DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Intelligent Ball Screw Fault Diagnosis Using Deep Learning Based Domain Adaptation and Transfer Learning

Fabian Kolb

Inventory Number: 63451

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Intelligent Ball Screw Fault Diagnosis Using Deep Learning Based Domain Adaptation and Transfer Learning

Intelligente Fehlerdiagnose von Kugelgewinden mittels Deep Learning, basierend auf Domain Adaptation und Transfer Learning

Author:	Fabian Kolb
Supervisor:	Zäh Michael; Prof. Dr.-Ing.
Advisor:	Benker Maximilian; M.Sc.
Submission Date:	15.Sept.2022

I confirm that this master's thesis in informatics: robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, 15.Sept.2022 Fabian Kolb

Abstract

In order to remain competitive in the ongoing globalization, companies are forced to optimize their productions and processes. The immense amount of data, as well as modern information and communication technologies offer great opportunities for modern prognostics and health management. By analyzing machine and production data, machines can be proactively maintained, increasing the quality standards and efficiency of production lines. Since ball screw feed drives are widely used components in industrial machines, their degradation monitoring is especially important. Due to varying working conditions, prognostics and health management systems must be developed robust enough to handle continuous changes in the fault characteristics of industrial machines. Modern domain adaptation approaches have recently become more popular in prognostics and health management systems to counteract the corresponding domain shift in the machine data. This thesis proposes a deep learning-based domain adaptation model, which predicts the health condition of ball screw feed drives. Throughout the proposed model training, the domain shift in the latent feature spaces of the model is measured and reduced by the maximum mean discrepancy. By applying the maximum mean discrepancy in the CNN layers, the domain shift is reduced more efficiently. The effects of different hyperparameter choices in the domain adaptation module are evaluated. A novel labeled maximum mean discrepancy metric is presented, which uses a small portion of the training target labels.

Kurzfassung

Um in der fortschreitenden Globalisierung wettbewerbsfähig zu bleiben, sind Unternehmen gezwungen, ihre Produktion und Prozesse zu optimieren. Die dabei generierten Datenmengen sowie moderne Informations- und Kommunikationstechniken bieten große Chancen für moderne Prognostics and Health Management Systeme. Durch die Analyse von Maschinen- und Produktionsdaten können Maschinen proaktiv gewartet und dadurch die Qualitätsstandards und Effizienz von Produktionslinien erhöht werden. Da Kugelgewindetriebe häufig in Industriemaschinen verbaut sind, ist deren Zustandsüberwachung besonders wichtig. Aufgrund sich ändernder Arbeitsbedingungen müssen Prognostics and Health Management Systeme entwickelt werden, welche robust genug sind, um mit kontinuierlichen Veränderungen in den Fehlercharakteristiken der Industriemaschinen umzugehen. Um dem damit einhergehenden Domain Shift entgegenzuwirken, wird der Einsatz von Domain Adaptation Ansätzen, welche ursprünglich aus dem Bereich Maschinelles Sehen kommen, in Prognostics and Health Management Systemen immer beliebter. In dieser Arbeit wird ein Deep Learning basiertes Domain Adaptation System vorgestellt, welches den Verschleißzustand von Kugelgewindetrieben vorhersagt. In dem erarbeiteten Modelltraining wird der Domain Shift in den Schichten des Modells durch die Maximum Mean Discrepancy gemessen und reduziert. Durch den Einsatz der Maximum Mean Discrepancy in CNN Schichten, wird der Domain Shift effizient reduziert. Die Effekte verschiedener Hyperparameter im Domain Adaptation Modul werden evaluiert. Eine neuartige Labeled Maximum Mean Discrepancy Metrik wird vorgestellt, welche zu einem kleinen Teil Target Labels verwendet.

Contents

List of Abbreviation

CNN Convolutional Neural Networks

FC Fully Connected

MMD Maximum Mean Discrepancy

PHM Prognostics and Health Management

BSD Ball Screw Feed Drive

LGS Linear Guiding Shoes

AVG Average

STD Standard Deviation

1. Introduction

Modern sensors, sensor networks, computing systems and data-driven techniques have revolutionized the industry. As a key role in modern production facilities, prognostics and health management (PHM) applications have accelerated the big data revolution [1]. By analyzing production and machine data, the degradation status and risk of failure can be predicted for machines. This is especially helpful to increase the production quality and to reduce the downtime in production facilities [2]. Ball screw feed drives (BSDs) are widely used in industrial machines since they transfer rotary motion to linear motion with high precision [3]. Generally, BSDs are highly relevant for the entire machine's precision, making its health condition monitoring especially important [3]. Surveys showed that BSDs are related in up to 19% of all machine tool failures [2]. However, the research about modern health condition monitoring of BSDs is still in its early stages [3]. This thesis evaluates modern PHM systems to predict the degradation of BSDs.

1.1. Relevance and Problems of Prognostics and Health Management of Ball Screw Feed Drives

BSDs represent an essential part of various industrial machines. Like all mechanical components that involve friction, BSDs also wear off. Especially the steel balls and screw shaft are significantly affected by degradation [4]. In order to guarantee high motion repeatability while maximizing the expected life, the preload level of BSDs is calibrated precisely [4]. The degradation of the BSD's components lowers the preload and therefore reduces the stiffness in the BSD system. This deteriorates the positioning accuracy and leads to a lower production quality or even a total machine failure [4]. For this reason, health condition monitoring of BSDs is of great interest to the industry [4]. The complex motion trajectory of the steel balls and the difficult installation of sensors make the health condition monitoring of BSDs challenging [3].

Due to the increasing computational power and amount of data, deep learning is considered a powerful and efficient method for extracting information from large datasets. Therefore, deep learning models can learn the mapping between machine data and machine health condition classes [1]. Unfortunately, many developed PHM models assume the training

and testing dataset to have the same data distribution [5]. In reality, this is not the case. If machines operate over long periods of time, their operational conditions change. This leads to changing fault characteristics in the machine data. Consequently, when applying PHM systems in real industrial scenarios, the systems' performance becomes unsatisfactory over time [5]. The created domain shift between the training and testing data can be compensated by domain adaptation and transfer learning approaches, which measure and reduce the domain discrepancy between the two datasets while learning to solve the classification task [5].

1.2. Traditional and Deep Learning Based Prognostics and Health Management Approaches

PHM systems are traditionally restricted to physical-based and conventional data-driven approaches [1]. Nowadays, deep learning has become more popular for extracting health condition information from machine data [1]. Physical-based models explain the underlying complexity and degradation of machines with physical laws [1]. These models are especially advantageous since they do not require historical fault data to make predictions [6]. If the physics projected on the data does not consider all relevant machine aspects, including noise and perturbation, then the performance of such approaches is reduced [1]. In conventional data-driven approaches, hand-crafted features are extracted from the machine data to retrieve expressive information about the machine's health condition. The features are rated by their suitability for the PHM task and the most promising ones are selected. Subsequently, a shallow classifier predicts the corresponding health condition of the machine [1]. The physical-based and conventional data-driven approaches suffer from several problems. Firstly, establishing physical-based models or conventional hand-crafted features in complex real-world scenarios is a laborious task and requires much experience [1]. Secondly, an online model update is difficult [1]. Thirdly, the physical-based and conventional data-driven approaches are restricted to the specifications made about the monitoring task beforehand [1]. The limited transferability, flexibility and adaptability of these more traditional approaches are the reason for the growing interest in deep learning based PHM. Deep neural networks can capture relations within complex and high-dimensional data. Automatic learning makes neural networks easily adjustable to various problems [1].

2. Theory

2.1. Ball Screw Feed Drive

As shown in figure 2.1, ball screw feed drives (BSDs) consist of steel balls, seals, a screw shaft, nut and tube [7]. The steel balls serve as ball bearing between the screw shaft and nut [7]. The screw shaft is mounted by a fixed and free bearing and actuated by a motor [8]. The screw nut, which carries the load, moves linearly along the screw shaft, which is rotated by the motor [7]. Linear guiding shoes (LGSs) are installed to direct the moving components [8]. While the steel balls are rotated under external load, the ball screw feed drive shaft is under constant compression [7]. Due to the rolling friction, defects usually occur in the grooves of the screw shaft, which guide the steel balls [7]. Defects usually start with minor abrasion on the surface. Each time the steel balls pass the surface defects, the system repetitively takes shocks. Depending on the location and severity of the defects, the periodicity and the intensity of the shock vary [7]. For this reason, vibration signals contain expressive information about the machine's health condition [7]. Additionally, the surface defects also lower the efficiency of the system. In order to move the load with the same speed and acceleration, the motor torque needs to be increased. Therefore, the motor torque and current signals are also worth to be investigated in the PHM context. These signals can be retrieved from the machine's controller [5].

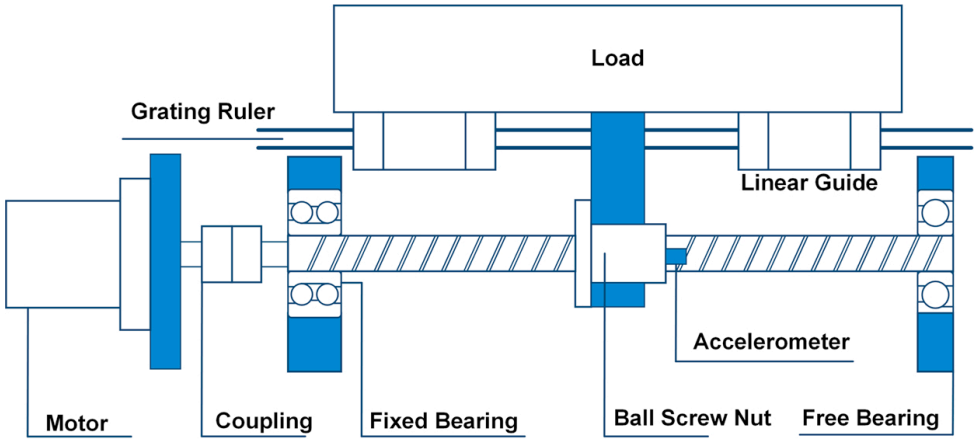

Figure 2.1.: Ball screw feed drive [8]

2.2. Neural Network

The big data ecosystem is continuously evolving and new technologies are coming up constantly. Many of them react progressively to the demands of the industry. Big data refers to an increase of unstructured data, high sampling rates and a variety of different data sources [9]. Machine learning tries to solve data-related problems by learning to extract expressive and informative features from the data. Deep learning is a specific branch of machine learning, which is inspired by the function of the human brain [10]. The increased data and computational power make deep learning applications more appealing for real-world use [1]. By using multiple layers, neural networks progressively extract features with different levels of abstraction [1]. In the following, different aspects of deep learning are explained in more detail. The different components of deep learning architectures and the optimization of such models are presented.

2.2.1. Neural Network Architecture

Neural networks consist of neurons that are layered in a hierarchical architecture. The neurons of consecutive layers are connected through weights and biases. During the model optimization, the weights and biases are updated [11]. Figure 2.2 shows the organization of neurons in a fully-connected network architecture. Each neuron from a layer is connected with all neurons from its subsequent layer and shares information with each other.

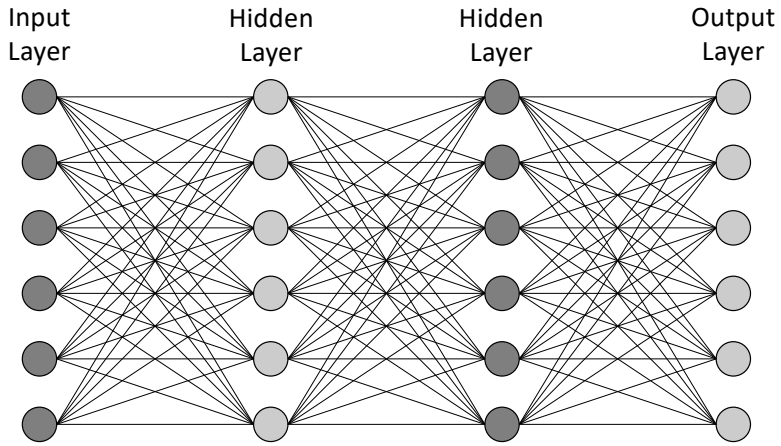

Figure 2.2.: Model architecture of fully-connected neural networks

The input of a neuron is calculated in two steps. Firstly, the weighted sum of all previous neurons and a bias are estimated. Afterward, an activation function is applied to the results, which gives the neural network a non-linear property [11]. Standard multilayer feedforward

networks with even one single hidden layer and an arbitrary bounded and non-constant activation function are universal approximators. This means that neural networks can learn to represent various functions. When providing a sufficient number of layers and corresponding activation functions, any continuous function can be modeled arbitrarily well [12]. Without activation functions, neural networks could only make linear assignments between inputs and outputs. Such neural networks cannot mathematically realize complex relationships in the data [13].

2.2.2. Activation Function

Activation function choices in neural networks mainly depend on the specific layer type and task to be solved. In classification tasks, one typically uses tanh, sigmoid and ReLU activations in hidden layers and a sigmoid or softmax function in the final layer [11]. The sigmoid function is used for binary and the softmax for multi-class classification. The softmax and sigmoid functions normalize the network output to a probability distribution over the predefined classes [11]. Selecting activation functions in the hidden layers does not follow such clear rules. All the mentioned functions have different characteristics, which lead to individual advantages and disadvantages. The sigmoid activation function squeezes the inputs in values between 0 and 1 and the tanh activation function squeezes them in values between -1 and 1. Both functions can suffer from vanishing gradients since the derivative of these functions is close to zero for very large or small inputs [10]. The ReLU activation function solves that problem, but maps all negative inputs to zero. The Leaky Relu function solves this so-called dead ReLU problem [14]. In table 2.1 the most popular activation functions are described.

Formula	Formulation s(x)	Derivative $\frac{ds(x)}{dx}$	Function Output Range
ReLU	$\begin{cases} 0 & \text{, for } x < 0 \\ x & \text{, for } x \geqslant 0 \end{cases}$	$\begin{cases} 0 & \text{, for } 1 < 0 \\ 1 & \text{, for } x \geqslant 0 \end{cases}$	$[0, \infty)$
Leaky ReLU	$\begin{cases} \alpha x & \text{, for } x < 0 \\ x & \text{, for } x \geqslant 0 \end{cases}$	$\begin{cases} \alpha & \text{, for } 1 < 0 \\ 1 & \text{, for } x \geqslant 0 \end{cases}$	$(-\infty, \infty)$
Sigmoid	$\frac{1}{1+e^{-x}}$	$\frac{e^{-x}}{(1+e^{-x})^2}$	(0,1)
Softmax	$\frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$	$\frac{e^{-x}}{(1+e^{-x})^2}$	(0,1)
tanh	$\frac{e^{2x}-1}{e^{2x}+1}$	$1 - tanh^2(x)$	$(-1, 1)$

Table 2.1.: Overview activation functions [11]

2.2.3. Loss Function

The loss function acts as an evaluation criterion for the neural network. During the optimization, the model is adapted to decrease the loss function. Deep learning can be applied in two different use cases: (1) regression tasks and (2) classification tasks. In a regression problem, the goal is to learn a mapping function from input variables to a continuous output variable. Conversely, in a classification problem, the model aims to predict a class label for each input sample [11]. Typically, the mean squared error (MSE) is applied as a criterion in regression tasks:

$$L(X) = \sum_{i=0}^{N} (\hat{y}(x_i) - y(x_i))^2, \tag{2.1}$$

where N is the number of training samples, $y(x_i)$ the ground truth and $\hat{y}(x_i)$ the predicted class label for the sample x_i [10]. On the other hand, a Cross-Entropy-loss (CE-loss) is common for classification tasks:

$$L(X) = \sum_{i=0}^{N} \sum_{j=0}^{C} y_j(x_i) log(p_j(x_i)), \tag{2.2}$$

where C is the number of predefined classes, $p_j(x_i)$ the predicted probability of the sample x_i belonging to the class j and $y_j(x_i)$ is the j-th entry of the one-hot encoding vector, representing the ground truth label of the sample x_i [11].

2.2.4. Optimizer

The optimizer is responsible for adapting the model according to the loss function. Usually, first-order methods are used to optimize neural networks. These methods solely rely on the first-order gradients to update the model parameters [11]. Second-order methods combine first- and second-order derivatives, which make the optimization converge faster. The calculation of the Hessian, which is required by this method, is expensive for large datasets and models [10][11]. When the dataset is large, full batch gradient descent methods, which calculate the gradient from the whole dataset and update the model accordingly, suffer from long training times [11]. The stochastic gradient descent (SGD) optimization tries to circumnavigate this problem. Repetitively, the model is updated with gradients calculated from a single sample picked randomly from the dataset. Since the choice of these samples is random, the optimization suffers from instability and fluctuation [11]. Therefore, a common practice is to separate the dataset into several subsets, referred to as mini-batches. For each mini-batch, the gradients are calculated and the model is updated accordingly. This process is repeated for all mini-batches retrieved from the dataset [11]. A training loop through the whole dataset is called an epoch. As soon as the loss converges, the training is terminated [11]. Despite convergence, an optimal solution is not assured due to the non-convexity of neural network optimizations [11].

Momentum

In order to accelerate and stabilize the optimization, historical gradients can be included. In this case, the model parameters are updated by a moving average over the past gradients. Those methods that use momentum accelerate the optimization in the relevant directions and dampen oscillations [15]. This gradient descent variant can adapt the step size in the different latent feature space dimensions. The step size is increased in the relevant and decreased in the irrelevant directions [15]. In the first step, the moving average over the past gradients is calculated and in the second step, the model parameters are updated accordingly:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta L(W_{t-1}) \\ W_t &= W_{t-1} - v_t, \end{aligned} \tag{2.3}$$

where v_t is the updated momentum, v_{t-1} the current momentum, W_t the updated model weights, W_{t-1} the current model weights, $\nabla_\theta L(W_{t-1})$ the derivative of the loss with respect to the current model weights, η the learning rate and γ the hyperparameter balancing the current momentum and gradient for calculating the updated momentum [15].

Nesterov Accelerated Gradient

Another well-known optimizer of this kind is the Nesterov Accelerated Gradient (NAG), which extends the regular first-order momentum update rules. When calculating the first-order momentum, NAG calculates the gradient with respect to the pre-updated weights:

$$\nabla_\theta L(W_{t-1} - \gamma v_{t-1}), \tag{2.4}$$

where W_{t-1} are the current model weights, which are pre-updated with the current first-order momentum v_{t-1}. This special gradient estimation is used to calculate the momentum, which is then used to update the model parameters as described in equation 2.3 [15].

Adagrad

Adagrad uses a squared version of the moving average over the past gradients:

$$W_t = W_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \bigodot \nabla_\theta L(W_{t-1}), \tag{2.5}$$

where W_{t-1} are the current and W_t the updated model weights, $\nabla_\theta L(W_t)$ is the derivative of the loss with respect to the current model weights, G_t is the second-order momentum, which is a diagonal matrix with each diagonal element i,i being the sum of the squared first-order gradients with respect to the model parameter i, ϵ denotes a small quantity, which prevents the division by zero and γ is the learning rate [15].

Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) is one of the most popular optimizers. ADAM combines the idea of first and second-order momentum:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta L(W_{t-1}) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\nabla_\theta L^2(W_{t-1}) \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ W_t &= W_{t-1} - \frac{\eta}{\sqrt[2]{\hat{v}_t + \epsilon}} \hat{m}_t, \end{aligned} \tag{2.6}$$

where m_t and v_t are the first- and second-order momentum, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first- and second-order momentum, β_1 and β_2 are the weighting factors for the moving average and W_{t-1} and W_t are the current and updated model weights [15].

2.2.5. Training Loop

During the training, the model parameters are adapted to minimize the loss function. In a two-stage process, the model is optimized by alternately applying the forward and backward pass [11]. Figure 2.3 shows the optimization process of a single neuron in more detail:

- **Forward pass**: The i inputs, which are connected with the single neuron j are multiplied with its weights $w_{i,j}$ and summed up together with a bias b_j. The resulting logit z_j is then processed by the activation function ϕ. Generally, different activation functions can be used throughout the network. After calculating the values for all neurons in the consecutive hidden layers, a loss function evaluates the prediction of the neural network [16].

- **Backward pass**: At first, the partial derivatives of the model layers are calculated. Afterwards, the derivatives of the loss with respect to the weights and biases are calculated by concatenating the corresponding partial derivatives in the reverse order of the forward pass. The chain rule is used for that concatenation [11]. Finally, the model parameters are updated in the negative direction of the corresponding gradient with a step size defined by the learning rate:

$$\theta = \theta - \eta \cdot \nabla_\theta L(\theta), \tag{2.7}$$

  where θ are the model parameters, η is the learning rate and $\nabla_\theta L(\theta)$ the derivative of the loss function with respect to the model parameters [17]. Exemplary, the derivative of the loss with respect to the weight $w_{i,j}$ between input i and the single neuron j is defined as follows:

$$\frac{\delta L}{\delta w_{i,j}} = \frac{\delta L}{\delta \hat{y}_j} \cdot \frac{\delta \hat{y}_j}{\delta z_j} \cdot \frac{\delta z_j}{\delta w_{i,j}}, \tag{2.8}$$

  where $\frac{\delta L}{\delta \hat{y}_j}$ is the derivative of the loss L with respect to the model's predicted probability of the training sample belonging to class j, $\frac{\delta \hat{y}_j}{\delta z_j}$ is the derivative of the activation function in the last layer, $\frac{\delta z_j}{\delta w_{i,j}}$ is the derivative of the logit z_j with respect to the weight of interest $w_{i,j}$ [11].

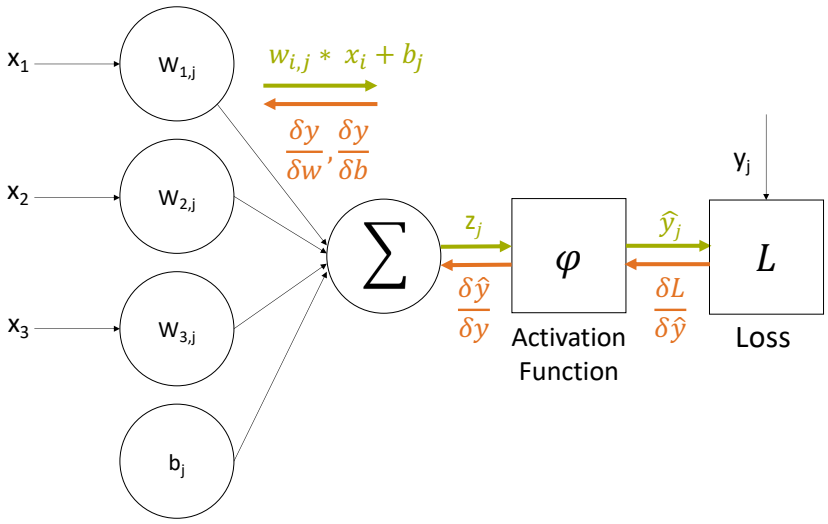

Figure 2.3.: Optimization of neural networks

2.3. Convolutional Neural Network

Equally to regular neural networks, convolutional neural networks (CNNs) consist of several neurons embedded in a fixed architecture. Developed for computer vision applications, the architecture of CNNs is optimized to process images. In CNNs, the neurons are structured in layers, just like in regular neural networks. Instead of organizing the neurons in one dimension, CNNs do that in three dimensions (height, width, depth) [18]. A typical CNN architecture is visualized in 2.4. One can identify four main characteristics of CNNs, which are described in more detail in the following:

1. The input data is organized as a structured and grid-like form. Each element in this structure is called a pixel, which is specified by a value and position. In the latent feature spaces, the data is stored as arrays with spatial dimension (height x width) and depth (channel size) [18].

