde{C}}\right) \\
c^t &= f^t \odot c^{t-1} + i^t \odot \tilde{C}^{t-1} \\
h^t &= o^t \odot \tanh\left(c^t\right)
\end{aligned}
\tag{2.38}
$$

Here W^i, W^f, W^o, W^C and V^i, V^f, V^o, V^C are weight matrices, corresponding to sequence elements and hidden states respectively, b^i, b^f, b^o, b^C are bias weights, and $\odot$ is the elementwise product. A flowchart illustrating these update rules is provided by the figure 2.12.

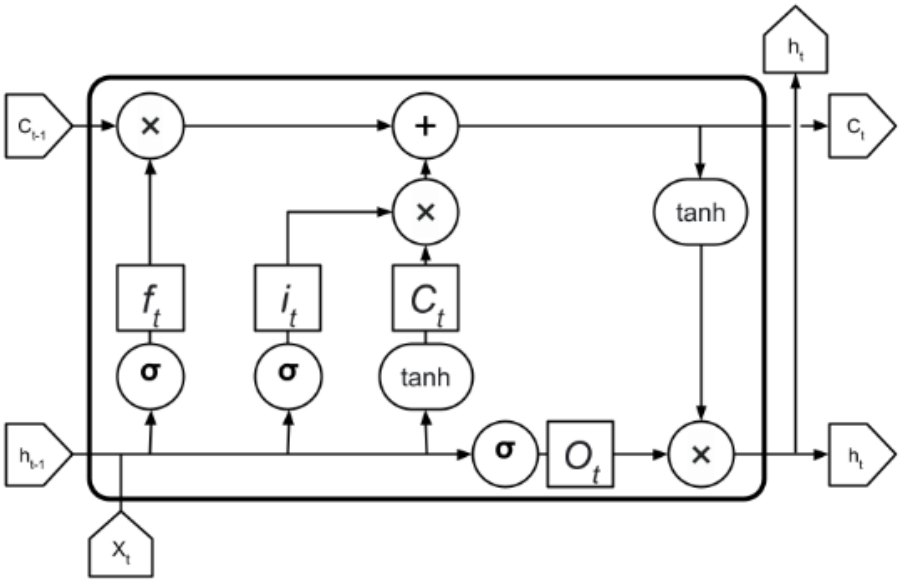


Figure 2.12: A flowchart of the LSTM neural network architecture.

2.9 Autoencoders

The concept of autoencoders emerged as a tool for unsupervised data dimensionality reduction. It was introduced in 1987 by Yann LeCun in his Master Of Science thesis [95]. Autoencoders are usually trained to reconstruct their input by solving the optimization problem of minimizing the MSE loss between input and output tensors of the neural network. An autoencoder-like network architecture consists of 2 parts (see fig. 2.13) - encoder f mapping input data x to the inner feature representation z, and decoder g, which maps z into the reconstruction of input r.

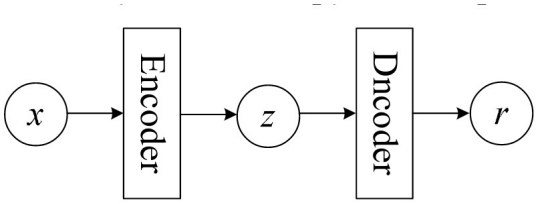


Figure 2.13: Flowchart of autoencoder architecture.

Of practical interest are those data representations z that have some particular properties, e.g., z having lower dimensionality than x or being sparse. To obtain this, one should somehow restrict the autoencoder. That being said, the problem of autoencoder training may be rewritten as a following set of formulas:

$$\begin{aligned} L(W_f, W_g) &= \frac{1}{n}\sum_{i=1}^{n} ||x_i - r_i||^2 \to \min \\ \text{s.t} \quad r_i &= g(z_i, W_g) \\ z_i &= f(x_i, W_f) \\ c_k(z_i) &= 0, \quad k = \overline{1, M} \end{aligned} \tag{2.39}$$

Here W_f, W_g - parameters of encoder and decoder, $c_k(z_i) = 0$ - constraints, providing the structure of representation z (for example, to build a compression autoencoder, we may simply require z to have lesser dimensionality than x). Once trained, the representation z_i may be used as a feature description of a sample x_i. Traditional usage of autoencoders includes such unsupervised machine learning problems as data compression [74] and feature extraction [96].

2.10 Model Ensembling and Bagging

An *ensemble* is a finite set of machine learning models, which combines predictive results gained from models individually and fuses them with various voting mechanisms in order to enhance the performance of any constituent model. During the past decade, ensemble learning was one of the most important centers of attention of the machine learning community. Currently, the field is supplemented with a variety of conducted research and plenty of examples of its successful applications in diverse engineering tasks and ML competitions.

Ensemble learning aims to integrate various machine learning models into a unified framework so that the complementary information of its parts is utilized to get better final performance. The whole working pipeline of an ensemble may be decomposed into two consecutive

steps (see fig. 2.14):

1. Obtaining prediction from weak models, constituting the ensemble.
2. Combining them with a voting scheme to get the final prediction.

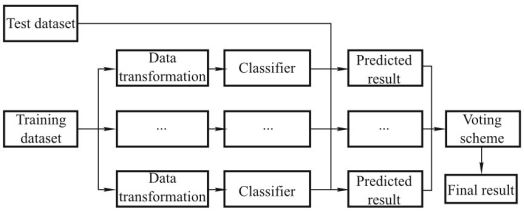


Figure 2.14: The framework for ensemble models [97].

Combining different models may be helpful only when they produce different outputs, as the composition of identical algorithms obviously cannot benefit from each other. There are many ways to obtain the output of ensemble given the prediction of its participants; a comprehensive review on the common ensembling approaches can be found in [98]. In the same work, the author states that there are at least three reasons why model ensembling may yield better results in comparison to a single model:

- In application to the classification problems, the ensemble error can be divided into two terms (see 2.5): *bias*, describing the average generalization error of each classifier in the ensemble, and *variance*, describing the disagreement among the classifiers. Let us suppose that we have managed to separate our training dataset $\mathcal{D}$ into m independent subsets and train m different models f_i to predict the target variable y given the sample x. If we combine these models into the ensemble, which returns the average prediction of constituent models,

$$\hat{f}(x) = \frac{1}{m}\sum_{i=1}^{M} f_i(x) \tag{2.40}$$

  then

  1. The noise component of the error will remain unchanged.
  2. For bias we have the following upper bound:

$$\mathbb{E}\left[\hat{f}(x)\right] = \frac{1}{m}\sum_{i=1}^{M}\mathbb{E}\left[f_i(x)\right] \leqslant \max_i \mathbb{E}\left[f_i(x)\right] \tag{2.41}$$

3. For variance we have the following upper bound:

$$\mathbb{D}\left[\hat{f}(x)\right] = \frac{1}{m^2}\sum_{i=1}^{M}\mathbb{D}\left[f_i(x)\right] \leqslant \frac{1}{m}\max_i \mathbb{D}\left[f_i(x)\right] \tag{2.42}$$

Hence, averaging outputs of the ensemble may reduce the total variance of predictions while maintaining the same bias, affecting overall performance positively.

- Many training algorithms for machine learning only guarantee convergence to the local optima of the loss function. Such a problem, for example, is featured by greedy splitting algorithms, used for training decision trees [99] and first-order stochastic gradient methods [78]. An ensemble constructed by running the local search from many different starting points may provide a better approximation to the actual unknown function than any of the individual classifiers [98].

- Representational: there might not be a true hypothesis in the hypothesis space. By combining several models from the hypothesis space, the true hypothesis may get a better approximation. For example, a non-linear curve may be approximated with piecewise linear functions.

Most popular ensembling methods revolve around alternating the training process, hoping different algorithms will provide different results. Among the various approaches of diversifying the outputs of the participants of the ensemble, *bagging* [100] stands out due to the ease of its implementation and applicability to almost every type of machine learning algorithm. This ensembling approach relies on training new members of the ensemble on random subsets of a training dataset. The latter are usually constructed with a procedure called "bootstrap sampling" - drawing samples uniformly from the training set with replacements. As a consequence, new subsets may contain duplicated samples, and the share of unique samples in them is $\left(1-\frac{1}{e}\right) \simeq 63\%$. Both sample drawing and training of such ensemble may run in parallel, allowing composing ensembles from many algorithms quickly. Work [101] proves that bagging algorithms reach good performance in the case when learning algorithms are "unstable" in the sense that small dataset changes greatly impact the algorithm structure. Examples of such algorithms are neural networks or decision trees; in particular, a popular *random forest* algorithm may be described as bagging of decision trees over random subspaces [102].

2.11 Sampling

In most cases, the procedure of fitting a supervised machine learning algorithm is the bottleneck of their computational efficiency. Modern machine learning applications often operate on significant amounts of data; thus, the brute-force exploration of the dataset as a whole is too computationally expensive [103]. This problem is especially relevant for neural networks since they are usually intended to process complex high-dimensional data. A possible approach to overcome this problem is to shrink or condense the training dataset, exploiting the tradeoff between shorter training time and the diminishing quality of the solution due to learning from only a part of the data. Plenty of the known strategies for dataset shrinkage rely on hybrid approaches, combining data clusterization with heuristic methods. However, the task of metric-based clustering in high-dimensional data spaces with mixed discrete/real-valued data is prone to failure due to increasing correlation between the samples and the curse of dimensionality [104].

Another well-established strategy to reduce the computational cost for training is to perform gradient descent on small chunks of data instead. This approach, also known as mini-batch training, remains state-of-the-art in almost all current machine learning applications, provided with considerable evidence of its effectiveness in practice and theoretical justification. However, mini-batching in stochastic optimization inevitably leads to the increased gradient variance, which eventually halts the progress of the gradient methods, sometimes rendering it unable to push the target loss value to the desired values.

While training complex models, it often appears to the practitioner that not all training set elements are equally important. A significant amount of them might already be adequately handled after a couple of epochs. Thus, they may be removed from the training dataset to reduce the computational cost of training without declining final quality. The process of picking examples from a fixed set is called *sampling*. The word "sampling" may be formalized in the following way. Consider the finite set:

$$S = \{a_1, a_2, \ldots, a_n\} \tag{2.43}$$

We will call sampling a set-valued stochastic mapping

$$\hat{S} : S \to 2^S \tag{2.44}$$

which maps the set S into its subset with a given probability. A sampling is uniquely charac-

terized with its *probability mass function*

$$\mathbf{P}(A) = P(\hat{S} = A) \tag{2.45}$$

assigning probabilities to all elements $A \in 2^S$. E.g., we may consider a uniform sampling procedure:

$$p\left(\hat{S}(S) = \{a_i\}\right) = \frac{1}{n} \tag{2.46}$$

In the scope of this work, numerical experiments were conducted with two different sampling techniques - *random sampling*, and *importance sampling*. The term "random sampling" here means uniform sampling over the set without repetitions; the idea behind importance sampling will be clarified in the following subsection.

2.12 Importance Sampling

Consider a distribution $\mathcal{D}$, and a weight function $w(i)$, assigning non-negative weight, or *importance* to each number i. The weighted distribution $D^{(w)}$ is defined by

$$\mathbb{P}_{\mathcal{D}^{(w)}}(I) \propto \mathbb{E}_{i\sim\mathcal{D}}\left[\mathbf{1}_I(i)w(i)\right] \tag{2.47}$$

where I is the subset of indices, and $\mathbf{1}_I(\cdot)$ is the indicator function. I.e., in the discrete case, this is equivalent to following transformation of probability mass function

$$p^{(w)}(i) \propto p(i)w(i) \tag{2.48}$$

whereas in continuous case this corresponds to multiplying density function by $w(i)$ and renormalizing. In practice $D^{(w)}$ may be constructed through procedure, known as *rejection sampling*: sample $i \sim \mathcal{D}$, and accept it with probability $\dfrac{w(i)}{W}$, where $W \geqslant \sup\limits_i w_i$.

The classical field of application of importance sampling is Monte-Carlo methods [105]: it often allows to reduce the variance of estimated integral. In addition, there exists a set of works devoted to applications of importance sampling in statistical ML. For instance, authors of [106] show that correct design of weights may improve the current upper bound estimate of the number of steps required for SGD to converge to the minimum of strongly convex function in terms of dependence on its average conditioning number from sublinear [107] to linear. Even though for some optimization problems there are analytical methods for constructing optimal sampling, supported with convergence rate estimates [108, 109], many works in the field of deep learning focus on sampling procedures, built heuristically [110, 111, 112], employing either loss values or gradient norm to choose the most relevant sample.

2.13 Machine Learning in numerical modeling

At the moment ML approaches are at the prominent place in different fields of scientific research, previously dominated by first-principle models - geosciences [113], astronomy [114], environmental sciences [115], fluid dynamics [116] and many others. The use of ML models attracts the community due to their ability to discover relations within data, describing poorly understood processes, or processes, which are difficult to model due to the impractical space-time resolution required to get a satisfactory numerical solution. Such models seem especially attractive when applied to real-world engineering problems. The forward-pass time of even the most complicated neural network is generally much less than the time required to conduct a numerical simulation of the same process. For example, high-accurate simulations of non-linear processes with finite elements are still considered computationally expensive procedures, despite the proliferation of computers in modernity. For that reason, the scientific community suggested direct substitution of virtual experiments, which aim to measure some characteristics of the real-world behaviour of solid media, with the ML model directly predicting the same characteristics from the "description" of the media. A natural way to obtain datasets in such work is to fix all the parameters of a numerical experiment, except for the description of the environment, and run the simulation through environments obtained by varying these descriptions, recording the required characteristics as target variables. A pleiad of similar works can be found in the field of biomedical research with finite element simulations. For example, Martínez-Martínez et al. [12] were predicting the biomechanical behaviour of the breast tissues in image-guided interventions such as biopsies or radiotherapy with random forests [102] and extremely randomized trees [117]; datasets for these experiments were obtained with FE simulations. Work [13] is devoted to predicting the stress distribution over the aortic based on the parameterized description of its geometry with the neural network, using the dataset of 729 thoracic aorta shapes and corresponding wall stress distributions, constructed in the scope of work [118]. Another research field that employs the same concept of using numerical methods as a dataset creation tool for ML algorithms is the ML-based solution of inverse problems. For instance, Yang and Ma [14] were solving seismic inversion problems, discovering the structure of artificially generated velocity distribution over the media from seismograms, obtained with modelling of the media with a finite difference scheme.

Unfortunately, the application of novel ML models as black boxes to real-world data has limited success in scientific domains. The data volume requirements for learning from com-

plex physical processes are not satisfied in the majority of cases; moreover, such models often demonstrate poor generalization for out-of-sample scenarios [119]. Thus, recently the research community has started to explore the verge between mechanistic and ML models, integrating both domains in various ways [120, 121]. At the moment, the taxonomy of different methodologies to merge both principles include five classes [122]:

- Physics-guided loss functions
- Physics-guided initialization
- Physics-guided design of architecture
- Residual modeling
- Hybrid physics-ML models

Hybrid physics-ML models assume simultaneous operation of both the numerical method and the ML model. The natural way to combine them is to use outputs of the physics-based model either as the additional input to the ML model. E.g., Karpatne et al. [123] demonstrated that adding the output of a physics-based model into the set of features describing the training data may enhance the quality of predictions of lake temperature.

2.14 Machine learning in crashworthiness

The problem of discovering the optimal design of vehicle parts in some cases can be reformulated as a task of optimization with constraints [124]. One of the possible ways to solve such a problem is to employ the response surface methodology [125], which is ideally suited for solving problems with noisy responses, where gradient-based algorithms would end up in a nearby local optimum. In application to crashworthiness, this approach was implemented, for example, by Liu, Detwiler, and Tovar [126] for mechanical compliance problems under the static load. Another work [127] uses Response surface methodology (RSM) to optimize the cylindrical tube impacting a rigid wall with the initial velocity of 10 m/s.

Response surface methodology within the design of experiment (DOE) approach requires a regression metamodel to relate the crashing and energy absorption responses to various design variables for analysis and optimization. Machine learning algorithms solving regression tasks

can act as such metamodels [128]; one can discover the use of radial basis functions (RBF) [129], support vector regression [130] and ANNs [131, 132, 133] as proper metamodels for RSM. Unsupervised learning paradigms are also applicable to RSM. For instance, Liu et al. suggested unsupervised design parameter clustering to reduce the time required to research the parameter space with RSM [134]. The research [135] applies the same intuition directly to crashworthiness optimization of the beam-like structure. In the work [136], authored by Acar and Solanki, metamodel ensembling was proposed to improve the quality of solutions obtained with RSM.

Another possible approach to ML-based crashworthiness optimization lies within the field of topological optimization. For example, in the work [137], authored by Liu et al., ML algorithms are used metamodelling in order to simplify the direct formulation of the problem for discovering an optimal design for thin wallet tubes; for that purpose, they used RBF algorithms, and Kriging [138]. Acar, Altin, and Güler [139] used gaussian joint probability models [140] to investigate the optimal design of the multi-cell profile for cylindrical aluminium tubes; work [141] for the same purposes employs least-square support vector regression [142].

A relatively new concept of applying machine learning algorithms to the crashworthiness simulations is suggested by Kohar et al. in [6]. In that work, the accent is shifted towards replacing FE modelling entirely through training the AI-based framework in a supervised way to predict the time-series response of the occupant crash-pulse. The value of such an algorithm lies in the fact that the time required to predict the new set of parameters is inferior compared to FE modelling; thus, design parameter space with the assistance of such a framework may be explored with much greater speed.

2.15 Deficiency in literature

As the verge of numerical modelling and machine learning is becoming the center of attention of the scientific community, many recent works are dedicated to replacing the finite element solvers with neural networks. Examples of such works exist in application to the numerical simulation of different physical processes, and media [143] - biological tissues [144], chemical kinetics [145], Hamiltonian dynamics [146]. However, the critical point of these researches is mostly the 'proof of the concept' style, elaborating only on the possibility to substitute numerical solvers with AI algorithms but omitting the investigation over the training data on

the subject of redundancy and representativeness.

