NTNU

Norwegian University of
Science and Technology

Exploring the capabilities of machine learning (ML) for 1D blood flow: Application to coronary flow

Vilde Sklet

Master of Science in Mechanical Engineering
Submission date: June 2018
Supervisor: Leif Rune Hellevik, KT

Norwegian University of Science and Technology
Department of Structural Engineering

Department of Structural Engineering
Faculty of Engineering
NTNU- Norwegian University of Science and Technology

ACCESSIBILITY
Open

MASTER THESIS 2018

SUBJECT AREA:	DATE:	NO. OF PAGES:
Biomechanics	June 25th 2018	79 + 18

TITLE:

Exploring the capabilities of machine learning (ML) for 1D blood flow: Application to coronary flow

Utforske mulighetene for maskinlæring (ML) på 1D blodstrøm: Anvendt på koronarstrømning

BY:

Vilde Sklet

SUMMARY:
The aim of this thesis is to explore the capabilities of deep neural networks to reproduce 1D computational models for the pressure in a coronary tree. A machine learning algorithm was implemented. The algorithm was trained with a synthetically generated database of coronary trees, where the anatomical data was retrieved from published literature. A grid search was performed to optimize the hyper-parameters in the machine learning model.

Two different models were trained to solve a steady state; coronary blood flow model and Young and Tsai's stenosis model. Correlation between the predicted values was excellent for both models with r^2= 1 for the steady state coronary blood flow model, and r^2=0.997 for Young and Tsai's stenosis model.

The established ML models were tested with patient specific data. The prediction on the patient specific data showed that the synthetically generated database did not represent the pathological variation of coronary arteries. For a reduced patient specific database, the model predicted the pressure drop along the healthy vessels with a coefficient of determination of 0.799 and for the reduced database with the patient specific stenoses the coefficient of determination was 0.997.

RESPONSIBLE TEACHER: Leif Rune Hellevik

SUPERVISOR(S): Lucas Omar Müller

CARRIED OUT AT: Department of Structural Engineering, NTNU

Assignment

Exploring the capabilities of machine learning (ML) for 1D blood flow: Application to coronary flow

Today's standard for diagnosing coronary artery disease is an invasive method to measure the Fractional flow reserve, (FFR). FFR is a functional index used for diagnosis of stable coronary artery disease. A non-invasive method is desirable to reduce the risk of the patients. There exist non-invasive methods using CT imaging and computational fluid dynamics (CFD) simulations.

The main objective of this thesis is to explore the capabilities of deep neural networks to reproduce 1D computational models for the pressure in a coronary tree. The work is connected to the Biomechanics group at the Department of Structural Engineering, at NTNU.

Suggested topics for the thesis are therefore:

- Applying ML to the stenosis model
- Identify the necessary features for a coronary tree to apply ML
- Generate a synthetic training database of coronary trees
- Calculate FFR for each coronary tree in the training database
- Train the ML algorithm and validate the result

Abstract

The aim of this thesis is to explore the capabilities of deep neural networks to reproduce 1D computational models for the pressure in a coronary tree. A machine learning algorithm was implemented. The algorithm was trained with a synthetically generated database of coronary trees, where the anatomical data was retrieved from published literature. A grid search was performed to optimize the hyper-parameters in the machine learning model.

Two different models were trained to solve a steady state; coronary blood flow model and Young and Tsai's stenosis model. Correlation between the predicted values was excellent for both models with coefficient of determination, $r^2 = 1$ for the steady state coronary blood flow model, and $r^2 = 0.997$ for Young and Tsai's stenosis model.

The established ML models were tested with patient specific data. The prediction on the patient specific data showed that the synthetically generated database did not represent the pathological variation of coronary arteries. For a reduced patient specific database, the model predicted the pressure drop along the healthy vessels with a coefficient of determination, $r^2 = 0.799$ and for the reduced database with the patient specific stenoses, $r^2 = 0.997$.

Sammendrag

Målet med denne oppgaven er å utforske mulighetene til å benytte neurale nettverk for å reprodusere 1D beregningsmodeller for trykk i koronærarterier. En algoritme for maskinlæring ble implementert. Algoritmen ble trent med bruk av en syntetisk generert database av koronærarterier der de anatomiske dataene ble hentet fra publisert litteratur. Et rutenettsøk ble utført for å optimalisere hyperparametrene i maskinlæringsmodellen.

To forskjellige modeller ble trent for å løse en stabil tilstand; koronær blodstrømsmodell og Young og Tsai's stenose-modell. Korrelasjonen mellom de anslåtte verdiene var utmerket for begge modellene med r2 = 1 for den stasjonære koronære blodstrømningsmodellen, og r2 = 0,997 for Young og Tsai's stenose-modell.

De etablerte ML-modellene ble testet med pasientspesifikke data. Prediksjonen av pasientspesifikke data viste at den syntetisk genererte databasen ikke representerte den patologiske variasjonen av koronærarteriene. For en redusert pasientspesifikk database predikerte modellen trykkfallet langs de friske arteriene med en koeffisient på 0,799, og trykkfallet over stenoser med en bestemmelseskoeffisienten på 0,997.

Contents

List of Tables

List of Figures

Acronyms and Abbreviations

Acr./Abbr.	Description
AI	Artificial intelligence
API	Application programming interface
CA	Coronary arteries
CAD	Coronary artery disease
CCTA	Coronary computed tomography angiography
CT	Computed tomography
CVD	Cardiovascular diseases
DNN	Deep neural network
FFR	Fractional flow reserve
$\mathbf{FFR_{calc}}$	Fractional flow reserve computed from CT images
$\mathbf{FFR_{ML}}$	Fractional flow reserve values predicted by the ML model
ICA	Invasive coronary angiography
LAD	Left anterior descending
LCx	Left circumflex artery
ML	Machine learning
PCI	Percutaneous coronary intervention
PDE	Partial differential equation
RCA	Right coronary artery
STARFiSh	Stochastic Arterial Flow Simulations
STD	Standard deviation

1 Introduction

One of the biggest healthcare challenges the world is facing are Cardiovascular Diseases (CVD), with nearly 18 million deaths a year [1]. Coronary Artery Disease (CAD) is by far the largest contributor and is responsible for a third of all deaths in industrialized countries [2]. In the United States, CVD is responsible for 17 % of the national health expenses, and it is expected to triple between 2010 and 2030 [3]. For this reason, it is important to improve diagnostic methods and treatment, thereby improving patient outcome and reduce the healthcare costs. Today's gold standard for diagnosing CAD is invasive pressure measurements to determine the Fractional Flow Reserve (FFR) [4, 5]. All invasive procedures have possible complications and possess a risk for the patient. To overcome this risk, a non-invasive diagnostic procedure is preferable.

The continuing development in fields like fluid dynamics, non-invasive imaging and patient specific modelling have made it possible to compute FFR non-invasively, from standard Coronary Computed Tomography Angiography (CCTA), referred to as FFR_{calc}. FFR_{calc} has the possibility to reduce the costs by 32% and several clinical trials indicates that FFR_{calc} improves the non-invasive assessment of stable CAD [6, 7]. Recently the field of Artificial Intelligence (AI) has been combined with computational FFR calculations to reduce the computation time for solving the problems. FFR values from Machine Learning (ML) models are referred to as FFR_{ML}.

1.1 Stable coronary artery disease

The coronary artery system is responsible for the blood supply to the heart muscle/myocardium. The two main branches are the Right Coronary Artery (RCA) and the Left Main artery (LM), as seen in Figure 1.1. The LM bifurcates early into the Left Anterior Descending (LAD) and the Left Circumflex (LCx) arterie.

CAD occurs when the Coronary Arteries (CA) cannot provide the myocardium with enough blood, and thus oxygen, resulting in ischemia. The pathogenesis of CAD consists of a stiffening and narrowing of the CA, stenotic lesions, most commonly caused by atherosclerosis. Atherosclerosis regards the accumulation of degenerative materials inside the arterial walls which are calcified into atheromatic plaques. An illustration of a diseased artery is seen in Figure 1.2. Common symptoms of CAD are chest pain (angina) and shortness of breath. Untreated CAD may result in heart failure and death [9].

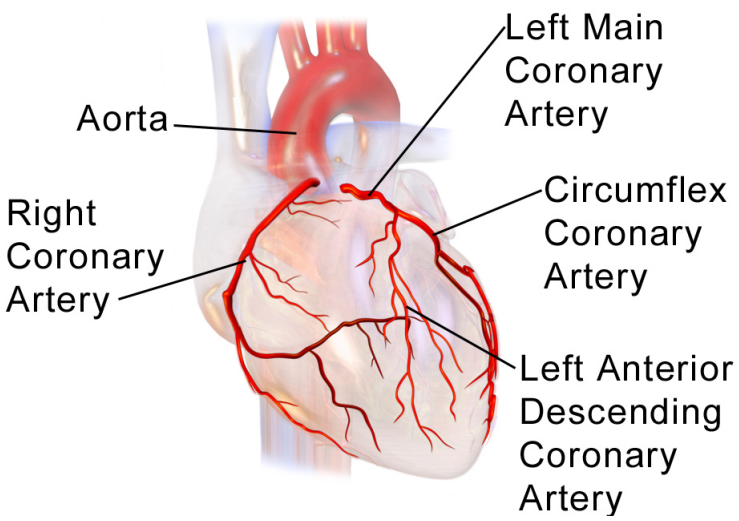


Figure 1.1: Anatomy of the coronary arteries [8].

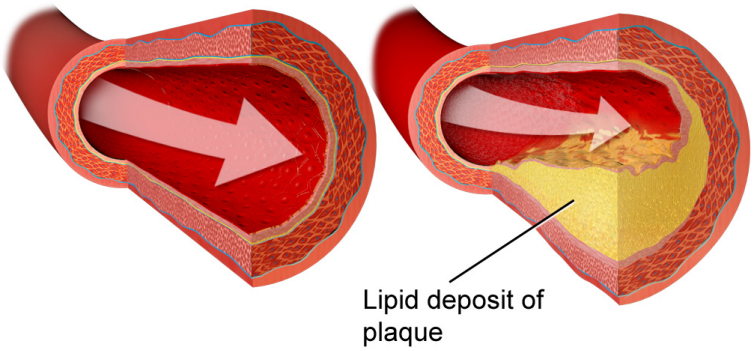


Figure 1.2: Illustration of a healthy vessel and a diseased vessel [10].

1.2 Fractional flow reserve

FFR measurements are used to assess the severity of a coronary artery stenosis because it provides information of the functional significance of the stenosis. This gives a better diagnostic performance than angiography[1] alone. FFR is approximated as the ratio between the mean distal coronary pressure, P_d, and the mean proximal coronary pressure, P_a.

$$FFR = \frac{P_d}{P_a}. \tag{1.1}$$

[1] A x-ray study of the blood vessel. A radiopaque substance, or dye is injected to the blood to make it visible on the x-ray images.

1.2.1 Clinical FFR

In clinical studies, FFR$\leq$ 0.80 is used as a threshold value for assessing the severity of the stenosis in terms of its contribution to ischemia. Patients with lower FFR benefits from Percutaneous Coronary Intervention (PCI) revascularization while patients with a higher FFR score has a better outcome from medical treatment alone [4]. To obtain FFR-values Invasive Coronary Angiography (ICA) is performed. Through catheterization, a pressure wire technique is used to obtain pressure measurements at specific locations, thereby being able to measure the pressure ratio over the stenosis [5].

1.2.2 Computational FFR

Recently, non-invasive methods to determine FFR-values have emerged. FFR_{calc} uses standard CT data. It is determined through image post-processing and computational fluid dynamics calculations. Several studies have shown good diagnostic performance for FFR_{calc} compared to invasive FFR measurements [11, 12].

Non-invasive computation of FFR_{calc} is a four-step procedure:

1. Performing CCTA, coronary computed tomography angiography
2. Anatomic 3D model/Physiology model
3. Computation of coronary blood flow
4. FFR_{calc} values for the coronary tree

Step 1 is a standard procedure, performed in the clinic by medical personnel. The CT image is segmented to create a 3D model of the coronary arteries. When the coronary artery geometry the mathematical model of coronary blood flow is solved, imposing boundary conditions. When the mathematical model is solved, the FFR_{calc} values are computed for the coronary tree.

1.2.3 Previous work on on-site FFR_{calc}

Different technical approaches have been investigated to calculate FFR_{calc}. These have been both full- and reduced-order computational fluid dynamic models, based on the Navier-Stokes equations and physical laws that govern fluid dynamics [13]. The three-

dimensional, full-order models are computationally expensive, and are currently computed off-site. HeartFlow, Inc., a Stanford University-based company, offers a commercial available FFR_{calc} diagnostic software. Each FFR_{calc} analysis requires 1 to 4 hours computational time, depending both on the quality of the CT image and the disease burden of the patient [14].

To overcome the on-site computational limitation of the full-order models, reduced order and steady-state models have been developed with promising results[15, 16].

Recently, Itu et al. has presented a deep machine learning approach to provide on-site calculations [17]. A synthetically generated database, combined with the reduced-order model described in [15], mentioned above, provides the training database. The ML algorithm is trained using a Deep Neural Network (DNN). The correlation between the ML predictions and the reduced-order model was excellent, and they managed to reduce the average execution time down to just a few seconds.

Sankaran et al. at HeartFlow, Inc. is using a ML algorithm to determine the hemodynamic sensitivity to lumen segmentation uncertainty [18].

1.3 Objectives

The main purpose of this thesis is to explore the capabilities of deep neural networks to reproduce 1D computational models for the pressure in a coronary tree. The 1D model is already developed by the Biomechanics group at the Department of Structural Engineering, NTNU, which makes the 1D results easily obtained. This step is mandatory before it is possible to move to a fully ML approach for quantification of the pressure drop in coronary arteries.

To achieve this, two main objectives must be fulfilled. The first one is to generate a synthetic training database of coronary trees. The database must represent a pathological variation for patients with stable CAD, and contain the input features and labels necessary to train the ML algorithm.

The second objective is to implement a ML model. The model must be established, trained with the developed database and its performance must be evaluated. The aim is to be able to predict the FFR values for a coronary tree with the trained ML model. This thesis is inspired by the work of Itu et al.[17].

1.4 Structure of the Report

The report contains 6 chapters. Chapter 1 presents the background and the objectives of the work. A theoretical background for the work is presented in Chapter 2. Chapter 3 describes the method used in this thesis, while Chapter 4 presents the results of the work. The method and results are discussed in Chapter 5. Chapter 6 presents the overall conclusions and propose ideas for further work.

2 Theoretical Background

The main theoretical aspects related to this thesis are presented in this chapter. First the theoretical framework for the fluid dynamics of coronary arteries, relevant for the FFR_{calc} computations is presented. Starting with the fundamental concepts of fluid mechanics, followed by the physiology of the coronary arteries. In the third sub-chapter, the fluid mechanical principles are combined with the physiology of coronary blood flow. This forms the basis for the mathematical modeling principles of coronary blood flow.

Secondly, some of the main concepts of machine learning are introduced. Hereby the focus lies on deep neural networks and the steps necessary to implement a machine learning algorithm.

2.1 Governing equations in fluid mechanics

The Navier-Stokes equations are fundamental in fluid mechanics, governing the flow for incompressible, viscous fluids. They describe the dynamical effect of external and internal forces for a Newtonian fluid. The equations impose conservation of mass and balance of momentum for fluids rising from Newton's second law.

2.1.1 3D Navier-Stokes

Consider the 3D compliant vessel illustrated in Figure 2.1, with the flow region $\Omega_d \in \mathbb{R}$. The boundaries are defined by the vessel wall, Γ_w, the inlet and the outlet cross-sections, S_0 and S_1. For an incompressible, Newtonian fluid the Navier-Stokes equations become:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla P + \upsilon \nabla^2 \mathbf{u} \tag{2.1a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.1b}$$

where $\mathbf{u}$ is the velocity vector, ρ is the density, P is the pressure and υ is the kinematic viscosity. Equation 2.1a is the vector form of the momentum equation and equation 2.1b represents the conservation of mass [19].

Because of the complexity of the incompressible Navier-Stokes equations, analytical solutions are only found for simple problems. Most practical fluid mechanical problems require either a numerical approach to solve the Navier-Stokes equations or need simplifying

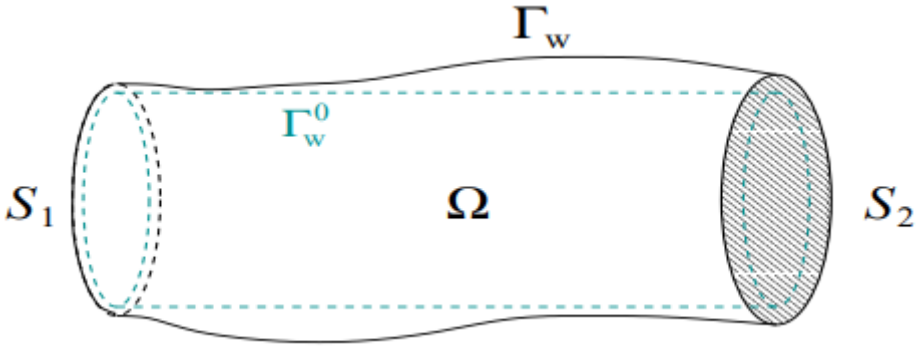


Figure 2.1: Simple 3D compliant pipe [19].

assumptions.

2.1.2 1D compliant vessel

The governing equations for a 1D compliant vessel can be found by integrating the Navier-Stokes equations (2.1a,2.1b) at each time, $t > 0$, and over each cross section $S(t, z)$, when the boundary Γ_w^0 of the reference configuration Ω_0 is a cylinder of radius r_0 as illustrated in Figure 2.2. The governing equations read:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial z} = 0 \tag{2.2a}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial z}\left(\alpha \frac{Q^2}{A}\right) = -\frac{A}{\rho}\frac{\partial p}{\partial z} + \frac{f}{\rho} \tag{2.2b}$$

where A is the cross-sectional area and Q is the volumetric flow. The pressure P is assumed constant over the cross section, ρ is the density of blood, f is the frictional term and α accounts for the non-linearity of the velocity profile. Given the velocity-profile

$$u(x, \xi, t) = U(x, t)\frac{\zeta + 2}{\zeta}\left[1 - \left(\frac{\xi}{r}\right)^{\zeta}\right], \tag{2.3}$$

where $U(x, t)$ is the cross-sectional averaged velocity, ξ is the radial coordinate, ζ is polynomial order and r is the vessel radius, the frictional term becomes $f = -2(\zeta+2)\mu\pi U$. $\zeta = 2$ corresponds to Poiseuille flow.

Written in terms of the steady state variables $\bar{P}$, $\bar{Q}$ and $\bar{A}$, equation 2.2a and 2.2b

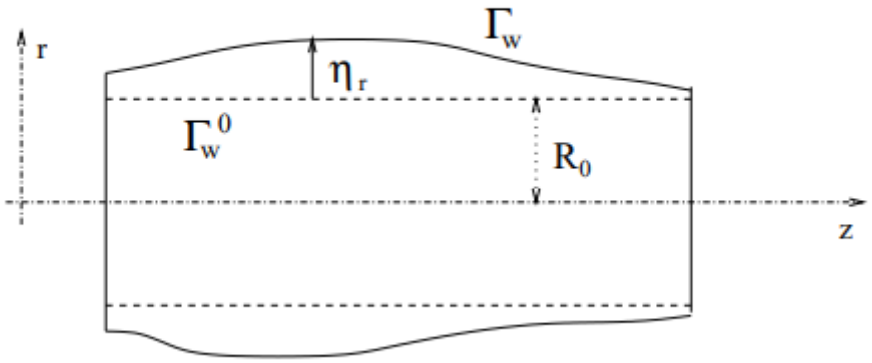


Figure 2.2: 1D compliant pipe [19].

become:

$$\bar{P}_{in} + \rho\frac{1}{2}\left(\frac{\bar{Q}_{in}}{\bar{A}_{in}}\right)^2 = \bar{P}_{out} + \rho\frac{1}{2}\left(\frac{\bar{Q}_{out}}{\bar{A}_{out}}\right)^2 + \bar{Q}_{in}\int_0^l \frac{2(\zeta+2)\pi\mu}{\bar{A}^2}\mathrm{d}x \tag{2.4a}$$

$$\bar{Q}_{in} = \bar{Q}_{out} \tag{2.4b}$$

2.2 Coronary blood flow

The behaviour of coronary blood flow is characterized by the complex interaction between the rheological properties of blood, geometry of the coronary tree, interaction between the blood and artery wall and the pulsatile flow [2].

Blood is considered an incompressible fluid [20]. It is a heterogeneous multi-phase fluid, consisting of platelet, red and white blood cells, suspended in a liquid plasma. The rheological properties of blood depend on the composition of these substances, as well as on external physical condition and applied deformation forces. Plasma is a Newtonian fluid, while the red blood cells contribute with the non-Newtonian effects [21]. The non-Newtonian behaviour of blood depends on its velocity. With increased velocity, the shear rate increases and blood passes from non-Newtonian to Newtonian flow. In arteries where the shear rate is higher than $100\,\mathrm{s}^{-1}$ the blood behaviour is generally treated as Newtonian [22].

The Reynolds number is the ratio of inertial forces to viscous forces in the fluid,

$$Re = \frac{\rho \bar{u} D}{\mu} \tag{2.5}$$

where $\bar{u}$ is the average flow velocity, D is the characteristic length, the diameter for vessels, and μ blood density and ρ is the viscosity [23]. The Reynolds Number is a dimensionless number. Flow in a pipe with $Re \leq 2300$ is laminar. With a $Re \approx 400$, the flow in the LAD is laminar [24]. Stenotic regions impact the flow characteristics, and experimental studies indicate a critical value of approximately 200 for the Reynolds number. For higher values, there will be flow separation downstream of the stenosis [25].

2.2.1 Pulsatile flow

The cardiac cycle consists of two periods, systole and diastole. The coronary vessels provide the myocardium with blood, and during systole the myocardium contracts and ejects blood. The aortic pressure is rising rapidly due to compressing of the peripheral vasculature and increasing its resistance, applying high external force on these vessels. This results in a decrease in the coronary blood flow. On the other side during diastole, the heart relaxes and fills with blood. The aortic pressure is coming down from its peak, giving an increase in the coronary blood flow. This means that the coronary blood flow is at its highest during diastole. This pulsating behaviour is an important characteristic, differentiating coronary blood flow from other cardiovascular flows [26].

2.2.2 Hyperemia

Previous studies have shown that the effect of a stenosis at resting coronary flow, low flow velocity, is small compared to the impact during hyperemia [27, 28]. Under baseline conditions, the average resting coronary flow for humans is around 4-5 % of the total cardiac output (q_{CO}). With $q_{CO} = 5000\,\mathrm{ml/min}$ the resting flow is approximately $q \approx 225\,\mathrm{mm/min}$ [2].

Hyperemia is the state of maximal coronary flow, and is a response to increased myocardial oxygen demand. Hyperemia must be provoked to investigate the severity of the stenosis. This may be done either through exercise, or by injecting pharmacologic agents. Adenosine is the most commonly used agent. The hyperemic coronary blood flow induced by adenosine is normally around 4.5 times the baseline coronary blood flow velocity [28, 29].

2.3 Coronary blood flow modelling

Coronary blood flow will be modelled by simplified models for the fluid dynamics of coronary arteries. Here the steady-state assumption will be presented, together with the models for the arterial junctions, boundary conditions and the Young and Tsai's stenosis model.

2.3.1 Steady-state assumptions

A normal simplification for coronary blood flow modelling is to assume steady-state flow, in contrast to pulsatile flow (2.2.1). Internal studies at the research group of Biomechanics at NTNU have shown that the steady-state assumption have limited effect on the pressure drop in coronary arteries. Moreover, Huo et al. obtained a pressure drop error of less than $\pm 5\%$ in an *in vitro* model, when the steady-state coronary blood flow is representing the mean value of the pulsatile flow rate [30, 31, 32].

2.3.2 Arterial junctions

Arterial junctions are modelled by ensuring conservation of mass and the coupling equation for the pressure,

$$P_0 + \frac{\rho}{2}(\frac{Q_0}{A_0})^2 = P_i + \frac{\rho}{2}(\frac{Q_i}{A_i})^2 + \Delta P_i, \quad i = 2, \ldots, N \tag{2.6a}$$

$$\sum_{i=1}^{N} Q_i = 0 \tag{2.6b}$$

where ΔP is an additional pressure loss and N is the number of vessels in the connection. For bifurcations in healthy conditions, it can be assumed that ΔP is zero, so Equation 2.6a describes continuity of total pressure.

2.3.3 Boundary conditions

Boundary conditions at the inlet and at all outlets of the coronary network must be defined to close the system.

Inlet conditions

The inlet condition is prescribed by the inlet pressure. Then the total peripheral resistance is mathces to the flow estimate of the cardiac output.

Outlet conditions

The pressure and flow at the coronary outlets are distributed according to the total peripheral resistances. The total peripheral resistance is lumped at each outlet, coupling the outlet pressure and flow at each location. The total peripheral resistance can be found from Murray's law. Based the minimum energy consumption hypothesis, Murray related the flow in a vessel to the radius [33]. Assuming Poiseuille flow, Murray's law yields:

$$Q_i \propto r_i^3 \tag{2.7}$$

For a bifurcating network this would be achieved if

$$r_{pa}^3 = r_{d1}^3 + r_{d2}^3 + \cdots + r_{dn}^3 \tag{2.8}$$

where r_{pa} refers to the parent vessel and $r_{d1}, \ldots, r_{dn}$ are the daughter vessels. For arterial blood flow, 2.76 is a good choice for the exponent [34].

2.3.4 Young and Tsai's stenosis model

When an arterial stenosis is present, a stenosis model has to be introduced to account for the three-dimensional flow regime, as the 1D assumptions are not valid any more. Young and Tsai proposed a stenosis model based on *in vitro* experiments [35]. They related the pressure drop to the geometry of the stenosis and other parameters, namely

$$\Delta P = \frac{K_v}{Re}\rho U^2 + \frac{K_t}{2}(\frac{A_0}{A_s} - 1)^2\rho U^2 \tag{2.9}$$

where ΔP is the pressure drop, Re is the Reynolds number defined in Equation 2.5, ρ is the density of the blood, U is the mean velocity in the unobstructed tube. A_0 and A_s is the reference and the stenotic cross-section area respectively with corresponding D_0 and D_s. K_v and K_t are empirical coefficients depending on the geometry and severity of the stenosis [36]. For a given flow rate Q, equation 2.9 becomes,

$$\Delta P = \frac{K_v\mu}{A_0 D_0}Q + \frac{K_t\rho}{2A_0^2}(\frac{A_0}{A_s} - 1)^2 Q \mid Q \mid = \alpha_1 Q + \alpha_2 Q^2 \tag{2.10}$$

with the empirical coefficients,

$$K_v = 32(0.83L_s + 1.64D_s)(\frac{A_0}{A_s})^2/D_0 \quad \text{and} \quad K_t = 1.52 \tag{2.11}$$

where L_s is the stenosis length and μ is the dynamic blood viscosity. The first term in 2.10 represents pressure loss from viscous effects due to the contraction of the vessel and the second term pressure loss due to turbulence in the expanding region of the stenosis. Young and Tsai performed their experiments on a straight tube, to account for change in pressure due to tapering in the vessel a convective term is introduced.

$$\Delta P = \alpha_1 Q + \alpha_2 Q^2 + \frac{\rho}{2}((\frac{Q}{A_0})^2 - (\frac{Q}{A_1})^2) \tag{2.12}$$

The severity of the stenosis is defined from the percentage reduction in diameter.

$$\text{Severity degree} = (1 - \frac{D_s}{D_0}) * 100\% \tag{2.13}$$

FFR is found from the ratio between the hyperemic flow in a stenotic vessel and the hypothetical flow if the vessel was healthy [37].

$$FFR = \frac{Q_{stenotic}}{Q_{healty}} = \frac{P_d - P_v}{P_a - P_v} \tag{2.14}$$

P_d being the mean distal coronary pressure, P_a the mean proximal coronary pressure and P_v the central venous pressure. With P_v assumed to be negligible equation 2.14 reduces to equation 1.1.

2.4 Machine learning

Machine learning is a branch of artificial intelligence. Arthur Samuel's definition of ML from 1959 reads, "[A] Field of study that gives computers the ability to learn without being explicitly programmed" [38]. An algorithm is constructed to learn from provided data, and then be able to solve similar problems based on experience. ML algorithms are categorized into two types of learning, unsupervised and supervised learning. In unsupervised learning, the algorithm is looking for patterns in the data. Supervised learning on the other hand, must learn to connect a set of features to prescribed labels [38].

2.4.1 Outline of the ML process

To successfully implement a ML algorithm, several steps must be performed. First, a database of training data must be established. For supervised training it must consist of enough data points containing the features and labels to be fed to the ML algorithm. The database is split into three sets, a training set, a validation set and a testing set. The training set is used to train the network, and the validation set is used to determine the hyper-parameters, to be introduced later. When the network is established the test set is used to evaluate the performance of the network. It is important that the database is sufficiently large to provide data for all three processes. A problem that might arise if the database it too small is overfitting. That means that the model corresponds too closely to the particular set of data, and may therefore fail to represent general behaviour of the real model. Thos can also happen if the network is too complex. The validation set is used to tune the hyper-parameters to avoid overfitting the hyper-parameters to the test set. This makes the performance measure reliable since the ML algorithm has not been presented to this database before [39].

Secondly, features must be extracted from the training data. Each example must contain all the features. The features must contain enough information for the ML algorithm to be able to solve the problem. Underfitting can be a result from a model trained with too few examples. The model does not have enough data to represent the problem [39].

The third step is to construct a ML model suitable for the problem. The model must be able to construct the relationship between the features and the label. There are many types of models to choose from, suited for different problems like image classification, analyzing numerical data or voice recognition. Some of the possible models are clustering,

decision trees, Bayesian networks and neural networks. The neural networks can have different structures, as deep and deep or shallow structure, convolutional networks and fully connected networks.

After the ML model has been constructed the training can start. During training, the ML model is gradually optimized. It learns from the provided training data, to be able to make predictions.

When the ML model is trained, it must be evaluated. The accuracy of the predicted results is used to evaluate the trained models performance. To obtain the accuracy, the predicted results are compared to the associated label. When the model's accuracy is inside the acceptable tolerance limit, it can be used to make predictions. These steps are summarized below.

1. Establish a training database
2. Feature extraction
3. Select a ML model
4. Train ML model
5. Evaluate performance of the ML model
6. Use the ML algorithm to make predictions

2.4.2 Deep neural network

Neural networks are inspired by the human brain. The neural network algorithm is programmed to mimic the neurons in the brain, originally inspired by the work of Warren McCulloch and Walter Pitts [40]. The network is built up from artificial neurons. Each neuron i, with their respective bias b_i, takes a set of inputs $x_j \in X$, with their corresponding weight w_{ij}. The output from a neuron y_i is found through the relation,

$$y_i = f(b_i + \sum_{j=1}^{n} w_{ij} x_j) \tag{2.15}$$

where f is the neurons activation function. Today, one of the most common artificial neurons is the sigmoid neuron [39]. The sigmoid activation function is defined in Equation

2.16.

$$f = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.16a}$$

$$z = b_i + \sum_{j=1}^{n} w_{ij} x_j \tag{2.16b}$$

To optimize the model, a cost function is computed based on the error of the model. By minimizing the cost function, the performance of the ML model improves. A common optimization technique for minimizing the cost function is gradient descent. An iterative method, starting from an initial guess of the gradient of the cost function is used to determine in which direction the model parameters are updated [41]. Given the cost function $C(v)$, where $v = (w, b)^T$, the gradient descent algorithm can be written as

$$v^{(t)} \leftarrow v^{(t-1)} - \eta \frac{\partial C(z_t, v)}{\partial v} \tag{2.17}$$

where η is the learning rate, a small positive number, telling the optimizing algorithm how fast to update the weights and biases at each iteration, z_t is an example sampled at iteration t. The batch size determines how many examples that are sampled at each iteration.

The deep neural network is built up from layers of neurons. The first layer is called the input layer, and the number of nodes corresponds to the number of input features. The last layer is the output layer and the layers in-between are referred to as the hidden layers. Figure 2.3 shows a fully connected, deep neural network with 4 hidden layers.

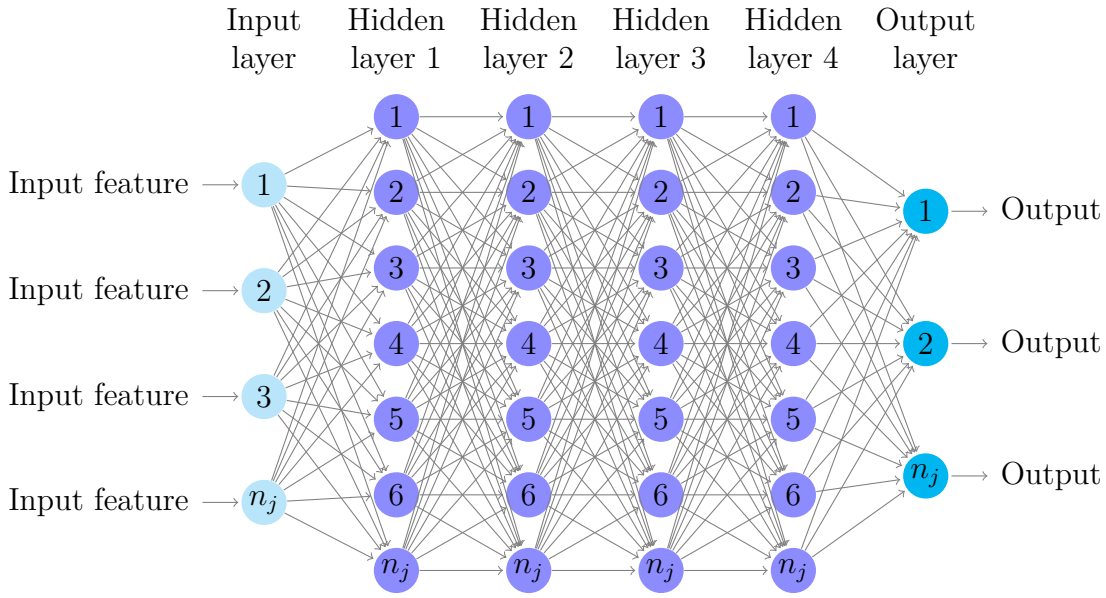


Figure 2.3: Deep neural network architecture.

2.4.3 Optimization of hyper-parameters

There are unlimited possibilities when constructing the ML model. The model will include several hyper-parameters, that is variables that must be set before training data is presented to the algorithm. The hyper-parameter choices affect the performance of the model, so it is important to tune the hyper-parameters well to obtain the best possible performance of the ML model. Hyper-parameters can be distinguished into two groups. In the first group, the hyper-parameters associated with the optimizer, as the learning rate, the batch size and number of training iterations are located. The second group contains hyper-parameters connected to the model itself, as the architecture of the model, type of neuron, weight initialization and pre-processing of the training data.

The tuning of hyper-parameters is a complicated process. One of the challenges is that it is computational expensive to perform many training sessions in the search for the optimal combinations of hyper-parameters. A lot of research has been done to come up with recommendations on how to tune hyper-parameters [42, 43]. Unfortunately, some of them are contradictory, and the best practice vary depending on the problem at hand. Here some recommendations found in "Practical recommendations for gradient-based training of deep architectures", by Y. Bengio [44] are presented:

- **Learning rate**: as mentioned earlier the learning rate determines how much the weights and biases are updated at each iteration. It is an important parameter, that should be prioritized, especially when stochastic gradient descent is used. $\eta = 0.01$ often works for multi-layer neural networks and can be used as a starting point.

- **Batch size**: determines how many training examples the ML model is presented before the weight and biases are updated. This hyper-parameter is important for training time and does not have a big effect on the test performance making it possible to optimize the batch size independent of the other parameters. Since it is not 100% independent, the batch size should also be re-optimized after the other hyper-parameters are optimized. It should be big enough to give a satisfactory representation of the full training set in each iteration.

- **Number of epochs**: defines how many training iterations the network performs over the whole database. It should be so big that the algorithm is not learning anymore. A good way to optimize the number of epochs is by "early stopping". When the cost is not decreasing anymore, the training should be programmed to stop after a prescribed number of epochs. The additional epochs are to avoid to stop at premature local minimums. Early stopping is a good way to avoid or reduce

overfitting.

