Department of Construction Sciences
Solid Mechanics

ISRN LUTFD2/TFHF-23/5255-SE(1-31)

Machine learning based Topology Optimization

Master's Dissertation by

Lina Wihrén

Supervisors:
Gunnar Granlund, Division of Solid Mechanics
Hannes Bergkvist, LEVTEK
Erdzan Hodzic, RISE

Examiner:
Mathias Wallin, Division of Solid Mechanics

Copyright © 2023 by the Division of Solid Mechanics
and Lina Wihrén
Printed by Media-Tryck AB, Lund, Sweden
For information, address:
Division of Solid Mechanics, Lund University, Box 118, SE-221 00 Lund, Sweden
Webpage: www.solid.lth.se

Abstract This thesis aims to find a design suggestion for the chassis of a new micromobility vehicle developed by the company LEVTEK SWEDEN AB by using a multiple load, compliance based topology optimization. For this purpose, 9 static load cases are suggested, 4 of which have been derived from dynamic scenarios using an equivalent static load inspired approach based on free-body diagrams. The results from the topology optimization is presented as a design suggestion, but further post-processing is needed. Additionally, the extent of which machine learning could be applied for speeding up of the topology optimization was explored, and it was concluded to be feasible on a 2D cross section of the deck given the state-of-the-art and available resources. For this purpose a convolutional neural network proposed by Sosnovik & Oseledets (2017) was used, which demonstrated strong potential for learning a specific design domain, and it was investigated if the close-to-optimal solutions found by the network could be used as an initial guess for further topology optimization. It is concluded that transferability and consistency needs to be further investigated for this deep-learning approach.

Keywords Machine learning · Topology Optimization · Artificial neural networks · Micromobility vehicles

Preface

This thesis was a collaboration between the research institute RISE, the company LEVTEK SWEDEN AB, and the Division of Solid Mechanics departpent, LTH.

The computations for performing topology optimization was enabled by resources provided by LUNARC, and honarble mentions also go to Gunnar Granlund, Filip Sjövall and Jonathan Ebbesson.

I want to thank my examiner Mathias Wallin and the rest of the division for the support and for welcoming me to your department.

I want to thank Hannes Bergkvist, Jake Snowdon, Ivar Bergkvist, and the rest of LEVTEK for the opportunity to be a part of this project. Thank you for all the support, enthusiasm, and for letting me drive around on your prototypes during coffee-breakes.

I want to thank my supervisor Erdzan Hodzic for your confidence in me. Thank you for introducing me to this new exiting topic and for providing valuable guidance and support.

A special thanks to my main supervisor Gunnar Granlund. Thank you for always having insightful inputs and providing guidance (and tech support) for all my various questions and thoughts during this thesis.

Contents

1 Introduction **1**
1.1 Background . . . 1
1.2 The field of Machine learning based Topology Optimization . . . 1
1.3 Purpose and goal . . . 2

2 Theoretical background **2**
2.1 Topology Optimization . . . 2
2.1.1 Solid Isotropic Material with Penalization (SIMP) . . . 3
2.1.2 Filters and projections . . . 3
2.1.3 Finite Element Formulation . . . 4
2.2 Artificial Intelligence and Machine Learning . . . 5
2.2.1 Autoencoders (AE) . . . 6
2.2.2 Convolutional Neural Network (CNN) . . . 7
2.2.3 Regularization . . . 8

3 Methodology **9**
3.1 Topology Optimization . . . 9
3.1.1 Model set-up . . . 10
3.1.2 Load cases . . . 10
3.1.3 Optimization task . . . 15
3.2 Machine Learning based Topology Optimization . . . 15
3.2.1 Data . . . 16
3.2.2 Neural Network . . . 18

4 Results **18**
4.1 Topology Optimization . . . 18
4.2 Machine Learning based Topology Optimization . . . 20
4.2.1 Performance on the shear case data-set . . . 20
4.2.2 Performance on new cases . . . 23

5 Discussion **27**
5.1 Topology Optimization . . . 27
5.2 Machine learning based Topology Optimization . . . 28

6 Conclusion **30**

References **31**

A Mesh **i**

B More results from neural network **i**

1 Introduction

1.1 Background

LEVTEK SWEDEN AB was founded based on a vision to realize a new type of micromobility vehicle that is both accessible and suitable for every day use. This electrical vehicle on 4 wheels aims to be both portable and light while maintaining stability, safety, and being suitable for additional load transport. The unique design of the chassis however poses new challenges for a design that is both light and durable.

Topology optimization is the process of finding an optimal arrangement of material given a certain design domain and boundary conditions. This is crucial for smaller vehicles that want to minimize the use of material while not compromising on performance. This approach to design has already been used for similar purposes, Xiao et al. (2012) for example employed topology optimization when finding the optimal shape of a frame for an electric bicycle.

1.2 The field of Machine learning based Topology Optimization

Topology optimization (TO) is an approach to design that has been around for decades (Bendsoe & Sigmund 2004). A more recent approach is to apply machine learning (ML) in this field. The aim is often to aid or reduce expensive calculations and evaluations during the classical iterative approach of TO, or in some cases completely replace the classical TO process with a model that can instantly predict a design given the problem formulation. Motivations for ensuring manufacturability and exploring different design options from an aesthetic motivation also exist. An overview of the state-of-the art is presented by Woldseth et al. (2022), where many of the articles within the field of ML and TO are summarized and discussed. Woldseth et al. (2022) suggests that the approaches of applying ML to TO has dominantly been by the use of Artificial Neural Networks (ANN), and further that the approaches often can be classified as falling in one of five main ideas, *Direct design, Acceleration, Post-processing, Reduction,* and *Design diversity*. Below follows a summary of Woldseth et al. (2022) literary review to get a feeling for some of the main motivations and challenges of the field so far. For a more thorough overview of the field, I refer to the original article.

The most popular approach to date has been within the category *direct design*, which is an image based learning with the goal of a model that can perform an instant design prediction given the problem formulation. In order to achieve this, the model is usually trained in a supervised environment with classically optimized structures as the target. Just like most models over all categories, a mesh is often used to represent the structure, and models will often be more or less mesh dependent. So far the *direct design* approach has resulted in models that require a lot of training data obtained by performing standard TO in order to perform well in very limited design spaces. Even then, structural disconnects or other non physical solutions are common, most likely due to lack of evaluation of structural performance as feedback to the model. An example of this approach is Yu et al. (2019) where a convolutional neural network (CNN) and generative adversarial network (GAN) were suggested, and where one also can observe the effect of disconnections in structure.

Acceleration instead aims not to replace the classical TO process, but rather achieve acceleration in computational time required. One approach to achieve this is by training a model to perform approximations of finite element analyses and the cost function which are often the most expensive computational part of the iterative TO algorithms. A CNN was suggested by Lee et al. (2020) for this purpose. Another approach would be to skip iterations in an approach similar to the idea behind *direct design*, but instead map structures to skip only a subset of iterations instead of trying to replace the whole iterative process. An example of this is the model proposed by Kallioras & Lagaros (2021) where a model was trained to predict a close-to-converged structure based on the element wise fluctuation of density during the first iterations. Models within the *acceleration* category are typically cheaper to train compared to models within *direct design*, and we see more attempts of models that are trained to evaluate smaller subsets of the structure resulting in models that can generalize better, due to less mesh dependency. The use of classical TO along side the ANN models also results in more reliable structures.

A *post-processing* approach differs in that it aims to modify an already optimized structure. An optimized structure is rarely fit for production directly, due to some suggested features being hard to produce. This makes it desirable to train a model to e.g. make boundaries smoother or match holes in the structure with the best replacement predefined shape. Other approaches that falls in the *post-processing*

category is to train a model to translate a structure onto a finer mesh than the one which the optimization was performed on, leading to possible speed up since it make evaluations during TO cheaper. The overall benefits here is that we start with an already optimized structure. So far, good results has been seen even on simple models within this category.

The *reduction* approach is less explored, but the idea is to train a network to perform reparameterization. If one manages to define fewer parameters that describes the main features of the design, we could focus our optimization with regards to this reduced space in order to achieve faster convergence for an otherwise unchanged TO algorithm, an idea explored by Guo et al. (2018) where a variational autoencoder (VAE) was trained for this purpose. Reparameterization has also been studied with the intention of achieving a grid free representation of a structure for which our final shape would not be restricted to a specific mesh, see e.g. Chandrasekhar & Suresh (2021). The last approach, *design diversity*, explores different ways to try to maximize design options that all meet the structural performance standards, so that the final modification can be made with regards to manufacturability and/or aesthetics.

Some main observations made by considering the different approaches from all the categories summarized above, as emphasized by Woldseth et al. (2022), is that the best results are achieved when a model is trained with, or the design process at some point considers feedback from finite element analysis or alternative evaluations of structural performance. This is in contrast to purely image based approaches. The prediction is thus that the most successful approaches would fall in the categories of *acceleration, post-processing,* and *reduction*. Another important observation is also that most of the models within the field so far have only been explored on a limited scale, in small design spaces, and with limited variation in boundary conditions and loads etc. This means that both the true potential of the models and general approaches suggested, along with their limitations, remains to be further investigated.

1.3 Purpose and goal

The main goal of the thesis is to propose a design recommendation for a new type of micromobility vehicle using topology optimization. The design space considered will be based on information provided by the company developing the product. The aim is additionally to find a way to effectively apply ML to aid the TO process for the purpose of evaluating this new field of research and its applications within the industry. Here, *effectively* refers to a reduction in computational time or iterations needed to achieve an optimal design that is structurally sound.

2 Theoretical background

2.1 Topology Optimization

Topology optimization (TO) is a structural design approach based on mathematically formulating the problem at hand such that we can optimize structural performance over the chosen design parameters (Christensen & Klarbring 2010). A popular choice of such design parameters are density values on a finite element mesh, where the values can range from 0 to 1. Representing the structure with continuous variables allows application of optimization algorithms based on gradient descent, but at the end we want to obtain a structure described by discrete values that are either 0 or 1. Common choices of structure characteristics to optimize is to minimize compliance or total weight. Additionally to the cost function itself we can formulate additional constraints e.g. maximum stress allowed in the structure, avoid certain eigenfrequencies , etc. Below follows an example for how such a formulation would look for a general case:

$$\mathbb{TO}\begin{cases}\min\limits_{\boldsymbol{z}} c(\boldsymbol{z},\boldsymbol{u}(\boldsymbol{z}))\\ \text{subject to:}\begin{cases}\text{design constraints on } \boldsymbol{z}\\ \text{equilibrium constraint}\\ \text{additional constraint (e.g. weight, etc)}\end{cases}\end{cases} \tag{1}$$

where $\boldsymbol{z}$ is a vector with our design variables and $c(\boldsymbol{z},\boldsymbol{u}(\boldsymbol{z}))$ is the cost function which we, in this case, want to minimize. This function could describe the total compliance of the structure subjected to

a specific load. Under *subject to* we add other constraints which we want to be fulfilled, e.g. maximum total weight of the structure, stress constraints etc. (Christensen & Klarbring 2010)

2.1.1 Solid Isotropic Material with Penalization (SIMP)

SIMP is a method used in order to encourage convergence to a binary solution even when intermediate densities are allowed. This is done by introducing an exponent q, that scales the actual density ρ of the design variable. ρ^q is then used to scale the material parameter Young's Modulus E, creating a relative elasticity used for evaluating the structures performance. A higher value of E means higher stiffness. Since the density is a value between 0 and 1, an intermediate solution is punished by disproportionately affecting the Young's Modulus. Since stiffness becomes more expensive for intermediate densities, an optimal solution should be binary. The scaling between the relative Young's modulus and the density value for each element for $q = 1$ and the commonly used $q = 3$ is shown in figure 1 below.

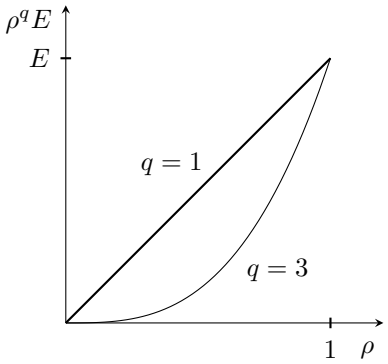


Figure 1: The relative Young's Modulus E as a function of density ρ for different values of penalization factors q

Another important part of the SIMP model is to avoid density values of $\rho = 0$. This is in order to avoid singularity issues and maintain solvable finite element equations even when varying the densities. This restriction is often formulated by introducing a small value ϵ and letting $\epsilon \leq \rho \leq 1$ (Christensen & Klarbring 2010)

2.1.2 Filters and projections

One challenge with TO is that a true optimal solution rarely exists, and the problems are therefore ill-posed. By limiting the design domain to a finite element mesh this is somewhat addressed but results are mesh-dependent and solutions prone to numerical instability. Additional regularization methods should therefore be introduced in order to ensure convergence to a feasible topology as well as allowing for some additional control over structure features which reduces the mesh-dependency. The SIMP method specifically tends to result in *checkerboard patterns*, illustrated in figure 2, oscillating behavior or convexity problems that leads to convergence to a local instead of global minima. (Christensen & Klarbring 2010)

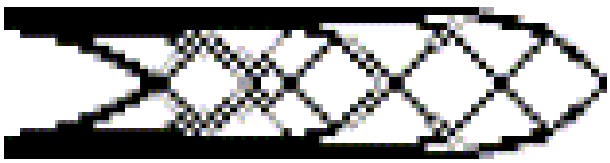


Figure 2: Example by Bruns & Tortorelli (2001) of the checkerboard effect

A regularization tool that addresses both the mesh-dependency problem as well as the occurrence of checkerboard patterns is a blurring filter as introduced by Bruns & Tortorelli (2001). First, a radius r

and a distance dependent weight ω is specified. This dependence could be linear, Gaussian, etc. Now a new filtered density value can be calculated by

$$\eta(z) = \frac{\sum_i \omega_i z_i}{\sum_i \omega_i}$$

for all z_i within proximity r from z. For resolving the checkerboard problem, r must be chosen to a length scale twice that of the element size given the mesh. Even higher values of r can be motivated as a counter to mesh dependency and ensuring manufacturability by introducing a minimum length scale and thus eliminating finer features.