2. Convolutional layers contain kernels that are convolved with the input. Each kernel contains weights and biases, which are learned during training. An elementwise activation function is applied to the kernel outputs [18].

3. Pooling layers downsample the spatial dimension. This reduces the height and width of the feature maps and therefore the network complexity [18].

4. Final fully-connected layers coupled with activation functions predict class labels for the input data [18].

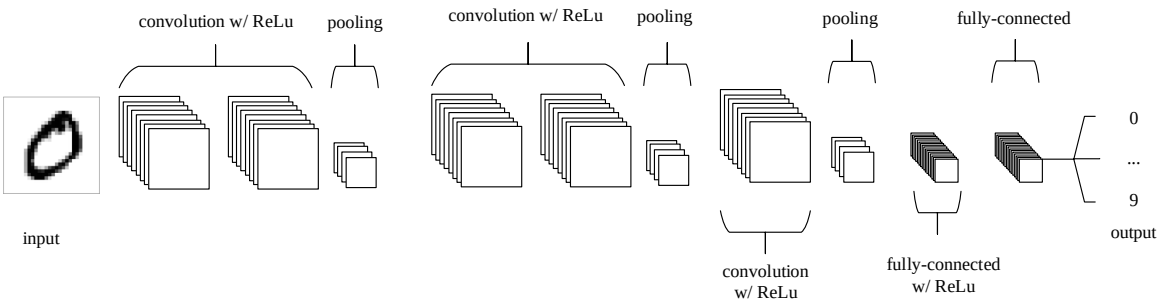

Figure 2.4.: Model architecture of CNNs [18]

In the following, the typical components of CNNs are presented. The function and properties of convolutional and pooling layers are described in more detail.

2.3.1. Convolutional Layer

The convolutional layers are the core elements in CNNs. The learnable parameters in a convolutional layer are the weights and biases of the corresponding kernels. During the optimization, each kernel learns to extract expressive features. The depth of the input layer defines the depth of a kernel and the number of applied kernels defines the depth of the subsequent feature map. Usually, the spatial dimensions (width, height) are reduced and the depth of the latent feature map is increased throughout the network. Therefore, the network extracts more global features at the beginning and more local features at the end of the network [18]. Looking at figure 2.5, one can see how the kernel of depth three is applied to the input of depth three. By using six kernels, the resulting feature map is of depth six.

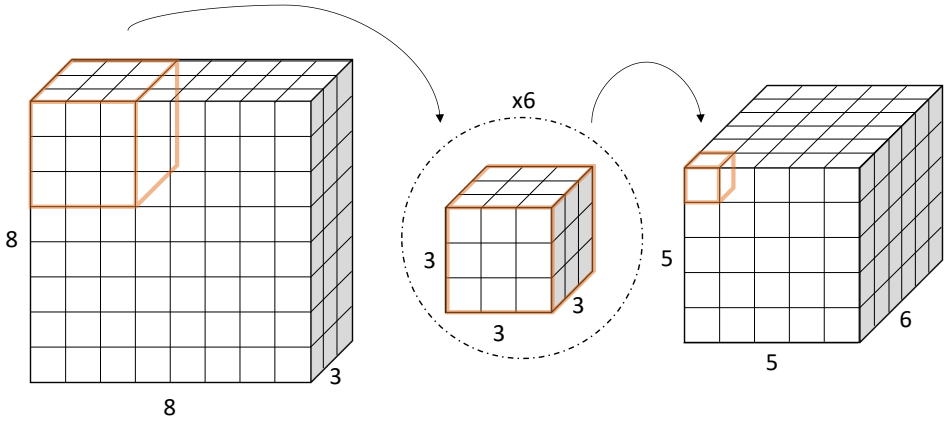

Figure 2.5.: 2D convolution: input feature map (left), kernel (middle), output feature map (right)

For the one-dimensional case, the convolution of a kernel with a subspace of the input is defined as following:

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n), \tag{2.9}$$

where p_n is one of the R kernel cells, p_0 is the lower bound pixel position of the input subspace, $x(\cdot)$ is the input and $w(\cdot)$ the kernel. Each kernel cell is multiplied with the corresponding input pixel. The R outputs are summed up in the pixel p_0 of the subsequent feature map [19]. Typically, a bias value is included in this weighted sum and a non-linearity is applied consecutively. The previously described convolution is visualized in figure 2.6.

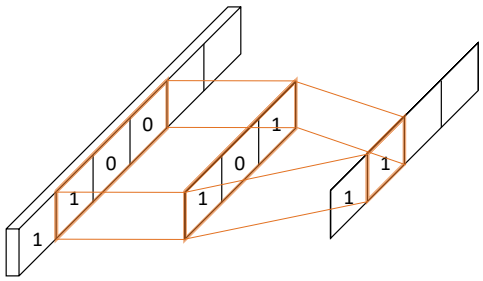

Figure 2.6.: 1D convolution: input feature map (left), kernel (middle), output feature map (right)

Compared to regular neural networks, CNNs profit a lot from its weight sharing concept. Since the same kernel is applied to different input areas, it is unnecessary to train weights for every pixel along the whole spatial dimension of the input. This reduces the number of learnable parameters in the network. Since the kernel is applied to different input subspaces, the feature search is insensitive to the feature location in the image [18].

2.3.2. Convolution Parameters

When defining a CNN architecture, one has to find a balance between training effort and model complexity. The CNN should be able to capture relevant information from the data without requiring extensive optimization. The spatial output dimension of a convolutional operation is defined as follows:

$$V_{out} = \frac{(V_{in} - R) + 2Z}{S + 1}, \tag{2.10}$$

where S is the stride, Z the zero padding, R the receptive field of the applied kernel and V_{in} the spatial dimension of the input [18]. The parameters of the convolutional layers

significantly influence the characteristic of CNNs, which is why they are specified in more detail in the following.

Stride

The stride defines the number of pixels skipped while shifting the kernel over the input. By increasing the stride, the spatial dimension of the resulting feature map is decreased [18]. Figure 2.7 visualizes the effect of the stride on the convolutional operation.

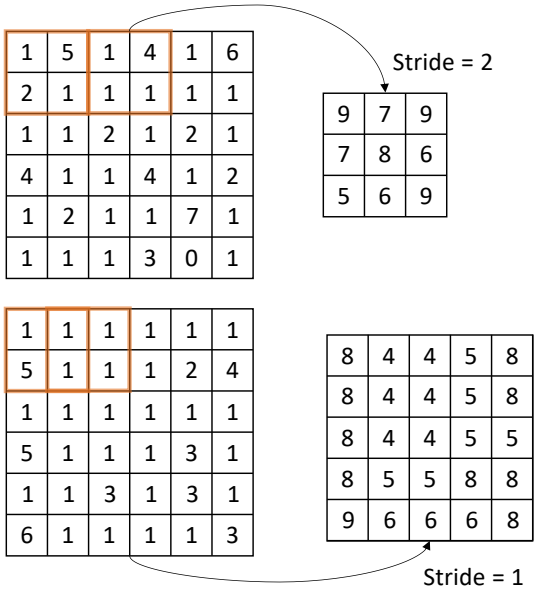

Figure 2.7.: Stride factor

Zero Padding

Zero padding, shown in figure 2.8, enlarges the input with a border of zeros. During the convolution, the kernel covers an increased spatial dimension, which increases the spatial dimension of the resulting feature map [18].

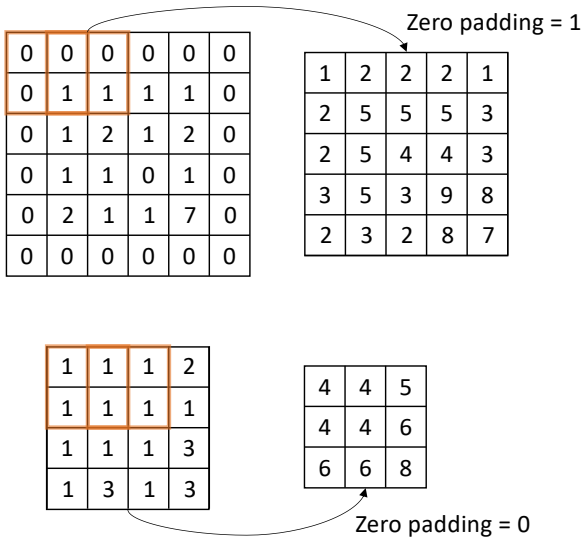

Figure 2.8.: Zero padding

Receptive field

The receptive field is the spatial dimension of the input subspace, which is covered by the kernel during a single convolutional operation. When increasing the receptive field, more global and otherwise more local features are extracted from the input. The receptive field is defined by the spatial dimension of the kernel [18]. Dilated convolution can be applied to increase the receptive field size while maintaining the model complexity [19].

2.3.3. Pooling Layer

Pooling layers are applied to change the spatial dimension of the latent feature spaces throughout the network. The functionality is generally similar to convolutional layers, the only difference being that no learnable parameters are involved. Pooling kernels are shifted over the input just as regular kernels in convolutional layers. For each kernel position, all pixels covered by the kernel are merged in a single value. Max-pooling layers return the maximal and AVG-pooling layers the average pixel value from all pixels covered by the kernel. The differences between Max-pooling and AVG-pooling layers are shown in figure 2.9. Often convolutional and pooling layers are applied consecutively [18].

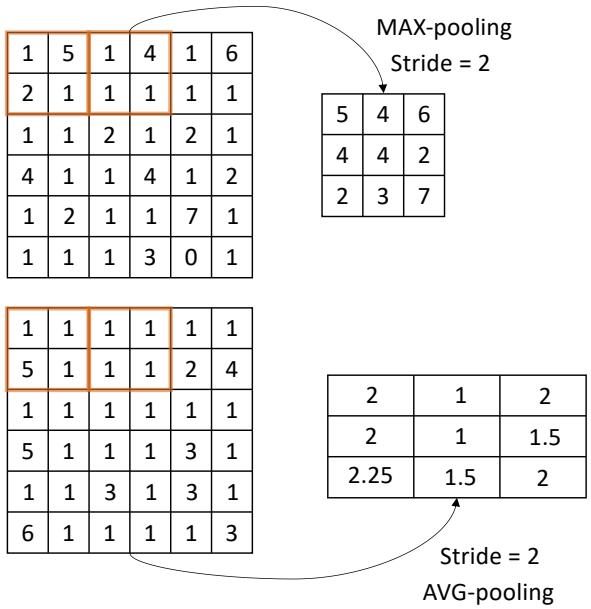

Figure 2.9.: Pooling layer types

2.4. Domain Adaptation and Transfer Learning

In the computer vision community, domain adaptation and transfer-learning techniques recently received more attention. Transfer learning addresses applications in which a model is trained to solve a specific task on a given dataset. The model is then applied to solve a different task on that same dataset [20]. Domain adaptation solves problems in which a model is trained on a labeled training dataset, denoted as the source domain. The model is then applied to solve the same task on a different unlabeled test dataset, denoted as the target domain. The data distribution of the target and source domain is different, but the data must be related in any sense and structured similarly [20]. The differences between domain adaptation and transfer learning are visualized in figure 2.10. Since this thesis focuses on domain adaptation approaches, the following passages explain different aspects of domain adaptation.

2.4.1. Notation

The labeled source domain data is denoted by $S = (x_i^s, y_i^s)_{i=0}^{i=N_s}$. Generally, the target domain data is separated into labeled $T_l = (x_i^{tl}, y_i^{tl})_{i=0}^{i=N_{tl}}$ and unlabeled data $T_u = (x_i^{tu})_{i=0}^{i=N_{tu}}$. It is assumed that there is a large amount of labeled data in the source and in some cases a small amount of labeled data in the target domain available ($N_{tl} \ll N_s$). The input samples are

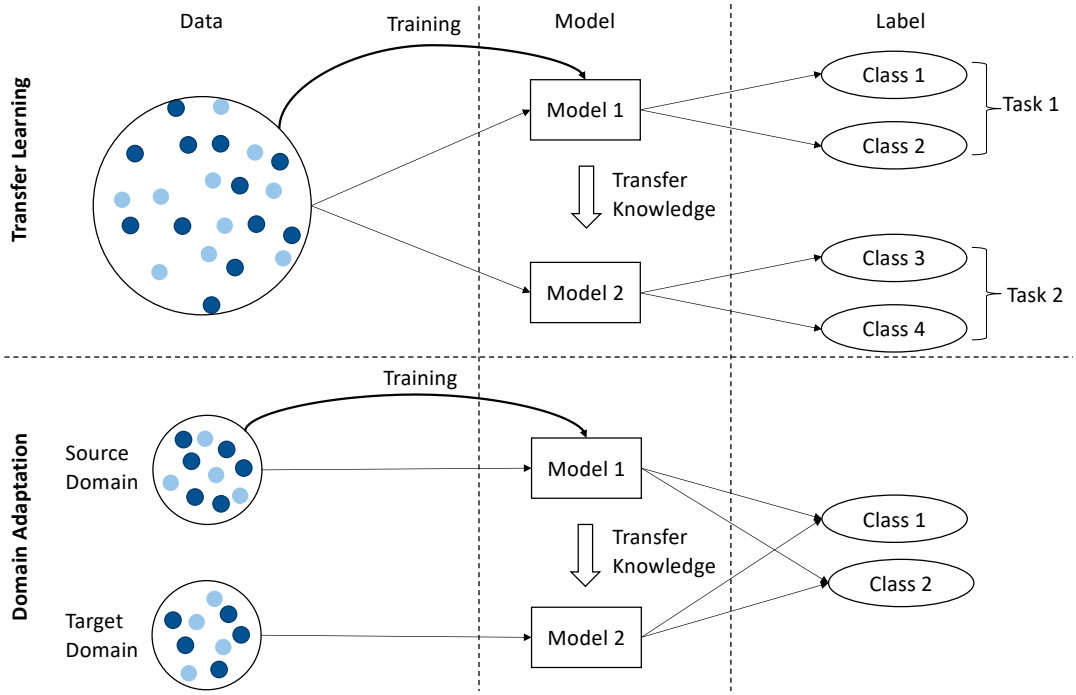

Figure 2.10.: Transfer learning vs. domain adaptation

defined by x_i and the corresponding label y_i [20]. Depending on the data available during training, one differs between different domain adaptation methods:

- **Semi-supervised domain adaptation**, where a model is trained by using the data from S, T_l [20].
- **Unsupervised domain adaptation**, where a model is learned by using the data from S and T_u [20].

From a statistical point of view, the source and target domain can be described by the marginal distribution $P(X)$ and conditional distribution $P(Y|X)$. The data from the source and target domain have the same data space and label space, but the marginal and conditional distribution differ $P(Y_s) \neq P(Y_t)$ and $P(Y_s|X_s) \neq P(Y_t|X_t)$ [21].

2.4.2. Domain Adaptation Types

Generally, domain adaptation approaches can be grouped into four different types:

- **Instance Weighting Methods** address the covariate shift by integrating weights into the loss function, which estimate the similarity between the source and target samples. Weighting factors like $\frac{P_t(x)}{P_s(x)}$ can be used. Source domain samples, which have a high

probability of being in the target domain, are quite similar to the target domain samples. Samples like that should be strongly included in the training to adapt the model to work well on the target domain data [5].

- **Feature-Based Methods** find a domain-invariant feature space in which the domain discrepancy is reduced. The classification problem of both domains becomes more similar when transferring the source and target samples in this domain-invariant feature space [5]. Figure 2.11 illustrates how feature-based domain adaptation can be used to find a cross-domain classifier, which accurately separates the source and target domain data [4].
- **Model-Based Methods** train a classifier on the source domain, which can be transferred or fine-tuned to perform well on the target domain [5].
- **Relation-Based Methods** utilize similarities between the two domains to transfer knowledge between the domains [5].

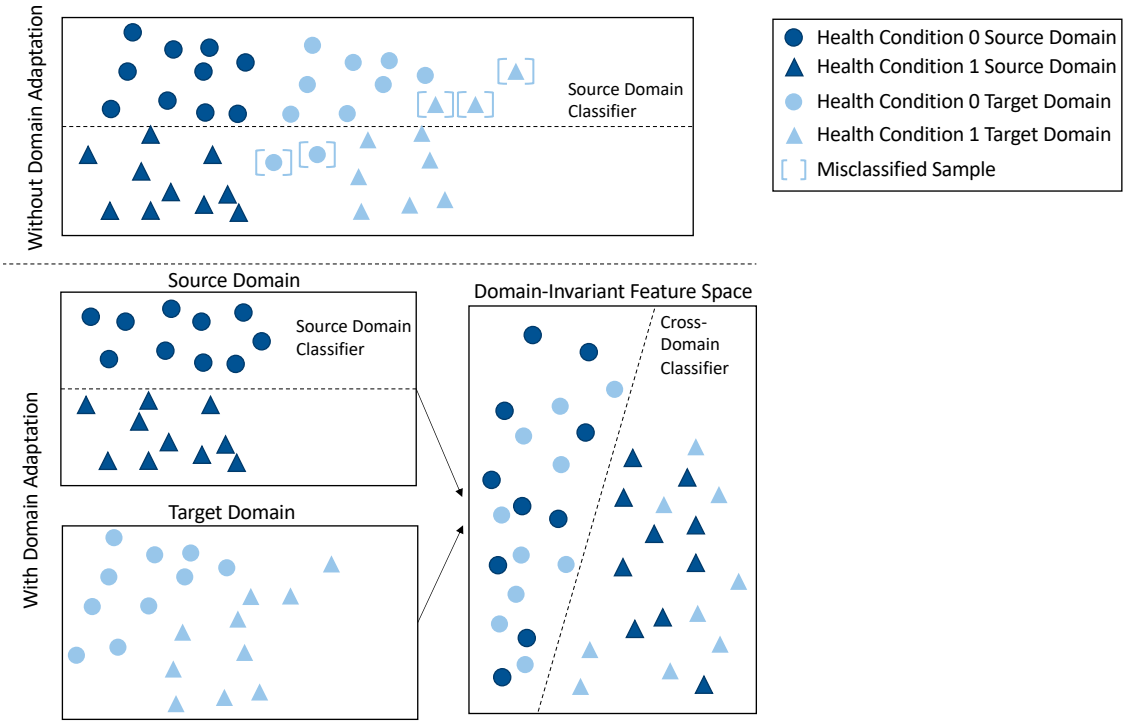

Figure 2.11.: Feature-based domain adaptation for PHM based on [4]

2.5. Maximum Mean Discrepancy

Maximum Mean Discrepancy (MMD) is a criterion estimating the discrepancy between two distributions. MMD can be used to optimize a neural network such that the distribution discrepancy in its latent feature space is reduced. In the reproducing kernel Hilbert space (RKHS), the discrepancy is measured as the squared distance between the marginal distribution kernel embeddings [4]. The distribution discrepancy across domains can be measured in several neural network layers. Including this information in the optimization of the model helps to avoid feature transferability degradation [22]. The MMD criterion is defined as follows:

$$M_k(P, Q) = \left| \boldsymbol{E_P}[\Phi(\boldsymbol{X^s})] - \boldsymbol{E_Q}[\Phi(\boldsymbol{X^t})] \right|^2_{Hk}, \tag{2.11}$$

where Hk denotes the RKHS, which is described by the characteristic kernel k and the mapping function Φ [5]. The discrepancy between the marginal distribution means is measured when the identity function is used as a mapping function. When using more complex mapping functions, higher order moments can also be matched [23]. The distributions of the source domain $X^s = \{x_i^s\}_{i=0,\dots,n_s}$ and target domain $X^t = \{x_i^t\}_{i=0,\dots,n_t}$ in the latent feature spaces are represented by P and Q and the corresponding expectations by $\boldsymbol{E_P}[\cdot]$ and $\boldsymbol{E_Q}[\cdot]$ [5]. When applying the MMD to optimize neural networks, the kernel choice is of great importance. For this reason, it makes sense to combine several kernels to profit from their individual performance:

$$k(\boldsymbol{X^s}, \boldsymbol{X^t}) = \sum_{i=0}^{N_k} k_{\sigma_i}(\boldsymbol{X^s}, \boldsymbol{X^t}), \tag{2.12}$$

where N_k denotes the number of kernels used in the RKHS and k_{σ_i} represents the individual RBF kernels [22]. Other kernels, like linear kernels, can also be used, but current research shows that RBF kernels usually perform best [5].

2.6. Non-Stationary Signal Analysis for Prognostic and Health Management

Non-stationary signal analysis methods investigate signals with changing statistical properties. Traditionally those approaches play an essential role in the fault diagnosis of industrial machines. Machine signals contain multiple frequencies and amplitudes, which might

change over time. Traditional signal analysis techniques make stationary assumptions. When applying those techniques to non-stationary signals, solely statistical averages in time or frequency domain are extracted [24]. Therefore, the demand for analysis methods, which allow to ascertain features of non-stationary signals, is increasing. Such methods seem promising for extracting health-related information from machine data. Time–frequency representations (TFRs) are techniques to transform non-stationary signals in a two-dimensional time-frequency domain, where each value describes the dominance of a specific frequency at a certain point in time. All TFRs, which fulfill the idea of linearity and superposition, are called linear TFRs. The two most popular linear TFRs are the short-time Fourier and the Wavelet transform [25].

Short-Time Fourier transform

Short-time Fourier transform (STFT) is a method that adds a time variable to the traditional Fourier spectrum. This allows to investigate variations in the signal spectrum over time. In STFT, the spectrum is assumed to be constant during a short time window. For each such window, a Fourier spectrum is obtained. The time-related changes are measured between consecutive window snapshots. The process is mathematically expressed in the following:

$$STFT_x(t,f) = \int_{-\inf}^{+\inf} x(\tau)w(\tau - t)exp(-j2\pi f\tau), \tag{2.13}$$

where $w(\tau - t)$ is the window function centered around t and $x(t)$ is the signal [24]. Specific window functions are defined to separate the signal. Shifting the window over the signal and applying the Fourier transform $exp(-j2\pi f\tau)$ to each window generates a local frequency spectrum of the signal for different points in time [24]. The time-frequency resolution is defined by the windowing function and the window length. STFT suffers from a trade-off between high resolution in the time or frequency domain. The optimum window length depends on the main interest behind the signal analysis. The window size needs to be reduced for precise time domain information and increased for accurate frequency domain information [25]. The STFT decomposes the signal in the existing sinusoidal and determines its frequency and phase for a local part of the signal [25].

Wavelet Transform

The Wavelet transform decomposes the signals in several wavelets. A wavelet is a wave-like oscillation, described by its function, location and scale. The location defines where the wavelet overlaps with the signal and the scale defines how much squished (small scale) or

stretched (large scale) the wavelet is [26]. The convolution of the wavelet and the signal is mathematically expressed as following:

$$WT_x(t,a) = \frac{1}{\sqrt{a}} \int_{-\inf}^{+\inf} x(\tau)\psi(\frac{\tau - t}{a})d\tau, \tag{2.14}$$

where $x(t)$ is the signal, $\psi(\frac{\tau-t}{a})$ the wavelet, a the scaling factor, t the time shift and $\frac{1}{\sqrt{a}}$ a normalization factor to maintain the energy conservation [24]. Different wavelet bases $\psi(t)$ can be convolved with the signal to analyze the signal for different patterns [26]. Popular wavelet bases are the Gaussian, Morlet, Shannon, Meyer, Laplace, Hermit, or the Mexican Hat wavelets in both simple and complex functions [27]. This enables more extensive, flexible, and detailed analysis. The wavelet transform can be adapted to extract patterns, which are especially relevant for the PHM task. In figure 2.12, Ricker wavelets with different scales and locations are visualized. Wavelet transforms can extract local spectral and temporal information parallel [26].