- **Network architecture**: refers to the number of hidden layers as well as the depth of each layer. The depth of each layer is defined from the number of neurons, controlling the capacity of the layer. A layer must contain enough neurons to make generalizations, unnecessary depth mostly affects the computational time, not the performance of the network, especially if early stopping has been implemented. Studies have shown that layers with the same depth work similar or better than layers with decreasing or increasing depth [45]. Y. Bengoi also recommends that the first hidden layer contains more neurons than the number of input features, but he does not suggest any recommendations for the number of layers in the network.

- **Neurons**: when deciding for the type of neuron the shape of the activation function must be considered. The shape of the function describes how a change in the weights and biases cause a change in the neurons output. The perception neuron, sigmoid neuron and rectified linear unit activation functions are shown in Figure 2.4.

- **Weight initialization**: the initialization of weights is important to break the symmetry of the neurons in each hidden layer, if the symmetry is not broken the network will not be able to solve non-linearity. One approach is that the neurons with more input from earlier layers should have smaller weights. Biases are normally initialized to zero.

- **Preprocessing**: the preparation of the training data before it is presented to the ML algorithm. A common step is uniformization of each feature.

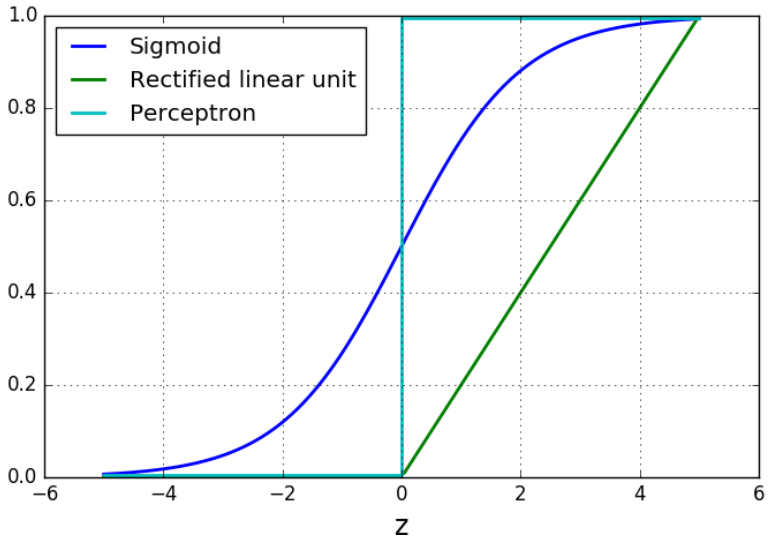


Figure 2.4: Activation functions for different neurons.

Different approaches as grid search, random search and manual search can be followed to determine the best values for hyper-parameters. The development in computational capacities has made it possible to perform systematically searches through a grid in hyper-parameter space. Advantages of this technique are that it is parallelizable and reproducible. Unfortunately, one grid search is often not enough, as the range for the hyper-parameters must adjusted to improve the performance. Therefore, a manual search is often the preferred method [43].

2.4.4 TensorFlow, a deep learning framework

Several deep learning frameworks are available for implementation of a deep learning network. The frameworks simplify the implementation and make it more efficient by providing a high-level Application Programming Interface (API). Premade algorithms and functions are provided to solve the underlaying mathematics of ML. These modules improve the user-friendliness and make ML available to a wider spectrum of programmers, that does not possess the necessary background in mathematics.

TensorFlow is an open-source machine learning framework developed by Google Brain team [46]. It provides interfaces for Python, C++, Java, Haskell, and Go. It has a flexible architecture and supports computation on CPUs, GPUs and TPUs.

3 Method

This chapter will follow the workflow of the implementation of a ML algorithm, presenting the methodology of each step. The workflow is presented in Figure 3.1, where the four main steps of the process are pre-processing, learning, evaluation and prediction.

There will be implemented two ML algorithms. One for the 1D segments along each vessel and one for the stenosis model. The post-processing step will combine the predictions for both models to compute the FFR_{ML} values for the coronary tree.

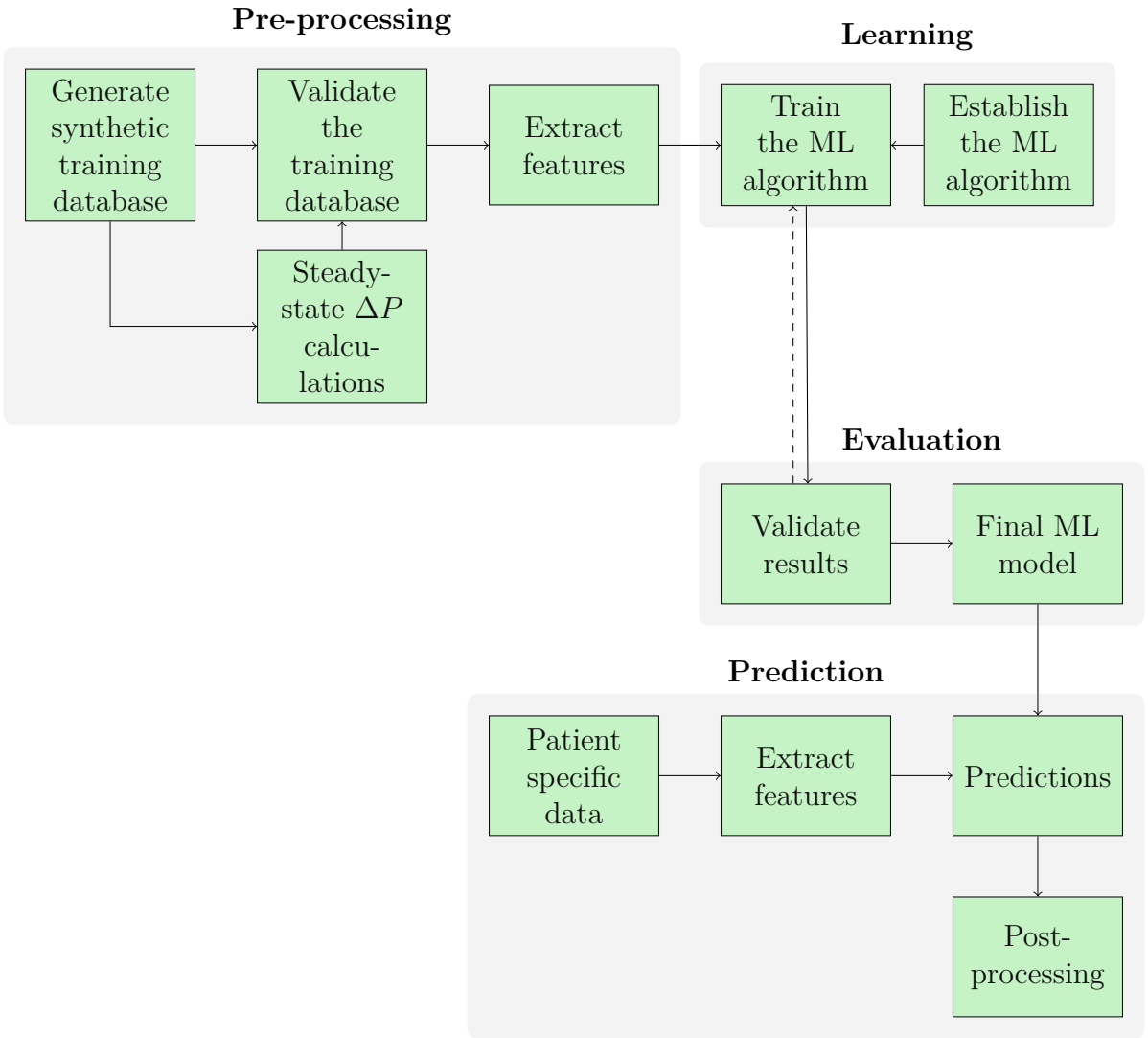


Figure 3.1: Workflow for implementation of a ML algorithm applied in the work.

3.1 Generate synthetic training database

A virtual population of coronary trees is constructed to provide training data for the ML algorithm. The virtual population is synthetically generated to contain 3000 coronary trees representing the LAD, LCx and RCA. A thorough literature search has been performed to generate the database. Anatomical data for the three arterial branches is retrieved from literature and sampled to represent anatomical variations, representative of patients with suspected CAD. The database does not distinguish between the three arteries.

3.1.1 Geometric parameters

The database was generated following the three-step algorithmic process presented by Itu et al. [17]. In the first step the coronary skeleton is initialized. The coronary skeleton consists of a main branch and two generation of side branches. The 1. generation of side branches are sampled between 2-5 and the 2. generation between $0-2$, both with a uniform distribution. The skeleton of a full coronary tree is seen in figure 3.2.

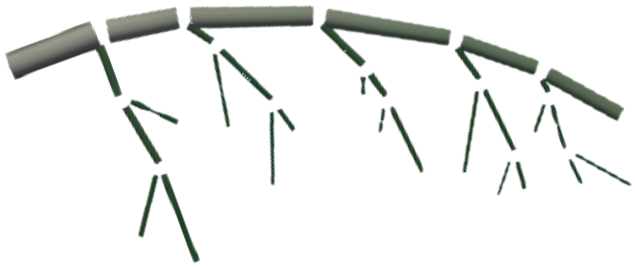


Figure 3.2: Skeleton of a coronary tree.

In the second step, each segment of the coronary skeleton is prescribed with geometrical parameters. The anatomical values prescribed are sampled in prespecified ranges derived from published literature. The geometric information defined in this step is the vessel length, vessel radius and the degree of tapering.

First, the root radius is specified. It is sampled with a uniform distribution, in the range $2-3.5$ mm. From Murray's law, the radius of the daughter vessels at each bifurcation is calculated. Equation 2.8 is combined with the radius ratio,

$$\text{Radius ratio} = \frac{r_{d2}}{r_{d1}} \tag{3.1}$$

where r_{d1} is the biggest vessel, used as the continuation of the main branch at each bifurcation. The radius ratio is sampled in different ranges for the main branching and bifurcations in the side branches.

The branch length is sampled with a uniform distribution in the ratio $1.5 - 4\,\text{cm}$. At last, the degree of tapering across each segment is defined, with a uniform distribution from $-20\% - 5\%$. All of the geometric parameters prescribed in step two are presented in Table 3.1. A schematic description is given in Figure 3.3

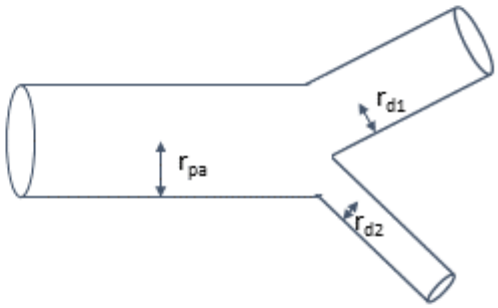


Figure 3.3: Geometric parameters at the bifurcation.

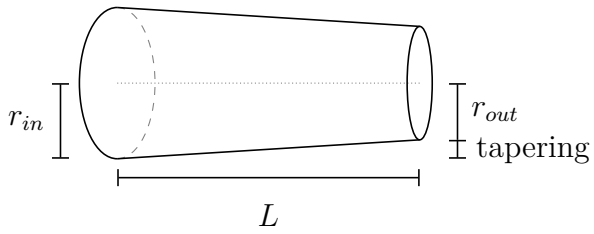


Figure 3.4: Geometric parameters of a healthy coronary vessel.

Table 3.1: Parameters with corresponding ranges used to generate synthetic coronary trees.

Parameter	Range
Main Branches	LAD, LCx and RCA
1. Generation of side branches	$2 - 5$
2. Generation of side branches	$0 - 2$
Vessel length [17]	$1.5 - 4$ cm
Root radius [47]	$2 - 3.5$ mm
Radius ratio [17]	Main branch, 0.35–0.45
	Side branch, 0.6–0.8
Degree of tapering [17]	-20% to +5% from top to bottom

In step three, stenoses are generated for the coronary tree and prescribed their geometric parameters. Stenoses are created in two steps. First $0 - 3$ stenoses are assigned in the main branches. Secondly, the first generation of side branches are assigned $0 - 1$

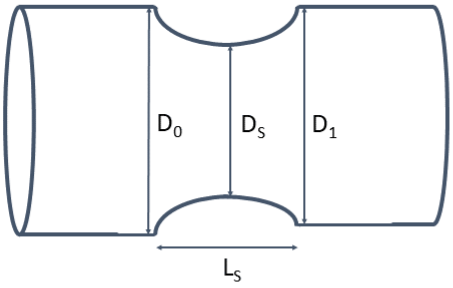


Figure 3.5: Geometric parameters of a stenosis.

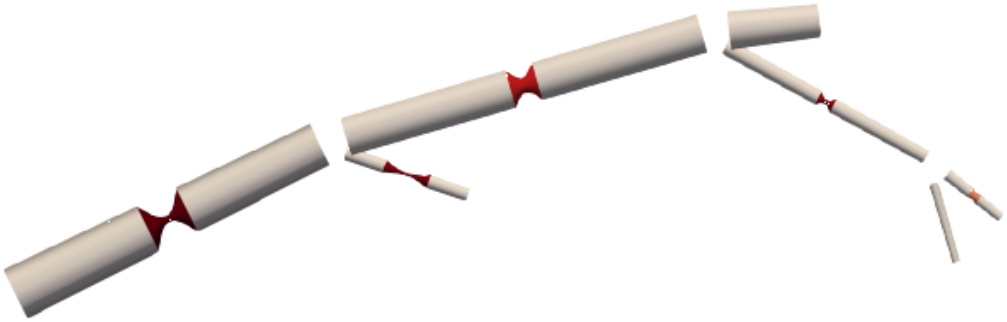


Figure 3.6: Coronary tree with stenoses.

stenoses. Both with a uniform distribution. No stenoses are assigned to the the third generation of vessels. A coronary tree with its respective stenoses is seen in Figure 3.6. For both the main branch and the first generation of side branches the stenoses are randomly distributed between the available segments. The geometric parameters described for each stenosis, are the total stenosis length, the severity of the stenosis, as defined in Equation 2.13, minimum radius and overall degree of tapering over the stenosis. The degree of tapering over the stenosis depends on the degree of tapering in the segment the stenosis is assigned to. All of the parameters prescribed for the stenoses are presented in Table 3.2, and a schematic description in figure 3.5. The stenosis is located at the center of each segment, splitting the vessel into two segments creating a stenotic junction. The stenosis length is subtracted from the vessel length.

3.1.2 Hemodynamic and mechanical features

In addition to the geometric parameters, hemodynamic and mechanical features must be defined. From the total peripheral resistance and coronary flow rate, the flow rate through

Table 3.2: Parameters with corresponding ranges used to generate stenoses.

Parameter	Range
Number of stenoses	Main branch, $0-3$ Side branch, $0-1$
Total stenosis length [48]	$1.3-8.9$ mm
Stenosis severity	$10-90\%$

each segment is approximated.

The coronary flow rate is distributed from the aorta to the left and right branches given in equation 3.2. γ_k^j is sampled from $40-60\%$, to include flow distribution to the RCx and the LM for both right and left dominance. The flow distribution is found in the work of Sakamoto et al [49], namely $Q_{baseline} = \langle 1.5-2.25 \rangle$ (or similar). The synthetically generated database does not distinguish between the different branches.

When the coronary flow rate is determined, the total peripheral resistance R_{tot} and total arterial compliance C_{tot} for each vessel segment are computed from the following relations [50].

The total baseline coronary flow q_{cor} is computed as 4.5% of the normal cardiac output q_{CO}. Under baseline conditions, $q_{CO} = 5000\,\mathrm{ml/min}$ [2]. The baseline coronary flow is distributed to the left and right branch from this relation:

$$q_{cor}^j = \gamma_k^j q_{cor} \tag{3.2}$$

where $j = \{LM, RCA\}$, $k = \{Rightdominant, Leftdominant\}$ and γ_k^j defines the distribution.

The total peripheral resistance and the total arterial compliance is distributed among outlets using Murray's law 2.8.

$$R_j = \frac{\sum_{i=1}^{N_{out}} r_i^3}{r_j^3} R_{tot} \tag{3.3}$$

With $R_{tot} = \frac{MAP - P_v}{q_{cor}^j}$, and the mean arterial pressure, $MAP = 100$ mmHg.

$$C_j = \frac{r_j^3}{\sum_{i=1}^{N_{out}} r_i^3} C_{tot} \tag{3.4}$$

Where $C_{tot} = \frac{q_{cor}^j}{q_{CO}} * 1.7$ mmHg.

To account for the effect of hyperemic conditions, a new the total peripheral resistance is estimated from the total peripheral resistance at baseline conditions, $R_{tot,hyp} = \frac{R_{tot,bln}}{3}$.

3.2 Steady-state ΔP calculations

For each of the 3000 coronary trees in the virtual population, the hyperbolic PDE's for compliant vessels, given in Equation 2.4a, 2.4b, 2.6a and 2.6b must be solved. The following material parameters were used for the blood: $\rho = 1.50\,\mathrm{g/cm^2}$, $\mu = 0.035\,\mathrm{g/(cm \cdot s)}$, $P_{in} = 100\,\mathrm{mmHg}$ and the venous pressure, $P_d = 5\,\mathrm{mmHg}$. The system is solved with a steady state solver, developed by the Biomechanics group at the Department of Structural Engineering, NTNU.

3.2.1 Numerical solver

For the coronary trees the arterial junctions are treated as bifurcations, and the stenotic regions are treated as junctions with $N = 2$. Where ΔP in Equation 2.6a is given by Young and Tsai's stenosis model 2.3.4. The pressure distribution in the 1D network, with resistance $R_{out,j}$ at the outlets are then uniquely defined by the given inlet pressure and the outlet flows. Assuming rigid domains, the steady state system can be solved as a system of nonlinear algebraic equations for the number of unknown outlet flows. With M outlet flows, there is $M - 1$ coupling equations. The last equation comes from the inlet boundary condition. Giving the unknowns $\mathbf{x} = [Q_{out,1}, Q_{out,j}, \ldots, Q_{out,M}]$, that has to satisfy the equations $\mathbf{f} = [f_{out,1}, f_{out,j}, \ldots, f_{out,M}]$.

From the initial guess of the outlet flows, the system is solved iterative until convergence, by first applying conservation of mass. Secondly, the pressure at the outlets are calculated:

$$P_{out,j} = P_d + R_{out,j} Q_{out,j} \tag{3.5}$$

where P_d refers to the daughter vessel. The pressure drop along each vessel is found from equation 2.4a, where the integral is estimated using the trapezoidal rule implemented in SciPy [51]. The coronary tree is traversed from the outlets to the inlet to calculate the pressure distribution, with

$$P_{in} = P_{out} + \Delta P \tag{3.6}$$

When the inlet pressure is calculated for all segments in the coronary tree, the residual equations f are evaluated.

The flow is matched to the assigned inlet flow and the peripheral resistance is updated:

$$R_i = \max(10^{-5}, R_{i-1} * (1 - \alpha(\frac{Q_{match} - Q_{solve}}{Q_{match}})) \tag{3.7}$$

where $\alpha = 0.1$, Q_{match} is the given flow at the inlet, Q_{solve} is the flow at the previous iteration.

The system assumes an axisymmetric cross-sectional area, with negligible velocity in the radial direction. The radial dependency of the velocity is assumed known. In this thesis it is given the velocity profile is given by the power law presented in equation 2.3. ξ defines the shape of the velocity profile. In this work $\xi = 2$, giving an parabolic velocity profile known as Poiseuille flow. This value was used to model flow in the carotid artery in [52].

3.3 Validate the training database

Before the database is presented to the ML model it has to be validated to ensure the physiological validity and variation of the coronary trees. Controlled features are:

1. Total number of segments and stenoses
2. Root radius, and its propagation to the outlets of the coronary tree
3. Distribution of pressure drop for the segments and stenosis

The ML algorithm requires enough training examples in the whole range of the features to be able to learn the behaviour. In addition, the pressure drop from the database will be compared to the Poiseuille equation and Young and Tsai's stenosis model respectively. The Poiseuille equation yields,

$$\Delta P = \frac{8\mu LQ}{\pi r^4}. \tag{3.8}$$

3.4 Extract features

Features are extracted from the synthetically generated database for training of both 1D segments and stenotic regions. Each coronary tree is described along the centerline, by

equally spaced nodes, $dz \approx 0.5\,\text{mm}$. Stenotic regions are defined at the nodes, while 1D segments are found in-between nodes.

1D segments The ML algorithm is trained to find ΔP for every segment defined by two nodes along the centerline of the coronary tree. One training example consist of four input features and their respective label, Q [m^3/s], $A_0[\text{m}^2]$, $A_1[\text{m}^2]$, $L[m]$ and ΔP [Pa].

Stenotic regions For stenotic regions, the ML model is trained to solve Young and Tsai's stenosis model, presented in section 2.3.4. The five features Q [m^3/s], D_0 [m], Stenosis degree[%], D_1 [m], L_s [m], and their label, ΔP [Pa] are extracted for each stenosis in the database.

3.5 Establish the ML algorithm

To solve the problem at hand, a deep DNN-regressor was implemented in Python, using Tensorflow[2](Kan jeg henvise slik?). The DNN-regressor is built to take hyper-parameters and training data as inputs, prepare the training data, build and train the neural network. When the training is done, it will evaluate the performance with the validation set.

First, the training data is randomly split into one training and one validation set. The split is set to take the same random state each time. The features are scaled with the StandardScaler in the scikit-learn, ML package in Python [53]. The features are scaled to a unit variance.

Next, the network is built with Sigmoid neurons, from the architecture given as an input argument. The cost function is defined as the mean squared error [39].

$$C(w, b) = \frac{1}{2n} \sum_x \|y(x) - a\|^2 \tag{3.9}$$

$y(x)$ is the desired output, given the input vector x, a is the output, and n is the total number of inputs.

The AdagradOptimizer is adopted to optimize the cost function. It is a first-order optimizer, adapted from the gradient decent optimizer, presented in Equation 2.17. In the AdagradOptimizer the learning rate is updated for each variable and at each step, and sub-gradients are computed instead of gradients. The learning rates are updated through

[2]Adapted from: `http://www.science.smith.edu/dftwiki/index.php/Tutorial:_Playing_with_the_Boston_Housing_Data`

the following relations:

$$v_i^{(t+1)} = v_i^{(t)} - \eta_i^{(t)} \frac{\partial C^{(t)}}{\partial v_i} \tag{3.10a}$$

$$\eta_i^{(t)} = \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} v_i^{(\tau)2} + \epsilon}} \tag{3.10b}$$

where μ is the learning rate, $\eta_i^{(t)}$ and $\eta_i^{(t+1)}$ are the values for the i^{th} parameter at iterations t and t +1 respectively [54].

Before the learning can begin, all internal variables are initialized. The model is provided with two stopping criteria, the maximum number of epochs and a tolerance limit for the successive cost function error for two following epochs. Both are defined in the input arguments. When the model reaches one of these criterion's, the network's variables are saved to disk and prediction performance is computed from the validation set. The model performance is measured from the mean squared error, standard deviation and the coefficient of determination(R^2). To run the DNN-regressor, the Tensorflow environment must be activated.

3.6 Train and validate the ML algorithm

To obtain a good ML model the hyper-parameters must be optimized. A grid search is performed to give a thorough understanding of the effect of each parameter. Four hyper-parameters are included in the grid, with their respective range presented in Table 3.3, resulting in a total of 180 training sessions.

Validation data set size, batch size and maximum number of epochs depends on the size of the training data set. For the 1D sections validation data set size is set to 15% of the data set, giving a training data set consisting of 175599 examples. By reducing the size of the training data, computational time is reduced. Batch size is sampled in the range of approximately $1 - 10\%$ of the training data. The stenosis database contains 13200 examples. The maximum number of epochs is set to 50000 epochs, a compromise between computational time and training speed. From preliminary training sessions it was seen that training had slowed down after 50000 epochs, giving a good understanding of the network's performance.

The data set consisting of stenoses is considerably smaller than the one for 1D segments. The validation data set is 20% of the data set. Batch size is set in the same range as for the 1D segments. The preliminary training showed that after 50000 epochs, training had not slowed down yet. Hence the maximum limit of epochs was increased to

75000.

The range for each hyper-parameter is determined based on the recommendations presented in section 2.4.3 and results from preliminary training sessions. From the preliminary training sessions it was seen that a learning rate of lower than 0.01 the recommended starting point, was not a good match and was therefor excluded from the grid.

Table 3.3: Hyper-parameter selection for the grid search.

Hyper-parameter	**Test values**
Learning rate	0.01
	0.1
	0.25
Batch size, 1D sections	1000
	2500
	5000
	10000
Batch size, stenotic region	100
	500
	1000
	2000
Numbers of hidden layers	2, 3 or 4
Depth of each layer	10
	20
	30
	50
	100

3.7 Final ML model

From the grid search, performed on a preliminary database the best performing network is found.

Then a network with the same hyper-parameters is trained with the final training database

3.8 Patient data

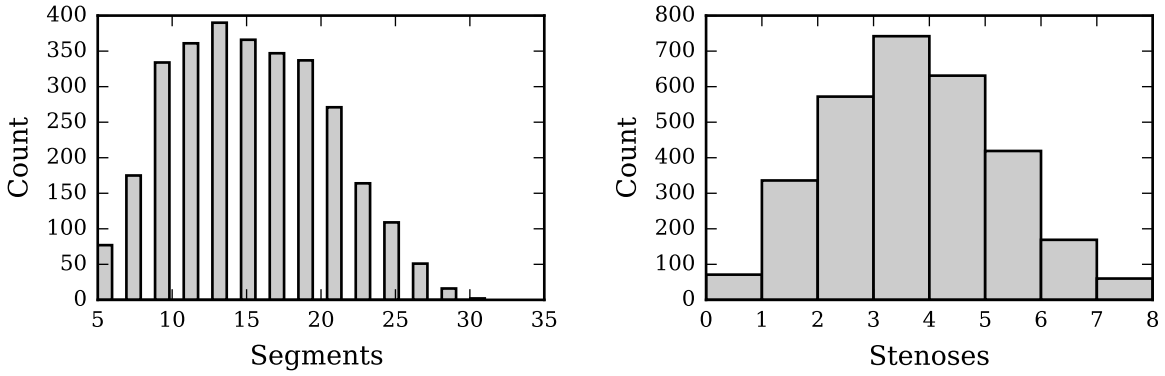
Patient data was obtained from CT images and clinical measurements from patients participation in the pilot study performed by the Biomechanics group at NTNU. As a part of the study, the CT images was segmented. The patient specific coronary trees was solved

with the same steady state solver as the synthetically generated coronary trees. In total, segmented geometries from 13 patients was solved.

4 Results

4.1 Validation of training database

The training database is validated before it is presented to the ML model, as mentioned in section 3.3. Each coronary tree contains from 5 to 31 vessels. The distribution is seen in Figure 4.1a. The maximum possible numbers of stenoses are 13, which requires a full continuation of the main branch and minimum one bifurcation for each one of the side branches. The distribution of stenoses are seen in Figure 4.1b and the final distribution of the total number of segments in figure 4.1a.



(a) Segments per coronary tree

(b) Stenoses per coronary tree

Figure 4.1: Segments and stenoses per coronary tree.

From the initial root radius, the radius of each segment is distributed dependent on Murray's law. In Figure 4.2 the distribution for root radius, the continuation of the main branch, the 1. generation of branches and the 2. generation of branches are seen. The radius of the distal branches are within the physical range reported in the study by Dodge et al. [55].

4.1.1 Steady state FFR results

The ML algorithm is trained to find the pressure drop over a stenosis and along the vessels. The variation for the 1D segments along the vessels is presented in Figure 4.3. The majority of the data sets represent smaller pressure drops than -0.5 mmHg.

The pressure drop distribution for the stenosis database is seen in Figure 4.4. The pressure drops are in the range $0 - 95$ mmHg, with an accumulation of training examples

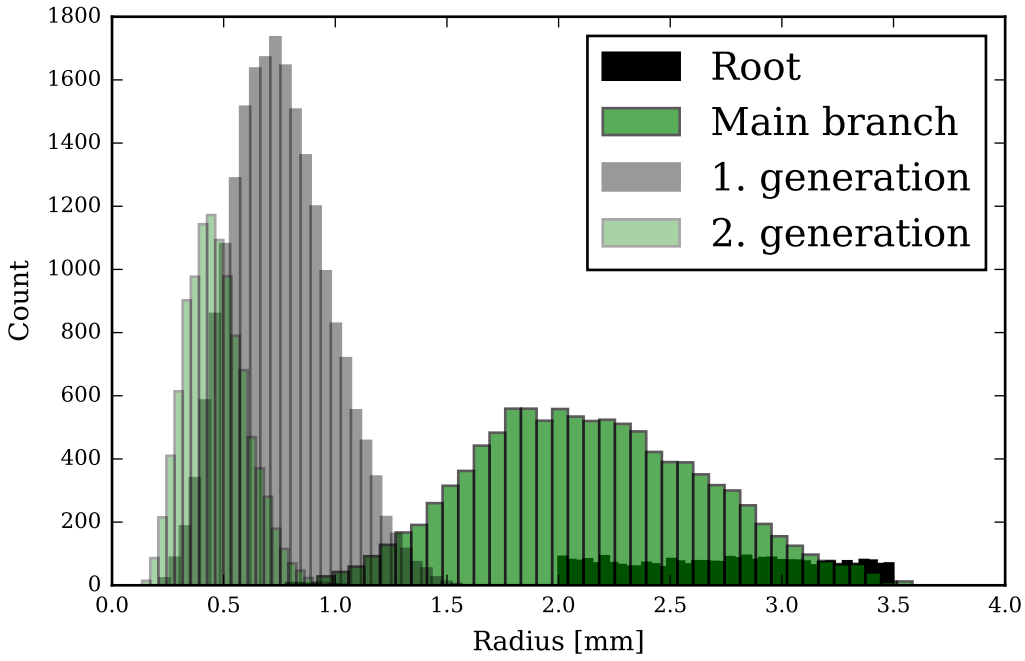


Figure 4.2: Radius distribution for the inlet of each segment.

below 20 mmHg.

The last control is to compare the examples in both databases to their respective models, the Poiseuille equation and Young and Tsai's stenosis model. Figure 4.5 shows a perfect match for both models. As this is the first step towards a fully developed ML model to predict the pressure drop in coronary arteries, it gives an insight in the complexity of the problem the ML model has to solve.

4.1.2 Feature distribution

The feature distribution for the two final databases are visualized in Figure 4.6 and 4.7.

The grid search was performed with a preliminary database. The parameters that distinguish the preliminary database from the final database are the root radius, vessel length, stenosis length and number of stenoses in side branches, and in the final database the stenoses are restricted to the first generation of side branches. The changes is presented in Table 4.1. The preliminary database is visualized in Appendix (A).

Because of the time restriction there was not time to perform a new grid search with the final database.

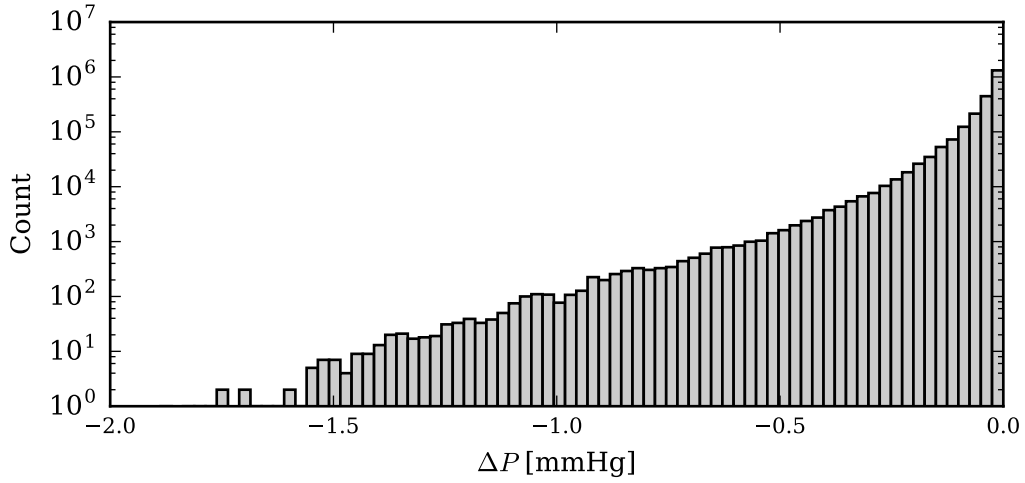


Figure 4.3: Pressure drop distribution for the 1D segments database.

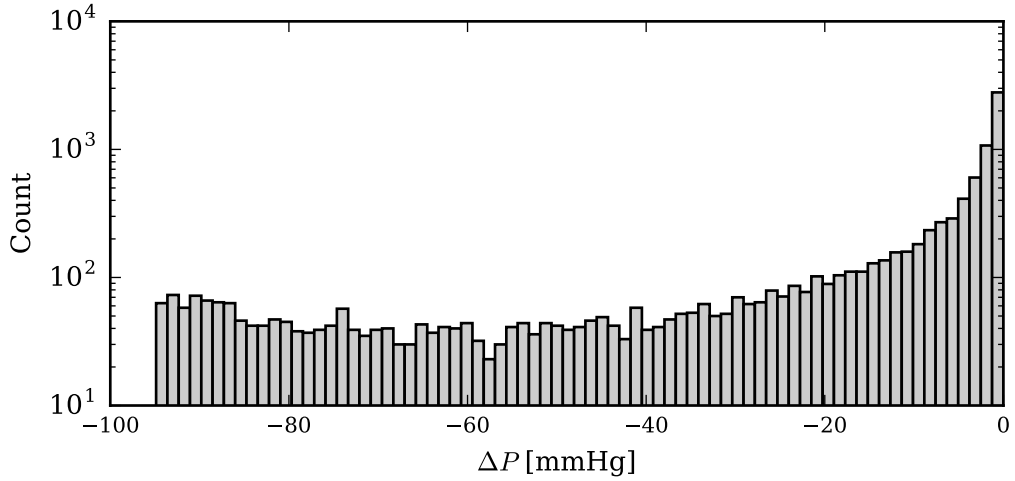
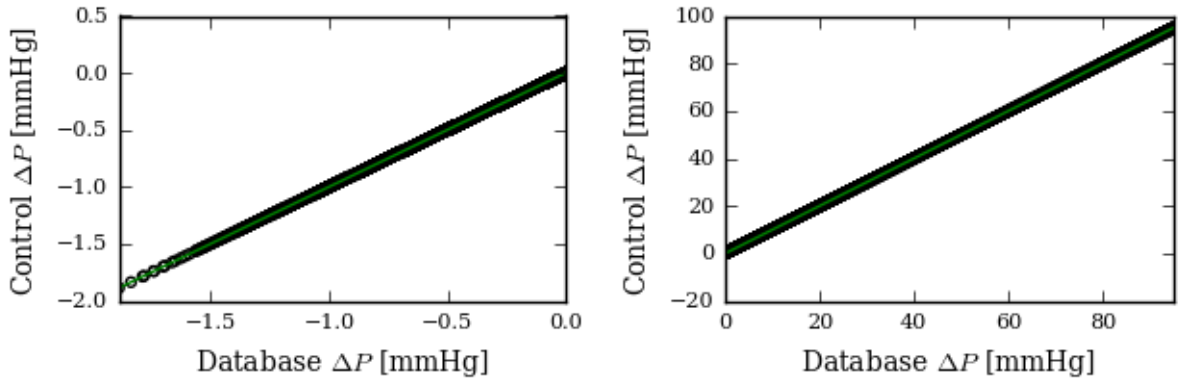


Figure 4.4: Pressure drop distribution for the stenosis database.



(a) 1D segments database

(b) Stenosis database

Figure 4.5: Control of ΔP.