A problem that arises is that density filters prevent convergence to a binary structure due to its blurring effect, and there will always be a transition of intermediate densities between solid and void. As a solution to this problem Guest et al. (2004) suggested the use of a Heaviside projection in combination with a density filter. The Heaviside projection serves as a smooth, differentiable, step-function, that, depending on a value β and a given threshold value, goes from a linear projection $\beta = 0$ to approaching a true step function when $\beta \to \infty$. Guest et al. (2004) suggests for the value of β to continuously increase through the iterative process of TO. This avoids convergence to local minima but ensures a close to binary convergence at the end if tuned correctly. A Heaviside projection h combined with the use of a density filter η results in an effective density vector $\tilde{\boldsymbol{z}}$

$$\tilde{\boldsymbol{z}} = h(\eta(\boldsymbol{z}))$$

for which the finite element analysis is performed.

2.1.3 Finite Element Formulation

As discussed, TO often relies on the idea of the domain being divided into finite design variables. This also allows us to find numerical solutions to the equations describing the TO problem. The finite element (FE) method is the numerical approach of solving differential equations by dividing the region where the differential equations apply into smaller sections, called *elements*. This creates a discrete description of the environment, a *mesh*, compared to a continuous one, and the original problem can now be solved in an approximate manner (Ottosen & Petersson 1992).

The differential equation for mechanical equilibrium of a static case is

$$\begin{cases} \nabla \boldsymbol{\sigma} = 0 & \text{in } \Omega \\ \boldsymbol{u} = u_0 & \text{on } \partial\Omega_u \\ \boldsymbol{t} = t_0 & \text{on } \partial\Omega_t \end{cases} \tag{2}$$

for a quasi-static case where body forces are neglected. Ω is the domain and $\partial\Omega_u$ the boundary where displacement restrictions u_0 applies, and $\partial\Omega_t$ the boundary where initial traction t_0 is applied. $\boldsymbol{\sigma}$ is the Cauchy stress tensor that for a linear elastic case is given by Hooke's law

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\epsilon} \tag{3}$$

where $\boldsymbol{\epsilon}$ is the strain which has a kinematic relation to the nodal displacement $\boldsymbol{u}$. $\boldsymbol{D}$ is dependent on material parameters Young's modulus E and Poisson's ratio v (Ottosen & Petersson 1992), and in the case of SIMP it is also scaled proportional to ρ^q (Christensen & Klarbring 2010). These equations are used for the construction of the FE-formulation of the equilibrium constraint:

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{F} \tag{4}$$

where $\boldsymbol{K}$ is the stiffness matrix and $\boldsymbol{F}$ is the vector of forces acting on the structure. Due to the densities ρ being updated during each iteration of TO, $\boldsymbol{K}$ has to be reconstructed and equation 4 has to be re-solved.

2.2 Artificial Intelligence and Machine Learning

The field of Artificial intelligence (AI) stems from the idea that machines, or computers, could display intelligent behavior (McCarthy 2007). Computers proved to be a very powerful tool for handling complex mathematical and logical statements, but some problems are hard to define formally and thus harder for the computer to handle. For example, identifying a cat in an image is an easy task for a human but much harder for a computer. As one can imagine there is a very diverse set of problems that can fall into this category of being hard, or even impossible to define or solve, making AI a diverse field. (Goodfellow et al. 2016). Machine learning (ML), which is a subcategory to the field of AI, more specifically encompasses a computers ability to learn from data. A common approach to ML is deep learning, which encompasses the use of Artificial Neural Networks (ANN). The idea and structure of a ANN are directly inspired by neurons, the building block of the human brain. The equation governing the artificial neuron is given by

$$y = \varphi(b + \sum_{k=1}^{K} \omega_k x_k) \tag{5}$$

where y i the output, x the set of input of size K, each with a corresponding weight ω, b is a bias, and φ is an activation function. Two common examples of activation functions are found in figure 3. ReLU and Sigmoid are especially popular for the TO application since they map values > 0, which is appropriate if the purpose is classification of density values. An ANN is constructed by these artificial neurons which are connect in complex networks, creating a non trivial, non linear, mapping of an input layer to an output layer through whats commonly referred to as the hidden layers. The most complex mappings arise from the use of many of these layers, and where the number of layers are called the depth of the network. This is what the word *deep learning* refers to. The way in which these hidden layers are designed is called the architecture of the ANN. In order for the ANN to learn we need to provide data and define a *loss function*. Our loss function will depend on if we use supervised or unsupervised learning. During supervised learning, there is a predefined right or wrong output for each input, while during unsupervised learning it is not known beforehand exactly what information from the data we want to extract or map, and there needs to be another way to formulate and measure performance of the ANN. The learning is defined as an optimization task, where the goal is to minimize the loss function by finding the optimal values of the weights (ω) in the network that maps the input to a desired output. (Ohlsson & Edén 2022)

This brief introduction summaries the basic idea behind the Artificial Neural Network. There are many different ways to create an ANN, by varying activation functions, architecture, introducing regularization, choice of optimization algorithm, etc. Below follows some more detailed descriptions of the specific types of ANNs referred to during this thesis.

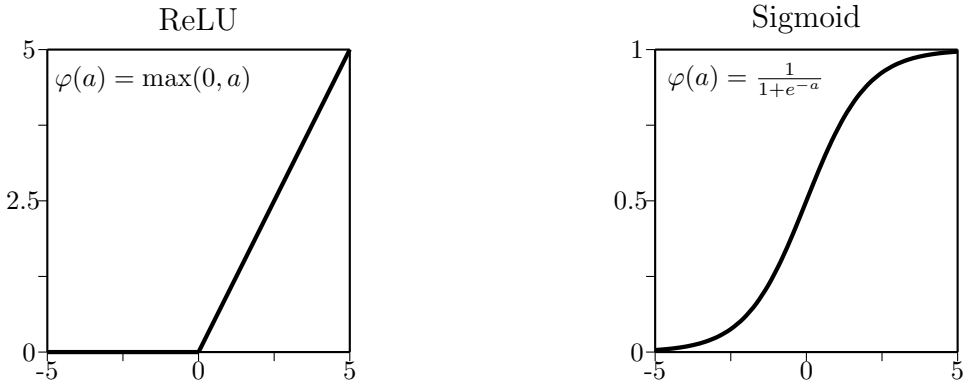


Figure 3: Two of the most widely used activation functions $\varphi(a)$ ReLU (Rectified linear unit) and Sigmoid (logistic function).

2.2.1 Autoencoders (AE)

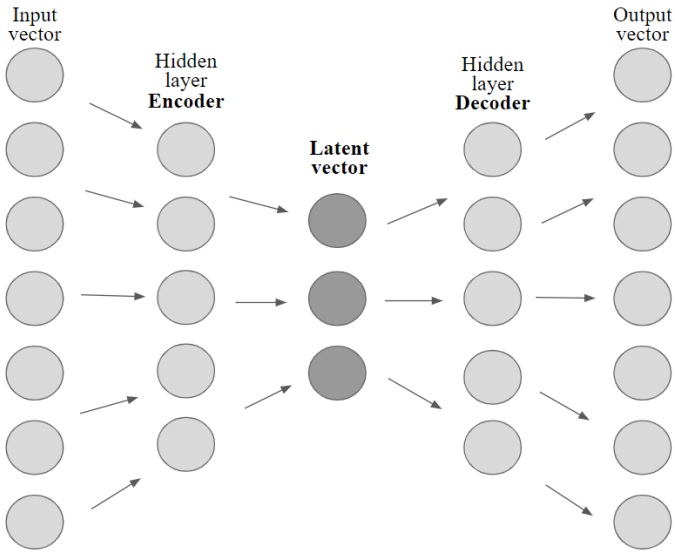


Figure 4: An example of a symmetrical auto encoder (AE) with 3 hidden layers. Arrows indicate direction (feed forward), but each node is fully connected to each node in a neighboring layer.

An autoencoder (AE), see figure 4, is a type of neural network for unsupervised learning. It is often symmetrical, where the hidden layers first gradually reduce in size compared to the input vector until it reaches whats referred to as the *latent space*, the smallest hidden layer in the network. After this the hidden layers increase in size again, until the output vector which is of the same dimension as the input vector. The different parts of the network on either side of the latent vector are often referred to as the encoder and decoder respectively. The loss function is typically defined as how similar the output is to the input, meaning that a successfully trained network should be able to recreate the input as accurately as possible. In order to minimize the loss function, the network needs to learn how to encode the information that is the input and represent it using only the number of parameters limited by the size of the latent vector, in such a way that the decoder can use this latent vector to extract the full resolution of the original information. This ability makes AE's a powerful tool for dimensionality reduction and feature detecting (Goodfellow et al. 2016).

2.2.2 Convolutional Neural Network (CNN)

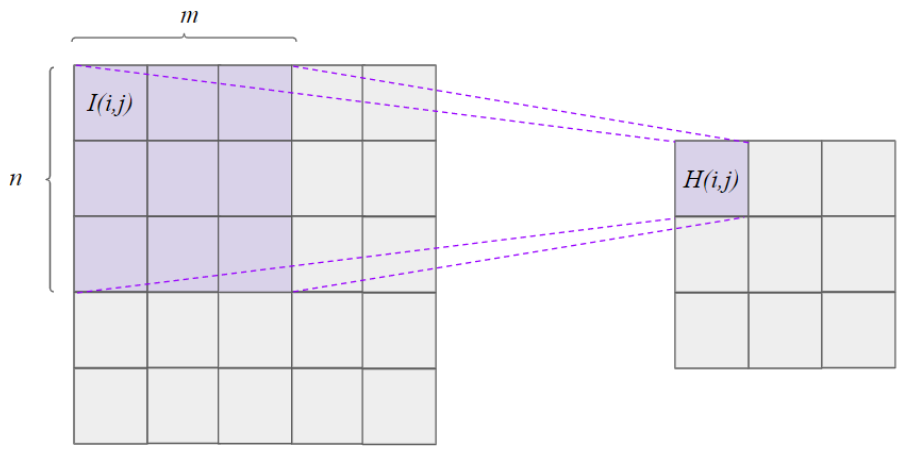


Figure 5: Example of a filter in a CNN, in this case $m, n = 3$. Input image I is to the left and the filtered image H to the right.

Convolutional neural networks (CNN) are specifically designed for data where spacial relations are of importance, making these networks useful for image analysis related tasks (Ohlsson & Edén 2022). They can be designed to process both 2D and 3D data. The key word of the CNN is *convolution*, which, since input images often are discrete (consisting of pixels), can be written as a sum. The full operation is called a filter:

$$H(i,j) = \varphi(b + \sum_{m,n} I(i+m, j+n)K(m,n)) \tag{6}$$

where I is the input image, in this example a 2 dimensional matrix, b is a bias, φ is the activation function, and $H(i,j)$ the output image. K is a *kernel*, in this case with m x n individually trainable variables. A visualization of equation 6 is found in figure 5. One kernel sweeps across an input image I, creating a new filtered image H that, depending on tunable parameters (padding and stride), can have the same or a different resolution. This gives the network the property of spatial feature detection that is transitionally invariant. (Ohlsson & Edén 2022)

Figure 5 is an example of a filter where no padding and stride=1 was used. As can be seen this results in a smaller filtered image H compared to I. By adding a *padding*, an additional 'frame' of pixels around the input image, the filtered image would keep the dimensions of the input. Two variants are zero- and copy-padding, where the added pixels either have the value zero, or copies the value of the nearest pixel. Stride refers to how big the 'steps' of the filter are when sweeping across the input image and thus also affects the dimensions of the filtered image (Ohlsson & Edén 2022). For example a filter with stride 2 would skip to analyze input $I(i+1, j)$ and go directly to $I(i+2, j)$ after $I(i, j)$. If padding=2 in both dimensions was used in the example in figure 5, the filtered image would be of size 2 x 2.

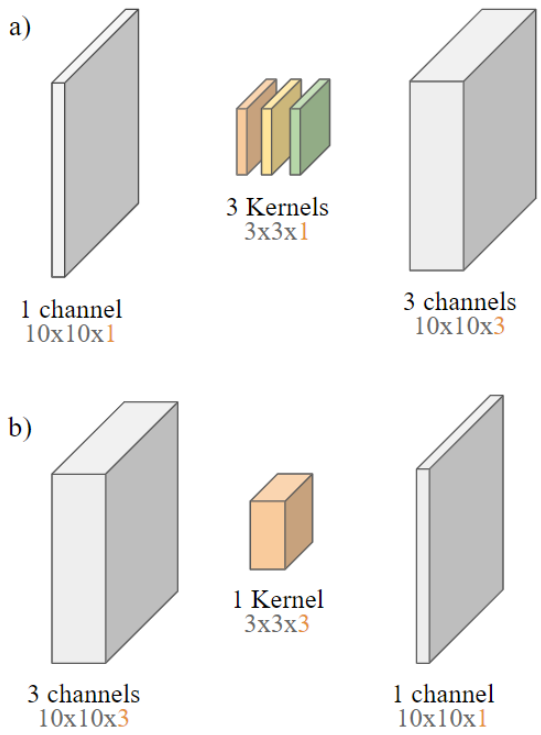


Figure 6: Example of a CNN layer with a 10x10 image that demonstrates the effect kernels have on channels. In a) 3 kernels are applied to create a filtered 3 channel image, while in b) one single kernel with *width* 3 is applied to a 3 channel image to create a filtered 1 channel image

Another important concept is a *channel*. We can apply multiple different filters to one image for the purpose of detecting multiple features. The individual filters will all produce an additional channel in the output image for that layer. A kernel also has the same channel depth as the input image, adding a dimension of trainable parameters. For the case of a multiple channel input equation 6 would instead be written as

$$H(i,j) = \varphi(b + \sum_{c} \sum_{m,n} I_c(i+m, j+n) K(m,n,c)) \tag{7}$$

where c is the channels of the layer (Ohlsson & Edén 2022).

In addition to trainable convolutional filters, it is common to include deterministic filters in a CNN, called *pooling* layers. Common pooling filters include max pooling (equation 8) and the average pooling, where the kernel return either the max value or the average value.

$$H(i,j) = \max_{m.n}(I(i+m, j+n)) \tag{8}$$

Combined with a large stride, this creates a filtered image that is smaller than the input but keeps key features (Ohlsson & Edén 2022). Contrary to a downsizing pooling filter is up-sampling, where the input image deterministically return a filtered image that is larger than the input.

