Master's Thesis Academic Year 2020-2021

PREDICTION OF PATHOLOGIES OF HUMAN NOSE BY MACHINE LEARNING WITH FLOW PHYSICS

Kazuto HASEGAWA
(Student ID: 936079)

Academic supervisor: Prof. Maurizio QUADRIO

POLITECNICO DI MILANO

Master of Science in AERONAUTICAL ENGINEERING

Abstract

Computational fluid dynamics is expected to play a crucial role to diagnose a condition of human nose. The detailed information and solutions provided by numerical simulations enable us to not only analyze flow characteristics but also visualize flow fields in an understandable manner. However, it is also true that we often have to include experted knowledge to achieve a precise assessment since there are considerable uncertainties caused by the difficulty of diagnosis for human nose. We here consider the use of data-driven frameworks for diagnosing nasal pathologies. Geometries and flow fields of 200 different noses, half of them exhibit some degree of turbinate hypertrophy, are utilized to predict their pathological parameters. First, the geometrical characteristics of the noses are extracted as a functional map between a reference healthy nose and other noses. The functional maps are then used to train neural networks and to predict pathological parameters. Three different machine learning models with different inputs and configurations are trained to examine the predictability of the methods for the pathologies. In addition to the prediction based on the geometric information, we also consider the utilization of flow measurements such as pressure and wall shear stress. The flow fields are considered as a function defined on nose which can be expressed by linear combinations of eigenfunctions of Laplace-Beltrami operator. The present results show reasonable agreements with the reference pathologies. We also find that the prediction performance can significantly be improved by including flow information as the input of machine learning models.

Acknowledgments

I really appreciate the grateful support from my supervisor, Professor Maurizio Quadrio. He kindly welcomed me and gave me the opportunity to work on this project. Even after I was forced to return to Japan due to COVID-19, he supported my remote study.

I would also like to express my gratitude to Professor Koji Fukagata for giving me the opportunity to study at Politecnico di Milano.

I am really grateful to the kind support from Mr. Andrea Schillaci. He gave me a lot of support. Despite the time difference, he was always available for meetings and fruitful discussions.

At last, I would like to thank my family for allowing me to study abroad at Politecnico di Milano and for their constant support during my stay.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 OpenNOSE project

The objective of OpenNOSE project is to create a reliable, patient-specific and open source tool to help diagnose and study nasal conditions. Most of the previous works of this project have focused on reproducing and analysing airflows inside the human nasal cavity by using computational fluid dynamics (CFD) (Saibene et al., 2020; Covello et al., 2018). As a results, the nasal airflow close to an practice one is reasonably simulated with some limitations (Quadrio et al., 2014). The next step of this project is then the utilization of CFD solutions for diagnosing the nasal pathologies. One of the works related to this step is performed by Schillaci et al. (2021). Schillaci et al. (2021) inferred the geometries of airfoils and nasal structures, i.e., pathological parameters, from flow features through simple fully-connected neural network models. The present study aims to predict the nasal pathologies from nose geometries and fluid dynamics features assuming that the CFD solutions provide a reasonable solution to improve the predictions, as a part of OpenNOSE project. The detailed background and objective of this study are offered in what follows.

1.2 Background and objective

CFD, which simulates flows by numerically solving the governing equations for the motion of fluids, has exhibited a great capability in assessing the detailed information in a system of a wide range of science and engineering thanks to the development of compu-

tational power. The medical field is, of course, not an exception for the application of them (Saibene et al., 2020; Covello et al., 2018). Zhao et al. (2004) developed a method to quickly convert nasal CT scans from an individual patient into an anatomically accurate 3-D numerical nasal model using CFD. The effects of nasal obstruction with swelling of inferior turbinates on the aerodynamic flow pattern are investigated with turbulence models (Chen et al., 2010). These CFD results are expected to be useful to diagnose human systems in an interpretable manner with the flow visualization. However, even though with the development of CFD in the medical field, diagnosing the human system is very challenging because the important information for the diagnosis cannot be provided directly by CFD solutions themselves, which implies that the incorporation of a priori knowledge by experts should be required for precise assessments. Particularly, the diagnosis of nasal breathing difficulties (NBD), an pathological condition of human nose that often requires corrective surgery, is known as a longstanding and challenging matter. Some previous studies have reported that the failure rate of surgery is up to 50% (Sundh and Sunnergren, 2015; Illum, 1997). Although the CFD solutions of the nasal airflow is considered to be effective for the diagnosis of NBD, a new framework, that can help the reasonable diagnosis while incorporating the accumulated knowledge as a database, has been eagerly desired.

Of particular interest here is the utilization of machine learning techniques which have recently been recognized as a powerful analysis tool in assessing fluid flows. (LeCun et al., 2015; Kutz, 2017; Brunton et al., 2020; Taira et al., 2020; Brenner et al., 2019) The reconstruction and estimation of flow data and characteristics can be regarded as one of main uses in machine learning for fluid mechanics (Fukami, Fukagata and Taira, 2020). Carrillo et al. (2016) used a neural network to estimate the Reynolds number from the information obtained by detector located in a wake flow behind two-dimensional cylinder. Fukami et al. (Fukami, Fukagata and Taira, 2019, 2021) performed super-resolution reconstruction to recover a high-resolution flow field from a low-resolution counterpart with convolutional neural network (CNN) while considering unsteady laminar wakes and

wall-bounded turbulence. A global field reconstruction from limited sparse measurements has also been performed with neural networks (Erichson et al., 2020; Fukami, Maulik, Ramachandra, Fukagata and Taira, 2021). The application of machine learning for data estimation is not only for numerical side but also for experiments. Cai et al. (2019) used CNNs in obtaining the flow field from particle images. The aforementioned references indicate the capability of machine learning to extract hidden relations and physics from big data associated with fluid flows.

The objective of the present study is to predict the nasal pathologies from geometrical information and flow fields simulated by CFD. The CFD data simulated by Romani (2017) are used as nasal airflows. The simplified nasal cavity model is built using Computer-aided Design (CAD) with eight anatomical variations, five of which behave as harmless anatomical differences between healthy individuals, while the remaining three have a pathological impact on the airflow. To compare the results, machine learning models are constructed with two conditions in order to infer the pathological parameters. First, the pathologies are predicted from the geometries by training three machine learning models with different structures. To extract the features from 3D shape of nose, a functional map, proposed by Ovsjanikov et al. (2012), are utilized. We also consider the use of flow field information such as pressure and wall shear stress to predict the pathologies. The ability of prediction is evaluated by unseen data for the machine learning models.

Chapter 2

Modeling and Numerical Simulation of Nasal Cavity

The methods for numerical simulation of nasal airflows are presented in this chapter. In this study, the flow fields simulated by Romani (2017) are used. For further details of numerical simulations, readers refer to Romani (2017).

2.1 Nasal cavity and turbinate hypertrophies

In this section, anatomical parts of human nose and pathologies investigated in this study, i.e., turbinate hypertrophies, are explained. The illustrations of nasal cavity are presented in figures 2.1 and 2.2. The human nose is the olfactory organ as well as the main pathway for air to enter and exit the lungs. The nose warms, humidifies, and cleans the air going to the lungs. The upper part of the external nose is supported by bone, while the lower part being supported by cartilage. The inner space of the nose is called nasal cavity, which is divided into two passages by the nasal septum. The nasal septum is made of bone and cartilage and extends from the nostrils to the back of the nose. Bones called nasal turbinate protrudes into the nasal cavity. These turbinates, shown in figure 2.2, greatly increase the surface area of the nasal cavity, thereby allowing for more effective exchange of heat and moisture. In the upper part of the nasal cavity, there are cells called olfactory receptors. They are specialized nerve cells equipped with hairs. The hairs of each cell respond to various chemicals and, when stimulated, produce nerve impulses that are sent to the nerve cells of the olfactory bulb, located in the cranium just above the nose. The nerve impulses

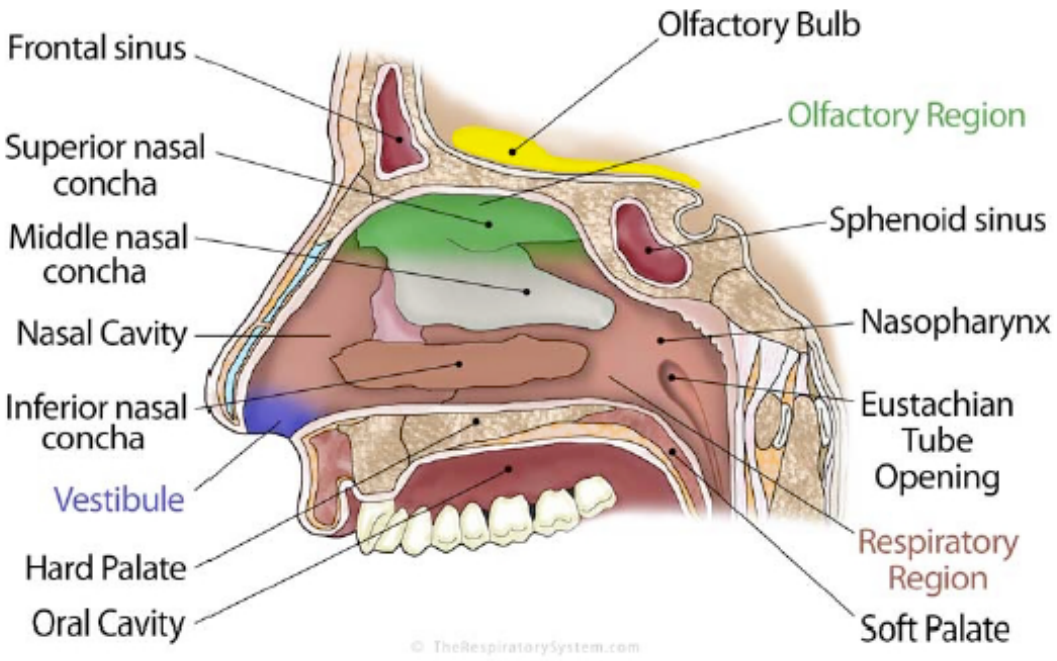

Figure 2.1: Schematic of nasal cavity (1) *The Respiratory System.* (n.d.).

are transmitted directly from the olfactory bulb to the brain by the olfactory nerve, where they are recognized as odors.

Turbinate hypertrophy is a type of chronic rhinitis in which the mucous membrane of the nasal cavity, especially the inferior turbinate mucosa, is chronically swelled, resulting in symptoms such as nasal congestion, nasal discharge, and decreased sense of smell. It is caused by chronic irritation from allergic rhinitis, drug-induced rhinitis caused by long-term use of over-the-counter nasal drops, and compensatory thickening of the wider inferior turbinate due to imbalance in the width of the left and right nasal cavities caused by nasal septal kyphosis.

2.2 Simplified model for simulations

The simplified model of the nasal cavity proposed by Romani (2017) is presented in figure 2.3. This model consists of inlet sphere, fossa, nasal valve, turbinates, central septum-like structure, nasopharynx and laryngopharynx. To create various geometries and airflows,

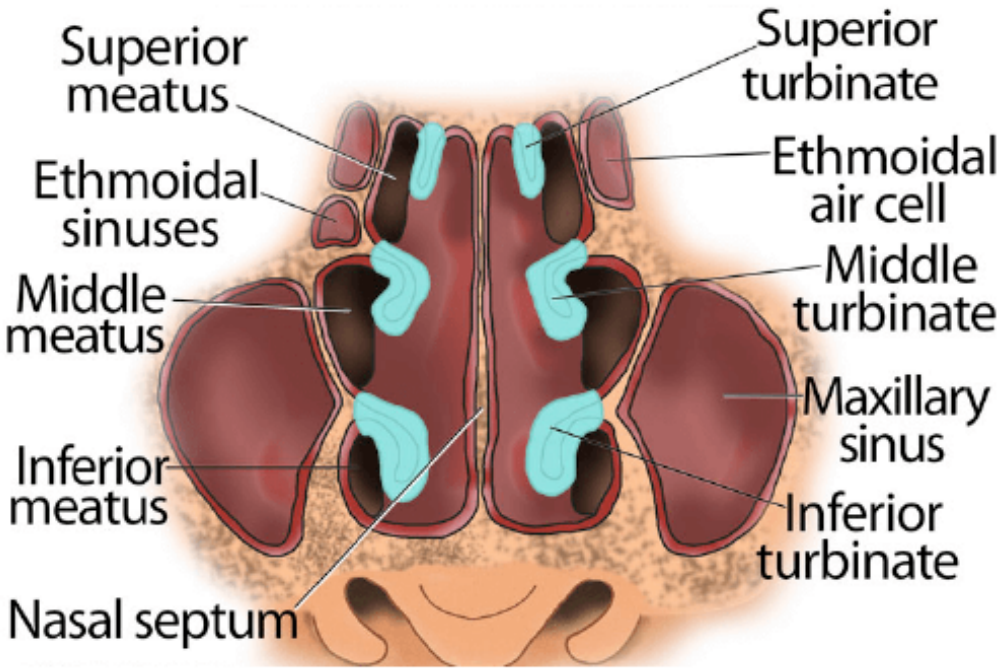

Figure 2.2: Schematic of nasal cavity (2) (*The Respiratory System.*, n.d.).

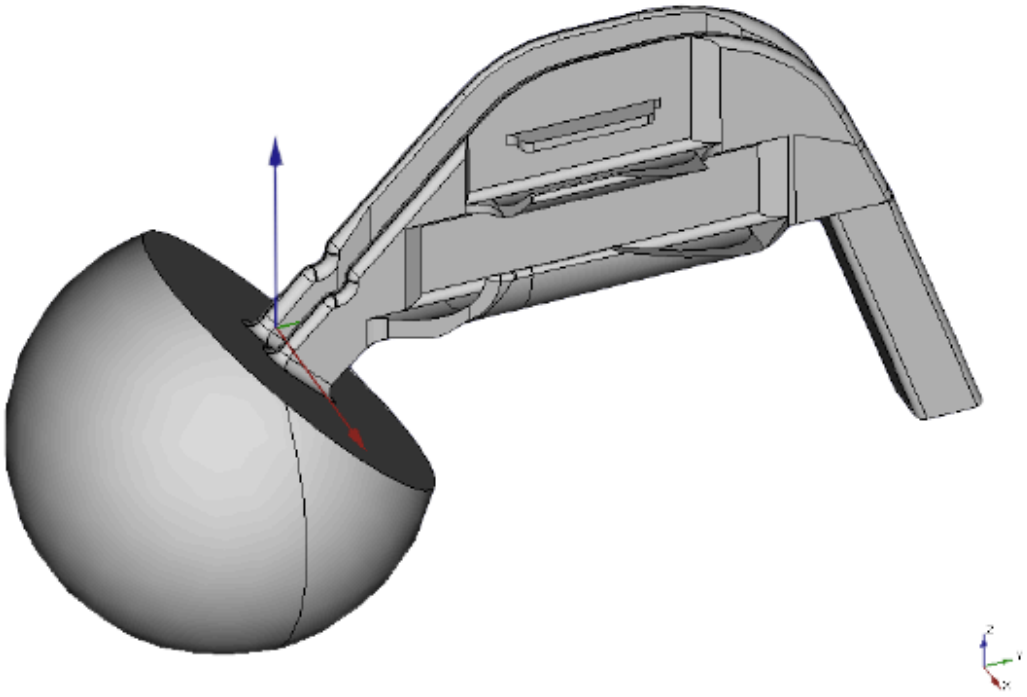

Figure 2.3: Model shape (Romani, 2017).

Table 2.1: The variations of nose geometry.

Variation	Non-pathological or pathological
Septum lateral position	non-pathological
Longitudinal position of superior meatus	non-pathological
Vertical position of superior meatus	non-pathological
Section change steepness at inferior meatus inlet	non-pathological
Nasal valve position	non-pathological
Head of inferior turbinate hypertrophy	pathological
Overall inferior turbinate hypertrophy	pathological
Head of middle turbinate hypertrophy	pathological

eight parameters for variations of geometry are defined. The eight parameters include five harmless anatomical differences between healthy individuals and three pathological variations, replicating the condition known as turbinate hypertrophies described above, as summarized in table 2.1. In this study, the last three parameters, i.e., head of inferior turbinate hypertrophy, overall inferior turbinate hypertrophy, and head of middle turbinate hypertrophy, are the target of the predictions. The range of variations of head of inferior turbinate hypertrophy, overall inferior turbinate hypertrophy and head of middle turbinate hypertrophy are set to 0.1 mm to 0.7 mm, 0.1 mm to 0.7 mm and 0.1 mm to 0.4 mm, respectively. Note that 0 mm indicates the healthy state for each parameter.

By combining these eight modifications, 200 different geometries are generated. They include 100 non-pathological cases and 100 combinations of turbinate hypertrophies.

2.3 Numerical simulations

In this study, the governing equations for nasal cavity flow are incompressible steady-state Reynolds-Averaged Navier-Stokes (RANS) equations, i.e.,

$$\nabla \cdot \overline{\boldsymbol{u}} = 0, \tag{2.1}$$

$$\nabla \cdot (\overline{\boldsymbol{u}\boldsymbol{u}}) + \nabla \cdot \left(\overline{\boldsymbol{u}'\boldsymbol{u}'}\right) + \frac{1}{\rho}\nabla \overline{p} = \nu \nabla^2 \overline{\boldsymbol{u}}. \tag{2.2}$$

In the equations above, $\boldsymbol{u}$, p and ρ denote the velocity vector, the pressure and the density, and $\bar{\cdot}$ and $\cdot'$ are the mean (time-averaged) component and the fluctuating component, e.g., $\overline{\boldsymbol{u}}$ represent mean velocity vector and $\boldsymbol{u}'$ velocity fluctuation. Using Reynolds decompo-

sition, the flow velocity vector can de decomposed as

$$\boldsymbol{u} = \overline{\boldsymbol{u}} + \boldsymbol{u}'. \tag{2.3}$$

The term $\overline{\boldsymbol{u}'\boldsymbol{u}'}$ in equation 2.2 is called as Reynolds stress tensor and represent the effect of the fluctuating motions in the mean flow field. To close the stress tensor term, we use k–ω SST (Menter, 1994) model as a turbulence model. This model contains two conservation equations: one for turbulent kinetic energy k and one for specific dissipation rate ω.