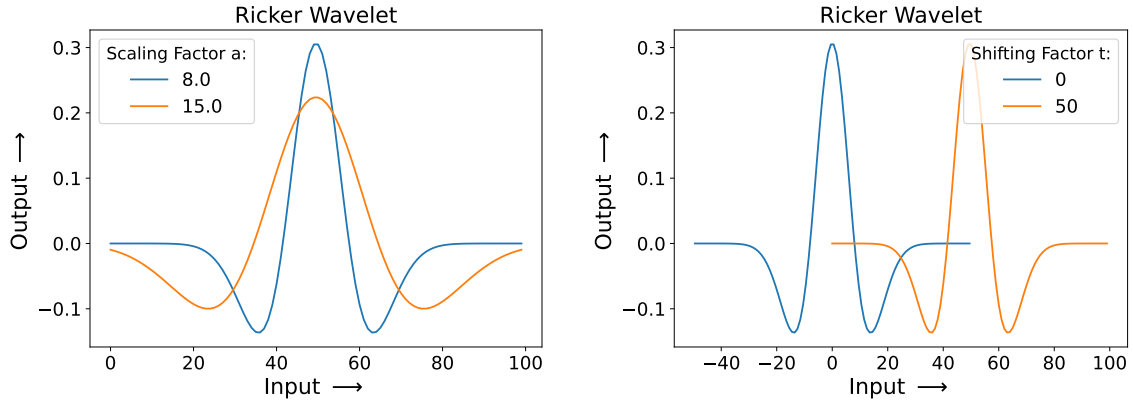

Figure 2.12.: Ricker wavelet: Different scaling factors (left) and shifting factors (right)

Spectrograms and Scalograms

Spectrograms are a graphic representation of the STFT and scalograms of the wavelet transform. Spectrograms and scalograms visualize the squared magnitudes of the previously presented STFT and Wavelet transform. This squared magnitude is loosely interpreted as signal energy [25]. The mathematical expressions are presented in the following:

$$\begin{aligned} SPEC_x(t,f) &= |STFT_x(t,f)|^2 \\ SCAL_x(t,f) &= |WT_x(t,f)|^2, \end{aligned} \tag{2.15}$$

where $STFT_x(t,f)$ is the Short-time Fourier transform, $WT_x(t,f)$ the wavelet transform,

$SPEC_x(t, f)$ the spectrogram and $SCAL_x(t, f)$ the scalogram [25]. This way of representing the system energy in a two-dimensional time and frequency domain may reveal useful information from a complex and high-dimensional signal without requiring additional feature extraction. As described before, spectrograms have a fixed frequency resolution defined by the window size. Scalograms, on the other hand, have a frequency-dependent frequency resolution [27].

3. Related Works

In this chapter, works from the literature are presented, which tackle problems related to the presented PHM task in this thesis. Section 3.1 discusses traditional PHM approaches for BSDs, which do not apply any domain adaptation. Then, in chapter 3.2, deep learning based domain adaptation approaches are introduced, which perform health condition monitoring for BSDs and rolling bearings. Finally, section 3.3 presents an advanced domain adaptation approach from the computer vision community.

3.1. Traditional Approaches for Prognostic and Health Management

Traditionally, model-based and data-driven models were used for PHM. Model-based methods predict the health condition based on physical models describing the underlying degradation mechanisms. Data-driven methods learn a mapping relationship between the machine's health condition and the monitoring data [8]. The following presents traditional data-driven and model-based PHM systems for monitoring the health condition of BSDs.

3.1.1. Model-Based Approach: Monitoring of Defect Frequencies Calculated from Rolling Bearings and Transferred to Ball Screw Feed Drives

Lee et al. [7] proposed a diagnosis system for estimating the flaking degradation of BSD screw shafts. By filtering the machine signals for previously calculated characteristic defect frequencies, the severity and location of the degradation were predicted. Lee et al. developed a testbed containing one BSD and two linear motion guides. An accelerometer was mounted on the BSD nut. The continuous fatigue process was simulated by punching holes with a diameter of 3 mm in the BSD screw shaft. The motor was actuated with a constant velocity while the data was recorded. Harris and McCool [28] proposed a method to estimate the characteristic defect frequencies of rolling bearings. Lee et al. [7] extended that method to make it applicable for BSDs. The BSD screw shafts were considered as inner rings and the BSD nuts as outer rings of the rolling bearings. The defect frequencies were calculated from the BSD construction details and relative speeds. The ball pass frequencies of the shaft (BPFS),

the ball pass frequencies of the nut (BPFN) and the ball spin frequency (BSF) were considered as defect frequencies:

$$BPFS = \frac{1}{120} zn(1 + \frac{D_w}{d_m} cos\alpha), \tag{3.1}$$

$$BPFN = \frac{1}{120} zn(1 - \frac{D_w}{d_m} cos\alpha), \tag{3.2}$$

$$BSF = \frac{1}{120} n\frac{d_m}{D_w}(1 - \frac{D_w}{d_m} cos\alpha)(1 + \frac{D_w}{d_m} cos\alpha), \tag{3.3}$$

where α is the contact angle between the ball, nuts and screw shaft, d_m is the pitch diameter of the balls, D_w is the diameter of a single ball, n is the rotational speed of the BSD screw shaft and z is the number of steel balls. A more detailed visualization of the bearing parameters is shown in figure 3.1.

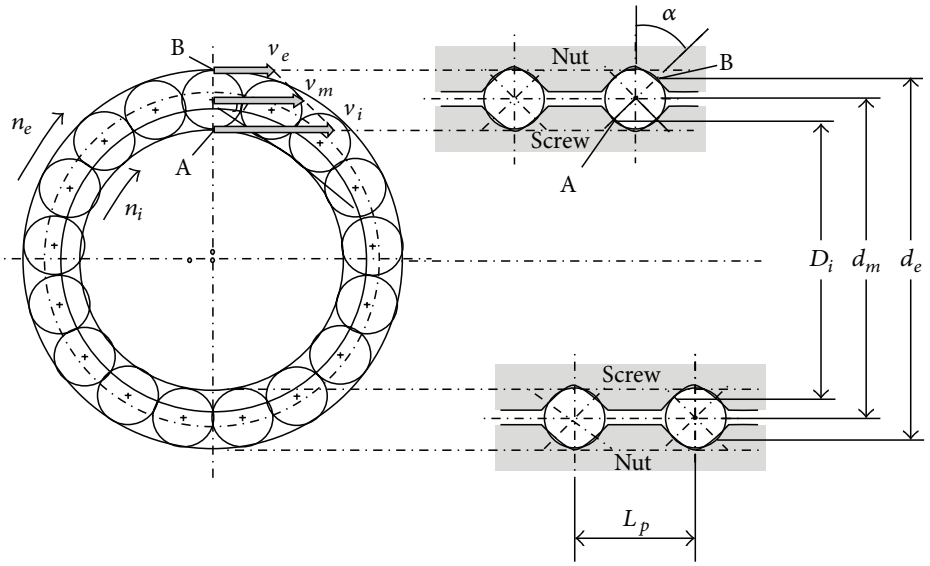

Figure 3.1.: Visualization of the parameters required for the calculation of the defect frequencies [7]

The derived frequencies above are valid for rolling bearings. To correctly apply those to BSDs, z and d_m need to be replaced by the effective number of steel balls $z^{'}$ and effective pitch parameter $d_m^{'}$, which are defined as follows:

$$d_m^{'} = (L_p^2 + (\pi D_b)^2)^{\frac{1}{2}}, \tag{3.4}$$

$$z^{'} = \frac{d_m^{'}}{D_w}. \tag{3.5}$$

The relation between the regular and effective parameters and the required parameters L_p and D_b for equation 3.4 are visualized in figure 3.2.

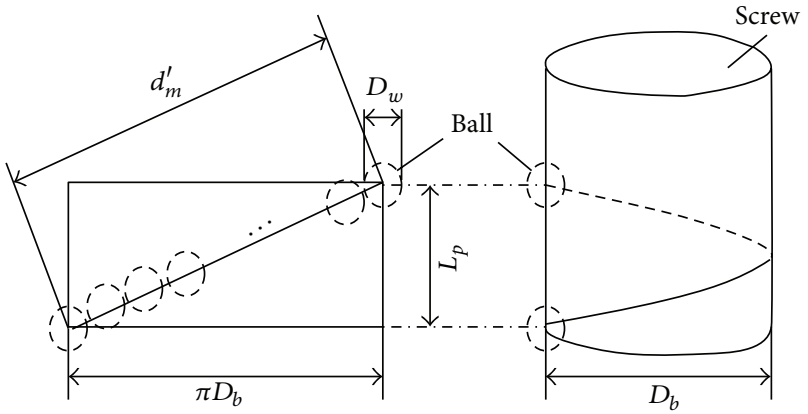

Figure 3.2.: Relationship between the regular and effective parameters required for transferring the calculated defect frequencies to the BSDs [7]

The BPFS frequency was identified as the most expressive and reliable for supervising the health condition of BSDs. To calculate the BPFS for ball screw feed drives, the equation 3.1 must be combined with the effective pitch parameter d'_m and the effective number of steel balls z'. During testing, the wavelet transform (Daubechies Wavelet (db14) function) was used to transform the machine signals in the two-dimensional time-frequency domain. The BSD's degradation status was monitored by supervising the magnitude of the calculated defect frequency in the machine signal's time-frequency domain. An alarm was triggered when the degradation was not acceptable anymore. The time-related information was an indicator of the defect location. The frequency-related information provided information about the degradation extent of the BSD [7]. The proposed approach during testing is visualized in figure 3.3.

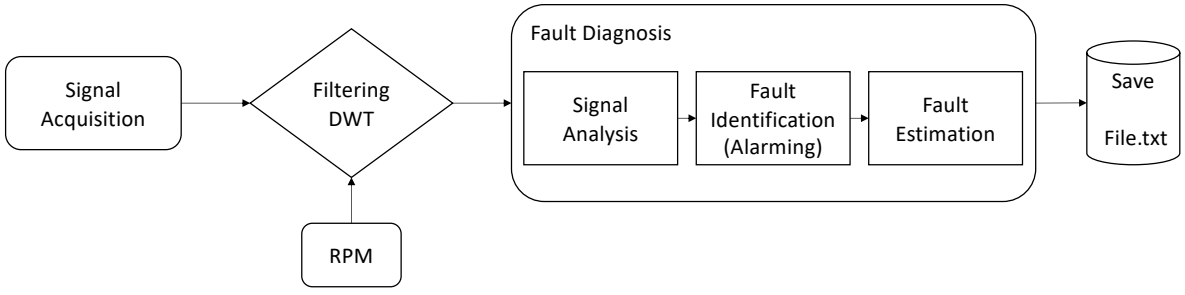

Figure 3.3.: Failure diagnosis system during testing based on [7]

Conclusion

Lee et al. [7] assumed that defects and degradation are mainly subjected to rolling friction. Such simplifying assumptions are often made when developing model-based PHM systems. In reality, the balls in the BSDs are rotating, revolving, and sliding [7]. These highly complex processes make accurate degradation modeling, which is essential to achieving a good PHM performance, more challenging. In data-driven PHM systems, the correlation between the machine data and the degradation patterns can be learned automatically [1]. Therefore, these systems can be adapted to monitor different degradation types and corresponding levels by selecting expressive signals and defining appropriate health condition classes. In comparison, Lee et al. [7] had to adhere to the defect frequencies developed by Harris et al. [28], which were limited to the flaking degradation of BSDs. Therefore, adapting the diagnosis system proposed by Lee et al. [7] to monitor other degradation types is nearly impossible. Furthermore, different operational conditions of the machine might influence the dominance of the characteristic defect frequencies in the vibration signal. Even in the experiments executed on the simplified testbed, Lee et al. [7] observed other distinct frequencies coming from significant electrical noise. The monitoring task could become more challenging if vibrations from other parts of the machine disturb the machine signals. This becomes particularly relevant when applying the developed approach to monitoring real-world machines, where the mutual influence of different machine parts is especially strong. Apart from that, the BSD vibration signals were recorded by an accelerometer mounted on the BSD nut [7]. The BSD nut is a highly fault-critical but also a very impractical sensor location for real-world use [4]. It is questionable how this method would work with vibration signals recorded from less fault-critical sensor locations. Additionally, finding all parameters to calculate the defect frequencies may require copious measuring and testing. Often these parameters change throughout the lifetime of the BSDs. Despite the rolling diameter being reduced due to degradation, it was still used as a constant parameter in the calculation of the BSD's defect frequencies [2] [4]. This indicates that the proposed method is not applicable to monitoring BSDs' health condition over long periods. Only structural differences were considered when transferring the calculated defect frequencies from the rolling bearing to the BSD [7]. The linear movement of the BSDs was completely ignored, representing a fundamental functional difference between rolling bearings and BSDs. The ongoing degradation was simulated by punching 3 mm diameter holes in the grooves of the BSD screw shaft. Variations in the dimension of these holes were not considered [7]. This simplified degradation simulation reduces the relevance of the presented results.

3.1.2. Model-Based Approach: Ball Screw Feed Drive Preload Estimation Based on a Discrete Dynamic Model

Nguyen et al. [29] applied a simplified discrete dynamic model (see figure 3.4) to investigate the relation between the preload variations and the dynamic characteristics of BSDs. By estimating the natural frequency of the screw nut in the axial direction from the vibration and current signals, this relationship was then used to predict the preload level of the BSDs. The proposed method was evaluated on a simplified testbed. In the experiments, the preload of the BSDs was changed by a built-in preload adjustment mechanism

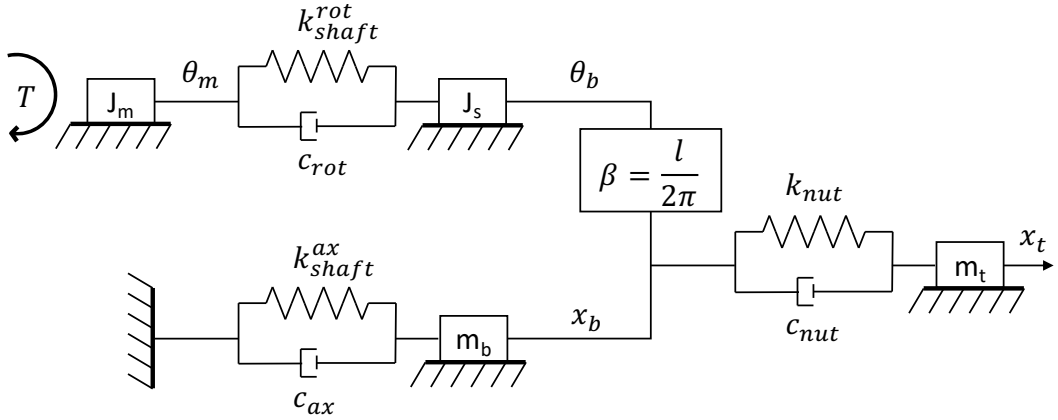

Figure 3.4.: Discrete dynamic model based on [29]

According to the discrete dynamic model, the preload variations of BSDs correlate with the screw nut stiffness:

$$k_{nut} = 0.8K(\frac{P}{0.1C_a})^{\frac{1}{3}}, \tag{3.6}$$

where P is the BSD preload, C_a is the BSD screw dynamic load, K is the BSD nut stiffness according to the manufacturer and k_{nut} is the actual BSD nut stiffness. The formula is valid if the BSD preload is less than 10% of the BSD dynamic load. The axial and rotational stiffness of the BSD screw shaft are determined by the configuration parameters and the working table displacement:

$$k_{shaft}^{ax} = \frac{EA}{x_t} = \frac{\pi}{4x_t} d_{minor}^2 E, \tag{3.7}$$

$$k_{shaft}^{rot} = \frac{\pi}{32L} d_{minor}^4 G, \tag{3.8}$$

where A is the cross sectional area of the BSD screw shaft, E the Young's modulus, d_{minor} the screw diameter, G the shear modulus, L the screw shaft length and x_t the working table

displacement. The total axial stiffness of the BSD is composed of several other rotational and axial stiffness:

$$\frac{1}{k_{ax}} = \frac{1}{k^{ax}_{shaft}} + \frac{1}{k^{ax}_{bearing}} + \frac{1}{k^{ax}_{nut}} + \frac{1}{k^{ax}_{bracket}} + \frac{1}{\frac{k^{rot}_{shaft}}{\beta^2}}, \tag{3.9}$$

where k^{ax}_{shaft} is the axial stiffness of the screw shaft, k^{rot}_{shaft} the rotational stiffness of the screw shaft, $k^{ax}_{bearing}$ the axial stiffness of the supporting bearing, k^{ax}_{nut} the axial stiffness of the screw nut and $k^{ax}_{bracket}$ the axial stiffness of the bracket. From there, the axial natural frequency can be calculated based on the axial stiffness k_{ax} and the sum of all masses in the ball screw system $\sum M$:

$$f \approx \frac{1}{2\pi}\sqrt{\frac{k_{ax}}{m_{table} + m_{screw} + m_{nut} + m_{bracket}}} = \frac{1}{2\pi}\sqrt{\frac{k_{ax}}{\sum M}}. \tag{3.10}$$

Finally, the preload of BSDs is related to the axial natural frequency, the mass and displacement of the ball screw system and other configuration parameters:

$$P = \frac{0.1C_a}{\{0.8K[-\frac{4x_t}{\pi d^2_{minor}E} - \frac{32\pi^2 L}{\pi d^4_{minor}G} - \frac{1}{k_{bearing}} - \frac{1}{k_{bracket}} + \frac{1}{(2\pi f)^2 \sum M}]\}^3}, \tag{3.11}$$

where C_a is the dynamic load rating, K the nut stiffness from catalog, $k_{bearing}$ the bearing stiffness and $k_{bracket}$ the bracket stiffness. The axial natural frequency increases with the preload and decreases with the mass of the working table. Based on the formulas derived from the simplified discrete dynamic model, a monitoring system was established, which extracted the axial natural frequency from the machine data and calculated the corresponding BSD preload according to equation 3.11. The monitoring system can be separated into different processing phases. Firstly, the vibration and motor current signals were measured with a uni-axial acceleration sensor mounted on the screw nut along with three Hall-effect-based current sensors. When the motor speed changed rapidly, the modal modes of the BSD system, including the axial natural frequency, were strongly activated. In these phases, the deformation of the BSD system was strong enough to make its modal modes observable and distinguishable from those of the other components. A trigger was implemented to detect the phases of rapid motor speed change. During those phases, the vibration and motor current signals were recorded and windowed afterward. The FFT transform was applied to extract an auto-spectrum and cross-spectrum for each window. By averaging the FFT transforms from several occurrences at the same positions along the BSD screw shaft, more stable results were achieved, which were less prone to noise. The average frequency response function (FRF) was composed from those spectra. The whole FRF extraction is visualized in figure 3.5. The axial natural frequency is the frequency in the FRF with maximum magnitude, which was found by a peak detection algorithm [29].

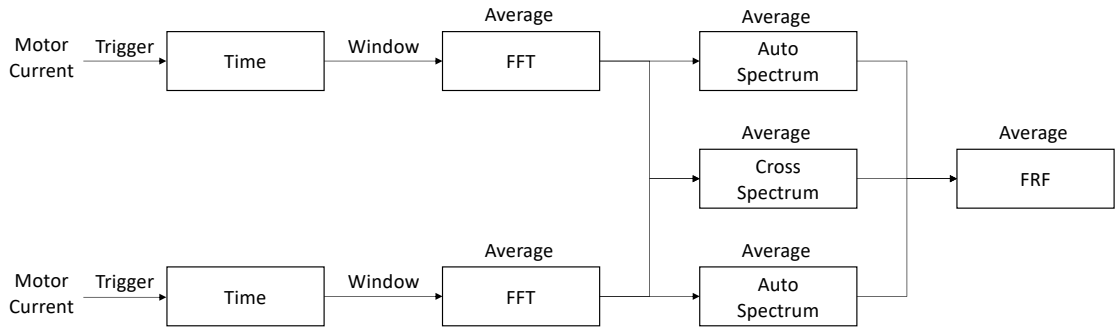

Figure 3.5.: Average FRF calculation based on [29]

Conclusion

Nguyen et al. [29] developed a complex inspection procedure to predict the preload of BSDs. The discrete dynamic model presented in figure 3.4 is the base of the preload prediction. The BSD system was simplified by springs and dampers [29]. The quality of such prediction systems highly depends on the accuracy of the underlying model and its closeness to the real-world BSD [1]. The modes of the BSD system were especially activated when the motor changed its velocity rapidly. In this case, the modes of the BSD were dominant enough to become visible and distinguishable from those of other components [29]. When inspecting BSDs installed in big industrial machines, vibrations from several other components might disturb the machine signals, making extracting the axial natural frequency more challenging than in the simplified testbed. Furthermore, the working table mass, displacement and axial natural frequency must be estimated while the system is operational [29], which can be extra laborious for big industrial machines. Since the operational conditions of industrial machines change, these parameters can not be assumed to stay constant. Therefore, re-estimating the required parameters is necessary for each inspection. Only if that is done correctly will the established relationship between the axial natural frequency, the displacement of the working table and the preload of the BSD be reliable. During the experiments in the isolated and simplified BSD testbed, the system partially failed to detect expressive operational phases for estimating the axial natural frequency [29]. Implementing a trigger to find such phases in big industrial machines might be even more difficult. Nguyen et al. [29] emphasized that at least ten trigger events from each position along the BSD screw shaft are required to accurately average the FRF in practical applications. Collecting this large amount of trigger events involves substantial effort and time. Equally to the work of Lee et al. [7], the BSD nut was used to record the vibration signal. The extraction of the axial natural frequency from signals recorded by sensors in less fault-critical sensor locations was not tested. The physical degradation of the shaft and ball surfaces was neglected in the experiments.

3.1.3. Data-Driven Approach: Sensor Fusion of Multiple Hand-Crafted Statistical Features

Denkena et al. [2] presented a method to monitor the preload of BSDs. Firstly, several hand-crafted statistical features were extracted from the machine signals. Afterwards, each feature was rated by its robustness and statistical significance. The most promising features were selected and fused based on the principal component analysis (PCA). In the end, decision trees solved the classification task. A testbed consisting of a single BSD and linear guideways was used to evaluate the proposed approach. The preload loss was simulated by installing ball sets with different oversizes. The internal controller provided internal control signals and three uniaxial acceleration sensors measured the system's vibration. The machine data was recorded during a constant feed rate over the whole length of the test bench. The proposed PHM method can be separated into four steps:

- **Data acquisition:** Signals from three uniaxial acceleration sensors and internal control signals were processed simultaneously. The signals were separated into phases of zero acceleration (constant movement of BSD nut on the BSD screw shaft) and non-zero acceleration (direction change movement of the BSD nut at each end of the BSD screw shaft). In the experiments, the ball screw was moved over the whole length of the test bench with various feed rates (6000 mm/min, 11000 mm/min, 17000 mm/min, 20000 mm/min) [2].

- **Feature extraction:** Information about the preload classes were extracted through statistical features (e.g. kurtosis, median, impulse factor, ...). The features were applied to each signal and segment. At this point, the features were unrated. Each feature was evaluated by its robustness and statistical significance. The robustness was measured by the feature's dispersion around the median. After normalizing the feature with the z-score, the significance was estimated by the f-statistics:

$$\textbf{Z-score:} \qquad \tilde{x}_{i,j} = \frac{x_{i,j} - \bar{x}_i}{\sigma_i}, \tag{3.12}$$

$$\textbf{F-statistic:} \qquad f = \frac{\sum_{j=1}^{J} i \cdot (\bar{x}_j - \bar{x})^2 / (J-1)}{\sum_{j=1}^{J} \sum_{i=1}^{I} i \cdot (\bar{x}_{j,i} - \bar{x}_j)^2 / (J \cdot (I-1))}, \tag{3.13}$$

  where $x_{i,j}$ denotes the feature value j of a sample of class i, $\bar{x}_i$ is the feature mean value of class i, s_i is the feature standard deviation of class i and $\bar{x}$ is the overall feature and class mean value, I and J are the number of all classes and features. Features were seen

as eligible for the diagnosing system if the dispersion around the median was smaller than $\pm$ 10 and the f-statistic was higher than a critical value of 10 [2].