2.2.3 Regularization

A central concept in ML is *overfitting*, and it describes the state in which a model has minimized the loss in regards to the data which it was trained on to an extent that it has failed to learn the underlying trends of that data. In order for a ANN to be a useful tool it should perform well on previously unseen but related data. A standard practice is to divide the available data into *training* and *validation*. This means that the parameters of the network are optimized for the training data-set, but later evaluated with

regards to the validation data-set. This favors models that has learned how to best *predict* on previously unseen data. A network which perform well on previously unseen data is said to generalize well. There are many different strategies that can be employed as means of regularization to avoid overfitting. One regularization tool is called dropout. Dropout means that during training there is a certain probability that a node will simply be left out for that particular pattern. This prevents the network from relying on a specific path or nodes becoming 'co-dependent' which is often the case for neural networks that don't generalize well. Good generalization is a constant ideal that is impossible to measure, but can be approximated and encouraged with some of the mentioned techniques. (Ohlsson & Edén 2022)

3 Methodology

In order to explore methods of applying ML to TO for the purpose of finding a design for a micromobility vehicle chassis, a base case of an optimized structure needs to be achieved first. This also provides a design suggestions for the chassis which is one of the main goals of the thesis. Given the time investment needed for this, only a smaller investigation of ML's ability to speed up TO on a limited design domain related to the chassis was conducted.

3.1 Topology Optimization

For topology optimization, a straight forward case considers linear static analysis. Introducing additional complexity such as non linearity and dynamic analysis results both in much more costly analysis and model set-up (Park 2010). For the scope of this thesis, the linear static approach was chosen. A vehicle chassis is however exposed to critical scenarios that are dynamic in nature and with no true static equivalence. An approach of solving this trade-off of cost and accuracy is to derive an Equivalent Static Load (ESL) for the non linear static case, in this case linear dynamic. Park (2010) defines ESL as the multiplication of the linear stiffness matrix and the displacement field from the non linear static analysis. In other words, one finds the static loads that would result in the same displacement fields as the dynamic analysis and uses these in place of the dynamic analysis in the optimization. Xiao et al. (2012) discusses the ESL approach in their work where they use multiple-load TO to find a design for a bike frame. Due to difficulty in directly applying the ESL approach, they motivate their static loads by actual measurements of forces following dynamic simulations and real-life validation. This approach is more versatile but due to the less strict definition it leaves some uncertainty in how to best model the measured dynamic loads. One of these approaches is to take the peak load during a certain time frame for the corresponding static case. The benefits with this approach is its simplicity, and is thus suitable for the purpose of this thesis. No dynamic analysis and validation from real-life trials of the vehicle was available at the given time. Estimations of peak-loads using available data about the vehicle was instead used to create static loads for key dynamic cases using free-body diagrams. Multiple load cases were considered for different dynamic and static cases, all of which were included in the cost function. The CAD geometry used for the design and analysis domain was provided by LEVTEK, and Abaqus was used for simulations and TO.

3.1.1 Model set-up

The CAD provided by LEVTEK makes up the design and analysis domain and is shown in figure 7.

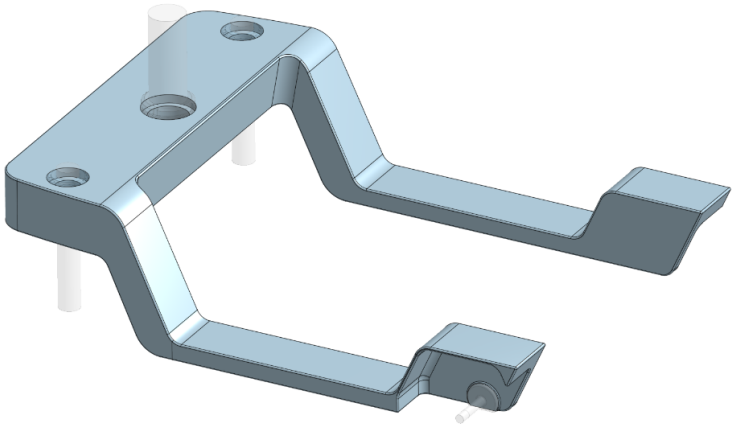


Figure 7: The chassis model which outlines the design domain of the TO-task. The opaque parts (Handle, 2 attachments for front wheels, 2 attachments for back wheels) are separate parts of the vehicle and outside of the design domain but included in the analysis domain.

In the model set-up in Abaqus all materials were defined as isotropic and linear elastic. The chassis was specified to be produced out of aluminum while the separate 5 parts (Handle, 2 attachments for front wheels, 2 attachments for back wheels) where assigned the material steel. The corresponding parameters as assigned in Abaqus can be found in table 1.

Table 1: Material parameters

	Young's Modulus (N/m^2)	Poisson ratio
Aluminum	$70 \cdot 10^9$	0.33
Steel	$200 \cdot 10^9$	0.30

The contact properties between the separate parts were initialized as rough in the tangential direction and non-separable/hard in the normal direction, meaning that no relative displacement can occur in any direction at the surface of contact between the parts. For meshing, the Tet shape was applied and the chassis was modeled using 465 490 separate elements. The remaining 5 separate parts were meshed independently and using a much coarser mesh. Figure A.1 displays the final mesh and can be found in Appendix.

3.1.2 Load cases

9 load cases have been suggested with the purpose representing both common situations the chassis will be exposed to (e.g. a person standing on the deck), along with worst-case scenarios to ensure robustness. The final load cases considered are presented below and can be divided into two main categories, static and dynamic. All load cases are evaluated statically for the purpose of the optimization. ESL inspired load cases has been derived for the scenarios that are dynamic. First the calculations performed as basis for the load cases is presented before the final loads are summarized in table 3. After that the load cases themselves are presented in figure 8 and 9

Table 2: Vehicle specifications provided by LEVTEK

Vehicle weight	15 kg	
Maximum load	200 kg	
Top speed	45 km/h	$\rightarrow$ 12.5 m/s
Acceleration	0-25 km/h, 2 s	$\rightarrow$ 3.5 m/s^2
Deceleration (brake)	25-0 km/h, 2 s	$\rightarrow$ 3.5 m/s^2

To create the load cases, 6 loads where specified and will be referred to as F_{Stand}, $F_{Stand-backplate}$, F_{Brake}, F_{Turn}, $F_{Bump-front}$, and $F_{Bump-back}$. The weight of a human used as reference was found by taking the average between the average weight of a man (84 kg) and a woman (68 kg) in Sweden, giving the value $m_{human} = 76$kg. Using this number along with the information provided in table 2 the loads can be calculated as follows:

$$|F_{Stand}| = |F_{Stand-backplate}| = m_{human} \cdot g = 76.0\text{kg} \cdot 10.0\text{m/s}^2 = 760.0\text{N}$$

$$|F_{Brake}| = (m_{human} + m_{vehicle}) \cdot a_{deceleration} \cdot F.S = (76.0 + 15.0)\text{kg} \cdot 3.5\text{m/s}^2 \cdot 3.0 = 955.5\text{N} \approx 1000.0\text{N}$$

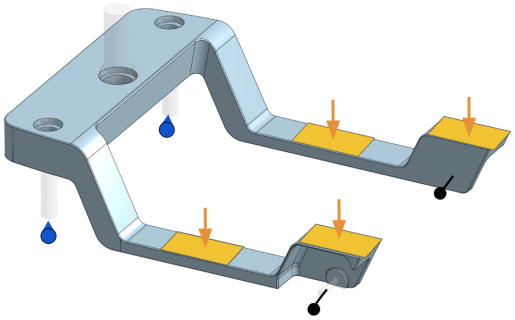
$$|F_{Turn}| = (m_{human} + m_{vehicle}) \cdot a_{acceleration} \cdot F.S = (76.0 + 15.0)\text{kg} \cdot 3.5\text{m/s}^2 \cdot 3.0 = 955.5\text{N} \approx 1000.0\text{N}$$

where $F.S$ is a factor of safety for the dynamic case to ensure robustness. $F.S = 3.0$ was chosen because it provides a good margin for a possible worst-case scenario while also resulting in a force similar in magnitude to $|F_{Stand}|$. This helps ensure that the effect of this specific load has an impact on the final design from the topology optimization task. Other simplifications include the assumption that the vehicle accelerate at full capacity perpendicular to normal direction when turning, again this is to consider a worst-case scenario. Lastly, both of the F_{Bump} loads are motivated in the same way: the vehicle is driving straight and hits an obstacle with one of the wheels (front or back), resulting in a force with a positive y- and z-component $(0, \left|F^y_{Bump}\right|, |F^z_{Bump}|)$. $\left|F^y_{Bump}\right|$ was chosen to have the same magnitude as both $|F_{Turn}|$and $|F_{Brake}|$, while $|F^z_{Brake}|$ was chosen as $4 \cdot |F^y_{Brake}|$, where the factor of 4 is motivated by the finding of Dattakumar & Ganeshan (2017). They performed a dynamic analysis of a car chassis and derived a ESL for the load case *pothole*. The ratio between the corresponding y- and z-component of the ESL and peak-loads can be found in the results of their thesis.

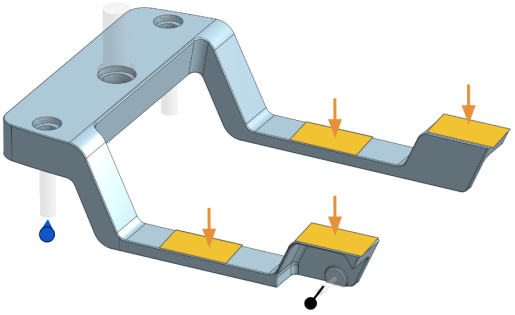
As a side-note, since the model is linear elastic the magnitude of the loads have no direct impact on the topology optimization results, but instead their relative magnitude will be crucial. Realistic magnitudes is however still valuable to consider and try to model, not only for comprehension and communication purposes, but also in order to be able to use these load cases for analysis in other stages of product evaluation or post processing of the optimization results.

Table 3: Summary of the loads used. Including the name used for reference, a description of type of load (pressure or point), its placement and magnitude.

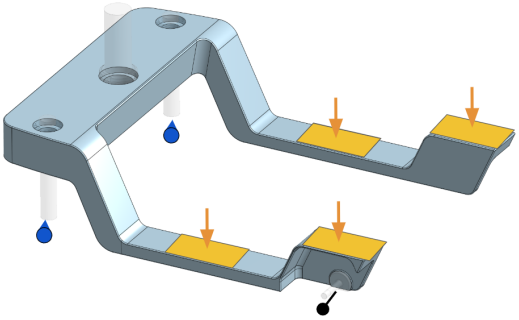
Load	Description	Magnitude (N)
F_{Stand}	Evenly distributed pressure, force in the normal direction. Placement is in the middle of the deck along the y-axis. Magnitude is the sum of the two equal pressure loads.	760
$F_{Stand-backplate}$	Evenly distributed pressure, force in the normal direction. Placement is full area of both plates on top of the back-wheels. Magnitude is the sum of the two equal pressure loads.	760
F_{Brake}	Point loads applied in positive y-direction where back wheels are attached. Magnitude is the sum of the two equal point loads.	(0,1000,0)
F_{Turn}	Point loads applied in positive x-direction where back-wheels are located. Magnitude is the sum of the two equal point loads.	(1000,0,0)
$F_{Bump-front}$	A point load applied in both positive y- and z- direction. Located at the attachment of the left front wheel.	(0,1000,4000)
$F_{Bump-back}$	A point load applied in both positive y- and z-direction. Located at the attachment of the left back wheel.	(0,1000,4000)



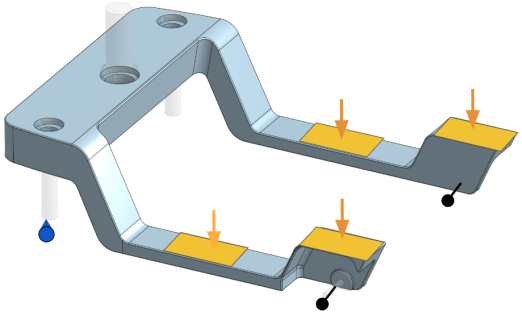
(a) Simple
Loads: F_{Stand}, $F_{Stand-backplate}$
BC: front-wheels locked in z-direction, pinned back-wheels



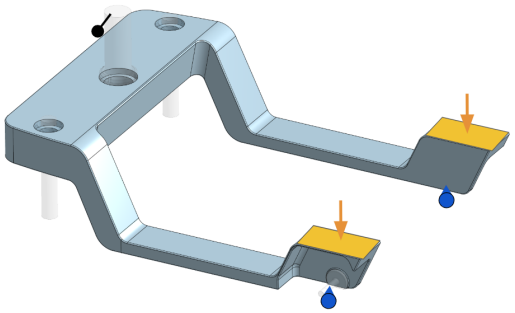
(b) Half-contact
Loads: F_{Stand}, $F_{Stand-backplate}$
BC: left front-wheel locked in z-direction, left back-wheel pinned



(c) ¾ contact-a
Loads: F_{Stand}, $F_{Stand-backplate}$
BC: front-wheels locked in z-direction, left back-wheel pinned



(d) ¾ contact-b
Loads: F_{Stand}, $F_{Stand-backplate}$
BC: left front-wheel locked in z-direction, pinned back-wheels



(e) Backplate
Loads: $F_{Stand-backplate}$
BC: back-wheels locked in z-direction, handle pinned

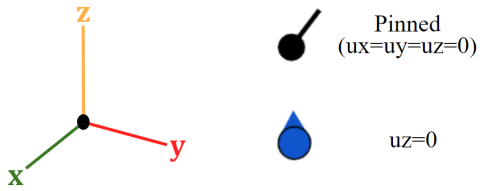
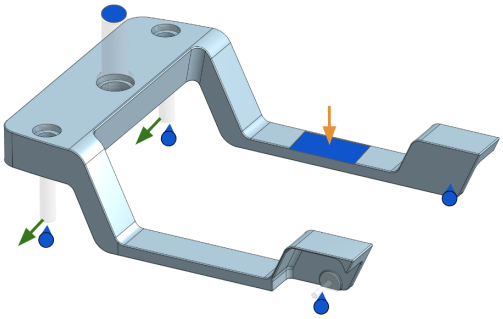
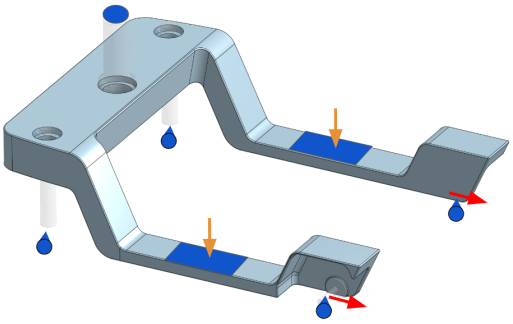


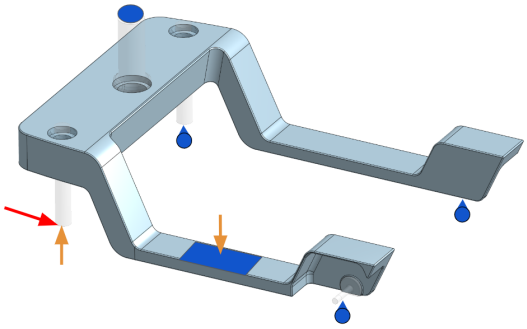
Figure 8: All static load cases displayed together with a name of reference and summary of loads and boundary conditions (BC) for each case. The colors of the arrows are to emphasize direction of the force applied.