The boundary condition driving the flow into the nasal cavity is a total pressure-over-density difference of 20 $\mathrm{m}^2/\mathrm{s}^2$ between the inlet, represented by the spherical cap, and the outlet, which is the lower face of the laryngopharynx. This condition is similar conditions to a real nasal cavity airflow. The no-slip boundary condition is applied to the solid boundaries and a null normal gradient condition is used for the inlet and the outlet.

The numerical simulation is performed with OpenFOAM (v1806+), a widely used open source CFD tool. The simulation results are verified and validated by Romani (2017).

Chapter 3

Machine Learning Techniques

In recent years, machine learning techniques, which can automatically extract key features from tremendous amount of data, have achieved noteworthy results in various fields including fluid dynamics owing to the advances in the algorithms centering on deep learning (LeCun et al., 2015; Kutz, 2017; Brunton et al., 2020; Taira et al., 2020; Brenner et al., 2019; Fukami, Fukagata and Taira, 2020), which has been enabled by the recent development of computational power. The machine learning methods which are used in this study are presented in this section.

3.1 Linear regression

Linear regression is a method to solve the typical linear problem,

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta}, \tag{3.1}$$

where Y and X denote response variables and explanatory variable, respectively. In this problem, we obtain the coefficient matrix β representing the relationships between Y and X. Using the linear regression method, we can find an optimal coefficient matrix β so that the error between the left-hand side and right-hand side is minimized:

$$\beta = \mathrm{argmin}||Y - X\beta||^2. \tag{3.2}$$

3.1.1 Lasso

Least absolute shrinkage and selection operator (Lasso) (Tibshirani, 1996) was proposed as a sparse regression method which takes a constraint for the sparsity of the coefficient

matrix β, i.e., some components are zero, in the minimization process. With Lasso, the sum of the absolute value of the coefficient matrix is incorporated with the error to determine the coefficient matrix:

$$\beta = \text{argmin}\left(||Y - X\beta||^2 + \alpha \sum_j |\beta_j| \right). \tag{3.3}$$

If the sparsity constant α is set to a high value, the estimation error becomes relatively large while the coefficient matrix results in sparse. The Lasso algorithm introduced above has, however, two drawbacks. One of them is that the Lasso cannot select variables properly if there is a group of variables whose correlations are very high. To overcome this issue known as the collinearity problem, elastic net (Enet) (Zou and Hastie, 2005) was proposed. Another problem here is that the penalization methods usually show biased estimation with a large penalty. In other words, the predictions may sometimes result in non-optimal and do not perform well with a high sparsity parameter used to obtain a parsimonious and sparse model. To solve this issue called the lack of oracle property, the adaptive Lasso (Alasso) (Zou, 2006) was presented.

3.2 Neural networks

Neural network (NN) is one of the most famous and popular methods in machine learning. For instance, convolutional neural network (CNN) (LeCun et al., 1998) is able to capture the spacial information of two- or three-dimensional data, and it has widely been used in a wide range of scientific fields. Recurrent neural network (RNN) can process sequences of inputs using their internal states. Due to the strength in handling sequential data such as time-series data sets, the RNN is utilized for language processing and time series regression. However, RNN is often suffered from vanishing gradient problem, which is explained in section 3.2.7, because of its configuration. To overcome this problem, Hochreiter and Schmidhuber (1997) proposed long-short term memory.

In this section, the NN based models used in this study and fundamentals for machine learning are presented.

3.2.1 Neural network models

Multi-layered perceptron

A conventional-type neural network called multi-layer perceptron (MLP) namely consists of fully-connected layers of perceptrons. The MLP has an input, an output and hidden layers, and they are connected with the weights between each layer. The input of a layer is weighted and propagate to next layer:

$$\boldsymbol{u}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{z}^{(l)} + \boldsymbol{b}^{(l)}, \tag{3.4}$$

$$\boldsymbol{z}^{(l+1)} = f(\boldsymbol{u}^{(l)}), \tag{3.5}$$

where $\boldsymbol{z}^{(l)}, \boldsymbol{u}^{(l)}, \boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ are the output, the input, weights and biases of layer l and $f(\cdot)$ denotes an activation function.

Convolutional neural network

The convolutional neural network (CNN) (LeCun et al., 1998) has been widely used in the field of image recognition, and it has also been applied to fluid dynamics in recent years (Fukami, Fukagata and Taira, 2020; Fukami, Nabae, Kawai and Fukagata, 2019; Hasegawa et al., 2020a,b) due to its capability to deal with spatially coherent information. The CNN is formed by connecting two kinds of layers: convolution layers and sampling layers.

The convolutional operation performed in the convolution layer can be expressed as

$$s_{ijm} = \sum_{k=0}^{K-1}\sum_{p=0}^{H-1}\sum_{q=0}^{H-1} z_{i+p,j+q,k} W_{pqkm} + b_m, \tag{3.6}$$

where z_{ijk} is the input value at point (i, j, k), W_{pqkm} denotes the weight at point (p, q, k) in the m-th filter, b_m represents the bias of the m-th filter, and s_{ijm} is the output of the convolution layer. The schematics of the convolutional operation and a convolution layer without bias are shown in figures 3.1(a) and (b), respectively. The input is a three-dimensional matrix with the size of $L_1 \times L_2 \times K$, where L_1, L_2, and K are the height, the width, and the number of channels (e.g., $K = 3$ for RGB images), respectively. There are M filters

with the length H and the K channels. After passing the convolution layer, an activation function $f(\cdot)$ is applied to s_{ijm}, i.e.,

$$z_{ijm} = f(s_{ijm}). \tag{3.7}$$

Usually, nonlinear monotonic functions are used as the activation function $f(\cdot)$.

The sampling layer performs compression or extension procedures with respect to the input data. Here, we use a max pooling operation for the pooling layer, as summarized in figure 3.1 (c). Through the max pooling operation, the CNN is able to obtain the robustness against rotation or translation of the images. In contrast, the upsampling layer copies the values of the low-dimensional images into a high-dimensional field when the CNN extends the data from an input to an output, as presented in figure 3.1(d).

3.2.2 Activation function

Activation function $f(\cdot)$ is one of the most important parameters inside neural networks. As presented in section 3.2.1, an activation function is applied for the output of each layer to take nonlinearity into account. Well used activation functions described below,

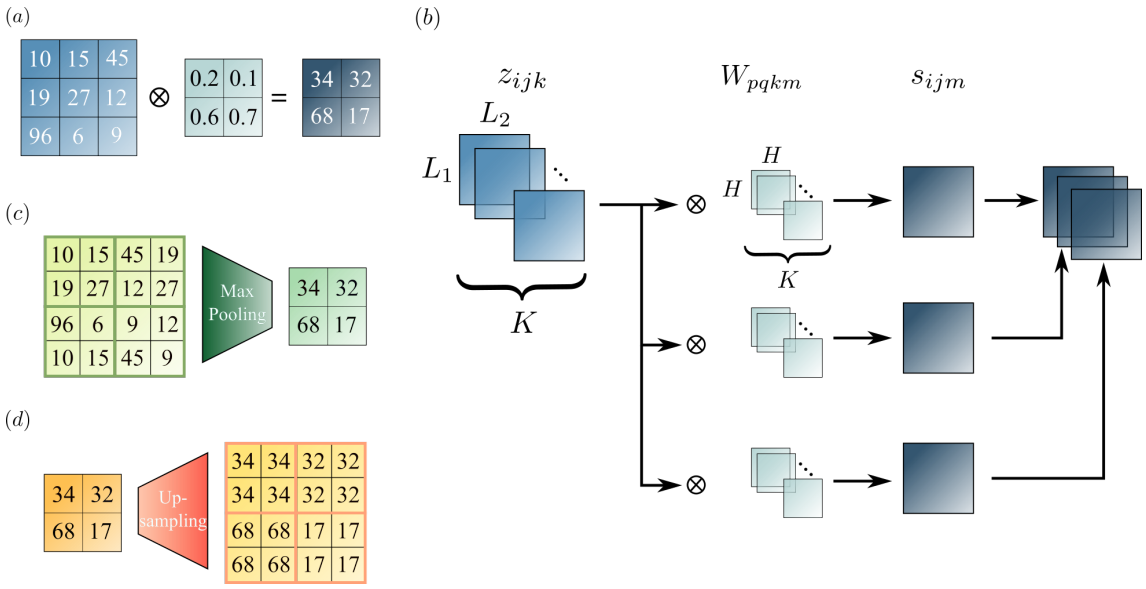

Figure 3.1: Operations in the convolutional layer and the sampling layer: (a) convolutional operation using a weighted filter W; (b) the computation in the convolution layer with $M = 3$; (c) max pooling operation. (d) upsampling operation.

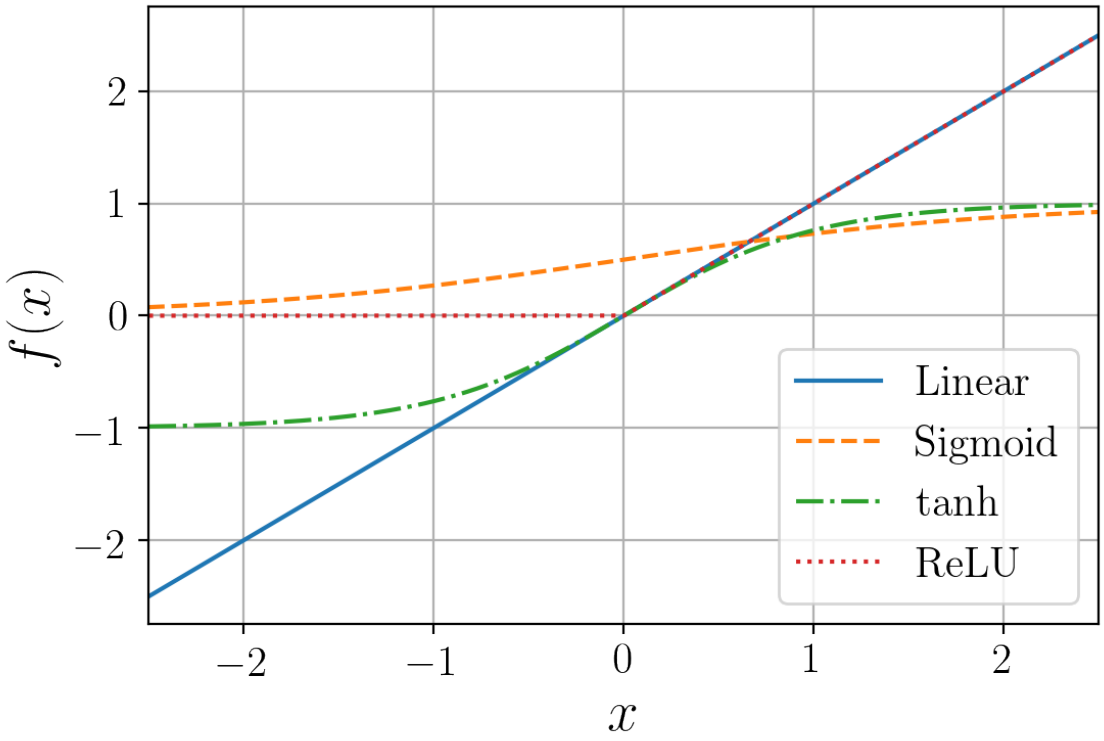

Figure 3.2: Activation functions.

i.e., linear mapping (mainly used in output layer of regression tasks), logistic sigmoid, hyperbolic tangent function (tanh), rectified linear unit (ReLU) (Glorot et al., 2011), are visually summarized in figure 3.2.

Linear mapping

Linear mapping function simply outputs the input

$$f(x) = x. \tag{3.8}$$

Therefore, the output of a layer becomes the multiplication of weights and inputs. Because the function output has the infinity range, it is usually used at the last layer for regression problems.

Logistic sigmoid

Logistic sigmoid is a nonlinear function defined as

$$f(x) = \frac{1}{1 + \exp(-x)}. \tag{3.9}$$

Since it maps the interval $(-\infty, \infty)$ to $(0, 1)$, it is often used for probability prediction, e.g., classification tasks. It is, however, well known that sigmoid function may be suffered from vanishing gradient problem when it is applied to hidden layers of deep and/or recurrent neural networks. The detailed explanation for the vanishing gradient problem is stated in section 3.2.7.

Hyperbolic tangent function

Hyperbolic tangent is also one of the well used nonlinear functions. The definition of this function is described as

$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^z + e^{-z}}. \tag{3.10}$$

The hyperbolic tangent is also suffered from vanishing gradient problem. However, it usually performs better than sigmoid because its derivatives can be lager than that of sigmoid.

ReLU

To avoid vanishing gradient problem, Glorot et al. (2011) proposed rectified linear unit (ReLU). This function can be expressed as

$$f(x) = \max\{0, x\}. \tag{3.11}$$

Since the gradient of this function is

$$f(x) = \begin{cases} x & (x > 0) \\ 0 & (x < 0), \end{cases} \tag{3.12}$$

it can avoid vanishing gradient problem. Moreover, ReLU function can be applied with less computational costs thanks to its simplicity compared to hyperbolic tangent and sigmoid, which require computing exponential function. From these reasons, ReLU function is the most commonly used activation function nowadays.

3.2.3 Loss function

The loss function is essential to train and evaluate machine learning models. Neural networks are optimized by maximizing or minimizing the loss function during its training

process. Here, we present two commonly used loss functions for regression problems.

Mean squared error (MSE)

Mean squared error is the mean value of squared error between the output data of network $\boldsymbol{q}^{\text{out}}$ and the desired output $\boldsymbol{q}^{\text{des}}$:

$$\text{MSE} = \frac{1}{M}\sum_{i=1}^{M}\left(\boldsymbol{q}_i^{\text{des}} - \boldsymbol{q}_i^{\text{out}}\right)^2, \tag{3.13}$$

where M denotes the total number of collocation points.

Mean absolute error (MAE)

Mean absolute error is the mean value of absolute error defined as:

$$\text{MAE} = \frac{1}{M}\sum_{i=1}^{M}\left(|\boldsymbol{q}_i^{\text{des}} - \boldsymbol{q}^{\text{out}_i}|\right), \tag{3.14}$$

3.2.4 Optimizer

As mentioned in section 3.2.3, the training of the neural networks is equivalent to optimize the weights $\boldsymbol{W}$ in the model such that

$$\boldsymbol{W} = \text{argmin}_{\boldsymbol{W}} E(\boldsymbol{W}), \tag{3.15}$$

where $E(\boldsymbol{W})$ is a loss function. To minimize the loss function, various methods based on stochastic gradient descent are used as the optimizers for the training.

Stochastic gradient descent

Stochastic gradient descent (SGD) is an application of gradient descent method. Therefore, the former gradient descent method has to be discussed before moving to SGD.

Gradient descent method is the basis of the optimization technique. The purpose of this method in neural network algorithms is to seek optimized weights which satisfy equation 3.15. The weights are updated as

$$\boldsymbol{W}^{n+1} = \boldsymbol{W}^n - \epsilon\nabla E, \tag{3.16}$$

where ϵ is a learning rate and ∇E is a gradient of a loss function with respect to the weights. The gradient can be written as

$$\nabla E = \frac{\partial E}{\partial \boldsymbol{W}} = \left[\frac{\partial E}{\partial W_1} \cdots \frac{\partial E}{\partial W_n} \right]^T, \tag{3.17}$$

where n is the number of weights. In this way, the optimized weights can be obtained by updating the weights until the loss function becomes small enough. This gradient descent method is also called batch learning because all data are used to compute the loss function. Suppose we have M data. The loss function to update the weights in the gradient method can be computed as

$$E(\boldsymbol{W}) = \frac{1}{M} \sum_{i=1}^{M} E_i(\boldsymbol{W}). \tag{3.18}$$

On the other hand, mini-batch learning, utilized in SGD, can improve the learning efficiency and avoid convergence to a local minima solution. In the SGD, the data are divided into mini-batches, which is the arbitrary number of samples of data, and the loss function is computed from each mini-batch. Here, we denote t_{mb} mini-batch as $D_{t_{\mathrm{mb}}}$ and the number of samples included in the batch is expressed as $M_{t_{\mathrm{mb}}} = |D_{t_{\mathrm{mb}}}|$. The loss function on mini-batch is computed as

$$E_{t_{\mathrm{mb}}}(\boldsymbol{W}) = \frac{1}{M_{t_{\mathrm{mb}}}} \sum_{i \in D_{t_{\mathrm{mb}}}} E_i(\boldsymbol{W}), \tag{3.19}$$

and the weights are updated as

$$\boldsymbol{W}^{n+1} = \boldsymbol{W}^n - \epsilon \nabla E_{t_{\mathrm{mb}}}. \tag{3.20}$$

The operation improves the efficiency of the training procedure, especially for the training data has high redundancy. Moreover, SGD can avoid converging to a local minima solution by evaluating the network with a number of loss functions because the loss function $E_{t_{\mathrm{mb}}}$ vary among the mini-batches.

Another important parameter for these optimization process is the learning rate ϵ. If the learning rate is too large, it may not be able to converge to a minimum point. On the other hand, a smaller learning rate increases the time to converge. There are methods to determine the learning rate, and two of them, i.e. RMSProp and Adam, are introduced as follows.

Momentum method

Momentum is introduced to suppress oscillations of the weight updating. In the momentum method, variation of the weight $\Delta \boldsymbol{W}^{n-1} = \boldsymbol{W}^{n} - \boldsymbol{W}^{n-1}$ is added to equation 3.16 such that:

$$\boldsymbol{W}^{n+1} = \boldsymbol{W}^{n} - \epsilon \nabla E + \mu \Delta \boldsymbol{W}^{n-1}, \tag{3.21}$$

where μ is the parameter to set the ratio of addition.