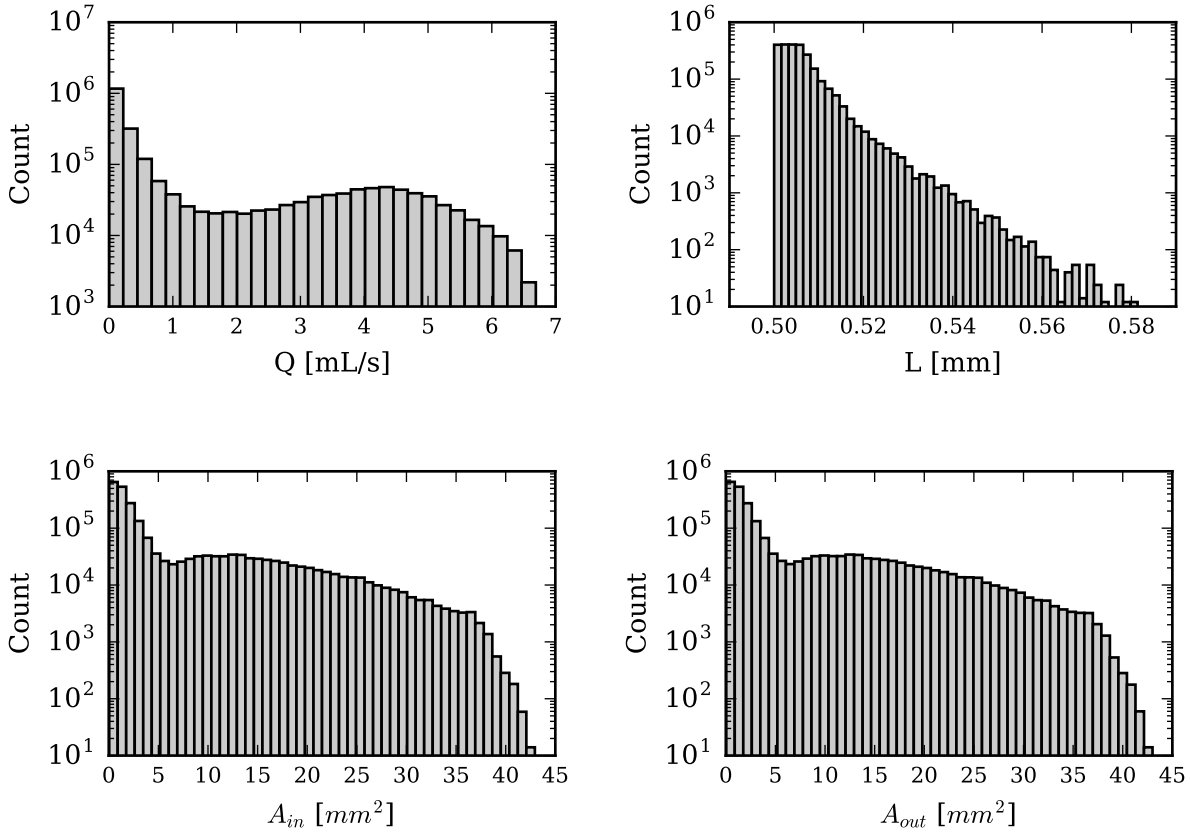


Figure 4.6: Features for the 1D-ML model.

Table 4.1: Parameters with corresponding ranges used to generate synthetic coronary trees.

Parameter	Final database	Preliminary database
Vessel length [mm]	Uniform distribution All branches: $15 - 40$	Normal distribution Main branch: 20.7 ± 13.5 (51.5/2.4) Side branch: 10.5 ± 6.4 (39.8/1.2)
Root radius	Uniform distribution	Normal distribution
Number of stenoses in the side branch	$0 - 2$	$0 - 1$
Stenosis length [mm]	$1.3 - 8.9$	5.1 ± 3.8

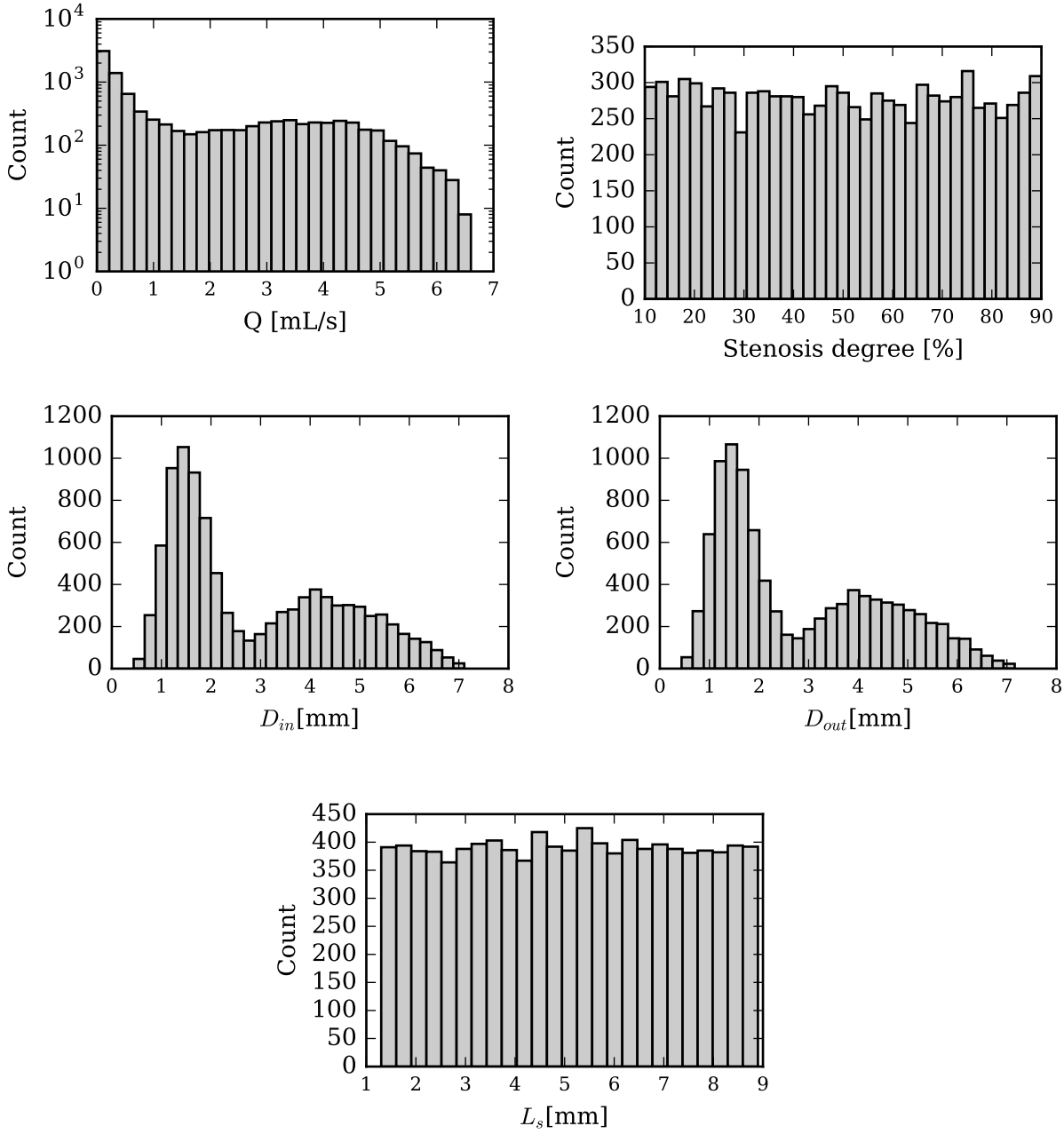


Figure 4.7: Features for the stenosis-ML model.

4.2 Grid search results

Two different sets of hyper-parameters are tuned to optimize training of the 1D sections and the stenotic region. For both of them a grid search is performed, with a total of 180 combinations. The range of each hyper-parameter is given in Table 3.3. In addition to the hyper-parameter's in the grid, the impact from the number of epochs, and the amount of training data fed to the 1D model will be presented.

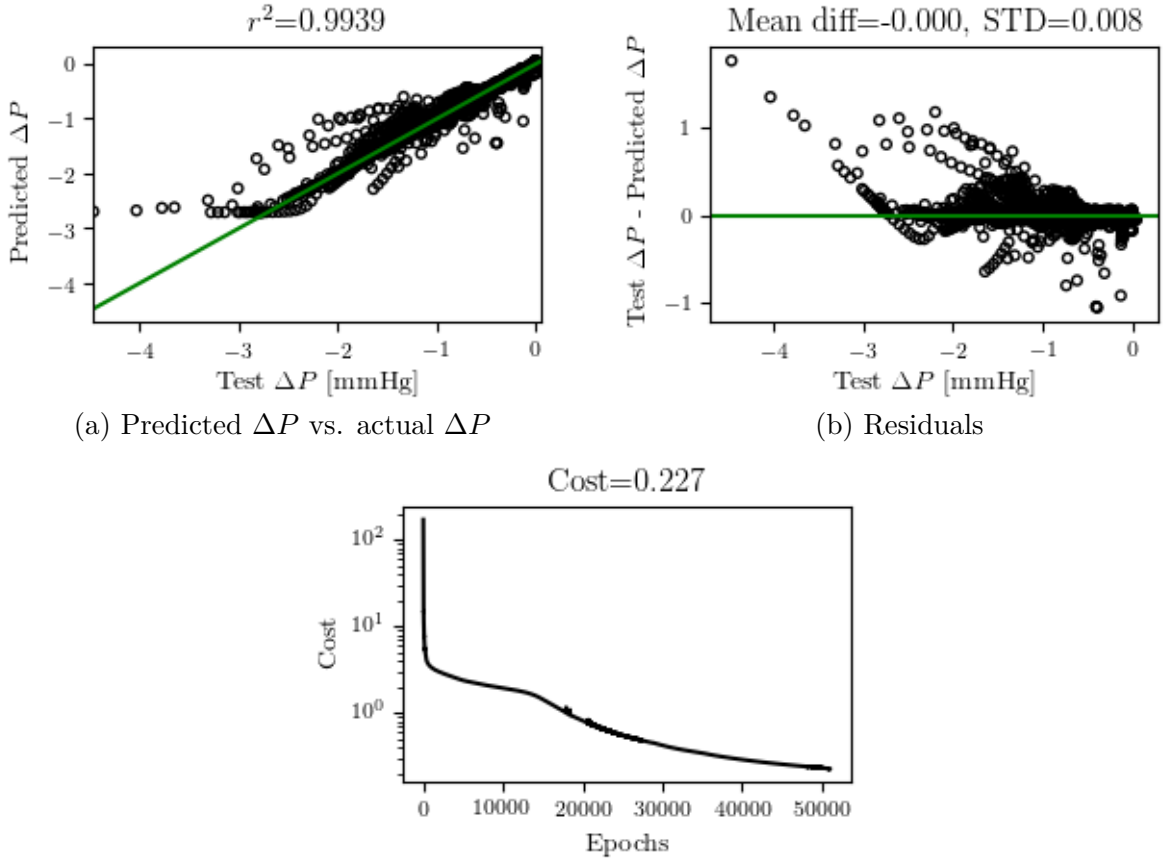
The models' ability to make predictions are evaluated based on the test cost, the mean squared error of the predicted values on the test set. The coefficient of determination (r^2), Standard Deviation (STD) and the mean difference for the test set. The final cost of the model during training, the cost for the last batch presented to the ML algorithm before the training was stopped is stated above cost v. epochs plots.

First the best results from each of the two different models will be presented, then the parameter specific results from the grid search is presented.

4.2.1 1D-ML model

The network with the best prediction accuracy in the 1D-model grid search is a four layer network, with 50 neurons in each layers, the batch size is 1000 and the learning rate is 0.25. With a coefficient of determination of 0.994, STD of $7.81 * 10^{-3}$ mmHg and a mean difference of $-2.73 * 10^{-5}$ mmHg. The final cost during training is 0.227, but still declining as seen in Figure 4.8. The test cost is 1.08 This is case number 119 in the grid search, and is the network with the best coefficient of determination, STD and test cost out of all 180 cases. The biggest prediction error is for the examples with the highest pressure drop. It is seen that the model does not predict any pressure drops over 3 mmHg. A full overview of the grid search results are listed in Appendix (B), and the presented results in this section for the 1D-ML model are summarized in Table 4.2.

Case 131 in the grid has the lowest mean difference of $3.51 * 10^{-7}$ mmHg. The residuals is seen in Figure 4.9, it is evident that the model is not able to predict any pressure drops over approximately 0.75 mmHg. The low mean difference comes from the high amount of data points located within 0.5 mmHg pressure drop.



(a) Predicted ΔP vs. actual ΔP

(b) Residuals

(c) Cost vs. epochs

Figure 4.8: Results from case 119, the best network on the 1D-ML model.

Table 4.2: Selection of results from the grid search, 1D-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
55	0.1	1,000	100 100 100	2.795	0.984	$3.542{\cdot}10^{-4}$	$1.254{\cdot}10^{-2}$
74	0.25	20,000	50 50 50 50	3.365	0.981	$5.01{\cdot}10^{-4}$	$1.375{\cdot}10^{-2}$
89	0.25	5,000	50 50 50 50	2.597	0.985	$1.144{\cdot}10^{-4}$	$1.209{\cdot}10^{-2}$
104	0.25	10,000	50 50 50 50	2.011	0.989	$8.499{\cdot}10^{-4}$	$1.06{\cdot}10^{-2}$
109	0.25	1,000	50 50	3.986	0.977	$4.091{\cdot}10^{-4}$	$1.497{\cdot}10^{-2}$
114	0.25	1,000	50 50 50	1.882	0.989	$7.192{\cdot}10^{-4}$	$1.027{\cdot}10^{-2}$
115	0.25	1,000	100 100 100	1.28	0.993	$3.431{\cdot}10^{-4}$	$8.481{\cdot}10^{-3}$
116	0.25	1,000	10 10 10 10	3.845	0.978	$3.455{\cdot}10^{-4}$	$1.47{\cdot}10^{-2}$
117	0.25	1,000	20 20 20 20	3.279	0.981	$1.468{\cdot}10^{-4}$	$1.358{\cdot}10^{-2}$
118	0.25	1,000	30 30 30 30	2.73	0.985	$6.315{\cdot}10^{-5}$	$1.239{\cdot}10^{-2}$
119	0.25	1,000	50 50 50 50	1.085	0.994	$-2.725{\cdot}10^{-5}$	$7.813{\cdot}10^{-3}$
120	0.25	1,000	100 100 100 100	1.392	0.992	$-1.854{\cdot}10^{-4}$	$8.847{\cdot}10^{-3}$
131	$1 \cdot 10^{-2}$	20,000	10 10 10 10	35.367	0.8	$3.505{\cdot}10^{-7}$	$4.461{\cdot}10^{-2}$
175	$1 \cdot 10^{-2}$	1,000	100 100 100	4.759	0.973	$2.111{\cdot}10^{-4}$	$1.636{\cdot}10^{-2}$

LR = learning rate, BS = batch size, r^2 = coefficient of determination, STD = standard deviation

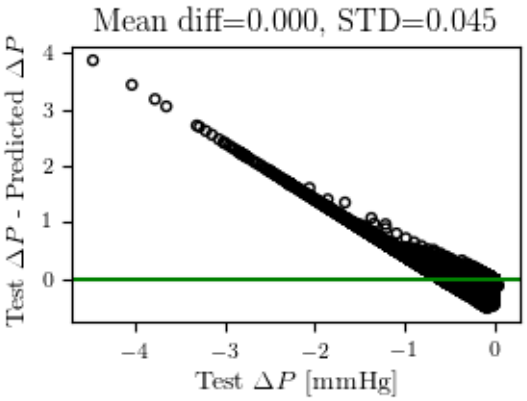


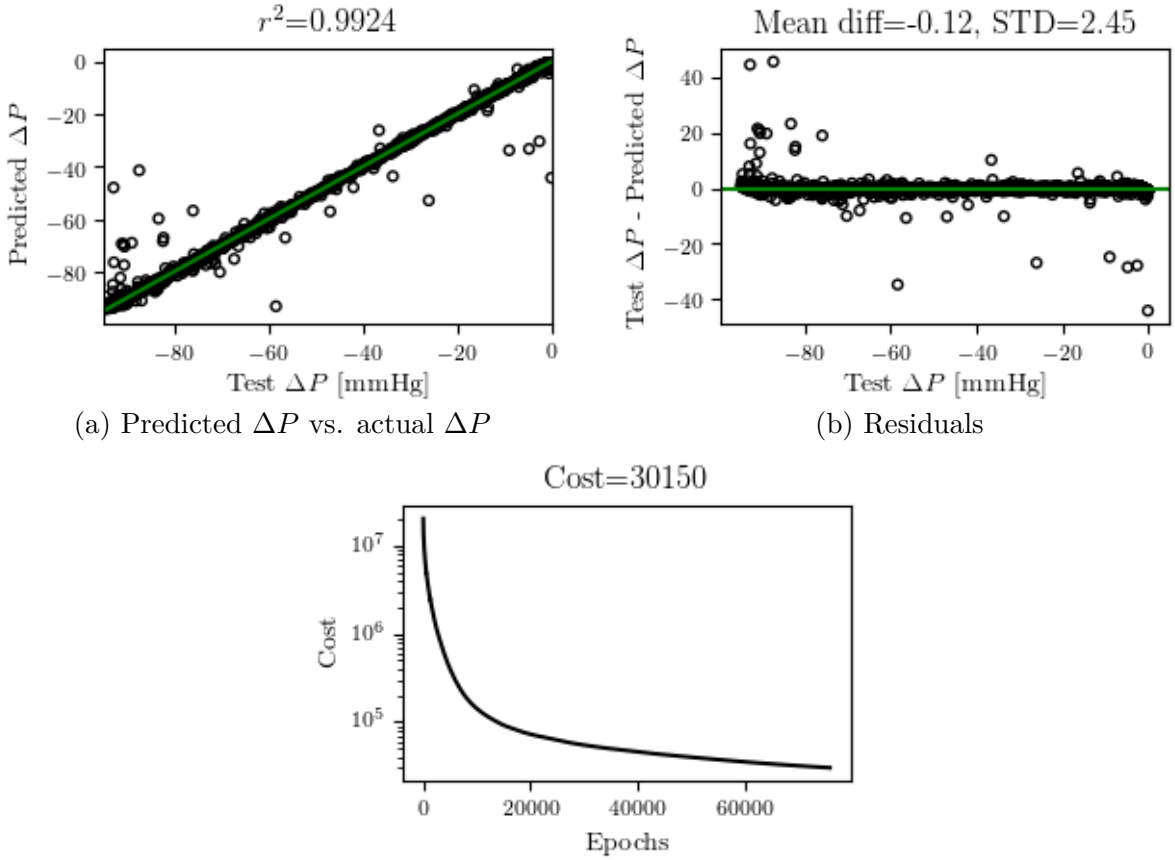
Figure 4.9: Residuals for case 131, stenosis model.

4.2.2 Stenosis-ML model

For the stenosis-ML model case 84 in the grid has the best score for the coefficient of determination, STD and test cost. The coefficient of determination is 0.992, STD is 2.45 mmHg and the test cost is $1.07 * 10^2$. The mean difference is -0.12 mmHg. This network has three hidden layers, with 50 neurons in each. The learning rate is 0.25 and the batch size is 100. In figure 4.10a and 4.10b the predicted ΔP vs. actual ΔP and the prediction residuals are shown. Figure 4.10c shows the development of the cost vs epochs, where the final cost is 30150. The cost is still improving, but the improvement rate has slowed down. It is noted that this is not the network that has the loves final cost, which is case 85 with a final cost of 5622 4.11.

The network that has the best mean difference is a two layer network, with 100 neurons in each. The learning rate is 0.25, the same as the best network, while the batch size is 500. The mean difference is $-6.1 * 10^-3$ mmHg, while the STD is 3.8 mmHg. From Figure 4.12a it is seen that there is more predictions that deviate from the regression line than for the best network, and $r^2 = 0.982$. The model struggles most with the stenoses with high or small pressure drops 4.12b. The test cost of the model is $2.56 * 10^5$, and the cost development is seen in 4.12c.

The presented results in this section for the stenosis-ML model are summarized in Table 4.3, and a full overview of the grid search results is listed in Appendix (C).



(a) Predicted ΔP vs. actual ΔP

(b) Residuals

(c) Cost vs. epochs

Figure 4.10: Results from case 84, the best network on the stenosis-ML model.

Table 4.3: Results from grid search, Stenosis-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
24	0.1	100	50 50 50	$1.875{\cdot}10^{5}$	0.987	$6.24{\cdot}10^{-2}$	3.247
69	0.25	1,000	50 50 50	$3.024{\cdot}10^{5}$	0.979	0.591	4.082
79	0.25	100	50 50	$2.513{\cdot}10^{5}$	0.982	$-1.274{\cdot}10^{-2}$	3.76
84	0.25	100	50 50 50	$1.072{\cdot}10^{5}$	0.992	−0.121	2.453
89	0.25	100	50 50 50 50	$1.412{\cdot}10^{7}$	$-1.398{\cdot}10^{-5}$	0.105	28.188
95	0.25	500	100 100	$2.563{\cdot}10^{5}$	0.982	$-6.132{\cdot}10^{-3}$	3.797
99	0.25	500	50 50 50	$1.632{\cdot}10^{5}$	0.988	−0.232	3.022
106	0.25	2,000	10 10	$9.85{\cdot}10^{6}$	0.303	10.219	21.208
107	0.25	2,000	20 20	$4.861{\cdot}10^{6}$	0.656	6.034	15.397
108	0.25	2,000	30 30	$2.419{\cdot}10^{6}$	0.829	3.553	11.111
109	0.25	2,000	50 50	$7.565{\cdot}10^{5}$	0.946	1.39	6.374
110	0.25	2,000	100 100	$3.117{\cdot}10^{5}$	0.978	0.388	4.17
114	0.25	2,000	50 50 50	$6.061{\cdot}10^{5}$	0.957	1.353	5.681
144	$1 \cdot 10^{-2}$	100	50 50 50	$1.098{\cdot}10^{7}$	0.222	11.524	22.024

LR = learning rate, BS = batch size, r^2 = coefficient of determination, STD = standard deviation

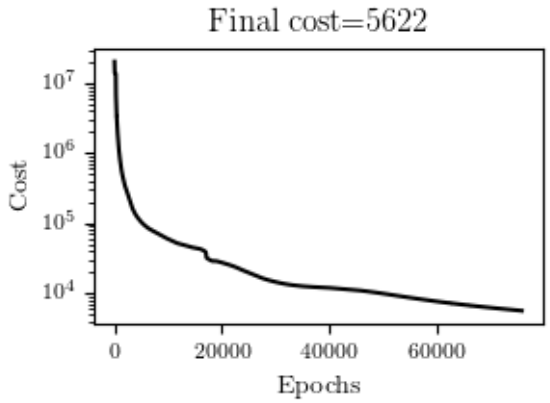
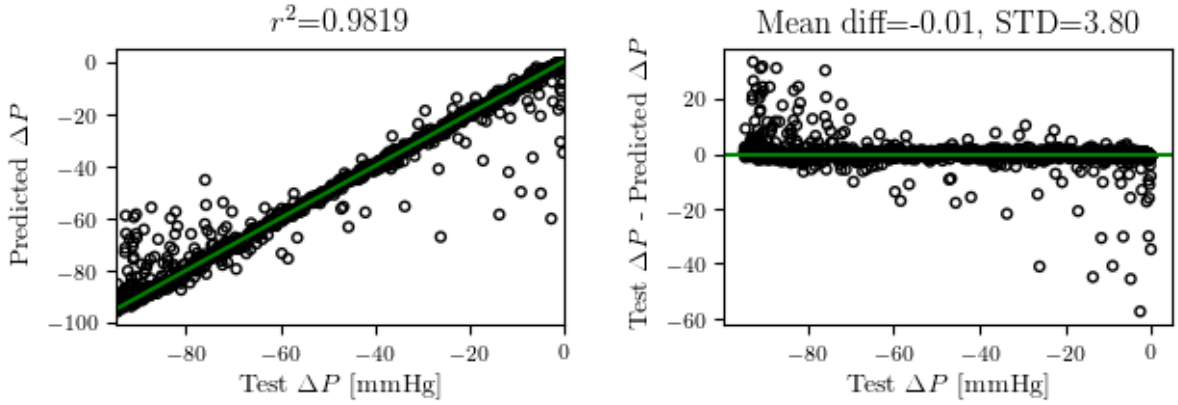
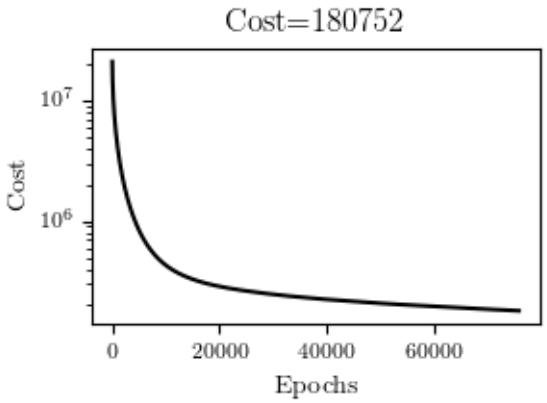


Figure 4.11: Cost vs epochs for case 85, stenosis model.



(a) Predicted ΔP vs. actual ΔP

(b) Residuals



(c) Cost vs. epochs

Figure 4.12: Results from case 95.

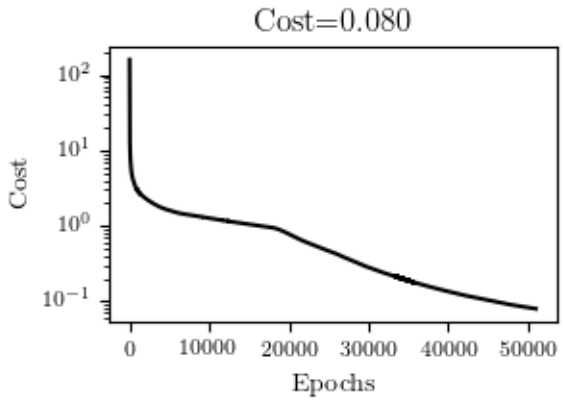
4.3 Impact of each of the hyper-parameters

The influence on the network ability to reproduce the 1D and stenosis model are presented here. The hyper-parameters will be compared to the best network in the grid search for the respective models, unless it is better visualized by other networks.

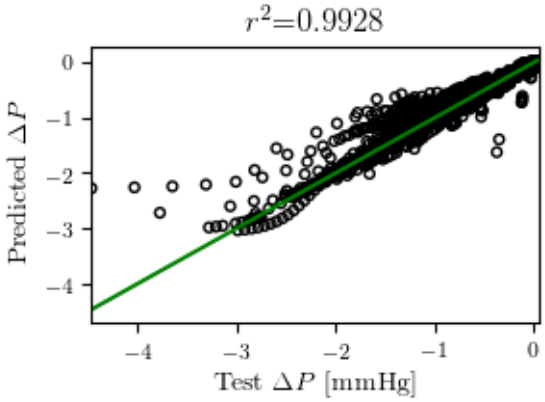
4.3.1 Learning rate

4.3.1.1 1D-ML network For the 1D-ML model the cost development is illustrated in Figure **??** for case 175, 55 and 115. The three sampled learning rates are 0.01, 0.1 and 0.25. The final cost is 4.53, 1,71 and 0.08, while the test cost is 4.76, 2.80 and 1.28 for the respective models. In Figure 4.13a it is seen that the reduction rate for the cost is increasing after approximately 20000 epochs.

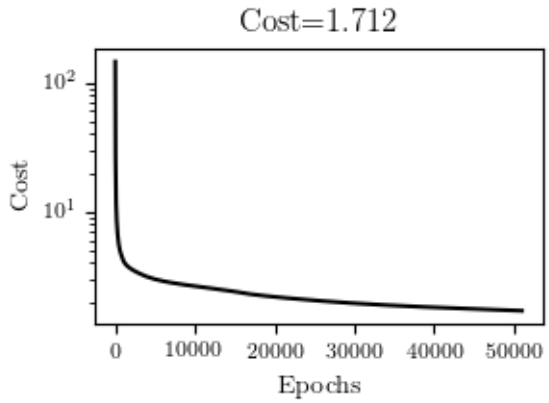
4.3.1.2 Stenosis-ML network For the best network in the stenosis model, the results are shown for all three learning rates in Figure 4.14. The curve of the cost development is much steeper for the highest learning rate, but for the lowest learning rate, $\mu = 0.01$, the curve has not started to flattened out after 75000 epochs. The test cost for the model is $1.098 * 10^7$ compared to $1.875 * 10^5$ for $\mu = 0.1$ and $1.072 * 10^5$ for $\mu = 0.25$. There is a big difference in the final cost of the networks with a final cost of 10621649 fir the lowest learning rate, and 20150 for the highest learning rate. All of the networks with $\mu = 0.01$ does still have a steep cost vs. epochs curve, and the network that reach the best cost in 75000 epochs is case 145 with a test cost of $1.875 * 10^5$.



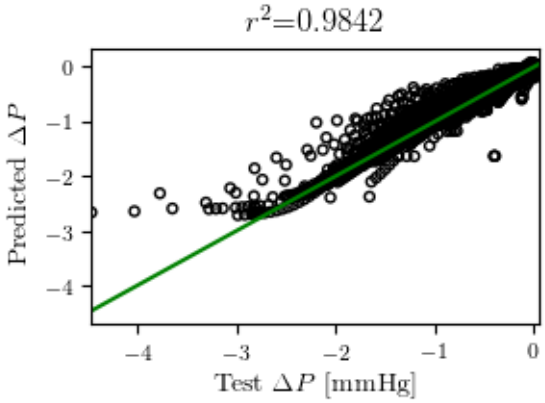
(a) Case 115, η=0.25



(b) Case 115, η=0.25

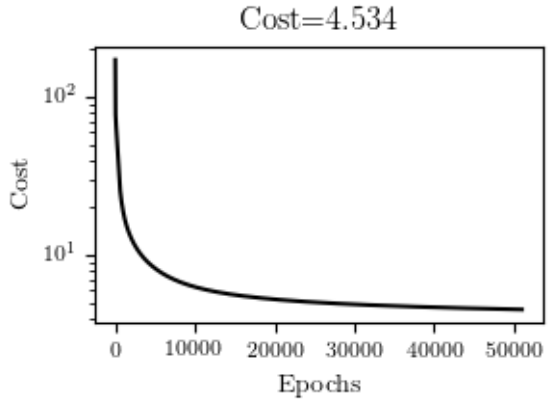


(c) Case 55, η=0.1

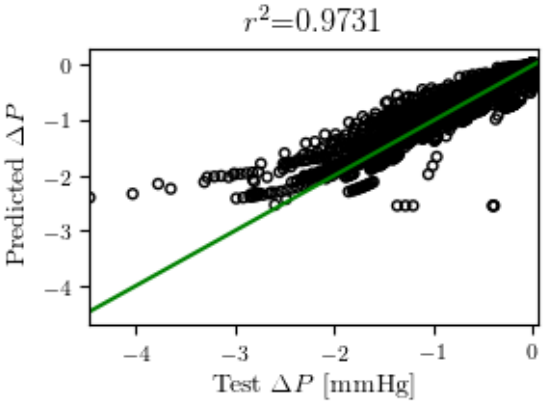


(d) Case 55, η=0.1

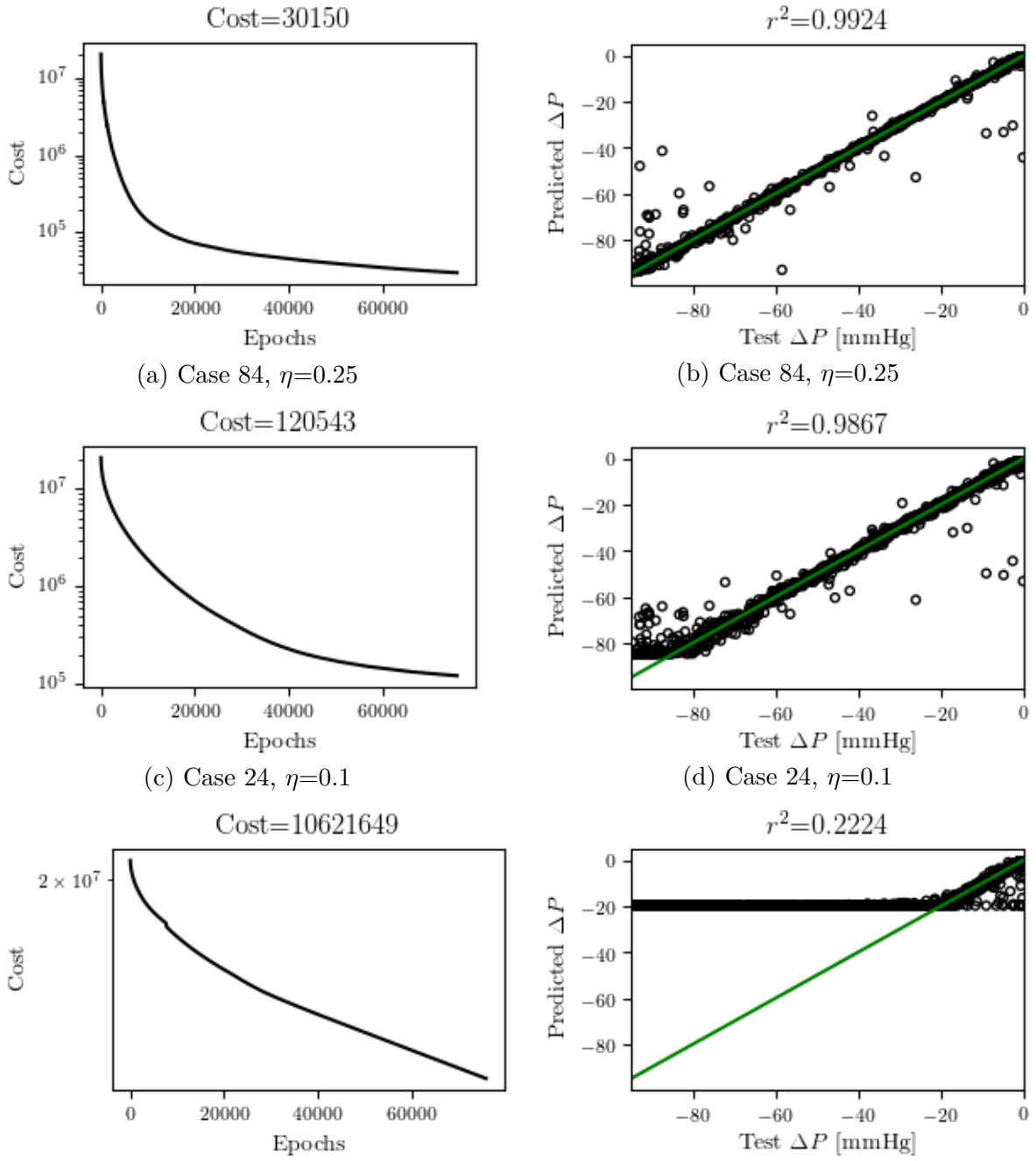
(e) Change in learning rate for the 1D model



(f) Case 175, η=0.01



(g) Case 175, η=0.01



(a) Case 84, η=0.25

(b) Case 84, η=0.25

(c) Case 24, η=0.1

(d) Case 24, η=0.1

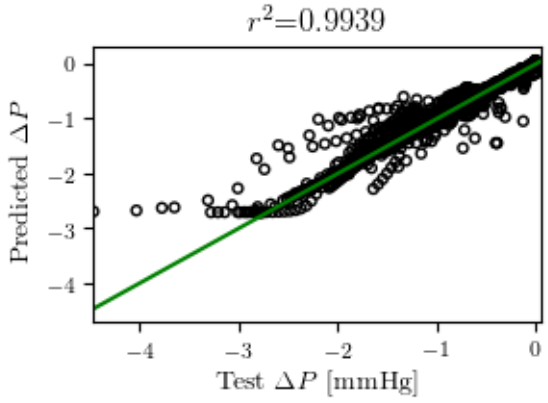
(e) Case 144, η=0.01

(f) Case 144, η=0.01

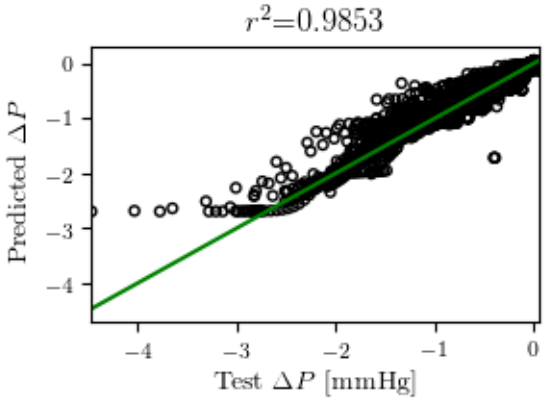
Figure 4.14: Change in learning rate for the stenosis model.