- **Principal Component Analysis:** PCA is a method to reduce the dimension of data while retaining most of the information. Principal components are the directions in the feature space, along which the variation of the data is maximal. By using only a few principal components, each sample can be represented by a smaller number of variables [30]. By applying PCA, the selected features were merged with the goal of maintaining the robustness and increasing the f-statistics [2].
- **Classification:** Decision trees were used to predict the preload class based on the extracted features. Due to its low classification effort and good traceability, decision trees are suitable for that classification task. For each signal and segment, a separate decision tree was used [2].

Conclusion

In order to find features that work well for predicting the BSD's degradation status, Denkena et al. [2] extracted 1500 features from the signal segments. Extracting, evaluating and fusing this amount of features can become computationally expensive. The performed experiments showed that the suitability of the features strongly depended on the machine's working conditions, classes and signals. The following observations proved these dependencies:

- **Working Conditions**: The features were not able to properly separate the classes on those vibration signals recorded with feed rates higher than 11000 mm/min [2].
- **Classes**: Certain features, like MAX, RMS and CRE, were only able to separate some of the defined preload classes [2].
- **Signals**: The robustness and statistical significance of some extracted statistical features depended on the signals [2].

The sensitivity to the machine's working conditions, classes and signals shows the system's low robustness, which makes its real-world use questionable. Furthermore, the features did not show proportional behavior to the physical degradation. When the features MAX and RMS were applied to the acceleration signals, they increased from C3 through C2 to C1, but decreased towards C0 [2]. Often deep learning is criticized for its black box principle and lack of understanding of the extracted features. Even though one generally understands the hand-crafted features, the meaning of the extracted information strongly depends on the task and signal. The correlation between those features and the physical degradation is often unknown, which makes such features an equal black box to the features extracted by deep

learning models. Generally, one can say that finding a set of features that reliably predict all BSD preload classes demands substantial effort. These features work reliably for specific signals, working conditions and preload classes. If any of those specifications change, the entire system must be completely reconfigured. The physical degradation of the shaft and ball surface was neglected in the experiments [2].

3.1.4. Data-Driven Approach: Multi-Level Feature Selection of Multiple Hand-Crafted Statistical Features

Li et al. [3] developed a prognosis system for BSDs, which simultaneously performed fault diagnosis, early diagnosis, health assessment and remaining useful life (RUL) prediction. Since this thesis focuses on the fault diagnosis of BSDs, this chapter is restricted to the processing units related to diagnosis. A testbed was built containing a BSD and horizontal guideways fixed to a concrete base. Instead of manually inserting faults to simulate degradation, it was caused by field use. Figure 3.6 shows the different processing steps and the parallel prediction units for the fault diagnosis, health assessment and remaining useful life (RUL) prediction. Vibration data was recorded by three accelerometers, speed and torque signals were retrieved from the controller [3].

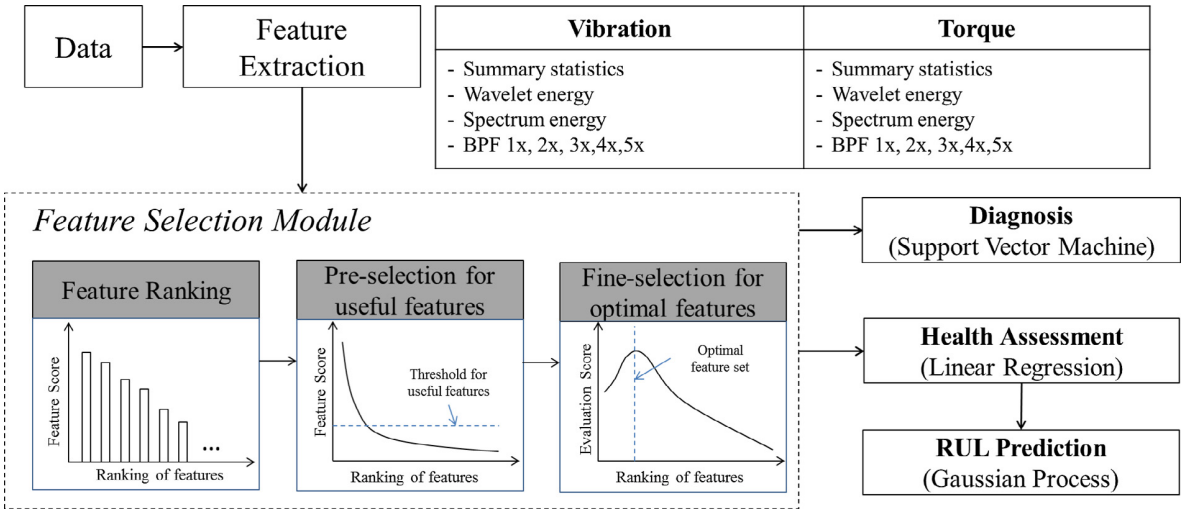

Figure 3.6.: Feauture extraction and selection for health diagnosis, health assessment and RUL prediction [3]

In the following, the three diagnosis steps of the proposed PHM system are presented in more detail:

- **Feature Extraction**: The system processed the signals in the time domain, frequency domain and time-frequency domain. The wavelet decomposition ('db4' wavelet) was applied to transform the signals. Similarly to Denkena et al. [2], features, such as RMS, mean, variance, kurtosis and skewness, were extracted from the signals in all three domains (time domain, frequency domain and time-frequency domain). Similar to the work of Lee et al. [7], the amplitudes corresponding to the calculated ball passing frequency and the ball screw rotation frequency, including its harmonics, were extracted as features for the monitoring system [3].

- **Feature Selection**: In a multi-level feature selection procedure, the extracted features were rated by their suitability for the prognosis task. This process was separated into three stages: primary feature ranking, pre-selection and fine-selection. In order to select expressive features, a hybrid strategy of filter-based (feature ranking and pre-selection) and wrapper-based (fine-selection) methods was applied. Wrapper-based methods rate features by the performance of a classifier using them and filter-based methods by the feature's intrinsic properties [31]. In the multi-layer feature selection procedure, the search space and search sequence corresponded to the selected features and their corresponding ranking scores found in the previous feature refinement phase. In the primary feature ranking phase, a selection criterion ranked and filtered all extracted features. In the pre-selection phase, the fisher score was applied to refine the previous feature choice:

$$S_c = \frac{\sum_{k=1}^{C} n_k (\mu_k^j - \mu^j)^2}{\sum_{k=1}^{C} n_k (\sigma_k^j)^2}, \tag{3.14}$$

  where C defines the number of classes, n_k the number of samples in k-th class, μ_k^j and σ_k^j the mean and standard deviation of the k-th class and the j-th feature and μ^j the mean of the j-th feature across all classes. For the multi-class classification task, the fisher score is biased. Therefore a fine feature-selection phase was added subsequently. Based on the prediction accuracy of an SVM using the pre-selected features, the final optimal features were chosen [3].

- **Classification**: A second SVM was applied to make the predictions for the diagnosis task. The SVM processed the optimal features found during the fine-selection phase [3].

Conclusion

Similar to Denkena et al. [2], Li et al. [3] extracted 440 features from the raw data. As mentioned before, the feature extraction and ranking can become computationally expensive when using that many features. Again, the expressiveness of the features strongly depended on the signals and the features did not correlate well with the degradation process [3]. In the multi-level feature selection, the SVM-based fine-selection and fisher score-based pre-selection diverged in their feature ranking. Especially the top features found during the pre-selection were rated down by the fine-selection [3]. Therefore, the application order of the feature selection methods significantly influenced the final feature choice. Denkena et al. [2] also mentioned that finding the empirical threshold for the fisher criterion in a nine-class classification task was challenging. When developing such feature selection modules, one has to select the single feature selection mechanisms, define their application order and find suitable parameters for each of them, which requires substantial effort. Generally, the definition of such feature selection modules strongly influences the performance of the PHM system and is highly dependent on the working conditions, health condition classes and signals. Minor variations in the predefined task specifications will most certainly reduce the performance of such PHM systems or could even make them fail. Using SVMs in the feature selection increases the training effort, which raises the training time and data requirements.

3.2. Domain Adaptation Approaches for Prognostic and Health Management

With the goal of developing robust monitoring systems for complex industrial machines, PHM systems need to become more flexibel, adaptable and intelligent. Inspired by the advances in the computer vision community, domain adaptation and transfer learning approaches have achieved greater popularity for industrial PHM.

3.2.1. Domain Adaptation Approaches for Prognostic and Health Management of Ball Screw Feed Drives

In the following, deep learning based domain adaptation models for monitoring the health condition of BSDs are presented. Similarly to the method proposed in this thesis, the presented models applied the MMD metric to measure and reduce the domain discrepancy in the latent feature spaces of the network.

Deep Learning Based Domain Adaptation Based on MMD-Loss

Azamfar et al. [5] proposed a deep learning based domain adaptation model for estimating the health condition of BSDs based on their preload level. The preload was considered a good indicator for estimating the health condition of the BSDs and guideways. An experimental test rig was built, containing a single horizontal guideway and a BSD fixed on a concrete base. Three accelerometers were installed to measure vibrations in the X and Y directions. These sensors were mounted on the BSD nut and the bottom and top attachments of the BSD screw shaft. A sound pressure sensor captured the acoustic level during the experiments. The torque and speed signals were acquired from the controller. Three different preload classes were defined for the guideways and BSDs. In the "normal" class the component was operating normally, in the "faulty level 1" class it was deviating from the healthy condition and in the "faulty level 2" class it needed to be replaced or repaired. In total, nine combinations of guideway and BSD degradation classes were defined. Data was recorded by performing a full cycle of BSD operation, containing two full forward and backward strokes along the guideways. The signals were split in phases of constant and changing BSD velocity. The training was restricted to those segments with constant BSD velocity, which were fed to the model as single input samples. The data dimension was reduced by a simple down-sampling method. The recordings included BSD operations with different BSD velocities (200, 400 and 600 mm/s). A domain shift was created by training and testing the model with data of varying BSD velocities. The proposed method was evaluated in the nine-class classification task, including all combinations of BSD and guideway degradation classes. The proposed neural network architecture is presented in figure 3.7. It contained a CNN of four alternating 1D convolutional and max-pooling layers and a subsequent classifier. To prevent overfitting, dropout layers with a rate of 0.3 were included. ReLU activation functions were used throughout the network. The proposed model optimization included a source CE-loss to improve the classification accuracy on the source domain data. Besides that, the domain discrepancy was reduced by an MMD-loss, which was applied in the penultimate fully connected layer [5].

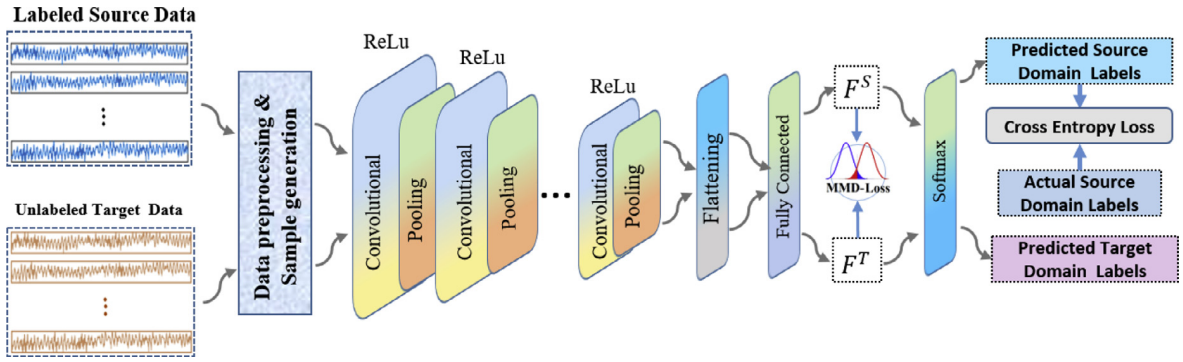

Figure 3.7.: Model architecture of deep learning based domain adaptation model for PHM of BSDs using an MMD-loss [5]

Deep Learning Based Domain Adaptation Based on MMD-Loss and PD Alignment

Similar to Azamfar et al. [5], Pandhare et al. [4] proposed a deep learning based domain adaptation model for estimating the health condition of BSDs based on their preload level. A similar test rig, as the one presented by Azamfar et al. [5], was used to evaluate the proposed models. In total, five accelerometers were mounted on the testbed. Two triaxial ones were placed close to the BSD nut, which is a promising position to represent the signature of the ball screw preload level. Three single-axial ones were mounted at the bottom and top attachments of the BSD screw shaft and on top of the load carried by the BSD nut. These sensor positions are more suitable and practical installations. Identical to Azamfar et al. [5], nine combinations of BSD and guideway preload classes were defined. By using data recorded with sensors mounted at different positions in the BSD testbed, a domain shift between the training and testing dataset was created. Pandhare et al. [4] tried to find an indirect sensing method to make PHM independent of impractical sensor locations. The proposed model architecture is presented in figure 3.8. It contained a CNN of two alternating 1D convolutional and max pooling layers and a consecutive classifier. The proposed model training included three losses. Again, a source CE-loss was used to improve the classification performance on the source domain data. An MMD-loss and PD alignment reduced the domain discrepancy between the training and testing dataset. The MMD-loss reduced the marginal and the PD alignment the conditional distribution discrepancy between the domains. The PD alignment matched source and target samples of the same class and reduced their L2-distance:

$$L_p = \frac{1}{n_p} \sum_{k=1}^{n_p} |h_k^{p,s} - h_k^{p,t}|_2, \tag{3.15}$$

where $h_k^{p,s}$ and $h_k^{p,t}$ are the k-th source and target domain samples and n_p is the used subspace of labeled source and target domain samples. The PD alignment was restricted to some of the nine classes and 20% of the training samples were used as PD samples. The MMD-loss and the PD alignment were restricted to the penultimate fully connected layer [4].

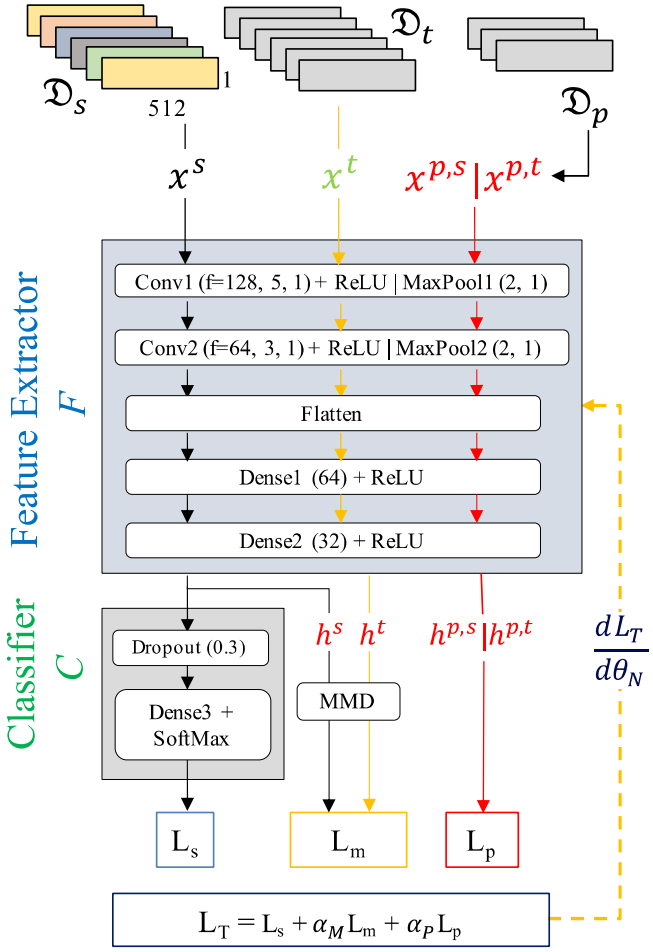

Figure 3.8.: Model architecture of deep learning based domain adaptation model for PHM of BSDs using an MMD-loss and PD alignment [4]

Conclusion

The disadvantages of the presented approaches by Azamfar et al. [5] and Pandhare et al. [4] are collectively presented in this section. Both approaches applied a regime separation to split the signals into phases of constant and changing BSD velocities [4] [5] . This extra step adds additional complexity to the data pre-processing. The segments with constant BSD velocity were fed to the models as single samples [4] [5]. Since those samples capture

the frequency and amplitude variations during the whole steady-state phase of the BSD operation, they were assumed to be more expressive for the monitoring task [5]. By doing so, only a few samples were collected from each recording. Experimental effort was required to record enough samples to train the neural network properly. Azamfar et al. [5] and Pandhare et al. [4] evaluated their monitoring approaches solely on a simplified testbed. Therefore, the evaluation of their approaches did not consider the mutual influence of the components installed in real-world industrial machines. Additionally, both approaches were restricted to predicting BSD preload forces. Other damages like pitting were ignored by the monitoring systems [4] [5]. In both cases, the MMD-loss was solely applied in the last fully connected layer. The domain discrepancy reduction was not evaluated in any other latent feature maps of the model [4] [5]. Azamfar et al. [5] created a domain shift by recording data with different BSD velocities and Pandhare et al. [4] by recording data with accelerometers mounted at different positions in the BSD testbed. In both cases, the domain shift was not generated by any differences on the physical component level. The same physical components were represented in both domains and therefore, in the training and testing dataset as well [4] [5]. The PHM systems only had to deal with domain shifts created by variations in the BSD operational conditions and data recording. It was not evaluated how the presented approaches react to physical variations in the systems. The PD alignment approach, presented by Pandhare et al. [4], requires target labels from some of the considered classes. This simplifies the health condition monitoring task and is rather more impractical for real-world PHM systems.

3.2.2. Domain Adaptation Approaches for Prognostic and Health Management of Rolling Bearings

A PHM system for rolling bearings was presented by Li et al. [32]. In a preprocessing step, a FFT transform was applied to represent the raw vibration signals in the time-frequency domain. As visualized in figure 3.9, the proposed model contained a CNN and a consecutive classifier. Max-pooling layers were included to reduce the model size. Batch-normalization layers reduced the internal covariate shift by normalizing the input distributions of the hidden layers. To prevent overfitting, dropout layers with a rate of 0.5 were included. The proposed method was evaluated on a rolling bearing dataset provided by the Bearing Data Center of Case Western Reserve University. A domain shift was generated by using testing data, which was exposed to additional environmental noise and collected under different working conditions. Ten health conditions with faults in different locations and with varying extents were defined.

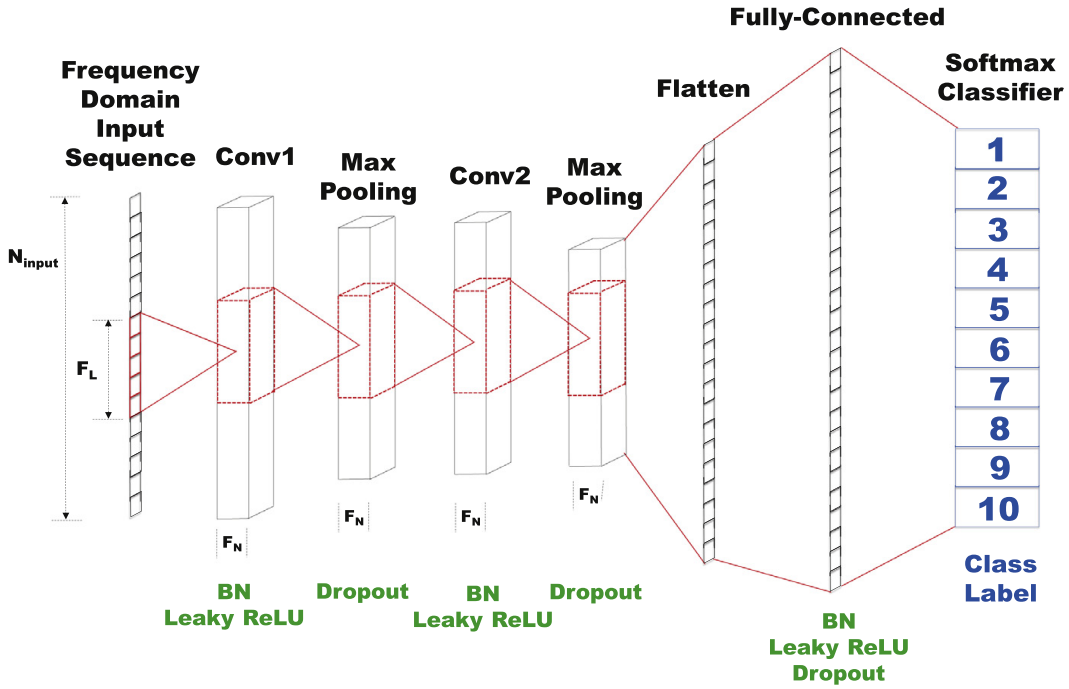

Figure 3.9.: Model architecture of deep learning based domain adaptation model for PHM of rolling bearings using an MMD-loss and deep distance metric learning [32]

The proposed system optimized the inter- and intra-class distance in the latent feature space. The distance between the source samples was minimized if they belonged to the same class and maximized otherwise. This increased the separability and compactness of the source domain classes, which made the algorithm more robust against environmental noise. In order to calculate the intra- and inter-class distances, the expectation and variance of the source domain samples belonging to the same class were measured in the feature maps:

$$\begin{aligned} D_{inter} &= |E[f^{(m)}x^{(i)}] - E[f^{(m)}x^{(j)}]|_2 - \sqrt{Var[f^{(m)}x^{(i)}]} - \sqrt{Var[f^{(m)}x^{(j)}]} \\ D_{intra} &= \sum_{i=1}^{N_{class}} \sqrt{Var[x^{(i)}]}, \end{aligned} \tag{3.16}$$

where N_{class} is the number of classes, $x^{(k)}$ denotes the raw input sample of class k, $f^{(m)}x^{(k)}$ denotes the feature representation of this sample in the m-th layer and $E[f^{(m)}x^{(i)}]$ and $Var[f^{(m)}x^{(i)}]$ are the corresponding expectation and variance. The inter- and intra-class distance were optimized with the following loss: $J_{Cluster} = -D_{inter} + \eta D_{intra}$. In addition to the distance metric learning, the discrepancy between the source and target domain was reduced by an MMD-loss applied in several fully connected (FC) layers. Lastly, the source

CE-loss in the final layer optimized the model to classify the source samples correctly. In total, the network was trained with the following weighted average of losses:

$$J_{total} = \alpha J_{Cluster} + \beta J_{MMD} + \gamma J_{CE}, \tag{3.17}$$

where $J_{Cluster}$ is the loss, optimizing the distances between the source domain samples, J_{MMD} the MMD-loss, J_{CE} the source CE-loss and α, β and γ are the weights for calculating the weighted average [32].

Conclusion

There are several MMD-based domain adaptation approaches for PHM of rolling bearing [16] [32] [33] [34] [35]. Generally, BSDs and rolling bearings are related components. The screw shaft of BSDs can be seen as the inner and the corresponding nut as the outer ring of rolling bearings [7]. In both cases, balls between those two components allow a rotatory motion around a fixed axis. Other than rolling bearings, BSDs also translate this rotatory motion into a linear one. The degradation of rolling bearings and BSDs is related in some sense. However, PHM systems for rolling bearings cannot be relied on to work as well for BSDs. Still, the research in this domain offers interesting applications and details for the PHM of BSDs.

Even though the MMD-loss was applied in several layers, it was still restricted to those of the classifier. Furthermore, only noise and different operational conditions created the domain shift but the same physical components were used in the training and testing dataset. Other than the work of Azamfar et al. [5] and Pandhare et al. [4], the system was able to monitor different degradation types.