RMSProp

The purpose of RMSProp, which is a revision of AdaGrad (Duchi et al., 2011), also suppresses the oscillation of the weight updating. The learning rate is adjusted in the iteration loop automatically. The weight updating equation is written as

$$v^{n} = \rho v^{n-1} + (1-\rho)(\nabla E)^2, \tag{3.22}$$

$$\eta = \frac{\epsilon}{\sqrt{v^{n} + \delta}}, \tag{3.23}$$

$$\boldsymbol{W}^{n+1} = \boldsymbol{W}^{n} - \eta \nabla E, \tag{3.24}$$

where ρ is a decay rate and δ is a small constant to prevent zero division. The v^{n} represented by equation 3.22 becomes large when the gradient ∇E changes rapidly. The original learning rate ϵ is divided by v^{n} in equation 3.24. Therefore, the actual learning rate η is small when the oscillations of loss function are occurred.

Adam

Adam (Kingma and Ba, 2014), a combination of momentum method and RMSProp, is one of the most commonly used optimizers due to its efficiency and stability. The equations

for updating weights are summarized as

$$
\begin{aligned}
m^{n+1} &= \mu m^n + (1-\mu)\nabla E, && (3.25)\\
v^{n+1} &= \rho v^n + (1-\rho)(\nabla E)^2, && (3.26)\\
\widehat{m^{n+1}} &= \frac{m^{n+1}}{1-\mu}, && (3.27)\\
\widehat{v^{n+1}} &= \frac{v^{n+1}}{1-\rho}, && (3.28)\\
\boldsymbol{W}^{n+1} &= \boldsymbol{W}^n - \epsilon \frac{\widehat{m^{n+1}}}{\sqrt{\widehat{v^{n+1}}} + \delta}. && (3.29)
\end{aligned}
$$

3.2.5 Backpropagation

In the above section, how neural networks seek optimized weights was presented. Next, a technique to compute the gradient of loss function with respect to the weights $\nabla E = \dfrac{\partial E}{\partial \boldsymbol{W}}$ is introduced. To compute this, there is an important strategy called backpropagation. In the backpropagation, the gradients are propagated from back, i.e., output of the network towards the input layer, as the name implies. Although the mathematical procedure for this process can be, of course, established for each neural network model, the procedure for MLP is only expressed here for brevity.

Consider a perceptron at the l th layer of MLP with the weights $W_{ij}^{(l)}$ and the input of the perceptron $u_j^{(l)} = W_{ij}^{(l)} z_j^{(l)}$. To simplify the problem, the bias $\boldsymbol{b}^{(l)}$ is neglected. The derivatives with respect to this weight $W_{ij}^{(l)}$ can be written as

$$
\frac{\partial E}{\partial W_{ij}^{(l)}} = \frac{\partial E}{\partial u_j^{(l)}} \frac{\partial u_j^{(l)}}{\partial W_{ij}^{(l)}}. \tag{3.30}
$$

The perturbation of $u_j^{(l)}$ propagates forward only through the input of perceptrons at the next layer. Therefore, ths derivative of E with respect to $u_j^{(l)}$, which is the left part of the right hand side of equation 3.30, is rewritten by the input of a k th perceptron at the $l+1$ layer $u_k^{(l+1)}$ as

$$
\frac{\partial E}{\partial u_j^{(l)}} = \sum_k \frac{\partial E}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}}. \tag{3.31}
$$

Here, we define $\delta_j^{(l)}$ as

$$\delta_j^{(l)} = \frac{\partial E}{\partial u_j^{(l)}}, \tag{3.32}$$

and the equation 3.31 can be transformed as

$$\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} \left(W_{kj}^{(l+1)} f'(u_j^{(l)}) \right), \tag{3.33}$$

where f is the activation function. Note that $\partial u_k^{(l+1)} / \partial u_j^{(l)}$ can be expressed as

$$\frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}} = W_{kj}^{(l+1)} f'(u_j^{(l)}), \tag{3.34}$$

using

$$u_k^{(l+1)} = \sum_j W_{kj}^{(l+1)} z_j^{(l)} = \sum_j W_{kj}^{(l+1)} f(u_j^{(l)}). \tag{3.35}$$

This equation 3.33 for $\delta_j^{(l)}$ means that the $\delta_j^{(l)}$ can be computed from $\delta_k^{(l+1)}$; in other words, δ is propagated from backward, i.e., the output of the network. Let us consider again the equation 3.30. The right part of the right hand side $\partial u_j^{(l)} / \partial W_{ij}^{(l)}$ is rewritten as

$$\frac{\partial u_j^{(l)}}{\partial W_{ij}^{(l)}} = z_i^{(l-1)}. \tag{3.36}$$

Using these transformations, the equation 3.30 finally becomes

$$\frac{\partial E}{\partial W_{ij}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}. \tag{3.37}$$

In this way, the derivatives of the loss function with respect to the weights are computed by the backpropagation.

3.2.6 Training and evaluation

In this section, how neural networks are trained and evaluated is discussed.

The data set with an input and an output is divided into training, validation, and test data. The dataset used to train a model is called training data. The data set used for checking the performance of a model is called validation data and is not utilized to update the weights of the algorithm. Test data are used to measure the performance of a trained model.

In training session, the training and validation data are used. When training a model, we usually use up all these data. Therefore we reuse the data to keep training the model. The unit of training between refreshing of the data is called epoch. The order of the data is shuffled for each epoch, which means that the training procedure is never the same, even though we are using the same data as a whole. For each epoch, the training data are divided into an arbitrary size of mini-batches, and the weights of model are updated using these mini-batches. A mini-batch is fed into the model, and the training loss, is computed from the output of the model and the desired output. The derivatives of the loss function with respect to each weight are computed using the backpropagation algorithm. An optimizer is then utilized to update the weights from the derivatives. Once the training for all mini-batches is done, the validation loss is computed with the validation data. The procedure is repeated until the validation loss dose not improve.

To evaluate a model, the test data is finally used. Note that the test data is not used in the training process. In other words, the test data can be regarded as unseen data for a neural network.

3.2.7 Major problems in neural network models

In this section, the major problems in neural network models, i.e., overfitting and vanishing gradient problem, are discussed.

Overfitting

Overfitting is one of the biggest problems not only for neural networks but also for a wide range of machine learning techniques (Brunton and Kutz, 2019). Generally, the models are updated with training data only as explained in section 3.2.6. As the training procedure, a model becomes to show the better result for the training data and worse for the test data. This phenomenon is called overfitting. To examine whether the model is overfitted or not, the validation loss plays an important role. The validation data, which is used in the training process but not used to update the weights, is used to investigate whether the model is overfitted or not. If the training loss decreases while the validation

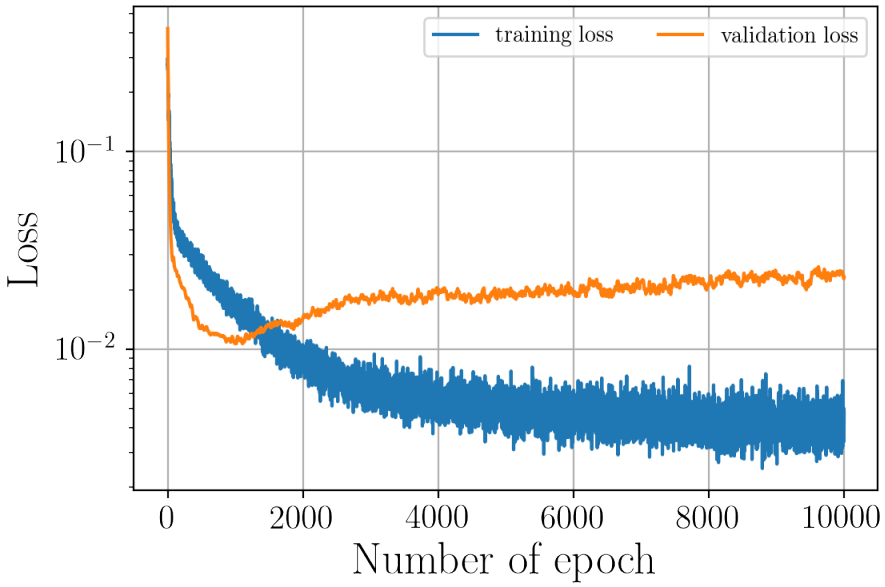

Figure 3.3: An example of training and validation loss history of the overfitted model.

loss increases, it can be regarded that the model is overfitting to the training loss. As an example, a history of the training and validation loss of an overfitted model is shown in figure 3.3.

Vanishing gradient problem

The vanishing gradient problem is namely the problem which the gradient of loss function vanishes in the backpropagation process. This problem often happens when the model has a large number of layer and/or is the recurrent type of neural networks, e.g., RNN and LSTM. The reason for this problem is easy to see from equation 3.33. When computing the derivative of loss function with backpropagation, the derivative of the activation function $f'(\cdot)$ is multiplied as many times as the number of layers. If the range of the derivative of the activation function $f'(\cdot)$ is lower than 1, the derivatives of loss function vanishes at the upper layers of the model. This problem stops the learning process of the model.

3.2.8 Batch normalization

Batch normalization proposed by Ioffe and Szegedy (2015) is one of the most important machine learning techniques. The advantages of using batch normalization are as follows:

- Being able to increase learning rate.
- Not so dependent on initial weights.
- Preventing overfitting.

These advantages solve major problems in machine learning. In the batch normalization layer, the data in the mini-batch are normalized as

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \tag{3.38}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2, \tag{3.39}$$

$$\widehat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \tag{3.40}$$

where $x_i, m, \mu, \sigma_B^2, \varepsilon$, and $\widehat{x_i}$ are the input of the batch normalization layer, the number of data in the mini-batch, the mean value of the data, the standard deviation of the data, a small constant to prevent zero division, and the output of the batch normalization layer.

3.2.9 Dropout

Dropout (Srivastava et al., 2014) is also widely used technique to prevent overfitting. Using dropout, a fixed percentage p of perceptrons is randomly disabled in the training session, as shown in figure 3.4 (b). On the other hand, in the prediction session, the predictions are computed with all perceptrons, as illustrated in figure 3.4 (a). This procedure is equivalent to ensemble learning, where multiple models are trained and the average of each prediction is the output. This operation is effective in suppressing overfitting.

3.2.10 Cross validation

Cross validation (CV) is a technique where we partition the training data into several folds and let the model learn and test on different dataset for each folds, as illustrated in figure

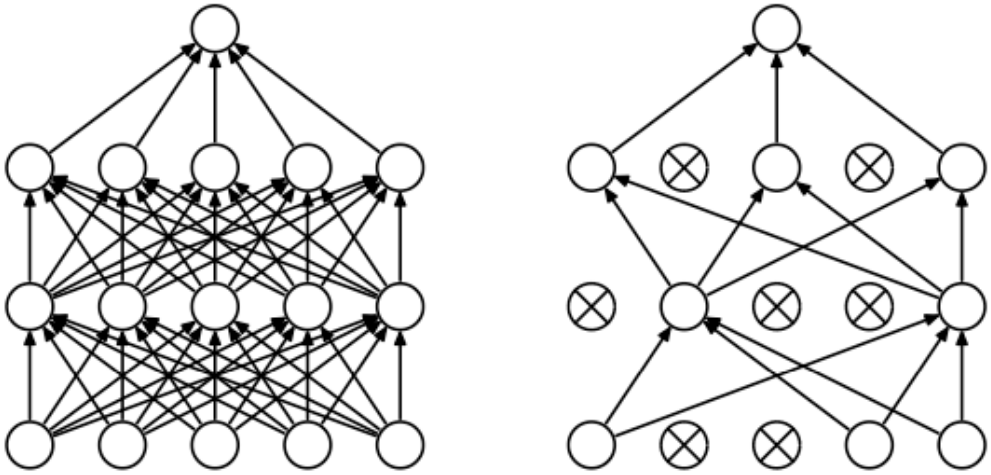

Figure 3.4: Dropout neural net model. (*a*): A standard neural net with 2 hidden layers. (*b*): An example of a thinned net produced by applying dropout to the network on (*a*). Crossed units have been dropped. (Srivastava et al., 2014)

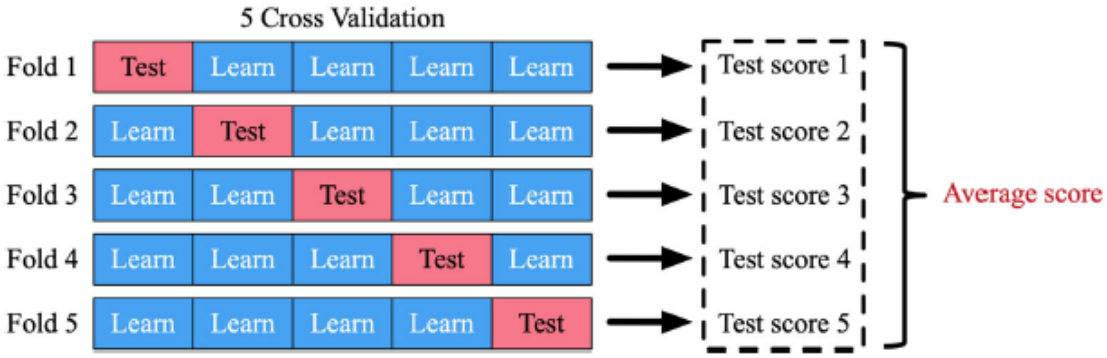

Figure 3.5: An example of fivefold cross validation. The blue boxes indicate training data used for deriving machine learning model.The red boxes indicate test data used to obtain test scores. An average score of five test scores are used for the optimization procedure and assessment (Fukami, Fukagata and Taira, 2020).

3.5. We then obtain an average score over the considered number of CVs. Although it is tedious, this method enables us to verify how well a model can deal with a population of data.

Chapter 4

Functional Maps

In this chapter, the theory of functional maps proposed by Ovsjanikov et al. (2012) is presented. The functional map is a method for shape matching and is utilized to extract the geometrical information in this study.

4.1 Functional map representation

The functional map representation of correspondence of two objects is summarized in this section. Consider two geometric objects such as a pair of shapes in 3D denoted by $\mathcal{M}$ and $\mathcal{N}$. A bijective mapping between points on $\mathcal{M}$ and $\mathcal{N}$ is expressed as $\mathcal{T} : \mathcal{M} \to \mathcal{N}$. In other words, if p is a point on $\mathcal{M}$, then $\mathcal{T}(p)$ is a corresponding point on $\mathcal{N}$. This $\mathcal{T}$ induces a natural transformation of functions on $\mathcal{M}$. Suppose that a scalar function $f : \mathcal{M} \to \mathbb{R}$ is defined on $\mathcal{M}$ then a corresponding function $g : \mathcal{N} \to \mathbb{R}$ can be described as $g = f \circ \mathcal{T}^{-1}$. Let us denote the transformation of functions by $\mathcal{T}_F : \mathcal{F}(\mathcal{M}, \mathbb{R}) \to \mathcal{F}(\mathcal{N}, \mathbb{R})$, where $\mathcal{F}(\cdot, \mathbb{R})$ is a generic space of real-valued functions. The transformed function on $\mathcal{N}$, i.e., g, can be expressed as $g = \mathcal{T}_F(f)$. This transformation $\mathcal{T}_F$ is called functional representation of T. There are following two important remarks regarding to this functional representation $\mathcal{T}_F$.

- The original mapping $\mathcal{T}$ can be recovered from the functional representation $\mathcal{T}_F$.
- For any fixed bijective map $\mathcal{T} : \mathcal{M} \to \mathcal{N}$, $\mathcal{T}_F$ is a linear map between functional spaces.

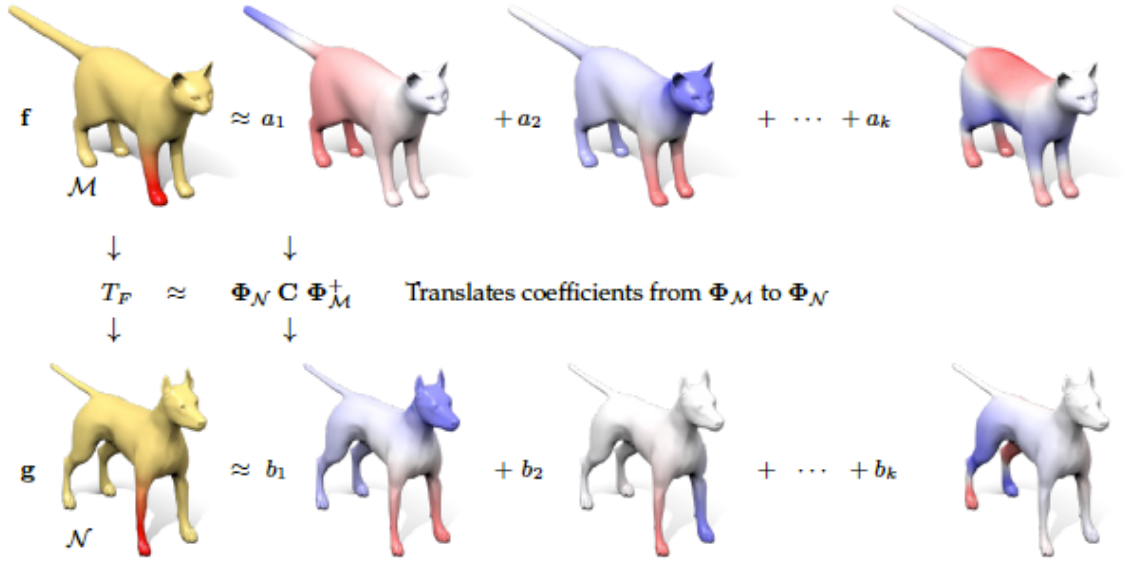

Figure 4.1: The schematic for the functional maps (Ovsjanikov et al., 2017). When the functional spaces of source and target shapes $\mathcal{M}$ and $\mathcal{N}$ are endowed with bases $\phi_{\mathcal{M}}$ and thus every function can be written as a linear combination of basis functions, then a linear functional map $\mathcal{T}_F$ can be expressed via a matrix C that intuitively "translates" the coefficients from one basis to another.