4.3.2 Batch size

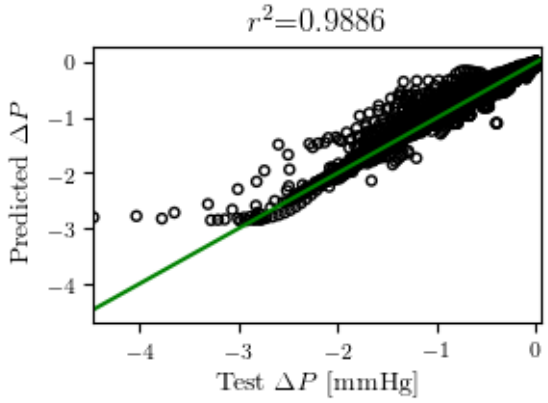
4.3.2.1 1D-ML network For both models the performance improves when the batch size is reduced within the range of the grid. In the 1D model the lowest batch size is 1000, approximately 0.5% of the training data that is fed to the ML model. The other batch sizes are approximately 2.5%, 5% and 10% of the training data. In Figure 4.15 the improvement of the predicted values when the batch size is reduces is shown. The results does not show any problems from the batch size being to small to represent the data set, but the highest predicted value increases when the batch size is reduced.



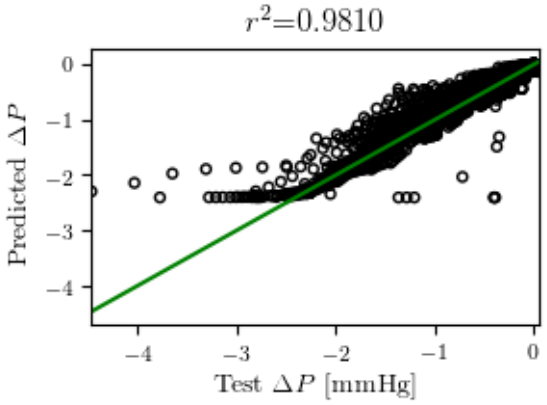
(a) Case 119: Batch size=1000, approximately 0.5% of the training data



(b) Case 89: Batch size=5000, approximately 2.5% of the training data



(c) Case 104: Batch size=10000, approximately 5% of the training data

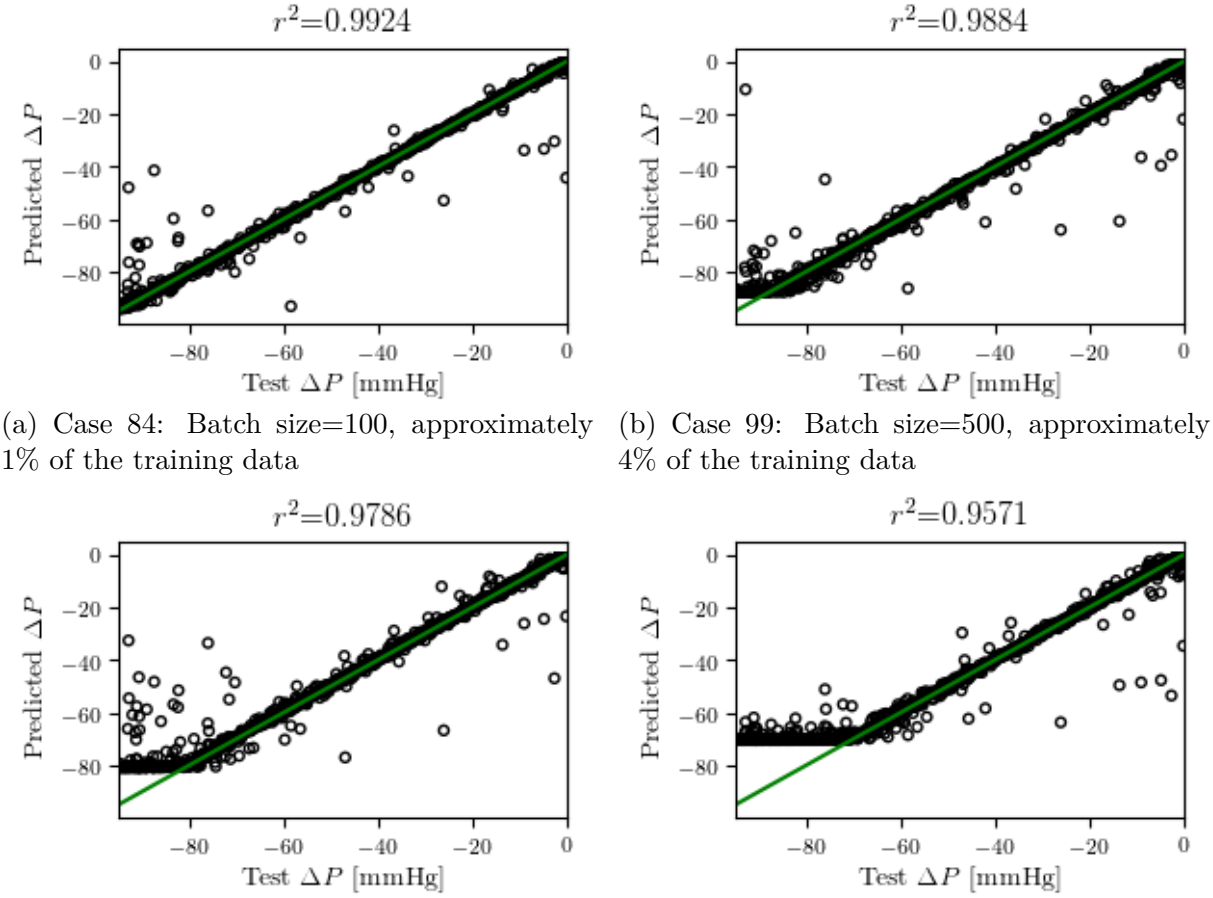


(d) Case 74: Batch size=20000, approximately 10% of the training data

Figure 4.15: Batch size, 1D model.

4.3.2.2 Stenosis-ML network For the stenosis model the best batch size is 100 examples, a little bit less than 1% of the training data that is fed to the ML model. The larger batch sizes are approximately 4%, 8% and 15% of the training data. The effect on the predicted values from the different batch sizes are seen in Figure 4.16. There is not

observed any problems with a too small batch size for this model neither.



(a) Case 84: Batch size=100, approximately 1% of the training data

(b) Case 99: Batch size=500, approximately 4% of the training data

(c) Case 69: Batch size=1000, approximately 8% of the training data

(d) Case 114: Batch size=2000, approximately 15% of the training data

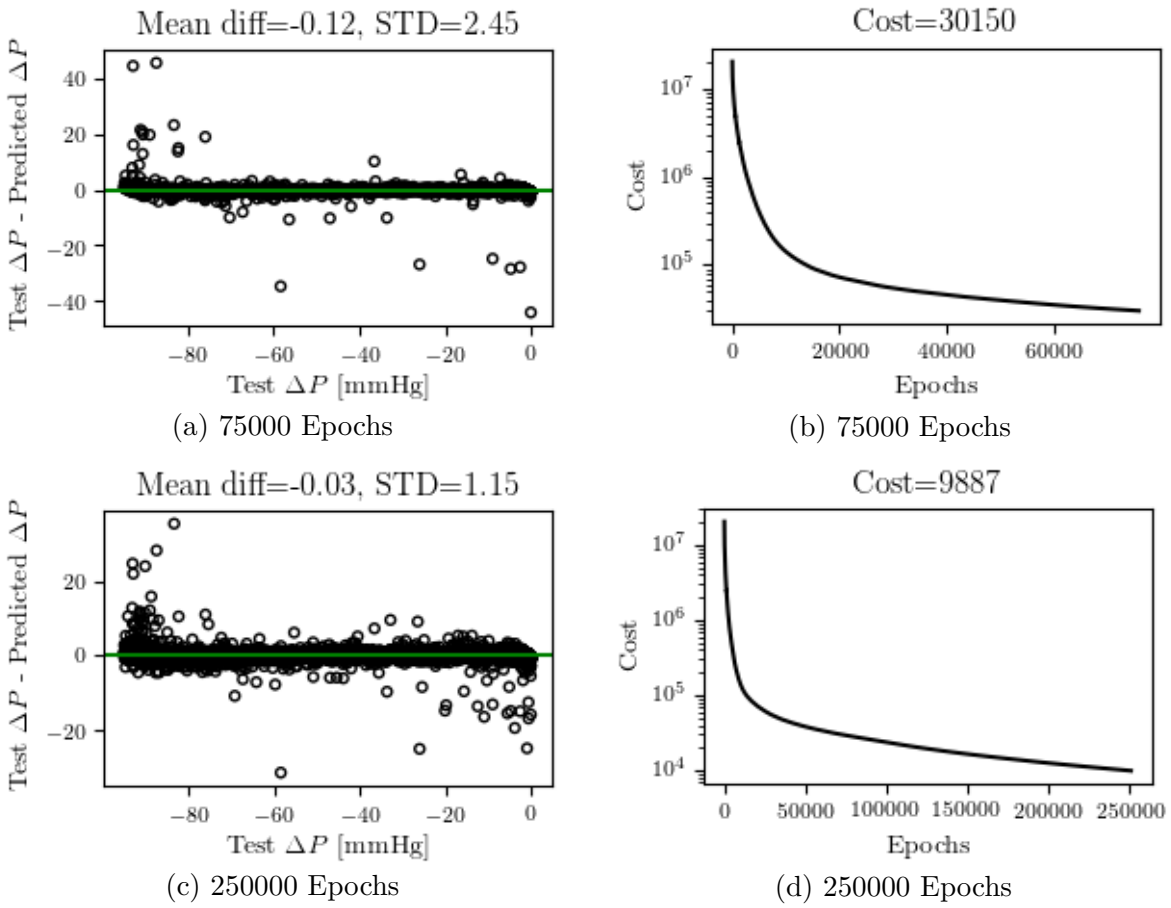
Figure 4.16: Batch size, stenosis model.

4.3.3 Number of epochs

4.3.3.1 Stenosis-ML network When the number of epochs in case 84 in the stenosis grid search is increased to 250000 epochs, the final cost goes down to 9887 from 30150 after 75000 epochs. From Figure 4.17 it is evident that the cost is still improving after 250000 epochs, so the model would benefit from an increased number of epochs. The STD is halved with this increase of epochs.

4.3.4 Number of hidden layers

4.3.4.1 1D-ML network The grid search included neural networks with 2, 3 and 4 hidden layers. For the 1D-model the best network is a four layer network. For this config-



(a) 75000 Epochs

(b) 75000 Epochs

(c) 250000 Epochs

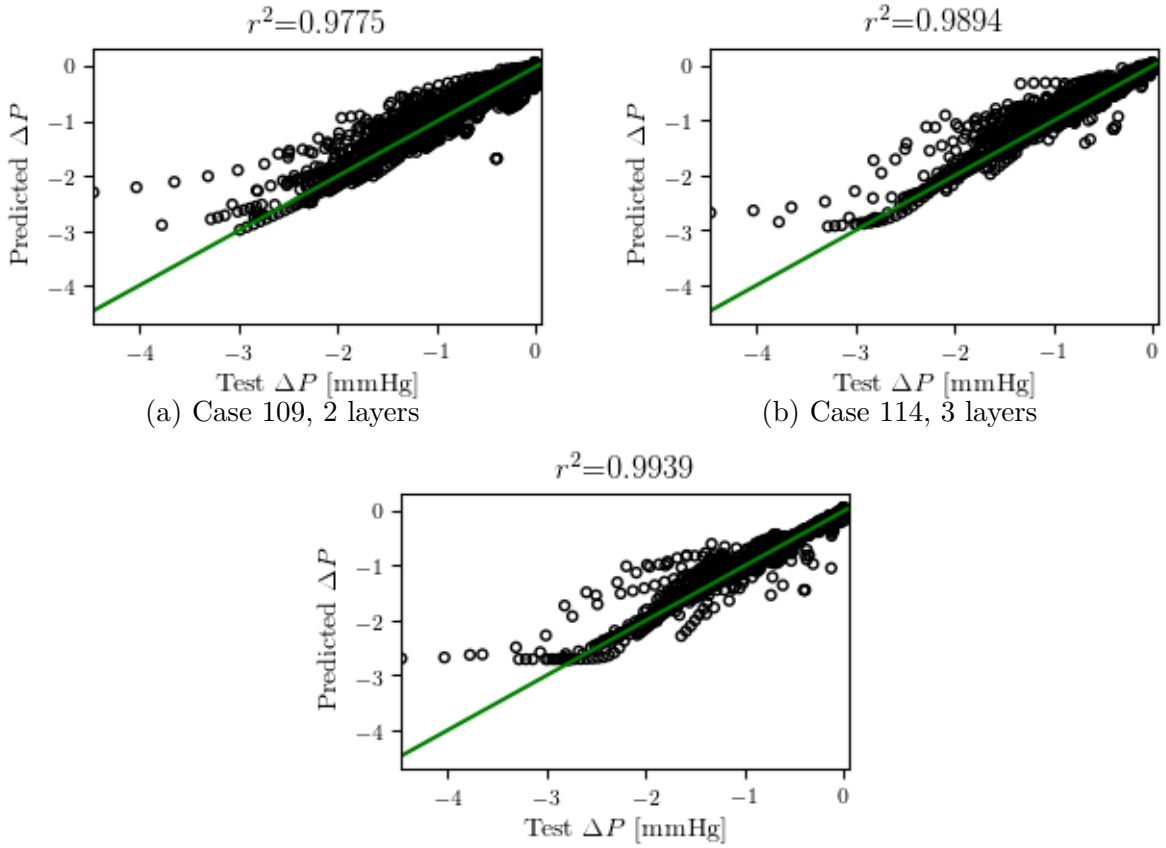
(d) 250000 Epochs

Figure 4.17: Case 84, increased number of epochs.

uration of hyper-parameters, but with different number of hidden layers the predictions are seen in Figure 4.18. It is seen that for the deepest network, the predictions are closer to the regression line. For case 109 the coefficient of determination is 0.977, for case 119 this has improved to 0.994.

4.3.4.2 Stenosis-ML network For the stenosis model it is seen that a deeper network does not improve the range of the predicted values in Figure 4.19, here illustrated by case 77, 82 and 87. As noticed for the 1D model, the predictions in the learned range have improved. There is less outliers, and the the coefficient of determination goes from 0.967 for case 77 with 2 hidden layers to 0.986 for case 87 with 4 hidden layers.

It is not the case that the model performance increases with the number of layers for all cases. The best network in the stenosis grid is a three layer network. In the four layer network with the same configurations the network did not learn, as seen in Figure 4.20.



(a) Case 109, 2 layers

(b) Case 114, 3 layers

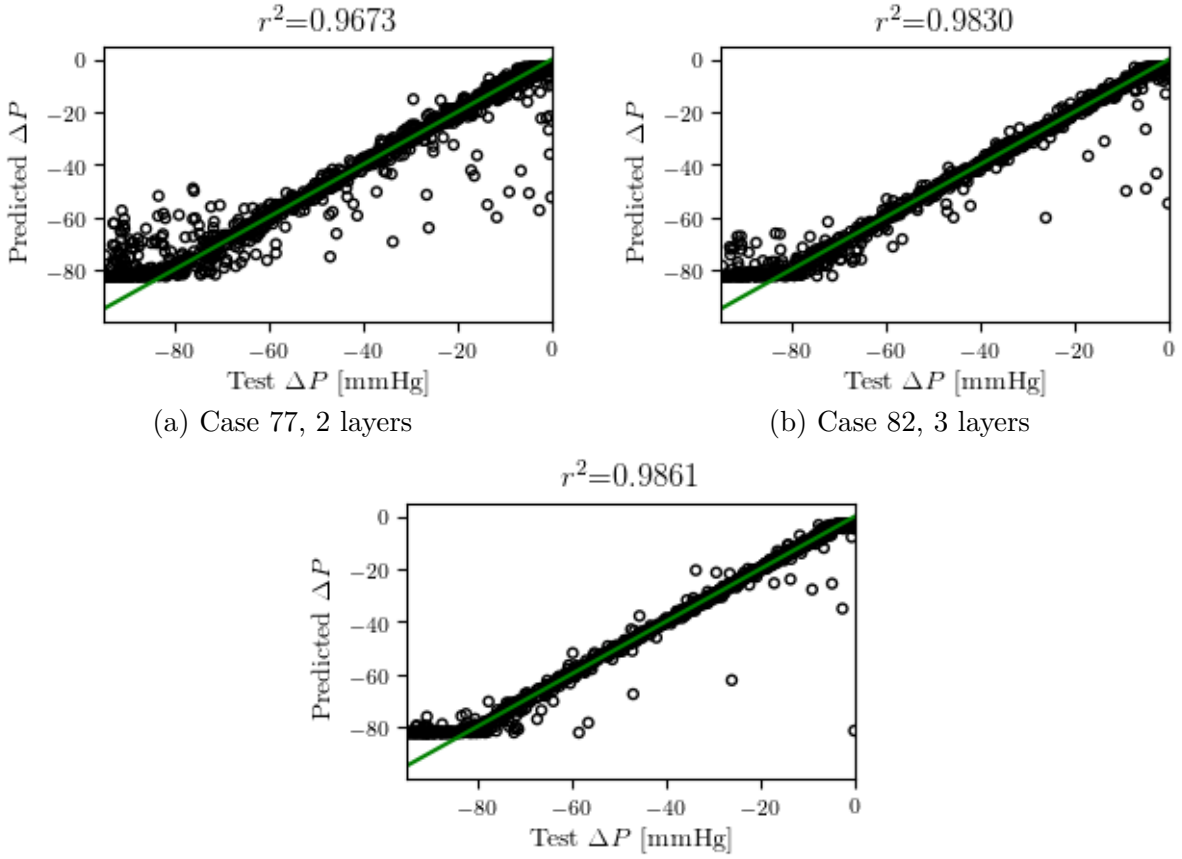
(c) Case 119, 4 layers

Figure 4.18: Predicted ΔP vs. actual ΔP, dept of the network for the 1D model.

4.3.5 Depth of the layers

4.3.5.1 1D-ML network The depth of the layers are varied between 10, 20, 30, 50 and 100 neurons. For the 1D model all five networks seen in Figure 4.21 have a good accuracy. The coefficient of determination only varies with 0.016 from the best case to the worst, where the best case has 50 neurons and the worst case has 10 neurons in each layer. The values for r^2 is found in Table 4.2.

4.3.5.2 Stenosis-ML network For most of the networks in the stenosis-ML model, the effect from increasing the number of neurons is illustrated by a two layer network, with batch size 2000, and learning rate 0.25. Predicted ΔP is plotted against the actual ΔP in Figure 4.22. As the number of neurons increases in the network, the model is able to predict a bigger range of pressure drops. For the smallest network with only 10 neuron in each layer the model has not learned to predict pressure drops over 20 mmHg. When the number of neurons is increased to 20, this doubles. For the biggest network, the highest predicted pressure drop is 88 mmHg.



(a) Case 77, 2 layers

(b) Case 82, 3 layers

(c) Case 87, 4 layers

Figure 4.19: Predicted ΔP vs. actual ΔP, dept of the network for the stenosis model.

Some of the hyper-parameter combinations shows another behaviour. When the number of neurons increases, the network suddenly does not learn anything. This shift happens between case 117 and case 118. Two four layers networks, with batch size 2000, and learning rate 0.25. The same as the examples above. Case 117 has 20 neurons in each layer and case 118 has 30 neurons. In Figure 4.23c it is seen that in case 118 all predictions are $\Delta P = 20\,\text{mmHg}$. This network has reach the stop criteria defined for the cost after 57000 epochs. The cost has stopped at $1.37 * 10^7$. This behaviour is mostly seen in 4 layers networks.

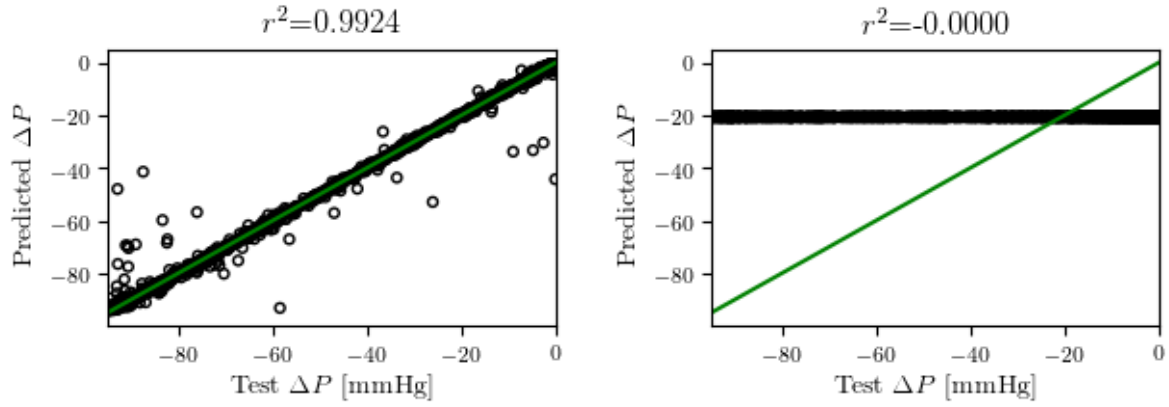
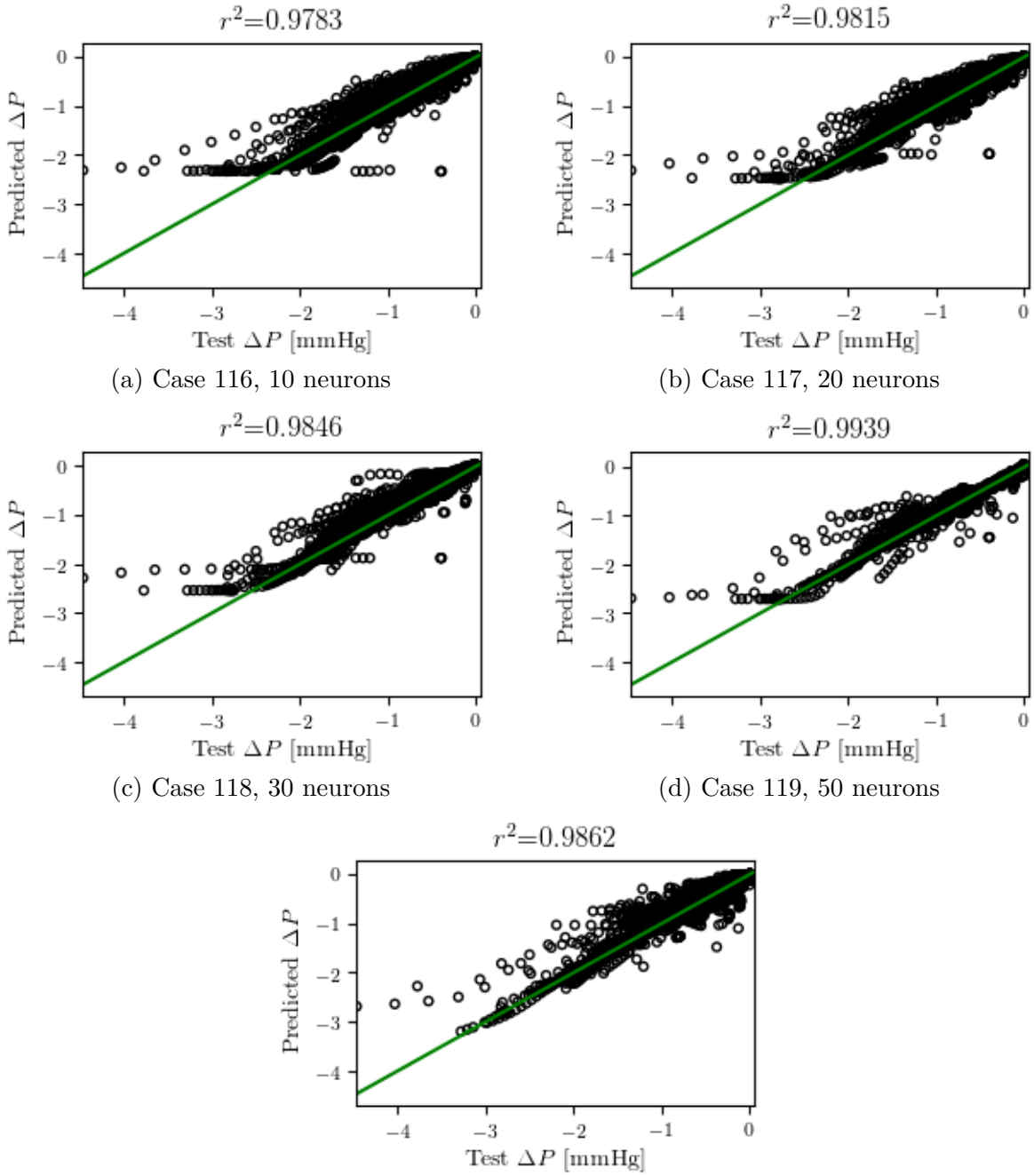


Figure 4.20: Predicted ΔP vs. actual ΔP, dept of the network for case 84 and 89 in the stenosis grid.



(a) Case 116, 10 neurons

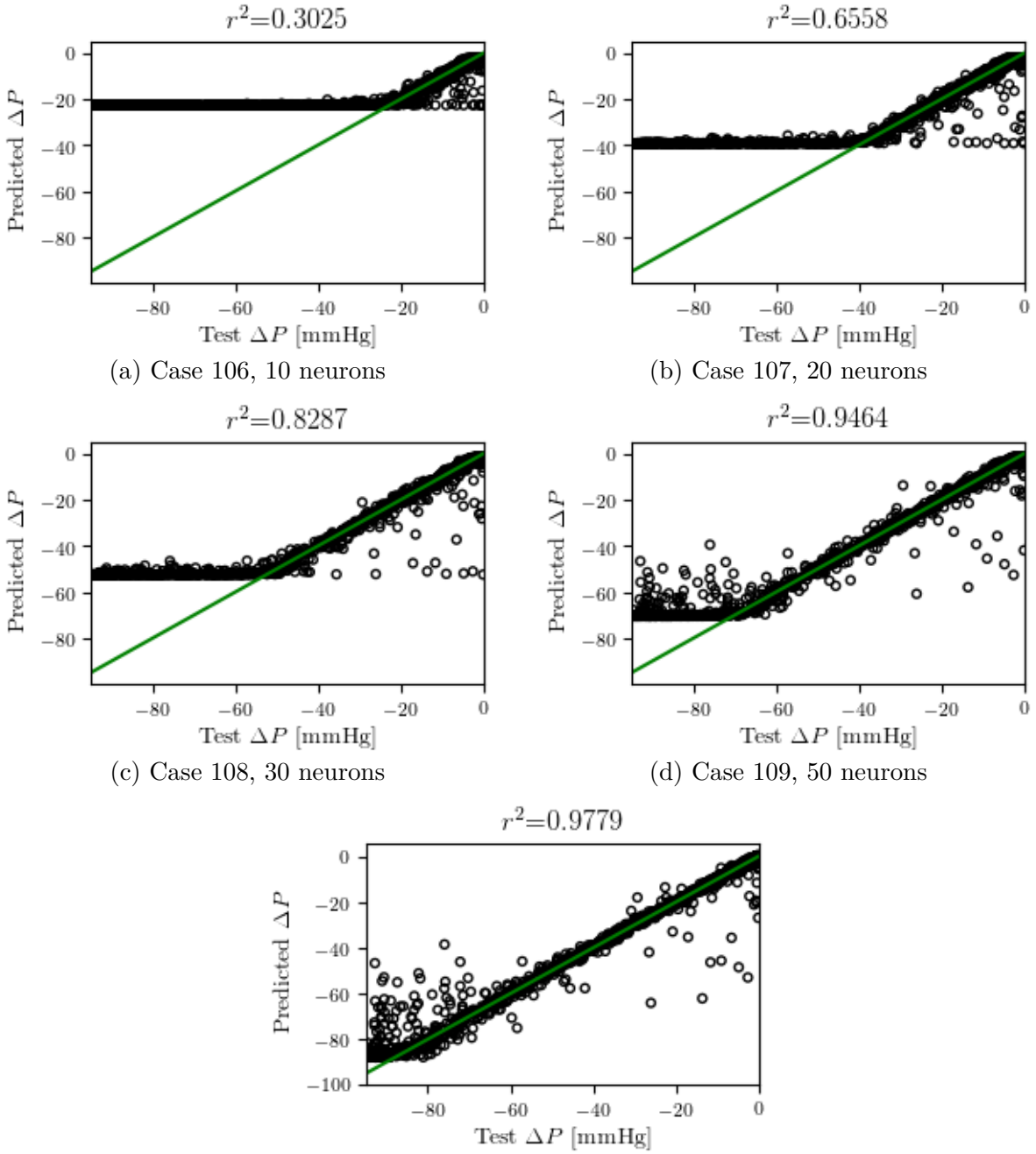
(b) Case 117, 20 neurons

(c) Case 118, 30 neurons

(d) Case 119, 50 neurons

(e) Case 120, 100 neurons

Figure 4.21: Predicted ΔP vs. actual ΔP, with a increasing number of neurons in each layer for the 1D-model.



(a) Case 106, 10 neurons

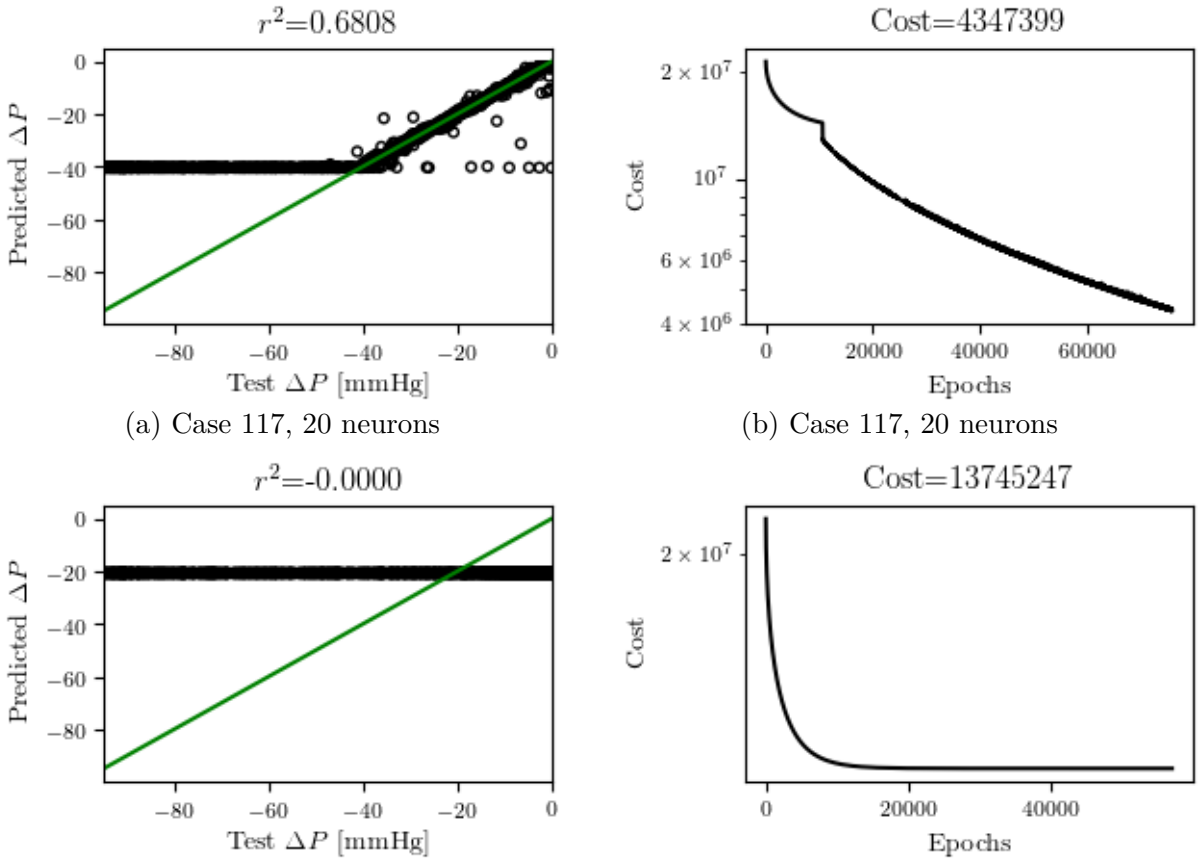
(b) Case 107, 20 neurons

(c) Case 108, 30 neurons

(d) Case 109, 50 neurons

(e) Case 110, 100 neurons

Figure 4.22: Predicted ΔP vs. actual ΔP, with a increasing number of neurons in each layer for the stenosis model.



(a) Case 117, 20 neurons

(b) Case 117, 20 neurons

(c) Case 118, 30 neurons

(d) Case 118, 30 neurons

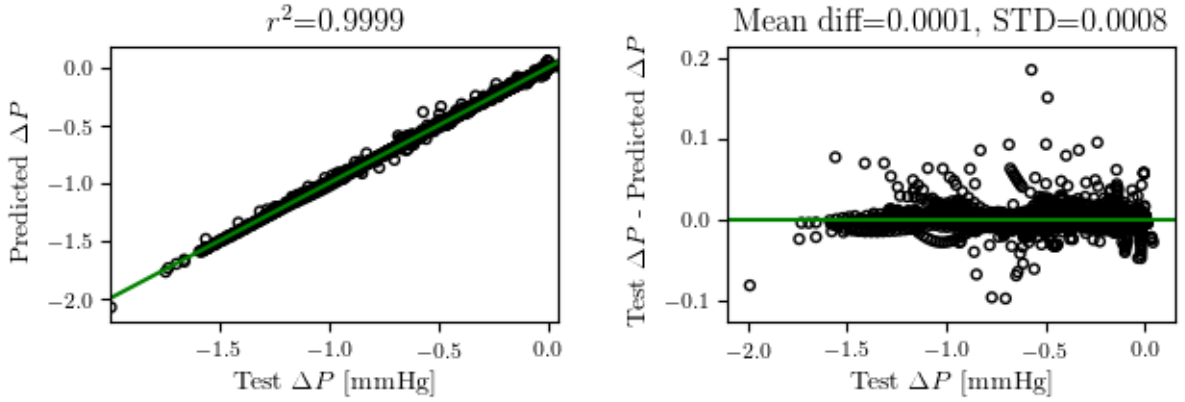
Figure 4.23: Unsuccessful training session in the stenosis model.

4.4 Training with the final database

The hyper-parameter composition from the best network in the two respective models are used to train a network with the final database. Because of the time limit on this thesis there was not time to perform the grid search on the final database. In addition to the best network form the grid search, a few random tries have been performed with the final database.

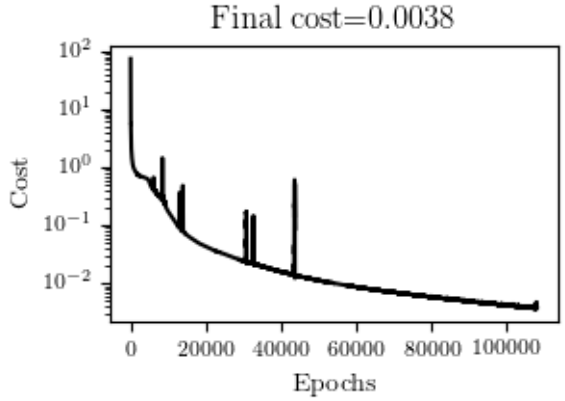
4.4.1 1D-ML model

A network was trained with the hyper-parameters from case 119 for the 1D-model, with the final database. The networks performance is visualized in Figure 4.24. The training stopped after 108000 epochs, because it reached the early stopping criteria, a lower reduction of the cost than 1^{-5}. It is noticed that the cost has some oscillations. The coefficient of determination is 0.9999, so the network gives an excellent representation of the test data-set.