Next, existing literature devoted to machine-learning approaches in crashworthiness is focused on discovering the optimal design of engineering systems and topology optimization ([126, 127, 136, 134, 135]). Most of these works use machine learning algorithms as metamodels for solving the regression tasks within the scope of the response surface methodology (RSM). RSM usually employs relatively simple algorithms to get coverage of design space parameters in a reasonable time. Due to the difficulties of training complicated ML models on big datasets, there were no attempts to use a sophisticated neural network as part of either RSM or topology optimization pipeline. The closest current work in terms of neural network usage is authored by Kohar et al. [6] and is devoted to the improvement of the iterative design for lightweighting in the automotive industries. The method proposed in that work suggests training the LSTM-based neural network to predict the crash-pulse response. However, research by [6] is unique in the field, and it still suffers from the same issues as [144, 145, 146]. – no attempts were made to improve neither the convergence time of the neural network nor to compare the resulting architecture with other algorithms. Together, these deficiencies demonstrate the actual need to investigate the possibility of replacing numerical simulations in crash tests with neural networks, especially from the point of view of improving the computational efficiency of similar solutions.

3 Scope and research objectives

3.1 Problem Statement and Objectives

The principal goal of this work is to design an efficient machine learning solution to the problem of predicting the evolution of shift of the nodes, belonging to the LS-DYNA model of the thin-walled aluminum extrusion profiles that were studied with axial crash simulations by Kohar et al. [16, 17, 18], during the process of designing UWR4 extruded profile. Since the dataset under investigation exhibits a pronounced sequence-like structure, the principal hypothesis of the research is that a neural network architecture specially designed for working with sequences would discover a solution of desirable quality much faster than LS-Dyna counterpart, which was taking 23 minutes 54 seconds per one simulation on system with 4 processors. Additionally, the research aims to test a particular set of assumptions about the general structure of data obtained during such numerical simulation procedures. Second hypothesis was that using a fraction of original train data randomly sampled may significantly reduce the required time to train the AI framework while saving accuracy. Another hypothesis is that original data contain redundancies, so it is possible to use training set data to figure an effective way to sample more information from the same or smaller fraction of original training data. The last hypothesis is that training a few smaller networks on different original data samples and combining them into ensembles inside the framework can increase the accuracy while still having lesser training time than the baseline AI framework. In order to prove these statements, the study will estimate the benefits of modifying the training procedure of the neural network model with ensembling techniques and sampling procedures from the points of view of computational effectiveness and the resulting performance on the test subset.

The main objectives of this research are:

- Provide faster solution to the problem of predicting the shift of the node of LS-DYNA model of the thin-walled aluminum extrusion profile during the axial crash of the profile using artificial neural networks framework
- Explore ways to increase the computational time and memory effectiveness of the suggested framework training process using data sampling:
  - 2 approaches of random sampling with a roulette algorithm

  - Heatmap-based sampling, built on the information from crash tests in the train data set

- Explore ensembling approach to the framework trained on samples and its impact on the accuracy of predictions

3.2 Limitations of the current work

The following set of assumptions is limiters of the scope of the current study:

- We have a dataset consisting of pairs (x_i, y_i), where x_i is the feature description of the node, and y_i is the evolution of shift of that node over time, obtained with numerical simulation of axial crash experiment. The size of the dataset is fixed, and it is principally prohibited to somehow expand the dataset by constructing another training sample. Such assumption sets the upper bound for the amount of information available, thus, allowing us to compare different sampling approaches.

- The trained ML framework provides the numerical solution to the problem under consideration if it reaches the local minima of an MSE loss function over its predictions and target variables. We assume that the lower value of loss function on the test subset means the better overall quality of numerical solutions, as there is no other comprehensible way to assess the accuracy of obtained predictions.

- All weights of all the models described in this work may be tuned with gradient-based optimization.

- The training process may be alternated by composing various subsets of a given dataset.

To summarize, it is assumed that we are allowed to alter the training process of a model only with the tweaks of data and techniques like sampling and bagging. Similar problem limitations may be found in the field of large-scale optimization [108, 109], where one usually focuses on designing a sampling strategy to obtain a better asymptotic estimate of convergence speed. However, in this work, we are prioritizing training time of the network as the main criterion of the efficiency of sampling strategy, as the background of the original framework originates from purely engineering problems, requiring rapid development to aid the workflow of car design.

4 AI framework for prediction of deformation shifts in crash experiments

This section describes the AI framework trained to predict node shift in time in the model of thin-walled UWR4-like [18] aluminum extrusion profile. The framework shares a similar design to that described in [6], employing the LSTM architecture to capitalize on the temporal structure of output data. The framework is trained on preprocessed data from FE simulations of axial crash experiments conducted by LS-DYNA package [11]; this section provides a detailed description of the dataset and feature representation of the given node. The section also includes information about two ML models, comprising the framework - deep autoencoder, used to compress the representation of 3D LS-DYNA model, obtained by preprocessing, into the low-dimensional vector, and the LSTM-based ANN, mapping node features into the target variable. Summary of the architecture of these models and their convergence plots are also presented in the section. Finally, the section provides a reference table with the training and test subsets' resulting metrics.

4.1 Training dataset

Data used in this work is obtained from numerical simulations of the axial crash of thin-walled aluminum rails with different profiles. A named example of such rail with good energy absorption properties, known as UWR4 extrusion profile, was discovered by Kohar et al. [15] (see fig. 4.1): Variety in profiles is provided by low-dimensional parameterization of their cross-

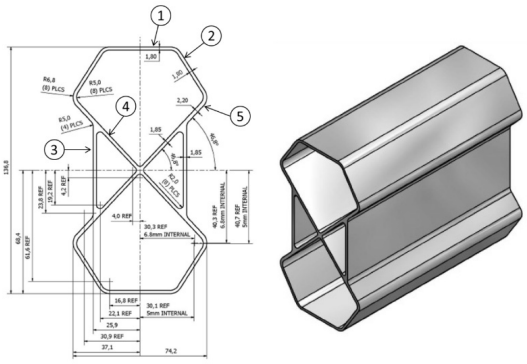


Figure 4.1: UW-R4 profile - cross-section and isometric view [15].

section. The upper-left part of the cross-section is described using six geometric points, and the complete rail profile is obtained with its quarter-symmetrization.

Numerical simulations with LS-DYNA assume having a separate model for each unique profile. These models consist of a finite amount of nodes connected with edges. Current work assumes using information about such nodes to predict their behaviour during the axial crash of the profile. Precisely, the used dataset consists of pairs 'feature representation of node - the evolution of its shift in time'. A canonical machine learning approach to tackle such complex data represents many features, each corresponding to a particular bit of information, complemented with a thorough analysis of chosen features. However, deep learning approaches often allow us to avoid manual feature design and more freedom to select the set of features to describe the given sample. This work constructs a feature representation of nodes from data extracted from the internal LS-DYNA format describing finite element models of such aluminum profiles. After subjecting these data structures to the preprocessing pipeline, a feasible data format for machine learning applications was constructed. It consists of the following fields:

- Node initial coordinates - a tuple (x, y, z), containing spatial coordinates of a given node in the global coordinate system, assigned to a model.
- Voxel map - a representation of 3d model as a whole, capturing the relation between nodes and the global geometrical features of the member. For the sake of computation efficiency, the field is represented with a 100-dimensional vector, acquired from deep autoencoder pretrained on voxel maps obtained from original data.
- Timestep indices - a field, required in order to keep the sequential structure of an input.
- Local geometry features - information about the local geometry given by connectivity between the neighbouring nodes. Connectivity information may be considered an unoriented 1-connected weighted graph, with weights corresponding to the distance from the selected node to the neighbouring nodes. The amount of information about node spatial location in relativity to its neighbours may be fine-tuned with the maximum order of neighbour to include into the graph. The given dataset includes coordinates of the nearest neighbours up to the fifth-order, resulting in a 105-dimensional vector.

Corresponding target variables for such feature maps were obtained by running a numerical simulation of axial crashes for these rails. Illustrative examples of real-life profiles after crash tests and their numerical simulations may be seen in figures 4.2, 4.3 respectively. The shift history for each node while running 160 timesteps of direct simulation of the problem with defined geometry was captured and coupled with corresponding nodes to form the dataset.

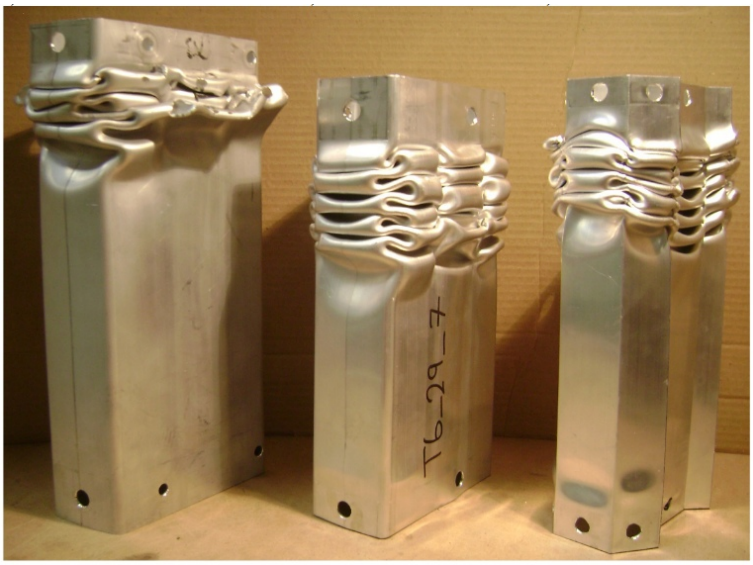


Figure 4.2: Experimentally crashed profiles [15]

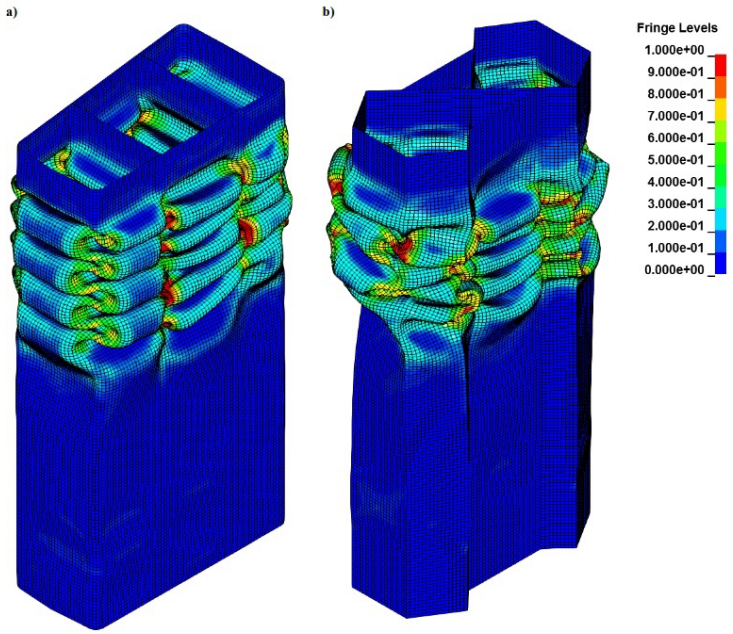


Figure 4.3: Simulated crash tube effective strain contours [15].

The flowchart of the autoencoder used to compress the representation of the 3D model is shown in the figure 4.4; its code implementation was written with Keras deep learning framework. The model was trained with Adam optimization algorithm with following set of hyperparameters: lr = 1e-3, $\beta_1 = 0.9, \beta_2 = 0.99$, eps = 1e-8. Additionally, the norm of model's gradient was clipped up to the value of 0.01. Convergence plots of autoencoder are yielded by figure 4.5.

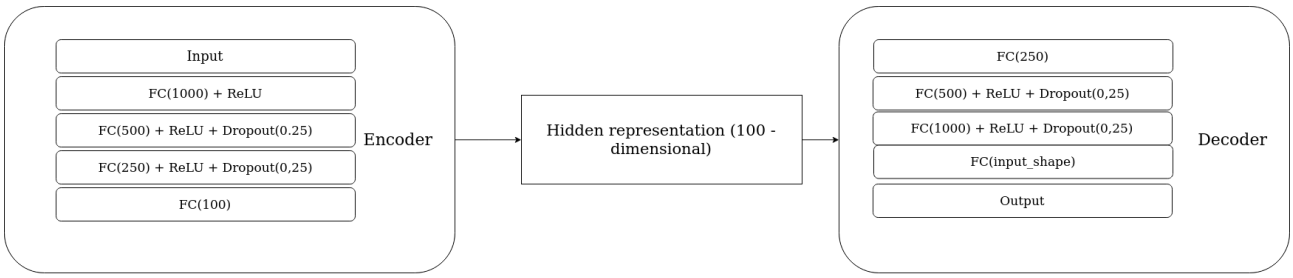


Figure 4.4: Flowchart of the autoencoder used for compressing LS-DYNA data representations.

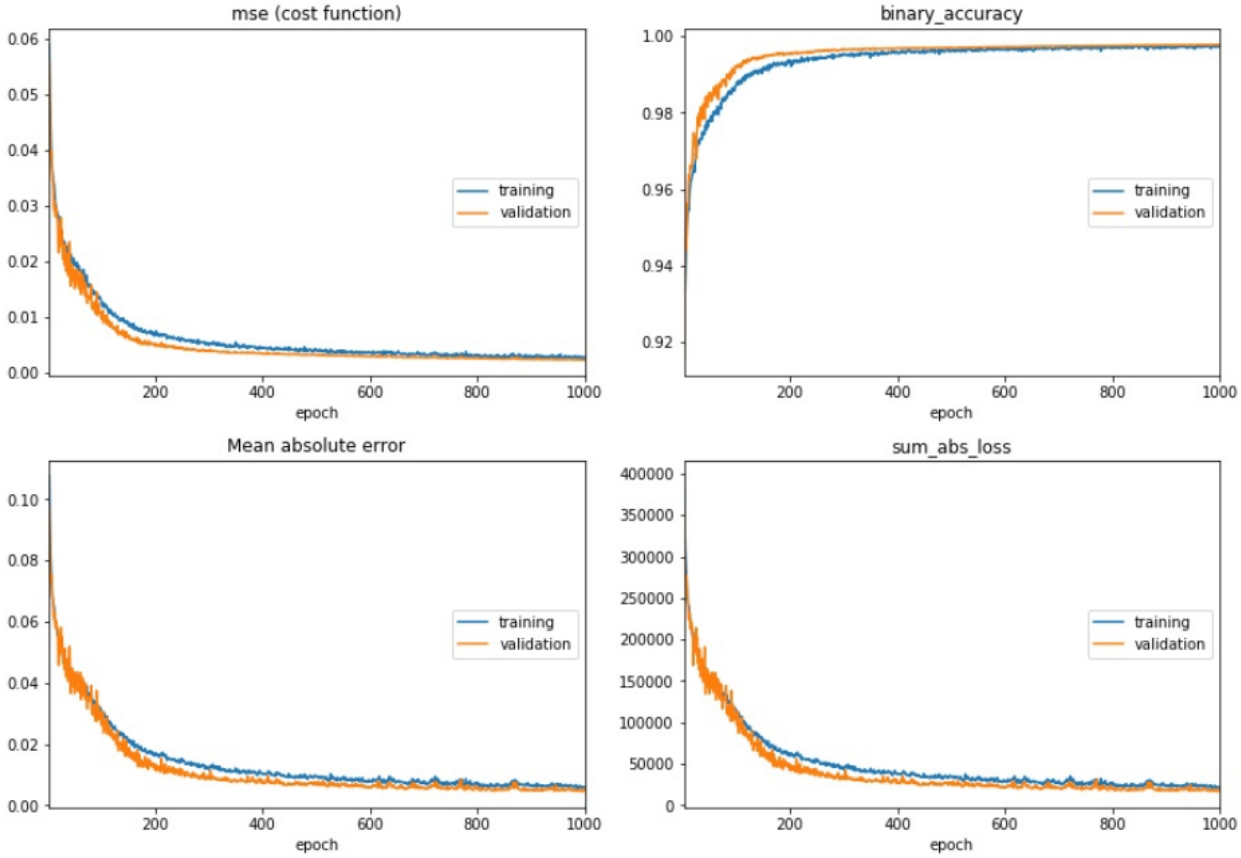


Figure 4.5: Convergence plots for deep autoencoder, trained on LS-DYNA data.

4.2 Neural network architecture

A flowchart of an architecture of a neural network used for shift prediction is shown in figure 4.6. In order to get a prediction, all node features except the time step indices are concatenated into a single vector. Further, the vector is copied 160 times; copies are stacked and concatenated with time step indices along the second axis. The resulting tensor is passed through a dense input layer, a stack of LSTM layers, and finally through a dense output layer. Weights of almost all layers are additionally subjected to regularization, limiting their norm. The first dense layer also applies dropout to its neurons with the probability of 0.25.