3.3. CNN-Based Domain Adaptation in Computer Vision Applications

Aljundi et al. [36] presented a domain adaptation approach for computer vision applications. The method analyzed the domain shift effects in all convolutional layers of the neural network. Strongly affected feature maps (bad filters) were identified and reconstructed by less affected ones (good filters). Additionally, the domain discrepancy of the feature maps was measured by the Kullback-Leibler (KL) divergence. The feature maps' influence during the reconstruction process was punished accordingly. During the optimization, one filter map was considered at

a time. In each filter map evaluation, those feature maps should be identified which were able to reconstruct the response of the evaluated one:

$$B^* = argmin_B\{\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\Delta_j^{KL} \cdot \beta_j|\}, \tag{3.18}$$

where y_i is the output of the evaluated filter map for the sample i, $x_{i,j}$ the output of the feature map j for the sample i, β_0 the residual, $B = \{\beta_j\}$ the coefficients, estimating the suitability of each feature map to reconstruct the output of the evaluated feature map, n the number of source samples, p the number of layers, λ a tuning parameter to punish the coefficients and Δ_j^{KL} the KL divergence measured between the source and target data representations in layer j. When a layer's coefficient β_j was big, it was considered as good (small domain shift effects) and otherwise as bad (big domain shift effects). The optimization was solved by using the coordinate descent method. Afterwards, linear regression was applied to reconstruct the output of the bad layers by a weighted average of the good ones. The reconstructed layer outputs were then passed to the subsequent layers. Aljundi et al. [36] identified the CNN's early layers to be especially relevant to counteract the domain shift. In this context Aljundi et al. [36] also gave a great hypothetical example of how early layers of CNNs can generate domain shifts. If one imagines a neural network for recognizing facial expressions, early layers of the CNN usually detect things like colors and edges. When training such a model with young faces and testing it with those of old people, the model is confronted with wrinkles, which it has not seen during training. These wrinkles could distract the system to recognize those edges, which are relevant for identifying facial expressions. By adapting those early layers, which are responsible for edge detection, to be less prone to wrinkles, the domain shift could be reduced efficiently [36].

Conclusion

The presented approach by Aljundi et al. [36] disagrees with most domain adaptation approaches, which reduce the domain discrepancy in task-specific layers but use a shared CNN backbone across all domains. The work shows the positive effect of reducing the domain shift in the early layers of the CNN [36]. There is no clear pattern of which layers contribute most to the domain discrepancy problem [36]. Therefore, it is a great approach to evaluate each layer individually and adapt them adequately to reduce the domain shift. The work shows, that early layers in the CNN already suffer from and contribute to the domain shift [36]. Often the computer vision community offers advanced solutions for complex research questions which were not intensively evaluated in real-world scenarios. For the PHM of BSDs such solutions can be taken as inspiration.

3.4. Research Gap

When industrial machines run over long periods of time, varying operational conditions change the fault characteristics in the machine data [5]. Due to the mutual influence of the machine's submodules, fault characteristics are often complex and highly dependent. Developing hand-crafted features, as they are quite common in traditional approaches, requires a lot of expertise and human labor [1]. Due to the lack of flexibility and robustness, hand-crafted features struggle to extract expressive information from the machine data when the corresponding fault characteristics vary over time [1]. Model-based PHM systems, like those presented by Lee et al. [7] and Nguyen et al. [29], predict the BSD's health condition based on simplified physical models. The quality of those approaches highly depends on the exactness and sophistication of the underlying models and the corresponding parameters. If the models miss details from the real-world machines or the underlying processes, the PHM performance will be unsatisfactory [1]. Data-driven PHM systems, like those proposed by Denkena et al. [2] and Li et al. [3], use shallow machine learning classifiers in the final steps of the processing chain. These classifiers learn to use the information extracted by the hand-crafted features to make accurate predictions [1]. This intelligent way of combining the knowledge retrieved by the hand-crafted features might make those systems less prone to minor variations in the fault characteristics. Since the classifiers automatically learn the correlation between the hand-crafted features and the health condition classes, adapting such systems to new circumstances could be less time-consuming. Nevertheless, the system's performance highly depends on the underlying training dataset. It is unlikely that the training data includes all operational conditions and fault scenarios. It can even happen that fault classes are unknown during training. Optimizing machine learning models with limited data, which does not represent the data distributions during testing, could lead to a low diagnosis performance [5].

Robust PHM systems, designed to reliably handle fault characteristic variations, will bring industrial health monitoring to the next level [37]. In order to achieve that, PHM systems can be extended with domain adaptation modules. This thesis should investigate the applicability and utility of deep learning based domain adaptation for health condition monitoring of BSDs. The advantages of the proposed system over regular deep learning based systems should be evaluated. It requires a lot of work to establish accurate physical models and expressive hand-crafted features, which capture the degradation of BSDs [1]. For this reason, it is difficult to compare the developed approach to any traditional model-based or data-driven PHM system.

Most MMD-based domain adaptation systems for PHM tasks [4] [5] [32], apply the MMD-loss solely in task-specific layers. In the PHM context, the literature rarely discusses variations

in this application of the MMD-loss. Inspired by the advances in the computer vision community [22] [36], this thesis should evaluate different MMD-loss types and corresponding hyperparameters for monitoring the health condition of BSDs.

All presented approaches in chapter 3 were evaluated on a simplified testbed. In most of the experiments executed in this process, the degradation of the monitored components was just simulated and not caused by field use. The traditional PHM approaches presented in chapter 3.1 did not consider varying degradation levels of LGSs while predicting the health condition status of BSDs. Pandhare et al. [4], Azamfar et al. [5] and Li et al. [32] created a domain shift by recording data with different sensors of by retrieving data from varying operational conditions of the machine. However, in the training and testing dataset the same physical components were represented. Furthermore, the models developed by Pandhare et al. [4] and Azamfar et al. [5] solely monitored the preload level of BSDs.

Since the lifetime of BSDs is shorter than that of LGSs, in industrial machines, the BSDs need to be replaced in shorter intervals than the LGSs. Thus, it is very common that BSDs and LGSs with different levels of degradation are combined in industrial machines [32]. This implies that the developed model in this thesis should be able to predict the health condition classes of BSDs independent of the degradation of LGSs. In real-world PHM tasks, only a limited number of observations of all health condition classes are available during training. Typically, the physical components available during training differ from those during testing. Therefore, a PHM system should be developed that is robust enough to predict the health condition of new and unseen components during testing. Additionally, it should investigated how a PHM system can learn to monitor different degradation types simultaneously. A strong domain shift should be included in the dataset to generate a challenging PHM task. The developed model should be evaluated on a real-world machine with BSDs and LGSs degraded by field use.

4. Research Questions

The thesis is centered around three main research questions, which are presented in the following. These research questions were defined beforehand to guarantee a structured development of the PHM systems. The questions were formulated based on common problems and challenges in the domain adaptation and the PHM community. At the end of this chapter, the research approach followed in this thesis is described.

4.1. Influence of the GAMMA Choice on the Domain Adaptation Performance

Since the source and target domains are correlated to some extent, the network itself can extract domain-independent features. The powerful CNN learned from the source domain can also increase the model performance on the target domain [22]. At the same time, features that are too sensitive to the source domain can reduce the model performance on the target domain [22]. To counteract that phenomenon, domain adaptation approaches can help to transfer knowledge learned from the source to the target domain [22]. However, one has to pay attention to not transfer noise or irrelevant information since this destroys the structure of the source and target domain data and makes the classification task even more difficult [22]. For this reason, it is essential to precisely balance the MMD- and source CE-loss. This thesis investigates the effects of different weighting factors, called GAMMA, on the model training.

4.2. Domain Adaptation Performance of the Labeled MMD-Loss

In addition to the MMD-based domain discrepancy reduction, Pandhare et al. [4] applied a PD alignment module, which specifically reduced the L2-distance between samples belonging to the same class but different domains. The PD alignment required target labels. Li et al. [32] presented a PHM algorithm, which optimized the source domain inter- and intra-class distance while reducing the domain discrepancy with an MMD-loss. Both approaches try to improve the classes' compactness and separability in the latent feature spaces while reducing the domain discrepancy. This enables a more accurate classification and improves the domain overlap in the latent feature space. A stronger domain similarity facilitates the extraction of

domain-invariant features. Inspired by those two approaches, this thesis developed a novel labeled MMD-loss, which minimizes the domain discrepancy between samples of the same class and maximizes the domain discrepancy between samples of different classes. Instead of applying a PD alignment in addition to an MMD-loss, this novel MMD-loss directly considers the labels of the source and target domain. Similar to the work of Pandhare et al. [4], the target labels are not used in the CE-loss. The advantages and disadvantages of the labeled MMD-loss over the unlabeled MMD-loss are further analyzed in this thesis.

4.3. Influence of the Latent Feature Space Choice on the Domain Adaptation Performance

Most domain adaptation approaches, just as the one presented by Azamfar et al. [5] and Pandhare et al. [4], reduce the domain discrepancy in the task-specific layers and use a shared CNN backbone across all domains. Throughout the neural network, the feature maps extract information with different levels of abstraction. In shallow, more global, and deeper layers, more task-specific features are extracted [36]. Often, it is assumed that early convolutional layers extract general information and act as simple detectors for things like color or texture [36]. However, the work of Aljundi et al. [36] shows that the layers of the CNN are responsible for specific characteristics in deeper latent feature space representations and thus the creation of a dataset bias. Since feature maps influence all subsequent ones, propagating the biased data through the neural network facilitates the domain shift [36]. Aljundi et al. [36] showed that the CNN layers significantly contribute to the increasing domain shift throughout the neural network. Aljundi et al. [36] mentioned that there is no clear pattern to which extent layers contribute to and suffer from the domain shift. Reducing the domain discrepancy in only task-specific layers might minimize but not completely eliminate it [22]. Inspired by the work of Aljundi et al. [36], this thesis investigates how applying the MMD loss in different model layers can improve the domain discrepancy reduction. In this context, a particular focus lies on the layers of the CNN.

4.4. Research Approach

This thesis investigates the ability of domain adaptation approaches to increase the performance of PHM systems. First, the developed approaches were pre-evaluated on a dummy dataset and afterwards tested on a real-world dataset. In the research community, simplified synthetically generated datasets are quite common to understand the underlying functionalities and mechanisms of new approaches and to pre-evaluate their applicability for a given

task. For the sake of completeness, it is necessary to test these approaches on real-world data. When this has been done correctly, statements about the approaches' robustness, applicability and utility can be made. Experiments on the mentioned datasets were performed to investigate the following topics:

- Visualization of the influence of MMD-based domain adaptation on the latent feature space representations of neural networks.
- Evaluation of the PHM performance gain due to MMD-based domain adaptation
- Investigation of the three formulated research questions in the context of MMD-based domain adaptation in industrial health condition monitoring.

The datasets, proposed model architecture and corresponding training strategy are described in chapter 5. The results from the experiments are presented in chapter 6.

5. Experiments and Methods

In the chapters 5.1 and 5.2 the synthetically generated dummy and real-world BSD datasets are presented. The generation of the datasets, the included domain shift and the resulting classification task are described. The proposed model architecture and the corresponding training are presented in the chapter 5.3.

5.1. Dummy Dataset

In the first step, different MMD-based domain adaptation approaches for PHM applications were evaluated on a synthetically generated dummy dataset. Such datasets are structured equally and show similar patterns as the corresponding real-world datasets. The complexity in such datasets and therefore, the difficulty of the corresponding tasks can be tuned arbitrarily. When analyzing new and unknown methods, dummy datasets help to visualize the data processing of the method. This gives deeper insight into the functionality of the method. Besides that, dummy datasets pre-evaluate the applicability of such methods for the corresponding real-world task.

In this thesis, a simple dummy dataset with a domain shift was created to investigate the ability of MMD-losses to facilitate the extraction of domain-invariant features in neural networks. Since one has to deal with irregularities, outliers and noise in real-world data, it is helpful to evaluate the MMD-loss on a dataset that is not disturbed by these effects. Similarly to PHM applications, the dummy dataset contained one-dimensional time sequences containing 1000 data points. In order to simulate a classification problem with two classes and two domains, the sequences were sampled from four cosine curves with characteristic amplitudes and frequencies. By adjusting the amplitude and frequency, the domain adaptation problem can be configured more or less difficult. The sampling process included certain randomness to allow variations between sequences of the same class and domain. For every sampled sequence, the characteristic amplitude and frequency were perturbed. This changed the underlying characteristic of each sequence. Besides that, noise was added to each of the 1000 data points. This is necessary to generate sequences similar to noisy real-world vibration signals. Exemplary samples for both classes and domains are shown in figure 5.1.

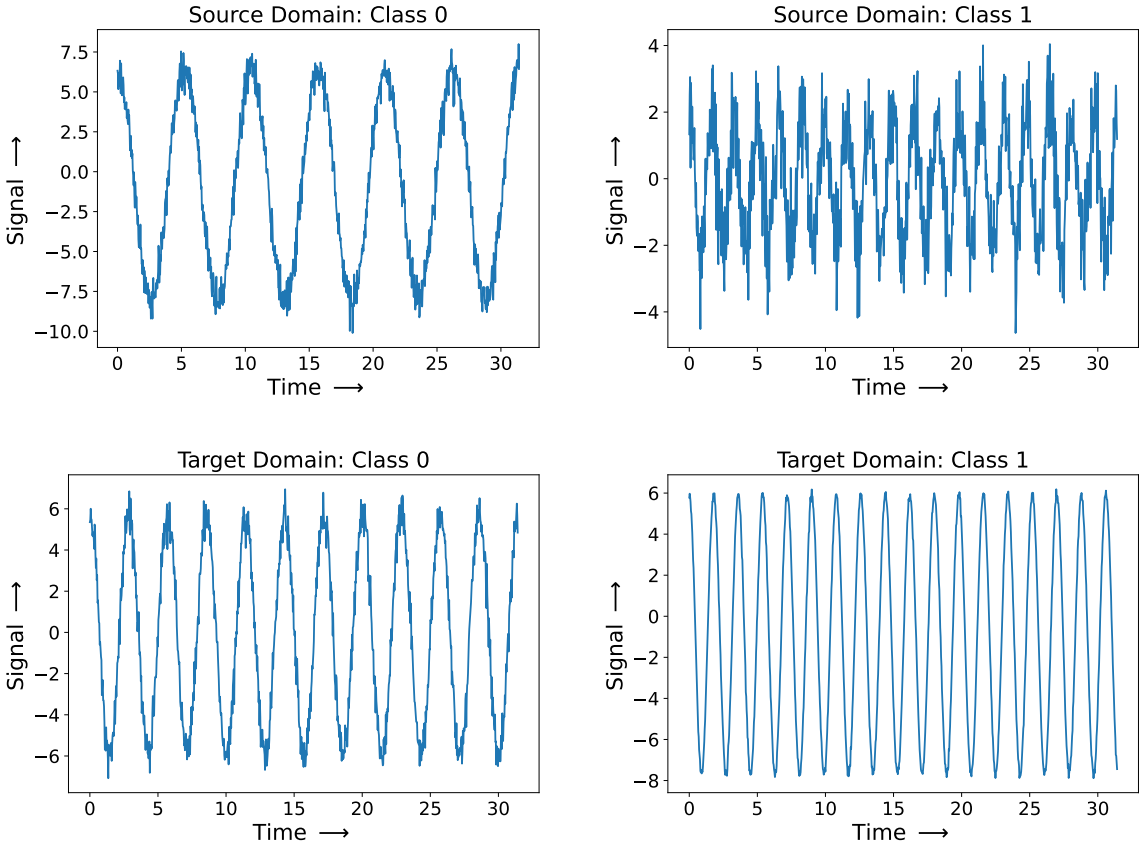

Figure 5.1.: Dummy data samples for all classes and domains

Figure 5.2 visualizes how the perturbation of the frequencies and amplitudes and the noise added to each data point, generate differences between samples of the same class and domain.

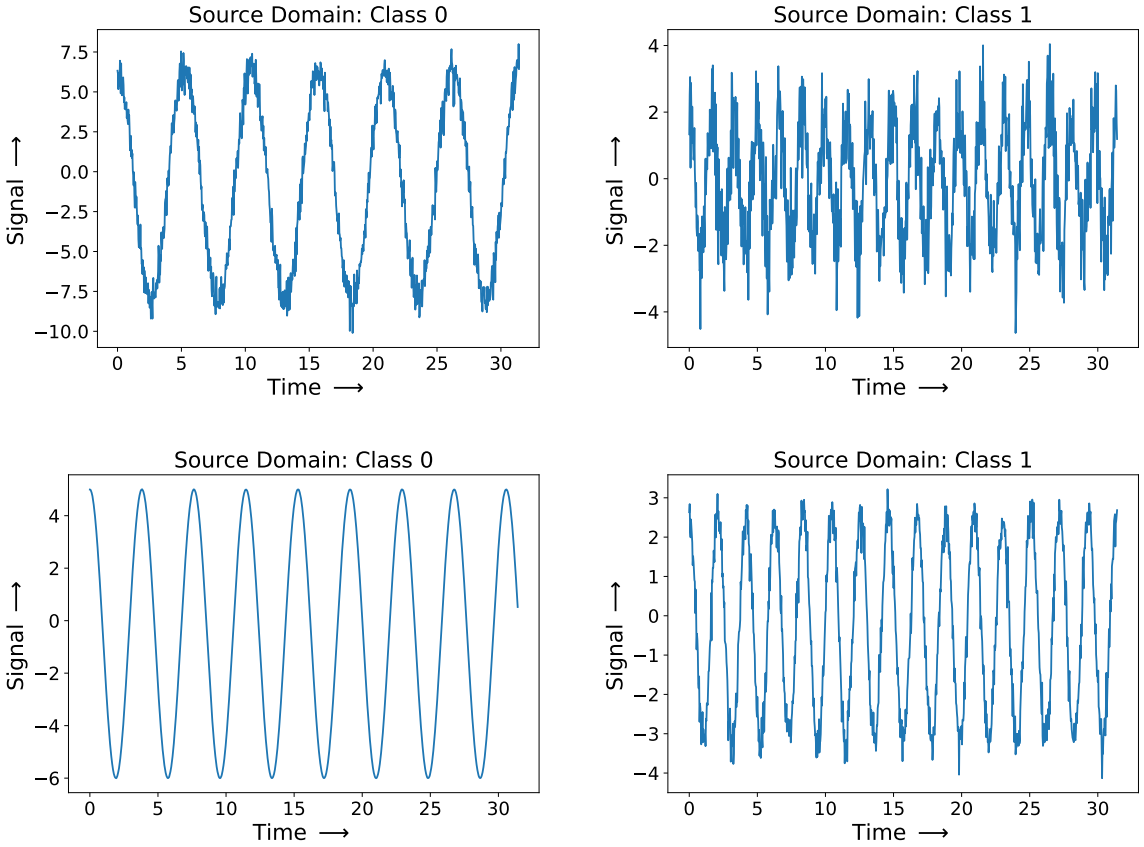

Figure 5.2.: Discrepancy between source samples of equal classes due to applied perturbation and noise during the sampling process

5.2. Ball Screw Feed Drive Dataset

Testing the developed PHM models on a real-world dataset is essential to make statements about the method's utility and applicability for industrial PHM tasks. Milling machines are common in the industry and contain several BSDs, which suffer from degradation. The continuous evaluation of the BSD health condition is therefore essential. A dataset was recorded on an industrial milling machine and the developed PHM models were evaluated by their ability to monitor the health condition of BSDs installed in that machine. In the following, the experimental setup, the mounted sensors, the classification task, the data recording procedure and the resulting PHM task, including a description of the domain shift problem, are presented.

5.2.1. Experimental Setup

Data from a DMG DMC 55H duoblock milling machine of the manufacturer DMG Mori was recorded. The machine tool's spindle, machine table, rotary axes, peripherals and the cladding were removed. The TNC control iTNC530 HSCI from Heidenhain GmbH was used. Figure 5.3 shows the experimental setup. The experiments focused on the moving hanger assembly of the machine tool. Generally, the machine tool can move in three spatial directions. The dataset only included data from the machine tool movement along the x-axis. For this reason, the single-threaded screw shaft of the BSD and the two LGSs responsible for that movement were supervised

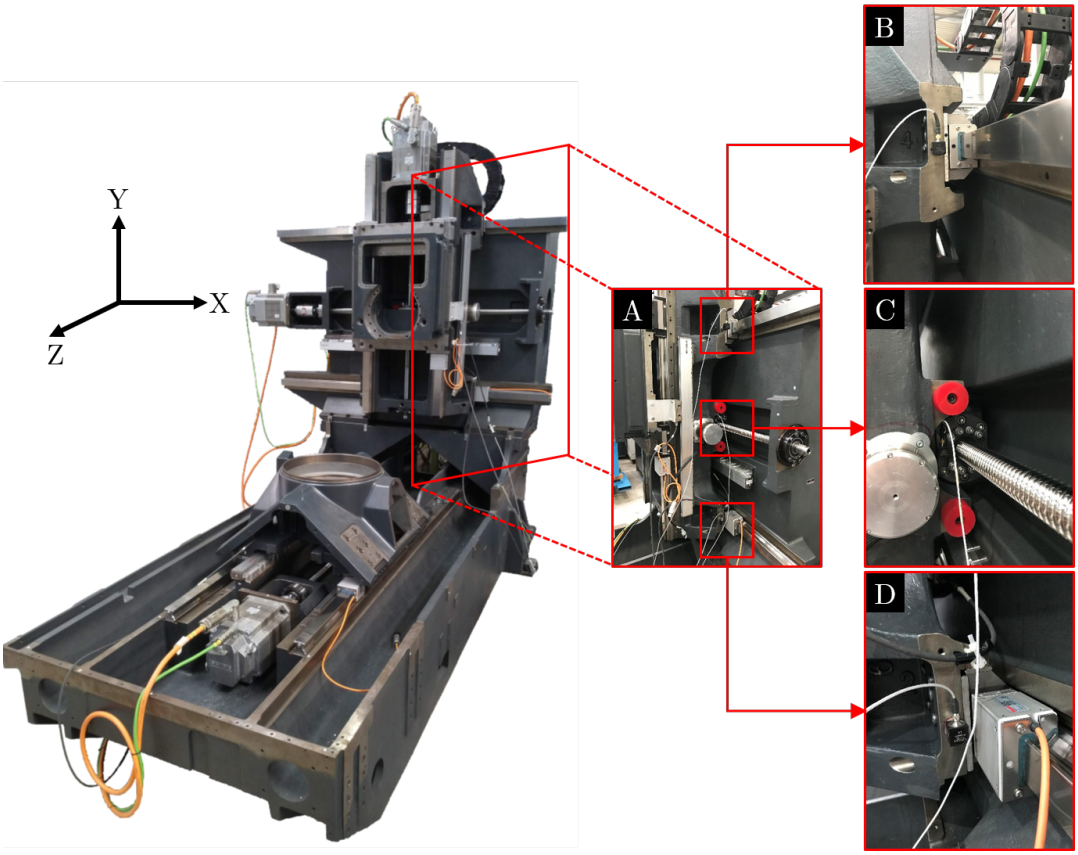

Figure 5.3.: Experimental setup: A: side-view of the machine, B: upper LGS, C: threaded screw shaft of BSD, C: lower LGS

5.2.2. Sensors

Signals from three different sources were included in the dataset. Three triaxial accelerometers recorded vibration signals and internal control data was accessed via the TNC Scope and

TNC Opt. The recordings were triggered by different software programs, which were synchronized by a python script. Minor time delays between the signals could not be eliminated completely. Figure 5.3 shows the placement of the three triaxial Kistler 8762A10 piezo-eletric accelerometers at the upper LGS (sub-figure B), the BSD nut (sub-figure C) and the lower LGS (sub-figure D). The software tool TNC Scope was used to access the internal control data of the machine. TNC Opt is a software tool intended to optimize controllers. During the experiments, it was used to access those control data channels, which differed from those accessible via the TNC Scope.