(a) Predicted ΔP vs. actual ΔP

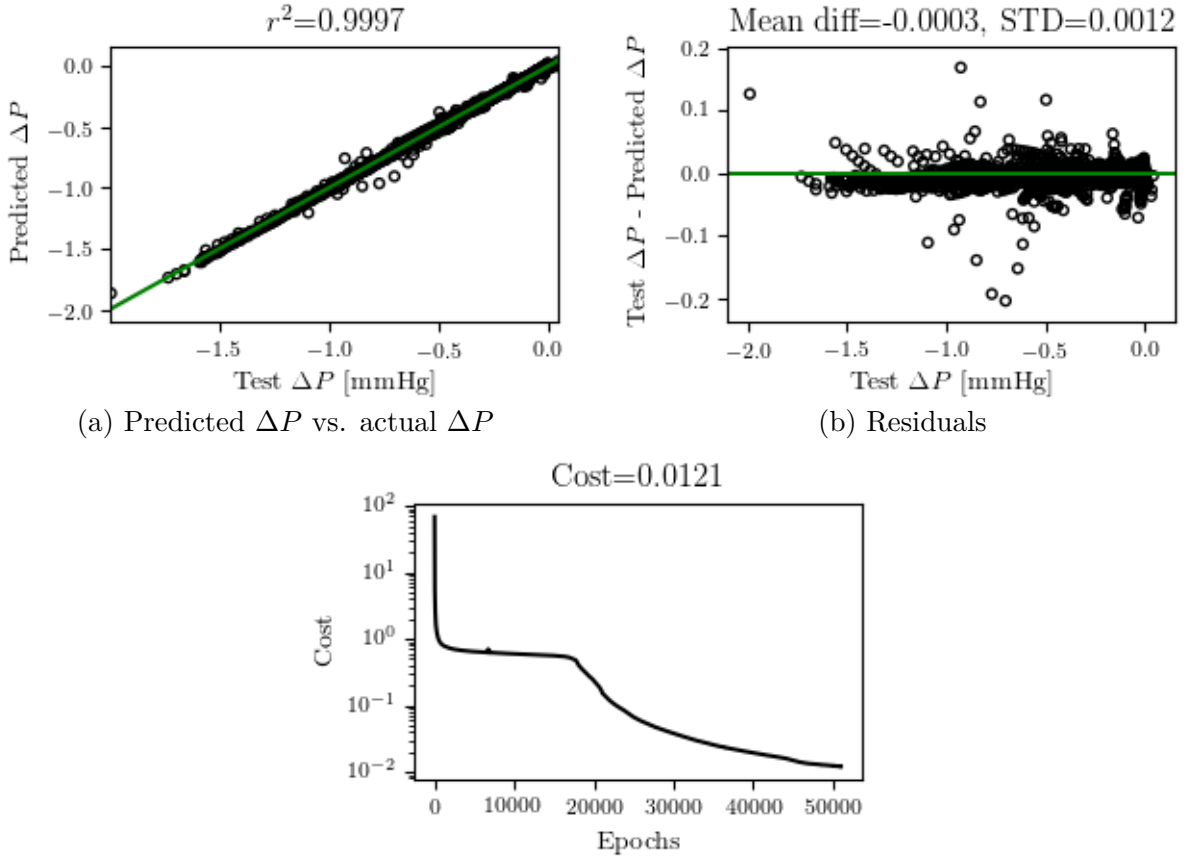
(b) Residuals



(c) Cost vs epochs

Figure 4.24: Case 1.119, training the 1D model with the final database.

So far, the model has been trained with 15% of the database. This was increased to 30%. The results are seen in Figure 4.25. The hyper-parameters were chosen from case 119, like case 1.119 presented above. It was trained for 50000 epochs. It is noticed that there are more examples with a higher residual than 0.1 mmHg for this model than the case 1.119.



(a) Predicted ΔP vs. actual ΔP

(b) Residuals

(c) Cost vs epochs

Figure 4.25: Case 2.119, training the 1D model with the final database, introducing more training data.

Table 4.4: Results from training sessions with the final database, 1D-ML model.

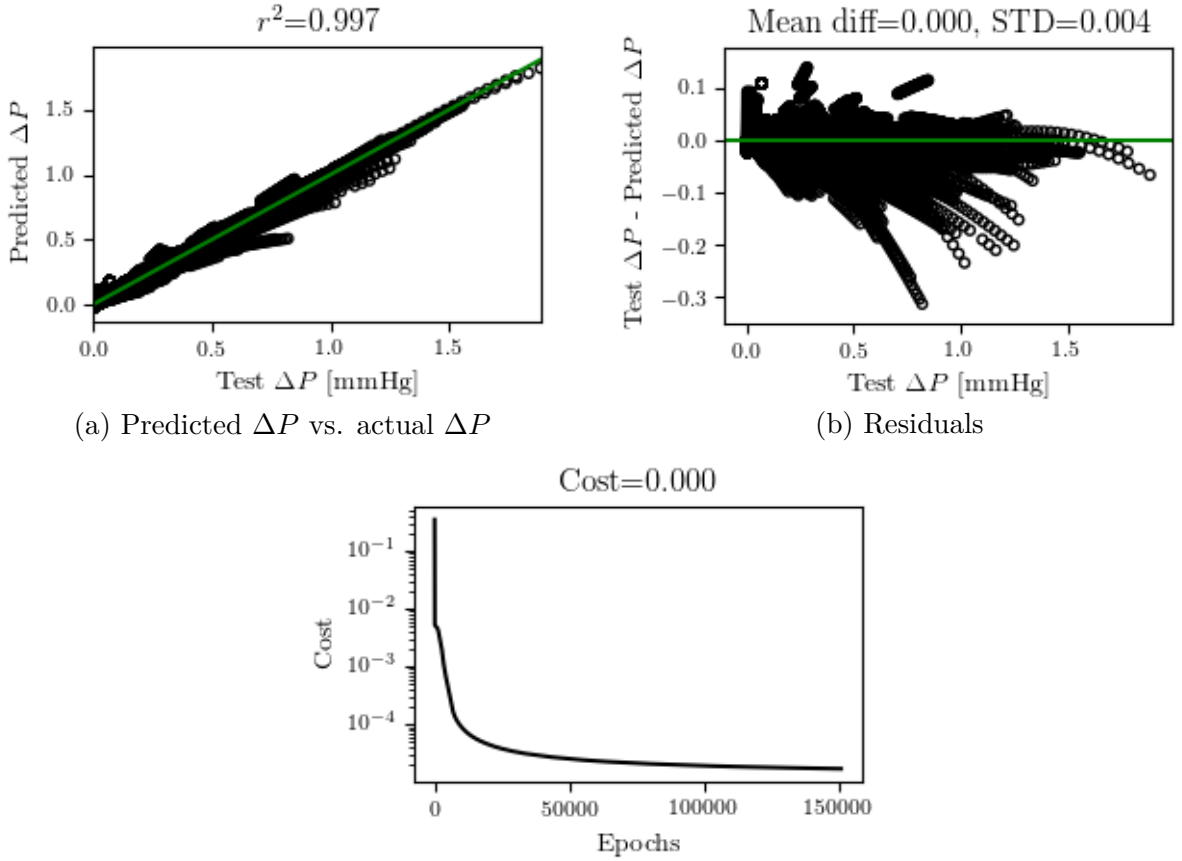
Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
119	0.25	1,000	50 50 50 50	1.085	0.994	$-2.725{\cdot}10^{-5}$	$7.813{\cdot}10^{-3}$
1.119	0.25	1,000	50 50 50 50	$1.107{\cdot}10^{-2}$	1	$6.16{\cdot}10^{-5}$	$7.867{\cdot}10^{-4}$
2.119	0.25	1,000	50 50 50 50	$2.533{\cdot}10^{-2}$	1	$-2.808{\cdot}10^{-4}$	$1.16{\cdot}10^{-3}$
3.119	0.25	1,000	50 50 50 50	$1.99{\cdot}10^{-5}$	0.997	$3.813{\cdot}10^{-7}$	$3.346{\cdot}10^{-5}$

LR = learning rate, BS = batch size, r^2 = coefficient of determination, STD = standard deviation

4.4.1.1 Additional changes For the final database, several changes where implemented to improve the ML models. These changes were related to the scaling of the

features. First the features were converted from SI units 3.4 to Q [mL/s], A_0 [cm^2], A_1 [cm^2], L [cm] and ΔP [mmHg]. After the features was scaled the flow was shifted into a positive range. The minimum value for the scaled flow was identified, and all values was increased with this amount. This gave a $Q_{scaled,min,new} = 0$ and the mean of $|Q_{scaled,min}|$.

The network was also trained with the same configurations, but with an adapted scaler. This scaler shifted the skaled values for the flow into a positive range. These results are seen in Figure 4.26.



(a) Predicted ΔP vs. actual ΔP

(b) Residuals

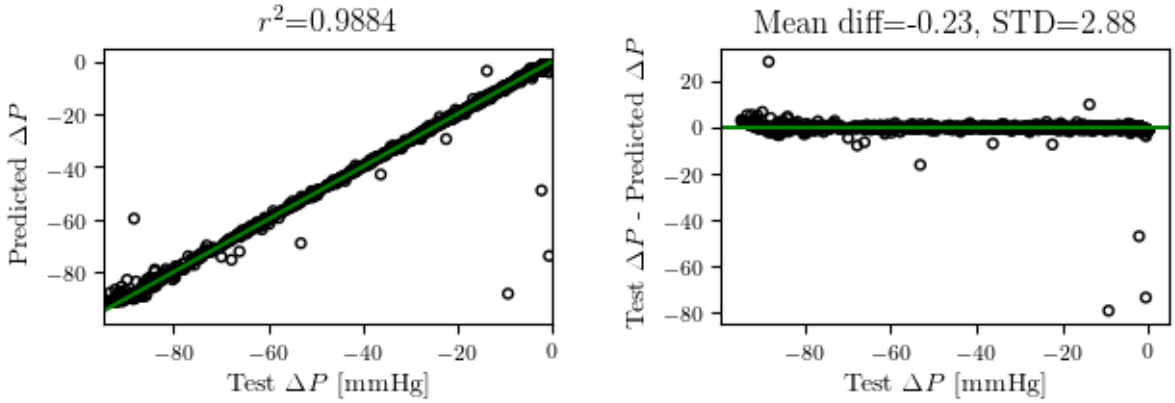
(c) Cost vs epochs

Figure 4.26: Case 3.119, trained with the adapted final database for the 1D model.

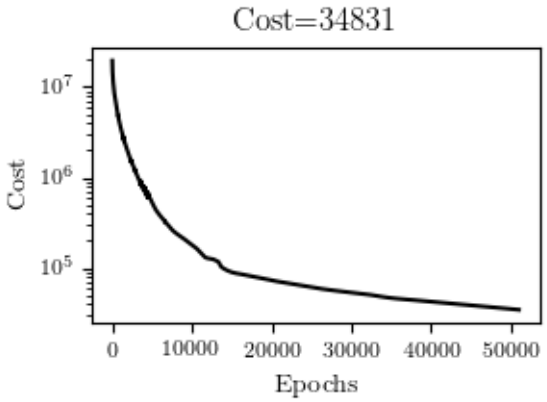
4.4.2 Stenosis-ML model

The final stenosis database was trained with the hyper-parameter configuration from case 84 in the stenosis grid. The networks performance is seen in Figure 4.27. The coefficient of determination has increased to 0.998 and the STD is 2.88 mmHg. The coefficient of determination has improved while the STD has increased with 0.43 mmHg.

To exploit the training potential for the model, the number of epochs was first increased



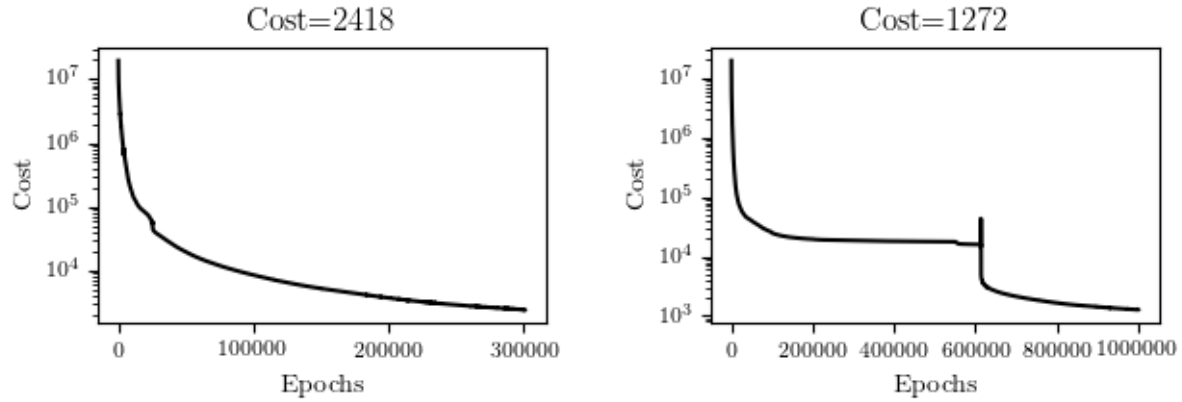
(a) Predicted ΔP vs. actual ΔP (b) Residuals



(c) Cost vs epochs

Figure 4.27: Training the stenosis model with the final database for 50000 epochs.

to 300000.The cost was still improving, so the number of epochs was further increased to 1000000 epochs. In Figure 4.28 if is seen a drop in the cost after approximately 600000 epochs. The final cost after 1000000 is 1272, compared to 34831 after 50000 epochs, and it is still decreasing. The network details and performance measurements are summarized in Table 4.5, together with the other network presented in this section.



(a) Case 1.84, 300000 epochs. (b) Case 2.84, 1000000 epochs.

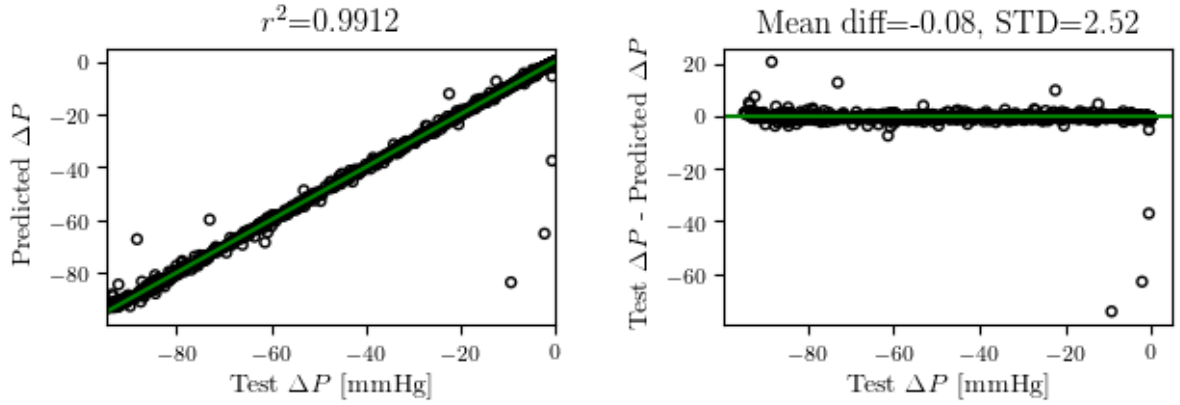
Figure 4.28: Training epochs.

Table 4.5: Results from training sessions with the final database, Stenosis-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
84	0.25	100	50 50 50	$1.072{\cdot}10^{5}$	0.992	-0.121	2.453
1.84	0.25	100	50 50 50	$1.489{\cdot}10^{5}$	0.988	-0.231	2.885
2.84	0.25	100	50 50 50	$1.134{\cdot}10^{5}$	0.991	$-8.434{\cdot}10^{-2}$	2.524
3.84	0.1	1,000	50 50 50 50 50	$1.571{\cdot}10^{-2}$	1	$-1.937{\cdot}10^{-6}$	$9.4{\cdot}10^{-4}$

LR = learning rate, BS = batch size, r^2 = coefficient of determination, STD = standard deviation

The networks performance after 1000000 epochs is visualized in Figure 4.29. With a coefficient of determination of 0.991 the model gives a good representation of the test data-set. The STD is 2.52 mmHg, and it is seen that 4 of the 1953 data points have a residual of more than 20 mmHg.

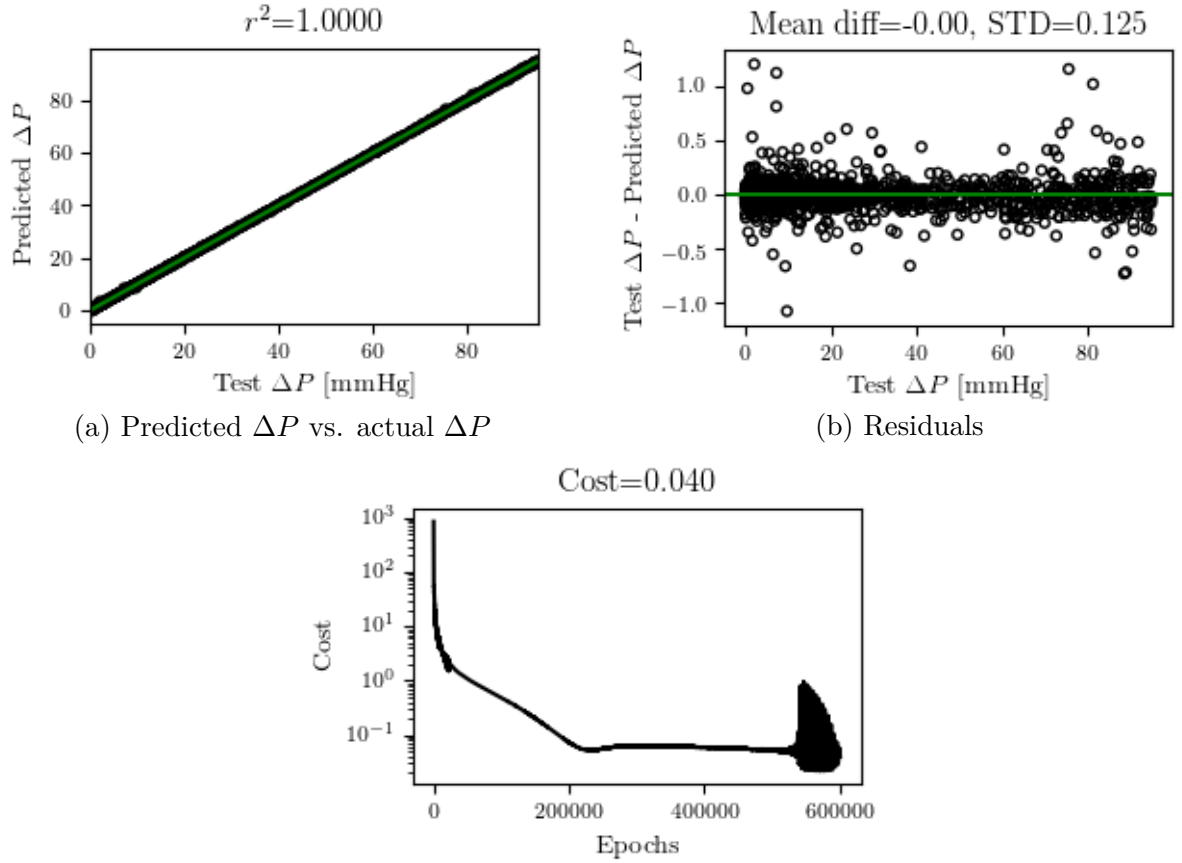


(a) (b)

Figure 4.29: Case 2.84, model performance after 1000000 epochs.

4.4.2.1 Additional changes As mentioned above, the units where changed for the final training database. The features for the stenosis was converted from SI units 3.4 to Q [mL/s], D_0 [cm], D_1 [cm], L [cm] and ΔP [mmHg]. The stenosis severity degree was changed from the percentage to the minimum stenosis diameter, D_s [cm]. The flow was scaled by the same method as the 1D to avoid negative values.

The best network in the small random search of for a hyper-parameter combination of learning rate 0.1, batch size 1000 and five hidden layers with 50 neurons in each layer. This is trained with the adapted scaling and units, and is stopped after 600000 epochs. In Figure 4.30 it is observed that the cost start to oscillate after approximately 550000 epochs.



(a) Predicted ΔP vs. actual ΔP

(b) Residuals

(c) Cost vs. epochs

Figure 4.30: Case 3.84, trained with the adapted final database for the stenosis model.

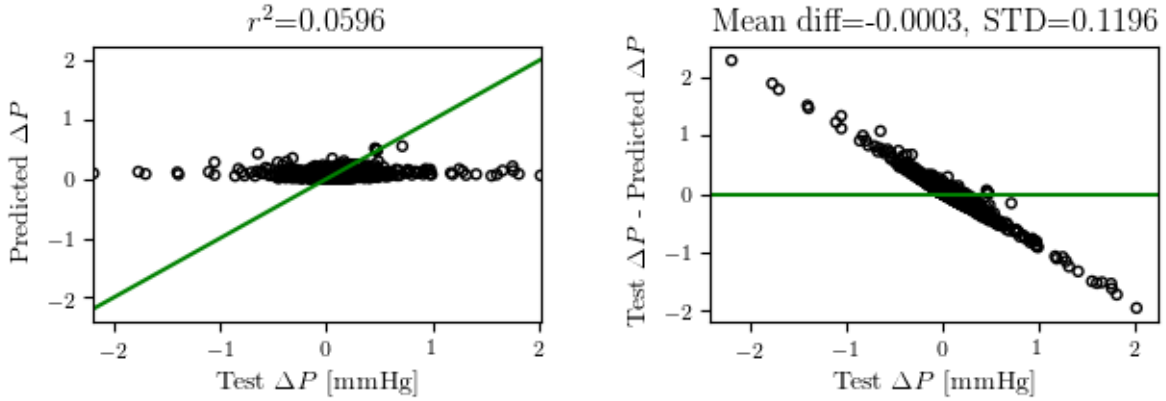
4.5 Final ML model

For the 1D-ML model, the best network is case 3.119. A four layer network, with 50 neurons in each layer. The learning rate is 0.25 and the batch size is 1000. The training was stopped after 150000 epochs. The network was trained with the adapted, final database. The STD is $3.35 * 10^{-5}$ mmHg, the mean difference is $3.81 * 10^{-7}$ mmHg and the coefficient of determination is 0.997 (Table 4.4).

The best network for the stenosis-ML model was found in case 3.84. A five layer network with 50 neurons in each layer. The learning rate is 0.1 and the batch size is 1000. The network is trained with the adapted, final stenosis database. After 600000 epochs it reaches a coefficient of determination of 1, the STD is $9.4 * 10^{-4}$ mmHg and the mean difference is $-1.94 * 10^{-6}$ mmHg (Table 4.5).

4.6 Prediction from patient specific coronary trees

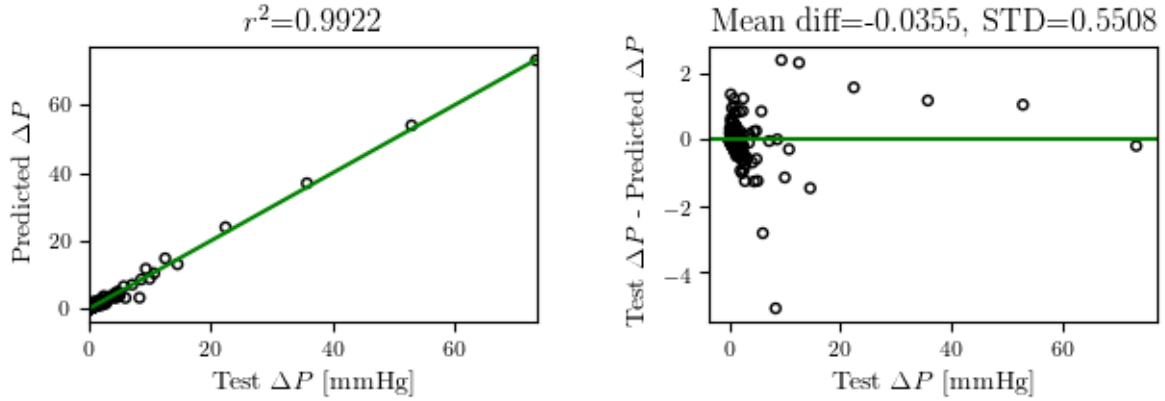
From the 13 patient specific cases, the features was extracted. These features was introduced to the final ML-models, to see how the ML algorithms performed on patient data. The predictions for the vessels segments are visualized in Figure 4.31. This shows that even if the 1D-ML model has good results on the test data set, it is not able to predict the pressure drops along the vessel. The coefficient of determination is as low as 0.06.



(a) Predicted ΔP vs. actual ΔP (b) Residuals

Figure 4.31: Predictions on the patient specific geometries for the vessel segments.

The Stenosis-ML model give better results on the patient specific data than the 1D model. As seen in Figure 4.32, there is an overweight of stenoses with a small pressure drop. The network has a coefficient of determination of 0.992, a mean difference of 0.04 mmHg and the STD is 0.55 mmHg.



(a) Predicted ΔP vs. actual ΔP (b) Residuals

Figure 4.32: Predictions on the patient specific geometries for the stenoses.

5 Discussion

The results will be evaluated based on their relevance to the objectives in this thesis. The aim is to develop a ML model that successfully predicts pressure drops in the coronary tree for the 1D model, and the Young and Tsai stenosis model.

The synthetic generated training database of coronary trees and the extracted features will be addressed together with the second objective, the implementation of the ML algorithm. The ML algorithms are evaluated from its prediction accuracy on the patient specific coronary trees.

5.1 Feature selection

This thesis was inspired by the work of Itu et al. [17], and was supposed to be an attempt to reproduce their work with a training database established with the steady state solver developed at NTNU. The coronary trees were generated following their approach, and the geometrical data was prescribed to mimic their database where they were stated. For the stenosis, they did not present the geometrical parameters.

Their ML algorithm required 28 input features. From their work, it was not possible to identify these features. One of their features, referred to as a segment-specific ischemic weight, was not defined. Another mysterious feature was the nonlinear product combinations between the four most significant stenoses upstream and downstream of the segment. The authors were contacted about providing clarifying details, however, they did not send enough information to be able to reproduce the results. They referred it to be an approach under patent protection, therefore intentionally not fully disclosed in the paper.

The feature selection is an important step in the implementation of a ML algorithm. In this thesis the features are selected to represent values that are found non-invasely through standard procedures. When it was clear that it was not possible to reproduce the features that were used in Itu et al., a new selection of features was made. As the main objective was to see if it was possible to reproduce the mathematical models, only the features directly related to the mathematical models were taken into consideration.

The other ML approach for a coronary flow model, presented by Sankaran et al. [18] includes 35 features, more than Itu et al. These features include the weight and height

of the patient. Their database is built up from patient data from CT images, and blood flow simulations.

5.2 Training database

A prerequisite to implement a ML algorithm is the existence of a training database. The database has to be reliable, contain all the information regarding the problem at hand, and it has to be large enough for the ML model to be able to represent the general behaviour of the real model. Ideally, this database should have been established from anatomical geometries extracted from CCTA images, with corresponding FFR measurements of each vessel in the coronary tree. The patient population should be representative for the anatomical variation of patients with suspected stable CAD. However, such a database is not available, and it is not feasible to establish a large enough database to be able to train the ML model. Hence, the training database had to be generated synthetically.

As pointed out earlier, the grid search was performed with a preliminary database. One of the problems in the preliminary database was that some of the distal branches received an unrealistically small flow, as low as 3.28×10^{-8} mL/s. One of these stenoses is seen in Figure 5.1. These branches are likely due to the normal distribution of the feature characteristics providing extreme values, outside of the physical range.

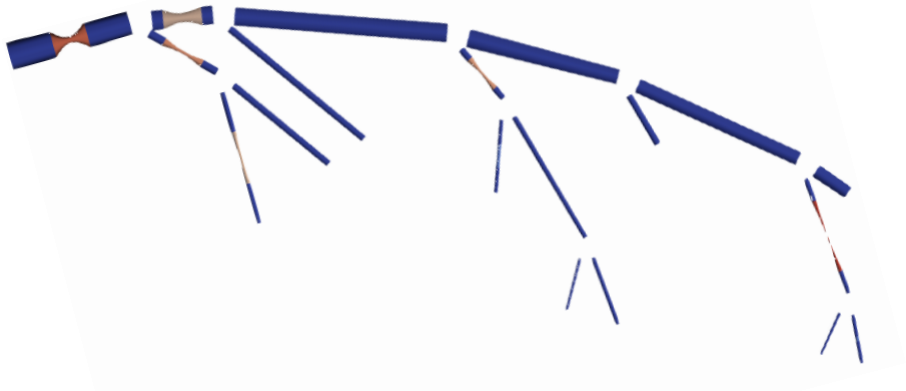


Figure 5.1: Illustration of a distal stenosis that received unrealistically small flow.

With the final database it is observed that there are less outliers, and the range of the pressure drop for each segment has reduced. By increasing the minimum length for the vessel segments, and using a uniform distribution for the stenosis length, a better physical

representation of the pressure drop over healthy segments was aquired. Vessel segments with the stenosis over the whole length of the segment were eliminated from the database.

5.3 Optimizing the ML model

Due to the time restriction of this thesis the grid search was performed on the preliminary database, and not on the final database. As mentioned, the best hyper-parameter configuration was adapted from the grid search to train a network with the final database.

5.3.1 Impact of the hyper-parameters

The hyper-parameters in the grid have different effect on the final results. For both the implemented ML models it is seen that the highest learning rate gives the best results when they are stopped after the same amount of epochs.

The learning rate has a bigger influence on the stenosis-ML model than of the 1D-ML model in the grid search. This is likely to come from the high staring value for the cost. With the low learning rate, the cost reduction in the stenosis-ML model has not slowed down yet. The difference in the final cost is $1.06 * 10^7$ when both sessions is stopped after 75000 epochs (Figure 4.14). Case 84 already yield a cost lower than the final cost in case 144 after 115 epochs. For the 1D model, the final cost difference for case 115 and case 175 is 4.45. Case 115 passes this cost already after 12 epochs. Hence, the highest learning rate has been considered the best option without exploring the behaviour if the number of epochs was increased for all three learning rates. Even though some oscillations can be observed in some of the models with a learning rate of 0.25, indicating that the learning rate is to big, and that the wights and biases are jumping across the optimal values.

The batch size is also important for how the network learns, as it determines how often the weights and biases are updated. In this grid search, the smallest batches have given the best results. A disadvantage of the small batch size compared to the biggest is that the training sessions is slower. It was observed that the network was able to predict a larger range of pressure drops when the batch size was reduced.

It is clear that the number of epochs is important for the ML models performance, and it has to be high enough. If the training is stopped after to few epochs the potential of the network is not utilized.

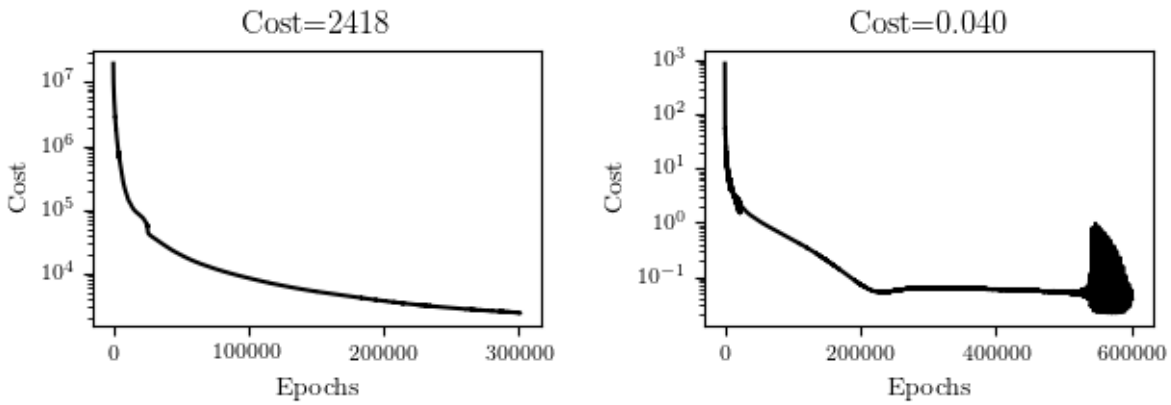
The depth of the network was limited to 2, 3 and 4 layers in the grid search due to computational capacity. This is the only hyper-parameter where the two different models had different optimal configurations, disregarding the batch size. For the 1D model a four layer model gave the best results, while a three layer model gave the best result for the stenosis model. For the stenosis model, the case with four layers and 50 neurons experienced trouble learning. It is interesting to see that the number of layers does not impact the final results drastically. As seen in Figure 4.19, the coefficient of determination only changes from 0.983 to 0.986 when the network is increased from 3 to 4 layers. It is not much smaller for the 2 layer network either where it is 0.967.

The number of neurons in each layer are more important than the number of layers for the accuracy of the model. Networks with only 10 neurons in each layer are only learning parts of the pressure drop range. It is noticed that both models have the best performance with 50 neurons in each layer. The networks with 100 neurons have a slightly lower score. This could mean that the optimal number of neurons lies in-between this two values.

In section 4.4.1, one training session that was trained with the double amount of training data compared to the other cases for the 1D model was introduced (case 2.119). The network was trained with the same hyper-parameter combination as case 1.119, both with the final database. The network that was trained with a larger database has a slightly worse coefficient of determination than the one obtained in case 1.119, 0.9997 against 0.9999. Case 1.119 was trained for 108000 compared to 51000 for case 2.119. So the weights and biases was updated approximately the same amount of times for the two networks. From this it is seen that the reduced amount of training data that has been used in the grid search do not compromise the performance of the network.

For the stenosis model the new units and the adapted scaler did not change the results for the model drastically. In the stenosis model the changes showed an improvement for the results. Especially when the cost vs epochs curve is compared, as illustrated in Figure 5.2. For the final database the initial cost is much higher than for the adapted database, after 300000 epochs it is still higher than the initial cost for case 1.84, where the cost is 849.

The best results are obtained at the edge of the sampled parameters for the batch size and the learning rate. Based on this it should have been performed a second search investigating if parameters outside the grid would give a better performance. This would also include a new range for the number of neurons in each layer.



(a) Case 1.84, final database

(b) Case 3.84, adapted final database

Figure 5.2: Cost vs epochs.

5.3.2 Unsuccessful networks

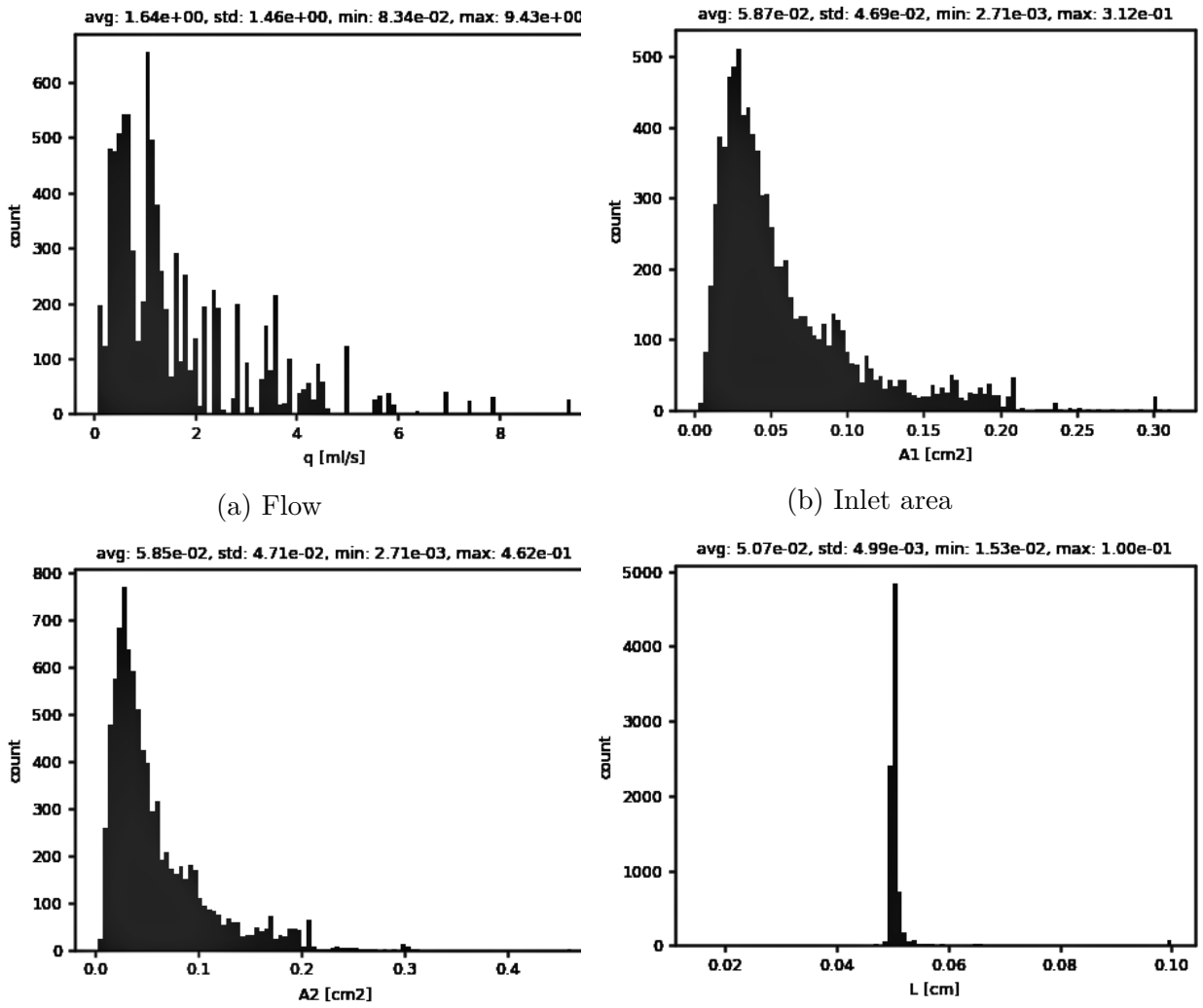
In the grid search for the stenosis model there is some of the networks that does not manage to learn and represent the training database. In these networks, all predicted pressure drops are -20 mmHg, similar as the mean value for the pressure drop in the training database.