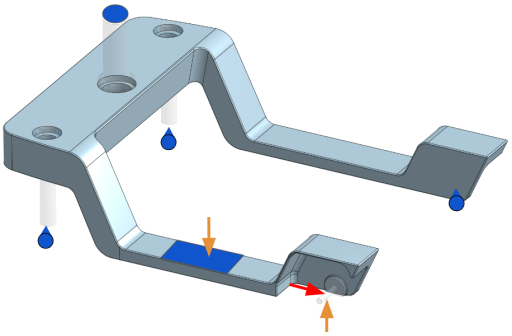
(a) Turn
Loads: F_{Turn}, F_{Stand*}
BC: all wheels locked in z-direction. Handle and area where load F_{Stand*} is applied are locked in x- and y-direction



(b) Brake
Loads: F_{Brake}, F_{Stand}
BC: all wheels locked in z-direction. Handle and area where load F_{Stand} is applied are locked in x- and y-direction



(c) Bump - front wheel
Loads: $F_{Bump-front}$, F_{Stand*}
BC: all wheels (except front left) locked in z-direction. Handle and area where load F_{Stand*} is applied are locked in x- and y-direction



(d) Bump - back wheel
Loads: $F_{Bump-back}$, F_{Stand*}
BC: all wheels (except left back) locked in z-direction. Handle and area where F_{Stand*} is applied are locked in x- and y-direction

Figure 9: All dynamic load cases displayed together with a name of reference and summary of loads and boundary conditions (BC) for each case. F_{Stand*} emphasizes change to area for load application compared to description in table 3. The colors of the arrows are to emphasize direction of the force applied. Coordinates of reference and symbol description are found in figure 8. Areas marked in blue indicates displacement restriction where $u_x = u_y = 0$

A further motivation behind some of the boundary conditions is that the wheels have been locked in the z-direction where they are assumed to be in contact with the ground. In the static cases the back wheels have been 'pinned' (restricted displacement in every direction) to ensure that a solution exists. For the dynamic cases however it is an unrealistic assumption that any wheels are 'pinned' since the vehicle is moving. Instead the displacement restriction in the z-plane was applied where an approximate center of mass would be found, the location where the force of the person standing was applied.

3.1.3 Optimization task

The optimization task $\mathbb{T}$ formulated in Abaqus is summarized below:

$$\mathbb{T}\begin{cases}\min\limits_{z} \sum_i w_i \cdot W(\boldsymbol{z}, \boldsymbol{u}_i(\boldsymbol{z})) \\ \text{s.t.} \begin{cases} V(\boldsymbol{z})/V_0 < f \\ \boldsymbol{K}\boldsymbol{u}(\boldsymbol{z}) - \boldsymbol{F} = 0 \\ 0.001 \leq z_e \leq 1, \forall z_e \in \boldsymbol{z} \\ z_e = 1, \forall z_e \in S \end{cases}\end{cases} \tag{9}$$

where w_i is the weight in the cost function for each load case i. $W = \frac{1}{2}\boldsymbol{u}_i^T \boldsymbol{K}\boldsymbol{u_i}$ is the strain energy, dependent on the displacement for each load case $\boldsymbol{u}_i(\boldsymbol{z})$. S is the set of elements connected to the the upper surface of the deck and plates above the back wheels, along with all surfaces in contact with external parts. The elements in set S have a fixed density of 1. $V(\boldsymbol{z})$ is the current volume of the design dependent on the design variables, V_0 is the total volume of the design domain and f the volume fraction, in this case $f = 0.3$. The final value of the volume fraction was chosen based on what encouraged an interesting topology during TO.

The weights w were chosen to 1 for each load case, meaning that no load case had a disproportional impact on the cost function in relation to the total strain energy for each load case. The one exception to this rule is $w_{Backplate} = 500$. This choice was made in order to ensure structure supporting the upper plate above the back wheels. Due to the load case *Backplate* having minimal displacement and stress in other parts of the structure, the value of $w_{Backplate}$ is not expected to affect the overall topology additional to compensating for the lack of material placed in the part supporting the surface above the back wheels occurring when $w_{Backplate} = 1$. Experimentation by varying w for other load cases could be interesting to explore further, but the final cost function gave sufficient results.

Convergence delta criteria was defined as 0.001 for the objective function and 0.005 for the element density. Density update strategy was set to normal, with the maximum update per element and design cycle being 0.25. SIMP with penalty 3 was applied.

3.2 Machine Learning based Topology Optimization

Due to limitations in digital storage and computational access, an implementation of machine learning based topology optimization on the full chassis was discarded. This approach can however still be used for the purpose of product development, and the approach that was chosen was to study the cross section of the deck (standing portion) of the vehicle, specifically for the case of torsion. The cross section can be elongated to a 3D structure using an additional script to create the design of a complete deck, see figure 10. This could provide a good compliment given that the final load cases don't capture the torsion effect, even though it could be an important feature. It would however require a different assembly process than casting the full chassis as one piece. This approach is not disregarded. The idea is thus to generate data using topology optimization on a diversity of shear cases. The neural network trained on this data could later be used to accelerate the iterative process of topology optimization once the final boundary conditions and torsion loads are initialized.

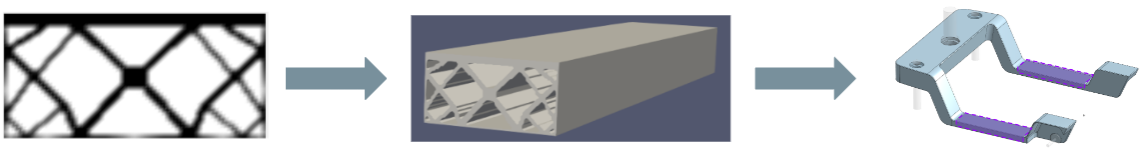


Figure 10: Visualization of how a 2D cross section could be used for the generation of a 3D beam and where on the chassis it would be constructed.

3.2.1 Data

Data samples 1-12 000 (11 999) were generated and used for training and evaluation of the performance. One data point consists of a tensor (40, 80, 500) and contains the design domain (40 x 80 grid) for each case up to 500 iterations. The topology optimization task $\mathbb{D}$ is specified below:

$$\mathbb{D} \begin{cases} \min\limits_{z} W(\boldsymbol{z}, \boldsymbol{u}(\boldsymbol{z})) \\ \text{s.t.} \begin{cases} \delta \leq z_e \leq 1 \\ V(\boldsymbol{z})/V_0 = f \\ \boldsymbol{K}\boldsymbol{u}(\boldsymbol{z}) - \boldsymbol{F} = 0 \end{cases} \end{cases} \tag{10}$$

where $\boldsymbol{F} = \boldsymbol{F_1} + \boldsymbol{F_2} + \boldsymbol{F_3} + \boldsymbol{F_4}$ in figure 11. Fixed design variables included the centered 8 x 8 elements and the top 4 x 80 elements (to ensure a that a person can stand on the deck). The centered elements were also locked for displacement, providing a boundary condition.

Volume fraction f was varied for each case according to the normal distribution N(0.4,0.1), but with a minimum fraction of 0.01. A filter was also used and varied according to $(2+|N(0,2)|)\cdot l_e$ to introduce even more variation in the data, l_e being the length scale of a single element. $L_x = 2$ and $L_y = 1$ is the width and height of the design domain. The loads were applied according to a shear case, a force acting on the surface of the design domain perpendicular to the normal. This was applied on all 4 borders of the domain. The loads where also applied centered with a standard length of $0.6 \cdot \min(L_x, L_y) = 0.6 \cdot 1 = 0.6$. Variance was introduced by uniformly varying the endpoints of the force up to 20% of $\min(L_x, L_y)$ independent from each other. Additionally the loads were varied uniformly $\pm 30\%$ in magnitude independently from each other. The forces applied along the same axis where applied in opposite directions.

Using these design parameters (volume constraint, filter radius), load cases, and boundary conditions, the design was optimized for a maximum of 500 iterations. After 50 iterations a Heaviside projection was introduced. If the structure converged before iteration 500, the final design was simply copied to ensure consistency in dimensions. A final threshold was applied on the converged structure to ensure a binary final design as a target. Randomized samples of the targets of the data-set is displayed in figure 12.

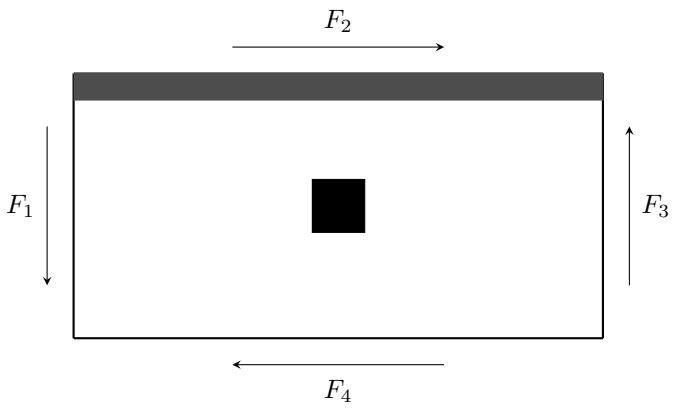


Figure 11: design domain of the shear case data-set. Black indicates displacement restriction and fixed element density of 1. Grey indicates fixed element density of 1.

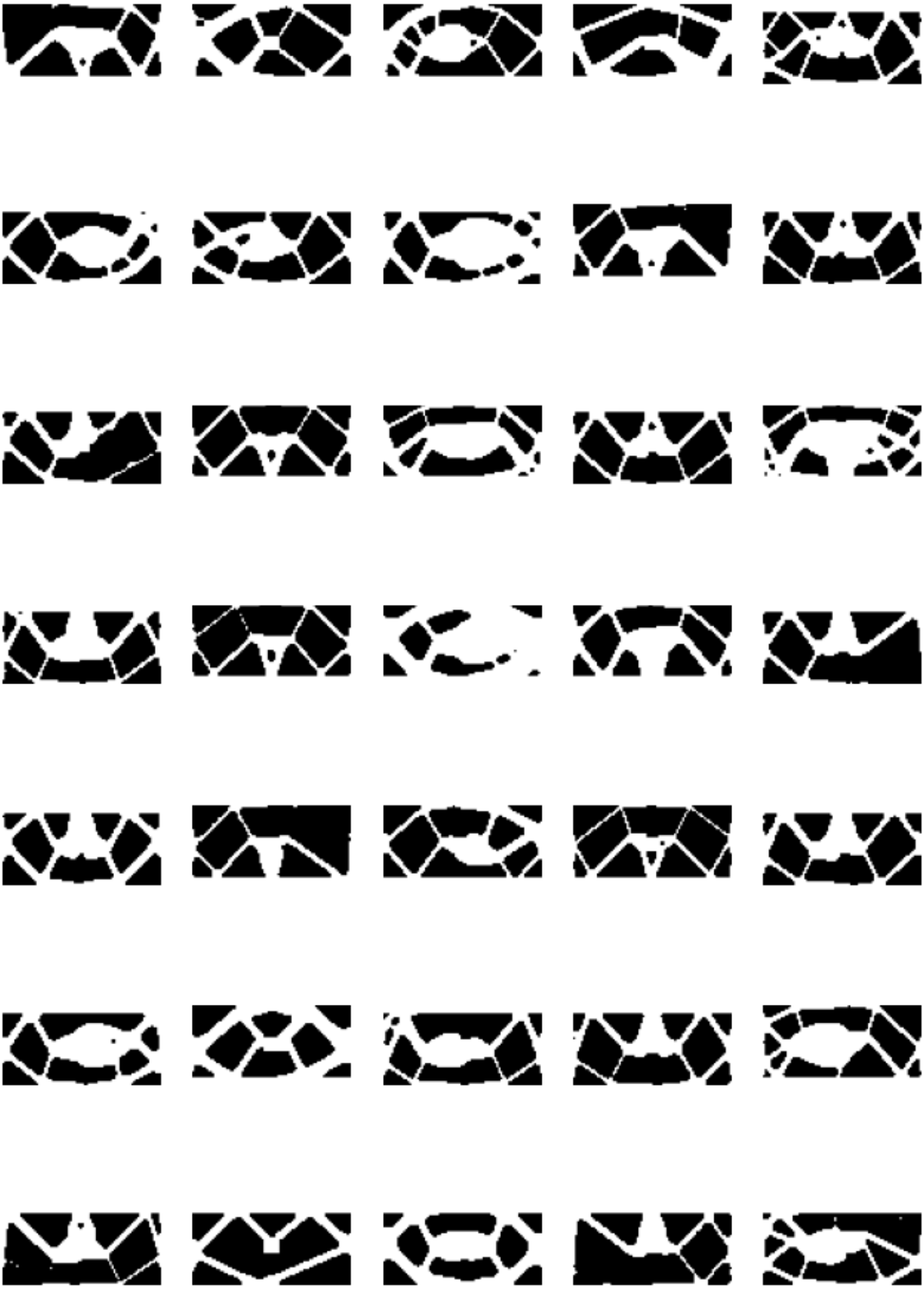


Figure 12: Randomized examples of the shear case data-set. Displayed is the converged structure after the final threshold filter was applied. White indicated structure and black indicate void.