5.2.3. Definition of the Health Condition Classes

Preload classes (C1, C2, C3) were defined, specifying the preload level in BSDs and LGSs. For the BSDs also pitting damages were observed, which is why a separate pitting class (P) was defined. In total, four health condition classes were defined for the BSDs and three for the LGSs. The degradation of BSDs was specified by an ID containing one letter and either two digits for the preload classes or one digit for the pitting class. The degradation of LGSs was specified by an ID containing one letter and one digit. The letter indicates the damage types: no pitting (C) or pitting (P). The first digit in the BSD preload classes and the single digit in the LGS preload classes specifies the preload level. The second digit in the BSD preload classes and the single digit in the BSD pitting class specifies the observation. The distinct preload classes for the BSDs and LGSs were specified by the preload forces (N) measured in the differently degraded components. Low preload forces result from high preload losses, which indicate strong component degradation. This lowers the precision of the machine and might lead to a complete machine failure. Preload class C3 was labeled as “healthy”, C2 as "slightly degraded" and C1 as "strongly degraded". Two sets of BSDs and one set of LGSs, each containing components of all corresponding health condition classes, were used during the data recording. The observation digit was included in the BSD ID to separate equally degraded BSDs from different sets. In total, ten recordings were made for each BSD and LGS combination. Table 5.3 shows all 24 combinations of differently degraded and observed BSDs and LGSs. The numbers in table 5.3 define the specific BSD and LGS combinations and go up to 27. Originally the dataset included a third observation for the BSD preload class 3. In order to prevent a class imbalance, this observation was excluded from the experiments. The preload class definitions differed for the BSD observations. Table 5.1 and table 5.2 show all used BSDs and LGSs, the measured preload forces and the corresponding health condition class IDs. Each experimental setup included two LGSs, each consisting of two counterparts and one BSD. This is the reason why in table 5.1 each ID maps to one and in table 5.2 to four preload forces.

ID	Preload in N
P1	2 070
P2	2 160
C11	950
C12	845
C21	1 450
C22	1 293
C31	2 390
C32	2 328

Table 5.1.: BSD health condition classes

ID	Component	Preload in N
C1	C1	4 060
	C2	4 430
	C3	4 430
	F1	3 880
C2	B1	8 860
	B2	9 700
	B3	9 070
	E1	8 230
C3	A9	13 470
	A10	14 530
	A11	12 840
	D3	12 840

Table 5.2.: LGS health condition classes

		BSD							
		C31	C21	C11	P1	C22	C12	C32	P2
LGS	C1	1	2	3	4	5	6	7	9
	C2	10	11	12	13	14	15	16	18
	C3	19	20	21	22	23	24	25	27

Table 5.3.: Combinations of LGS and BSD health condition classes

5.2.4. Recording of the Dataset

For reproducibility, the experiments were executed with a fixed test cycle, defined in figure 5.4. The machine data was collected during constant speed, direction change and sweep excitation along the machine tool's X-axis. During the constant speed excitation, the machine tool was moved back and forth along the whole axis (Δx = 600mm). During the direction change excitation, the movement of the machine tool was restricted to a small part of the axis (Δx = 1mm) and the directions were changed with a high frequency. In the sweep excitation, the motor received a target speed in the form of a sine sweep. Before recording data, the machine was warmed up for 60min with a constant speed to create identical circumstances for all runs. Ten measurement cycles were executed and recorded, each with all three types of excitations. Between each recording the machine was reheated for 2 minutes.

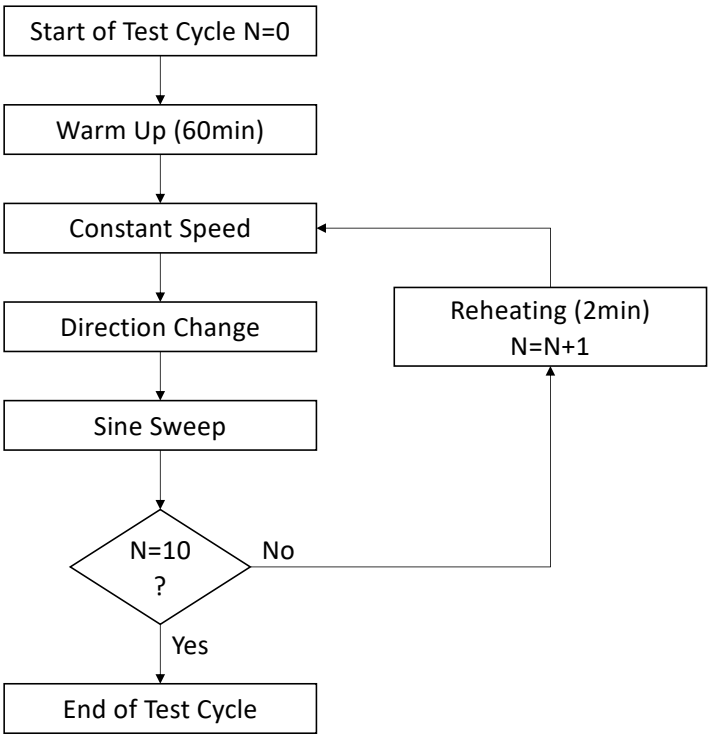

Figure 5.4.: Test cycle for data recording

In total 49 different signals were recorded. A more detailed specification of the signals can be found in the table. A.1 in the appendix.

5.2.5. Definition of the PHM Task

A binary health condition classification task was created by combining the BSD health condition classes C2 and C3 in a "healthy" and C1 and P1 in a "degraded" class. Therefore, the goal of the proposed model was to predict unacceptable extents of different degradation types (preload loss, pitting damage). A domain shift was generated by combining all samples recorded with BSDs from set 1 (observation digit 1) in the source and set 2 (observation digit 2) in the target domain. Since the measured preload forces differed for the two applied BSD sets (see table 5.1), a domain shift is guaranteed. Additionally, marginal differences in the production and installation of the BSDs could have increased the domain shift. All combinations of differently degraded LGSs and BSDs were represented in the source and target domain data.

5.3. Methods

The proposed model architecture and the corresponding training are presented in the following. All models applied to the dummy and real-world datasets had the same architecture. The model training partially deviated from the proposed one during the approaches' pre-evaluation on the dummy dataset. In chapter 6.1, the training variations are specified in more detail.

5.3.1. Model

The architecture of the proposed PHM model consists of an one-dimensional CNN and a subsequent classifier. A detailed visualization of the architecture is shown in figure 5.5. The CNN extracts expressive features, which are later used by the classifier to predict the health condition of the BSDs. The CNN consists of three convolutional layers. Throughout the network, the spatial dimension of the feature map is decreased and its depth increased. This helps to extract more global features in shallow and more specific and local features in deeper layers. The exact parameters of the convolutional layers are specified in table 5.4

Parameter	Conv 1	Conv 2	Conv 3
kernel size	100	10	5
padding size	0	1	1
stride	1	1	1

Table 5.4.: Parameters in convolutional layer

In order to reduce the spatial dimension of the feature maps, pooling layers are included after the convolutional layers. The resulting model complexity reduction prevents problems like overfitting and exploding gradients. Batch normalization is applied after the convolutional layers. For each batch, the means and variances of the layers' inputs are fixed, which makes the training faster and more stable. After iteratively applying these three types of layers, the output of the CNN is flattened and normalized to an one-dimensional vector. This vector is fed to the subsequent classifier. The latent feature space dimensions of the classifier are reduced constantly. The probability of the BSD health condition classes is predicted at the end.

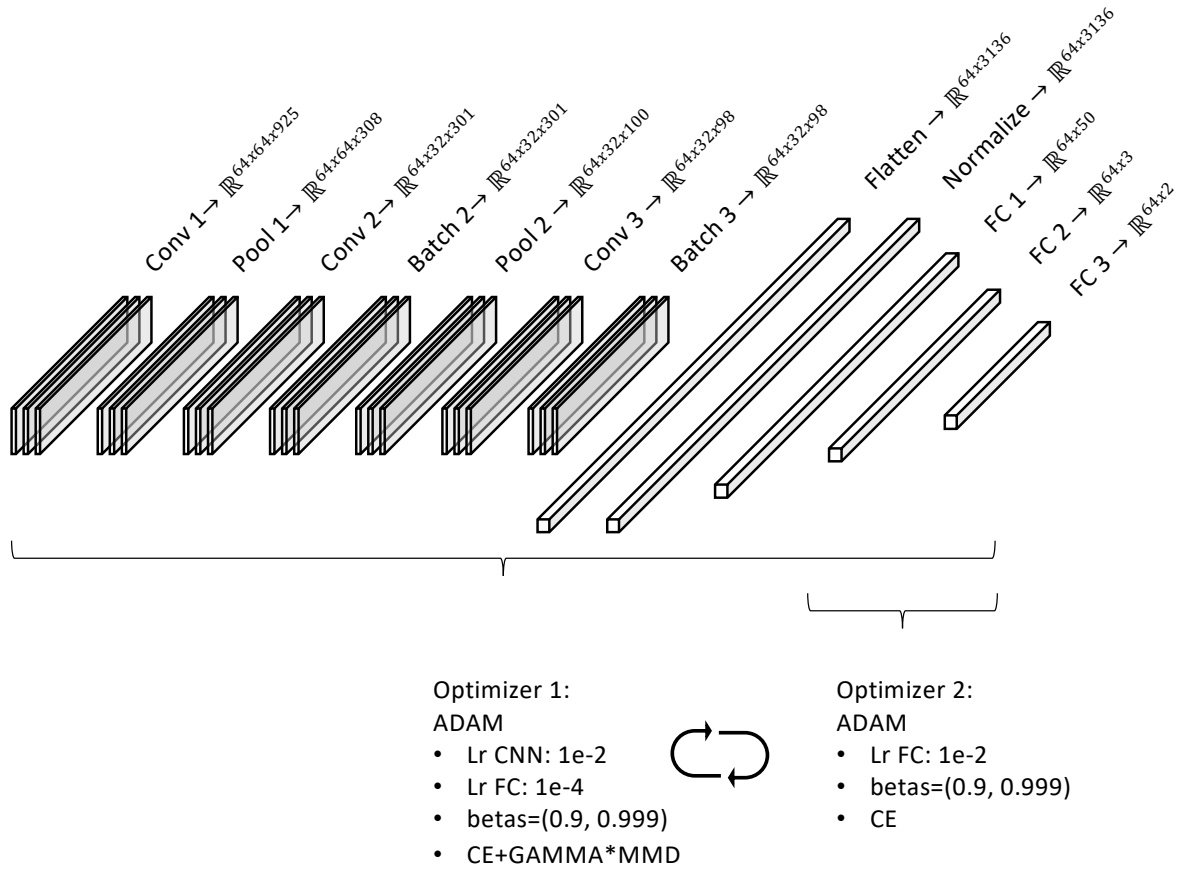

Figure 5.5.: Model architecture

5.3.2. Model Training

In the first step, the data used for the training is pre-processed. In this process, the dataloader separates the data into shorter sequences of length 1024. These windows, which can include several of the recorded 49 signals, are fed to the model as single samples. The sequenced signals are cleaned from Nan values and synchronized afterward. Separate source and target domain dataloaders split the corresponding datasets into four different parts: MMD-Training (40%), Source CE-Training (20%), Validation (20%) and Testing (20%). Other pre-processing steps, like wavelet transforms, can easily be added to the dataloader. The model training and testing is visualized in figure 5.7. The repetitive model training is separated into two phases. In the first phase, a weighted average of source CE-loss and MMD-loss is used to optimize the whole network:

$$\text{Total Loss} = \text{Source CE-Loss} + \text{GAMMA} \cdot \text{MMD-Loss}, \tag{5.1}$$

where GAMMA is a hyperparameter balancing the influence of the source CE-loss and MMD-loss. Different learning rates in the CNN and classifier allow the training intensity to be adjusted separately for the different modules. In this phase, an ADAM optimizer is applied with a learning rate of 1e-2 in the layers Conv1 - FC1 and a learning rate of 1e-4 in the layers FC2 - FC3. In the second phase, only the source CE-loss is applied to optimize the layers FC2 - FC3. Again an ADAM optimizer with a learning rate of 1e-2 is used. In both optimization steps, the beta values are 0.9 and 0.999. Two-thirds of the training data is used in the first and one-third in the second training phase. The application of the two different optimizers is visualized in figure 5.5.

The MMD-loss estimates the domain discrepancy in the latent feature maps of the neural network. The MMD-loss facilitates the extraction of domain-invariant features. The domain discrepancy is measured as the squared distance between the distribution kernel embeddings in the reproducing kernel Hilbert space (RKHS). The kernel choice is of great importance for the performance of the MMD-loss. By combining several RBF kernels with bandwidth parameters 1, 2, 4, 8, 16, the model training profits from their individual strength. The source and target samples are randomly coupled up and processed by the MMD-loss. The classes of these samples are not considered in the MMD-loss. Therefore, the MMD-loss minimizes the domain discrepancy between source and target domain samples of different and equal classes. The MMD-loss is applied in several layers of the CNN and classifier. The source CE-loss optimizes the model to increase the classification accuracy on the source domain data. Figure 5.6 symbolically shows how the MMD and the source CE-loss are applied in different model layers. The training is repeated until the maximum number of epochs is reached. After the training is completed, the model can be used to predict the BSD health condition classes of unseen target domain samples.

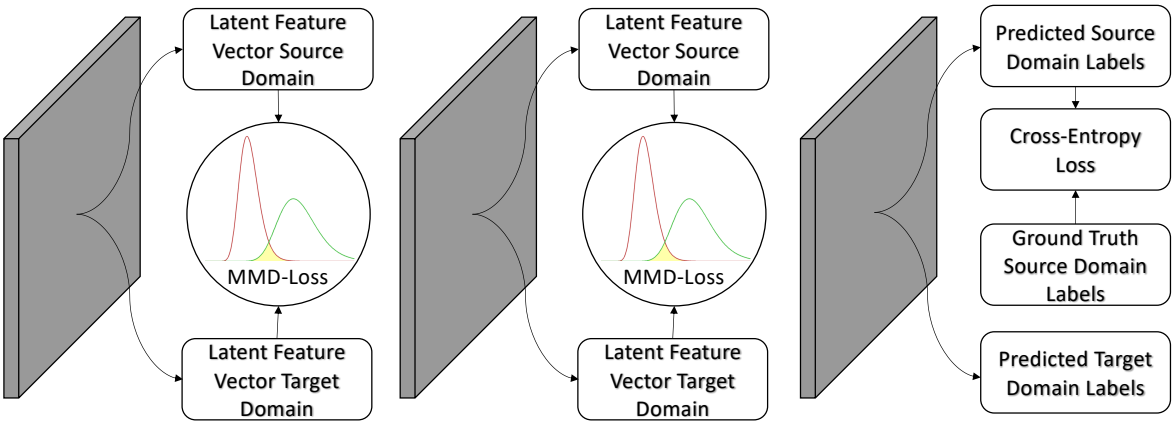

Figure 5.6.: Optimization of neural networks with a source CE- and MMD-loss

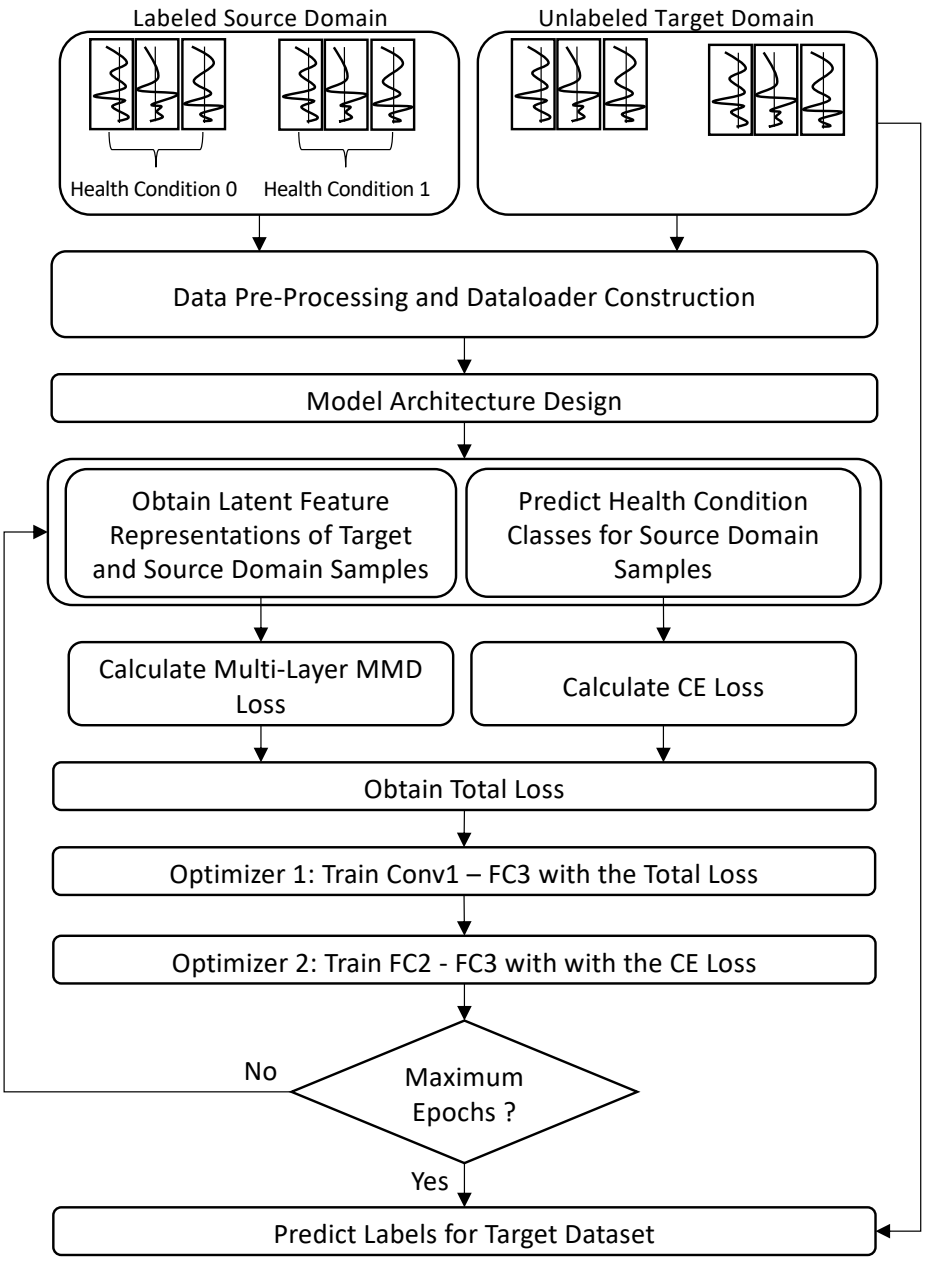

Figure 5.7.: Model training and testing

6. Results

The following presents the results from different experiments on the dummy and real-world BSD datasets. All defined research questions in chapter 4 were investigated on the dummy dataset. The results are presented in section 6.1. Excluding the labeled MMD-loss evaluation, these research questions were also analyzed on the real-world BSD dataset. The outcomes are discussed in section 6.2. The overall PHM performance of the proposed model is presented in chapter 6.3. In the last chapter 6.4, the experimental findings from both datasets are concluded.

6.1. Dummy Dataset

Several experiments were performed on the dummy dataset to analyze the effects of the MMD-loss in a simplified setting and to make initial estimates about its applicability for PHM tasks. The model architecture used in these experiments was the same as described in section 5.3.1. In sections 6.1.1 and 6.1.2, the influence of the GAMMA choice on the model training and the differences between the labeled and unlabeled MMD-loss were analyzed. In the experiments executed in this context, a single SGD optimizer with a learning rate of 0.01 was used. All layers were trained simultaneously with a weighted average of an MMD- and a source CE-loss. The visualization of the MMD-loss influence on the latent feature space representations became more visible when using that simplified model optimization. Chapter 6.1.3 investigates the application of MMD-losses in different latent feature spaces. In the corresponding experiments the models were optimized as explained in section 5.3.2. The MMD-losses were evaluated by the performance of the models trained with them. In order to evaluate the MMD-losses for the real application, the more sophisticated model training, which was also applied to the real-world BSD dataset, was used.

6.1.1. Influence of the GAMMA Choice on the Domain Adaptation Performance

The following section evaluates the influence of the hyperparameter GAMMA on the training. The FC2 latent feature representations of the source and target domain samples during different epochs are visualized in figure 6.1. The corresponding MMD-loss and source CE-loss development are shown in figure 6.2.

- **Small GAMMA (0.001)**: When a very small GAMMA was selected, the source CE-loss was reduced, whereas the MMD-loss was increased throughout the training (see figure 6.2). This showed the dominance of the source CE-loss. Instead of reducing the domain discrepancy, the model training focused on predicting the source samples correctly. The model's performance was solely improved on the source domain but the knowledge was not properly transferred to the target domain. Since the influence of the MMD-loss was negligible, the domain discrepancy was not reduced properly. Therefore, the separability and compactness of the classes were low and the class representations did not overlap properly across the domains (see figure 6.1).

- **Medium GAMMA (0.1)**: When the GAMMA was chosen correctly, the source CE- and MMD-loss were reduced simultaneously (see figure 6.2). For this GAMMA choice, the model was optimized to correctly classify the source domain samples while reducing the domain discrepancy. Knowledge learned from the source domain was successfully transferred to the target domain. In this case, the training profited from both losses equally. An optimization with multiple goals was achieved and neither of them solely dominated the training. Compared to the smaller GAMMA choice, the class distributions in the latent feature space showed increased compactness and separability of the classes in both domains (see figure 6.1). Especially for class 1, the point clouds were more dense and there were fewer outliers far away from their corresponding class center (see figure 6.1). The distance between different classes was increased and the subspace separating those was less dense (see figure 6.1). The point clouds of equal classes were structured more similarly and their overlap was improved (see figure 6.1). This showed that the samples were successfully transferred in a more domain-invariant feature space, where the domain discrepancy was reduced.

- **Large GAMMA (20)**: When a very large GAMMA was selected, the MMD-loss was reduced efficiently, whereas the source CE-loss was increased throughout the training (see figure 6.2). The correct prediction of source domain samples became irrelevant. Since the target labels were unknown, the MMD-loss was calculated between source and target domain samples of the same and different classes. Therefore, the MMD-loss reduced the inter- and intra-class distance between the latent feature representations of the source and target domain. The compactness of the classes was increased but their separability was reduced. This led to a trivial solution, where all latent feature representations collapsed to a point- or needle-like subspace (see 6.1). The classification task became more challenging for that specific latent feature representation.