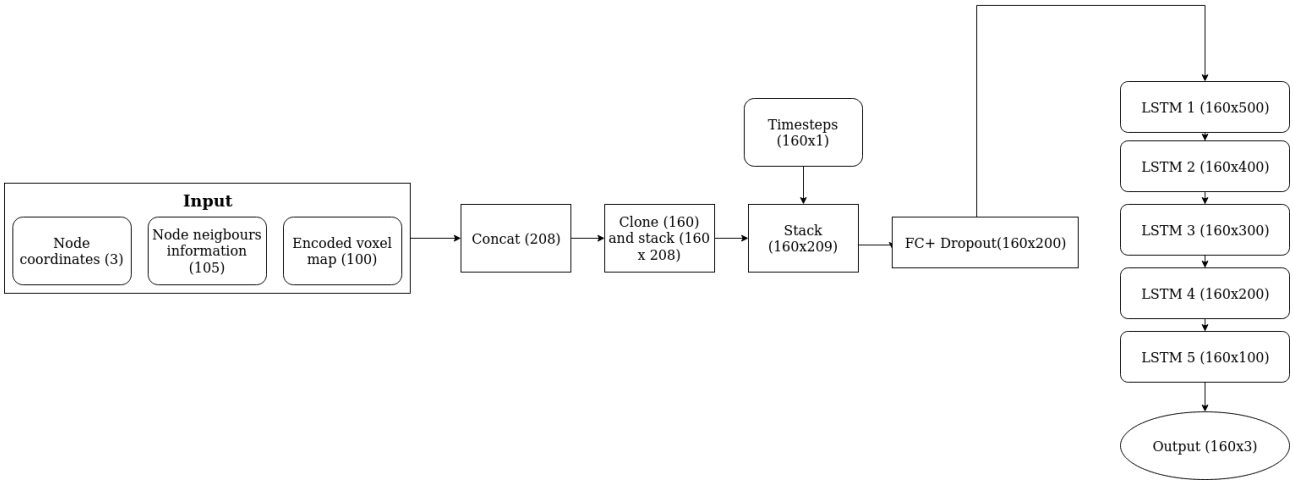


Figure 4.6: Block scheme of the baseline neural network architecture

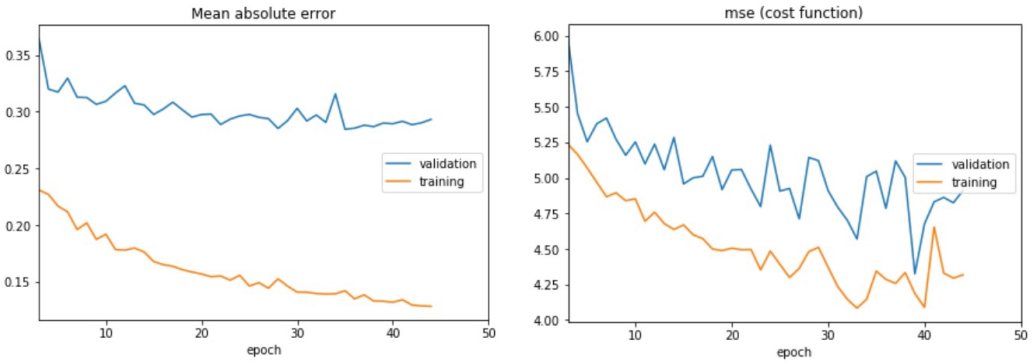
Layer №	Type	Output shape	Regularization	Dropout
1	Dense	(160,200)	l1_l2	0.25
2	CuDNN_LSTM1	(160, 500)	l1_l2	0.0
3	CuDNN_LSTM2	(160, 400)	l1_l2	0.0
4	CuDNN_LSTM3	(160, 300)	l1_l2	0.0
5	CuDNN_LSTM4	(160, 200)	l1_l2	0.0
6	CuDNN_LSTM5	(160, 100)	l1_l2	0.0
7	Dense	(160, 3)	No	–

Table 1: Summary on layer parameters for baseline neural network architecture

Described architecture can be trained end-to-end with a backpropagation algorithm. Implementation of it was made within the scope of Keras [20] deep learning framework; a summary of hyperparameters used in architecture is presented by table 1. The model was trained to minimize MSE loss with Adam optimization algorithm; the set of hyperparameters used was lr = 1e-3, $\beta_1 = 0.9, \beta_2 = 0.99$, eps = 1e-8. Model gradient norm was subjected to clipping, setting its highest possible value as 1 to prevent gradient exploding.

4.3 Results

Convergence plots after 50 epochs of training are provided with figure 4.7, and the final loss values are summarized in the table 2



(a) (b)

Figure 4.7: (a) MAE and (b) MSE convergence plots for AI framework final model.

Dataset	Mean Squared Error	Mean Average Error
Train	4.085	0.128
Test	4.325	0.285

Table 2: Final metrics for AI Framework trained on full data.

Figures 4.8 - 4.13 demonstrate the 3d heatmaps, built with LS-DYNA over the last timestep predictions, obtained with a neural network for various LD-DYNA models, included in training and test data.

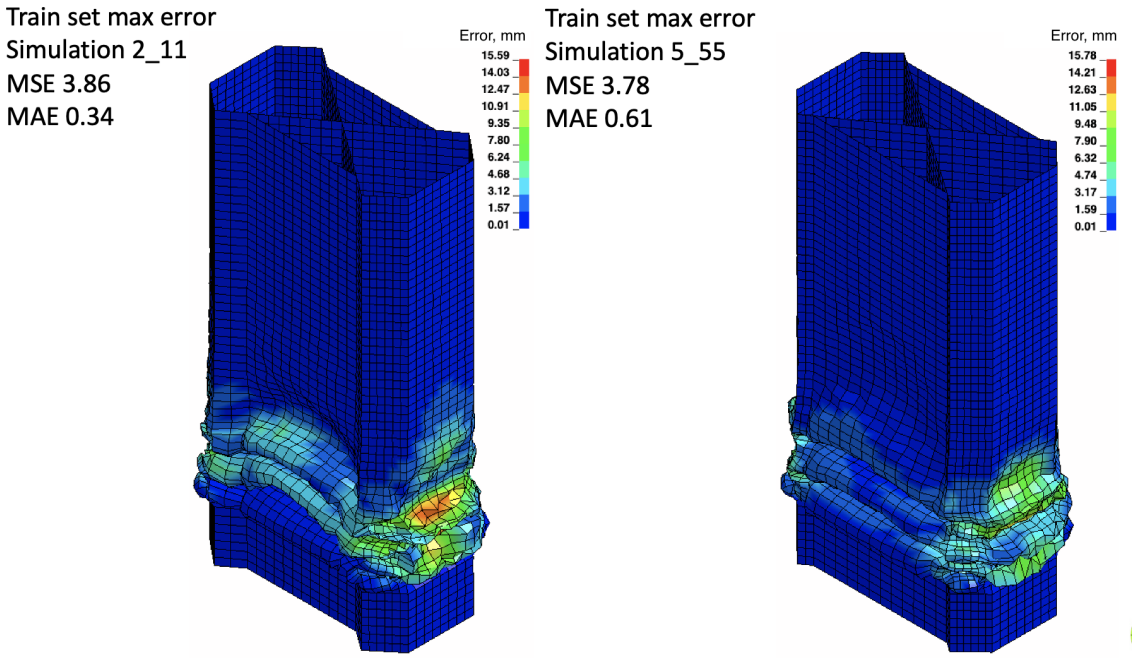


Figure 4.8: Predictions of framework painted over the LS-DYNA models from training subset with higher values of MAE error.

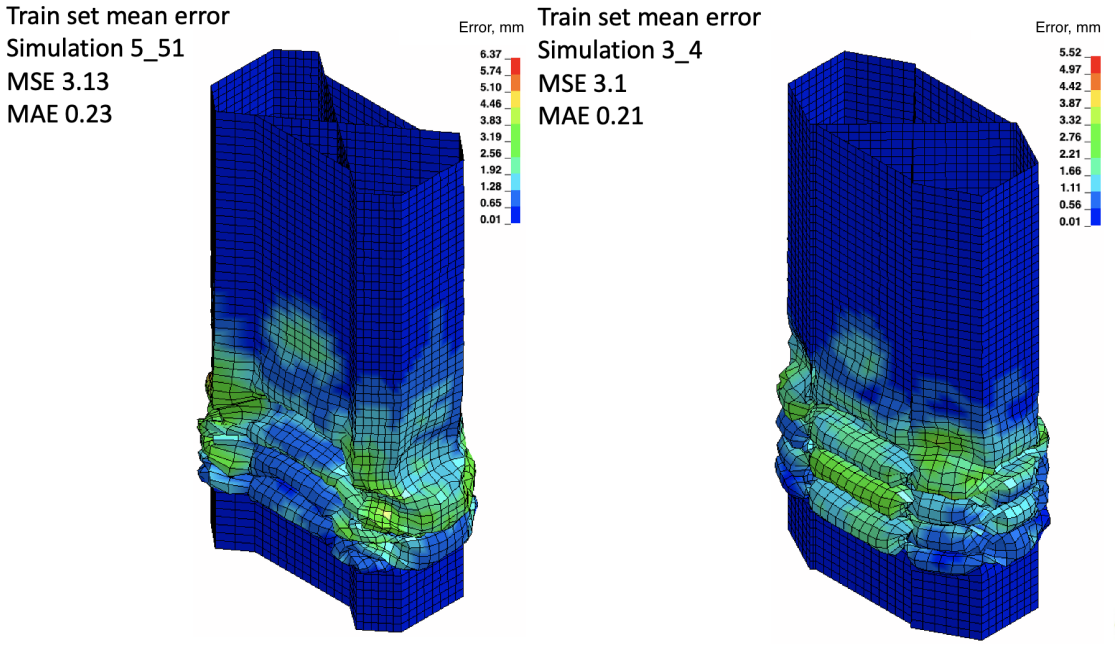


Figure 4.9: Predictions of framework painted over the LS-DYNA models from training subset with average values of MAE error.

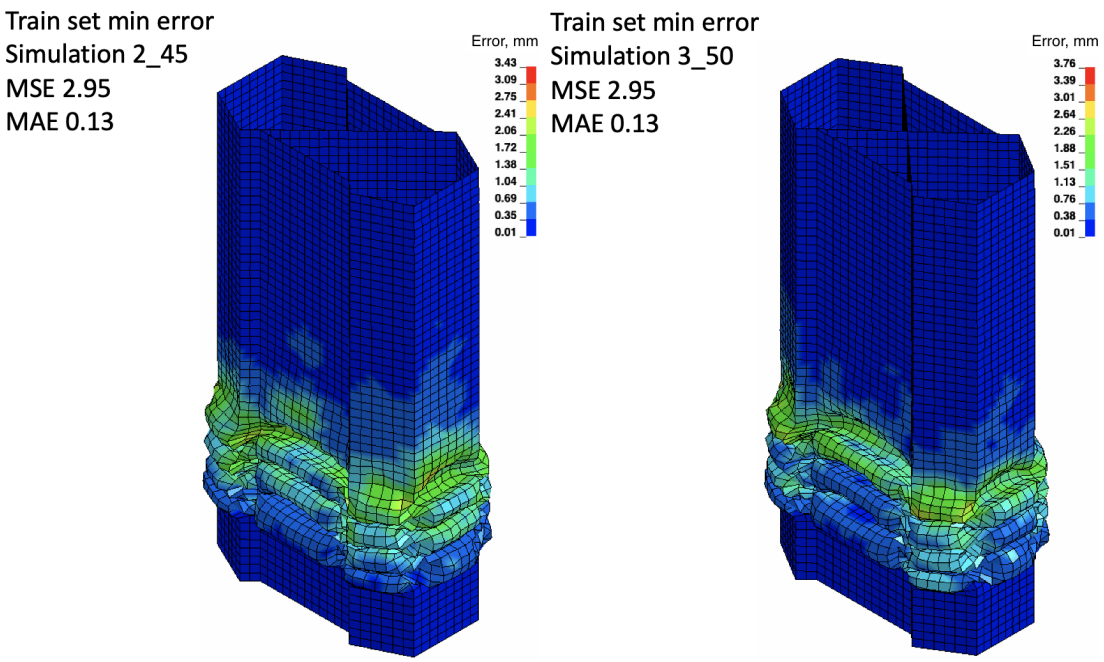


Figure 4.10: Predictions of framework painted over the LS-DYNA models from training subset with low values of MAE error.

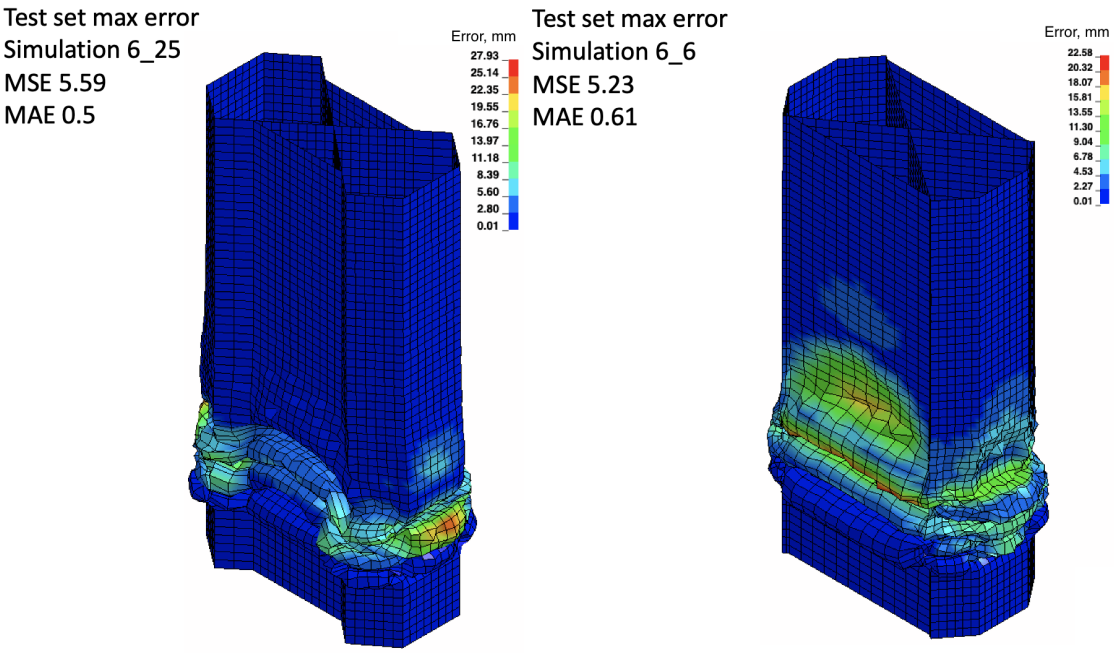


Figure 4.11: Predictions of framework painted over the LS-DYNA models from test subset with higher values of MAE error.

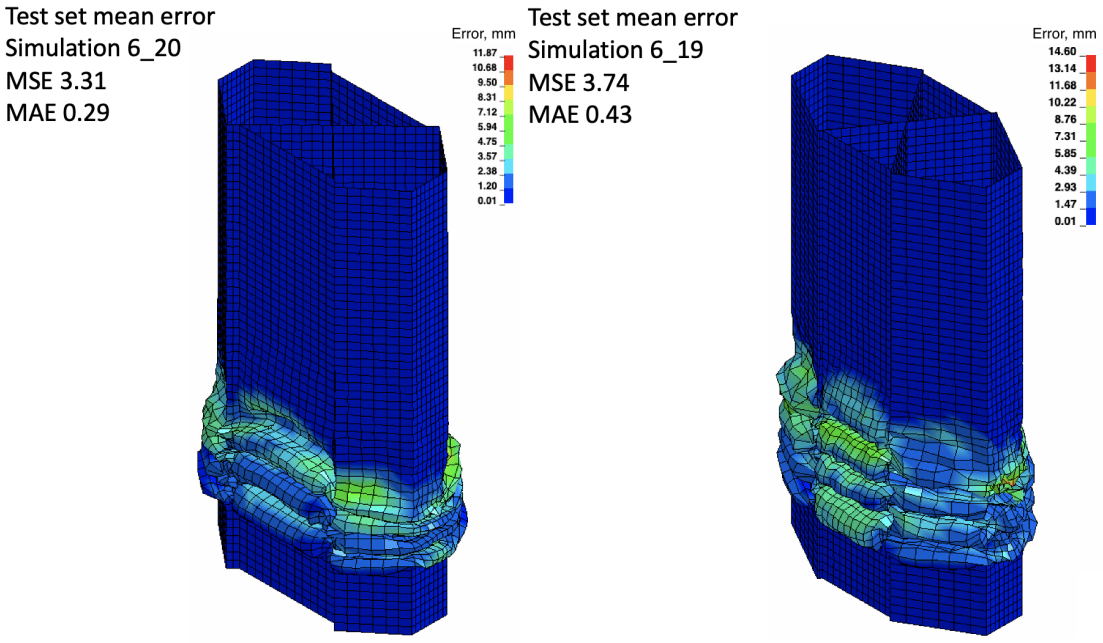


Figure 4.12: Predictions of framework painted over the LS-DYNA models from test subset with average values of MAE error.

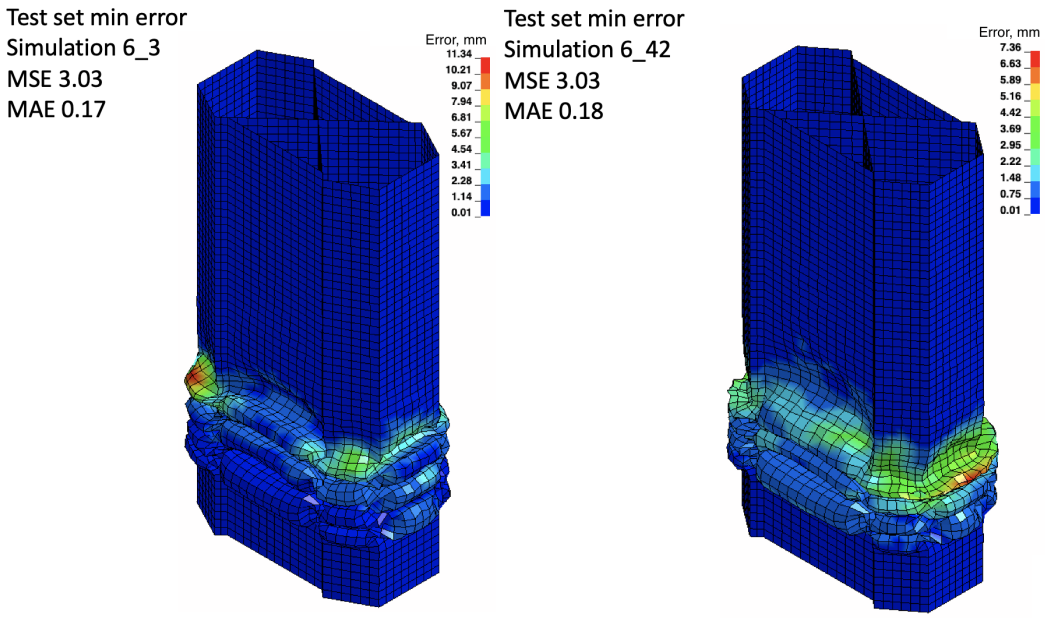


Figure 4.13: Predictions of framework painted over the LS-DYNA models from test subset with low values of MAE error.