This behaviour is mostly seen in networks with four hidden layers, and with 100 or 50 neurons, but it is not limited to these networks. It is also observed in networks with 30 neurons. Two of the cases where this happens are 88 and 89, interestingly is it not the case for case 90 that has the same configuration, but 100 neurons in each hidden layer. The reason behind this behaviour has not been identified. One possible reason can be vanishing gradient problem, meaning that the gradient in the deep neural network has vanished in the earlier layers [39].

5.4 Patient specific performance

5.4.1 1D-ML model

The patient specific prediction accuracy for both models were lower than expected, especially the 1D-ML model. As mentioned in section 4.6, the coefficient of determination is as low as 0.06. The feature distribution for the patient specific data is illustrated in Figure 5.3. It is seen that the features are within the range of the training database, but from the pressure drop range from Figure 4.31 it is clear that the area combinations are different from the training database.



(a) Flow

(b) Inlet area

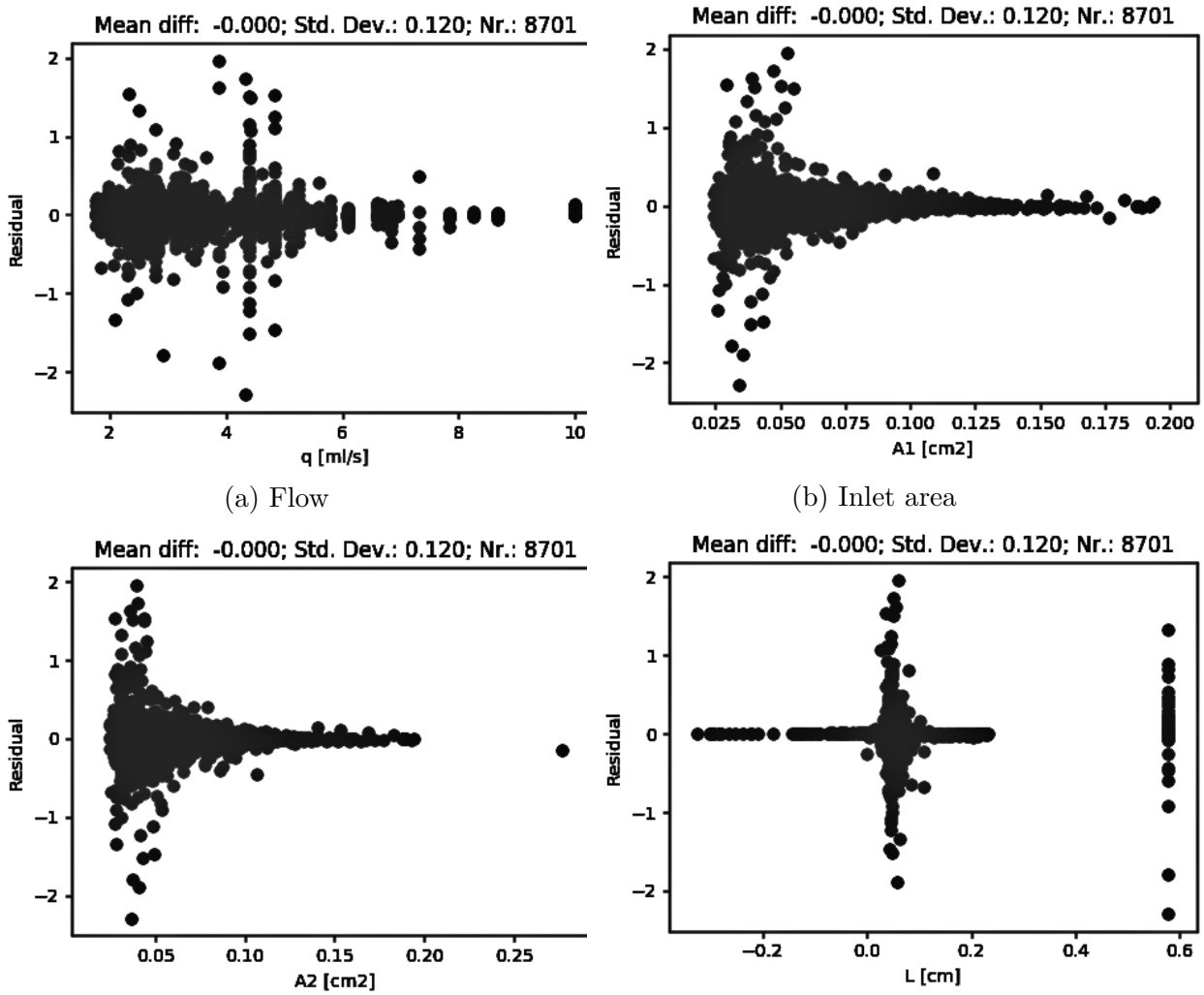
(c) Outlet area

(d) Segment length

Figure 5.3: Feature distribution for patient specific vessel segments.

From Figure 5.4, it becomes clear that the ML algorithm struggles most with the vessels segments with small areas. This is the segments with the highest residuals.

In the patient specific coronary trees, the vessel geometries have a non-linear behaviour compared to the straight vessel walls in the training database. There are more abrupt changes in the cross sectional area, and there are expanding sections allowing for pressure recovery. If the patient data is restricted to $0.98 =< \frac{A_2}{A_1} =< 1.00$, an area ratio that is represented in the synthetic database the models performance increases (see Figure 5.8). This shows the importance of a training database, representative for the whole behaviour of interest.



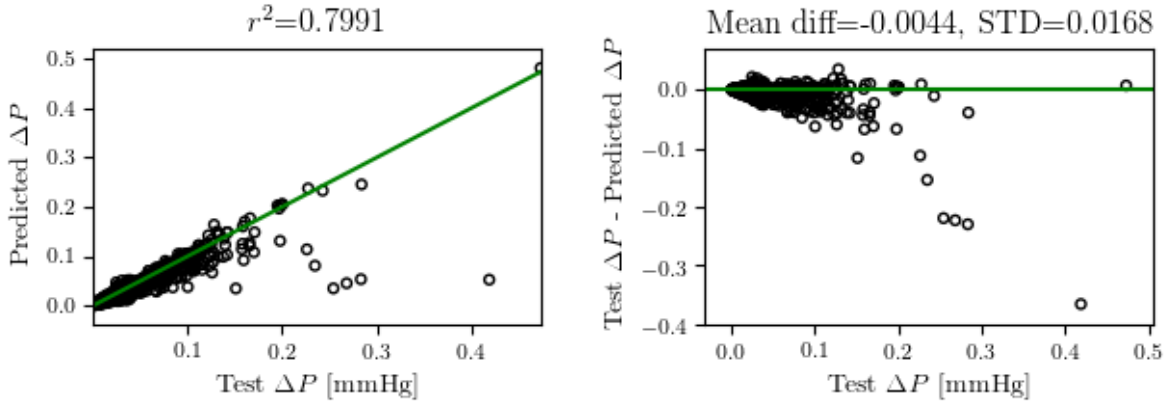
(a) Flow

(b) Inlet area

(c) Outlet area

(d) Segment length

Figure 5.4: Residual by feature, for patient specific vessel segments.



(a) Predicted ΔP vs. actual ΔP

(b) Residuals

Figure 5.5: Predictions with a restricted database of patient specific coronary geometries, 1D model.

5.4.2 Stenosis-ML model

The prediction accuracy for the stenosis model is better than for the 1D model. Still, the feature distribution from the synthetic training database does not correspond to the range in the patient specific database, seen in Figure 5.6. The range of the stenosis length for the 13 patients is not compatible with the stenosis length range in the synthetically generated database. The minimum stenosis length in the training database is 1.3 mm, while it exist patient specific stenoses that are shorter than 0.5 mm.

One of the stenoses in the patient specific database, provided an increase in the pressure due to an expanding geometry. This example was excluded from the database. The training database did not contain examples that represented this behaviour. Hence, the ML model does not have any prerequisite to predict a positive change in the pressure.

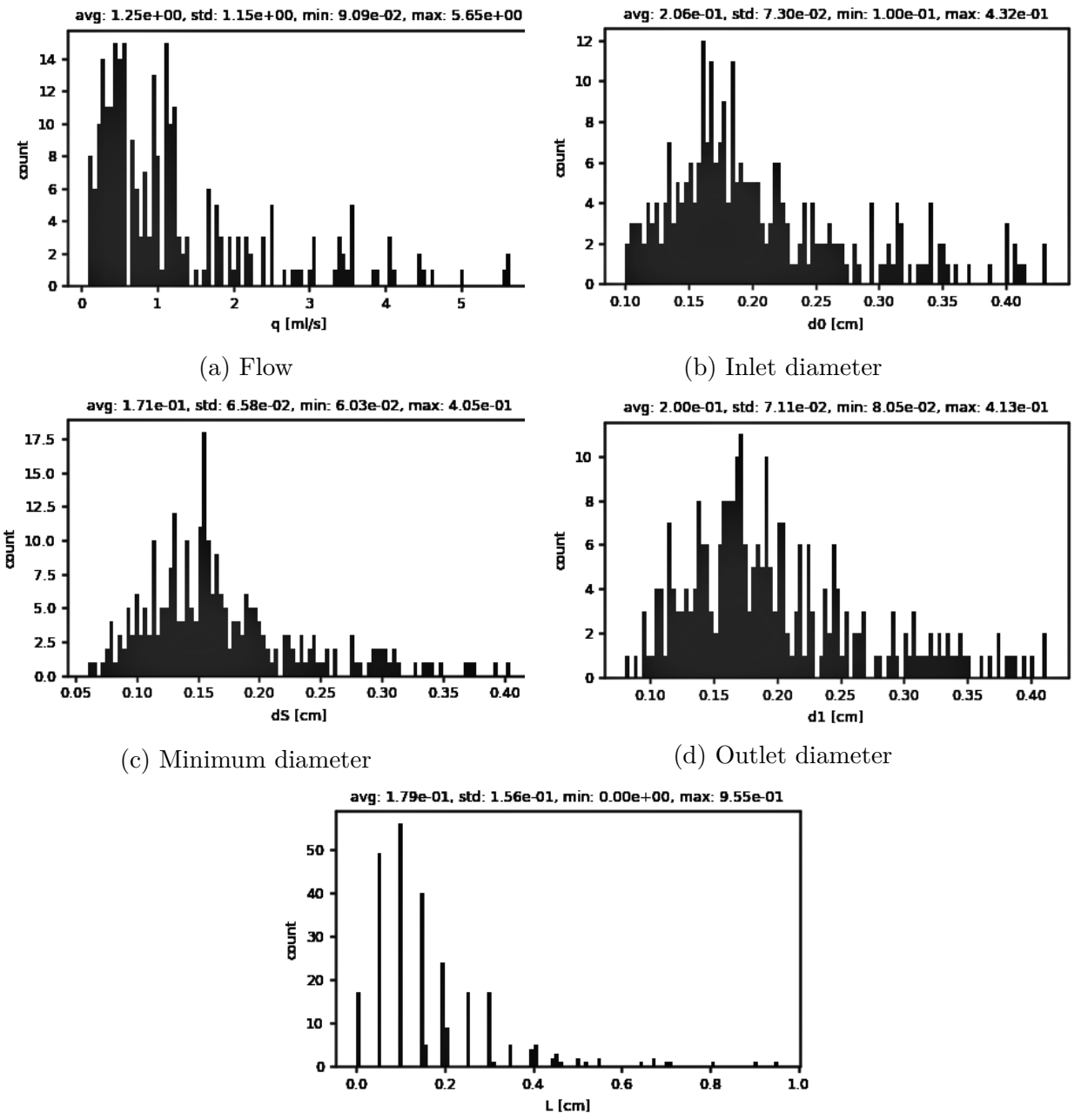
In Figure 5.7, the feature specific residual for the stenosis model is shown. The highest residuals for the stenosis length are found for stenosis length that is outside the range represented in the training database. The stenosis model has lower residuals for the stenoses with bigger diameters. This is the same as what was observed for the 1D model.

To exclude the stenoses with geometrical parameters not represented in the training database, stenoses with $L_s > 0.02$ mm, and $0.95 =< \frac{D_1}{D_0} <= 0.05$ was kept out. This improved the prediction accuracy, and the model obtained a coefficient of determination of 0.997. The maximum prediction error was reduced from approximately 5 mmHg to approximately 0.25 mmHg.

5.5 Evaluation of the ML based FFR predictions

For model-based FFR predictions, based on the the method presented by Itu et al. there are several limitations. These limitation are primary connected to the computational model for coronary flow. The ML algorithm is provided with data obtained from this model, therefore the accuracy of the ML algorithm can not exceed the accuracy of the mathematical model for coronary flow.

The steady state solver rely on accurate measurements of the coronary artery geometry, and so will the ML algorithm, where the coronary artery geometry will be provided from segmentation of CCTA images. In addition, the steady state solver rely on an accurate representation of system properties as blood density and viscosity, an accurate enforcement of boundary conditions and the accuracy of the physiological assumptions in



(a) Flow

(b) Inlet diameter

(c) Minimum diameter

(d) Outlet diameter

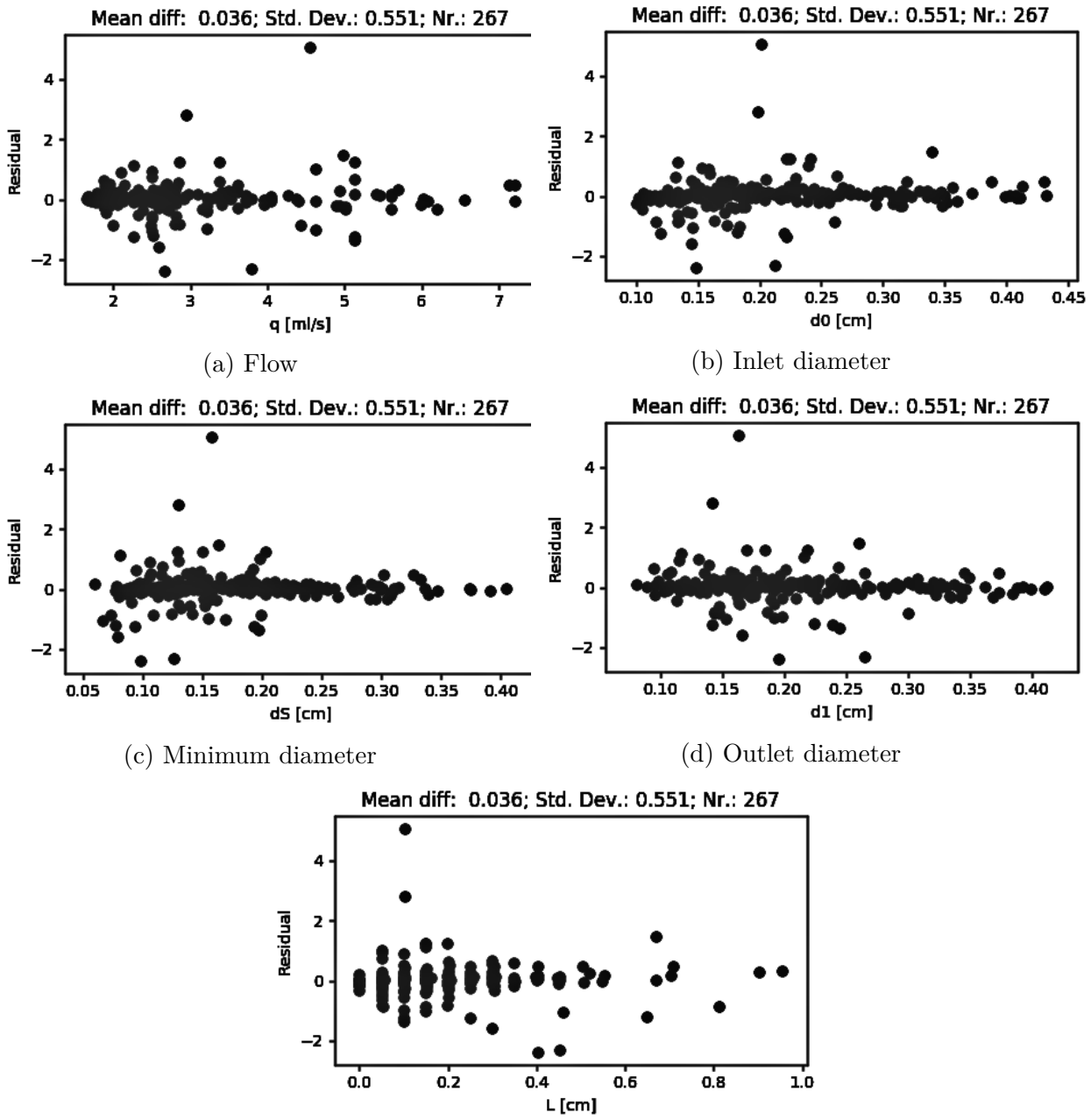
(e) Stenosis length

Figure 5.6: Feature distribution for patient specific stenoses.

the mathematical model.

In a clinical setting, a limitation on the real time delivery of FFR_{ML} is the preprocessing of data before it can be fed to the ML model. Which consist of lumen segmentation of CCTA images to prepare the geometric values. In addition, the cardiac output is derived from ultrasonographic examination.

The reduced-order, steady state solver presented in this thesis is very fast. Therefore, computational time was not the reason for the implementation of the ML algorithm. It is



(a) Flow

(b) Inlet diameter

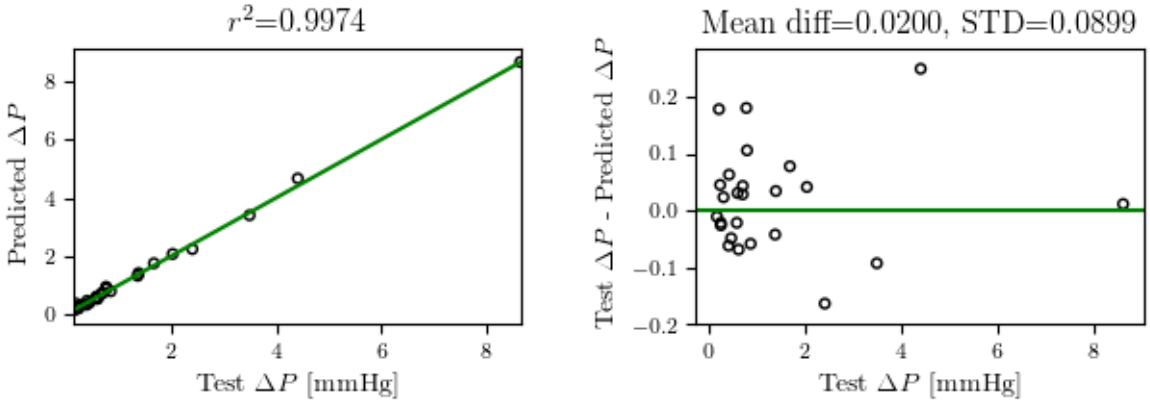
(c) Minimum diameter

(d) Outlet diameter

(e) Stenosis length

Figure 5.7: Residual by feature, for patient specific stenoses.

the first step towards a fully implemented ML model, where other patient specific features are incorporated to provide reliable FFR_{ML} predictions.



(a) (b) Residuals

Figure 5.8: Predictions with a restricted database of patient specific coronary geometries, stenosis model.

6 Conclusions and further work

In this thesis it is shown that it is possible to reproduce 1D computational model for flow in coronary arteries. The implemented ML algorithm shows good results when predicting the pressure drop over a segment of a healthy vessel. The hyper-parameters were optimized with a grid search. The architecture of best network for the 1D segments of the vessel was four hidden layers, consisting of 50 neurons each. The mean difference between the predicted values and the test values was $3.81 * 10; -7$ mmHg, and the standard deviation was $3.35 * 10^{-5}$ mmHg.

To find the pressure drop across a stenosis, the DNN was trained to solve Young and Tsai's stenosis model. The best network's architecture had three layers with 50 neurons in each. The mean difference was $1.94 * 10^{-6}$ mmHg, and the standard deviation was $9.4 * 10^{-4}$ mmHg.

The challenges were to generate a reliable training database with enough variation to represent the pathological variations of patients with suspected stable CAD. When the features were extracted from the 13 patients, and the ML models where used to predict the pressure drop it was clear that the training database did not contain enough variation to provide a good representation of the coronary anatomy. For a reduced patient specific database, the model predicted the pressure drop along the healthy vessels with a coefficient of determination of 0.799 and for the reduced database with the patient specific stenoses the ceofficient of determination was 0.997.

When the ML algorithms are providing good results for the 1D model and Young and Tsai's stenosis model, the next step is to train the network with patient specific data that accounts for features as the vessel curvature, tapering, shape factor to mention some of them. Such an approach is foreseen to result in a better prediction of the pressure drop on the coronary arteries compared to the 1D0D model that is used in this thesis.

This thesis was the first step towards a fully implemented ML approach for quantification of the pressure drop in coronary arteries due to CAD. Following steps towards this goal should be:

1. Enhance the virtual database by adding the feature ranges observed in the patient specific coronary trees to cover a broader range of pathological anatomies.

2. Perform an exhaustive hyper-parameter optimization.

3. Adapt the best DNN architecture found in this thesis, by introducing 3D simulations and 3D geometrical features. In order to get a better matching between 3D and 1D0D simulations, which would improve the 1D0D model.

7 References

References

[1] E. J. Benjamin, S. S. Virani, C. W. Callaway, A. M. Chamberlain, A. R. Chang, S. Cheng, S. E. Chiuve, M. Cushman, F. N. Delling, R. Deo *et al.*, "Heart disease and stroke statistics—2018 update: a report from the american heart association," *Circulation*, vol. 137, no. 12, pp. e67–e492, 2018.

[2] J. E. Hall, *Guyton and Hall Textbook of Medical Physiology*, ser. Guyton Physiology. Elsevier Health Sciences, 2010. [Online]. Available: https://books.google.no/books?id=Po0zyO0BFzwC

[3] P. A. Heidenreich, J. G. Trogdon, O. A. Khavjou, J. Butler, K. Dracup, M. D. Ezekowitz, E. A. Finkelstein, Y. Hong, S. C. Johnston, A. Khera, D. M. Lloyd-Jones, S. A. Nelson, G. Nichol, D. Orenstein, P. W. Wilson, and Y. J. Woo, "Forecasting the future of cardiovascular disease in the united states," *Circulation*, vol. 123, no. 8, pp. 933–944, 2011. [Online]. Available: http://circ.ahajournals.org/content/123/8/933

[4] T. F. Members, G. Montalescot, U. Sechtem, S. Achenbach, F. Andreotti, C. Arden, A. Budaj, R. Bugiardini, F. Crea, T. Cuisset *et al.*, "2013 esc guidelines on the management of stable coronary artery disease: the task force on the management of stable coronary artery disease of the european society of cardiology," *European heart journal*, vol. 34, no. 38, pp. 2949–3003, 2013.

[5] N. H. Pijls, B. de Bruyne, K. Peels, P. H. van der Voort, H. J. Bonnier, J. Bartunek, and J. J. Koolen, "Measurement of fractional flow reserve to assess the functional severity of coronary-artery stenoses," *New England Journal of Medicine*, vol. 334, no. 26, pp. 1703–1708, 1996.

[6] T. Kimura, H. Shiomi, S. Kuribayashi, T. Isshiki, S. Kanazawa, H. Ito, S. Ikeda, B. Forrest, C. K. Zarins, M. A. Hlatky, and B. L. Norgaard, "Cost analysis of non-invasive fractional flow reserve derived from coronary computed tomographic angiography in japan," *Cardiovascular Intervention and Therapeutics*, vol. 30, no. 1, pp. 38–44, Jan 2015. [Online]. Available: https://doi.org/10.1007/s12928-014-0285-1

[7] S. Sankaran, L. J. Grady, and C. A. Taylor, "Real-time sensitivity analysis of blood flow simulations to lumen segmentation uncertainty," in *International Conference on*

Medical Image Computing and Computer-Assisted Intervention. Springer, 2014, pp. 1–8.

[8] BruceBlaus. (2013) Wikimedia commons: Coronary arteries. Accessed: 15.06.2018. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/9/93/Blausen_0256_CoronaryArteries_02.png

[9] "Chapter 1 - pathophysiology of atherosclerosis," in *Coronary Artery Disease*, D. Tousoulis, Ed. Academic Press, 2018, pp. 31 – 41.

[10] BruceBlaus. (2014) Wikimedia commons: Coronary artery disease. Accessed: 16.06.2018. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/6/6d/Blausen_0259_CoronaryArteryDisease_02.png

[11] J. Röther, M. Moshage, D. Dey, C. Schwemmer, M. Tröbs, F. Blachutzik, S. Achenbach, C. Schlundt, and M. Marwan, "Comparison of invasively measured ffr with ffr derived from coronary ct angiography for detection of lesion-specific ischemia: Results from a pc-based prototype algorithm," *Journal of cardiovascular computed tomography*, 2018.

[12] R. Wang, M. Renker, U. J. Schoepf, J. L. Wichmann, S. R. Fuller, J. D. Rier, R. R. Bayer, D. H. Steinberg, C. N. D. Cecco, and S. Baumann, "Diagnostic value of quantitative stenosis predictors with coronary ct angiography compared to invasive fractional flow reserve," *European Journal of Radiology*, vol. 84, no. 8, pp. 1509 – 1515, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0720048X15002387

[13] C. Tesche, C. N. De Cecco, M. H. Albrecht, T. M. Duguay, R. R. Bayer, S. E. Litwin, D. H. Steinberg, and U. J. Schoepf, "Coronary ct angiography–derived fractional flow reserve," *Radiology*, vol. 285, no. 1, pp. 17–33, 2017.

[14] B. L. Nørgaard, J. Leipsic, S. Gaur, S. Seneviratne, B. S. Ko, H. Ito, J. M. Jensen, L. Mauri, B. D. Bruyne, H. Bezerra, K. Osawa, M. Marwan, C. Naber, A. Erglis, S.-J. Park, E. H. Christiansen, A. Kaltoft, J. F. Lassen, H. E. Bøtker, and S. Achenbach, "Diagnostic performance of noninvasive fractional flow reserve derived from coronary computed tomography angiography in suspected coronary artery disease: The nxt trial (analysis of coronary blood flow using ct angiography: Next steps)," *Journal of the American College of Cardiology*, vol. 63, no. 12, pp. 1145 – 1155, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S073510971400165X

[15] L. Itu, P. Sharma, V. Mihalef, A. Kamen, C. Suciu, and D. Lomaniciu, "A patient-specific reduced-order model for coronary circulation," in *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*. IEEE, 2012, pp. 832–835.

[16] K. Ri, K. K. Kumamaru, S. Fujimoto, Y. Kawaguchi, T. Dohi, S. Yamada, K. Takamura, Y. Kogure, N. Yamada, E. Kato *et al.*, "Noninvasive computed tomography–derived fractional flow reserve based on structural and fluid analysis: Reproducibility of on-site determination by unexperienced observers," *Journal of computer assisted tomography*, vol. 42, no. 2, pp. 256–262, 2018.

[17] L. Itu, S. Rapaka, T. Passerini, B. Georgescu, C. Schwemmer, M. Schoebinger, T. Flohr, P. Sharma, and D. Comaniciu, "A machine-learning approach for computation of fractional flow reserve from coronary computed tomography," *Journal of Applied Physiology*, vol. 121, no. 1, pp. 42–52, 2016.

[18] S. Sankaran, L. Grady, and C. Taylor, "Fast geometric sensitivity analysis in hemodynamic simulations using a machine learning approach," in *APS Division of Fluid Dynamics Meeting Abstracts*, 2013.

[19] L. Formaggia, J. Gerbeau, F. Nobile, and A. Quarteroni, "On the coupling of 3d and 1d navier–stokes equations for flow problems in compliant vessels," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 6, pp. 561 – 582, 2001, minisymposium on Methods for Flow Simulation and Modeling. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045782501003024

[20] W. F. Boron and E. L. Boulpaep, *Medical Physiology, 2e Updated Edition E-Book: with STUDENT CONSULT Online Access*. Elsevier Health Sciences, 2012.

[21] T. Sochi, "Non-newtonian rheology in blood circulation," *arXiv preprint arXiv:1306.2067*, 2013.

[22] N. Westerhof, N. Stergiopulos, and M. I. Noble, *Snapshots of hemodynamics: an aid for clinical research and graduate education*. Springer Science & Business Media, 2010.

[23] Y. Çengel and J. Cimbala, *Fluid Mechanics: Fundamentals and Applications*, ser. Çengel series in engineering thermal-fluid sciences. McGraw-Hill Higher Education, 2010. [Online]. Available: https://books.google.no/books?id=GhWVmQEACAAJ

[24] L. Formaggia, A. Quarteroni, and A. Veneziani, *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*. Springer Science & Business Media,

2010, vol. 1.

[25] R. E. Mates, R. L. Gupta, A. C. Bell, and F. J. Klocke, "Fluid dynamics of coronary artery stenosis." *Circulation Research*, vol. 42, no. 1, pp. 152–162, 1978.

[26] M. Zamir, *The physics of coronary blood flow*. Springer Science & Business Media, 2006.

[27] K. L. Gould, K. Lipscomb, and G. W. Hamilton, "Physiologic basis for assessing critical coronary stenosis: instantaneous flow response and regional distribution during coronary hyperemia as measures of coronary flow reserve," *American Journal of Cardiology*, vol. 33, no. 1, pp. 87–94, 1974.

[28] N. G. Uren, J. A. Melin, B. De Bruyne, W. Wijns, T. Baudhuin, and P. G. Camici, "Relation between myocardial blood flow and the severity of coronary-artery stenosis," *New England Journal of Medicine*, vol. 330, no. 25, pp. 1782–1788, 1994.

[29] R. F. Wilson, K. Wyche, B. V. Christensen, S. Zimmer, and D. D. Laxson, "Effects of adenosine on human coronary arterial circulation." *Circulation*, vol. 82, no. 5, pp. 1595–1606, 1990.

[30] Y. Huo, M. Svendsen, J. S. Choy, Z.-D. Zhang, and G. S. Kassab, "A validated predictive model of coronary fractional flow reserve," *Journal of The Royal Society Interface*, p. rsif20110605, 2011.

[31] C. Bulant, P. Blanco, G. M. Talou, C. G. Bezerra, P. Lemos, and R. Feijóo, "A head-to-head comparison between ct- and ivus-derived coronary blood flow models," *Journal of Biomechanics*, vol. 51, pp. 65 – 76, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002192901631260X

[32] P. D. Morris, D. A. S. Soto, J. F. Feher, D. Rafiroiu, A. Lungu, S. Varma, P. V. Lawford, D. R. Hose, and J. P. Gunn, "Fast virtual fractional flow reserve based upon steady-state computational fluid dynamics analysis: results from the virtu-fast study," *JACC: Basic to Translational Science*, vol. 2, no. 4, pp. 434–446, 2017.

[33] C. D. Murray, "The physiological principle of minimum work i. the vascular system and the cost of blood volume," *Proceedings of the National Academy of Sciences*, vol. 12, no. 3, pp. 207–214, 1926.

[34] M. S. Olufsen, C. S. Peskin, W. Y. Kim, E. M. Pedersen, A. Nadim, and J. Larsen, "Numerical simulation and experimental validation of blood

flow in arteries with structured-tree outflow conditions," *Annals of Biomedical Engineering*, vol. 28, no. 11, pp. 1281–1299, Nov 2000. [Online]. Available: https://doi.org/10.1114/1.1326031

[35] D. F. Young and F. Y. Tsai, "Flow characteristics in models of arterial stenoses—i. steady flow," *Journal of biomechanics*, vol. 6, no. 4, pp. 395–402, 1973.

[36] F. Liang, K. Fukasaku, H. Liu, and S. Takagi, "A computational model study of the influence of the anatomy of the circle of willis on cerebral hyperperfusion following carotid artery surgery," *Biomedical engineering online*, vol. 10, no. 1, p. 84, 2011.

[37] N. H. Pijls, J. A. van Son, R. L. Kirkeeide, B. De Bruyne, and K. L. Gould, "Experimental basis of determining maximum coronary, myocardial, and collateral blood flow by pressure measurements for assessing functional stenosis severity before and after percutaneous transluminal coronary angioplasty." *Circulation*, vol. 87, no. 4, pp. 1354–1367, 1993. [Online]. Available: http://circ.ahajournals.org/content/87/4/1354

[38] J. Bell, *Machine learning: hands-on for developers and technical professionals.* John Wiley & Sons, 2014.

[39] M. A. Nielsen, *Neutral Networks and Deep Learning.* Determination Press, 2015.

[40] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec 1943. [Online]. Available: https://doi.org/10.1007/BF02478259

[41] S. Pattanayak, *Introduction to Deep-Learning Concepts and TensorFlow.* Berkeley, CA: Apress, 2017, pp. 89–152. [Online]. Available: https://doi.org/10.1007/978-1-4842-3096-1_2

[42] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade.* Springer, 1998, pp. 9–50.

[43] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2188385.2188395

[44] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade.* Springer, 2012, pp. 437–478.

[45] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *Journal of machine learning research*, vol. 10, no. Jan, pp. 1–40, 2009.

[46] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[47] A.-A. A. E. B. E. H. K. Atta-Alla S, El Sawa E, "Morphometric study of the right coronary artery," *International Journal of Anatomy and Research*, vol. 3, pp. 1362–1370, Sep 2015.

[48] S. G. Ellis, M. G. Vandormael, M. J. Cowley, G. DiSciascio, U. Deligonul, E. J. Topol, and T. M. Bulle, "Coronary morphologic and clinical determinants of procedural outcome with angioplasty for multivessel coronary disease. implications for patient selection. multivessel angioplasty prognosis study group." *Circulation*, vol. 82, no. 4, pp. 1193–1202, 1990.

[49] S. Sakamoto, S. Takahashi, A. U. Coskun, M. I. Papafaklis, A. Takahashi, S. Saito, P. H. Stone, and C. L. Feldman, "Relation of distribution of coronary blood flow volume to coronary artery dominance," *American Journal of cardiology*, vol. 111, no. 10, pp. 1420–1424, 2013.

[50] L. Müller, "Patient-specific ffr simulations using a 3d rigid domain framework: first experiences," nTNU.

[51] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

[52] N. Xiao, J. Alastruey, and C. Alberto Figueroa, "A systematic comparison between 1-D and 3-D hemodynamics in compliant arterial models," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 30, no. 2, pp. 204–231, Feb. 2014.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[54] S. Pattanayak, "Pro deep learning with tensorflow."

[55] J. T. Dodge, B. G. Brown, E. L. Bolson, and H. T. Dodge, "Lumen diameter of normal human coronary arteries. influence of age, sex, anatomic variation, and left ventricular hypertrophy or dilation." *Circulation*, vol. 86, no. 1, pp. 232–246, 1992.