The functional map which fully encodes the original map $\mathcal{T}$ is derived by considering a basis of the objects. Suppose that the function space of $\mathcal{M}$ is decomposed by a basis so that any function $f : \mathcal{M} \to \mathbb{R}$ can be expressed as a linear combination of basis functions $f = \sum_i a_i \phi_i^{\mathcal{M}}$, where a_i and $\phi_i^{\mathcal{M}}$ are coefficients and basis functions. Suppose also that $\mathcal{N}$ is equipped with a set of basis functions $\phi_j^{\mathcal{N}}$ and any function $g : \mathcal{N} \to \mathbb{R}$ can be phrased as $g = \sum_j b_j \phi_j^{\mathcal{N}}$, where a_j and $\phi_j^{\mathcal{N}}$ are coefficients and basis functions. Then the functional representation is described as

$$\mathcal{T}_F(f) = \mathcal{T}_F\left(\sum_i a_i \phi_i^{\mathcal{M}}\right) = \sum_i a_i \mathcal{T}_F\left(\phi_i^{\mathcal{M}}\right). \tag{4.1}$$

Then $\mathcal{T}_F(\phi_i^{\mathcal{M}}) = \sum_j c_{ji} \phi_j^{\mathcal{N}}$ for coefficients c_{ji}, and $\mathcal{T}_F(f)$ can be rewritten as

$$\mathcal{T}_F(f) = \sum_j \sum_i a_i c_{ji} \phi_j^{\mathcal{N}}. \tag{4.2}$$

Note again that $\mathcal{T}_F(f) = g$ and $g = \sum_j b_j \phi_j^{\mathcal{N}}$, then equation 4.2 simply denotes:

$$b_j = \sum_i c_{ji} a_i, \tag{4.3}$$

where c_{ji} is independent of f and is completely determined by the bases and the map T. The coefficients c_{ji} represent jth coefficient of $\mathcal{T}_F(\phi_i^{\mathcal{M}})$ in the basis $\phi_j^{\mathcal{N}}$. The coefficients matrix $\boldsymbol{C}$ can be expressed as $c_{ji} = \langle \mathcal{T}_F(\phi_i^{\mathcal{M}}), \phi_j^{\mathcal{N}} \rangle$, where $\langle \cdot, \cdot \rangle$ denotes inner product, when the basis functions are orthogonal. The functional representation $\mathcal{T}_F$ can be represented by a functional map C such that $\mathcal{T}_F(\boldsymbol{a}) = \boldsymbol{C}\boldsymbol{a}$ for any function f represented as a vector of coefficients $\boldsymbol{a}$ of basis functions.

Based on the above discussion, let us describe the definition of the general functional maps defined by Ovsjanikov et al. (2012).

Definition

Let $\phi_i^{\mathcal{M}}$ and $\phi_j^{\mathcal{N}}$ be bases for $\mathcal{F}(\mathcal{M}, \mathbb{R})$ and $\mathcal{F}(\mathcal{N}, \mathbb{R})$, respectively. A generalized linear functional mapping $\mathcal{T}_F : \mathcal{F}(\mathcal{M}, \mathbb{R}) \to \mathcal{F}(\mathcal{N}, \mathbb{R})$ with respect to these bases is the operator defined by

$$\mathcal{T}_F\left(\sum_i a_i \phi_i^{\mathcal{M}}\right) = \sum_j \sum_i a_i c_{ji} \phi_j^{\mathcal{N}}, \tag{4.4}$$

where c_{ji} is a possibly infinite matrix of real coefficients (subject to conditions that guarantee convergence of the sums above).

For choice of basis, the Laplace-Beltrami eigenfunctions are suited for the basis in the view point of compactness and stability. Compactness means that functions on a shape can be approximated by a small number of basis, while stability represents that the function space defined by all linear combinations of basis functions are stable under small shape deformations. The eigenfunctions of the Laplace-Beltrami operator are ordered from "low frequency" to "higher frequency," and the space of functions defined by the first n eigenfunctions of the Laplace-Beltrami operator is stable under near-isometries as long as the nth and $(n + 1)$th eigenvalues are well separated.

4.2 Functional map inference

In this section, how the functional maps are inferred is discussed when a map $\mathcal{T}$ between two discrete objects $\mathcal{M}$ and $\mathcal{N}$ are unknown. The functional maps are suited for

map inference, i.e, constrained optimization. Many constraints on the map become linear constraints in its functional representation, and the functional maps are inferred by optimizing the matrix C that satisfies the constraints in a least square sense. The constraints are described below.

4.2.1 Function preservation

The correspondence between $f : \mathcal{M} \to \mathbb{R}$ and $g : \mathcal{N} \to \mathbb{R}$ can be written as

$$\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b}. \tag{4.5}$$

This function preservation constraint is quite general and includes the following as special cases.

Descriptor preservation

If f and g correspond to a point descriptor, e.g. $f(x) = \kappa(x)$ where $\kappa(x)$ is Gauss curvature of $\mathcal{M}$ at x, then the functional preservation means that the descriptors are approximately preserved by the mapping.

Landmark point correspondances

If the known corresponding landmark point is given, e.g. $T(x) = y$ for some known $x \in \mathcal{M}$ and $y \in \mathcal{N}$, these knowledge can be included in the functional constraints by considering, for example, normally distributed functions around x and y.

Segment correspondances

If the correspondences between segments if objects, they are similarly incorporated into the functional constraints by selecting an appropriate functions.

To summarize, the functional constraints are expressed as

$$\boldsymbol{C}\boldsymbol{a}_i \approx \boldsymbol{b}_i, \tag{4.6}$$

where $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ represent vector of coefficients of functions f_i and g_i, respectively. Here, f_i and g_i denote a set of pairs of functions for constraints. Considering $\boldsymbol{A} = \boldsymbol{a}_i$ and $\boldsymbol{B} = \boldsymbol{b}_i$, the energy to be minimized to optimize $\boldsymbol{C}$ is expressed as

$$E_1(\boldsymbol{C}) = \|\boldsymbol{C}\boldsymbol{A} - \boldsymbol{B}\|^2. \tag{4.7}$$

4.2.2 Operator commutativity

The another constraint on the map is commutativity with respect to linear operator on $\mathcal{M}$ and $\mathcal{N}$. Consider given functional operators $\mathcal{S}^{\mathcal{M}_F}$ and $\mathcal{S}^{\mathcal{N}_F}$ on $\mathcal{M}$ and $\mathcal{N}$, respectively. It may natural that the functional map $\boldsymbol{C}$ commutes with these operators. This can be written as

$$\|\mathcal{S}^{\mathcal{N}_F}\boldsymbol{C} - \boldsymbol{C}\mathcal{S}^{\mathcal{M}_F}\| = 0. \tag{4.8}$$

The energy to minimize to satisfy this constraint is described as

$$E_2(\boldsymbol{C}) = \|\mathcal{S}^{\mathcal{N}_F}\boldsymbol{C} - \boldsymbol{C}\mathcal{S}^{\mathcal{M}_F}\|^2. \tag{4.9}$$

When the $\boldsymbol{C}$ is decomposed by the basis of the first k eigenfunctions of the Laplace-Beltrami operator and $\mathcal{S}_F^{\mathcal{M}}$ and $\mathcal{S}_F^{\mathcal{N}}$ represent the Laplace-Beltrami operators, the equation 4.9 can be rewritten as

$$E_2(\boldsymbol{C}) = \sum_{i,j} \boldsymbol{C}_{ij}^2 \left(\lambda_i^{\mathcal{N}} - \lambda_j^{\mathcal{M}}\right)^2, \tag{4.10}$$

where $\lambda^{\mathcal{M}}$ and $\lambda^{\mathcal{N}}$ are the eigenvalues of the corresponding operators.

4.2.3 Estimating functional maps

An unknown functional maps between a pair of objects can be recovered by solving the following optimization problem:

$$\begin{aligned} \boldsymbol{C} &= \operatorname{argmin}_{\boldsymbol{X}} \left(E_1(\boldsymbol{X}) + E_2(\boldsymbol{X})\right) \\ &= \operatorname{argmin}_{\boldsymbol{X}} \left(\|\boldsymbol{X}\boldsymbol{A} - \boldsymbol{B}\|^2 + \alpha\|\boldsymbol{\Lambda}^{\mathcal{N}}\boldsymbol{X} - \boldsymbol{X}\boldsymbol{\Lambda}^{\mathcal{M}}\|^2\right), \end{aligned} \tag{4.11}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the function preservation constraints expressed in the basis of the eigenfunctions of the Laplace-Beltrami operator, $\boldsymbol{\Lambda}^{\mathcal{M}}$ and $\boldsymbol{\Lambda}^{\mathcal{N}}$ are diagonal matrices of eigenvalues of the Laplace-Beltrami operators and α is a weight parameter for the operator commutativity.

Chapter 5

Prediction with machine learning models

In this chapter, the pathological parameters of the nasal cavity models are predicted by using the machine learning models. This chapter is divided into two main sections: prediction from geometrical feature only and prediction from geometry and flow variables such as pressure and wall shear stress. Since the fluid dynamics feature is considered to be effective for the diagnosis of the nasal pathologies, these two predictions are compared and evaluated.

5.1 Prediction from functional maps

In this section, the results of the machine learning model trained by the functional maps between a reference nose and target noses are presented.

5.1.1 Data preparation

To extract features from the geometry of nose, the functional maps are computed. One of the healthy noses is selected from 200 noses as a reference nose. The functional maps are computed between the reference nose and the other 199 noses. To compute initial functional map, 30 bases of the Laplace-Beltrami operator for both reference and target noses are used. For the Laplace-Beltrami operator, discrete FEM Laplacians is used. The the obtained 30×30 functional maps are upsampled by ZoomOut (Melzi et al., 2019) up to the size of the functional maps become 171×171. Note that the map is computed

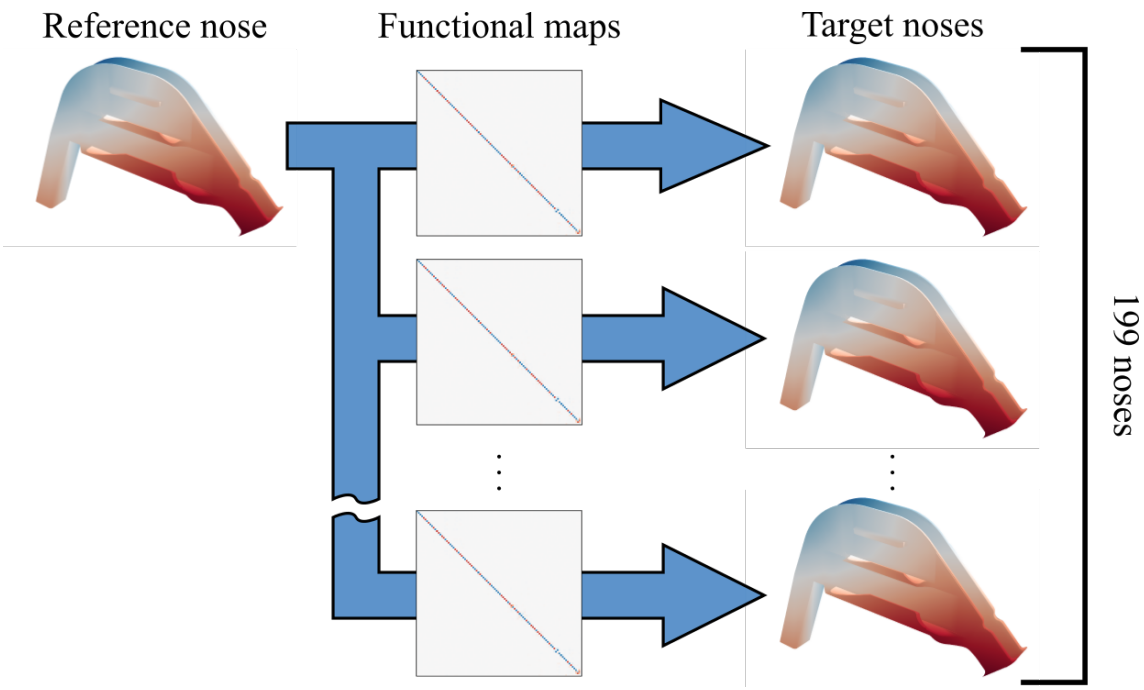

Figure 5.1: Mapping a function defined on the reference nose onto each target nose by corresponding functional map. The color scale on noses represents the function. The functional maps are computed between the healthy reference nose and the other noses.

for each target nose, i.e., the number of map computed here is 199. Here, the mapping results by computed functional maps are illustrated in figure 5.1. A function defined on the reference nose, which is represented in color scale on the nose, is mapped into target noses using the corresponding functional maps. To ensure this, the function highlighting the inferior turbinate is defined on the reference nose and mapped into a pathological nose, as shown in figure 5.2. The values highlighting the inferior turbinate, colored by red, are successfully mapped from the left reference nose into the right pathological nose, although the pathological nose has a thinner inferior turbinate.

This figure shows that the function successfully mapped into target nose. Of course, the same result is obtained for the target noses which are not presented in this figure.

The size of functional maps fed into a machine learning model is an important parameter for predicting the pathologies. This parameter indicates how many bases are included in the map. The higher-order bases correspond to finer nasal structures, thus the maps with the larger size can capture smaller scales of nasal structures. As the examples, some

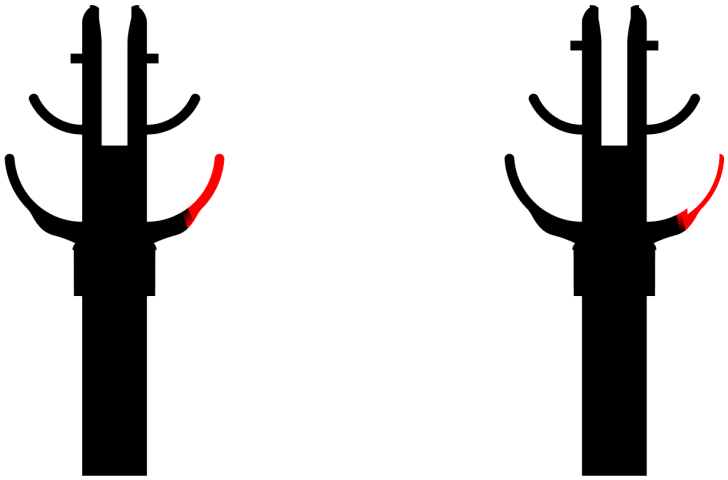

Figure 5.2: An example of the correspondences encoded in the functional map. The function to highlight the inferior turbinate defined on the left reference nose are mapped onto the right pathological nose.

of the functional maps with size of 20×20 extracted from original 171×171 maps are shown in figure 5.3. The heat maps represent the functional maps, and the values above the maps are the pathological parameters for inferior turbinate head, inferior turbinate body and middle turbinate head from the left. respectively. Note that again the functional map, shown here as the heat map, is the coefficients matrix $\boldsymbol{C}$ explained in section 4.1.

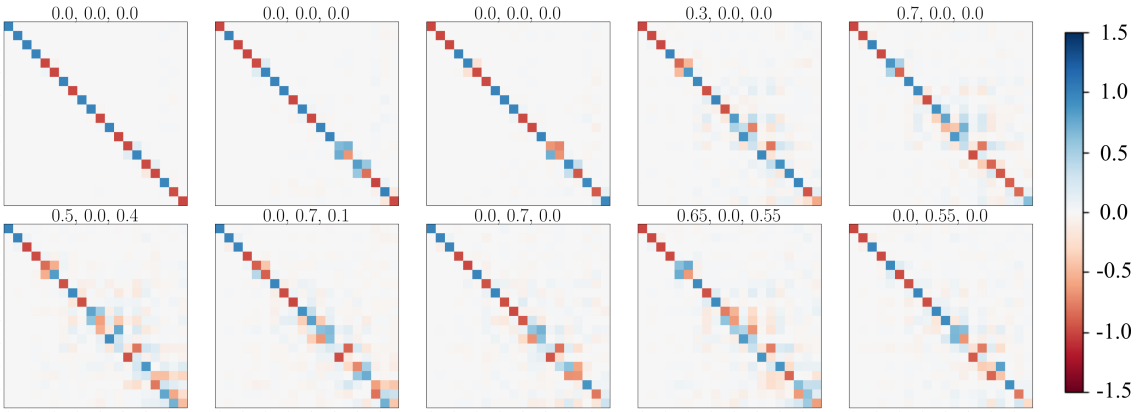

Figure 5.3: The examples of 20×20 functional maps. The heat map represent the values of the matrix, i.e., functional map. The values above the heat map are the pathological parameters for inferior turbinate head, inferior turbinate body and middle turbinate head.

The coefficients matrix $\boldsymbol{C}$ can be expressed as $c_{ji} = \langle \mathcal{T}_F(\phi_i^{\mathrm{ref}}), \phi_j^{\mathrm{tar}} \rangle$, where ϕ_i^{ref} and ϕ_j^{tar} are the basis functions of reference and target noses, which implies that the coefficients matrix $\boldsymbol{C}$ becomes completely diagonal if the reference and target noses are identical. The trend, that the functional maps of the pathological noses have non-zero values at the higher order non-diagonal components of the matrix, is confirmed. This trend appears between 20×20 and 40×40 maps. Therefore, the sizes of the map are set as $20\times20, 30\times30$ and 40×40, and their dependencies on prediction are investigated. Note again that these maps are extracted from original 171×171 maps.