3.2.2 Neural Network

The architecture of the artificial neural network chosen for the purpose of this thesis was the one proposed by Sosnovik & Oseledets (2017). Very little alterations were done to the code, as both hyper parameters and and architecture were kept unchanged. However, small alterations such as introduction of additional parameters in training.py, where input_size was replaced by input_height and input_width in order to allow training on data on where $l_{height} \neq l_{width}$. This also required the 90° rotation feature in random_d4_translation in data_utils.py to be disabled, resulting in a decrease in variation of input data compared to the original code.

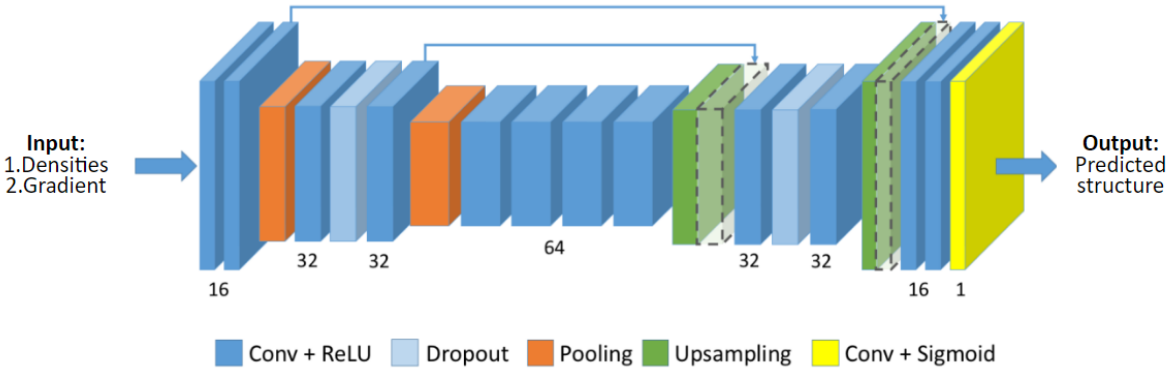


Figure 13: The architecture of the neural network used. The number under each layer indicated the number of channels. The opaque boxes represent concatenation of that current layer, and as indicated by the small blue arrow, a previous layer. The figure is (modified) from Sosnovik & Oseledets (2017)

The neural network, presented in figure 13 takes two pictures as input, the current density state after initial optimization for n iterations, as well as a gradient defined as the density wise difference between iteration n and $n-1$. The network is fully convolutional and the kernels used are of size 3 x 3. The activation function used is ReLU except for the final layer where a pixel-wise classification using Sigmoid is applied for the output image. The network is not an AE since it does not seek to reconstruct the input images, but it does adopt some of the architecture that is commonly associated with an AE, namely an encoder and decoder. This serves the purpose of identifying important features, and helps prevent small inconsistencies that could be devastating for the performance of a final structure. To prevent a loss of detail however, concatenation layers are introduced before the pooling step. This type of architecture was inspired by the U-Net introduced by Ronneberger et al. (2015). Max-pooling is applied with a kernel size of 2 x 2. See Sosnovik & Oseledets (2017) for further details.

4 Results

4.1 Topology Optimization

Using the design domain, load cases, and optimization tasks described in section 3.1. The TO task converged after 38 design updates. The final result is in the form of the density field $\boldsymbol{z}$, displayed in figure 14. Due to the design not being binary there needs to be a threshold introduced in order to visualize a final design. Displayed in figure 15 is the visualized result where elements of density 0.3 and higher are included as material.

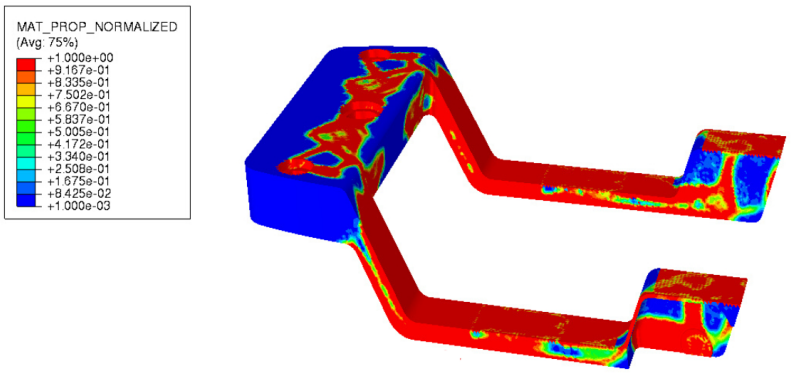


Figure 14: Visualization of the raw data result from TO. MAT_PROP_NORMALIZED indicates element density values.

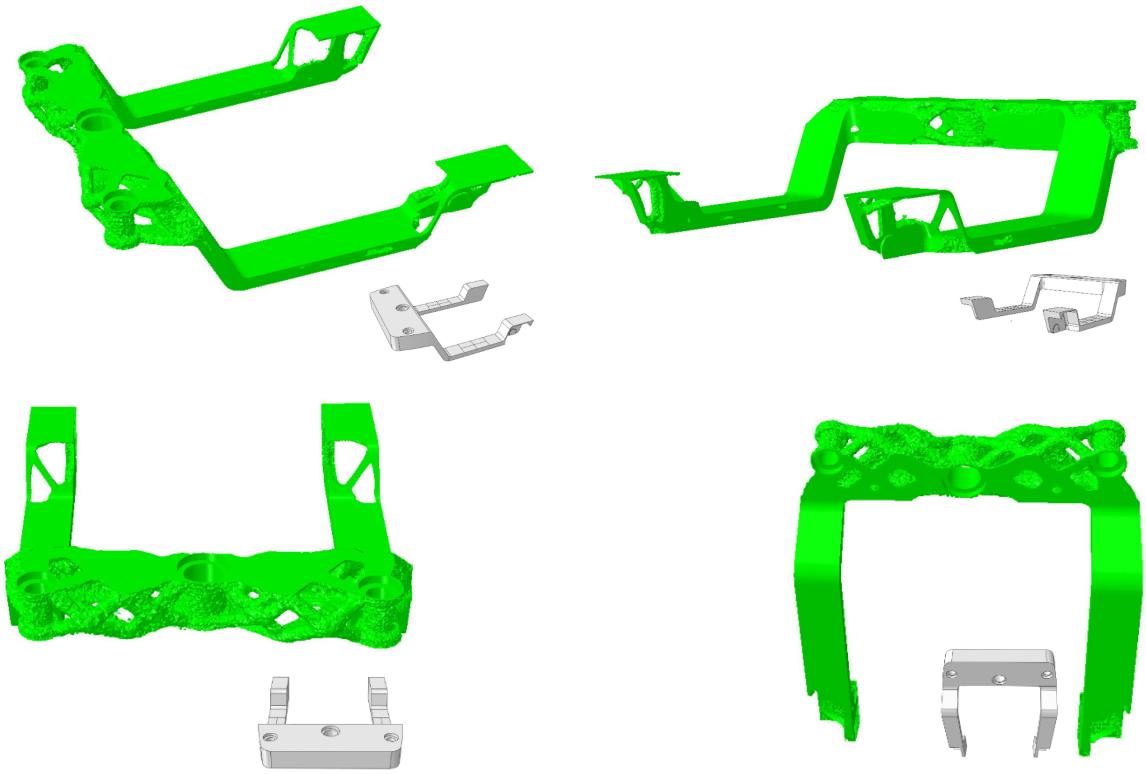


Figure 15: Results from the topology optimization task presented in section 3.1 . The cut-off value for the displayed densities are 0.3. Each sub-figure contains a reference of the original chassis viewed from the same angle.

4.2 Machine Learning based Topology Optimization

4.2.1 Performance on the shear case data-set

The network proposed by Sosnovik & Oseledets (2017) was trained with standard parameters and unchanged architecture but with the shear case data-set. Performance is measured in terms of binary accuracy when comparing the topology predicted by the network and the actual converged structure. Several networks where trained using different distributions for sampling iteration-index of the input. Their performance are plotted as a function over the index of the iteration used as input during evaluation. These results are presented in figure 16. For a visual evaluation of the networks performance on the training data set some examples are presented in the figures below (17-20), where the input images are displayed along with the network prediction and converged TO-result.

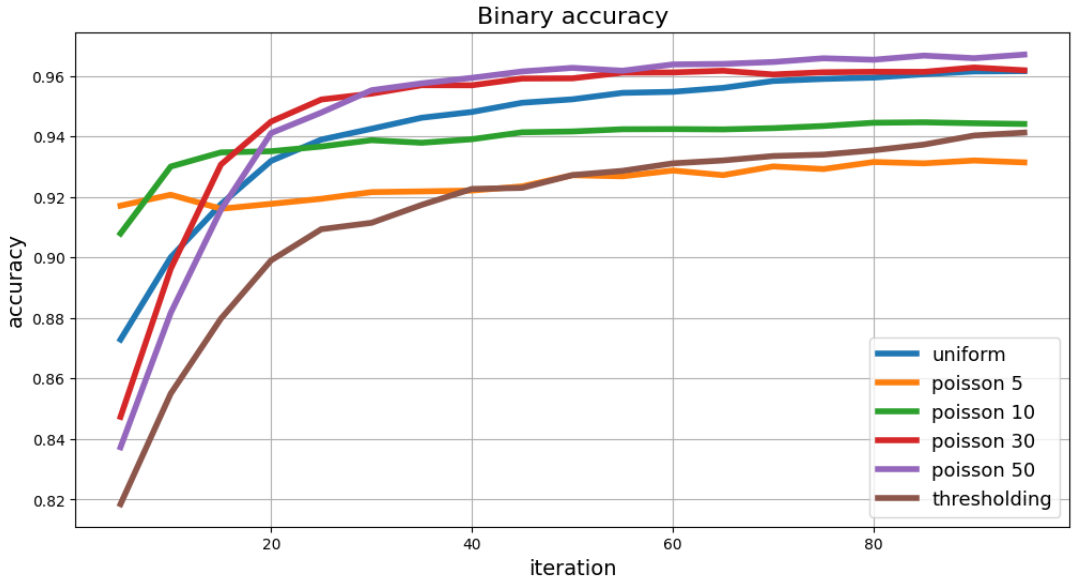


Figure 16: Binary accuracy for different networks trained using different sampling strategies (uniform, Poisson 5, Poisson 10, Poisson 30, Poisson 50), displayed as a function over the TO-iteration that was used for input. The performance is measured on the training data-set. Thresholding as a strategy is included as reference. The code used for performance analysis and graph is attributed Sosnovik & Oseledets (2017).

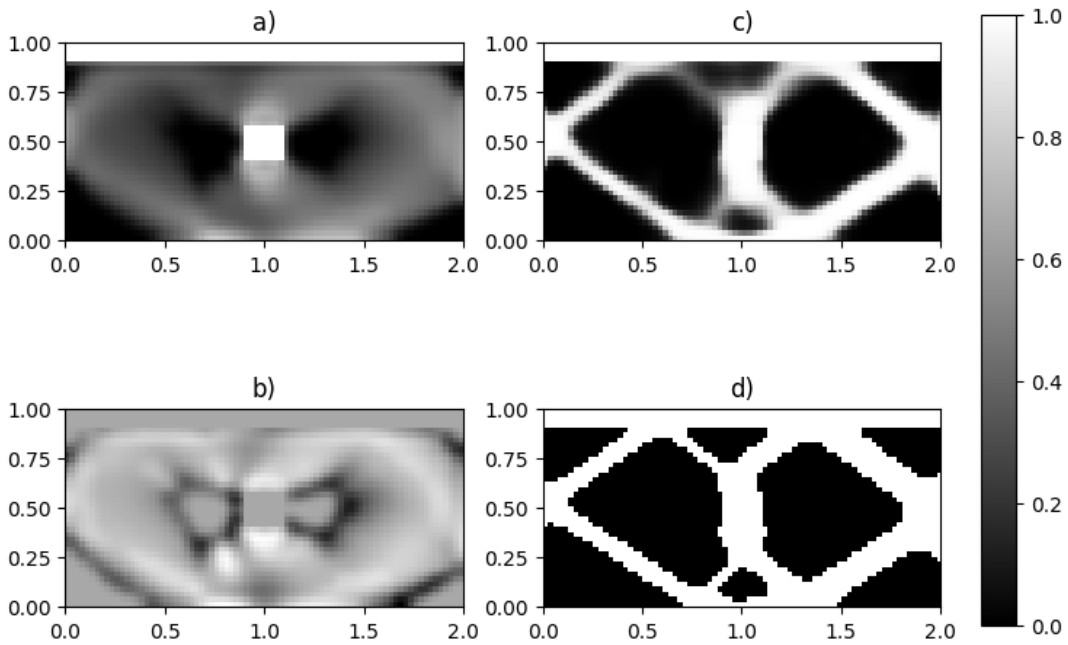


Figure 17: Results on the shear data-set used for training, using sampling with distribution $P(5)$. The input displayed in a) is the element density after 5 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

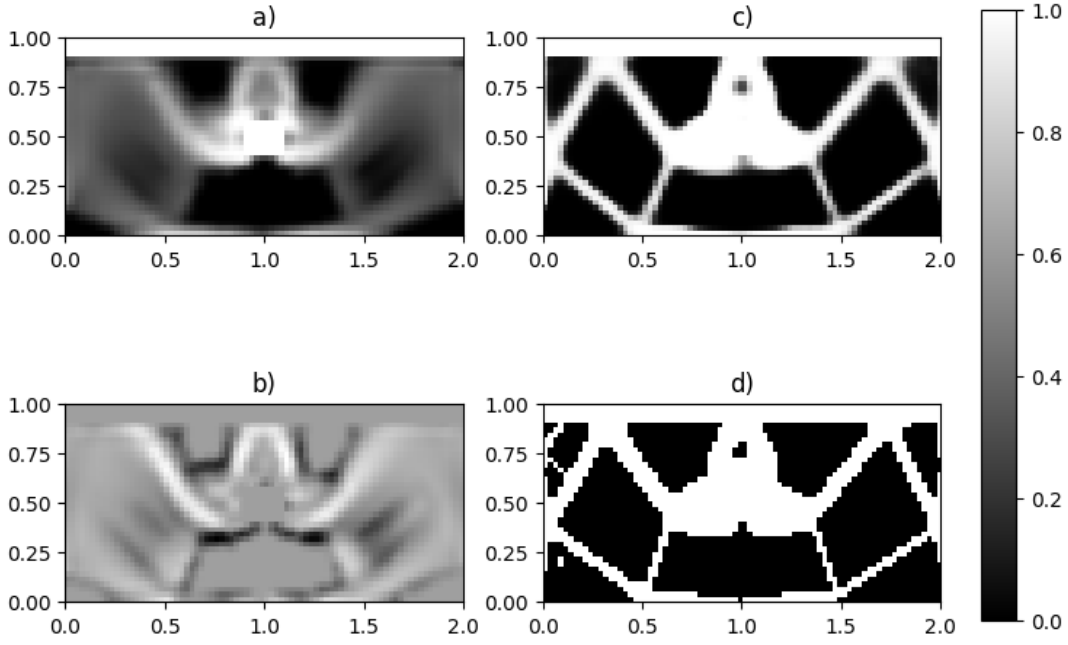


Figure 18: Results on the shear data-set used for training, using sampling with distribution $P(10)$. The input displayed in a) is the element density after 10 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