8 Appendix

(A) Feature distribution in the preliminary database

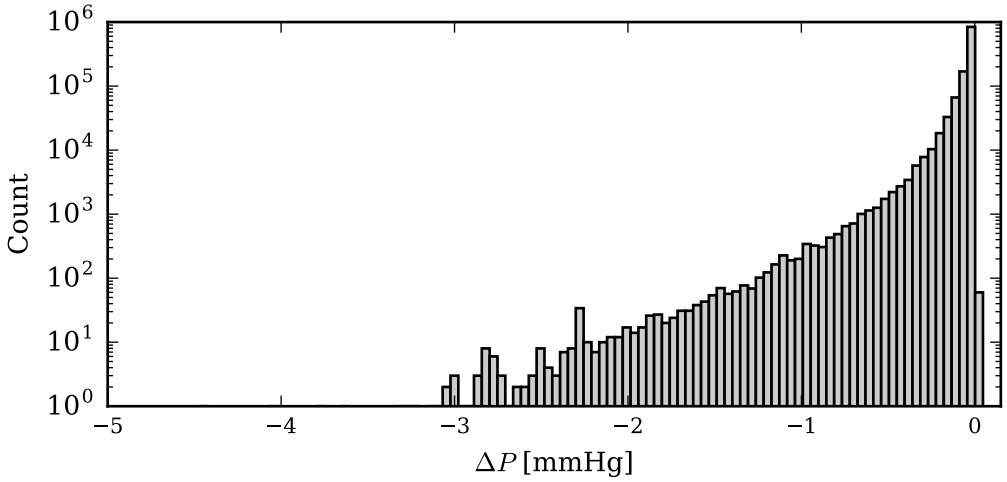


Figure 8.1: Pressure drop distribution for the preliminary 1D database.

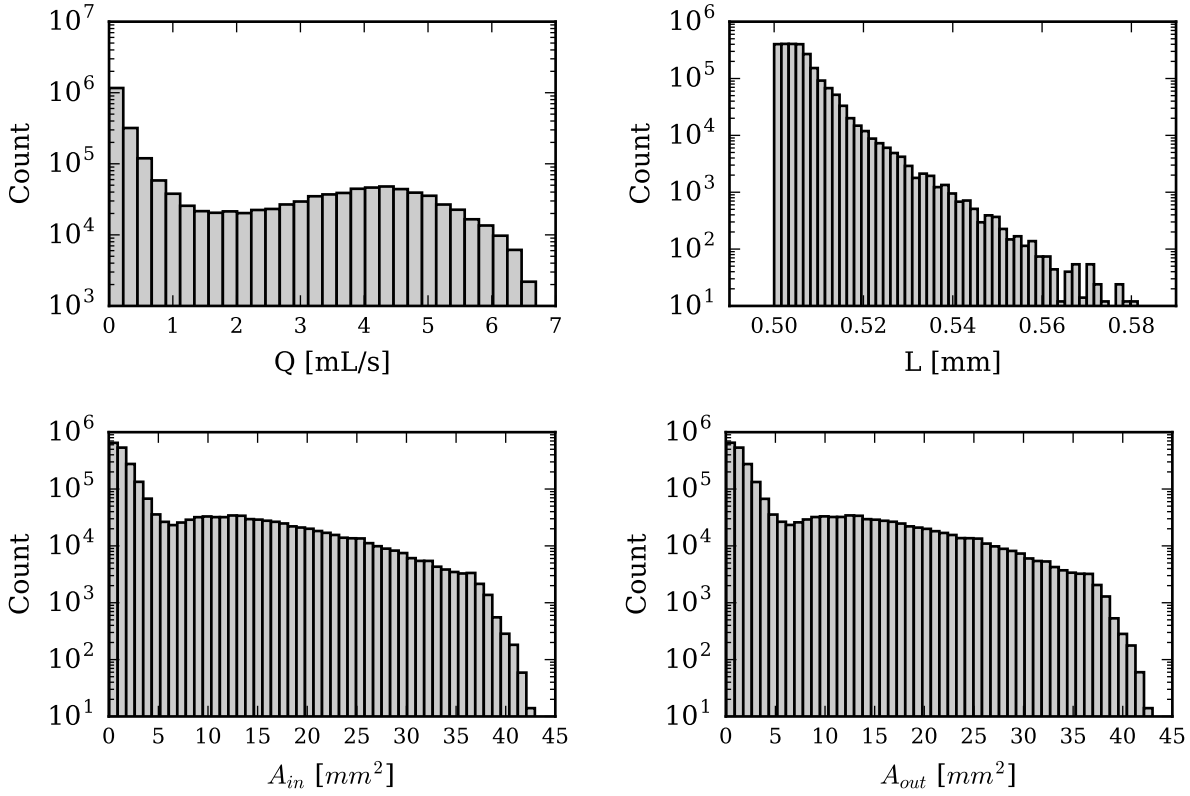


Figure 8.2: Features for the 1D-ML model

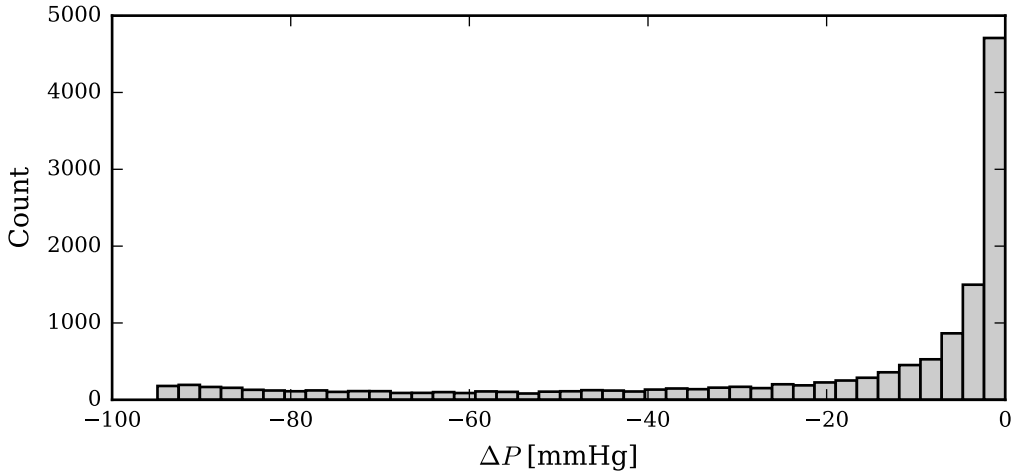


Figure 8.3: Pressure drop distribution for the preliminary stenosis database.

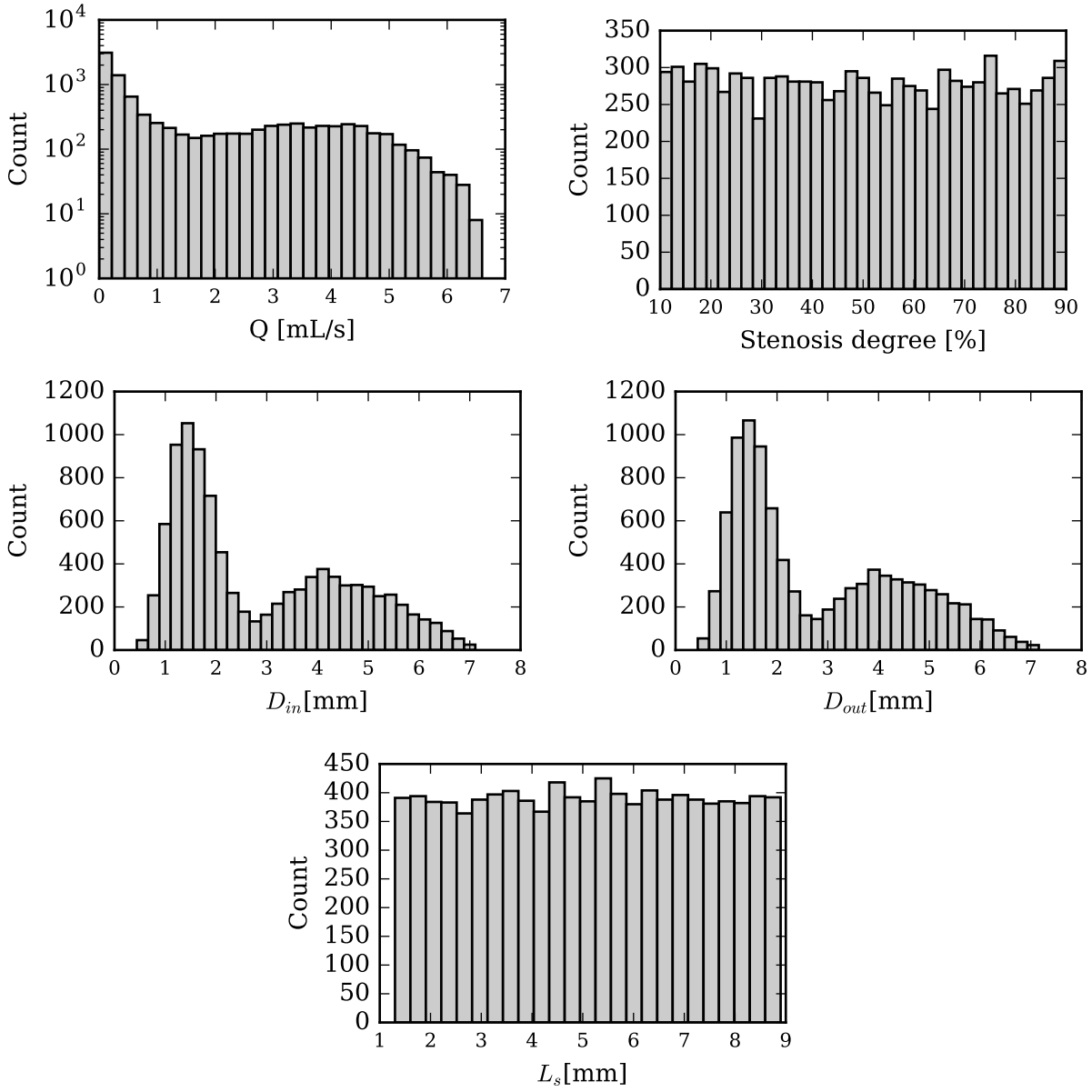


Figure 8.4: Features for the stenosis-ML model.

(B) Results from grid search, 1D sections

Results from grid search, 1D-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
1	0.1	20,000	10 10	10.093	0.943	$4.43{\cdot}10^{-5}$	$2.383{\cdot}10^{-2}$
2	0.1	20,000	20 20	10.445	0.941	$1.631{\cdot}10^{-4}$	$2.424{\cdot}10^{-2}$
3	0.1	20,000	30 30	8.686	0.951	$1.506{\cdot}10^{-4}$	$2.211{\cdot}10^{-2}$
4	0.1	20,000	50 50	6.581	0.963	$1.783{\cdot}10^{-4}$	$1.924{\cdot}10^{-2}$
5	0.1	20,000	100 100	4.534	0.974	$3.269{\cdot}10^{-4}$	$1.597{\cdot}10^{-2}$
6	0.1	20,000	10 10 10	5.468	0.969	$-3.047{\cdot}10^{-5}$	$1.754{\cdot}10^{-2}$
7	0.1	20,000	20 20 20	5.662	0.968	$1.516{\cdot}10^{-4}$	$1.785{\cdot}10^{-2}$
8	0.1	20,000	30 30 30	4.928	0.972	$1.923{\cdot}10^{-4}$	$1.665{\cdot}10^{-2}$
9	0.1	20,000	50 50 50	4.293	0.976	$2.573{\cdot}10^{-4}$	$1.554{\cdot}10^{-2}$
10	0.1	20,000	100 100 100	3.6	0.98	$3.831{\cdot}10^{-4}$	$1.423{\cdot}10^{-2}$
11	0.1	20,000	10 10 10 10	8.141	0.954	$3.754{\cdot}10^{-4}$	$2.14{\cdot}10^{-2}$
12	0.1	20,000	20 20 20 20	4.428	0.975	$3.106{\cdot}10^{-4}$	$1.578{\cdot}10^{-2}$
13	0.1	20,000	30 30 30 30	4.3	0.976	$1.833{\cdot}10^{-4}$	$1.555{\cdot}10^{-2}$
14	0.1	20,000	50 50 50 50	3.676	0.979	$3.953{\cdot}10^{-4}$	$1.438{\cdot}10^{-2}$
15	0.1	20,000	100 100 100 100	3.379	0.981	$-5.393{\cdot}10^{-5}$	$1.379{\cdot}10^{-2}$
16	0.1	5,000	10 10	7.589	0.957	$1.827{\cdot}10^{-4}$	$2.066{\cdot}10^{-2}$
17	0.1	5,000	20 20	5.906	0.967	$2.283{\cdot}10^{-4}$	$1.823{\cdot}10^{-2}$
18	0.1	5,000	30 30	5.699	0.968	$2.772{\cdot}10^{-4}$	$1.79{\cdot}10^{-2}$
19	0.1	5,000	50 50	5.275	0.97	$2.818{\cdot}10^{-4}$	$1.723{\cdot}10^{-2}$
20	0.1	5,000	100 100	3.687	0.979	$2.946{\cdot}10^{-4}$	$1.44{\cdot}10^{-2}$
21	0.1	5,000	10 10 10	5.159	0.971	$1.333{\cdot}10^{-4}$	$1.704{\cdot}10^{-2}$
22	0.1	5,000	20 20 20	4.461	0.975	$2.422{\cdot}10^{-4}$	$1.584{\cdot}10^{-2}$
23	0.1	5,000	30 30 30	3.841	0.978	$3.551{\cdot}10^{-4}$	$1.47{\cdot}10^{-2}$
24	0.1	5,000	50 50 50	3.691	0.979	$4.642{\cdot}10^{-4}$	$1.44{\cdot}10^{-2}$
25	0.1	5,000	100 100 100	2.776	0.984	$2.793{\cdot}10^{-4}$	$1.249{\cdot}10^{-2}$
26	0.1	5,000	10 10 10 10	4.738	0.973	$3.018{\cdot}10^{-4}$	$1.632{\cdot}10^{-2}$
27	0.1	5,000	20 20 20 20	3.885	0.978	$5.238{\cdot}10^{-4}$	$1.477{\cdot}10^{-2}$
28	0.1	5,000	30 30 30 30	3.552	0.98	$4.864{\cdot}10^{-4}$	$1.413{\cdot}10^{-2}$
29	0.1	5,000	50 50 50 50	3.357	0.981	$2.5{\cdot}10^{-4}$	$1.374{\cdot}10^{-2}$
30	0.1	5,000	100 100 100 100	2.791	0.984	$6.553{\cdot}10^{-5}$	$1.253{\cdot}10^{-2}$

Results from grid search, 1D-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
31	0.1	10,000	10 10	11.578	0.935	$1.151 \cdot 10^{-4}$	$2.552 \cdot 10^{-2}$
32	0.1	10,000	20 20	9.324	0.947	$1.186 \cdot 10^{-4}$	$2.29 \cdot 10^{-2}$
33	0.1	10,000	30 30	6.492	0.963	$1.138 \cdot 10^{-4}$	$1.911 \cdot 10^{-2}$
34	0.1	10,000	50 50	5.685	0.968	$5.958 \cdot 10^{-5}$	$1.788 \cdot 10^{-2}$
35	0.1	10,000	100 100	4.017	0.977	$-1.495 \cdot 10^{-4}$	$1.503 \cdot 10^{-2}$
36	0.1	10,000	10 10 10	5.567	0.969	$1.248 \cdot 10^{-5}$	$1.77 \cdot 10^{-2}$
37	0.1	10,000	20 20 20	5.456	0.969	$-1.635 \cdot 10^{-4}$	$1.752 \cdot 10^{-2}$
38	0.1	10,000	30 30 30	4.267	0.976	$-1.743 \cdot 10^{-4}$	$1.549 \cdot 10^{-2}$
39	0.1	10,000	50 50 50	4.164	0.976	$-2.319 \cdot 10^{-4}$	$1.53 \cdot 10^{-2}$
40	0.1	10,000	100 100 100	3.213	0.982	$-4.089 \cdot 10^{-4}$	$1.344 \cdot 10^{-2}$
41	0.1	10,000	10 10 10 10	5.313	0.97	$-1.717 \cdot 10^{-4}$	$1.729 \cdot 10^{-2}$
42	0.1	10,000	20 20 20 20	4.446	0.975	$-2.218 \cdot 10^{-4}$	$1.581 \cdot 10^{-2}$
43	0.1	10,000	30 30 30 30	3.91	0.978	$-2.699 \cdot 10^{-4}$	$1.483 \cdot 10^{-2}$
44	0.1	10,000	50 50 50 50	4.003	0.977	$-3.06 \cdot 10^{-4}$	$1.5 \cdot 10^{-2}$
45	0.1	10,000	100 100 100 100	2.831	0.984	$-8.541 \cdot 10^{-5}$	$1.262 \cdot 10^{-2}$
46	0.1	1,000	10 10	7.741	0.956	$1.676 \cdot 10^{-4}$	$2.087 \cdot 10^{-2}$
47	0.1	1,000	20 20	5.907	0.967	$2.039 \cdot 10^{-4}$	$1.823 \cdot 10^{-2}$
48	0.1	1,000	30 30	4.868	0.973	$2.435 \cdot 10^{-4}$	$1.655 \cdot 10^{-2}$
49	0.1	1,000	50 50	4.673	0.974	$4.069 \cdot 10^{-4}$	$1.621 \cdot 10^{-2}$
50	0.1	1,000	100 100	3.486	0.98	$4.354 \cdot 10^{-4}$	$1.4 \cdot 10^{-2}$
51	0.1	1,000	10 10 10	5.415	0.969	$1.61 \cdot 10^{-4}$	$1.745 \cdot 10^{-2}$
52	0.1	1,000	20 20 20	4.153	0.977	$3.334 \cdot 10^{-4}$	$1.528 \cdot 10^{-2}$
53	0.1	1,000	30 30 30	3.977	0.978	$3.405 \cdot 10^{-4}$	$1.495 \cdot 10^{-2}$
54	0.1	1,000	50 50 50	3.448	0.981	$5.529 \cdot 10^{-4}$	$1.392 \cdot 10^{-2}$
55	0.1	1,000	100 100 100	2.795	0.984	$3.542 \cdot 10^{-4}$	$1.254 \cdot 10^{-2}$
56	0.1	1,000	10 10 10 10	4.213	0.976	$2.18 \cdot 10^{-4}$	$1.539 \cdot 10^{-2}$
57	0.1	1,000	20 20 20 20	3.742	0.979	$4.809 \cdot 10^{-4}$	$1.45 \cdot 10^{-2}$
58	0.1	1,000	30 30 30 30	3.539	0.98	$5 \cdot 10^{-4}$	$1.41 \cdot 10^{-2}$
59	0.1	1,000	50 50 50 50	3.454	0.98	$3.011 \cdot 10^{-4}$	$1.394 \cdot 10^{-2}$
60	0.1	1,000	100 100 100 100	2.55	0.986	$-2.918 \cdot 10^{-5}$	$1.198 \cdot 10^{-2}$
61	0.25	20,000	10 10	10.451	0.941	$1.459 \cdot 10^{-4}$	$2.425 \cdot 10^{-2}$
62	0.25	20,000	20 20	6.026	0.966	$4.471 \cdot 10^{-4}$	$1.841 \cdot 10^{-2}$
63	0.25	20,000	30 30	7.358	0.958	$2.367 \cdot 10^{-4}$	$2.034 \cdot 10^{-2}$

Results from grid search, 1D-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
64	0.25	20,000	50 50	5.283	0.97	$2.995{\cdot}10^{-4}$	$1.724{\cdot}10^{-2}$
65	0.25	20,000	100 100	4.085	0.977	$8.868{\cdot}10^{-4}$	$1.513{\cdot}10^{-2}$
66	0.25	20,000	10 10 10	5.721	0.968	$1.865{\cdot}10^{-4}$	$1.794{\cdot}10^{-2}$
67	0.25	20,000	20 20 20	4.728	0.973	$1.158{\cdot}10^{-4}$	$1.631{\cdot}10^{-2}$
68	0.25	20,000	30 30 30	3.982	0.978	$3.544{\cdot}10^{-4}$	$1.496{\cdot}10^{-2}$
69	0.25	20,000	50 50 50	3.579	0.98	$6.027{\cdot}10^{-4}$	$1.418{\cdot}10^{-2}$
70	0.25	20,000	100 100 100	2.516	0.986	$4.312{\cdot}10^{-4}$	$1.189{\cdot}10^{-2}$
71	0.25	20,000	10 10 10 10	4.966	0.972	$2.665{\cdot}10^{-4}$	$1.671{\cdot}10^{-2}$
72	0.25	20,000	20 20 20 20	3.92	0.978	$3.355{\cdot}10^{-4}$	$1.485{\cdot}10^{-2}$
73	0.25	20,000	30 30 30 30	3.257	0.982	$1.088{\cdot}10^{-4}$	$1.354{\cdot}10^{-2}$
74	0.25	20,000	50 50 50 50	3.365	0.981	$5.01{\cdot}10^{-4}$	$1.375{\cdot}10^{-2}$
75	0.25	20,000	100 100 100 100	3.008	0.983	$1.884{\cdot}10^{-4}$	$1.301{\cdot}10^{-2}$
76	0.25	5,000	10 10	6.728	0.962	$2.437{\cdot}10^{-4}$	$1.945{\cdot}10^{-2}$
77	0.25	5,000	20 20	6.69	0.962	$3.564{\cdot}10^{-4}$	$1.94{\cdot}10^{-2}$
78	0.25	5,000	30 30	6.302	0.964	$3.734{\cdot}10^{-4}$	$1.883{\cdot}10^{-2}$
79	0.25	5,000	50 50	5.292	0.97	$5.094{\cdot}10^{-4}$	$1.725{\cdot}10^{-2}$
80	0.25	5,000	100 100	3.405	0.981	$7.237{\cdot}10^{-4}$	$1.382{\cdot}10^{-2}$
81	0.25	5,000	10 10 10	4.294	0.976	$3.821{\cdot}10^{-4}$	$1.554{\cdot}10^{-2}$
82	0.25	5,000	20 20 20	3.476	0.98	$4.517{\cdot}10^{-4}$	$1.398{\cdot}10^{-2}$
83	0.25	5,000	30 30 30	3.561	0.98	$5.116{\cdot}10^{-4}$	$1.415{\cdot}10^{-2}$
84	0.25	5,000	50 50 50	2.772	0.984	$3.429{\cdot}10^{-4}$	$1.248{\cdot}10^{-2}$
85	0.25	5,000	100 100 100	1.878	0.989	$-4.605{\cdot}10^{-4}$	$1.027{\cdot}10^{-2}$
86	0.25	5,000	10 10 10 10	4.198	0.976	$6.765{\cdot}10^{-4}$	$1.535{\cdot}10^{-2}$
87	0.25	5,000	20 20 20 20	3.571	0.98	$3.675{\cdot}10^{-4}$	$1.417{\cdot}10^{-2}$
88	0.25	5,000	30 30 30 30	3.26	0.982	$4.433{\cdot}10^{-4}$	$1.353{\cdot}10^{-2}$
89	0.25	5,000	50 50 50 50	2.597	0.985	$1.144{\cdot}10^{-4}$	$1.209{\cdot}10^{-2}$
90	0.25	5,000	100 100 100 100	2.461	0.986	$1.818{\cdot}10^{-4}$	$1.177{\cdot}10^{-2}$
91	0.25	10,000	10 10	5.935	0.966	$7.715{\cdot}10^{-5}$	$1.827{\cdot}10^{-2}$
92	0.25	10,000	20 20	5.478	0.969	$5.107{\cdot}10^{-5}$	$1.756{\cdot}10^{-2}$
93	0.25	10,000	30 30	7.052	0.96	$4.476{\cdot}10^{-5}$	$1.992{\cdot}10^{-2}$
94	0.25	10,000	50 50	4.606	0.974	$-5.879{\cdot}10^{-5}$	$1.61{\cdot}10^{-2}$
95	0.25	10,000	100 100	3.485	0.98	$-4.681{\cdot}10^{-4}$	$1.399{\cdot}10^{-2}$
96	0.25	10,000	10 10 10	4.169	0.976	$-1.188{\cdot}10^{-4}$	$1.531{\cdot}10^{-2}$

Results from grid search, 1D-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
97	0.25	10,000	20 20 20	3.992	0.977	$-2.326 \cdot 10^{-4}$	$1.499 \cdot 10^{-2}$
98	0.25	10,000	30 30 30	3.97	0.978	$-2.22 \cdot 10^{-4}$	$1.494 \cdot 10^{-2}$
99	0.25	10,000	50 50 50	3.144	0.982	$-4.049 \cdot 10^{-4}$	$1.329 \cdot 10^{-2}$
100	0.25	10,000	100 100 100	2.401	0.986	$-3.017 \cdot 10^{-4}$	$1.162 \cdot 10^{-2}$
101	0.25	10,000	10 10 10 10	4.747	0.973	$-5.565 \cdot 10^{-4}$	$1.633 \cdot 10^{-2}$
102	0.25	10,000	20 20 20 20	3.765	0.979	$-2.373 \cdot 10^{-4}$	$1.455 \cdot 10^{-2}$
103	0.25	10,000	30 30 30 30	3.257	0.982	$-1.103 \cdot 10^{-4}$	$1.354 \cdot 10^{-2}$
104	0.25	10,000	50 50 50 50	2.011	0.989	$8.499 \cdot 10^{-4}$	$1.06 \cdot 10^{-2}$
105	0.25	10,000	100 100 100 100	2.612	0.985	$2.981 \cdot 10^{-4}$	$1.212 \cdot 10^{-2}$
106	0.25	1,000	10 10	7.021	0.96	$1.743 \cdot 10^{-4}$	$1.987 \cdot 10^{-2}$
107	0.25	1,000	20 20	6.11	0.965	$3.359 \cdot 10^{-4}$	$1.854 \cdot 10^{-2}$
108	0.25	1,000	30 30	3.913	0.978	$3.865 \cdot 10^{-4}$	$1.483 \cdot 10^{-2}$
109	0.25	1,000	50 50	3.986	0.977	$4.091 \cdot 10^{-4}$	$1.497 \cdot 10^{-2}$
110	0.25	1,000	100 100	1.612	0.991	$7.264 \cdot 10^{-4}$	$9.495 \cdot 10^{-3}$
111	0.25	1,000	10 10 10	4.022	0.977	$4.793 \cdot 10^{-4}$	$1.504 \cdot 10^{-2}$
112	0.25	1,000	20 20 20	3.435	0.981	$5.253 \cdot 10^{-4}$	$1.389 \cdot 10^{-2}$
113	0.25	1,000	30 30 30	1.738	0.99	$3.84 \cdot 10^{-4}$	$9.88 \cdot 10^{-3}$
114	0.25	1,000	50 50 50	1.882	0.989	$7.192 \cdot 10^{-4}$	$1.027 \cdot 10^{-2}$
115	0.25	1,000	100 100 100	1.28	0.993	$3.431 \cdot 10^{-4}$	$8.481 \cdot 10^{-3}$
116	0.25	1,000	10 10 10 10	3.845	0.978	$3.455 \cdot 10^{-4}$	$1.47 \cdot 10^{-2}$
117	0.25	1,000	20 20 20 20	3.279	0.981	$1.468 \cdot 10^{-4}$	$1.358 \cdot 10^{-2}$
118	0.25	1,000	30 30 30 30	2.73	0.985	$6.315 \cdot 10^{-5}$	$1.239 \cdot 10^{-2}$
119	0.25	1,000	50 50 50 50	1.085	0.994	$-2.725 \cdot 10^{-5}$	$7.813 \cdot 10^{-3}$
120	0.25	1,000	100 100 100 100	1.392	0.992	$-1.854 \cdot 10^{-4}$	$8.847 \cdot 10^{-3}$
121	$1 \cdot 10^{-2}$	20,000	10 10	76.304	0.569	$-7.729 \cdot 10^{-4}$	$6.552 \cdot 10^{-2}$
122	$1 \cdot 10^{-2}$	20,000	20 20	29.007	0.836	$4.811 \cdot 10^{-4}$	$4.04 \cdot 10^{-2}$
123	$1 \cdot 10^{-2}$	20,000	30 30	25.181	0.858	$1.026 \cdot 10^{-4}$	$3.764 \cdot 10^{-2}$
124	$1 \cdot 10^{-2}$	20,000	50 50	15.35	0.913	$3.457 \cdot 10^{-5}$	$2.939 \cdot 10^{-2}$
125	$1 \cdot 10^{-2}$	20,000	100 100	11.485	0.935	$6.013 \cdot 10^{-5}$	$2.542 \cdot 10^{-2}$
126	$1 \cdot 10^{-2}$	20,000	10 10 10	34.306	0.806	$-2.503 \cdot 10^{-4}$	$4.393 \cdot 10^{-2}$
127	$1 \cdot 10^{-2}$	20,000	20 20 20	21.295	0.88	$3.323 \cdot 10^{-4}$	$3.461 \cdot 10^{-2}$
128	$1 \cdot 10^{-2}$	20,000	30 30 30	14.878	0.916	$1.295 \cdot 10^{-4}$	$2.893 \cdot 10^{-2}$
129	$1 \cdot 10^{-2}$	20,000	50 50 50	9.292	0.948	$2.418 \cdot 10^{-4}$	$2.286 \cdot 10^{-2}$

Results from grid search, 1D-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
130	$1 \cdot 10^{-2}$	20,000	100 100 100	6.586	0.963	$2.403 \cdot 10^{-4}$	$1.925 \cdot 10^{-2}$
131	$1 \cdot 10^{-2}$	20,000	10 10 10 10	35.367	0.8	$3.505 \cdot 10^{-7}$	$4.461 \cdot 10^{-2}$
132	$1 \cdot 10^{-2}$	20,000	20 20 20 20	19.266	0.891	$8.63 \cdot 10^{-4}$	$3.291 \cdot 10^{-2}$
133	$1 \cdot 10^{-2}$	20,000	30 30 30 30	10.29	0.942	$3.819 \cdot 10^{-4}$	$2.406 \cdot 10^{-2}$
134	$1 \cdot 10^{-2}$	20,000	50 50 50 50	7.785	0.956	$4.328 \cdot 10^{-4}$	$2.092 \cdot 10^{-2}$
135	$1 \cdot 10^{-2}$	20,000	100 100 100 100	6.363	0.964	$4.63 \cdot 10^{-4}$	$1.891 \cdot 10^{-2}$
136	$1 \cdot 10^{-2}$	5,000	10 10	71.55	0.596	$8.735 \cdot 10^{-5}$	$6.345 \cdot 10^{-2}$
137	$1 \cdot 10^{-2}$	5,000	20 20	22.229	0.874	$1.128 \cdot 10^{-4}$	$3.536 \cdot 10^{-2}$
138	$1 \cdot 10^{-2}$	5,000	30 30	16.29	0.908	$2.253 \cdot 10^{-6}$	$3.027 \cdot 10^{-2}$
139	$1 \cdot 10^{-2}$	5,000	50 50	12.328	0.93	$9.689 \cdot 10^{-5}$	$2.634 \cdot 10^{-2}$
140	$1 \cdot 10^{-2}$	5,000	100 100	9.075	0.949	$1.288 \cdot 10^{-4}$	$2.259 \cdot 10^{-2}$
141	$1 \cdot 10^{-2}$	5,000	10 10 10	69.224	0.609	$1.429 \cdot 10^{-4}$	$6.241 \cdot 10^{-2}$
142	$1 \cdot 10^{-2}$	5,000	20 20 20	15.609	0.912	$-3.223 \cdot 10^{-4}$	$2.963 \cdot 10^{-2}$
143	$1 \cdot 10^{-2}$	5,000	30 30 30	8.166	0.954	$2.621 \cdot 10^{-4}$	$2.143 \cdot 10^{-2}$
144	$1 \cdot 10^{-2}$	5,000	50 50 50	6.072	0.966	$2.442 \cdot 10^{-4}$	$1.848 \cdot 10^{-2}$
145	$1 \cdot 10^{-2}$	5,000	100 100 100	4.86	0.973	$2.234 \cdot 10^{-4}$	$1.653 \cdot 10^{-2}$
146	$1 \cdot 10^{-2}$	5,000	10 10 10 10	19.581	0.889	$9.469 \cdot 10^{-5}$	$3.319 \cdot 10^{-2}$
147	$1 \cdot 10^{-2}$	5,000	20 20 20 20	9.524	0.946	$3.182 \cdot 10^{-4}$	$2.315 \cdot 10^{-2}$
148	$1 \cdot 10^{-2}$	5,000	30 30 30 30	6.618	0.963	$2.83 \cdot 10^{-4}$	$1.929 \cdot 10^{-2}$
149	$1 \cdot 10^{-2}$	5,000	50 50 50 50	4.776	0.973	$2.605 \cdot 10^{-4}$	$1.639 \cdot 10^{-2}$
150	$1 \cdot 10^{-2}$	5,000	100 100 100 100	4.409	0.975	$3.341 \cdot 10^{-4}$	$1.575 \cdot 10^{-2}$
151	$1 \cdot 10^{-2}$	10,000	10 10	73.29	0.586	$1.548 \cdot 10^{-4}$	$6.421 \cdot 10^{-2}$
152	$1 \cdot 10^{-2}$	10,000	20 20	25.028	0.859	$2.404 \cdot 10^{-4}$	$3.752 \cdot 10^{-2}$
153	$1 \cdot 10^{-2}$	10,000	30 30	19.272	0.891	$1.009 \cdot 10^{-4}$	$3.293 \cdot 10^{-2}$
154	$1 \cdot 10^{-2}$	10,000	50 50	14.744	0.917	$8.894 \cdot 10^{-5}$	$2.88 \cdot 10^{-2}$
155	$1 \cdot 10^{-2}$	10,000	100 100	9.911	0.944	$8.617 \cdot 10^{-5}$	$2.361 \cdot 10^{-2}$
156	$1 \cdot 10^{-2}$	10,000	10 10 10	26.239	0.852	$5.625 \cdot 10^{-5}$	$3.842 \cdot 10^{-2}$
157	$1 \cdot 10^{-2}$	10,000	20 20 20	15.545	0.912	$7.998 \cdot 10^{-5}$	$2.957 \cdot 10^{-2}$
158	$1 \cdot 10^{-2}$	10,000	30 30 30	10.062	0.943	$2.367 \cdot 10^{-4}$	$2.379 \cdot 10^{-2}$
159	$1 \cdot 10^{-2}$	10,000	50 50 50	6.958	0.961	$1.985 \cdot 10^{-4}$	$1.978 \cdot 10^{-2}$
160	$1 \cdot 10^{-2}$	10,000	100 100 100	5.774	0.967	$9.607 \cdot 10^{-5}$	$1.802 \cdot 10^{-2}$
161	$1 \cdot 10^{-2}$	10,000	10 10 10 10	19.453	0.89	$4.696 \cdot 10^{-5}$	$3.308 \cdot 10^{-2}$
162	$1 \cdot 10^{-2}$	10,000	20 20 20 20	10.369	0.941	$1.67 \cdot 10^{-4}$	$2.415 \cdot 10^{-2}$