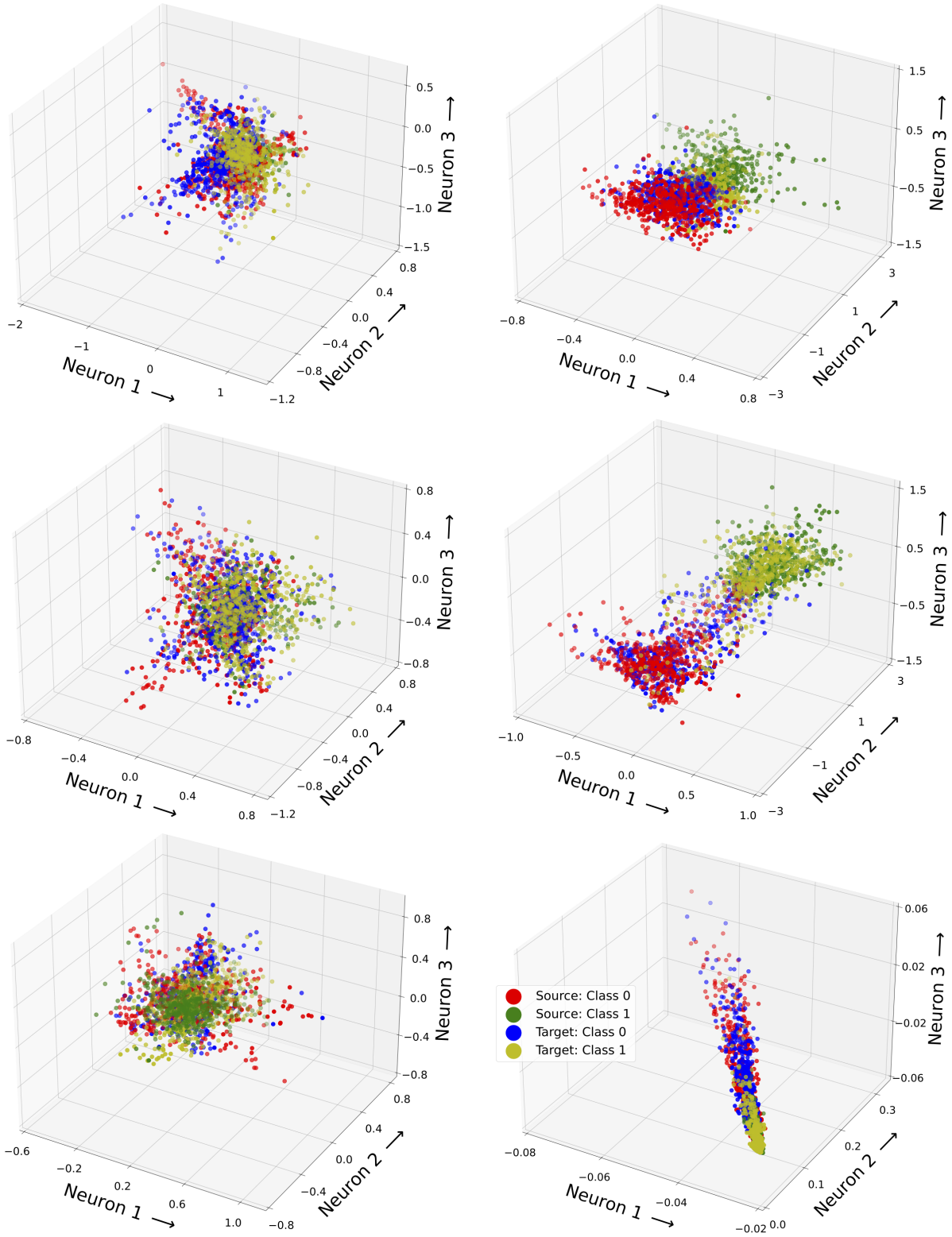

Figure 6.1.: Data distribution: Influence of the GAMMA choice on the model training, GAMMA = 0.001 (top), GAMMA = 0,1 (middle), GAMMA = 20 (bottom), Epoch = 0 (left) , Epoch = 8 (right)

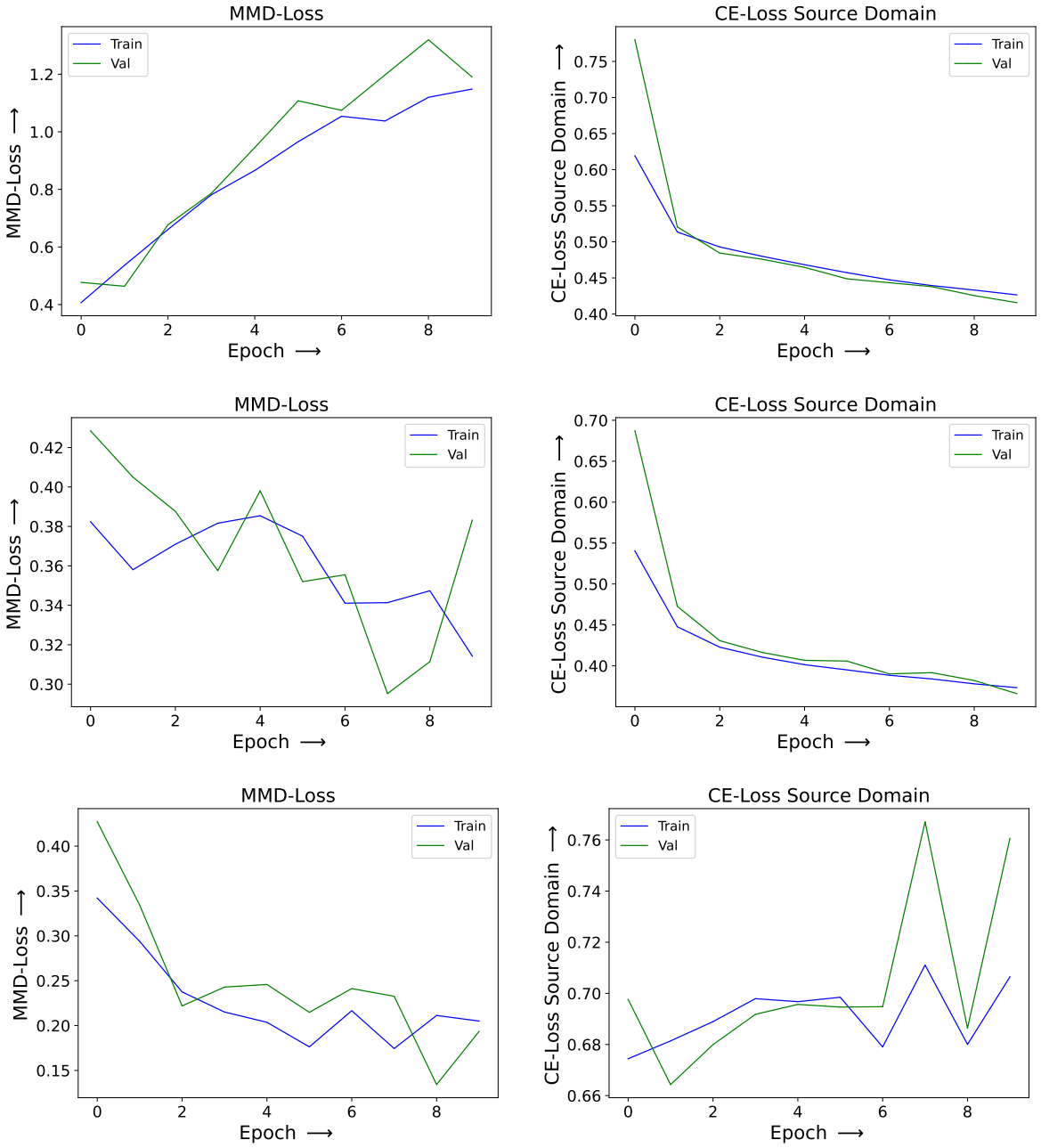

Figure 6.2.: MMD- and source CE-loss: Influence of the GAMMA choice on the model training: GAMMA = 0.001 (top), GAMMA = 0.1 (middle), GAMMA = 20 (bottom), Epoch 0 (left), Epoch 8 (right)

6.1.2. Domain Adaptation Performance of the Labeled MMD-loss

In the literature, some approaches, like the one proposed by Pandhare et al. [4], extended the unlabeled MMD-based model training with a distance-based optimization, which included target labels. A novel labeled MMD-loss was developed in this thesis, which separately optimizes the domain discrepancy between the source and target domain samples of similar and different classes. In the model training with the labeled MMD-loss type, the source CE-loss is still restricted to the source domain data.

Description of Labeled MMD-loss

The labeled MMD-loss minimizes the intra-class and maximizes the inter-class discrepancy between the domains. This enables a simultaneous improvement of the classes' separability and compactness. However, additional hyperparameters are required, which need to be defined beforehand. The hyperparameters GAMMA_Intra_Class and GAMMA_Inter_Class balance the maximization of the inter-class domain discrepancy, the minimization of intra-class domain discrepancy and the reduction of the source CE-loss:

$$\begin{aligned} \text{Total Loss} = &\text{GAMMA\_Intra\_Class} \cdot \text{MMD\_Loss\_Intra\_Class} - \\ &\text{GAMMA\_Inter\_Class} \cdot \text{MMD\_Loss\_Inter\_Class} + \text{CE\_Loss}. \end{aligned} \tag{6.1}$$

The MMD-loss, which includes target labels, is named "labeled MMD-loss" and the one restricted to the source labels is referred to as "unlabeled MMD-loss". Only target labels of 20% of the target training samples were used in the labeled MMD-loss.

Results of Labeled MMD-loss

Figure 6.3 visualizes the FC2 latent feature representations of the source and target domain samples during different epochs. Compared to the unlabeled MMD-loss, the labeled MMD-loss more efficiently reduced the domain discrepancy while increasing the separability and compactness of the classes in both domains. The following observations in figure 6.3 proved these advantages of the labeled MMD-loss over the unlabeled MMD-loss:

- **Separability**: The distance between the classes was substantially increased and the separating subspace was less dense.
- **Compactness**: The samples belonging to the same class were represented in a more distinct subset. The distance of the samples to their corresponding class center was reduced.

- **Domain Discrepancy**: The subsets corresponding to the same classes overlapped more across the domains.

These improvements, achieved by the labeled MMD-loss, simplified the classification problem. In a research aspect, the application of the labeled MMD-loss was particularly interesting as it revealed the deficits of the unlabeled MMD-loss. Due to the lack of target labels, the unlabeled MMD-loss equally minimizes the intra- and inter-class domain discrepancy. Therefore, the unlabeled MMD-loss is more prone to minimizing the classes' separability while reducing the domain discrepancy. When the effect of the unlabeled MMD-loss becomes too dominant, the separability of the classes is destroyed. In this case, the optimization ends up in a trivial solution, in which the latent feature representations of all samples collapse to a point- or needle-like subspace (see last row of figure 6.2). To prevent that, the GAMMA of the unlabeled MMD-loss must be selected precisely. Due to the separate optimization of the domain discrepancy between samples of equal and different classes, the model training with the labeled MMD-loss is less prone to the previously described trivial solution. Therefore, the influence of the labeled MMD-loss can be increased during the optimization. This enables a more efficient domain discrepancy reduction while improving the classes' separability and compactness. Due to the lower sensitivity to the GAMMA choice, GAMMA_Inter_Class and GAMMA_Intra_Class of the labeled MMD-loss were both chosen to be 30 in the experiments. A GAMMA of 0.1 had to be selected for the unlabeled MMD-loss to not reduce the separability between the classes while reducing the domain discrepancy.

6.1.3. Influence of the Latent Feature Space Choice on the Domain Adaptation Performance

This section analyzes how applying the MMD-loss in different latent feature maps of the CNN and classifier influences the model training. Inspired by Aljundi et al. [36], the CNN and FC layers were evaluated to reduce the domain discrepancy. Two different MMD-loss types were developed and afterward compared on the dummy dataset. Table 6.1 specifies the layers of the neural network considered by those MMD-loss types.

MMD-loss	Conv1	Conv2	Conv3	Flatten	FC1	FC2
FC MMD	-	-	-	✓	✓	✓
CNN MMD	✓	✓	✓	-	-	-

Table 6.1.: Overview of the latent feature maps included in the different MMD-loss types

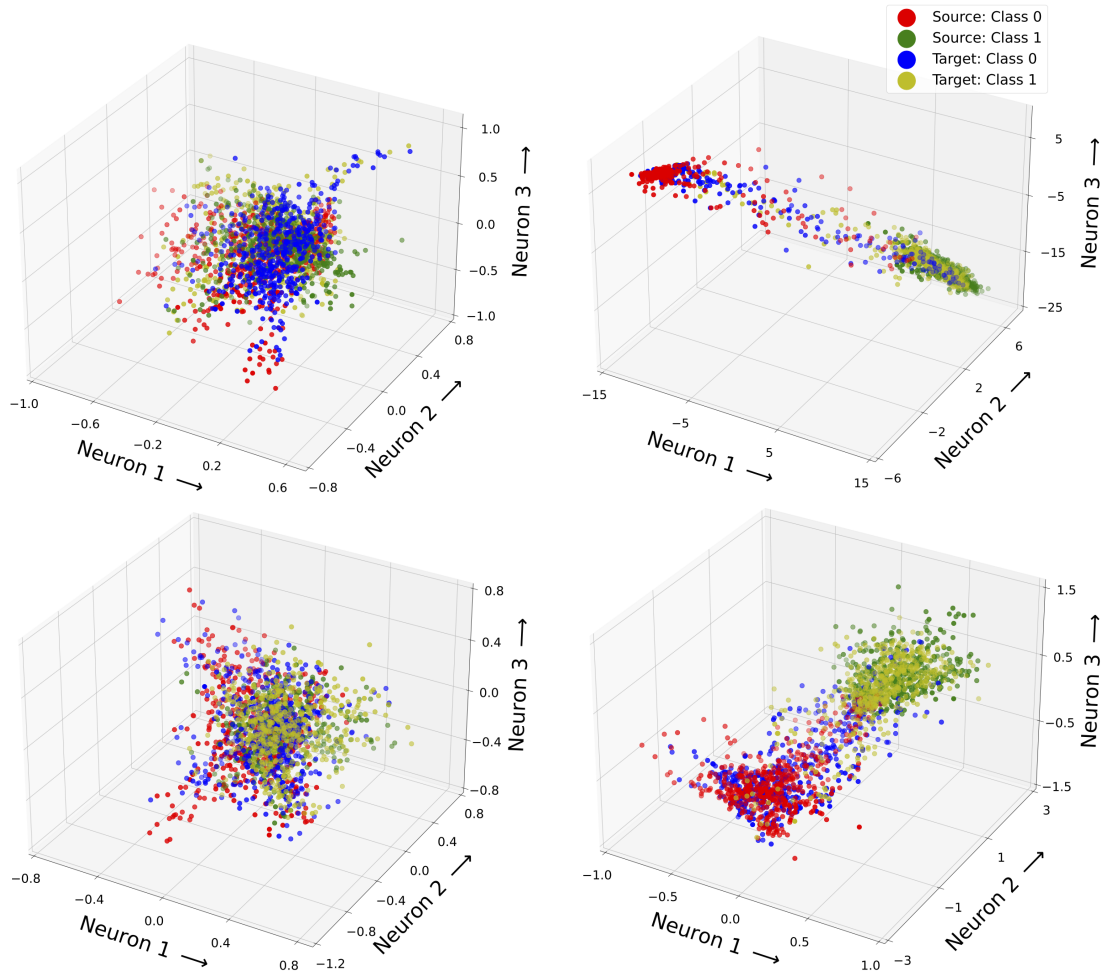

Figure 6.3.: Data Distribution: Labeled MMD-loss (top) vs. unlabeled MMD-loss (bottom): Epoch 0 (left) vs. Epoch 8 (right)

The development of the source accuracy, target accuracy, source CE-loss and MMD-loss throughout the executed model training are visualized in figure 6.4 and figure 6.5. When using the CNN feature maps in the MMD-loss, higher source and target accuracies were achieved (see figure 6.4). The increased classification performance in both domains indicated that the domain discrepancy was reduced more efficiently by the CNN MMD-loss. Additionally, the losses converged faster and smoother (see figure 6.5). This proves the increased training stability when applying the MMD-loss in the CNN layers. When estimating the MMD-loss in the fully-connected layers, it seemed like the two training goals of the MMD- and source CE-loss were contradicting. The model training appeared to be more prone to get stuck in local minima. This was mainly reflected by the training instabilities in the FC MMD-based model training. The resulting fluctuations are observable in the development of the source and target accuracies (see figure 6.4) and in the loss curves (see figure 6.5).

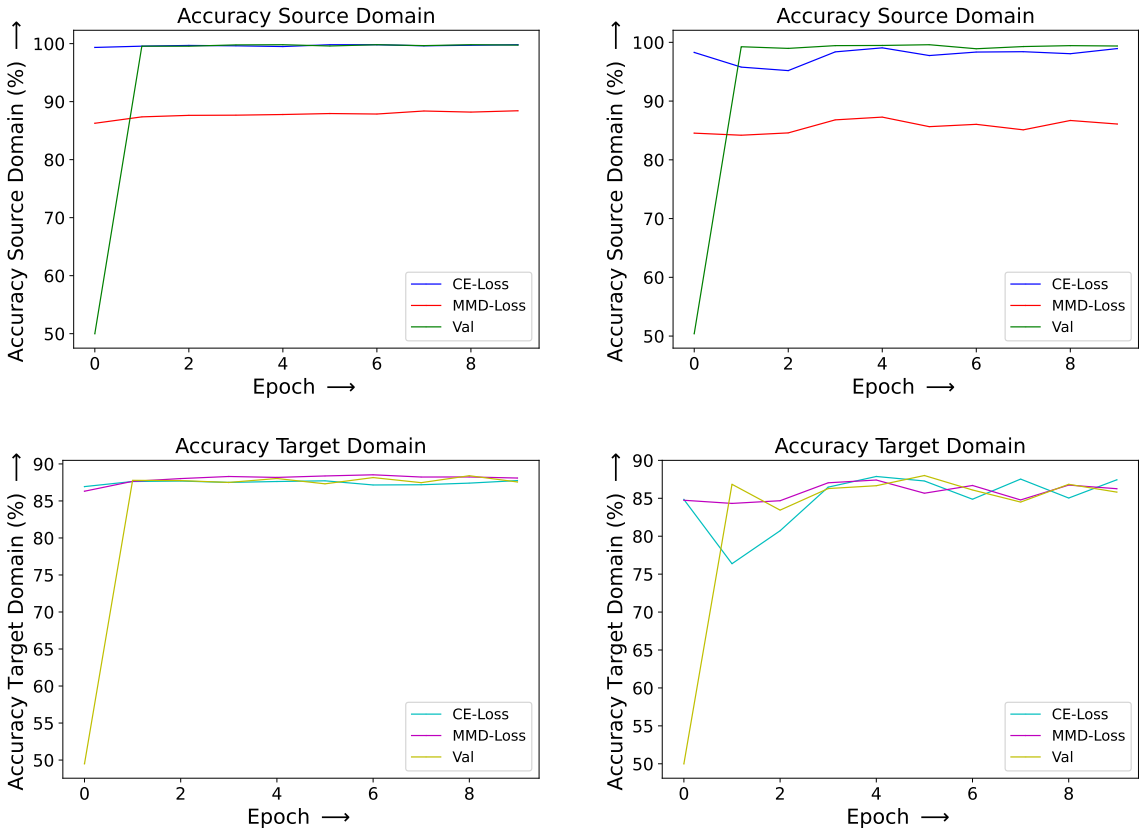

Figure 6.4.: Source and target accuracy: Influence of the MMD layer choice on the model training: CNN MMD-loss (left), FC MMD-loss (right)

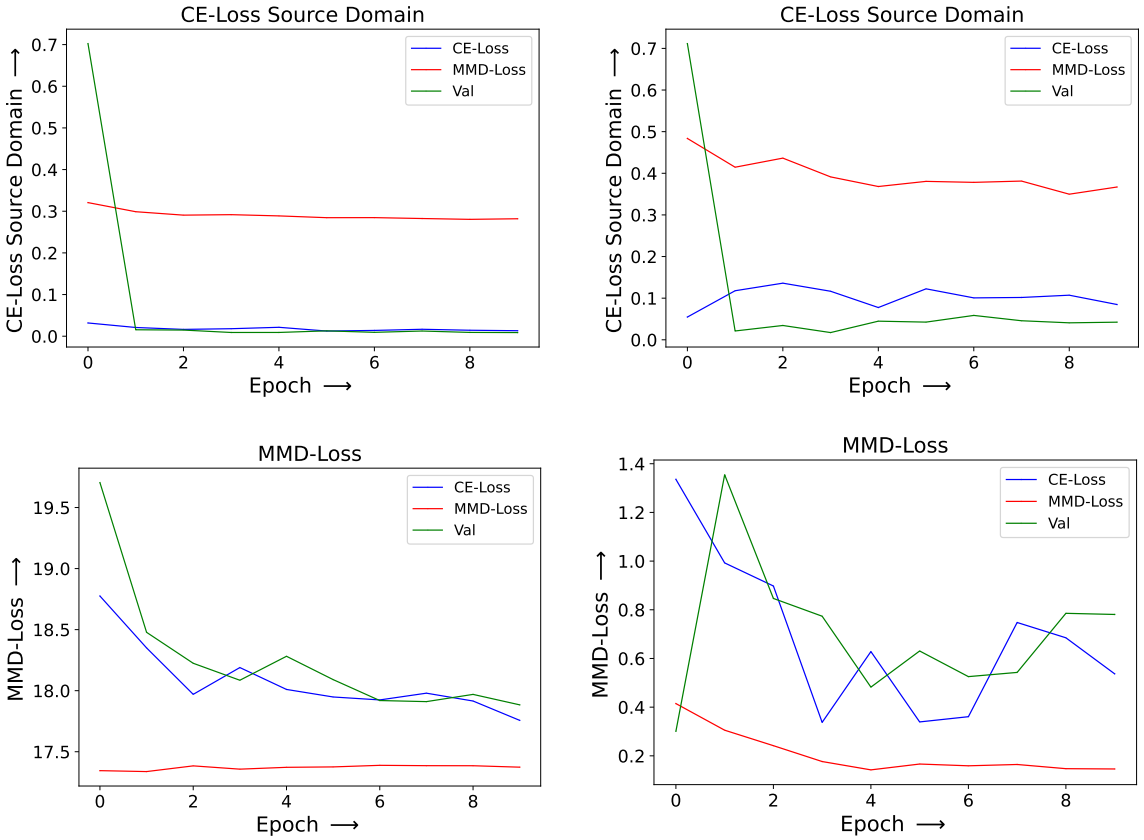

Figure 6.5.: MMD- and source CE-Loss: Influence of the MMD layer choice on the model training: CNN MMD-loss (left), FC MMD-loss (right)

6.2. Real-World Dataset

In the following, models trained with different MMD-losses are evaluated on the real-world BSD dataset. The goal of this chapter is to investigate the benefits and problems of MMD-based domain adaptation for PHM of industrial machines. All trained models had the same architecture and optimization strategy but differed in the applied MMD-losses. The evaluated MMD-losses differed in their GAMMA and latent feature space choices. A baseline model that did not apply any MMD-loss during the training was used to show the PHM performance gain due MMD-based domain adaptation. Table 6.2 specifies the latent feature spaces included in the different applied MMD-loss types:

MMD-loss	Conv1	Conv2	Conv3	Flatten	FC1	FC2
BASE-LINE	-	-	-	-	-	-
FULL MMD	✓	✓	-	✓	✓	✓
FC MMD	-	-	-	✓	✓	✓
CNN MMD	✓	✓	✓	-	-	-

Table 6.2.: Overview of the latent feature maps included in the different MMD-loss types

Similar to the dummy dataset, the influence of the GAMMA and latent feature space choice on the performance of the developed models are examined in chapters 6.2.1 and 6.2.2. For several reasons, the previously presented labeled MMD-loss was excluded from this evaluation on the real-world BSD dataset.

- The labeled MMD-loss uses target labels, which no other MMD-loss type in this thesis does. To achieve good comparability, only models with equal prerequisites and data access during the training are considered in this evaluation.
- Optimization methods using target labels are generally impractical for real-world use. In domain adaptation tasks, neural networks are first trained on the source domain and afterward are transferred to the target domain. If optimization methods require target labels, one could train the neural networks directly on the target domain. The necessity of target labels prevents the easy model adaptation by solely including the unlabeled target dataset in the model training. Labeling a dataset is often done manually and can be time-consuming.
- The labeled MMD-loss requires additional hyperparameters. Further experiments need to be executed to find those. The limited time in this thesis is also why the experiments on the real-world dataset are restricted to the unlabeled MMD-loss.