5 Training optimization

The current chapter contains a detailed description of techniques used to improve the performance of the machine learning framework described in the previous chapters. The chapter elaborates on the motivation of their usage and technical details of the exact implementation of the techniques. Section 5.1 describes general intuition behind the employment of sampling methods and formulates the criteria of sampling strategy appropriate for the problem. Section 5.2 defines two different ways in which the random sampling approach can be applied to the current problem statement. A review of the heatmap-based sampling technique used further in work is given by the section 5.3. Finally, section 5.4 states how exactly the bagging technique is used in the scope of the work.

5.1 Node sampling

Direct numerical simulation results show that the complexity of a node trajectory heavily depends on the node's position. E.g., fixed nodes do not move at all, so the displacement value is always zero, and nodes on the top of the LS-DYNA model tend to move straightforwardly. In comparison, nodes belonging to the parts of the LS-DYNA model under strain move much more elaborately. It is natural to assume that such nodes generally carry more information about the crash process. As the benchmark dataset contains many non-informative nodes, we may suggest constructing a subset of significantly lesser volume without losing much information about the underlying process. If we have a machine learning model, potent to fit into the original data, training this model on such a subset may significantly reduce the required time for the convergence up to the excellent value of target metrics.

In order to assemble an informative subset from the original data, we propose data sampling. A suitable sampling strategy should match the following criteria:

- It should be generalizable and should utilize exclusively information obtained from the model to make decisions; ideally with no ad-hoc or human adjustment.
- It should be computationally efficient and should be able to process a piece of single model information in a matter of seconds.
- It should assume the exact representation of data for each new sample.

We employ two different sampling strategies, matching the criteria listed above - *random sampling*, and *importance sampling*.

5.2 Random sampling

Constructing the training subset with a random sampling strategy assumes drawing samples from the training set uniformly without repetitions. The most straightforward implementation of random sampling in application to the given dataset assumes merging nodes belonging to different models before the selecting procedure, thus, treating the aggregate of nodes as a single bucket of samples. Even though such an approach meets all the criteria mentioned in 5.1, it omits the inherent cluster-like structure of the nodes, emerging from being a part of the aluminum rail. This fact may eventually lead to the lesser informativity of the constructed subset than the original data, as nodes from a specific model may not enter the collected subset. We suggest composing the training subset by taking the constant number of nodes from each LS-DYNA model to address this issue. This work explores both of these approaches, training the baseline model with a total of 12 different sampling strategies. These strategies combine the datawide/modelwise sampling with the different nodes taken from each LS-DYNA model - 32, 64, 128, 256, 1024, or 2048, respectively (in case of datawide sampling, it means selecting $n * 300$ samples from the whole dataset). In order to verify model stability for training on the part of the data, the neural network's performance was cross-validated on the subsets obtained with suggested strategies while varying the seed of the random number generator.

5.2.1 Modelwise random sampling

Figure 5.1 provides convergence plots for MAE and MSE loss functions respectively for a neural network trained with random sampling strategy, sampling 32 nodes from each LS-DYNA model. Graph demonstrating the alternations in the convergence plots with resampled subsets is given by the figure 5.2.

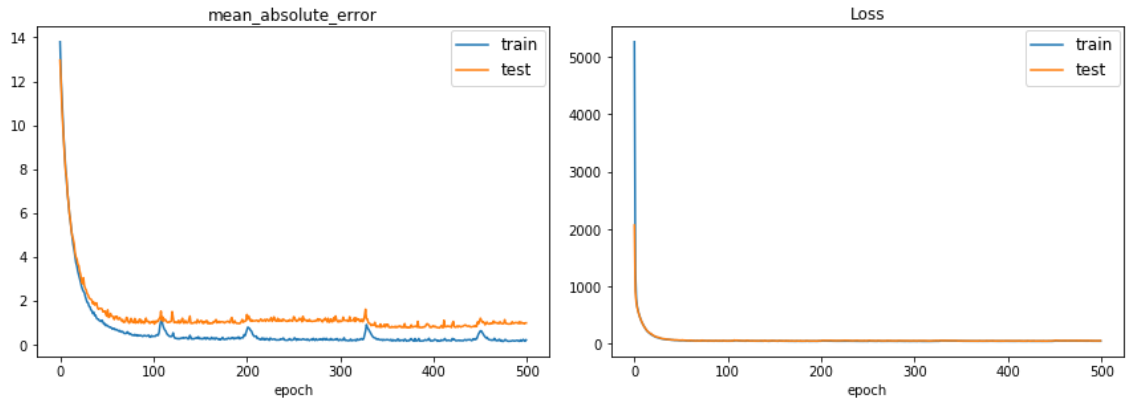


Figure 5.1: MAE and MSE history for training the neural network with dataset, obtained with sampling 32 random nodes from each LS-DYNA model.

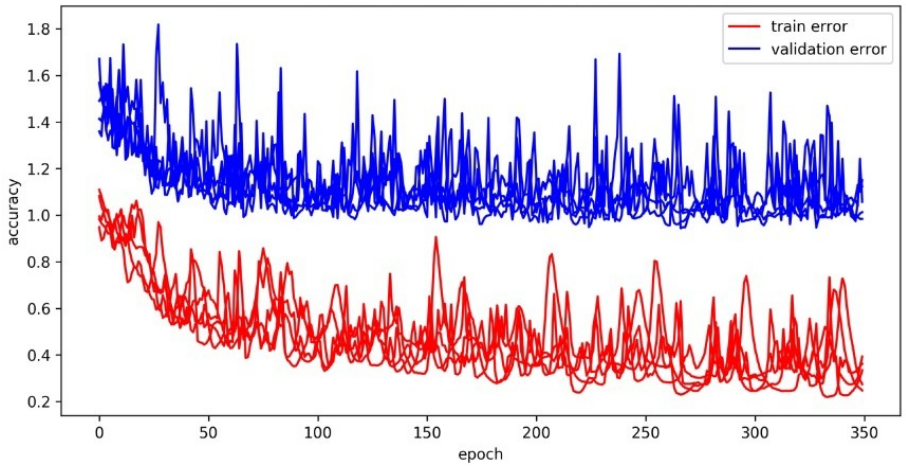


Figure 5.2: MAE evolution in time during the training over different subsets, obtained with sampling 32 random nodes from each LS-DYNA model.

Sample	**Mean Squared Error**	**Mean Average Error**
1	6.523	1.042
2	6.773	1.036
3	6.881	1.052
4	6.946	1.053
5	6.766	1.030

Table 3: Final metrics value on test data for individual models, trained on subsets obtained with randomly sampling 32 nodes from each LS-DYNA model.

Table 3 summarizes the metrics values obtained while training different model instances with such approach. It can be seen that the final value of the loss function is one order of magnitude more significant in comparison to the baseline model, and the MAE error is two

times bigger. However, sampling 32 nodes out of 8600 means that the training subset is 269 times smaller than the original data, resulting in 15s average epoch processing time. That being said, 500 training epochs yields a total training time of 2.08 hours, while the estimated time for the baseline experiment is 111 hours. On average, the evolution of loss function stagnates after approximately 150 training epochs, which means that the actual time required for a model to converge is even lower. Moreover, according to figure 5.2, the stochastic nature of subset selection does not affect the model performance, as the differences between trials are minor, further confirming the robustness of the process.

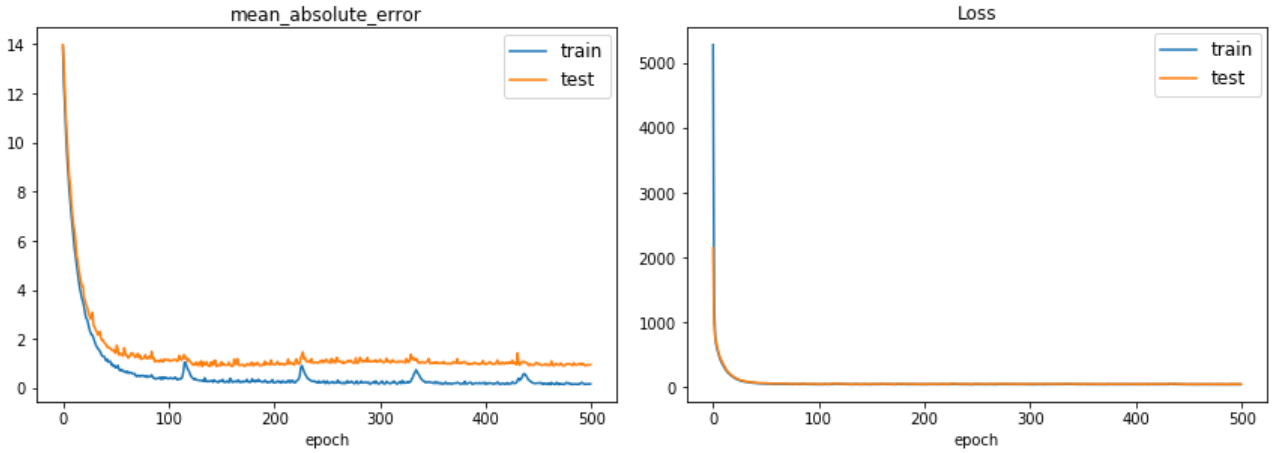


Figure 5.3: MAE and MSE history for training the neural network with dataset, obtained with sampling 32 random nodes from each LS-DYNA model.

5.2.2 Datawide random sampling

Figure 5.3 gives convergence plots for MAE and MSE loss functions respectively for a neural network trained with random sampling strategy, sampling $32 * 300 = 9600$ nodes in total from the training dataset. An illustration of how convergence plots change depending on the subsets obtained with such procedure is provided by figure 5.4. Final metrics values for the datawide random sampling are provided by table 4. The study shows no noticeable differences between applying random sampling to the existing data, which points out the uniformity of its structure. The efficiency of the suggested approach in reducing the time required to train the model up to the sub-optimal value of target metrics opens up new possibilities for ML algorithms; e.g., learned weights may serve as an initial point for the fine-tuning the model on the whole dataset.

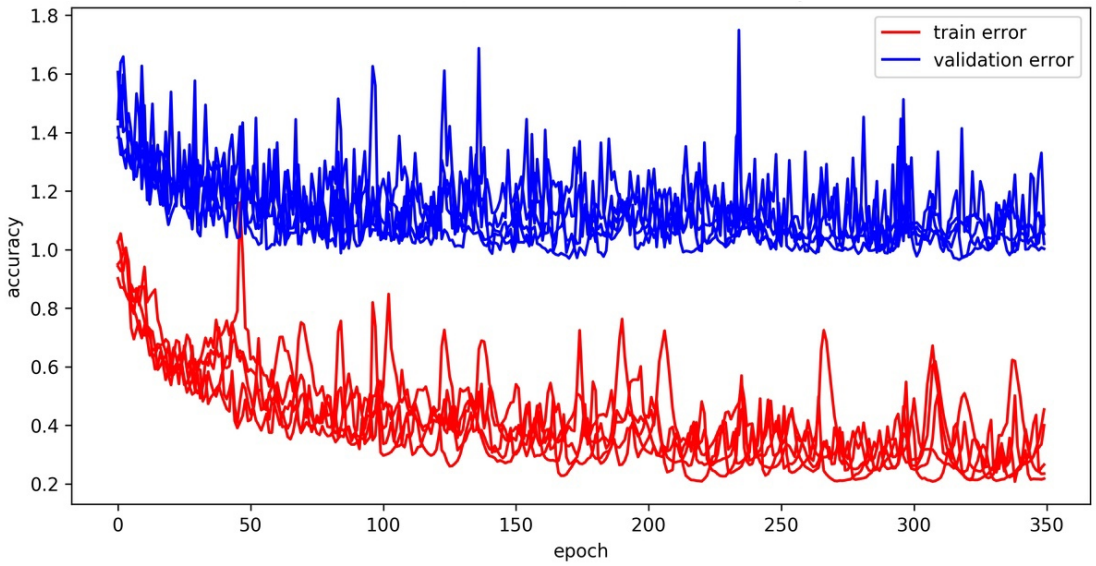


Figure 5.4: MAE evolution in time during the training over different subsets, obtained with sampling 9600 nodes from the dataset as a whole.

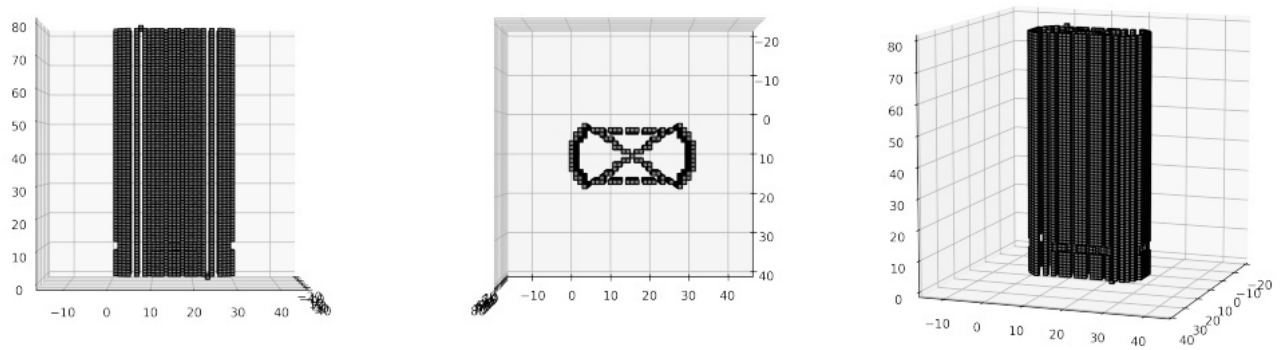
Sample	Mean Squared Error	Mean Average Error
1	6.452	1.024
2	6.584	1.077
3	6.683	1.052
4	7.299	1.070
5	7.102	1.064

Table 4: Final metrics value on test data for individual models, trained on subsets obtained with sampling 9600 nodes in total from the whole dataset.

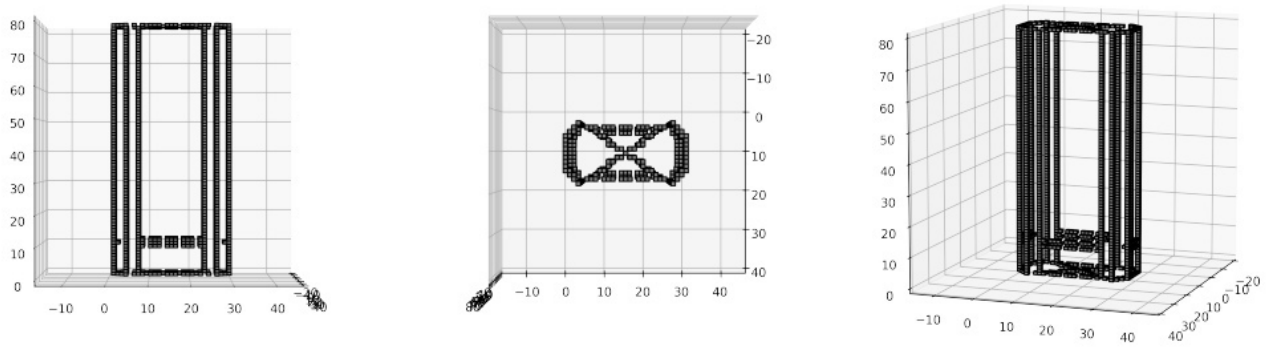
5.3 Importance sampling

Following the general intuition of importance sampling, we may construct a sampling algorithm that can separate highly informative nodes from the others. One possible way to do that is to avail ourselves of a node's connectivity and geometry information. Uninformative nodes may be filtered by several criteria, such as the total number of neighbours of a given order or estimates based on the quantity of output deformation, e.g., total absolute shift value or total absolute shift value along the chosen axis. Figure 5.5 demonstrates the separation of nodes of a given model by evaluating the sum of relative displacements of neighbouring nodes. Specifically, we relate a node into the first group if it has a zero-valued sum of direct neighbours normalized relative coordinates and into the second group otherwise. It can be seen that such criterion

yields node separation based on the uniformity of node clusters - the nodes on the model edges are in group one, while those on the borderlines are assigned to the second group. An approach



(a) Group 1.



(b) Group 2.