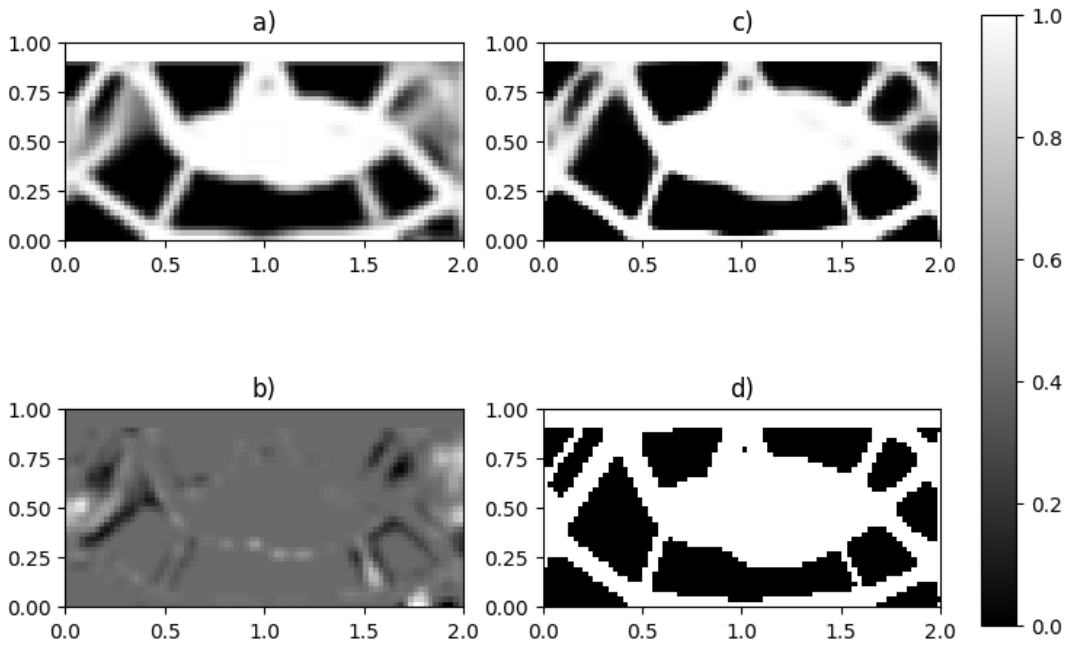


Figure 19: Results on the shear data-set used for training, using sampling with distribution $P(30)$. The input displayed in a) is the element density after 20 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

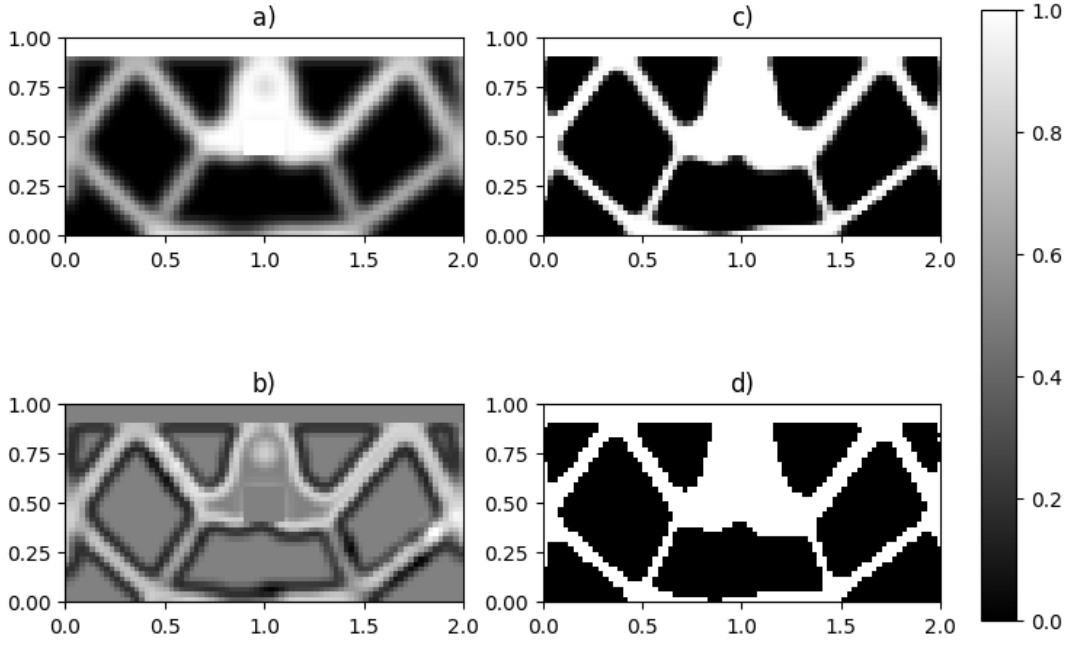


Figure 20: Results on the shear data-set used for training, using sampling with distribution $P(50)$. The input displayed in a) is the element density after 80 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

4.2.2 Performance on new cases

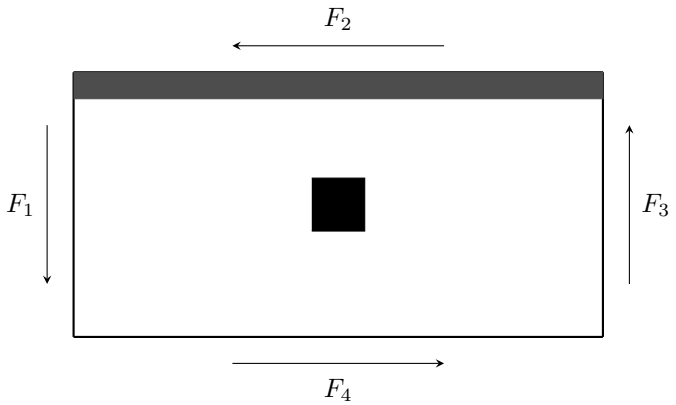


Figure 21: Pure torsion case. Black indicates displacement restriction and fixed element density of 1. Grey indicates fixed element density of 1.

In order to investigate if the network could be used for predicting final structures on previously unseen cases, and specifically the pure torsion case, as well as design domains of other dimensions, some example cases were chosen and presented below. Figure 22, 23, 24, and 25 show the performance of the networks on the pure torsion case on different grids. For the pure torsion case, $\boldsymbol{F_1} = \boldsymbol{F_2} = \boldsymbol{F_3} = \boldsymbol{F_4}$ was chosen and applied as shown in figure 21. $f = 0.4$ and filter radius of 2 was applied. Additionally the networks performance on a completely different topology task, the MBB beam, is included and found in figure 26. The ground truth for these cases was found using topology optimization task $\mathbb{D}$.

An additional small experiment was performed motivated by the results in figure 16. The graphs show that the networks 'poisson 5' and 'poisson 10' for input of iteration ≤ 10 outperforms the thresholding operation by the greatest margin compared to any other network and iteration. The idea being that one could exploit this early prediction from the networks to gain a good approximation of the final topology, which could be used as an initial guess in the optimization for a faster convergence. The network 'poisson 30' applied on iteration 20 was also considered since it seemed to generalize slightly better based on visual results. This idea was briefly explored for the pure torsion cases of dimensions 40x80, 40x120, and 80x160, and for when the second phase of optimization employed all the same specifications as used for the first iterations. The effect of the Heaviside filter was however considered, and all cases were also tried without employing Heaviside projection in the second phase of TO. The choice of a second phase of topology optimization was motivated by the results that the predictions from these earlier iterations not being completely binary (example figure 22), or not constructionally sound (example figure 23). The results were that in none of the cases, except for 1, convergence was achieved in fewer iterations than if purely topology optimization was used. The one case which successfully reduced the number of iterations needed was on the 40x80 grid where the topology at iteration 5 was used as input to the network 'poisson 5' and the second phase optimization (which employed Heaviside projection) with the network prediction as input converged after an additional 148 iteration. Compared to the total 374 iterations when no network was used, the network saved a total of 221 iterations. Both structures fulfilled the volume constraints, but the optimization that employed the network converged to a compliance loss value of 22.80% increase compared to standard topology optimization.

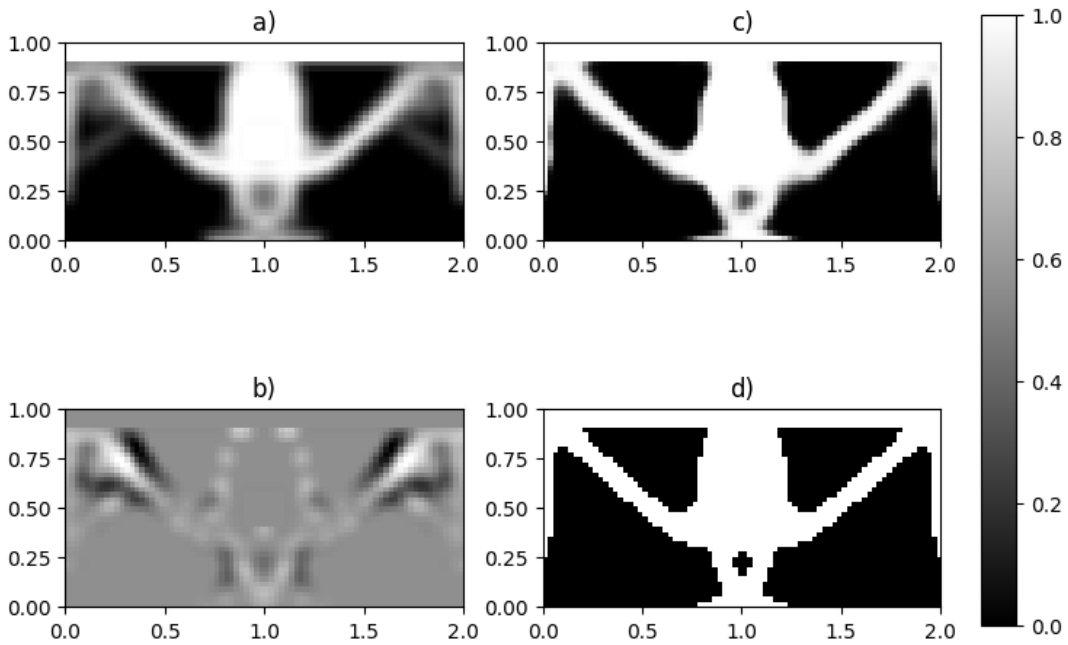


Figure 22: Results and input for the pure torsion case on a 40x80 grid. The network trained on the shear data-set, using sampling with distribution $P(30)$. The input displayed in a) is the element density after 20 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

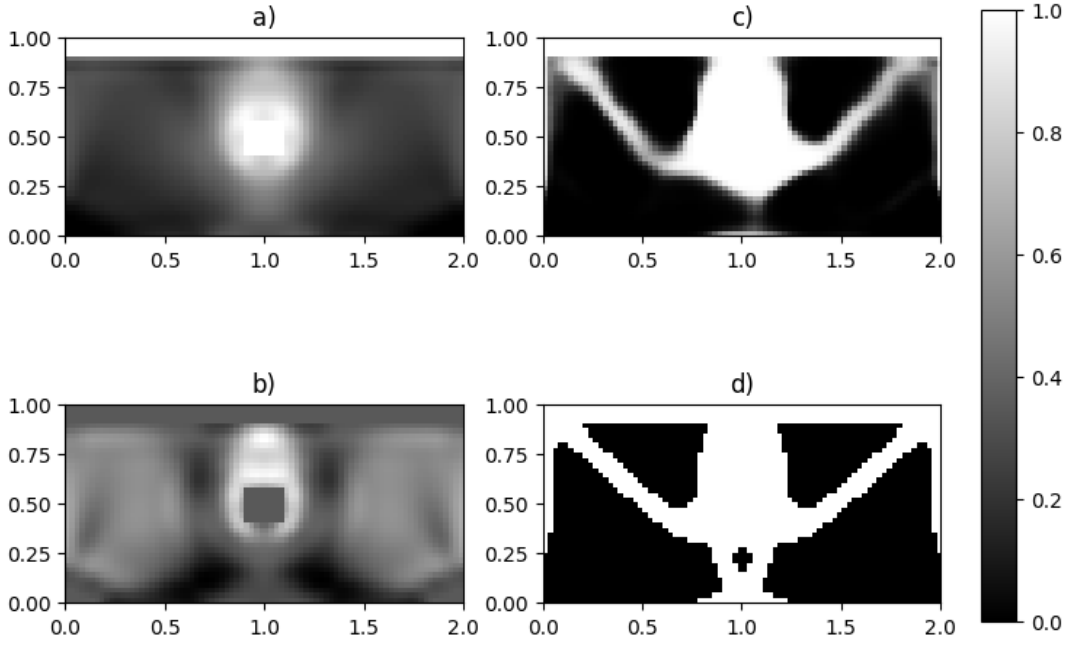


Figure 23: Results and input for the pure torsion case on a 40x80 grid. The network trained on the shear data-set, using sampling with distribution $P(5)$. The input displayed in a) is the element density after 5 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