5.1.2 Model configuration and learning conditions

The machine learning model configuration is really important for machine learning models to reasonably learn data. We propose three models: CNN, MLP, and singular value decomposition (SVD) model. The detailed algorithms to construct each model are presented in figure 5.4. The blocks of layers are defined in figure 5.4 (*a*). In the "Conv block", the data are fed into a convolution layer with 3×3 filter, which is expressed as "Conv(3, 3)" in the figure. The dropout (Srivastava et al., 2014) is applied to this layer with dropout ratio 0.2. In the training session, 20% of the weights of this layer are randomly deactivated. After the convolution layer, the data are normalized by batch normalization layer (Ioffe and Szegedy, 2015). These dropout and batch normalization are applied to avoid the overfitting. Since the machine learning models in the present study are trained with the limited amount of data, i.e., 200 noses, and it is widely known that learning from the insufficient amount of data causes overfitting, the utilization of these techniques can be expected as powerful methods to retain the generalizability of the model. The output of the batch normalization layer are input to tanh function which is selected as activation function. Then, the data size is reduced through the maxpooling layer with 2×2 filter. The "FC block(*n*)" defines the block which consists of the fully-connected layer with *n* perceptrons, dropout, batch normalization, and hyperbolic tangent function. This configuration aims to avoid overfitting as well as the "Conv block". The presented three machine learning models are constructed with these blocks. The CNN model handles the

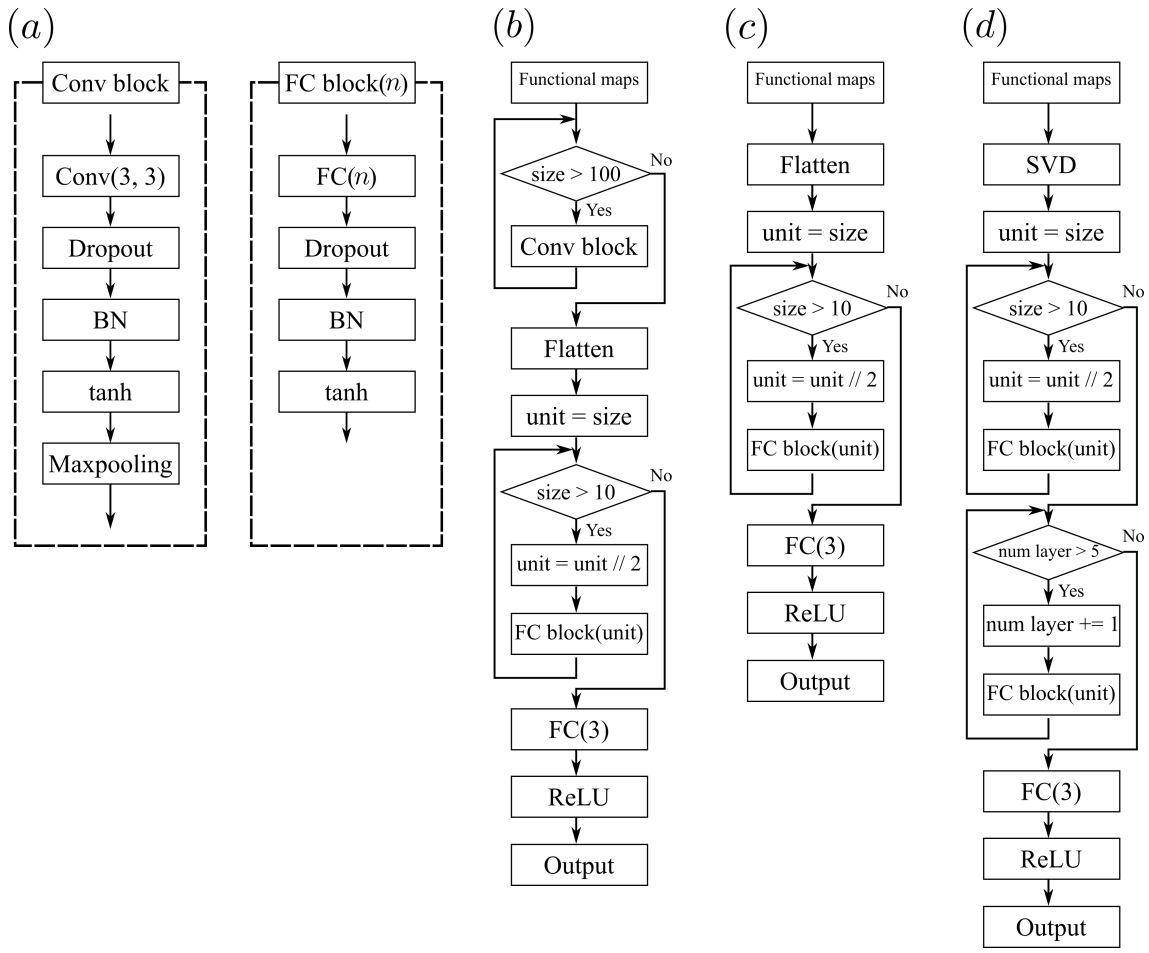

Figure 5.4: The algorithms to construct the machine learning models. (*a*): The blocks used in the models. The "Conv(3, 3)", "BN" and "FC(*n*)" in the blocks denote the convolution layer with 3×3 filter, batch normalization layer and fully-connected layer, which is a layer of the perceptrons, with *n* perceptrons. (*b*): The diagram of the CNN model. The "size" in the diagram represents the size of data, and "unit" is variable for "FC block". (*c*): The MLP model structure. In this mode, the functional maps fed into the model are flatten to input "FC block". (*d*): The schematics to construct the SVD model. The singular values of the functional maps are computed and input to "FC block". The "num layer" is the number of layers of the model.

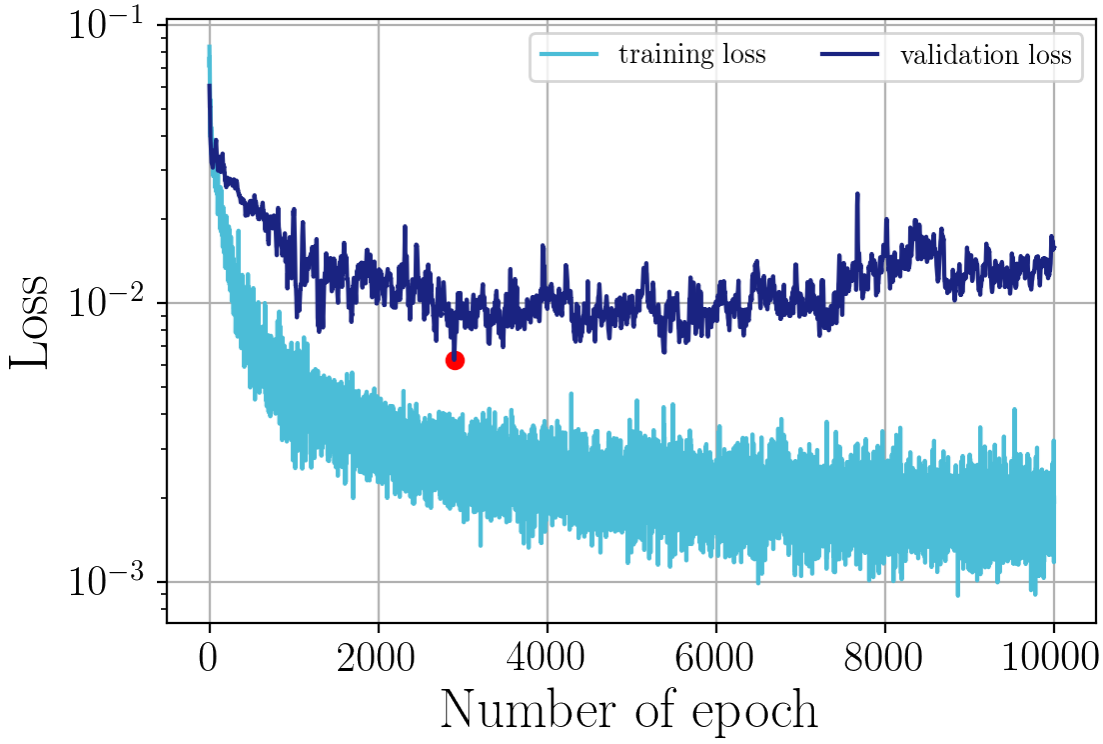

Figure 5.5: The example of the history of the training and validation loss of the CNN model with 20×20 functional maps.

functional maps directly. The input maps are fed forward and reduced through the "Conv block" while the data size become less than 100. Note that the "Conv block"s are not reused. Each time the data pass through the "Conv block" on the diagram, new "Conv block" is defined. This is also the case for the following "Conv block"s and "FC block"s. Once the data size is less than 100, the data are flatten and fed into the "FC block". The number of units of the "FC block" is divided by 2 each time it is fed into the "FC block". When the size of the data become less than 10, the data are input 3 units which correspond to the number of pathological parameters of fully connected layer with the ReLU function. The ReLU function is selected because the pathological parameters to be predicted have the range from 0 to 0.7. For the MLP model, the functional maps are flatten firstly. The flatten data are fed forward to the same structure as after the "CNN block"s in the CNN model. The SVD model also has the similar structure. The singular values of the maps are computed and fed into the "FC block"s up tp the data size become less than 10

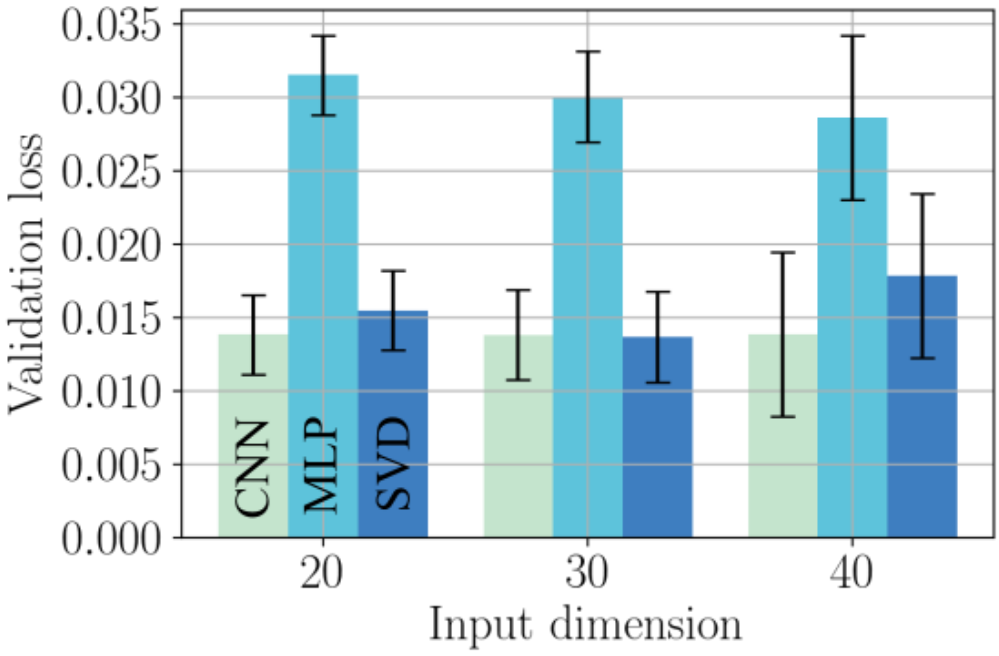

Figure 5.6: The dependencies on the size of the input functional maps and the model configuration. The fivefold cross-validation is performed. The error bar represents the standard deviation of the validation loss with respect to each fold.

as well as the MLP model. Not to be small model, the "FC block"s with the same number of perceptrons are added while the total number of layers becomes more than 5. As the output of the MLP model, the 3 units fully-connected layer with ReLU function is added. These three models are trained and evaluated.

The learning conditions for the machine learning models such as optimizer and loss function are also important for training the models properly. The mean squared error is used as the loss function ε, i.e.,

$$\varepsilon = \frac{1}{M}\sum_{i=1}^{M}\left(\boldsymbol{q}_i - \boldsymbol{q}_i^{\mathrm{pred}}\right)^2, \tag{5.1}$$

where $M, \boldsymbol{q}_i$, and $\boldsymbol{q}_i^{\mathrm{pred}}$ are the number of data which are included in the mini-batch, the ith pathological parameters and the predicted i th pathological parameters. The objective of regression tasks with supervised machine learning is to obtain optimized weights $\boldsymbol{W}$ by minimizing the loss function ε such that

$$\boldsymbol{W} = \mathrm{argmin}_{\boldsymbol{W}}\|\varepsilon\|. \tag{5.2}$$

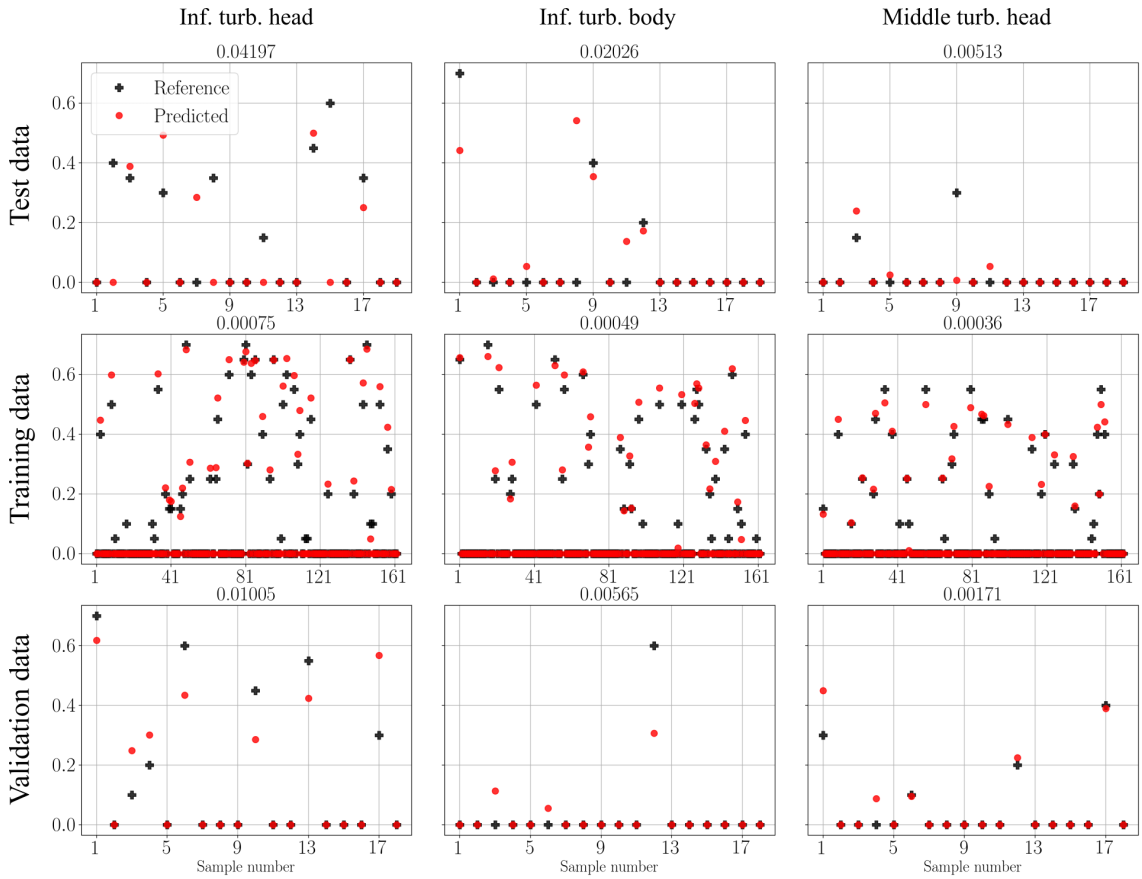

Figure 5.7: The predictions of the CNN model with the 20×20 functional maps. The column represents the pathological parameters and the row is the kind of data. The values above the scatter plots indicate the mean squared errors.

The adam algorithm (Kingma and Ba, 2014) is applied as the optimizer for weight updating. The number of the epochs is set as 10000, and the model with the best validation loss is saved to avoid overfitting. As an example, the history of the train and validation loss of the CNN model with input of 20×20 functional maps are shown in figure 5.5. For this figure, the best model highlighted with the red point is saved. The fivefold cross-validation is performed to evaluate the dependencies of prediction on the parameters, i.e., the input size and the model configuration. Note in passing that the input functional maps are normalized before feeding into the machine learning models since the preprocessing of the data is one of the well-known candidates to achieve an efficient training and to avoid an overfitting issue.

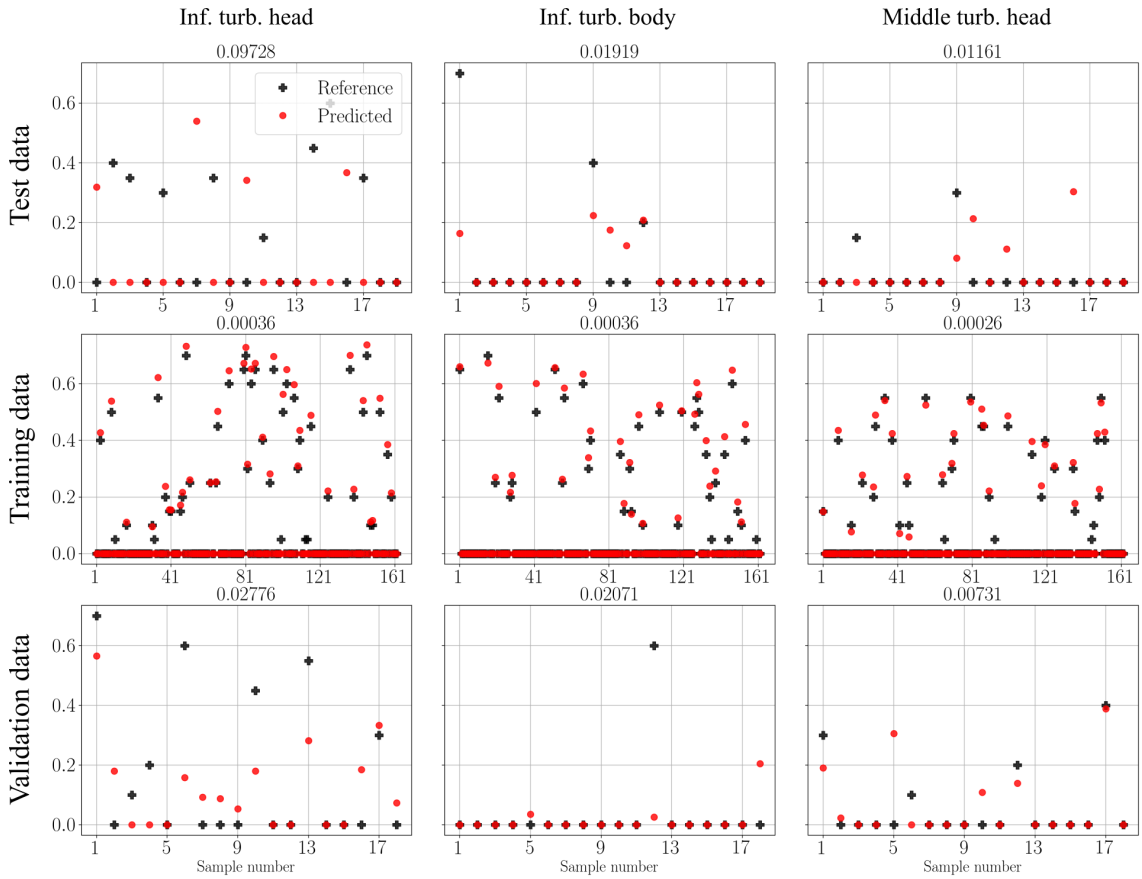

Figure 5.8: The predictions of the MLP model with the 20×20 functional maps. The column represents the pathological parameters and the row is the kind of data. The values above the scatter plots indicate the mean squared errors.