Figure 5.5: An example of sampling-based division of LS-DYNA model nodes.

suggested in this work uses the metrics evaluates the importance of a single sample, consisting of multiple terms, 6 of them being

- Neighbour count q_n
- Total neighbour distance q_d
- Absolute shift - the norm of the total shift of node from initial q_{st}
- Shift along each of the axes q_{sx}, q_{sy}, q_{sz}

In addition, we assume that each of the mentioned terms is normalized, i.e., taken as the ratio between the value of given metrics on a given sample and the global maximum of the same metrics over the whole dataset. The final estimate of importance for sample i has the

following form:

$$p^i = \frac{\hat{p}^i}{\max\limits_i \hat{p}^i} \cdot \mathbb{I}\left(\frac{\hat{p}^i}{\max\limits_i \hat{p}^i} >= 0.25\right) + 0.25 \cdot \mathbb{I}\left(\frac{\hat{p}^i}{\max\limits_i \hat{p}^i} <= 0.25\right) \tag{5.1}$$

$$\hat{p}^i = q_n^i + q_d^i + q_{st}^i + \frac{1}{3}\left(q_{sx}^i + q_{sy}^i + q_{sz}^i\right) + 3q_r^i \tag{5.2}$$

, where the quantity q_r^i is defined as

$$q_r^i = \tilde{q}_r^i \left(\frac{\tilde{q}_r^i}{\max\limits_i \tilde{q}_r^i} - 1\right) \tag{5.3}$$

$$\tilde{q}_r^i = \sum_{t=1,\, \mathrm{j} \in x,y,z}^{160} |s_j^i(t)| \tag{5.4}$$

Here $s_j^i(t)$ is the shift of the node i at the time step t along the axis j. Figure 5.6 shows the sample of a heatmap, built over the geometry of the LS-DYNA model according to the designed importance policy. It can be seen that the chosen sampling strategy prioritizes nodes situated near the edges and concavities of the model, which yields a higher shift in the crash experiment. Figure 5.7 exposes the n top nodes of the fixed LS-DYNA model, filtered by their importance, $n \in 50, 100, 500, 2000$.

To evaluate how the quality of obtained solution depends on the number of nodes drawn from each model, we construct different subsets, extracting 32, 64, 128, 256, 1024, or 2048 node instances from each LS-DYNA model. To verify the stability of the selected sampling strategy, we cross-validate the neural network, resampling such subsets repeating the corresponding experiments.

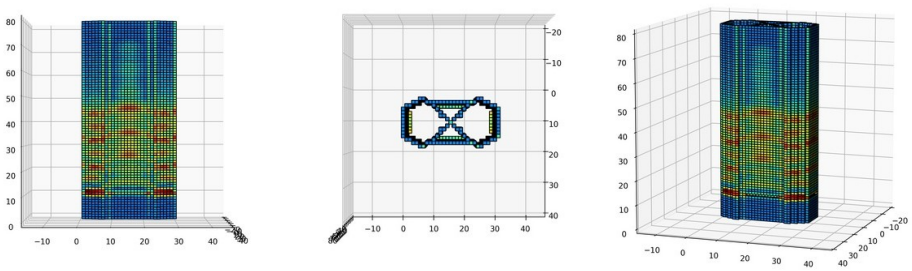
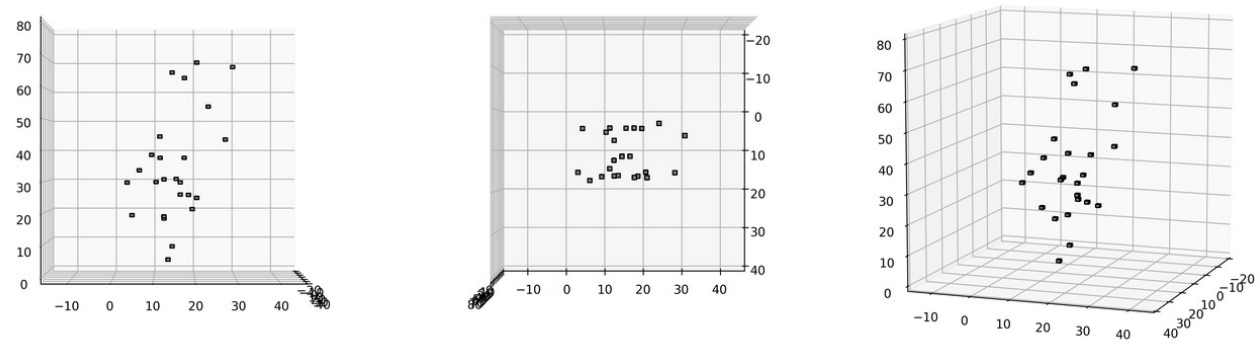
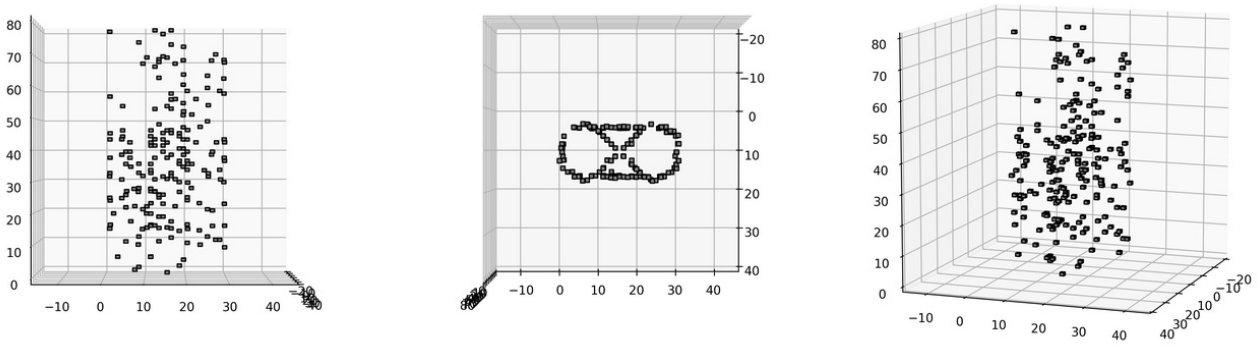


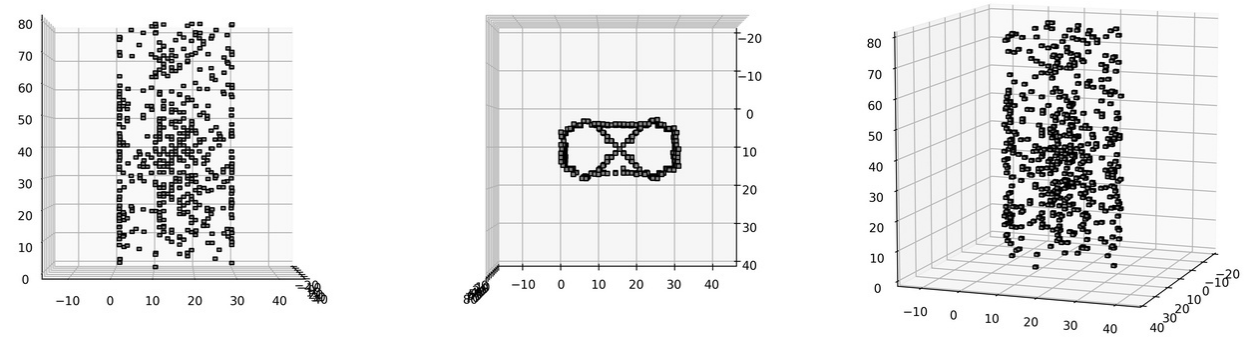
Figure 5.6: Heatmap sample over the LS-DYNA model.



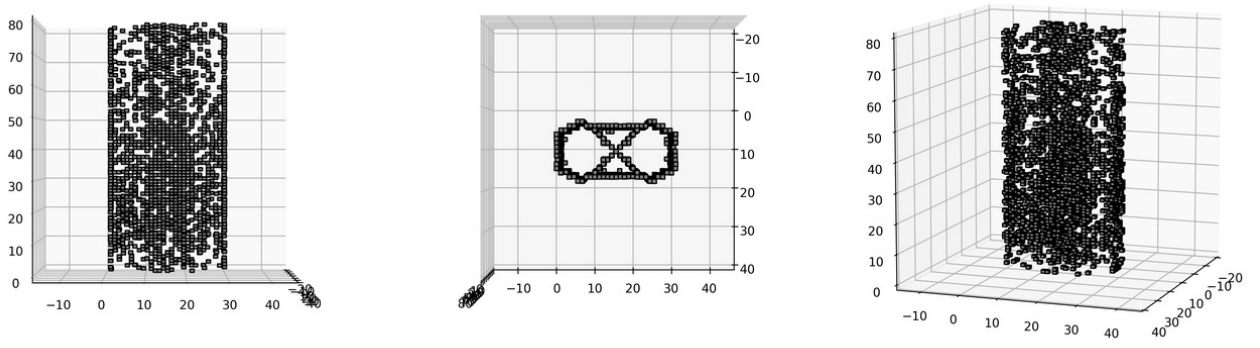
(a) $n = 25$.



(b) $n = 100$.



(c) $n = 500$.



(d) $n = 2000$.

Figure 5.7: Top a) 25, b) 100, c) 500, d) 2000 nodes sorted by their importance in accordance to the designed sampling policy.

Convergence plots for MAE and MSE for a neural network trained with importance sampling strategy are given by the figure 5.8. Figure 5.9 gives an insight on how the training process changes depending on different subsets acquired with the sampling strategy.

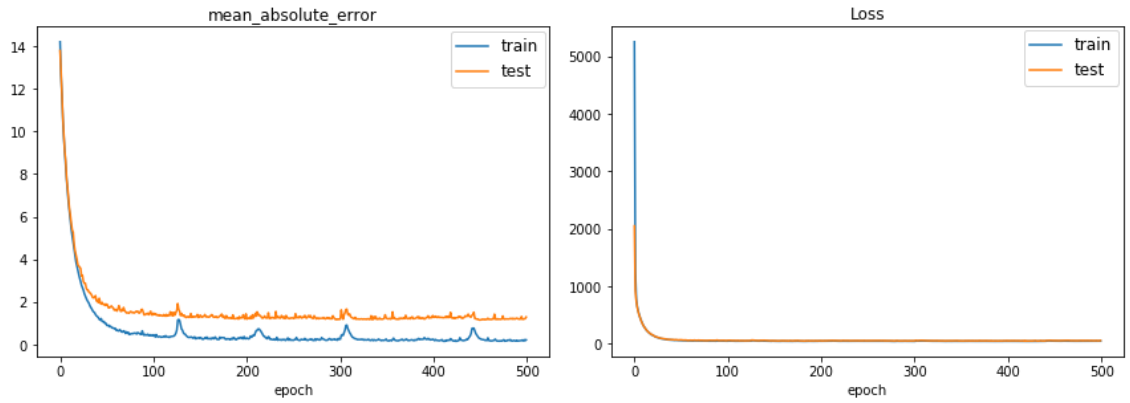


Figure 5.8: MAE and SE history for training the neural network with dataset, obtained with sampling 32 nodes from each LS-DYNA model according to the sampling policy, introduced in 5.3.

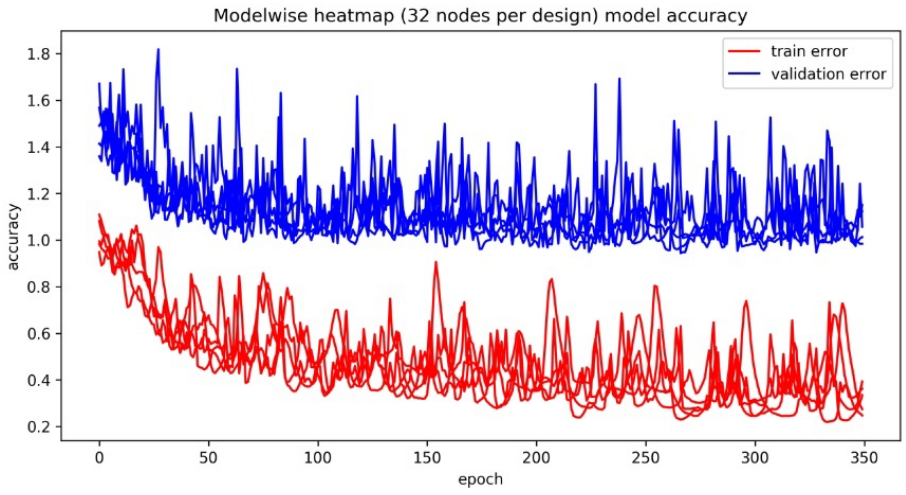


Figure 5.9: MAE evolution in time during the training over different subsets, obtained with sampling 32 random nodes from each LS-DYNA model.

Sample	Mean Squared Error	Mean Average Error
1	6.299	1.030
2	6.403	1.033
3	6.606	1.216
4	6.512	1.041
5	6.140	1.023

Table 5: Final metrics value on test data for individual models, trained on subsets obtained with sampling 32 nodes from each model using heatmap strategy.

Table 5 summarizes the metrics values observed during the training process for different model instances. The resulting metrics values for the designed sampling approach are of the same magnitude as for the random sampling and differ from that of the baseline approach in the same way. The figures above show that there is little to no gain from using importance sampling over the random in the case of the considered dataset. Creating the prior over the data yields a small overhead in total training time. However, the proposed importance formula does not depend on the model output; thus, the prior estimation can be done at the data preprocessing stage. For that reason, the estimated total training time remains the same as in 5.2. Figure 5.9 shows that the proposed sampling design is also stable in the sense of the subset variation, and the procedure of training the neural network with such sampling strategy is robust.

5.4 Model ensembling

As mentioned in 2.6.4, overfitting is one of the most common problems emerging in the practice of neural networks usage. This issue is especially relevant in the case when complex neural networks are trained on small datasets. As current work explicitly suggests sampling subsets of the original data to improve the computational effectiveness of the framework. Predictions of models trained in such a way are more unstable than the baseline, despite low bias due to the high statistical capacity of the used neural network architecture. The possible way to address this issue is to combine models into an ensemble, averaging their predictions.

In the scope of this work, we are following the bagging ensembling technique. Our bagging implementation assumes training several baseline-like neural networks on the different subsets

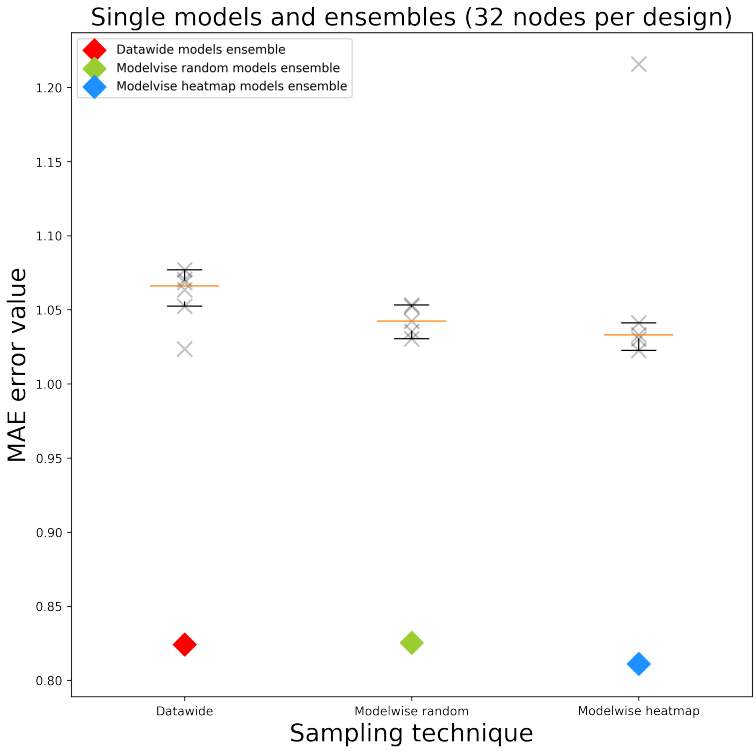
of the original training dataset due to the different realization of the chosen sampling function. This procedure is repeated for each sampling algorithm. We consider the average value of the predictions of these neural networks as the prediction of the whole ensemble.

Table 6 represents final metrics values for the ensembles, assembled from the models, trained with modelwise random sampling, datawide random sampling, and modelwise heatmap sampling strategies, respectively. Conducted experiments show that combining models into an ensemble by averaging their outputs impacts the quality of predictions positively: the value of MAE on the test subset is 1.26/1.28/1.32 times lesser in average compared to the models included in the ensemble for employed sampling strategies. As for MSE values on test subset, the respective fractions are 1.76/1.81/1.81. Scatterplot, comparing ensembles with its constituents by metrics values, is given by the figure 5.10

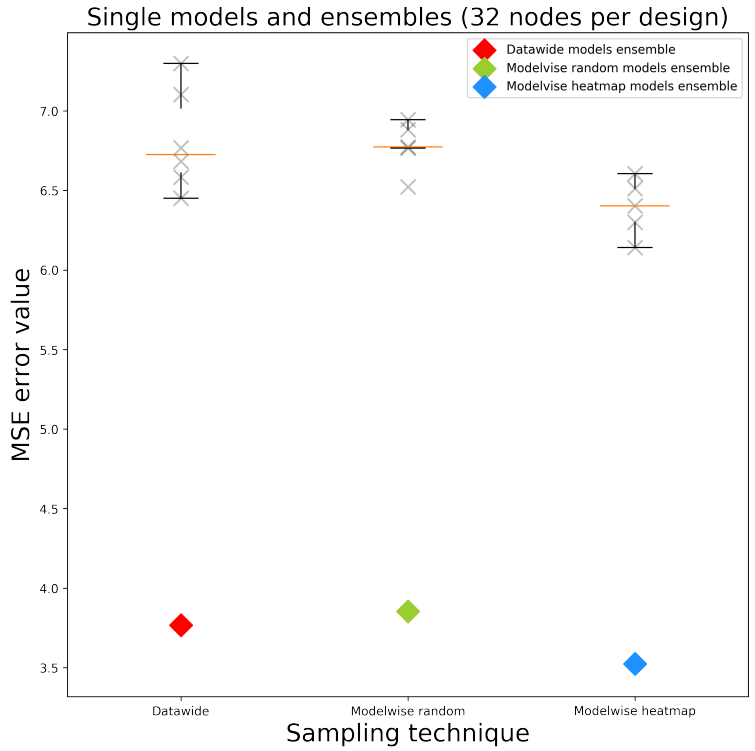
Sampling Strategy	**Mean Squared Error**	**Mean Average Error**
Random modelwise	3.855	0.825
Random datawide	3.767	0.824
Heatmap-based	3.523	0.811

Table 6: Final metrics value on test data for model ensembles, trained on subsets obtained with sampling 32 nodes from each LS-DYNA model using heatmap strategy.