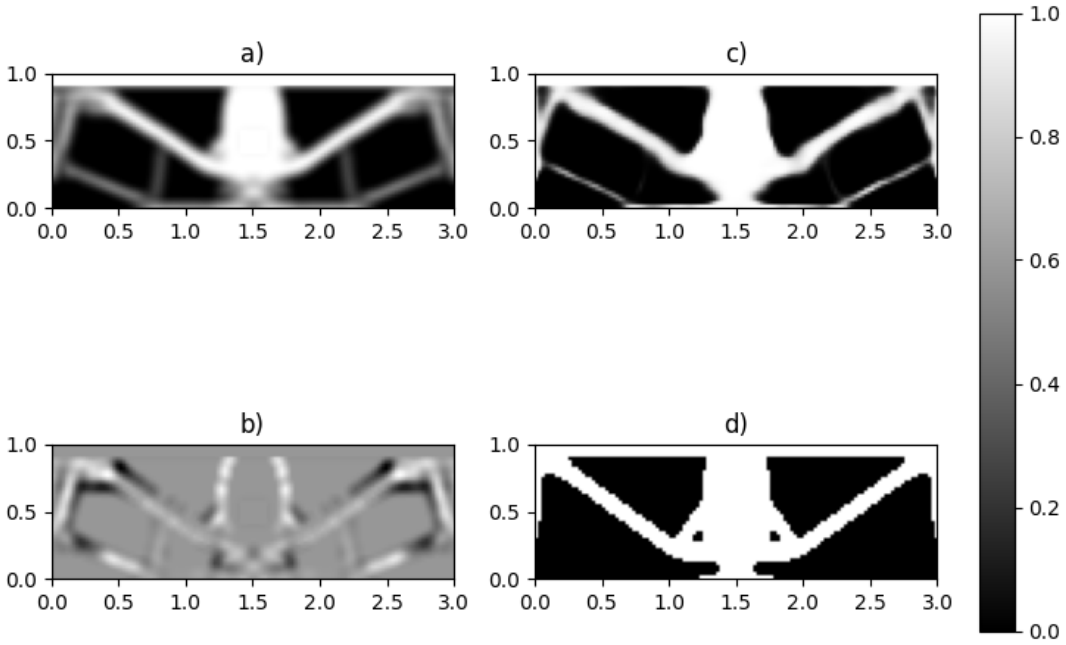


Figure 24: Results for the pure torsion case on a 40x120 grid. The network trained on the shear data-set, using sampling with distribution $P(30)$. The input displayed in a) is the element density after 20 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

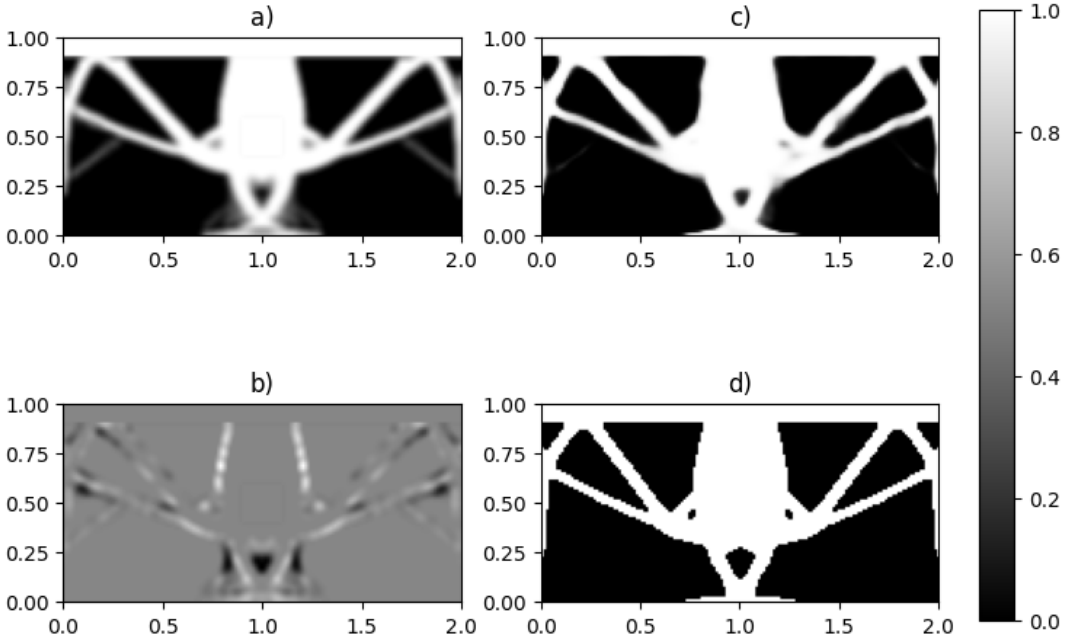


Figure 25: Results for the pure torsion case on a 80x160 grid. The network trained on the shear data-set, using sampling with distribution $P(30)$. The input displayed in a) is the element density after 20 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

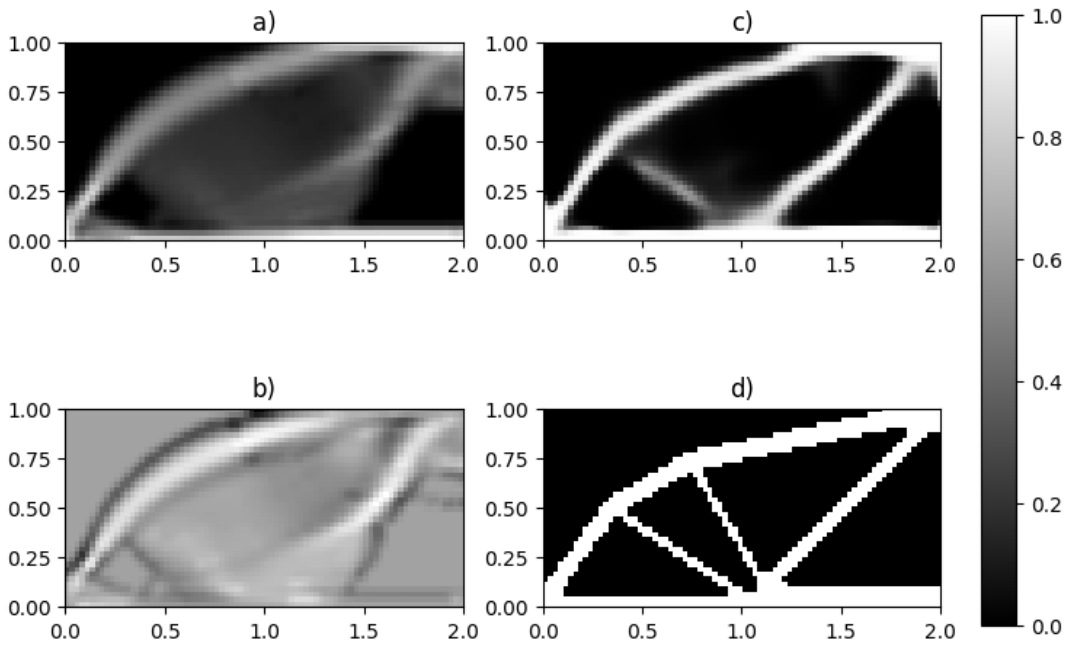


Figure 26: Results for the MBB beam case on a 40x80 grid. The network was trained on the shear data-set, using sampling with distribution $P(10)$. The input displayed in a) is the element density after 10 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

5 Discussion

5.1 Topology Optimization

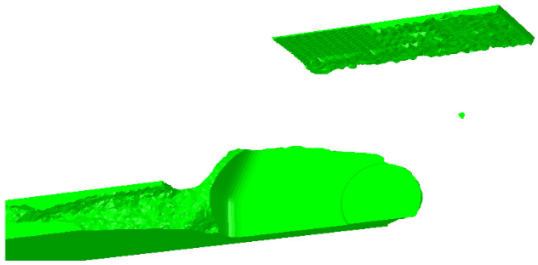
Finding the final load cases and TO task was a challenging task as it required testing and experimenting to get to know the design space. Mostly with regards to the dynamic load cases, the forces and especially the boundary conditions has been a challenge to model in a way which capture realistic response. In order to define a static load case there needs to be boundary conditions that prevents the object from accelerating. This makes modeling an actual moving object as a static case non-trivial. When trying to model forces and boundary conditions for the dynamic cases, a separate analysis when displacement and stress response was evaluated was conducted before TO was applied to ensure that the load case captured a realistic response. The effect on the TO result for the individual, or different combinations of the load cases, also gave good insight and was subsequently used for consideration. It was found that the external parts (back and front wheel attachments and handle) are important to include in the analysis domain. Firstly, they allow the use of point-forces and displacement boundary conditions without resulting in localized stress and displacement since these are not directly applied to the chassis. This lead to a more realistic response. Secondly they allow the modeling of the torque effect caused by a force applied on the wheels. The importance of including this effect became obvious during TO when the converged topology in the front of the vehicle often would be '2 dimensional' if this was disregarded, meaning that the structure would be very thin and and flat. This was solved by introducing the front wheel attachments and the handle, which introduced a torque which lead to a more robust topology. The dynamic load cases in general has been crucial for the front topology of the chassis. The strictly static cases, especially *Simple* on its own results in a completely void front as the optimal structure. The introduction of the static cases *Half-contact*, and both cases of *¾ contact* also helps address this problem of the topology converging to a multiple component structure by introducing asymmetric variations of the standard *Simple* case.

A good final result from TO does not only depend on finding good load cases. As the final design is the optimal given a cost function, the final topology is sensitive to factors such as how the load cases are weighed against each other, and how much total displacement each load case causes. These problems where addressed by introducing a safety factor for the dynamic loads which not only provides a safety margin for the analysis but also help ensure that they trigger a design response. The only case where the load cases were weighted differently from each other in the cost function was for the load case *Backplate*. It needed to be amplified by a factor of 500 compared to the other cases in order to trigger a design response. This is probably due to most of the displacement being only in the plates above the back wheels themselves, which even if the nodal displacement is large, results in a much less total displacement due to few elements being effected. Compared to a force which acts on the front for example, where a lot of elements and a large total displacement would affect the cost function more. The effect of this change is displayed in figure 27.

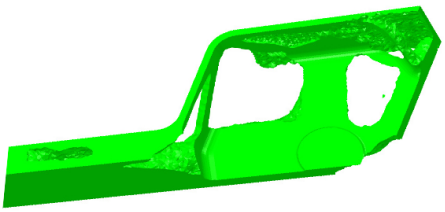
The final load cases aims to capture the most common and crucial situations the chassis will be exposed to during normal use. There are however limitations both regarding the model chosen, the load cases, and optimization task. Firstly the limitation by choosing a linear elastic model is based on the assumptions of small displacements, which could be limiting especially in regards to the load cases *Bump*, where more 'extreme' impact scenarios would result in highly non-linear behavior. Other effects that were not included in the final load cases are torsion-like effects of someone standing asymmetrically on the deck. In the end this was discarded due to it resulting in too many additional load cases to consider in order to capture a variety of momentum-creating cases. Other limitations include some of the modeled displacement restrictions resulting in localized high concentrations of stress. One example includes that the displacement restriction in the z-plane at the location of 'center of mass' in the dynamic load cases has problems with localized stress around this boundary condition. Overall it resulted in a more realistic response than locking e.g. the back wheels. Additionally, the final design might have more or less obvious limitations that could be hard to predict, figure 27 being a good example. Other factors such as aesthetics is also hard to formulate mathematically and include as a constraint in the TO task. These examples all emphasize the importance of post-processing, to compensate for both known and unpredictable limitations.

There is a lot of room for improvement to the proposed model. Future aspects to be considered include mechanical testing to measure loads, dynamic analysis to detect peak loads that could be used instead of the estimated ones. An introduction of non-linear analysis, finer mesh, and additional constraints in

the TO task such as stress could also be investigated. Additional post-processing such as extracting and finalizing the design to perform validation tests, including buckling, stress, etc, are important next steps in product development but outside the scope of this thesis.



(a) $w_{Backplate=1}$



(b) $w_{Backplate} = 500$

Figure 27: Comparison of the converge structure supporting the 'back plate' after cut-off given when load case *Backplate* was weighed differently in the target function.

5.2 Machine learning based Topology Optimization

Without any additional modification of architecture or parameter optimization, the network proposed by Sosnovik & Oseledets (2017) was able to achieve similar results as in their original paper on a completely different data-set. The shear data-set used for training was based on SIMP with Heaviside projection and a varying density filter radius, on a 40 x 80 grid. This is compared to the data-set used by Sosnovik & Oseledets (2017) which was optimized without Heaviside or a varying density filter radius on a 40 x 40 grid for several node-loads of magnitude -1. Both data-sets were pseudo random with variation based on predefined distributions, and they were both compliance-based volume-constrained TO tasks. It therefore shows some transferability in terms of which domain spaces this network can learn. Even though trends of performance are similar (see figure 16), that some networks outperform thresholding by a larger margin in the earlier iterations (poisson 5 and 10), while other networks (poisson 30 and 50) outperform thresholding in the later iterations, with a relative peak around iteration 20. The accuracy is not as high for the shear data-set when compared to networks trained and validated on the original data-set, which could probably be explained by a lack of parameter optimization with regards to the new shear data-set, and possible limitations of the shear data-set. The results achieved were deemed sufficient for the purpose of this report but it is a topic that could be further investigated. The choice of not including a separate validation set of the shear data was made due to limitations in variety of the pseudo randomized shear data-set, a separate validation set would probably not give a much better indication of generality, and instead it is only evaluated if the network can learn one specific design domain. In order for a neural network to be a useful tool, transferability is one of the major aspects that needs to be further investigated. The difficulty in achieving good variety in pseudo-randomly produced data for the purpose of training and validation data is also recognized, and a topic that needs further consideration. For future work, further investigation of additional regularization methods using validation and test data could be considered.

Based on the results in figure 16, it was decided to perform a brief investigation on whether the networks could be applied during the earlier iterations of a new case in order to find a good approximation

of a converged solution, and if this image could be used as an initial guess for another phase of topology optimization. Using this approach, only 1 of the tested cases resulted in a reduced number of optimization iterations, and with only a slight increase in compliance. This indicates that this is not a reliable approach without further tuning of the topology optimization parameters, or additional training of the network. Interesting is that the prediction made by poisson 5 on iteration 5, using Heaviside during the second phase of optimization (see figure 23) outperformed poisson 30 applied on iteration 20 (figure 22) which visually seems to be a better approximation. For the case when no Heaviside projection was used this could be explained as a result of conflict between the true optimum and the binary optimum, the latter which is what the network is trained to predict. This means that even if the network gives a good estimation of the final structure (figure 28 (right image)), the optimization algorithm will spend several iterations converging to an optimal state (figure 28 (left image)), finding low density features which are removed by thresholding in the end. One guess for why the optimization using Heaviside fails to converge given a close-to-optimal structure is that the Heaviside makes it harder for the optimization to change an already binary solution, and it ends up stuck in in a local minima. It is emphasized that these results are only based on a few example cases from which no generalized conclusion can be drawn other than the specific obstacles encountered for these cases.