Results from grid search, 1D-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
163	$1 \cdot 10^{-2}$	10,000	30 30 30 30	8.352	0.953	$3.535 \cdot 10^{-4}$	$2.167 \cdot 10^{-2}$
164	$1 \cdot 10^{-2}$	10,000	50 50 50 50	5.846	0.967	$1.743 \cdot 10^{-4}$	$1.813 \cdot 10^{-2}$
165	$1 \cdot 10^{-2}$	10,000	100 100 100 100	5.085	0.971	$-6.691 \cdot 10^{-5}$	$1.691 \cdot 10^{-2}$
166	$1 \cdot 10^{-2}$	1,000	10 10	71.604	0.596	$3.48 \cdot 10^{-4}$	$6.347 \cdot 10^{-2}$
167	$1 \cdot 10^{-2}$	1,000	20 20	16.934	0.904	$1.264 \cdot 10^{-4}$	$3.087 \cdot 10^{-2}$
168	$1 \cdot 10^{-2}$	1,000	30 30	15.562	0.912	$1.549 \cdot 10^{-4}$	$2.959 \cdot 10^{-2}$
169	$1 \cdot 10^{-2}$	1,000	50 50	13.563	0.923	$1.592 \cdot 10^{-4}$	$2.762 \cdot 10^{-2}$
170	$1 \cdot 10^{-2}$	1,000	100 100	8.929	0.95	$1.583 \cdot 10^{-4}$	$2.241 \cdot 10^{-2}$
171	$1 \cdot 10^{-2}$	1,000	10 10 10	68.935	0.611	$2.599 \cdot 10^{-4}$	$6.228 \cdot 10^{-2}$
172	$1 \cdot 10^{-2}$	1,000	20 20 20	10.785	0.939	$1.036 \cdot 10^{-4}$	$2.463 \cdot 10^{-2}$
173	$1 \cdot 10^{-2}$	1,000	30 30 30	9.44	0.947	$1.385 \cdot 10^{-4}$	$2.305 \cdot 10^{-2}$
174	$1 \cdot 10^{-2}$	1,000	50 50 50	5.371	0.97	$1.798 \cdot 10^{-4}$	$1.738 \cdot 10^{-2}$
175	$1 \cdot 10^{-2}$	1,000	100 100 100	4.759	0.973	$2.111 \cdot 10^{-4}$	$1.636 \cdot 10^{-2}$
176	$1 \cdot 10^{-2}$	1,000	10 10 10 10	18.454	0.896	$1.353 \cdot 10^{-4}$	$3.222 \cdot 10^{-2}$
177	$1 \cdot 10^{-2}$	1,000	20 20 20 20	7.197	0.959	$1.925 \cdot 10^{-4}$	$2.012 \cdot 10^{-2}$
178	$1 \cdot 10^{-2}$	1,000	30 30 30 30	5.727	0.968	$1.896 \cdot 10^{-4}$	$1.795 \cdot 10^{-2}$
179	$1 \cdot 10^{-2}$	1,000	50 50 50 50	4.383	0.975	$2.393 \cdot 10^{-4}$	$1.57 \cdot 10^{-2}$
180	$1 \cdot 10^{-2}$	1,000	100 100 100 100	4.215	0.976	$3.818 \cdot 10^{-4}$	$1.539 \cdot 10^{-2}$

LR = learning rate, BS = batch size, r^2 = coefficient of determination, STD = standard deviation [mmHg]

(C) Results from grid search, stenotic regions

Results from grid search, Stenosis-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
1	0.1	1,000	10 10	$1.382{\cdot}10^{7}$	$2.128{\cdot}10^{-2}$	13.526	24.387
2	0.1	1,000	20 20	$9.24{\cdot}10^{6}$	0.346	9.994	20.493
3	0.1	1,000	30 30	$6.2{\cdot}10^{6}$	0.561	7.466	17.119
4	0.1	1,000	50 50	$2.871{\cdot}10^{6}$	0.797	4.167	12.006
5	0.1	1,000	100 100	$6.74{\cdot}10^{5}$	0.952	1.258	6.028
6	0.1	1,000	10 10 10	$1.379{\cdot}10^{7}$	$2.332{\cdot}10^{-2}$	13.58	24.324
7	0.1	1,000	20 20 20	$9.188{\cdot}10^{6}$	0.349	10.08	20.38
8	0.1	1,000	30 30 30	$6.105{\cdot}10^{6}$	0.568	7.574	16.915
9	0.1	1,000	50 50 50	$2.808{\cdot}10^{6}$	0.801	4.51	11.733
10	0.1	1,000	100 100 100	$4.638{\cdot}10^{5}$	0.967	1.2	4.965
11	0.1	1,000	10 10 10 10	$1.38{\cdot}10^{7}$	$2.288{\cdot}10^{-2}$	13.596	24.321
12	0.1	1,000	20 20 20 20	$9.042{\cdot}10^{6}$	0.36	10.006	20.213
13	0.1	1,000	30 30 30 30	$5.909{\cdot}10^{6}$	0.582	7.282	16.715
14	0.1	1,000	50 50 50 50	$2.521{\cdot}10^{6}$	0.822	4.228	11.133
15	0.1	1,000	100 100 100 100	$1.412{\cdot}10^{7}$	$-7.384{\cdot}10^{-6}$	$7.66{\cdot}10^{-2}$	28.188
16	0.1	100	10 10	$5.285{\cdot}10^{6}$	0.626	5.685	16.28
17	0.1	100	20 20	$1.581{\cdot}10^{6}$	0.888	1.967	9.225
18	0.1	100	30 30	$8.542{\cdot}10^{5}$	0.94	0.931	6.87
19	0.1	100	50 50	$4.116{\cdot}10^{5}$	0.971	$1.849{\cdot}10^{-2}$	4.812
20	0.1	100	100 100	$3.179{\cdot}10^{5}$	0.977	0.119	4.227
21	0.1	100	10 10 10	$5.14{\cdot}10^{6}$	0.636	5.826	15.976
22	0.1	100	20 20 20	$1.343{\cdot}10^{6}$	0.905	1.964	8.468
23	0.1	100	30 30 30	$5.921{\cdot}10^{5}$	0.958	0.941	5.694
24	0.1	100	50 50 50	$1.875{\cdot}10^{5}$	0.987	$6.24{\cdot}10^{-2}$	3.247
25	0.1	100	100 100 100	$1.102{\cdot}10^{5}$	0.992	$-8.185{\cdot}10^{-2}$	2.489
26	0.1	100	10 10 10 10	$4.965{\cdot}10^{6}$	0.648	5.786	15.68
27	0.1	100	20 20 20 20	$1.23{\cdot}10^{6}$	0.913	1.909	8.097
28	0.1	100	30 30 30 30	$5.005{\cdot}10^{5}$	0.965	0.856	5.237
29	0.1	100	50 50 50 50	$1.524{\cdot}10^{5}$	0.989	0.157	2.924
30	0.1	100	100 100 100 100	$1.087{\cdot}10^{5}$	0.992	−0.13	2.47
31	0.1	500	10 10	$1.136{\cdot}10^{7}$	0.196	11.47	22.526
32	0.1	500	20 20	$6.343{\cdot}10^{6}$	0.551	7.484	17.346
33	0.1	500	30 30	$3.636{\cdot}10^{6}$	0.743	4.658	13.523
34	0.1	500	50 50	$1.289{\cdot}10^{6}$	0.909	2.138	8.243
35	0.1	500	100 100	$3.822{\cdot}10^{5}$	0.973	0.474	4.613
36	0.1	500	10 10 10	$1.13{\cdot}10^{7}$	0.2	11.568	22.409
37	0.1	500	20 20 20	$6.24{\cdot}10^{6}$	0.558	7.57	17.14

Results from grid search, Stenosis-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
38	0.1	500	30 30 30	$3.448 \cdot 10^{6}$	0.756	4.962	13.015
39	0.1	500	50 50 50	$1.117 \cdot 10^{6}$	0.921	1.981	7.676
40	0.1	500	100 100 100	$1.902 \cdot 10^{5}$	0.987	$-7.243 \cdot 10^{-2}$	3.27
41	0.1	500	10 10 10 10	$1.121 \cdot 10^{7}$	0.206	11.534	22.306
42	0.1	500	20 20 20 20	$6.094 \cdot 10^{6}$	0.568	7.466	16.945
43	0.1	500	30 30 30 30	$3.297 \cdot 10^{6}$	0.767	4.767	12.759
44	0.1	500	50 50 50 50	$1.161 \cdot 10^{6}$	0.918	1.443	7.954
45	0.1	500	100 100 100 100	$1.412 \cdot 10^{7}$	$-1.289 \cdot 10^{-5}$	0.101	28.188
46	0.1	2,000	10 10	$1.574 \cdot 10^{7}$	−0.114	15.098	25.643
47	0.1	2,000	20 20	$1.182 \cdot 10^{7}$	0.163	12.107	22.773
48	0.1	2,000	30 30	$8.901 \cdot 10^{6}$	0.37	9.766	20.135
49	0.1	2,000	50 50	$5.047 \cdot 10^{6}$	0.643	6.504	15.546
50	0.1	2,000	100 100	$1.473 \cdot 10^{6}$	0.896	2.646	8.712
51	0.1	2,000	10 10 10	$1.572 \cdot 10^{7}$	−0.113	15.138	25.602
52	0.1	2,000	20 20 20	$1.179 \cdot 10^{7}$	0.165	12.143	22.709
53	0.1	2,000	30 30 30	$8.846 \cdot 10^{6}$	0.374	9.847	20.018
54	0.1	2,000	50 50 50	$4.949 \cdot 10^{6}$	0.65	6.616	15.318
55	0.1	2,000	100 100 100	$1.197 \cdot 10^{6}$	0.915	2.477	7.825
56	0.1	2,000	10 10 10 10	$1.588 \cdot 10^{7}$	−0.124	15.254	25.704
57	0.1	2,000	20 20 20 20	$1.174 \cdot 10^{7}$	0.168	12.138	22.658
58	0.1	2,000	30 30 30 30	$8.701 \cdot 10^{6}$	0.384	9.823	19.826
59	0.1	2,000	50 50 50 50	$4.766 \cdot 10^{6}$	0.663	6.25	15.135
60	0.1	2,000	100 100 100 100	$1.274 \cdot 10^{6}$	0.91	2.291	8.15
61	0.25	1,000	10 10	$7.176 \cdot 10^{6}$	0.492	7.734	18.546
62	0.25	1,000	20 20	$2.737 \cdot 10^{6}$	0.806	3.586	11.88
63	0.25	1,000	30 30	$1.141 \cdot 10^{6}$	0.919	1.693	7.832
64	0.25	1,000	50 50	$3.951 \cdot 10^{5}$	0.972	0.429	4.695
65	0.25	1,000	100 100	$3.17 \cdot 10^{5}$	0.978	0.162	4.22
66	0.25	1,000	10 10 10	$7.081 \cdot 10^{6}$	0.499	7.889	18.335
67	0.25	1,000	20 20 20	$2.558 \cdot 10^{6}$	0.819	3.698	11.413
68	0.25	1,000	30 30 30	$9.965 \cdot 10^{5}$	0.929	1.656	7.302
69	0.25	1,000	50 50 50	$3.024 \cdot 10^{5}$	0.979	0.591	4.082
70	0.25	1,000	100 100 100	$1.412 \cdot 10^{7}$	$-6.753 \cdot 10^{-6}$	$7.325 \cdot 10^{-2}$	28.188
71	0.25	1,000	10 10 10 10	$6.95 \cdot 10^{6}$	0.508	7.709	18.21
72	0.25	1,000	20 20 20 20	$2.322 \cdot 10^{6}$	0.836	3.491	10.884
73	0.25	1,000	30 30 30 30	$1.412 \cdot 10^{7}$	$-7.43 \cdot 10^{-6}$	$7.684 \cdot 10^{-2}$	28.188
74	0.25	1,000	50 50 50 50	$1.412 \cdot 10^{7}$	$-6.704 \cdot 10^{-6}$	$7.298 \cdot 10^{-2}$	28.188
75	0.25	1,000	100 100 100 100	$1.412 \cdot 10^{7}$	$-6.831 \cdot 10^{-6}$	$7.367 \cdot 10^{-2}$	28.188
76	0.25	100	10 10	$1.128 \cdot 10^{6}$	0.92	$-4.279 \cdot 10^{-2}$	7.967
77	0.25	100	20 20	$4.614 \cdot 10^{5}$	0.967	−0.247	5.089

Results from grid search, Stenosis-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
78	0.25	100	30 30	$3.106 \cdot 10^5$	0.978	−0.249	4.173
79	0.25	100	50 50	$2.513 \cdot 10^5$	0.982	$-1.274 \cdot 10^{-2}$	3.76
80	0.25	100	100 100	$2.096 \cdot 10^5$	0.985	$4.93 \cdot 10^{-2}$	3.434
81	0.25	100	10 10 10	$8.107 \cdot 10^5$	0.943	$-5.518 \cdot 10^{-2}$	6.753
82	0.25	100	20 20 20	$2.397 \cdot 10^5$	0.983	−0.384	3.652
83	0.25	100	30 30 30	$1.459 \cdot 10^5$	0.99	−0.246	2.855
84	0.25	100	50 50 50	$1.072 \cdot 10^5$	0.992	−0.121	2.453
85	0.25	100	100 100 100	$2.034 \cdot 10^5$	0.986	$-7.657 \cdot 10^{-2}$	3.382
86	0.25	100	10 10 10 10	$7.565 \cdot 10^5$	0.946	−0.137	6.522
87	0.25	100	20 20 20 20	$1.97 \cdot 10^5$	0.986	−0.359	3.31
88	0.25	100	30 30 30 30	$1.412 \cdot 10^7$	$-1.25 \cdot 10^{-5}$	$9.966 \cdot 10^{-2}$	28.188
89	0.25	100	50 50 50 50	$1.412 \cdot 10^7$	$-1.398 \cdot 10^{-5}$	0.105	28.188
90	0.25	100	100 100 100 100	$2.823 \cdot 10^5$	0.98	$-4.907 \cdot 10^{-2}$	3.985
91	0.25	500	10 10	$4.484 \cdot 10^6$	0.682	4.86	15.121
92	0.25	500	20 20	$1.235 \cdot 10^6$	0.913	1.42	8.214
93	0.25	500	30 30	$5.071 \cdot 10^5$	0.964	0.334	5.331
94	0.25	500	50 50	$3.059 \cdot 10^5$	0.978	$7.151 \cdot 10^{-3}$	4.148
95	0.25	500	100 100	$2.563 \cdot 10^5$	0.982	$-6.132 \cdot 10^{-3}$	3.797
96	0.25	500	10 10 10	$4.3 \cdot 10^6$	0.696	5.003	14.727
97	0.25	500	20 20 20	$1.049 \cdot 10^6$	0.926	1.3	7.57
98	0.25	500	30 30 30	$3.584 \cdot 10^5$	0.975	$-7.269 \cdot 10^{-2}$	4.49
99	0.25	500	50 50 50	$1.632 \cdot 10^5$	0.988	−0.232	3.022
100	0.25	500	100 100 100	$2.79 \cdot 10^5$	0.98	−0.241	3.954
101	0.25	500	10 10 10 10	$4.171 \cdot 10^6$	0.705	4.808	14.544
102	0.25	500	20 20 20 20	$1.056 \cdot 10^6$	0.925	0.793	7.668
103	0.25	500	30 30 30 30	$2.496 \cdot 10^5$	0.982	0.146	3.744
104	0.25	500	50 50 50 50	$1.412 \cdot 10^7$	$-1.29 \cdot 10^{-5}$	0.101	28.188
105	0.25	500	100 100 100 100	$1.412 \cdot 10^7$	$-1.303 \cdot 10^{-5}$	0.102	28.188
106	0.25	2,000	10 10	$9.85 \cdot 10^6$	0.303	10.219	21.208
107	0.25	2,000	20 20	$4.861 \cdot 10^6$	0.656	6.034	15.397
108	0.25	2,000	30 30	$2.419 \cdot 10^6$	0.829	3.553	11.111
109	0.25	2,000	50 50	$7.565 \cdot 10^5$	0.946	1.39	6.374
110	0.25	2,000	100 100	$3.117 \cdot 10^5$	0.978	0.388	4.17
111	0.25	2,000	10 10 10	$9.804 \cdot 10^6$	0.306	10.224	21.143
112	0.25	2,000	20 20 20	$4.76 \cdot 10^6$	0.663	6.156	15.163
113	0.25	2,000	30 30 30	$2.325 \cdot 10^6$	0.835	3.691	10.824
114	0.25	2,000	50 50 50	$6.061 \cdot 10^5$	0.957	1.353	5.681
115	0.25	2,000	100 100 100	$2.212 \cdot 10^5$	0.984	$7.791 \cdot 10^{-2}$	3.527
116	0.25	2,000	10 10 10 10	$9.679 \cdot 10^6$	0.315	10.147	21.014
117	0.25	2,000	20 20 20 20	$4.508 \cdot 10^6$	0.681	5.87	14.804

Results from grid search, Stenosis-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
118	0.25	2,000	30 30 30 30	$1.412{\cdot}10^{7}$	$-8.398{\cdot}10^{-6}$	$8.169{\cdot}10^{-2}$	28.188
119	0.25	2,000	50 50 50 50	$1.412{\cdot}10^{7}$	$-6.811{\cdot}10^{-6}$	$7.356{\cdot}10^{-2}$	28.188
120	0.25	2,000	100 100 100 100	$1.412{\cdot}10^{7}$	$-6.807{\cdot}10^{-6}$	$7.355{\cdot}10^{-2}$	28.188
121	$1 \cdot 10^{-2}$	1,000	10 10	$2.076{\cdot}10^{7}$	-0.47	19.477	28.081
122	$1 \cdot 10^{-2}$	1,000	20 20	$1.99{\cdot}10^{7}$	-0.409	18.553	27.841
123	$1 \cdot 10^{-2}$	1,000	30 30	$1.908{\cdot}10^{7}$	-0.351	17.757	27.539
124	$1 \cdot 10^{-2}$	1,000	50 50	$1.765{\cdot}10^{7}$	-0.25	16.504	26.844
125	$1 \cdot 10^{-2}$	1,000	100 100	$1.445{\cdot}10^{7}$	$-2.321{\cdot}10^{-2}$	14.149	24.755
126	$1 \cdot 10^{-2}$	1,000	10 10 10	$2.076{\cdot}10^{7}$	-0.47	19.327	28.188
127	$1 \cdot 10^{-2}$	1,000	20 20 20	$1.992{\cdot}10^{7}$	-0.411	18.061	28.188
128	$1 \cdot 10^{-2}$	1,000	30 30 30	$1.916{\cdot}10^{7}$	-0.356	16.827	28.188
129	$1 \cdot 10^{-2}$	1,000	50 50 50	$1.774{\cdot}10^{7}$	-0.256	16.746	26.79
130	$1 \cdot 10^{-2}$	1,000	100 100 100	$1.487{\cdot}10^{7}$	$-5.277{\cdot}10^{-2}$	14.19	25.202
131	$1 \cdot 10^{-2}$	1,000	10 10 10 10	$2.076{\cdot}10^{7}$	-0.47	19.328	28.188
132	$1 \cdot 10^{-2}$	1,000	20 20 20 20	$1.992{\cdot}10^{7}$	-0.411	18.061	28.188
133	$1 \cdot 10^{-2}$	1,000	30 30 30 30	$1.915{\cdot}10^{7}$	-0.356	16.824	28.188
134	$1 \cdot 10^{-2}$	1,000	50 50 50 50	$1.788{\cdot}10^{7}$	-0.266	14.534	28.188
135	$1 \cdot 10^{-2}$	1,000	100 100 100 100	$1.618{\cdot}10^{7}$	-0.145	10.752	28.188
136	$1 \cdot 10^{-2}$	100	10 10	$1.879{\cdot}10^{7}$	-0.331	17.475	27.422
137	$1 \cdot 10^{-2}$	100	20 20	$1.651{\cdot}10^{7}$	-0.169	15.415	26.29
138	$1 \cdot 10^{-2}$	100	30 30	$1.45{\cdot}10^{7}$	$-2.705{\cdot}10^{-2}$	13.672	25.083
139	$1 \cdot 10^{-2}$	100	50 50	$1.105{\cdot}10^{7}$	0.218	11.357	22.197
140	$1 \cdot 10^{-2}$	100	100 100	$6.153{\cdot}10^{6}$	0.564	7.34	17.097
141	$1 \cdot 10^{-2}$	100	10 10 10	$1.879{\cdot}10^{7}$	-0.33	17.624	27.323
142	$1 \cdot 10^{-2}$	100	20 20 20	$1.645{\cdot}10^{7}$	-0.165	15.737	26.036
143	$1 \cdot 10^{-2}$	100	30 30 30	$1.439{\cdot}10^{7}$	$-1.895{\cdot}10^{-2}$	14.142	24.69
144	$1 \cdot 10^{-2}$	100	50 50 50	$1.098{\cdot}10^{7}$	0.222	11.524	22.024
145	$1 \cdot 10^{-2}$	100	100 100 100	$5.491{\cdot}10^{6}$	0.611	6.976	16.132
146	$1 \cdot 10^{-2}$	100	10 10 10 10	$1.888{\cdot}10^{7}$	-0.337	16.356	28.188
147	$1 \cdot 10^{-2}$	100	20 20 20 20	$1.693{\cdot}10^{7}$	-0.199	12.572	28.188
148	$1 \cdot 10^{-2}$	100	30 30 30 30	$1.468{\cdot}10^{7}$	$-3.953{\cdot}10^{-2}$	14.242	24.963
149	$1 \cdot 10^{-2}$	100	50 50 50 50	$1.159{\cdot}10^{7}$	0.179	12.034	22.524
150	$1 \cdot 10^{-2}$	100	100 100 100 100	$7.055{\cdot}10^{6}$	0.5	8.512	18.013
151	$1 \cdot 10^{-2}$	500	10 10	$2.033{\cdot}10^{7}$	-0.439	19.005	27.974
152	$1 \cdot 10^{-2}$	500	20 20	$1.914{\cdot}10^{7}$	-0.355	17.808	27.562
153	$1 \cdot 10^{-2}$	500	30 30	$1.803{\cdot}10^{7}$	-0.277	16.797	27.063
154	$1 \cdot 10^{-2}$	500	50 50	$1.599{\cdot}10^{7}$	-0.132	15.319	25.789
155	$1 \cdot 10^{-2}$	500	100 100	$1.208{\cdot}10^{7}$	0.145	12.292	22.99
156	$1 \cdot 10^{-2}$	500	10 10 10	$2.034{\cdot}10^{7}$	-0.44	18.705	28.188
157	$1 \cdot 10^{-2}$	500	20 20 20	$1.921{\cdot}10^{7}$	-0.36	16.911	28.188

Results from grid search, Stenosis-ML model.

Case	LR	BS	Network	Cost	r^2	Mean Diff. [mmHg]	STD [mmHg]
158	$1 \cdot 10^{-2}$	500	30 30 30	$1.804 \cdot 10^{7}$	-0.277	17.009	26.939
159	$1 \cdot 10^{-2}$	500	50 50 50	$1.606 \cdot 10^{7}$	-0.137	15.43	25.799
160	$1 \cdot 10^{-2}$	500	100 100 100	$1.229 \cdot 10^{7}$	0.13	12.379	23.194
161	$1 \cdot 10^{-2}$	500	10 10 10 10	$2.034 \cdot 10^{7}$	-0.44	18.703	28.188
162	$1 \cdot 10^{-2}$	500	20 20 20 20	$1.92 \cdot 10^{7}$	-0.36	16.905	28.188
163	$1 \cdot 10^{-2}$	500	30 30 30 30	$1.822 \cdot 10^{7}$	-0.29	15.175	28.188
164	$1 \cdot 10^{-2}$	500	50 50 50 50	$1.667 \cdot 10^{7}$	-0.18	11.974	28.188
165	$1 \cdot 10^{-2}$	500	100 100 100 100	$1.523 \cdot 10^{7}$	$-7.837 \cdot 10^{-2}$	7.891	28.188
166	$1 \cdot 10^{-2}$	2,000	10 10	$2.104 \cdot 10^{7}$	-0.49	19.802	28.138
167	$1 \cdot 10^{-2}$	2,000	20 20	$2.041 \cdot 10^{7}$	-0.445	19.094	27.998
168	$1 \cdot 10^{-2}$	2,000	30 30	$1.982 \cdot 10^{7}$	-0.403	18.472	27.814
169	$1 \cdot 10^{-2}$	2,000	50 50	$1.868 \cdot 10^{7}$	-0.323	17.386	27.365
170	$1 \cdot 10^{-2}$	2,000	100 100	$1.616 \cdot 10^{7}$	-0.144	15.45	25.893
171	$1 \cdot 10^{-2}$	2,000	10 10 10	$2.105 \cdot 10^{7}$	-0.49	19.742	28.188
172	$1 \cdot 10^{-2}$	2,000	20 20 20	$2.043 \cdot 10^{7}$	-0.446	18.833	28.188
173	$1 \cdot 10^{-2}$	2,000	30 30 30	$1.984 \cdot 10^{7}$	-0.405	17.936	28.188
174	$1 \cdot 10^{-2}$	2,000	50 50 50	$1.882 \cdot 10^{7}$	-0.332	16.252	28.188
175	$1 \cdot 10^{-2}$	2,000	100 100 100	$1.724 \cdot 10^{7}$	-0.221	13.244	28.188
176	$1 \cdot 10^{-2}$	2,000	10 10 10 10	$2.105 \cdot 10^{7}$	-0.49	19.737	28.188
177	$1 \cdot 10^{-2}$	2,000	20 20 20 20	$2.042 \cdot 10^{7}$	-0.446	18.829	28.188
178	$1 \cdot 10^{-2}$	2,000	30 30 30 30	$1.985 \cdot 10^{7}$	-0.405	17.944	28.188
179	$1 \cdot 10^{-2}$	2,000	50 50 50 50	$1.885 \cdot 10^{7}$	-0.335	16.303	28.188
180	$1 \cdot 10^{-2}$	2,000	100 100 100 100	$1.728 \cdot 10^{7}$	-0.224	13.33	28.188

LR = learning rate, BS = batch size, r^2 = coefficient of determination, STD = standard deviation

(D) DNNRegressor.py

```
from __future__ import print_function

import os

import argparse
import timeit
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.contrib import learn
from sklearn import cross_validation
from sklearn import preprocessing
from sklearn import metrics
from sklearn.externals import joblib

# Adapted from: http://www.science.smith.edu/dftwiki/index.php/Tutorial:_Playing_with_the_Boston_Housing_Data

parser = argparse.ArgumentParser(description='DNNRegressor')

parser.add_argument('--learningRate', type=float, nargs=1,help='learning rate',default='0.25')
parser.add_argument('--testGroupSize', type=float, nargs=1,help='testGroupSize',default='0.2')
parser.add_argument('--max_epochs', type=int, nargs=1,help='max_epochs',default='20000')
parser.add_argument('--tolerance', type=float, nargs=1,help='tolerance',default='1e-3')
parser.add_argument('--batch_size', type=int, nargs=1,help='batch_size',default='800')
parser.add_argument('--layers', nargs='+', type=int, help='<Required> Set flag', required=True)
parser.add_argument('--case', type=int, nargs=1,help='case',default='0')
parser.add_argument('--eval', type=int, nargs=1,help='evaluate only',default='0')
parser.add_argument('--dataName', type=str, nargs=1,help='evaluate only',default='data.dat')
parser.add_argument('--dropoutActive', type=int, nargs=1,help='evaluate only',default='0')
parser.add_argument('--dropoutVal', type=float, nargs=1,help='# Dropout, probability to drop a unit',default=0.)
parser.add_argument('--featureCols', nargs='+', type=int, help='<Required> Set flag', required=True)
parser.add_argument('--labelCol', type=int, nargs=1,help='evaluate only',default='-1')
parser.add_argument('--scalerType', type=int, nargs=1,help='evaluate only',default='0')
parser.add_argument('--caseTrain', type=int, nargs=1,help='evaluate only',default=None)
args = parser.parse_args()

#==================================================
# Get input arguments
learningRate = args.learningRate[0]
testGroupSize = args.testGroupSize[0]
max_epochs  = args.max_epochs[0]      # total number of training sessions
tolerance   = args.tolerance[0]       # we stop when diff in costs less than that
batch_size  = args.batch_size[0]         # we batch the data in groups of this size
layers = args.layers
layers.append(1)
case = args.case[0]
eval = args.eval[0]
dropoutActive = args.dropoutActive[0]
dataName = args.dataName[0]
dropoutVal = args.dropoutVal
featureCols = args.featureCols
labelCol = args.labelCol
scalerType = args.scalerType[0]
caseTrain = args.caseTrain[0]

folder = 'res_'+str(case)
try:
    os.stat(folder)
except:
    os.mkdir(folder)

data = np.genfromtxt(dataName,skip_header=1)
ncol = data.shape[1]
x, y = data[:,featureCols].copy(), data[:,labelCol].copy()
y.resize( y.size, 1 ) #make y = [[x], [x], [x], ... ]

train_x, test_x, train_y, test_y = cross_validation.train_test_split(
                                    x, y, test_size=testGroupSize, random_state=42)

if eval==1:
    test_x,test_y=x,y

```

```
74  print( "Dimension of test_x = ", test_x.shape )
75  print( "Dimension of test_y = ", test_y.shape )
76
77  print( "Dimension of train_x = ", train_x.shape )
78  print( "Dimension of train_y = ", train_y.shape )
79
80  train_x_ori = train_x.copy()
81  test_x_ori = test_x.copy()
82
83  scaler_filename=folder+'/scaler'+str(case)
84  if eval==0:
85      if scalerType==0:
86          scaler = preprocessing.StandardScaler()
87      elif scalerType==1:
88          scaler = preprocessing.QuantileTransformer(output_distribution='uniform')
89      elif scalerType==2:
90          scaler = preprocessing.RobustScaler(quantile_range=(25, 75))
91      train_x = scaler.fit_transform( train_x )
92      joblib.dump(scaler,scaler_filename)
93  else:
94      if caseTrain!=None:
95          folderTrain = 'res_'+str(caseTrain)
96          scaler_filename=folderTrain+'/scaler'+str(caseTrain)
97
98      scaler=joblib.load(scaler_filename)
99
100 test_x  = scaler.transform( test_x )
101
102 numFeatures =  train_x.shape[1]
103 layers.insert(0,numFeatures)
104
105 #print(scaler.mean_,scaler.scale_)
106
107 print( "number of features = ", numFeatures )
108
109 with tf.name_scope("IO"):
110     inputs = tf.placeholder(tf.float32, [None, numFeatures], name="X")
111     outputs = tf.placeholder(tf.float32, [None, 1], name="Yhat")
112
113 with tf.name_scope("LAYER"):
114     # network architecture
115     #Layers = [numFeatures, 52, 104, 52, 52, 52, 1]
116     Layers = layers #[numFeatures, 10, 10, 1]
117     h = []
118     b = []
119     for i in range( 1, len( Layers ) ):
120         h.append( tf.Variable(tf.random_normal([Layers[i-1], Layers[i]], 0, 0.1, dtype=tf.float32), name="h%d" % i
                ) )
121         b.append( tf.Variable(tf.random_normal([Layers[i]], 0, 0.1, dtype=tf.float32 ), name="b%d" % i ) )
122
123     dropout = dropoutVal                            # Dropout, probability to keep units
124     keep_prob = tf.placeholder(tf.float32)   # dropout (keep probability)
125
126
127 def model( inputs, h, b ):
128     lastY = inputs
129     for i, (hi, bi) in enumerate( zip( h, b ) ):
130         y =  tf.add( tf.matmul( lastY, h[i]), b[i] )
131
132         if i==len(h)-1:
133             return y
134
135         lastY =  tf.nn.sigmoid( y )
136         #lastY =  tf.nn.relu( y )
137         if dropoutActive:
138             lastY =  tf.nn.dropout( lastY, dropout )
139
140 with tf.name_scope("train"):
141
142     yout = model( inputs, h, b )
143
144     cost_op = tf.reduce_mean( tf.pow( yout - outputs, 2 ))
145     #train_op = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost_op)
146     #train_op = tf.train.AdamOptimizer( learning_rate=learning_rate ).minimize( cost_op )
147     train_op = tf.train.AdagradOptimizer( learning_rate=learningRate ).minimize( cost_op )
148
```

```
# define variables/constants that control the training
epoch       = 0          # counter for number of rounds training network
last_cost   = 0          # keep track of last cost to measure difference

num_samples = train_y.shape[0]                  # number of samples in training set
if batch_size==-1:
    batch_size=num_samples
num_batches = int( num_samples / batch_size )   # compute number of batches, given
                                                # batch size


print( "batch size = ", batch_size )
print( "test length= ", num_samples )
print( "number batches = ", num_batches )
print( "--- Beginning Training ---" )

sess = tf.Session() # Create TensorFlow session
start = timeit.timeit()
if eval==0:

    with sess.as_default():

        # initialize the variables
        init = tf.initialize_all_variables()
        sess.run(init)

        # start training until we stop, either because we've reached the max
        # number of epochs, or successive errors are close enough to each other
        # (less than tolerance)

        costs = []
        epochs= []
        while True:
            # Do the training
            cost = 0
            for n in range(  num_batches ):
                batch_x = train_x[ n*batch_size : (n+1)*batch_size ]
                batch_y = train_y[ n*batch_size : (n+1)*batch_size ]
                sess.run( train_op, feed_dict={inputs: batch_x, outputs: batch_y} )
                c = sess.run(cost_op, feed_dict={inputs: batch_x, outputs: batch_y} )
                cost += c
            cost /= num_batches

            costs.append( cost )
            epochs.append( epoch )

            # Update the user every 1000 epochs
            if epoch % 1000==0:
                print( "Epoch: %d - Error diff: %1.8f" %(epoch, cost) )

                # time to stop?
                if epoch > max_epochs  or abs(last_cost - cost) < tolerance:
                    print( "--- STOPPING ---" )
                    break
                last_cost = cost

            epoch += 1

        # we're done...
        # print some statistics...

        print( "Test Cost =", sess.run(cost_op, feed_dict={inputs: test_x, outputs: test_y}) )

        # compute the predicted output for test_x
        pred_y = sess.run( yout, feed_dict={inputs: test_x, outputs: test_y} )


    # Add ops to save and restore all the variables.
    saver = tf.train.Saver()

    # Save the variables to disk.
    save_path = saver.save(sess, folder+"/trainedDNN_%i.ckpt" % case)

else:
    saver = tf.train.Saver()
    if caseTrain!=None:
```

```
225
226         saver.restore(sess, folderTrain+"/trainedDNN_%i.ckpt" % caseTrain)
227     else:
228         saver.restore(sess, folder+"/trainedDNN_%i.ckpt" % case)
229     pred_y = sess.run( yout, feed_dict={inputs: test_x, outputs: test_y} )
230
231 end = timeit.timeit()
232 tt = end-start
233
234
235 np.savetxt(folder+'/pred_y',pred_y)
236 np.savetxt(folder+'/test_y',test_y)
237 np.savetxt(folder+'/test_x',test_x)
238 np.savetxt(folder+'/test_x_ori',test_x_ori)
239 np.savetxt(folder+'/train_x_ori',train_x_ori)
240
241
242
243 f = open(folder+'/stats','w')
244 r2 =  metrics.r2_score(test_y, pred_y)
245 f.write( "mean squared error = %.15e\n" % (metrics.mean_squared_error(test_y, pred_y)))
246 f.write( "r2 score (coef determination) = %.15e\n" % (metrics.r2_score(test_y, pred_y)))
247
248 #fig = plt.figure()
249 #xmin = min(test_y)
250 #xmax = max(test_y)# + 5
251 #plt.xlim(xmin, xmax)
252
253 #x = np.linspace( xmin, xmax )
254 #plt.scatter( test_y, pred_y )
255 #plt.plot( x, x )
256
257
258 #plt.xlabel( "Test y" )
259 #plt.ylabel( "predicted y" )
260 #plt.title( "Prediction vs. Actual Y (r2=%1.4f)" % r2 )
261 #plt.save( "images/sigmoid_adagrad_52_39_26_13_1.png")
262 #plt.show()
263 #fig.savefig('PredVsReal_%i.png' % case, bbox_inches='tight')
264
265 #fig = plt.figure()
266 avg = np.average(pred_y-test_y)
267 #plt.axhline(y=avg,color='k')
268 std = np.std(pred_y-test_y)
269 f.write('Mean diff:  %.15e; Std. Dev.: %.15e; Nr.: %i' % (avg,std,test_y.shape[0]))
270 f.close()
271 #tit = 'Mean diff:  %.3f; Std. Dev.: %.3f; Nr.: %i' % (avg,std,test_y.shape[0])
272 #plt.scatter( test_y, - test_y + pred_y )
273 #plt.axhline(0, color='black')
274 #plt.xlabel( "Test y" )
275 #plt.ylabel( "Test y - Predicted Y" )
276 #plt.title( tit )
277 #plt.show()
278 #fig.savefig('Residuals_%i.png' % case, bbox_inches='tight')
279 if eval==0:
280     f = open(folder+'/costVsEpochs','w')
281     for i in range(len(epochs)):
282         f.write("%.15e %.15e\n" % (epochs[i],costs[i]))
283     f.close()
284 #if eval==0:
285 #    fig = plt.figure()
286 #    plt.semilogy( epochs, costs )
287 #    plt.xlabel( "Epochs" )
288 #    plt.ylabel( "Cost" )
289 #    plt.title( "Cost vs. Epochs")
290     #plt.show()
291 #    fig.savefig('CostVsEpochs_%i.png' % case, bbox_inches='tight')
292
293 print("DONE")
```