As mentioned before, creating a new subset for training each model sample results in an additional computational overhead; however, both the additional expenses and the training time for an ensemble scale linearly with the number of models. Experiments show that combining only five models with the trivial voting mechanism results in a noticeable quality jump. Model ensembles still have higher MAE compared to the baseline, but all of them notably overperform the baseline on MSE while being much easier to train. Thus, building an ensemble of models trained on tiny fractions of the data may be viewed as an appropriate technique to enhance the framework's performance as a whole.



(a) Mean Average Error



(b) Mean Squared Error

Figure 5.10: Summary scheme, comparing a)MAE and b) MSE errors for separate model instances and model ensembles.

5.5 Comparison of sampling strategies and scalability

Figure 5.11 demonstrates a bar chart, showing the dependency of final value of MAE on the test set on the sampling strategy used for training model instances for exact-sized training subsets of 9600 samples in total. According to it, in general, model instances trained on datasets

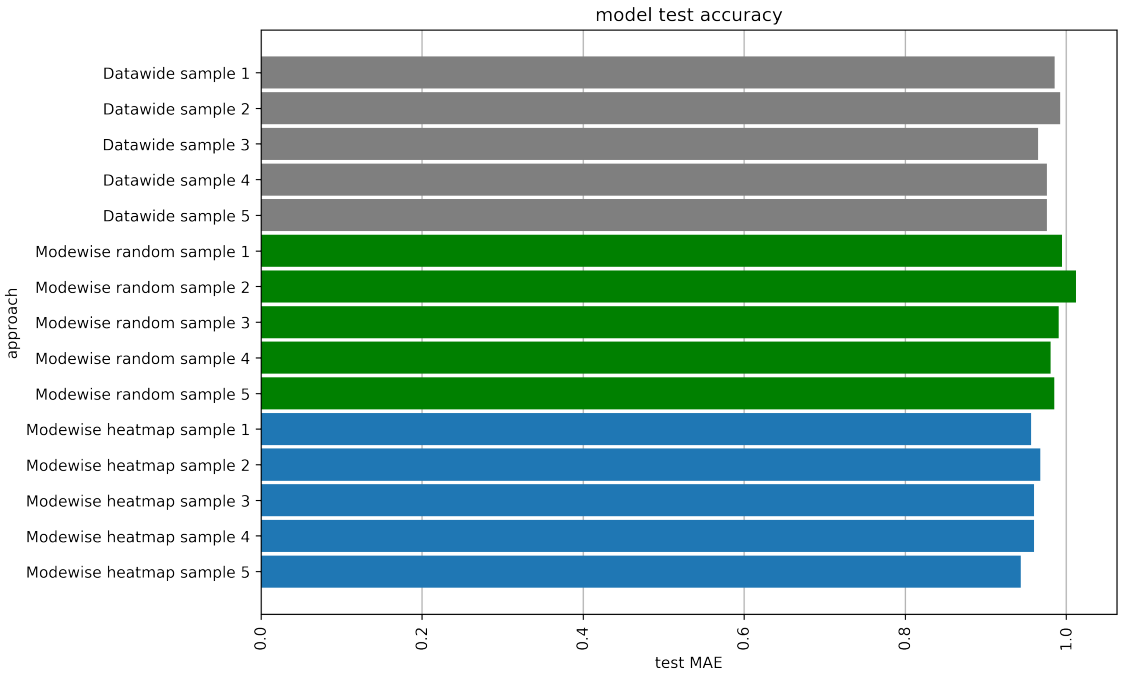


Figure 5.11: MAE values for model instances, trained with different sampling strategies on different subsets of fixed size (9600 samples).

created with heatmap sampling achieve better performance than other strategies. However, the actual difference between the metrics values does not allow us to state that discrepancies are caused precisely by using a different sampling strategy.

To investigate the issue, non-systematic research was conducted to discover how the quality of the solution depends on the size of subsets used to train model instances. Figures 5.12, 5.14 provides convergence plots for model instances, trained with the employed sampling strategies while picking 64 / 128 nodes from each LS-DYNA model. The comparative bar charts, demonstrating final MAE values for such model instances are given by figures 5.13, 5.15. As expected, predictions quality non-linearly scales depending on the size of the subset for the model instances to be trained. When 128 nodes are taken from each model, heatmap-based sampling demonstrates top performance in terms of final metrics value. However, this is not the case for 64 samples being picked from the models.

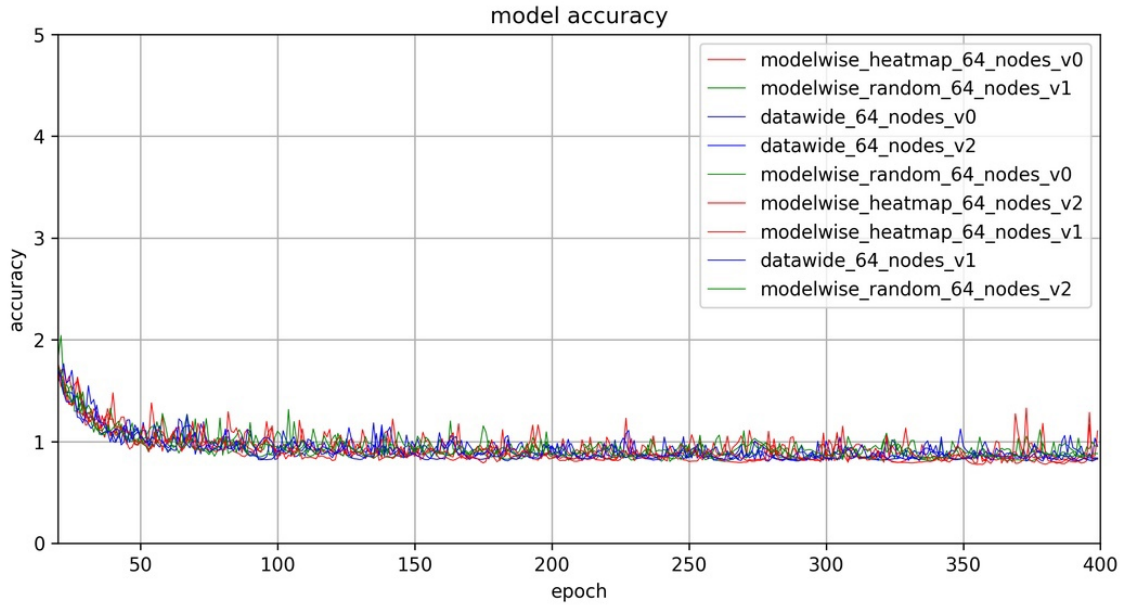


Figure 5.12: Metrics evolution for model instances trained on the subsets made with different sampling strategies by picking top 64 samples.

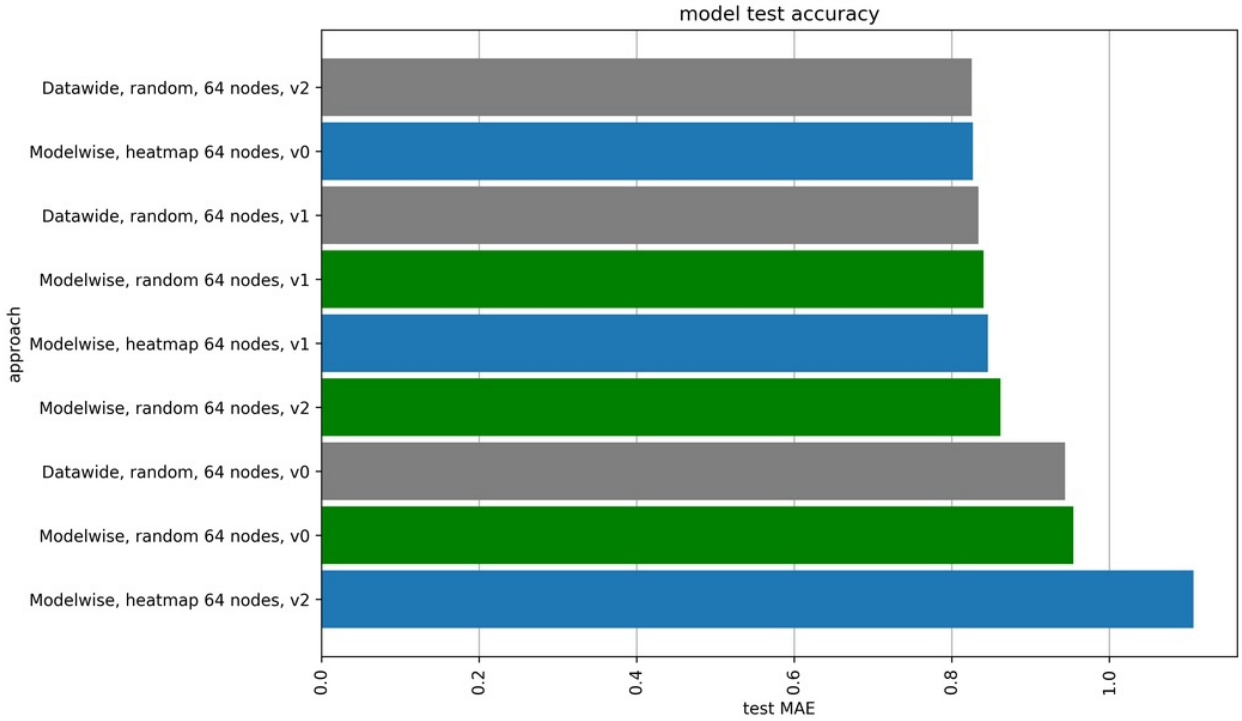


Figure 5.13: Summary on estimated MAE error values on the test dataset for the model instances trained on the subsets made with different sampling strategies by picking top 64 samples.

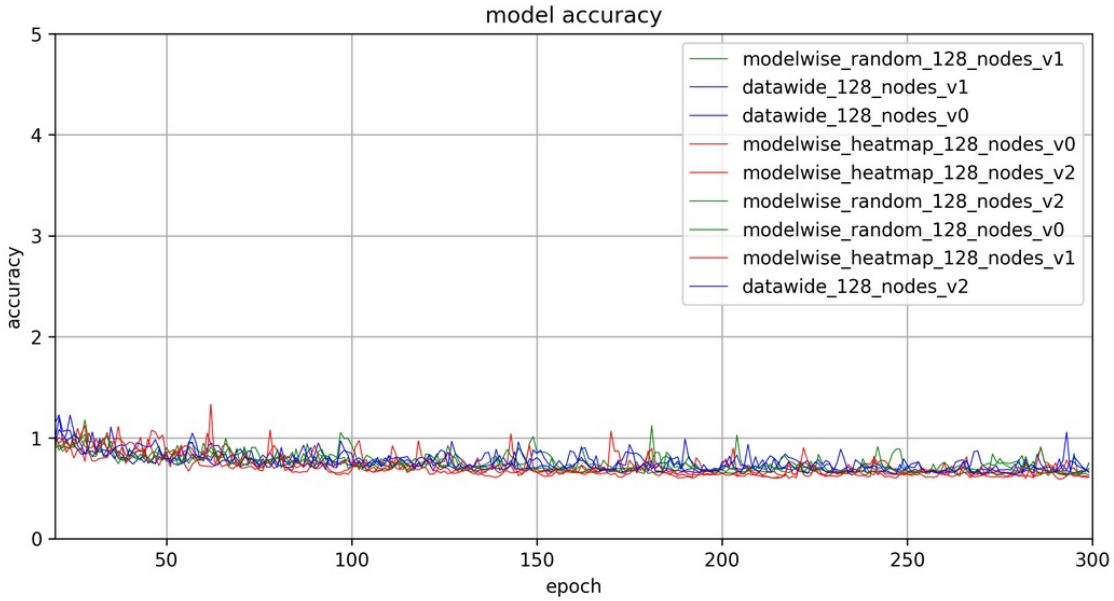


Figure 5.14: Metrics evolution for model instances trained on the subsets made with different sampling strategies by picking top 128 samples.

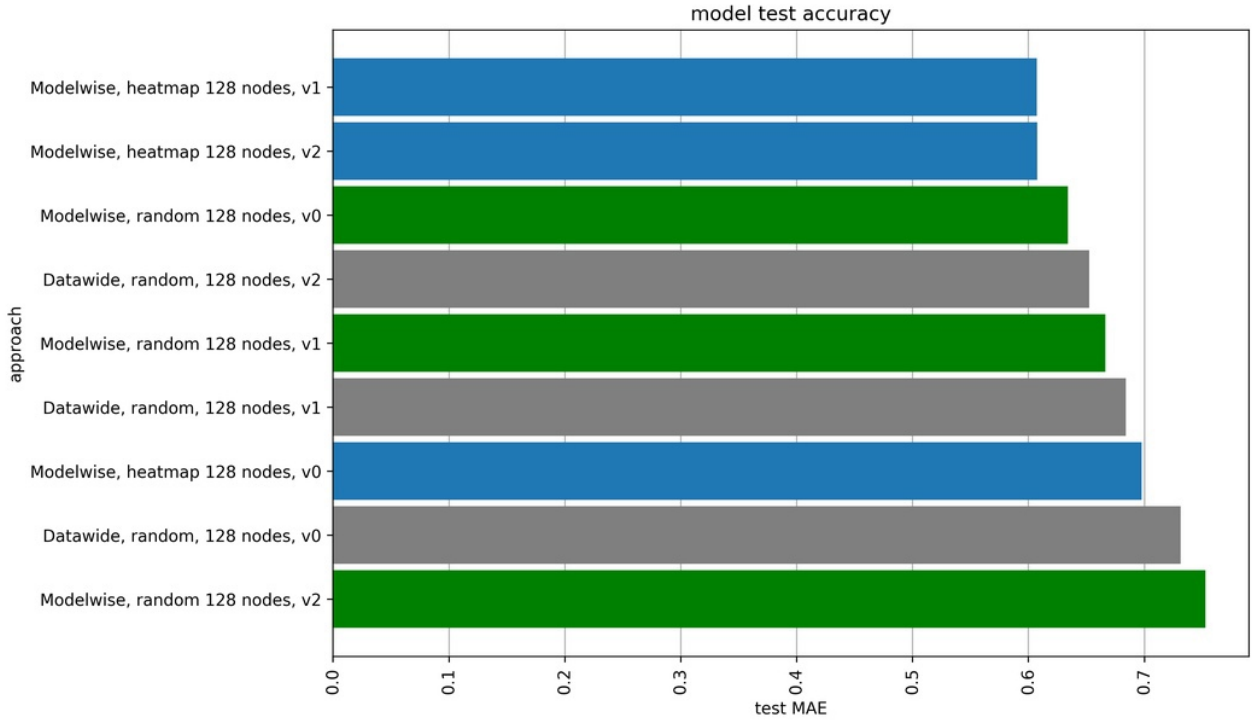


Figure 5.15: Summary on estimated MAE error values on the test dataset for the model instances trained on the subsets made with different sampling strategies by picking top 128 samples.

Figures 5.16, 5.17 yield bar charts with final MAE values for neural network instances trained on datasets, obtained with sampling 256, 1024 and 2048 nodes from each LS-DYNA model. It can be seen that expanding the training subsets leads to substantially better values of MAE on the test subset. However, detailed research on the scalability of the obtained numerical solution is out of the scope of the thesis.