Given more time, a more robust evaluation of this method could have been performed. Especially the interaction between the networks prediction and further topology optimization on these predictions as input would be interesting to explore since these examples have indicated that there could be potential speed up, but that it would be just as important to adapt the optimization algorithm and parameters for this to be successful as it is to have a network well adapted for the given design domain and task. Another approach could be to not rely on the final phase of optimization to reach convergence, but instead let it run for a finite number of iterations, as was the approach by Kallioras & Lagaros (2021), for an evaluation of the structure as well as small tweaking of the topology. A network trained on a true optimum instead of thresholded result as target could maybe also be a better choice for this type of application of the network and could also be investigated.

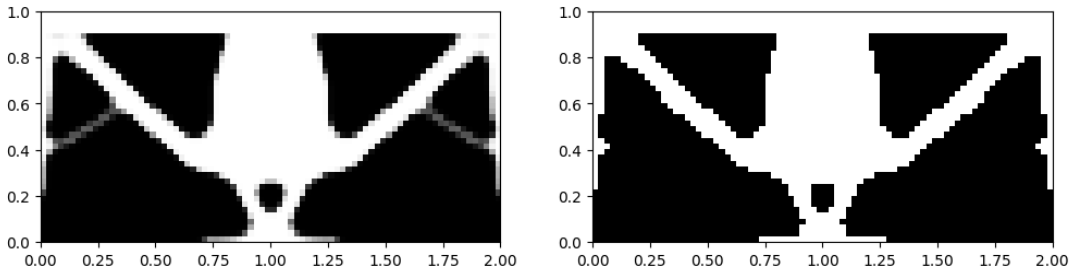


Figure 28: Comparision between the converged structure of the pure torsion case before (left image) and after (right image) a final threshold filter

Other general observations include that, even if the 'poisson 5' and 'poisson 10' give the most impressive results on the data which they were trained on, 'poisson 30' seems to generalize better when applied to new problems. Figure 24 also indicated that the network could struggle with predicting features whose converged state have traveled a longer distance compared to in the earlier iterations. The network shows potential to generalize well based on section 4.4, where even some important features of the MBB beam was predicted (figure 26). Half finished structures like this would however rely on the existence of a reliable way to continue the optimization as discussed above. The network is however superior at problems it has seen before, and is probably heavily biased towards topology commonly occurring in the training data-set. This could especially be a problem with the networks trained on the shear data-set, where, as can be seen in figure 12, a lot of structure seems very similar to each other.Another problem is that some structures have lost important features due to the final threshold when the data was produced. The subject of generalization was somewhat more investigated in the original article by Sosnovik & Oseledets (2017). However, their approach was slightly different as they relied on the network to predict the final structure without any further optimization. Given that most of the actual changes to the topology in the structure happens during the first few iterations of topology optimization, and that the networks that are good at making early predictions show less potential of generalizing well, this could be an argument for

relying on topology optimization for finding the general topology, and use a neural network as a 'smart' threshold-filter, compared to applying the network during the first iterations and letting the optimization do the fine tuning.

6 Conclusion

This thesis explored the use of topology optimization using the industrial software Abaqus as a part of the design process for a chassis of a new micromobility vehicle. Topology optimization can be a helpful tool for indication of where extra material is abundant, reducing the weight of the small vehicle, which is crucial for portability. 9 load cases have been suggested mainly for the purpose of topology optimization, but can also be used for evaluation. Using a relatively simple model, linear elastic, static analysis, and compliance based optimization, a final topology and design suggestion (see figure 15) was obtained. Post-processing and additional analysis of the extracted design is however required. Secondly the use of machine learning, and specifically the deep learning approach of a convolution neural network was investigated. It was found that a lack of resources for producing and storing the amount of data required, as well as the lack of established approaches on design domains as complex as the chassis indicates that the field of machine learning based topology optimization is not yet a mature enough field for this type of industrial application. A study of a smaller scale problem, still in relation to the original task, was however conducted. Instead of applying machine learning to the full domain it was limited to finding the topology of the cross section of the deck of the chassis. The network proposed by Sosnovik & Oseledets (2017) showed good promise of learning the domain of the training data. Such a network could be used as a tool for skipping iterations in the topology optimization process by early prediction of a final design. An alternative approach was briefly investigated where further topology optimization was performed on the output from the network to ensure a good structure, and one example shows a 59% reduction of total iterations of topology optimization for convergence. Transferability and consistency of these approaches are however the next important step in research on the topic of machine learning based topology optimization.

References

Bendsoe, M. P. & Sigmund, O. (2004), *Topology optimization: theory, methods, and applications*, Springer Berlin, Heidelberg.

Bruns, T. E. & Tortorelli, D. A. (2001), 'Topology optimization of non-linear elastic structures and compliant mechanisms', *Computer Methods in Applied Mechanics and Engineering* **190**(26), 3443–3459.

Chandrasekhar, A. & Suresh, K. (2021), 'Tounn: Topology optimization using neural networks', *Structural and Multidisciplinary Optimization volume* **63**. 1135–1149.

Christensen, P. W. & Klarbring, A. (2010), *An Introduction to Structural Optimization*, Springer.

Dattakumar, S. S. & Ganeshan, V. (2017), Converting dynamic impact events to equivalent static loads in vehicle chassis, Master's thesis, Chalmers University of Technology.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Guest, J., Prévost, J. & Belytschko, T. (2004), 'Achieving minimum length scale in topology optimization using nodal design variables and projection functions', *Int. J. Numer. Meth. Engng.* **61**, 238–254.

Guo, T., Lohan, D., Cang, R., Ren, Y. & Allison, J. (2018), An indirect design representation for topology optimization using variational autoencoder and style transfer, *in* '2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference', p. 0804.

Kallioras, N. A. & Lagaros, N. D. (2021), 'Dl-scale: a novel deep learning-based model order upscaling scheme for solving topology optimization problems', *Neural Computing and Applications* **33**. 7125–7144.

Lee, S., Kim, H., Lieu, Q. X. & Lee, J. (2020), 'Cnn-based image recognition for topology optimization', *Knowledge-Based Systems* **198**. 0950-7051.

McCarthy, J. (2007), 'What is artificial intelligence?'. Computer Science Department, Stanford University.

Ohlsson, M. & Edén, P. (2022), *Introduction to Artificial Neural Networks and Deep Learning*, Department of Astronomy and Theoretical Physics Lund University. Lecture notes.

Ottosen, N. & Petersson, H. (1992), *Introduction to the Finite Element Method*, Pearson Education Limited.

Park, G.-J. (2010), 'Technical overview of the equivalent static loads method for non-linear static response structural optimization', **43**, 319–337.

Ronneberger, O., Fischer, P. & Brox, T. (2015), 'U-net: Convolutional networks for biomedical image segmentation', *International Conference on Medical Image Computing and Computer-Assisted Intervention* p. 234–241.

Sosnovik, I. & Oseledets, I. (2017), 'Neural networks for topology optimization', *arXiv preprint arXiv:1709.09578* .

Woldseth, R. V., Aage, N., Bærentzen, J. A. & Sigmund, O. (2022), 'On the use of artificial neural networks in topology optimisation', *Structural and Multidisciplinary Optimization* **65**(10), 294.

Xiao, D., Liu, X., Du, W., Wang, J. & He, T. (2012), 'Application of topology optimization to design an electric bicycle main frame', **46**, 913–929.

Yu, Y., Hur, T., Jung, J. & Jang, I. G. (2019), 'Deep learning for determining a near-optimal topological design without any iteration', *Structural and Multidisciplinary Optimization* **59**, 787–799.

A Mesh

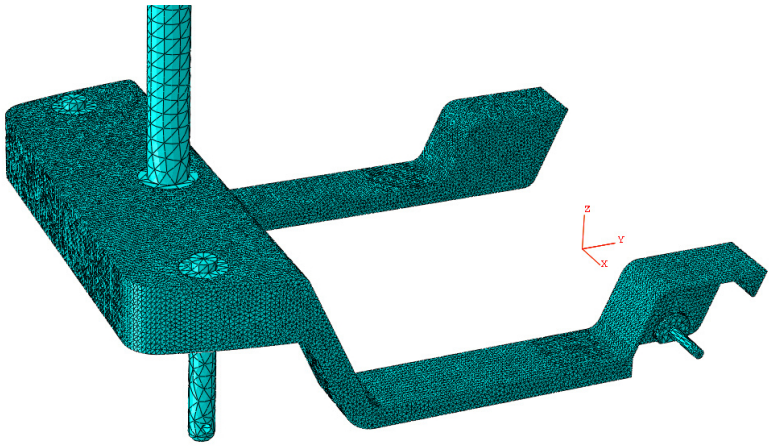


Figure A.1: Mesh of the chassis and coarser mesh of the separate parts outside of the design domain.

B More results from neural network

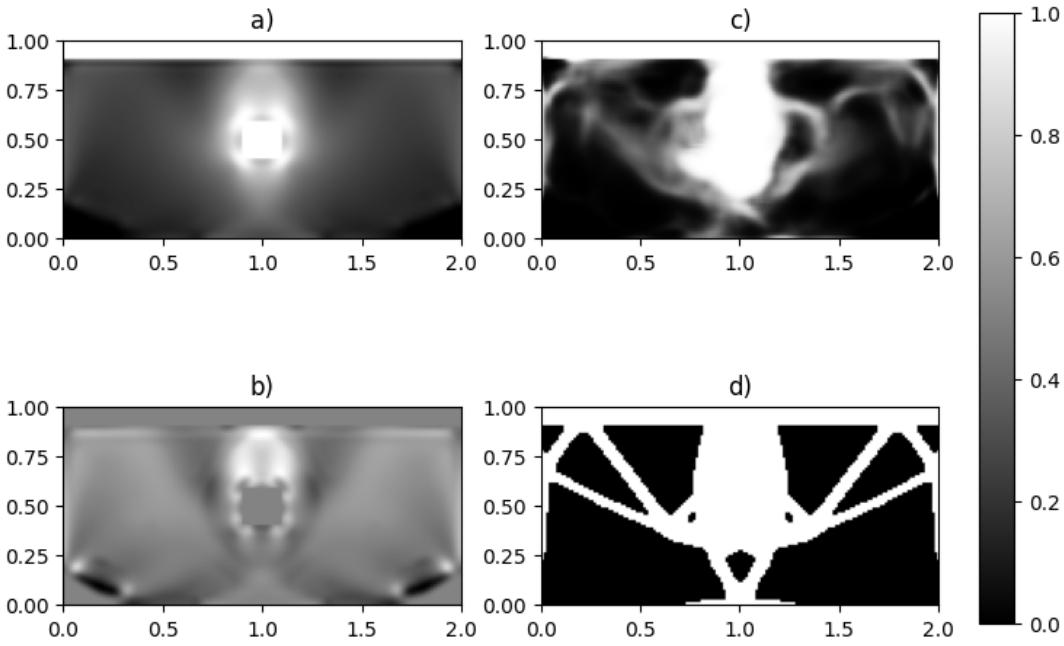


Figure B.1: Results for the pure torsion case on a 80x160 grid. The network trained on the shear data-set, using sampling with distribution $P(5)$. The input displayed in a) is the element density after 5 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).

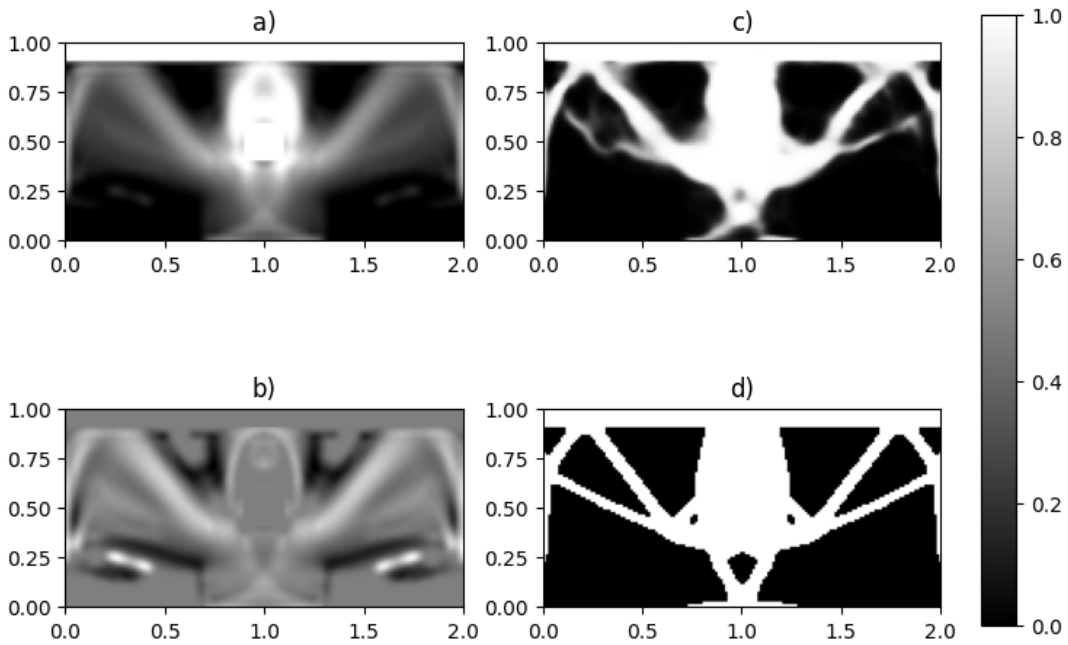


Figure B.2: Results for the pure torsion case on a 80x160 grid. The network trained on the shear data-set, using sampling with distribution $P(5)$. The input displayed in a) is the element density after 10 iterations of topology optimization, and the input displayed in b) is the gradient of the latest element density update. The output of the network is displayed in c) and the ground truth is shown in d).