5.1.3 Results

As mentioned above, the dependencies of the model performance on the size of functional maps and the machine learning model are evaluated with the fivefold cross-validation. As test data, 10% of data, i.e., 19 maps excluded from the data, are used for evaluating the models. The best validation losses of each fold and parameter are averaged. These results are summarized in figure 5.6. The colors and horizontal axis represent the model configuration and the size of the input functional maps. The error bars denote the standard deviations with respect to each fold. The validation losses are almost same against input dimensions. This results imply that the features for prediction of the pathological parameters are included enough in the 20×20 functional maps. Based on this result, the 20×20 functional maps are used to evaluate the predictions in the following. As for the model

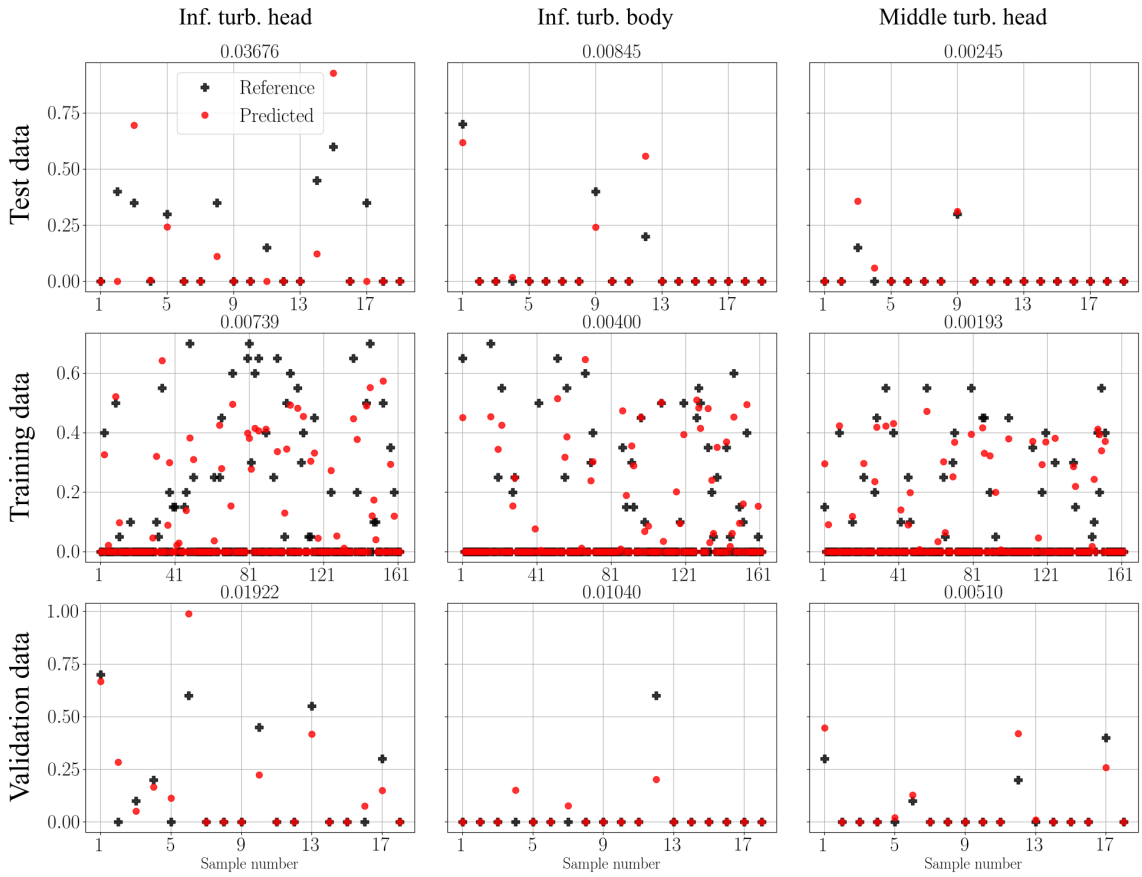

Figure 5.9: The predictions of the SVD model with the 20×20 functional maps. The column represents the pathological parameters and the row is the kind of data. The values above the scatter plots indicate the mean squared errors.

configuration, the MLP models show higher validation losses than that of the CNN and the SVD model. Hence, the MLP model is not suited for this regression problem.

The predicted results with the reference pathological parameters of the CNN, MLP, and SVD models with the 20×20 functional maps are summarized in figures 5.7, 5.8 and 5.9, respectively. The column and row represents the pathological parameters and the data type used for the assessment, i.e., the test, training and validation data, and the values above the scatter plots indicate the mean squared errors. Although what is the most important is the performance for the test data, the results of the training and validation data are presented to confirm whether the models are overfitted or not. These results show the similar trend with the cross-validation results. The CNN and SVD models show a good agreement with the reference pathological parameters at the same level. On the other

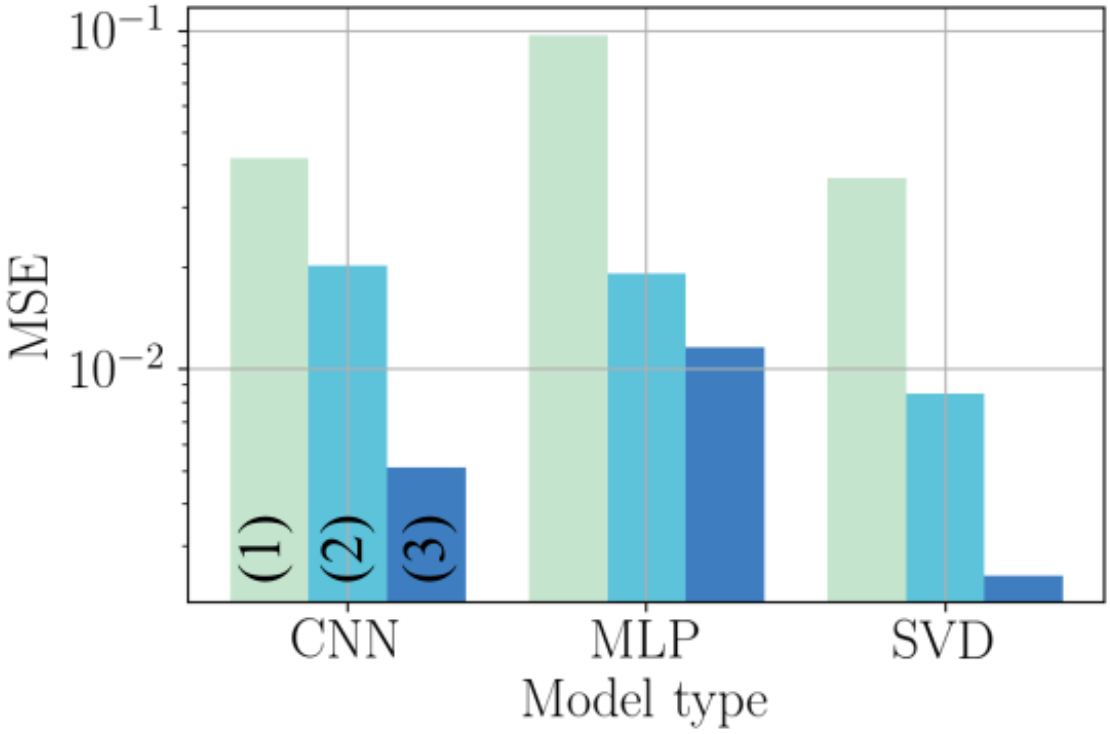

Figure 5.10: The mean squared errors of the test data against the pathological parameters and models. The (1), (2) and (3) denote the pathological parameters for the inferior turbinate head, inferior turbinate body and middle turbinate head, respectively.

hand, the mean squared errors of the MLP model are relatively high comparing with two other models. In addition, it can be seen that the MLP model is overfitted, as the results for the training data show reasonable agreement with the reference values, while they were not predictive at all for the test and validation data. This is likely because the number of weights in the MLP model is too large to retain the generalizability. It is known that the model with the large number of weights may induce overfitting (Fukami, Hasegawa, Nakamura, Morimoto and Fukagata, 2020). Especially in the present analysis, the MLP model has the larger amount of weights than that of the CNN and SVD models because the CNN model consists of the efficient filtering operations and the feature of the functional map is extracted as singular values for the SVD model. To assess the results with respect to each pathological parameter, the mean squared error of test data for each model and parameter are also shown in figure 5.10. The predictions of the middle turbinate head show the lowest mean squared errors for each model; otherwise, the parameter for the

inferior turbinate head is the most difficult in the prediction.

The accuracy of the diagnosis is, of course, an important metric. We applied the step function, i.e.,

$$\mathrm{step}(x) = \begin{cases} 1 & (x > 0) \\ 0 & (x \leq 0) \end{cases}, \tag{5.3}$$

to the output of the models to classify the healthy and pathological noses as 0 and 1, respectively. The accuracies of the predictions of the test data for the pathologies are summarized in table 5.1. The SVD model shows notably good results in the view point of accuracy. The accuracies of the SVD model for 2nd and 3rd parameters are 0.947, which is almost 1. Moreover, the confusion matrixes for each pathology and model are shown in figure 5.11. Each row of the matrix represents the instances in the reference class while each column represents the instances in the predicted class. It is quantitatively confirmed that the prediction of pathological nose is more difficult than that of healthy nose.

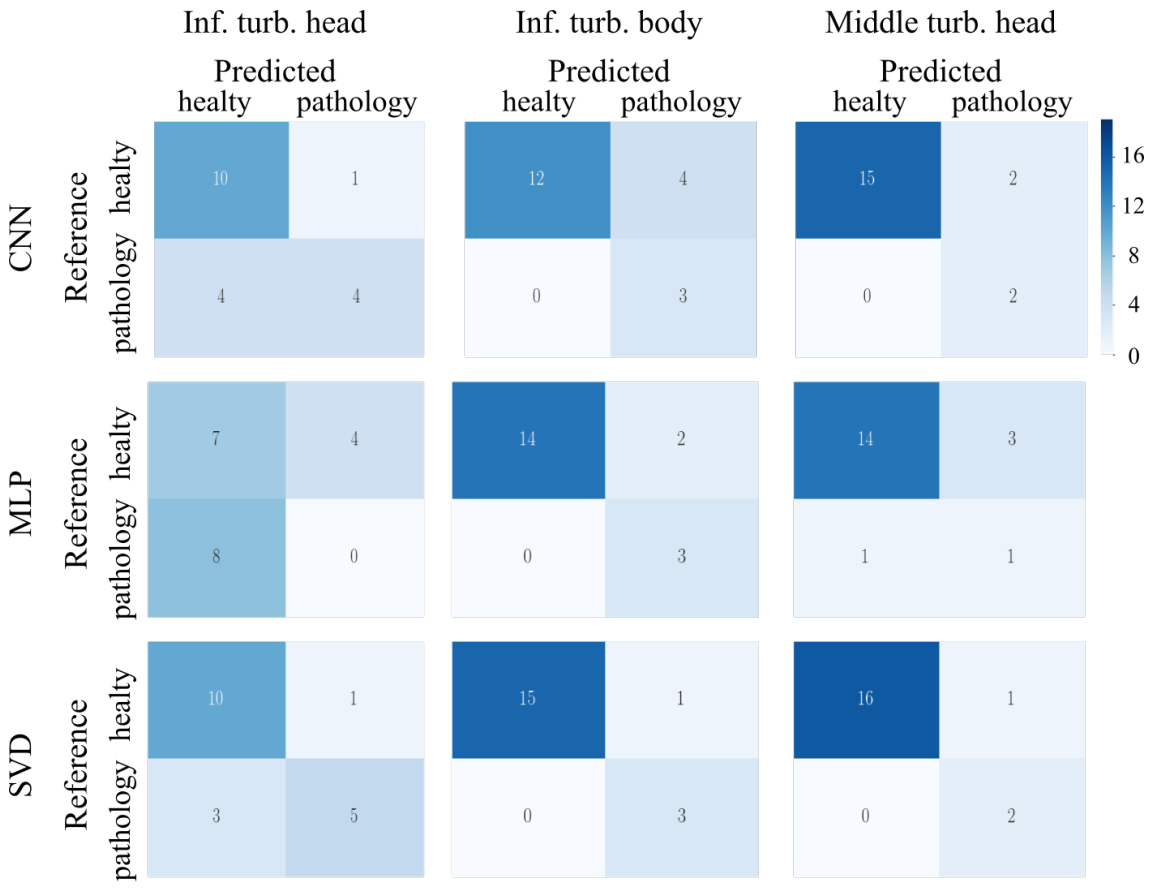

Figure 5.11: The confusion matrixes of the predictions for each pathology and model.

Table 5.1: The accuracies of the predictions.

	Inf. turb. head	Inf. turb. body	Middle turb. head
CNN	0.737	0.789	0.895
MLP	0.368	0.895	0.789
SVD	0.789	0.947	0.947

5.2 Prediction from flow field and geometry

In addition to the geometrical information of the noses, the pressures and wall shear stress (WSS) on the noses are used to train the machine learning model. The methods and results are presented in this section.

5.2.1 Data preparation

Using the flow fields for machine learning models directly is not easy because of the high dimensionality of the field. The number of points on the nose is too large to handle with the machine learning models. To use them for predicting the pathologies, the feature of the data are extracted as well as the previous model with the functional maps. The schematic for extraction of the feature from pressure field is summarized in figure 5.12. The pressure on a target nose p_t is mapped onto the reference nose by using the functional map. The pressure difference $\Delta\widehat{p_t}$ between the pressure on the reference nose p_r and the mapped pressure of target nose $\widehat{p_t}$ is computed on the reference nose as:

$$\Delta\widehat{p_t} = p_r - \widehat{p_t}. \tag{5.4}$$

Suppose that the pressure difference can be expressed by the linear combination of the eigenfunctions $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n\}$, i.e., the bases, of the reference nose. Using the coefficients for the eigenfunctions $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, \cdots, \gamma_n\}^T$, the pressure difference is described as

$$\Delta\widehat{p_t} \approx \boldsymbol{B}\boldsymbol{\gamma}. \tag{5.5}$$

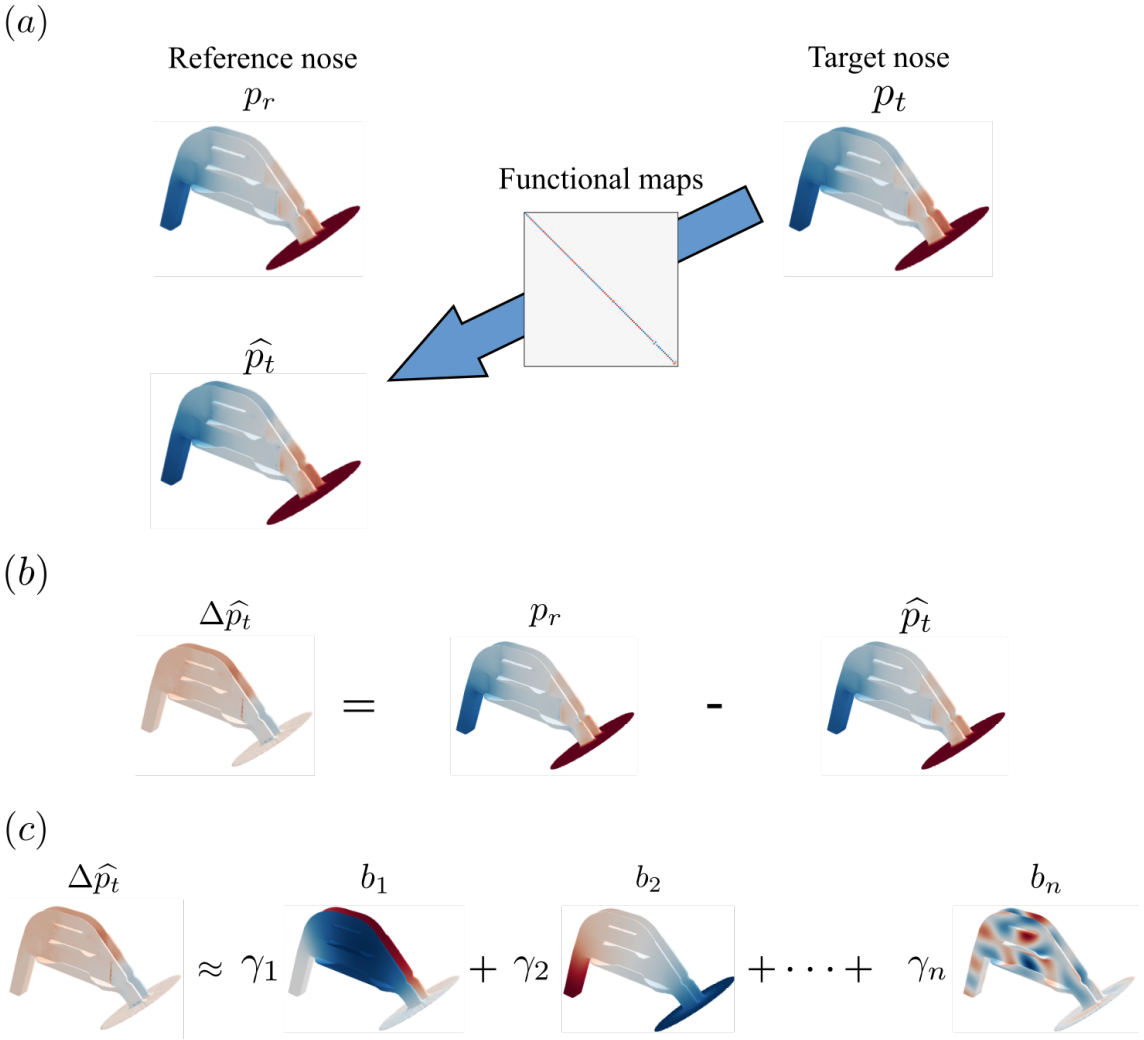

Figure 5.12: The schematic of the data preparation. (a): The pressure on the target nose p_t is mapped onto the reference nose as $\widehat{p_t}$. (b): The pressure difference $\Delta\widehat{p_t}$ between the pressure of the reference nose p_r and the mapped pressure $\widehat{p_t}$ is computed on the reference nose. (c): The pressure difference $\Delta\widehat{p_t}$ is assumed to be the linear combination of the eigenfunctions of the reference nose $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n\}$ and its coefficients $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, \cdots, \gamma_n\}^T$.