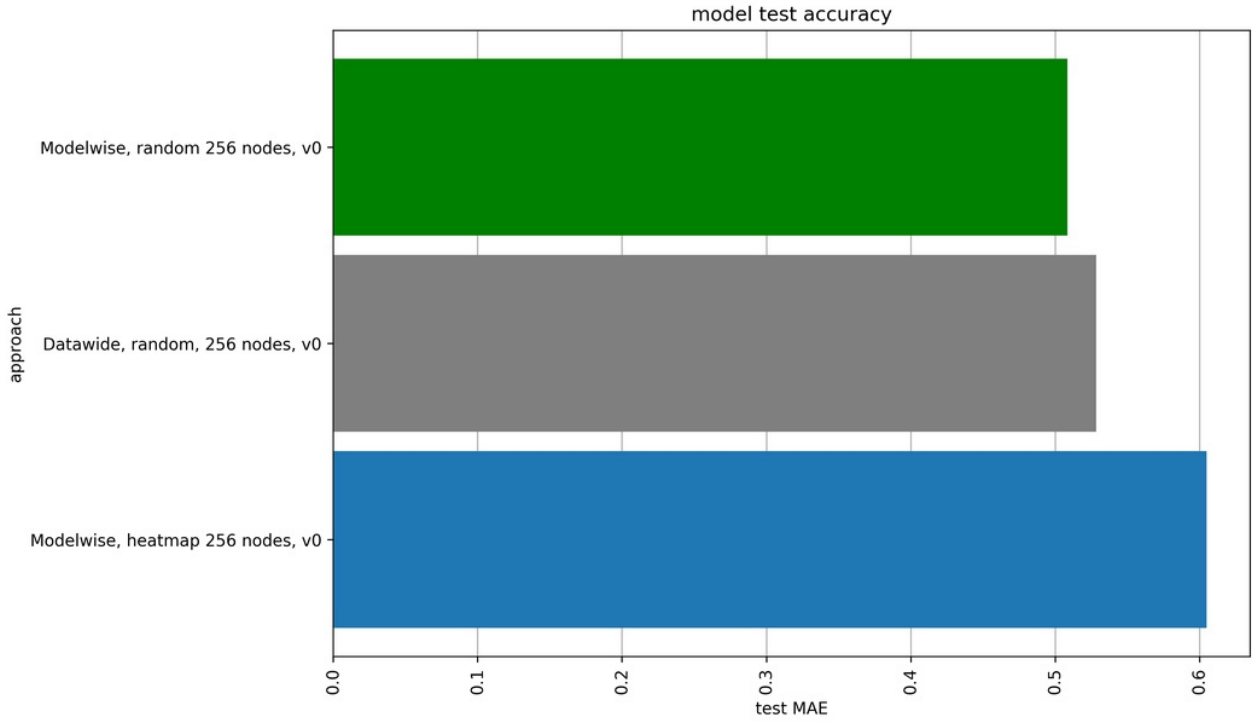
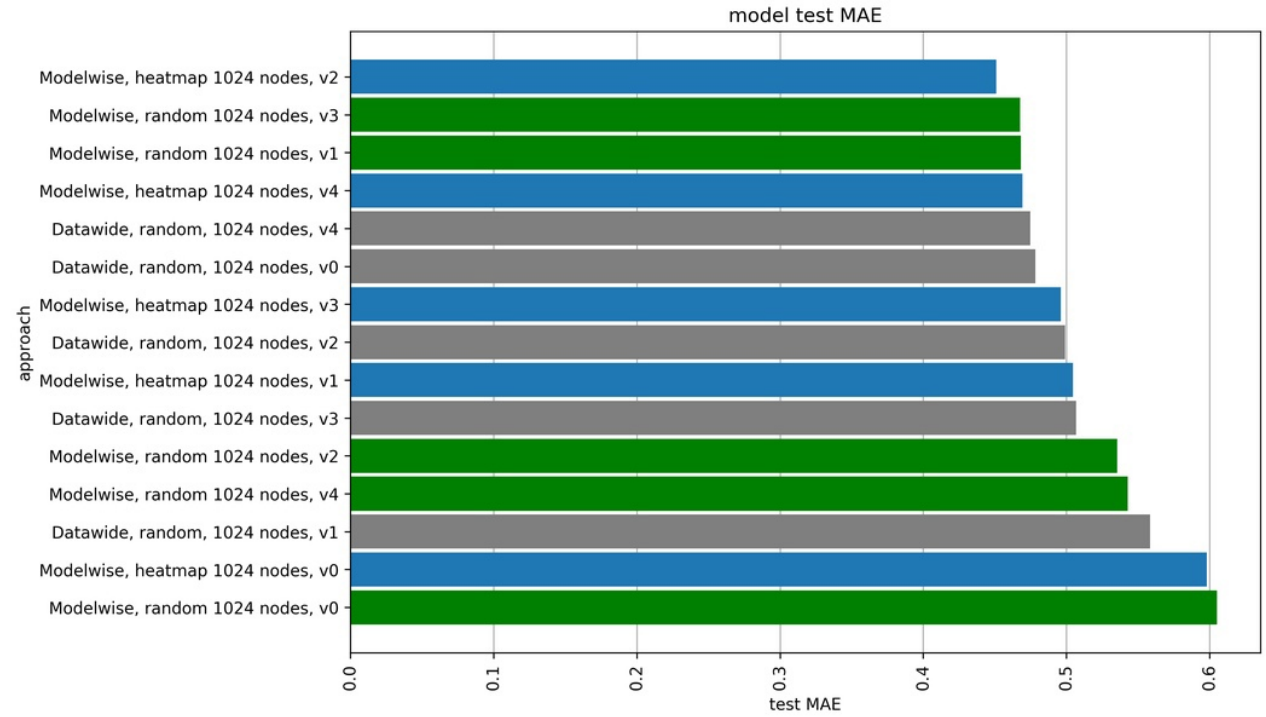
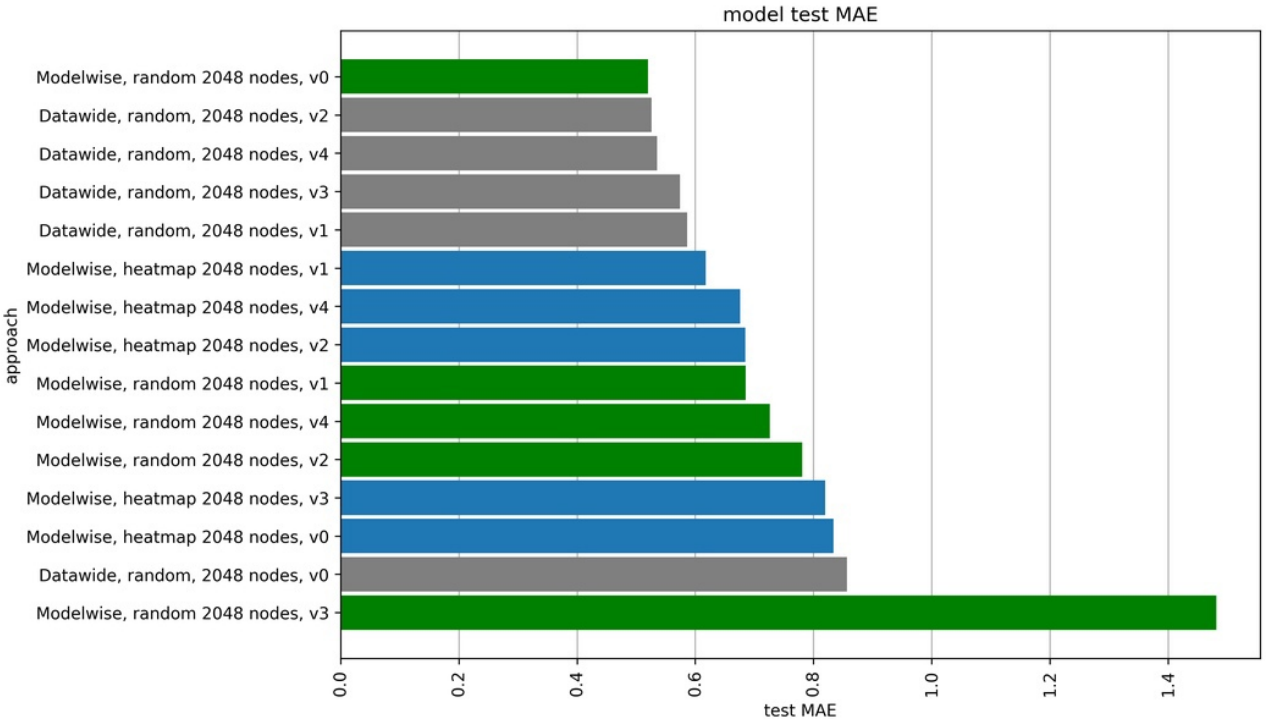


Figure 5.16: Summary on estimated MAE error values on the test dataset for model instances, trained on the subsets made with different sampling strategies by picking top 256 samples



(a) $n = 1024$



(b) $n = 2048$

Figure 5.17: Summary on estimated MAE error values on the test dataset for model instances, trained on the subsets made with different sampling strategies by picking top n samples

6 Conclusions

This work presented a faster AI-based solution for the problem of predicting the behaviour of thin-walled aluminum crash components during the simulation of an axial crash with numerical analysis on the subject of diminishing the computational expenses of framework training. The solution extends the ideas suggested by Kohar et al. in [6] as a potential replacement of FE simulations in application to crashworthiness testing. The framework was trained and tested on a dataset obtained via numerical modelling of the axial crash of the UWR4-like [18] extrusion profiles. The critical component of the described framework is the LSTM neural networks, used to predict the sequence of nodal coordinates during the test. The time required to obtain one simulation achieved roughly 5 seconds which is 286.8 times faster than the original 23 minutes 54 seconds required for LS-Dyna simulation. Considering that LS-Dyna simulation required 4 processors to run the simulation and one required for the AI framework, an actual factor of time reduction is 1,147.2. The time required for the neural networks to process a learning epoch was roughly 67 minutes, resulting in a total training time of approximately 110 hours. The error in predicted coordinates of nodes compared to the LS-Dyna simulation was 0.285 millimetres on average, which is 0.07 percent of the extrusion profile scale.

The approach for data processing for the framework was modified to reduce the computational time required to train the neural network and propose combining data sampling and models ensembling. As a baseline, the original framework's neural networks were trained to predict the evolution of the shift of the node during the experiment based on the feature representation of a node belonging to the LS-DYNA model of the profile using the whole training set of data consisting of 300 extrusion profiles. The network's performance was assessed by calculating MSE and MAE over the test subset, consisting of 60 extrusion profiles, reaching the values of 4.325 and 0.285 respectively after 100 training epochs.

Current work studied how the different sampling strategies affected the final metric values for the exact subset sizes and training time for the framework. Different subsets were constructed by: 1) randomly sampling 9600 nodes from the whole dataset; 2) randomly sampling 32 nodes from each LS-DYNA model; 3) employing the particular heatmap-based sampling procedure to select 32 nodes from each LS-DYNA model. The work reveals that training model instances on subsets of the same size result in the metrics values of the same order of magnitude – approximately two times higher than the baseline MSE and 5 times higher than the baseline

MAE – or 1.05 mm on average. However, the time required to process a single epoch was reduced to 15 seconds, with 150 epochs required for the framework to achieve optimal performance total training time was approximately 37.5 minutes, which is 175 times lower than for full data.

Further, the impact of combining five models trained on different subsets obtained with suggested sampling strategies into an ensemble was measured. The research showed that bagging allows reducing the value of the target metrics, reaching an average error of 0.825 / 0.824 / 0.811 millimetres on the test subset for random datawide / random modelwise / heatmap-based sampling, respectively. As the total time required to train the ensemble was 37.5 hours, which is significantly lower than the training time for the baseline model, the effectiveness of model ensembling for obtaining better results faster was confirmed. Using more subsets or more data for subsets could potentially increase the ensemble's performance, but that was out of the scope of this research.

To summarize, the AI framework was able to speed up simulation significantly. Additional research on speeding up the training process demonstrated a possible tradeoff between the drop in the accuracy of predictions and the speedup in the time required to train the framework. Thus, this research encourages the usage of preprocessing techniques in application to substituting numerical modelling with machine learning models since their reasonable use makes it possible to improve the numerical efficiency of algorithms used to solve such problems.

7 Recommendations for the future work

Current work demonstrates plenty of room for optimization of the performance of the ML-based method in the numerical simulation of crashworthiness. However, the current study still cannot be called exhaustive since the whole diversity of existing techniques to improve the performance of a given ML model cannot be encompassed in the scope of a single work. Possible directions for further research in the field of aiding the vehicle development process with AI may include

- *Implementation of more complex neural network types, capable of working with long-term time dependencies in data.* Although LSTM models show good performance on the sequential data, at the moment, their results in the field are far from the state-of-the-art. Replacing the essential part of the framework with the model more suitable for such a task may significantly increase the observed performance. E.g., transformer-based models [147], which predominate the field of natural language processing, may outperform LSTM-models due to their higher statistical capacity and the ability to consider correlations between the sequence elements directly.

- *Physics-based models.* As stated in [119], it is not always possible to brute-force the ML tasks in application to real-world physical processes by applying elaborate models to them, as such models often generalize poorly. The framework used in this work shows a similar behaviour, as its performance decreases on the test subset. A possible way to overcome this hardship is to use models directly incorporating the physics of the problem. Work [148] shows that physics-informed neural networks can reconstruct complicated dependencies within the data while having relatively simple architecture, thus, diminishing the chance for the model to overfit the training data.

- *Research on the effectivity of the other sampling techniques, which incorporate model response on the sample explicitly.* Heatmap sampling strategy, presented in the current work, estimates the importance of the nodes only using information obtained directly from data, following the general intuition on selecting nodes, useful for training. A heatmap, considering model outputs in the ways described in [110, 111, 112], may improve the procedure, shifting the training process to the samples, challenging from the point of view of the model, despite giving a computational overhead due to runtime importance reevaluation.

- *Synthesis of the RSM methodology with the suggested approach to run virtual experiments* Despite that RSM arguably allows for faster design space exploration compared to the presented approach, surrogate models used in it may not be complicated enough to build a good approximation of mapping between the parameter values and the target variables of interest. A combination of discovering the good initial point with RSM and fine-tuning the resulting design with the current approach may be promising for the efficient discovery of optimal design.

References

[1] Daniel A Lashof and Dilip R Ahuja. "Relative contributions of greenhouse gas emissions to global warming". In: *Nature* 344.6266 (1990), pp. 529–531.

[2] David L Greene, Howard H Baker Jr, and Steven E Plotkin. "Reducing greenhouse gas emissions from US transportation". In: (2010).

[3] Rajendra K Pachauri et al. *Climate change 2014: synthesis report. Contribution of Working Groups I, II and III to the fifth assessment report of the Intergovernmental Panel on Climate Change*. Ipcc, 2014.

[4] Hyung-Ju Kim, Gregory A Keoleian, and Steven J Skerlos. "Economic assessment of greenhouse gas emissions reduction by vehicle lightweighting using aluminum and high-strength steel". In: *Journal of Industrial Ecology* 15.1 (2011), pp. 64–80.

[5] Anders Kullgren et al. "Neck injuries in frontal impacts: influence of crash pulse characteristics on injury risk". In: *Accident Analysis & Prevention* 32.2 (2000), pp. 197–205.

[6] Christopher P Kohar et al. "Using Artificial Intelligence to Aid Vehicle Lightweighting in Crashworthiness with Aluminum". In: *MATEC Web of Conferences*. Vol. 326. EDP Sciences. 2020, p. 01006.

[7] Ping Zhu et al. "Use of support vector regression in structural optimization: Application to vehicle crashworthiness design". In: *Mathematics and Computers in Simulation* 86 (2012). The Seventh International Symposium on Neural Networks + The Conference on Modelling and Optimization of Structures, Processes and Systems, pp. 21–31. ISSN: 0378-4754.

[8] Neal Patel et al. "Crashworthiness Design Using Topology Optimization". In: *Journal of Mechanical Design - J MECH DESIGN* 131 (June 2009).

[9] Eric W Johnson, Jay B Brockman, and Rik Vigeland. "Sensitivity analysis of iterative design processes". In: *Proceedings of International Conference on Computer Aided Design*. IEEE. 1996, pp. 142–145.

[10] E.W. Johnson, L.A. Castillo, and J.B. Brockman. "Application of a Markov model to the measurement, simulation, and diagnosis of an iterative design process". In: *33rd Design Automation Conference Proceedings, 1996*. 1996, pp. 185–188.

[11] L. S. T. Coporation. *LS-DYNA Theory Manual*. Livermore, California: Livermore Software Technology Corporation, 2017.

[12] F. Martínez-Martínez et al. "A finite element-based machine learning approach for modeling the mechanical behavior of the breast tissues under compression in real-time". In: *Computers in Biology and Medicine* 90 (2017), pp. 116–124. ISSN: 0010-4825.

[13] Liang Liang et al. "A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis". In: *Journal of The Royal Society Interface* 15.138 (2018), p. 20170844.

[14] Fangshu Yang and Jianwei Ma. "Deep-learning inversion: A next-generation seismic velocity model building method". In: *Geophysics* 84.4 (2019), R583–R599.

[15] Christopher P. Kohar et al. "Development of high crush efficient, extrudable aluminium front rails for vehicle lightweighting". In: *International Journal of Impact Engineering* 95 (2016), pp. 17–34. ISSN: 0734-743X.

[16] Christopher P Kohar et al. "Effects of elastic–plastic behaviour on the axial crush response of square tubes". In: *Thin-Walled Structures* 93 (2015), pp. 64–87.

[17] Christopher P Kohar et al. "Effects of coupling anisotropic yield functions with the optimization process of extruded aluminum front rail geometries in crashworthiness". In: *International Journal of Solids and Structures* 128 (2017), pp. 174–198.

[18] Christopher Kohar. "Multi-scale Modeling and Optimization of Energy Absorption and Anisotropy in Aluminum Alloys". In: (2017).

[19] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), 1735–1780. ISSN: 0899-7667.

[20] François Chollet et al. *Keras*. 2015.

[21] Lawrence L Hershman. "The US new car assessment program (NCAP): Past, present and future". In: (2001).

[22] Roger P Daniel, Kenneth R Trosien, and Burgess O Young. *The Impact Behavior of the Hybrid II Dummy*. Tech. rep. SAE Technical Paper, 1975.

[23] Harry Singh et al. *Vehicle interior and restraints modeling development of full vehicle finite element model including vehicle interior and occupant restraints systems for occupant safety analysis using THOR dummies*. Tech. rep. 2018.

[24] J. M. Alexander. "An approximate analysis on the collapse of thin cylindrical shells under axial loading". In: *The Quarterly Journal of Mechanics and Applied Mathematics* 13.1 (Jan. 1960), pp. 10–15. ISSN: 0033-5614. eprint: https://academic.oup.com/qjmam/article-pdf/13/1/10/5413837/13-1-10.pdf.

[25] T. Wierzbicki and W. Abramowicz. "On the Crushing Mechanics of Thin-Walled Structures". In: *Journal of Applied Mechanics* 50 (1983), pp. 727–734.

[26] Shujuan Hou et al. "Design optimization of regular hexagonal thin-walled columns with crashworthiness criteria". In: *Finite Elements in Analysis and Design* 43.6 (2007), pp. 555–565. ISSN: 0168-874X.

[27] Wlodzimierz Abramowicz and Norman Jones. "Dynamic axial crushing of square tubes". In: *International Journal of Impact Engineering* 2.2 (1984), pp. 179–208. ISSN: 0734-743X.

[28] W. Abramowicz. "Thin-walled structures as impact energy absorbers". In: *Thin-Walled Structures* 41.2 (2003). Buckling strength and Failure Mechanics of Thin walled structures, pp. 91–107. ISSN: 0263-8231.

[29] Ahmad Baroutaji, Mustafa Sajjia, and Abdul-Ghani Olabi. "On the crashworthiness performance of thin-walled energy absorbers: recent advances and future developments". In: *Thin-Walled Structures* 118 (2017), pp. 137–163.