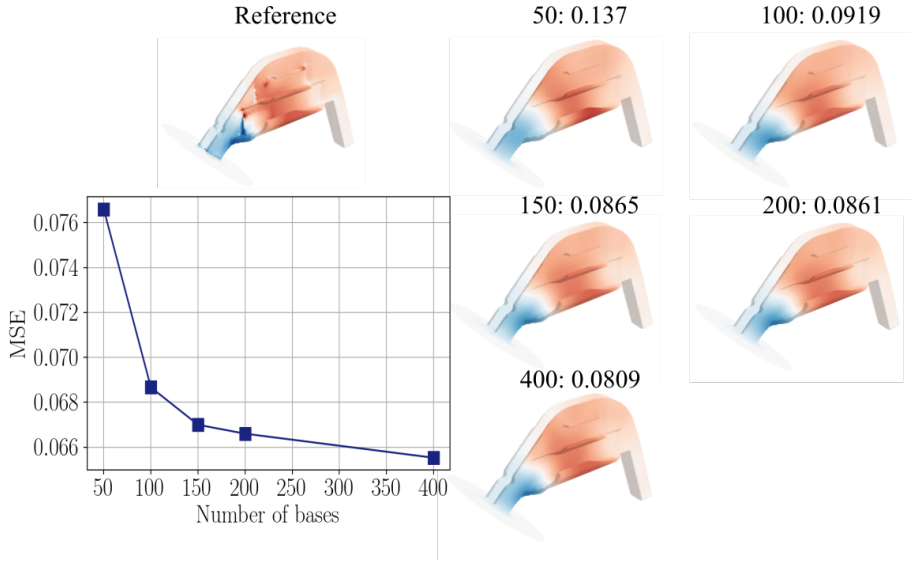

Figure 5.13: Dependence of reconstruction ability on the number of the bases used to reconstruct field. The chart represents the mean squared errors against the number of the bases, averaged also over all target noses. The color scale on the 3D plots of nose are the reference and reconstructed pressure difference fields of a target nose. The values above the 3D plots denote (the number of bases): (the mean squared error of the presented pressure difference).

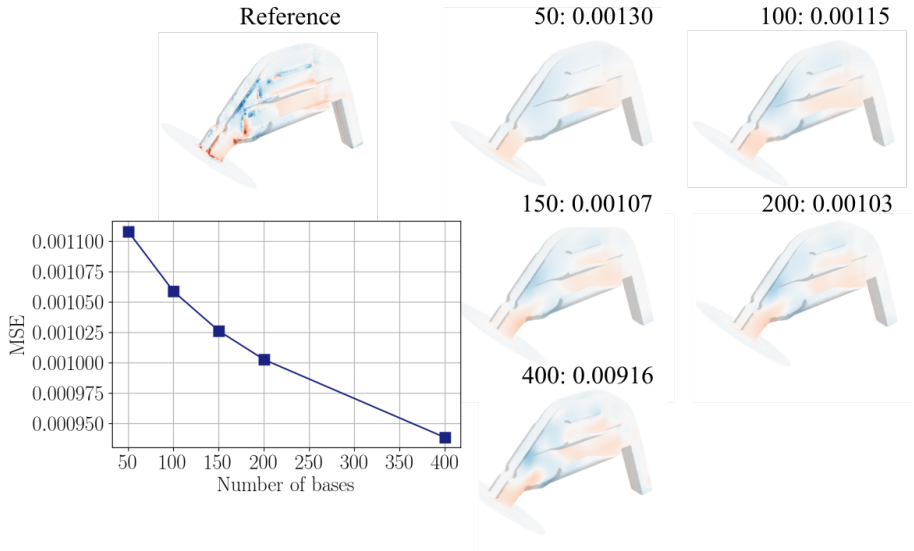

Figure 5.14: Dependence of reconstruction ability on the number of the bases used to reconstruct field. The chart represents the mean squared errors against the number of the bases, averaged also over all target noses. The color scale on the 3D plots of nose are the reference and reconstructed wall shear stress difference fields of a target nose. The values above the 3D plots denote (the number of bases): (the mean squared error of the WSS difference).

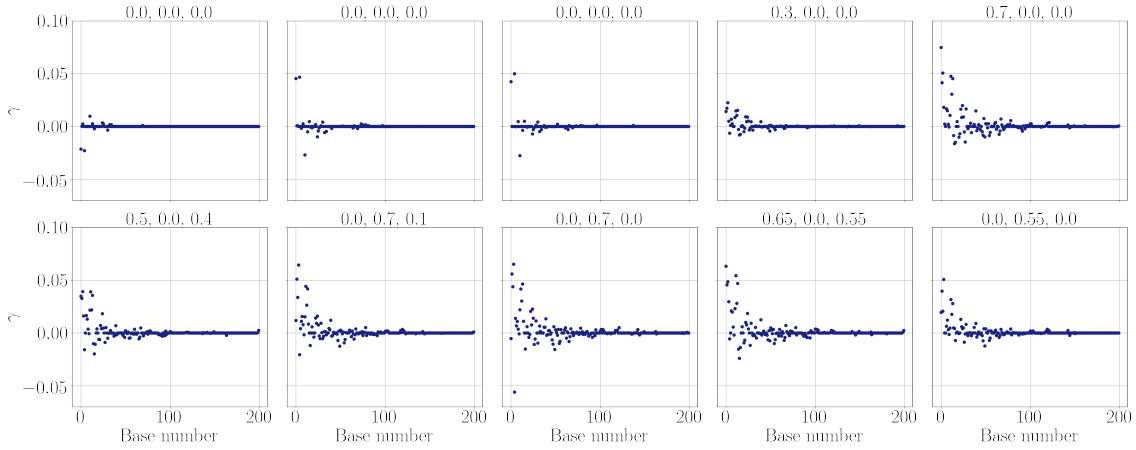

Figure 5.15: The examples of the coefficients γ of 200 bases for pressure. The values above the plots are the pathological parameters for inferior turbinate head, inferior turbinate body and middle turbinate head.

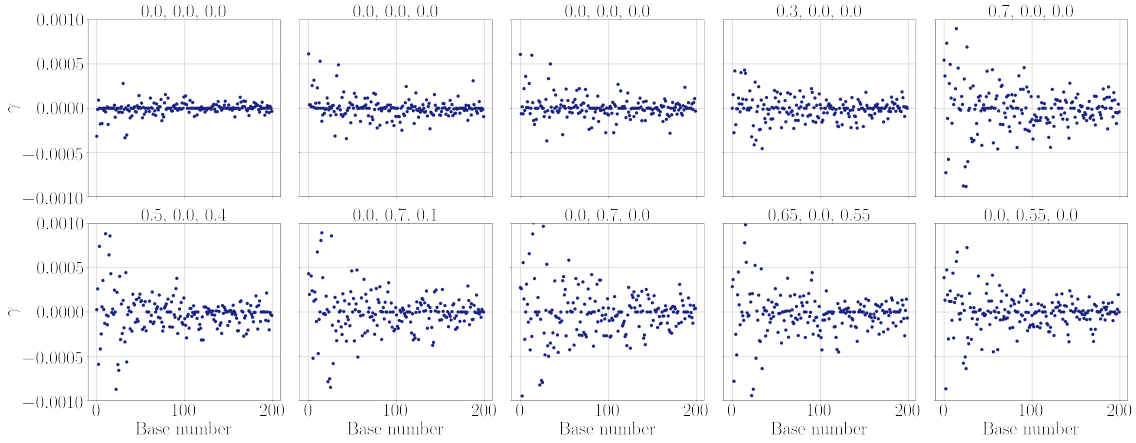

Figure 5.16: The examples of the coefficients γ of 200 bases for wall shear stress. The values above the plots are the pathological parameters for inferior turbinate head, inferior turbinate body and middle turbinate head.

Lasso (Tibshirani, 1996) is applied to solve this linear regression. The optimized γ which satisfies the equation 5.5 is

$$\boldsymbol{\gamma} = \mathrm{argmin}\left(\|\Delta\widehat{p_t} - \boldsymbol{B}\boldsymbol{\gamma}\|^2 + \alpha \sum_j |\gamma_j| \right), \tag{5.6}$$

where α is the sparsity constant. In this study, α is set to 0.1 to ensure the sparsity. This coefficients $\boldsymbol{\gamma}$ represent the amplitudes of the bases to reconstruct the pressure difference and are used as the input for the machine learning model. It is considered that the $\boldsymbol{\gamma}$ includes the pressure and geometrical information. This strategy is also applied to WSS

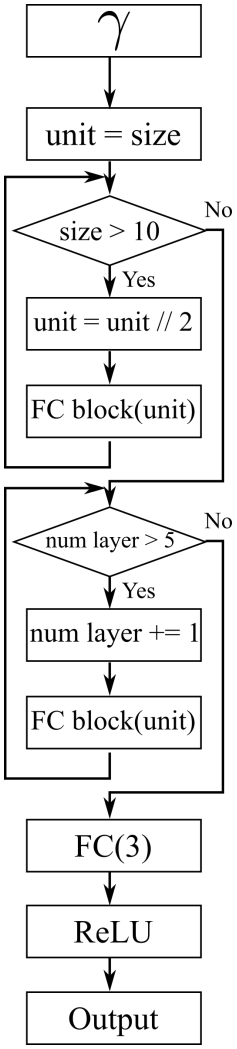

Figure 5.17: The model configuration for prediction the pathologies from the coefficients γ. The blocks used in this figure are defined in figure 5.4.

fields. Although the WSS is, of course, vector fields, the norm of WSS is used as a scalar function on the noses. The sparsity constant α for Lasso is set to 0.001, because the WSS fields are more complex and local than pressure fields.

The number of bases used to reconstruct the difference fields is one of the important parameters. As mentioned above, the higher bases include the small structure of the nose. To investigate the dependence of the reconstructed fields on the number of bases, the difference fields are reconstructed with $50, 100, 150, 200$ and 400 bases. The mean squared errors between the reference and the reconstructed fields for each target nose are averaged and shown in the chart of figures 5.13 and 5.14. Figures 5.13 and 5.14 are

respectively for pressure and WSS fields. The examples of reconstructed and reference fields are also shown as 3D heat maps in figures. The values above the 3D plots denote (the number of bases): (the mean squared error). Note that the mean squared error above the 3D plot is value for the presented nose in 3D plot. On the other hand, the mean squared errors in the chart are the averaged mean squared errors over all target noses. The reconstructed fields show reasonable agreement with the reference field qualitatively even for the field reconstructed by 50 bases. The mean squared error decreases with the increase in the number of bases used to reconstruct the field. This is, of course, because that using more bases brings a detailed field reconstruction. As the example, some obtained coefficients γ are shown in figures 5.15 and 5.16 for pressure and WSS. These γ are computed by using 200 bases. The values above the maps are the pathological parameters for inferior turbinate head, inferior turbinate body and middle turbinate head from the left. respectively. Note again that 0 of these values indicates a healthy state. The

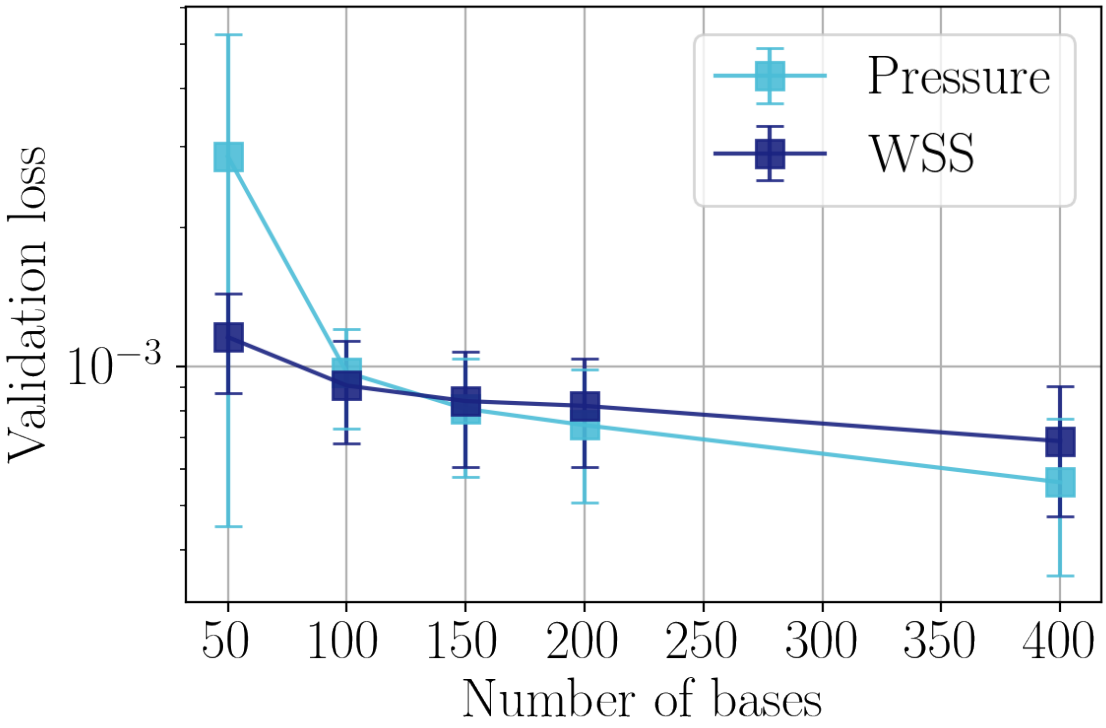

Figure 5.18: The dependency on the number of the bases. The fivefold cross-validation is performed. The error bar represents the standard deviation.

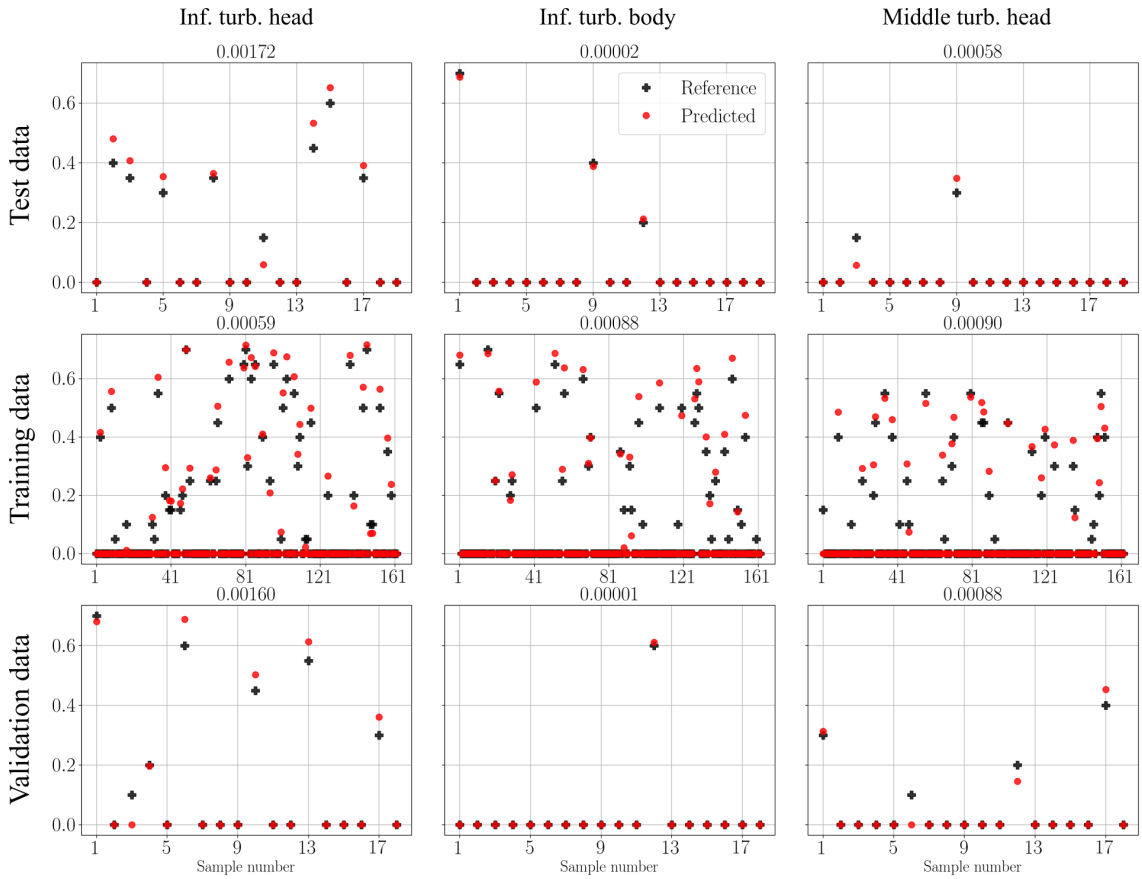

Figure 5.19: The predictions of the model trained by the coefficients γ of pressure. The column represents the pathological parameters and the row is the kind of data. The values above the scatter plots indicate the mean squared errors.

trend, that the coefficients γ for low-order bases have non zero values when the target nose has pathology, is confirmed. Note that the number of coefficients of WSS which is not 0 is larger than that of pressure because of the values of the sparsity constant. From this results, it is considered that the coefficients γ can extract the feature of the pathologies.

5.2.2 Model configuration and learning conditions

The model configuration for predicting the pathologies from the coefficients γ is presented. The block diagram to construct the model is illustrated in figure 5.17. The coefficients γ are input to the “FC block”, which is defined in figure 5.4, consisted of units with half size of input. The “FC block” with the half size of units is added up to the size of the data becomes less than 10. If the number of the layers is less than 5, the ”FC

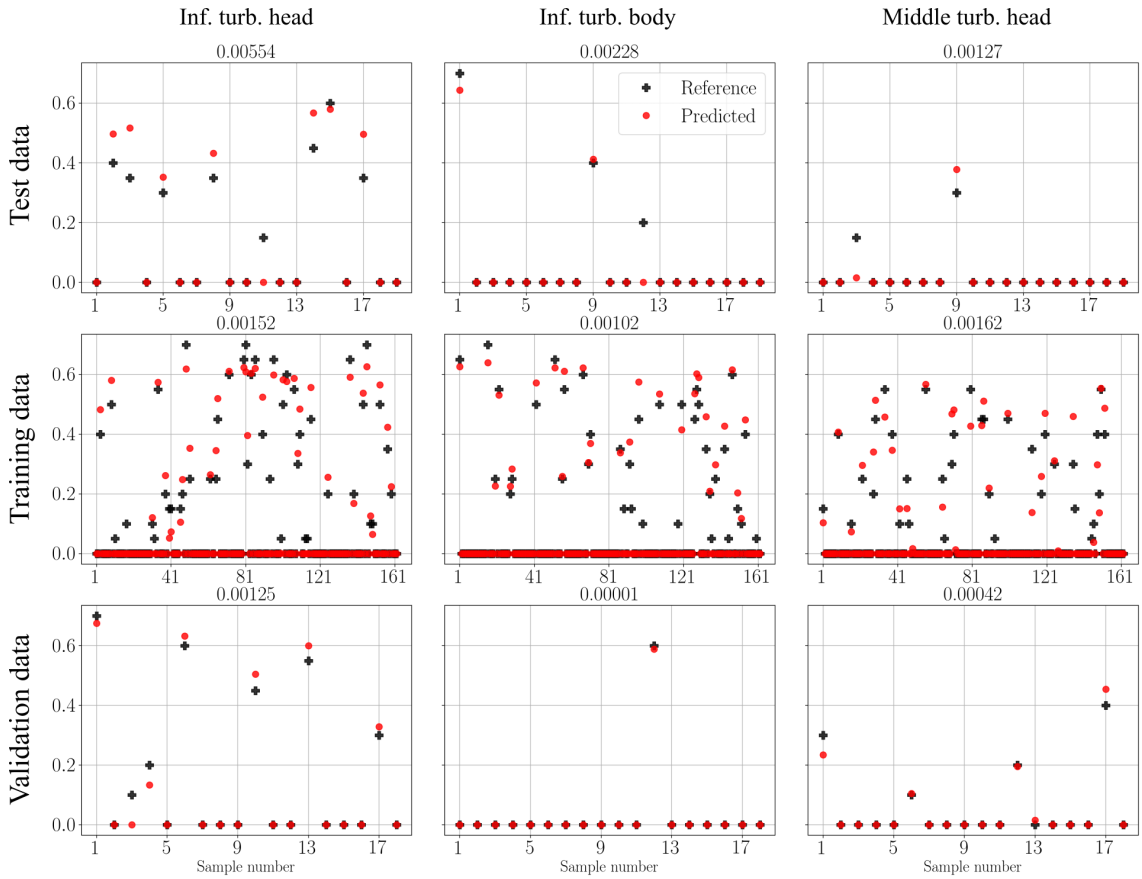

Figure 5.20: The predictions of the model trained by the coefficients γ of wall shear stress. The column represents the pathological parameters and the row is the kind of data. The values above the scatter plots indicate the mean squared errors.

block" with same size of units is added while it satisfies (the number of layers) > 5. The fully-connected layer consisted of 3 units with ReLU function is applied as output layer of the models.

For the learning condition of the models, the mean squared error is used as the loss function. To update the weights in the models, the adam algorithm is applied. The number of epochs is set as 10000 and the best validation model is saved to prevent overfitting.

5.2.3 Results

The dependence on the number of bases used to reconstruct the pressure difference fields are evaluated by fivefold cross-validation, excluding 10% of data are as test data. Note that the coefficients γ are computed for each case. Therefore, the coefficients of the

Table 5.2: Accuracy.

	Inf. turb. head	Inf. turb. body	Middle turb. head
Pressure	1.00	1.00	1.00
WSS	0.947	0.947	1.00

1st base are not same for all cases. The averaged validation losses are shown in figure 5.18. The error bars in the figure represent the standard deviations. The validation loss decreases with the increase in the number of bases. The more numbers of bases bring more informative inputs and the number of weights of the model are increased with the number of bases. From these observations, we find that the validation loss decreases with the number of bases. The evaluated predictions in the following are the output of the model trained by 150 bases.

The reference values and predicted pathologies, which are by the model trained using 150 coefficients, of the test, training, and validation losses are summarized in figures 5.19 and 5.20 for pressure and WSS. The predicted values show reasonable agreements with the reference values not only for the training data but also for the test and validation data.

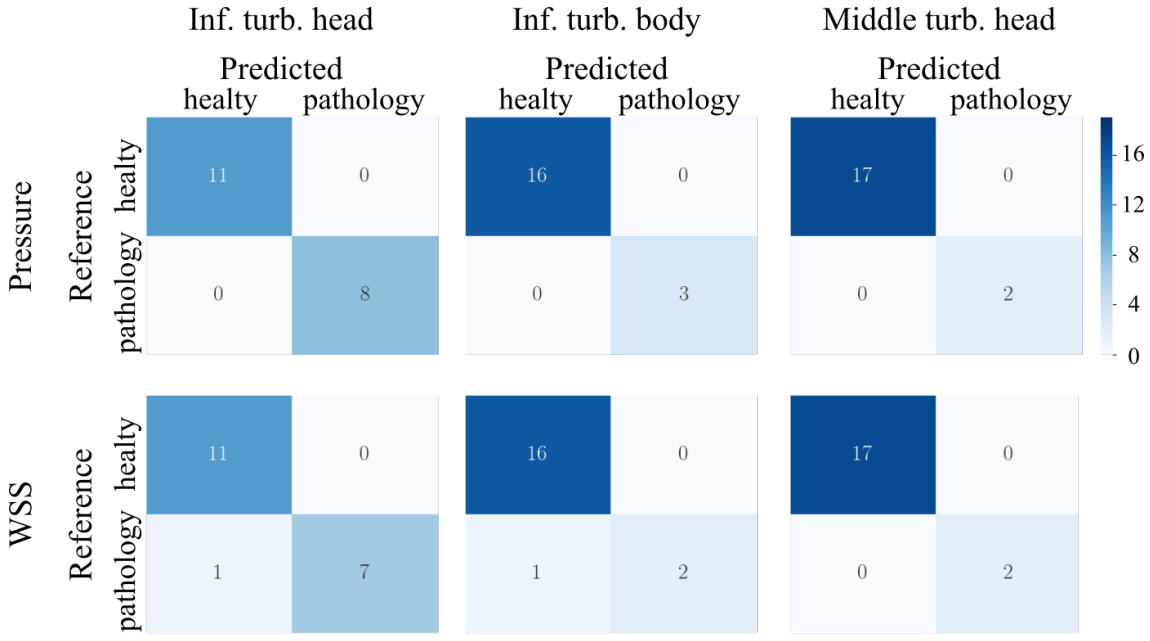

Figure 5.21: The confusion matrixes of the predictions for each pathology by pressure and wall shear stress.

This indicates that the model obtain the generalized ability to predict the pathological parameters from the coefficients γ. The accuracies of the test prediction are computed for each pathological parameters in the same way with the predictions from functional maps, as summarized in table 5.2. The predictions with the pressure field completely match with the reference data and the accuracies are 1.00 for all pathological parameters. Moreover, the predictions by WSS also show high accuracies. Of course, the confusion matrixes shown in figure 5.21 also show perfect results. Based on these results, the pressure and WSS information are important and effective in predicting the pathologies or diagnosing.

5.3 Discussion

The predictions by the functional maps and the characteristics produced by the geometry and the flow fields are presented above. In this section, let us compare the results of them. As described above, the prediction from the geometry and the flow field shows the better results than the prediction from the geometry information alone. This indicates that flow field information is useful in predicting pathologies. This is likely because the flow field is sensitive for a small changes in the geometry of nose. A local change in a geometry spread out as a global change in the flow field. For instance, the small variations of pathological parameters induce the flow field that a patient has a difficulty breathing. Consequently, the difference in flow field more apparent than the change in the geometry. Since it is intuitively true that the geometrical changes, i.e., pathological parameters, can be predicted from the geometry itself, it is, of course, expected that the prediction with the geometry alone can be improved as much as the one with the flow and geometry if enough information of the geometry is gained. However, the inference with geometry alone is difficult because the change to be predicted is so small that requires detailed structures of the nose, and detailed information is sometimes redundant for machine learning models. On the other hand, the prediction from the flow and geometry is easier than geometry alone because a local change in geometry becomes a global change in the flow fields. From these reasons, the predictions of the pathologies are improved by adding the flow

field information.

Chapter 6

Conclusions

In this study, the pathological parameters of noses were predicted by machine learning models. The nasal cavity model and its CFD simulation presented by Romani (2017) were used. The geometrical features of the noses were extracted as functional maps and were fed into the machine learning models to predict three pathological parameters, i.e., inferior turbinate head, inferior turbinate body, and middle turbinate head. Three machine learning models, i.e., CNN, MLP, and SVD models, were proposed and evaluated. The dependence of the ability to predict the pathologies on the size of the functional maps was also investigated. The predictions of the CNN and SVD models exhibited reasonable agreements with reference pathological parameters. On the contrary, the MLP model was in overfitting. The flow information, i.e., pressure and wall shear stress, were added to the input of machine learning model. To extract features which include the flow and geometrical information, the difference of the flow quantities on the reference healthy nose was computed by mapping the quantities on target nose through functional map. The difference field was assumed to be a linear combination of eigenfunctions of the Laplace-Beltrami operator of the reference nose. The optimized coefficients of the eigenfunctions were obtained by solving the linear problem using Lasso. These coefficients were used to predict the pathologies by the machine learning models. Moreover, the dependence on the number of the eigenfunctions were evaluated. The accuracies of the predictions presented almost 1.0 for all pathologies and were improved from the results of the model with the functional maps. These results conclude that the flow information are efficient

for predicting the pathologies.

The idea of using machine learning to predict the nasal pathologies was based on the assumption that CFD can help get better predictions. As a preliminary test, we examined the idea with limited conditions: the number of noses, low-fidelity CFD, simplified nose model, low-fidelity anatomy, and manufactured pathologies. Although the machine learning models were constructed under the conditions, the results supported the validity of the aforementioned assumption. This suggests the possibility of the application of the proposed method to practical noses.

Reference

Brenner, M. P., Eldredge, J. D. and Freund, J. B. (2019). Perspective on machine learning for advancing fluid mechanics, *Phys. Rev. Fluids* **4**: 100501.

Brunton, S. L. and Kutz, J. N. (2019). *Data-driven science and engineering: Machine learning, dynamical systems, and control*, Cambridge University Press.

Brunton, S. L., Noack, B. R. and Koumoutsakos, P. (2020). Machine learning for fluid mechanics, *Annu. Rev. Fluid Mech.* **52**: 477–508.

Cai, S., Zhou, S., Xu, C. and Gao, Q. (2019). Dense motion estimation of particle images via a convolutional neural network, *Experiments in Fluids* **60**: 73.

Carrillo, M., Que, U. and Gonzalez, J. A. (2016). Estimation of reynolds number for flows around cylinders with lattice boltzmann methods and artificial neural networks, *Phys. Rev. E* **94**: 063304.

Chen, X. B., Lee, H. P., Chong, V. F. and Wang, D. Y. (2010). Impact of inferior turbinate hypertrophy on the aerodynamic pattern and physiological functions of the turbulent airflow - a cfd simulation model., *Rhinology.* **48**(2): 163–168.

Covello, V., Pipolo, C., Saibene, A., Felisati, G. and Quadrio, M. (2018). Numerical simulation of thermal water delivery in the human nasal cavity, *Comput. Biol. Med.* **100**: 62–73.

Duchi, J., Hazan, E. and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization, *J. Mach. Learn. Res.* **12**: 2121–2159.

Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W. and Kutz, J. N. (2020). Shallow learning for fluid flow reconstruction with limited sensors, *Proc. R. Soc. A* **476**(2238): 20200097.

Fukami, K., Fukagata, K. and Taira, K. (2019). Super-resolution reconstruction of turbulent flows with machine learning, *J. Fluid Mech.* **870**: 106–120.

Fukami, K., Fukagata, K. and Taira, K. (2020). Assessment of supervised machine learning methods for fluid flows, *Theor. Comput. Fluid Dyn.* https://doi.org/10.1007/s00162-020-00518-y .

Fukami, K., Fukagata, K. and Taira, K. (2021). Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows, *J. Fluid Mech.* **909**: A9.

Fukami, K., Hasegawa, K., Nakamura, T., Morimoto, M. and Fukagata, K. (2020). Model order reduction with neural networks: Application to laminar and turbulent flows, arXiv:2011.10277 .

Fukami, K., Maulik, R., Ramachandra, N., Fukagata, K. and Taira, K. (2021). Global field reconstruction from sparse sensors with voronoi tessellation-assisted deep learning, arXiv:2101.00554 .

Fukami, K., Nabae, Y., Kawai, K. and Fukagata, K. (2019). Synthetic turbulent inflow generator using machine learning, *Phys. Rev. Fluids* **4**: 064603.

Glorot, X., Bordes, A. and Bengio, Y. (2011). Deep sparse rectifier neural networks, *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323.

Hasegawa, K., Fukami, K., Murata, T. and Fukagata, K. (2020a). CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different reynolds numbers, *Fluid Dyn. Res.* https://doi.org/10.1088/1873-7005/abb91d **52**(6): 065501.

Hasegawa, K., Fukami, K., Murata, T. and Fukagata, K. (2020b). Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes, *Theor. Comput. Fluid Dyn.* https://doi.org/10.1007/s00162-020-00518-y .

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory, *Neural Comput.* **9**: 1735–1780.

Illum, P. (1997). Septoplasty and compensatory inferior turbinate hypertrophy: long-term results after randomized turbinoplasty, *Eur Arch Otorhinolaryngol* **254**(S89–S92).

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37 of *Proceedings of Machine Learning Research*, PMLR, Lille, France, pp. 448–456.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization, arXiv:1412.6980.

Kutz, J. N. (2017). Deep learning in fluid dynamics, *J. Fluid Mech.* **814**: 1–4.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning, *Nature* **521**: 436–444.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition, *Proc. IEEE* **86**(11): 2278–2324.

Melzi, S., Ren, J., Rodolà, E., Sharma, A., Wonka, P. and Ovsjanikov, M. (2019). Zoomout: Spectral upsampling for efficient shape correspondence, *ACM Trans. Graph.* **38**(6).

Menter, F. R. (1994). Two-equation eddy-viscosity turbulence models for engineering applications, *AIAA Journal* **32**(8): 1598–1605.

Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A. and Guibas, L. (2012). Functional maps: A flexible representation of maps between shapes, *ACM Trans. Graph.* **31**(4).

Ovsjanikov, M., Corman, E., Bronstein, M., Rodol à , E., Ben-Chen, M., Guibas, L., Chazal, F. and Bronstein, A. (2017). Computing and processing correspondences with functional maps, *In ACM SIGGRAPH 2017 Courses* **Article 5**: 5:1–5:62.

Quadrio, M., Pipolo, C., Corti, S., Lenzi, R., Messina, F., Pesci, C. and Felisati, G. (2014). Review of computational fluid dynamics in the assessment of nasal air flow and analysis of its limitations., *Eur Arch Otorhinolaryngol.* **271**(9): 2349–2354.

Romani, G. (2017). *Machine learning techniques for evaluating nasal airflow: preliminary results*, Master's thesis, Politecnico di Milano.

Saibene, A., M., Felisati, G., Pipolo, C., Bulfamante, A., M., Quadrio, M. and Covello, V. (2020). Partial preservation of the inferior turbinate in endoscopic medial maxillectomy: A computational fluid dynamics study, *Am J Rhinol Allergy* **32**(3): 409–416.

Schillaci, A., Quadrio, M., Pipolo, C., Restelli, M. and Boracchi, G. (2021). Inferring functional properties from fluid dynamics features, *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 4091–4098.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* **15**(56): 1929–1958.

Sundh, C. and Sunnergren, O. (2015). Long-term symptom relief after septoplasty, *Eur Arch Otorhinolaryngol* **272**(2871–2875).

Taira, K., Hemati, M. S., Brunton, S. L., Sun, Y., Duraisamy, K., Bagheri, S., Dawson, S. and Yeh, C.-A. (2020). Modal analysis of fluid flows: Applications and outlook, *AIAA J.* **58**(3): 998–1022.

The Respiratory System. (n.d.). `https://www.therespiratorysystem.com/nasal-cavity/`.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso, *J. R. Stat. Soc. Ser. B-Stat. Methodol.* **58**(1): 267–288.

Zhao, K., Scherer, P. W., Hajiloo, S. A. and Dalton, P. (2004). Effect of anatomy on human nasal air flow and odorant transport patterns: implications for olfaction., *Chem Senses.* **29**: 365–379.

Zou, H. (2006). The adaptive lasso and its oracle properties, *J. Am. Stat. Assoc.* **101**(476): 1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net, *J. R. Stat. Soc. Ser. B-Stat. Methodol.* **67**(2): 301–320.