[30] Amir Zhumagulov. "Crashworthiness and Material Characterization of Multi-Cellular AA6063 Extrusions". MA thesis. University of Waterloo, 2017.

[31] Alan L Browne and Nancy L Johnson. "DYNAMIC CRUSH TESTS USING A "FREE-FLIGHT" DROP TOWER: THEORY". In: *Experimental Techniques* 26.5 (2002), pp. 43–46.

[32] Tomasz Wierzbicki. "Crushing analysis of metal honeycombs". In: *International Journal of Impact Engineering* 1.2 (1983), pp. 157–174.

[33] Wlodzimierz Abramowicz. "The effective crushing distance in axially compressed thin-walled metal columns". In: *International Journal of Impact Engineering* 1.3 (1983), pp. 309–317.

[34] Tomasz Wierzbicki and Wlodzimierz Abramowicz. "On the crushing mechanics of thin-walled structures". In: (1983).

[35] Wlodzimierz Abramowicz and Norman Jones. "Dynamic axial crushing of circular tubes". In: *International Journal of Impact Engineering* 2.3 (1984), pp. 263–281.

[36] Wlodzimierz Abramowicz and Norman Jones. “Dynamic axial crushing of square tubes”. In: *International Journal of Impact Engineering* 2.2 (1984), pp. 179–208.

[37] Wlodzimierz Abramowicz and Norman Jones. “Dynamic progressive buckling of circular and square tubes”. In: *International Journal of Impact Engineering* 4.4 (1986), pp. 243–270.

[38] Norman Jones and Tomasz Wierzbicki. *Structural crashworthiness*. Tech. rep. Butterworths London, 1983.

[39] Wlodek Abramowicz. “The macro element approach in crash calculations”. In: *Crashworthiness of transportation systems: structural impact and occupant protection*. Springer, 1997, pp. 291–320.

[40] Minoru Yamashita, Manabu Gotoh, and Yasuhiko Sawairi. “A numerical simulation of axial crushing of tubular strengthening structures with various hat-shaped cross-sections of various materials”. In: *Key Engineering Materials*. Vol. 233. Trans Tech Publ. 2003, pp. 193–198.

[41] Ali Najafi and Masoud Rais-Rohani. “Mechanics of axial plastic collapse in multi-cell, multi-corner crush tubes”. In: *Thin-Walled Structures* 49.1 (2011), pp. 1–12.

[42] P. Angeleri et al. “PAM-CRASH on the IBM 3090/VF: An integrated environment for crash analysis”. In: *IBM Systems Journal* 27.4 (1988), pp. 541–560.

[43] Michael Smith. *ABAQUS/Standard User’s Manual, Version 6.9*. English. United States: Dassault Systèmes Simulia Corp, 2009.

[44] T Wierzbicki et al. “Stress profiles in thin-walled prismatic columns subjected to crush loading-II. Bending”. In: *Computers and structures* 51.6 (1994), pp. 625–641.

[45] T Wierzbicki et al. “Stress profiles in thin-walled prismatic columns subjected to crush loading—I. Compression”. In: *Computers & Structures* 51.6 (1994), pp. 611–623.

[46] A Otubushin. “Detailed validation of a non-linear finite element code using dynamic axial crushing of a square tube”. In: *International Journal of Impact Engineering* 21.5 (1998), pp. 349–368.

[47] M. Langseth, O.S. Hopperstad, and A.G. Hanssen. “Crash behaviour of thin-walled aluminium members”. In: *Thin-Walled Structures* 32.1 (1998), pp. 127–150. ISSN: 0263-8231.

[48] M. Langseth, O.S. Hopperstad, and T. Berstad. "Crashworthiness of aluminium extrusions: validation of numerical simulation, effect of mass ratio and impact velocity". In: *International Journal of Impact Engineering* 22.9 (1999), pp. 829–854. ISSN: 0734-743X.

[49] BW Williams et al. "Effect of anisotropy, kinematic hardening, and strain-rate sensitivity on the predicted axial crush response of hydroformed aluminium alloy tubes". In: *International Journal of Impact Engineering* 37.6 (2010), pp. 652–661.

[50] Ø Fyllingena et al. "Brick versus shell elements in simulations of aluminium extrusions subjected to axial crushing". In: *7th European LS-DYNA conference, Salzburg*. 2009.

[51] Christopher P Kohar et al. "The effects of the yield surface curvature and anisotropy constants on the axial crush response of circular crush tubes". In: *Thin-Walled Structures* 106 (2016), pp. 28–50.

[52] Neal M Patel et al. "Crashworthiness design using topology optimization". In: *Journal of mechanical design* 131.6 (2009).

[53] Hongbing Fang et al. "A comparative study of metamodeling methods for multiobjective crashworthiness optimization". In: *Computers & structures* 83.25-26 (2005), pp. 2121–2136.

[54] Horace B Barlow. "Unsupervised learning". In: *Neural computation* 1.3 (1989), pp. 295–311.

[55] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[56] Christopher M Bishop. "Pattern recognition". In: *Machine learning* 128.9 (2006).

[57] Vladimir N Vapnik. "An overview of statistical learning theory". In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999.

[58] Emin Elmar oglu Mammadov. "Predictive Maintenance of Wind Generators based on AI Techniques". MA thesis. University of Waterloo, 2019.

[59] Tong Zhang. "Solving large scale linear prediction problems using stochastic gradient descent algorithms". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 116.

[60] S Rasoul Safavian and David Landgrebe. "A survey of decision tree classifier methodology". In: *IEEE transactions on systems, man, and cybernetics* 21.3 (1991), pp. 660–674.

[61] Yoav Freund, Robert E Schapire, et al. “Experiments with a new boosting algorithm”. In: *icml*. Vol. 96. Citeseer. 1996, pp. 148–156.

[62] John J Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[63] David J Livingstone, David T Manallack, and Igor V Tetko. “Data modelling with neural networks: advantages and limitations”. In: *Journal of computer-aided molecular design* 11.2 (1997), pp. 135–142.

[64] Ron Kohavi and David Wolpert. “Bias Plus Variance Decomposition for Zero-One Loss Functions”. In: (Sept. 1997).

[65] G. Cybenko. “Approximation by Superpositions of a Sigmoidal Function”. In: *Mathematics of Control, Signals, and Systems* 2 (1989).

[66] Daniel W Otter, Julian R Medina, and Jugal K Kalita. “A survey of the usages of deep learning for natural language processing”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[67] Martin Thoma. “A survey of semantic segmentation”. In: *arXiv preprint arXiv:1602.06541* (2016).

[68] Licheng Jiao et al. “A survey of deep learning-based object detection”. In: *IEEE Access* 7 (2019), pp. 128837–128868.

[69] Ian J Goodfellow et al. “Generative adversarial networks”. In: *arXiv preprint arXiv:1406.2661* (2014).

[70] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).

[71] Rewon Child et al. “Generating long sequences with sparse transformers”. In: *arXiv preprint arXiv:1904.10509* (2019).

[72] Chigozie Nwankpa et al. “Activation functions: Comparison of trends in practice and research for deep learning”. In: *arXiv preprint arXiv:1811.03378* (2018).

[73] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Icml*. 2010.

[74] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[75] D. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. “Learning internal representations by error propagation”. In: 1986.

[76] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[77] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[78] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.

[79] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]`.

[80] Federico Girosi, Michael Jones, and Tomaso Poggio. “Regularization theory and neural networks architectures”. In: *Neural computation* 7.2 (1995), pp. 219–269.

[81] Arthur E Hoerl and Robert W Kennard. “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1 (1970), pp. 55–67.

[82] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[83] Jean Duchon. “Splines minimizing rotation-invariant semi-norms in Sobolev spaces”. In: *Constructive theory of functions of several variables*. Springer, 1977, pp. 85–100.

[84] Tomaso Poggio and Federico Girosi. “Networks for approximation and learning”. In: *Proceedings of the IEEE* 78.9 (1990), pp. 1481–1497.

[85] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[86] Martin Sundermeyer, R. Schlüter, and H. Ney. “LSTM Neural Networks for Language Modeling”. In: *INTERSPEECH*. 2012.

[87] Mike Schuster and Kuldip Paliwal. “Bidirectional recurrent neural networks”. In: *Signal Processing, IEEE Transactions on* 45 (Dec. 1997), pp. 2673 –2681.

[88] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. *Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription*. 2012. arXiv: `1206.6392 [cs.LG]`.

[89] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[90] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*. PMLR. 2013, pp. 1310–1318.

[91] Jingzhao Zhang et al. "Why gradient clipping accelerates training: A theoretical justification for adaptivity". In: *arXiv preprint arXiv:1905.11881* (2019).

[92] Corentin Tallec and Yann Ollivier. "Unbiasing truncated backpropagation through time". In: *arXiv preprint arXiv:1705.08209* (2017).

[93] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization". In: *arXiv preprint arXiv:1409.2329* (2014).

[94] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. eprint: `https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf`.

[95] Yann Lecun. *PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)*. English (US). Universite P. et M. Curie (Paris 6), June 1987.

[96] Jonathan Masci et al. "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction". In: *ICANN*. 2011.

[97] Xibin DONG et al. "A survey on ensemble learning". English. In: *Frontiers of Computer Science* 14.2 (Apr. 2020), 241–258. ISSN: 2095-2228.

[98] Thomas G. Dietterich. "Ensemble Methods in Machine Learning". In: *Multiple Classifier Systems*. 2000.

[99] S Rasoul Safavian and David Landgrebe. "A survey of decision tree classifier methodology". In: *IEEE transactions on systems, man, and cybernetics* 21.3 (1991), pp. 660–674.

[100] L. Breiman. "Bagging predictors". In: *Machine Learning* 24 (1996), pp. 123–140.

[101] Peter Bühlmann, Bin Yu, et al. "Analyzing bagging". In: *The Annals of Statistics* 30.4 (2002), pp. 927–961.

[102] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[103] Meng Wang et al. "A Survey on Large-scale Machine Learning". In: *CoRR* abs/2008.03911 (2020). arXiv: `2008.03911`.

[104] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. "Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering". In: *ACM Trans. Knowl. Discov. Data* 3.1 (Mar. 2009). ISSN: 1556-4681.

[105] Benyamin Ghojogh et al. *Sampling Algorithms, from Survey Sampling to Monte Carlo Methods: Tutorial and Literature Review*. 2020. arXiv: 2011.00901 [stat.ME].

[106] Deanna Needell, Rachel Ward, and Nati Srebro. "Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz algorithm". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[107] Eric Moulines and Francis Bach. "Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011.

[108] Peter Richtárik and Martin Takáč. *On Optimal Probabilities in Stochastic Coordinate Descent Methods*. 2013. arXiv: 1310.3438 [stat.ML].

[109] Dominik Csiba and Peter Richtárik. "Importance Sampling for Minibatches". In: *Journal of Machine Learning Research* 19.27 (2018), pp. 1–21.

[110] Angelos Katharopoulos and François Fleuret. "Not All Samples Are Created Equal: Deep Learning with Importance Sampling". In: *CoRR* abs/1803.00942 (2018). arXiv: 1803.00942.

[111] Guillaume Alain et al. *Variance Reduction in SGD by Distributed Importance Sampling*. 2016. arXiv: 1511.06481 [stat.ML].

[112] Ilya Loshchilov and Frank Hutter. *Online Batch Selection for Faster Training of Neural Networks*. 2016. arXiv: 1511.06343 [cs.LG].

[113] Karianne J Bergen et al. "Machine learning for data-driven discovery in solid Earth geoscience". In: *Science* 363.6433 (2019).

[114] Željko Ivezić et al. *Statistics, data mining, and machine learning in astronomy: a practical Python guide for the analysis of survey data*. Vol. 1. Princeton University Press, 2014.

[115] William W Hsieh. *Machine learning methods in the environmental sciences: Neural networks and kernels*. Cambridge university press, 2009.

[116] J Nathan Kutz. "Deep learning in fluid dynamics". In: *Journal of Fluid Mechanics* 814 (2017), pp. 1–4.

[117] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees". In: *Machine learning* 63.1 (2006), pp. 3–42.

[118] Liang Liang et al. "A machine learning approach to investigate the relationship between shape features and numerically predicted risk of ascending aortic aneurysm". In: *Biomechanics and modeling in mechanobiology* 16.5 (2017), pp. 1519–1533.

[119] Anuj Karpatne et al. "Theory-guided data science: A new paradigm for scientific discovery from data". In: *IEEE Transactions on knowledge and data engineering* 29.10 (2017), pp. 2318–2331.

[120] Mark Alber et al. "Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences". In: *NPJ digital medicine* 2.1 (2019), pp. 1–11.

[121] Nathan Baker et al. *Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence*. Tech. rep. USDOE Office of Science (SC), Washington, DC (United States), 2019.

[122] Jared Willard et al. "Integrating physics-based modeling with machine learning: A survey". In: *arXiv preprint arXiv:2003.04919* (2020).

[123] Anuj Karpatne et al. "Physics-guided neural networks (pgnn): An application in lake temperature modeling". In: *arXiv preprint arXiv:1710.11431* (2017).

[124] Jimmy Forsberg and Larsgunnar Nilsson. "On polynomial response surfaces and Kriging for use in structural optimization of crashworthiness". In: *Structural and multidisciplinary optimization* 29.3 (2005), pp. 232–243.

[125] WJ Roux, Nielen Stander, and Raphael T Haftka. "Response surface approximations for structural optimization". In: *International journal for numerical methods in engineering* 42.3 (1998), pp. 517–534.

[126] Kai Liu, Duane Detwiler, and Andres Tovar. "Optimal design of nonlinear multimaterial structures for crashworthiness using cluster analysis". In: *Journal of Mechanical Design* 139.10 (2017).

[127] Hu Wang, GY Li, and Enying Li. "Time-based metamodeling technique for vehicle crashworthiness optimization". In: *Computer Methods in Applied Mechanics and Engineering* 199.37-40 (2010), pp. 2497–2509.

[128] H. Fang et al. "A comparative study of metamodeling methods for multiobjective crashworthiness optimization". In: *Computers & Structures* 83.25 (2005), pp. 2121–2136. ISSN: 0045-7949.

[129] Shahabedin Salehghaffari, Masoud Rais-Rohani, and Ali Najafi. "Analysis and optimization of externally stiffened crush tubes". In: *Thin-walled structures* 49.3 (2011), pp. 397–408.

[130] Hanfeng Yin et al. "Multiobjective crashworthiness optimization design of functionally graded foam-filled tapered tube based on dynamic ensemble metamodel". In: *Materials & Design* 55 (2014), pp. 747–757.

[131] Javad Marzbanrad and Mohammad Reza Ebrahimi. "Multi-objective optimization of aluminum hollow tubes for vehicle crash energy absorption using a genetic algorithm and neural networks". In: *Thin-Walled Structures* 49.12 (2011), pp. 1605–1615.

[132] M Shakeri, R Mirzaeifar, and S Salehghaffari. "New insights into the collapsing of cylindrical thin-walled tubes under axial impact load". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 221.8 (2007), pp. 869–885.

[133] Milad Abbasi et al. "A new approach for optimizing automotive crashworthiness: concurrent usage of ANFIS and Taguchi method". In: *Structural and Multidisciplinary Optimization* 49.3 (2014), pp. 485–499.

[134] Kai Liu et al. "Towards nonlinear multimaterial topology optimization using unsupervised machine learning and metamodel-based optimization". In: *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection. 2015.

[135] Kai Liu et al. "Optimal design of cellular material systems for crashworthiness". In: *SAE World Congress and Exhibition* (2016). ISSN: 0148-7191.

[136] Erdem Acar and Kiran Solanki. "Improving the accuracy of vehicle crashworthiness response predictions using an ensemble of metamodels". In: *International Journal of Crashworthiness* 14.1 (2009), pp. 49–61.

[137] Kai Liu et al. "Thin-walled compliant mechanism component design assisted by machine learning and multiple surrogates". In: *SAE World Congress and Exhibition* (2015). ISSN: 0148-7191.

[138] Jack PC Kleijnen. "Kriging metamodeling in simulation: A review". In: *European journal of operational research* 192.3 (2009), pp. 707–716.

[139] Erdem Acar, MURAT Altin, and Mehmet Ali Güler. "Evaluation of various multi-cell design concepts for crashworthiness design of thin-walled aluminum tubes". In: *Thin-Walled Structures* 142 (2019), pp. 227–235.

[140] Carl Edward Rasmussen and CK Williams. *Gaussian processes for machine learning, vol. 1*. 2006.

[141] Hu Wang, GY Li, and Enying Li. "Time-based metamodeling technique for vehicle crashworthiness optimization". In: *Computer Methods in Applied Mechanics and Engineering* 199.37-40 (2010), pp. 2497–2509.

[142] Johan AK Suykens and Joos Vandewalle. "Least squares support vector machine classifiers". In: *Neural processing letters* 9.3 (1999), pp. 293–300.

[143] Genki Yagawa and H Okuda. "Neural networks in computational mechanics". In: *Archives of Computational Methods in Engineering* 3.4 (1996), pp. 435–512.

[144] Felix Meister et al. "Towards Fast Biomechanical Modeling of Soft Tissue Using Neural Networks". In: (2018). arXiv: `1812.06186 [q-bio.QM]`.

[145] Cheng Chi, Gábor Janiga, and Dominique Thévenin. "On-the-fly artificial neural network for chemical kinetics in direct numerical simulations of premixed combustion". In: *Combustion and Flame* 226 (2021), pp. 467–477. ISSN: 0010-2180.

[146] Sam Greydanus, Misko Dzamba, and Jason Yosinski. "Hamiltonian Neural Networks". In: *CoRR* abs/1906.01563 (2019). arXiv: `1906.01563`.

[147] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[148] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.