By comparing the performance of models trained with the different MMD-loss types in chapter 6.3, their applicability and utility for real-world PHM tasks are evaluated. In chapter 6.4, the results from the dummy and real-world dataset are concluded. All computations were performed on a Leibniz Supercomputing Centre7 compute node virtual machine with 20 Intel® Xeon® Gold 6148 vCPUs, one Nvidia® Tesla V100, 368 GB RAM, PyTorch v.1.4.0 and CUDA 10.1.

6.2.1. Influence of the GAMMA Choice on the PHM Performance

Three models were trained with the FULL MMD-loss and different GAMMAs (0, 0.05, 1) on the D:P mech./X signal for 100 epochs. Figure 6.6 shows the FC2 latent feature representations of the source and target domain samples during different epochs. Similar to the dummy dataset, the MMD-based model training was highly sensitive to the GAMMA choice. When GAMMA was selected precisely, the domain discrepancy was reduced, while guaranteeing sufficient compactness and separability of the source and target domain classes. When applying an MMD-based model optimization with a GAMMA of 0.05, the generated latent feature space FC2 after 100 epochs seemed to be more suitable for the classification task than that created by the baseline model training (GAMMA=0). This is proved by the following observations in figure 6.6:

- **Separability**: The structure of the data offered greater clarity and homogeneity. This raised the assumption of a less complex class separation.
- **Compactness**: Class 1 appeared to be more compact in both domains. All samples lay in a more distinct subspace with a decreased distance to the corresponding class center.
- **Domain Discrepancy**: The target domain samples of class 0 appeared to be less mixed with the source and target domain samples belonging to class 1.

However, from figure 6.6 it is difficult to compare the classes' separability, compactness and the domain overlap created by the baseline and MMD-based (GAMMA=0.05) model training. Nevertheless, the plots make the disadvantages of choosing a GAMMA of 1 very obvious. In this case, the MMD-loss was too dominant. As a result, noise and unimportant information were transferred between the domains. The MMD-loss reduced the distance between the source and target domain samples of all classes. This destroyed the class structure of the source and target domain. In this case, the feature representations of all samples collapsed to a small needle-like subspace, which made the classification task impossible.

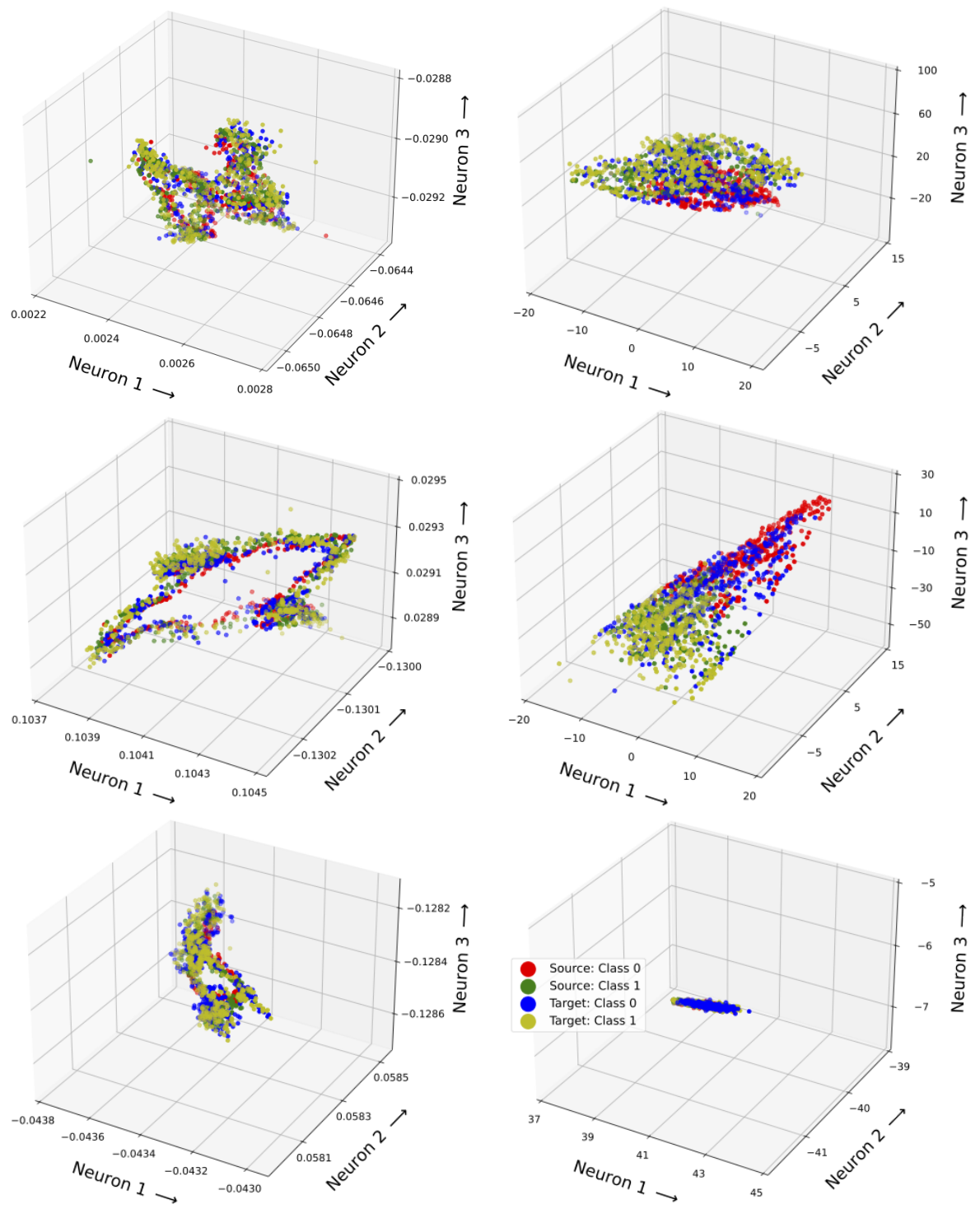

Figure 6.6.: Data distribution: Influence of the GAMMA choice on the model training: GAMMA = 0 (top), GAMMA = 0.05 (middle), GAMMA = 1 (bottom), Epoch 0 (left), Epoch 100 (right)

6.2.2. Influence of the Latent Feature Space Choice on the PHM Performance

This section analyzes the effects of including different latent feature spaces in the MMD-based model training. All models were trained with a GAMMA of 1 on the D:I soll/X signal for 100 epochs. Each model training was repeated five times in an equal matter. Figure 6.7 shows the corresponding target accuracy development during the executed training. When the MMD-loss was solely applied in the FC layers, the training collapsed in two of the five experiments (row 3, 5). In the other three cases, the accuracies had a slight tendency to decrease during the training. The CNN MMD-based model training showed worse reproducibility throughout the five experiments than the FULL MMD-based model training. Often, the fluctuations on the validation data were not reduced properly throughout the training (row 1, 4, 5). Especially towards the end of the training, the FULL MMD-based model training appeared to be more stable. Nevertheless, minor fluctuations were also observable for the FULL MMD-based model training (row 3, 4). In this thesis, the final model was selected based on its performance on the validation dataset of the target domain. In real-world scenarios, the target labels are usually unknown and the model at the end of the training is chosen as the final one. When the model performance decreases or fluctuates throughout the training, the best possible model is unlikely to be found. For this reason, a stable and converging model training is important for real-world applications. A more detailed description of how the final models are chosen in this thesis is given in chapter 6.3.

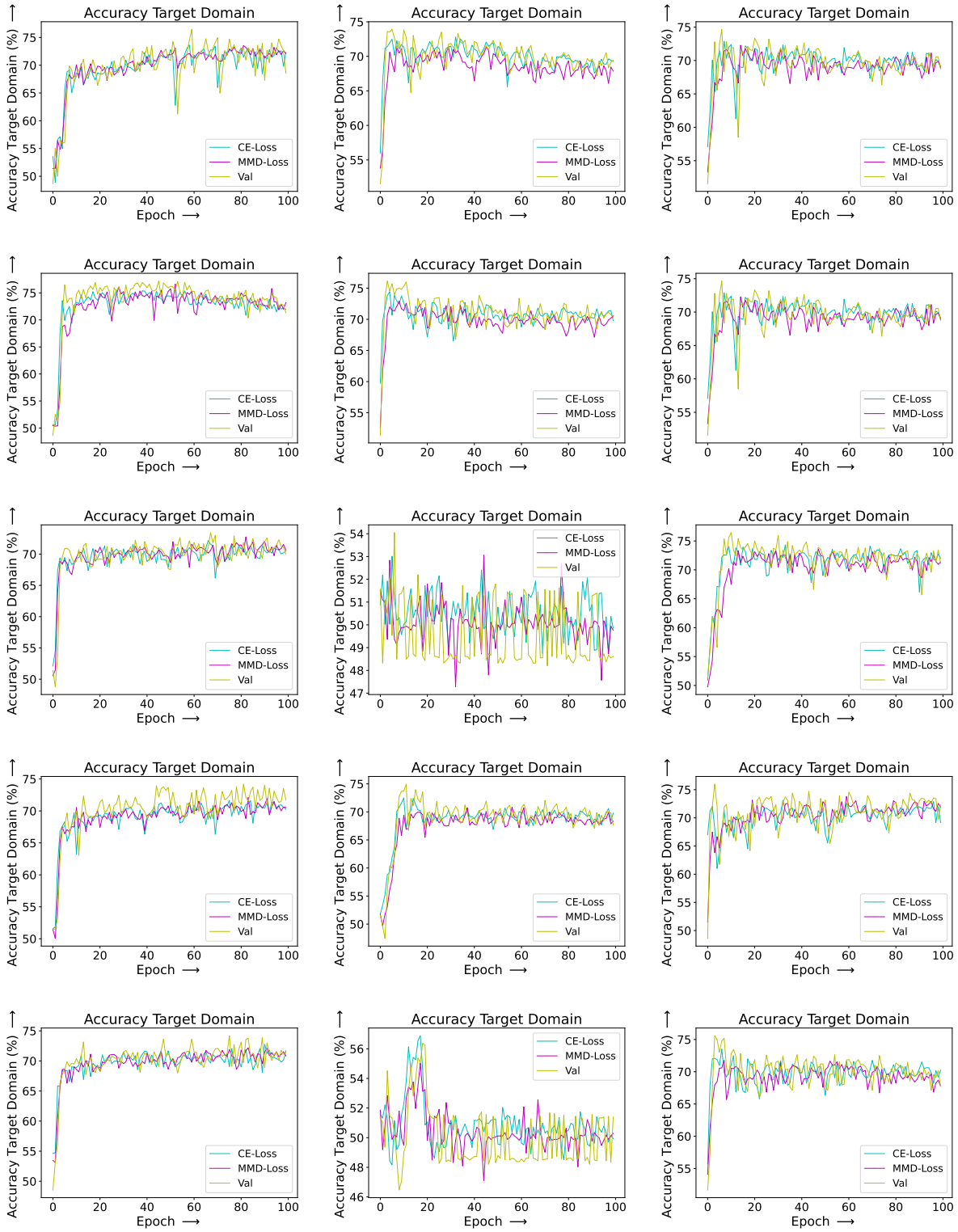

Figure 6.7.: Target accuracy: Influence of the MMD layer choice on the model training: CNN MMD-loss (left), FC MMD-loss (middle), FULL MMD-loss (right)

6.3. Overall PHM Performance

In this chapter, different MMD-loss types are evaluated on the real-world BSD dataset. This is done by comparing the performance of models trained with those MMD-loss types. The performance was measured by the models' accuracy on the target domain test dataset. The evaluation of the MMD-loss types required several stages of experiments. In the first stage, all of the 49 signals were evaluated by their suitability for PHM tasks. The models were trained on these signals without applying any MMD-loss. Based on the performance of the resulting models, seven promising signals were selected for further testing. In the second stage, the models were optimized with different MMD-based training strategies on those seven signals. The models were optimized with FULL MMD-, FC MMD- and CNN MMD-losses and three different GAMMA choices (0.05, 0.5, 1). All nine MMD-based model training (all combinations of MMD-loss types and GAMMAs) and the baseline model training that did not apply any MMD-loss were repeated five times on all seven signals. During each epoch in the model training, the models were evaluated based on the balanced accuracy on the target domain validation dataset and the best-performing model was stored. The different MMD-loss types were compared by the performance of those stored models on the unseen target domain test dataset.

For each training strategy (specific MMD-loss type and GAMMA choice), the accuracies achieved by the five equally trained models on the target domain test dataset were averaged. The results are shown in table 6.3. The FULL MMD models performed best on four of the seven signals and the CNN MMD models on the other three. The FC MMD models never outperformed the other MMD-based models. Compared to the baseline model, the MMD-based models increased the accuracy on the target domain test dataset by up to 10.18%. Table 6.4 shows the standard deviations corresponding to the averaged accuracies seen in table 6.3. The best-performing models usually showed low to average standard deviations, which verifies a high degree of reproducibility throughout the repeated model training. This demonstrates the relevance of the results and proves the applicability and utility of the corresponding MMD-losses for reducing the domain discrepancy in the PHM task. For all training strategies (BASELINE, FULL MMD, FC MMD, CNN MMD) the calculated standard deviations were averaged in the last column of table 6.4. For the MMD-based model training, the average standard deviations were calculated over all signals and GAMMAs and for the baseline model training over all signals. From the MMD-based models, the FULL MMD models had the lowest average standard deviation and the FC MMD models had the highest. This proves that the FULL MMD models achieved the most consistent performance throughout repeated training with different GAMMAs and on different signals. Therefore, the FULL MMD-based model training is more robust and less sensitive to GAMMA and signal

choices. As mentioned in the results of the dummy dataset, a reason for the sensitivity of the FC MMD-based model training to the GAMMA and signal choices might be the contradicting training goals when evaluating the MMD- and the source CE-loss solely in the FC layers. In this case, the training might be more prone to getting stuck in local minima, which leads to instabilities during the optimization. Furthermore, the results achieved by the different MMD-loss types strongly depended on the GAMMA choices. In the FULL MMD-based model training on the D:P_mech./X signal, varying GAMMA choices led to accuracy differences of nearly 30%.

6.4. Conclusion of the Experimental Results

In this chapter, the results from the experiments on the dummy and real-world datasets are interpreted and the corresponding findings are summarized. Variations in the functionality of different MMD-loss types and corresponding GAMMA choices became especially visible in the experiments on the dummy dataset. Performance differences between the MMD-loss types and GAMMA choices were evaluated on the real-world dataset. A novel labeled MMD-loss was developed, which considers the source and target labels. When applying the labeled MMD-loss, the domain discrepancy is reduced between the samples of the same class and increased between those of different classes. This guarantees improved compactness and separability of all classes while reducing the domain discrepancy. The experiments with the labeled MMD-loss on the dummy dataset highlighted the deficits of the unlabeled MMD-loss. Since the unlabeled MMD-loss has no access to the target labels, class-specific properties like the compactness and separability can not be optimized for the target domain. The unlabeled MMD-loss reduces the domain discrepancy between all samples without considering their class labels. If the unlabeled MMD-loss becomes too dominant, the separability of the classes is reduced and the data structure is destroyed. In this case, the latent feature space representations of all samples collapse to a point- or needle-like subspace. This makes the classification more challenging. For this reason, the GAMMA choice is highly relevant and has to be selected precisely and individually for each signal. The experiments on the real-world BSD dataset showed that varying GAMMA choices can lead to huge accuracy differences. Furthermore, the evaluation in this real-world PHM task demonstrated that the MMD-losses, applied in early CNN layers, reduce the domain discrepancy more efficiently than those restricted to the FC layers. This greater efficiency is mainly reflected in the overall model performance and the increased training stability. When applying the MMD-loss in the CNN layers, the model training is less sensitive to specific GAMMA and signal choices.

Model	GAMMA	D:I_ist/X	D:I_soll/X	D:P_mech./X	C:z_top	C:z_nut	D:x_nut	D:z_top
BASE-LINE	-	70.08	75.62	74.7	69.14	57.4	58.74	57.88
FULL MMD	0.05	71.56	75.42	**78.96**	72.36	58.10	49.52	**62,64**
	0.5	73.78	74.84	52.78	**75.92**	**64.76**	50.52	49,94
	1	72.98	75.18	49.80	74.12	57.02	50.62	50.52
FC MMD	0.05	73.60	76.38	74.98	71.36	57.48	52.44	61.3
	0.5	70.74	74.86	72.62	69.32	59.76	50.12	53.22
	1	73.42	65.46	76.96	62.34	58.86	50.92	53.96
CNN MMD	0.05	71.62	**76.44**	51.80	73.20	59.30	58.04	53.82
	0.5	**73.90**	75.86	51.58	72.28	55.76	68.04	51.72
	1	73.80	73.82	51.12	72.18	54.28	**68.92**	51.28
MMD GAIN:		+3.82	+0.82	+4.26	+6.78	+7.36	+10.18	+4.76

Table 6.3.: Average target test accuracy (%)

Model	GAMMA	D:I_ist/X	D:I_soll/X	D:P_mech./X	C:z_top	C:z_nut	D:x_nut	D:z_top	AVG STD
BASE-LINE	-	2.07	0.27	2.20	2.55	1.11	1.04	1.22	1.50
FULL MMD	0.05	1.06	0.74	**1.79**	1.80	2.04	1.39	**2.48**	1.81
	0.5	0.72	1.44	3.33	**1.72**	**1.42**	1.23	1.02	
	1	0.76	0.81	0.87	6.06	5.57	0.96	0.79	
FC MMD	0.05	1.96	0.61	1.86	1.82	1.63	3.86	1.60	3.39
	0.5	1.34	0.72	11.06	6.42	2.28	0.45	3.14	
	1	0.96	12.01	5.46	8.18	2.13	0.93	2.70	
CNN MMD	0.05	2.13	**0.53**	2.39	1.12	4.01	4.18	7.26	2.45
	0.5	**0.25**	1.57	1.08	1.22	3.27	3.51	3.22	
	1	0.51	1.25	2.02	2.51	4.54	**2.54**	2.34	

Table 6.4.: Standard deviation target test accuracy (%)

7. Conclusion

In this thesis, a deep learning based PHM system for monitoring BSDs was successfully extended with a domain adaptation module. A novel way of applying the MMD-loss in the layers of the CNN was proposed, which reduced the domain discrepancy efficiently. The proposed MMD-loss outperformed those restricted to the task-specific layers, which were mainly used in the literature. This was reflected primarily in the increased prediction accuracy and training stability but was also seen in the reduced sensitivity to different signal and GAMMA choices. The development of a novel labeled MMD-loss revealed the main deficit of the regular unlabeled MMD-loss. If the unlabeled MMD-loss became too dominant, the class separability in both domains was reduced, which made the classification task more challenging. Only when balancing the source CE- and MMD-loss properly, the domain discrepancy was reduced while maintaining or increasing the compactness and separability of the classes. For this reason, the GAMMA choice was identified as highly relevant for the PHM performance. It is suggested to select the GAMMA precisely and individually for each signal.

The developed models were evaluated on a dataset recorded on an industrial machine. Therefore, the mutual influence of different components of the machine was considered in the model evaluation. Additionally, the degradation of the BSDs and LGSs was caused by field use. In summary, the evaluation of the proposed PHM system was more realistic and further elaborated than in the works presented in chapter 3.

Furthermore, the developed model was able to predict the health condition of the BSDs independent of the LGS degradation. Compared to Pandhare et al. [4] and Azamfar et al. [5], different sets of BSDs were used in the training and testing data. Since the preload class definitions differed for both domains, the domain shift was significant. Moreover, Pandhare et al. [4] and Azamfar et al. [5] restricted their system to monitor the preload level of BSDs. The proposed system in this thesis learned features to monitor different degradation types. Pitting damages and preload levels were predicted simultaneously. In conclusion, one can say that the developed model shows robustness and can achieve good results in challenging real-world PHM tasks. The requirements for real-world PHM tasks, which were formulated in chapter 3.4, were fulfilled by the proposed model.

8. Outlook

In future work, the dataloader could be extended with more sophisticated pre-processing steps. Wavelet transforms or FFTs, could be applied to the raw input data to feed more expressive data to the neural network. In the literature, it is quite common to include such steps [3] [32] [35] [38]. Investigating to what extent such methods could increase the PHM performance would be very interesting.

In the experiments of this thesis, the best PHM results were achieved with the direction change excitation and the worst were accomplished with the sweep excitation. The excitations of the machine and, thus, the signals showing the machine's reaction to those differ fundamentally in their periodicity and constant BSD velocity phases. Even though the pre-processing effort might increase, it should be investigated how the adaptation and synchronization of the data windowing to the periodicity and constant BSD velocity phases of the machine's excitation can increase the PHM performance. Azamfar et al. [5] mention that the amplitude and frequency variations during a phase of constant BSD velocity deliver more expressive information about the machine's degradation. In the literature, regime separation strategies, which split the data into physical meaningful sequences, are widespread [5] [4].

Furthermore, new domain shift problems could be generated to test the proposed PHM system. Training and testing the model on differently degraded LGSs would be a realistic and challenging task.

Moreover, the application of the labeled MMD-loss on real-world data is worth being further investigated. A labeled MMD-loss that only uses a small amount of target labels might be beneficial for some PHM tasks. If labeling some target samples is acceptable, a more effective adaptation of PHM systems to changing fault characteristics might be achievable with the labeled MMD-loss.

There are several other approaches that should be considered for monitoring the health condition of BSDs. Recently, GANs became more popular for PHM of industrial machines [39]. The adversarial training of those models is promising for extracting domain-invariant features. However, the development and training of GANs is more complex. Especially when applying such models to noisy and disturbed real-world data, this might lead to training instabilities [39].

A. Description of Recorded Signals

Signal	Sensor	Frequency	Samples
C:s ist/X	TNC Scope	10 kHz	75000
C:s soll/X	TNC Scope	10 kHz	75000
C:s diff/X	TNC Scope	10 kHz	75000
C:v (n ist)/X	TNC Scope	10 kHz	75000
C:v (n soll)/X	TNC Scope	10 kHz	75000
C:P mech./X	TNC Scope	10 kHz	75000
C:Pos. Diff./X	TNC Scope	10 kHz	75000
C:I ist/X	TNC Scope	10 kHz	75000
C:I soll/X	TNC Scope	10 kHz	75000
C:x bottom	Acc	10 kHz	75000
C:y bottom	Acc	10 kHz	75000
C:z bottom	Acc	10 kHz	75000
C:x nut	Acc	10 kHz	75000
C:y nut	Acc	10 kHz	75000
C:z nut	Acc	10 kHz	75000
C:x top	Acc	10 kHz	75000
C:y top	Acc	10 kHz	75000
C:z top	Acc	10 kHz	75000
D:s ist/X	TNC Scope	10 kHz	75000
D:s soll/X	TNC Scope	10 kHz	75000
D:s diff/X	TNC Scope	10 kHz	75000
D:v (n ist)/X	TNC Scope	10 kHz	75000
D:v (n soll)/X	TNC Scope	10 kHz	75000
D:P mech./X	TNC Scope	10 kHz	75000
D:Pos. Diff./X	TNC Scope	10 kHz	75000
D:I ist/X	TNC Scope	10 kHz	75000
D:I soll/X	TNC Scope	10 kHz	75000
D:x bottom	Acc	10 kHz	75000

D:y bottom	Acc	10 kHz	75000
D:z bottom	Acc	10 kHz	75000
D:x nut	Acc	10 kHz	75000
D:y nut	Acc	10 kHz	75000
D:z nut	Acc	10 kHz	75000
D:x top	Acc	10 kHz	75000
D:y top	Acc	10 kHz	75000
D:z top	Acc	10 kHz	75000
S:x bottom	Acc	10 kHz	153601
S:y bottom	Acc	10 kHz	153601
S:z bottom	Acc	10 kHz	153601
S:x nut	Acc	10 kHz	153601
S:y nut	Acc	10 kHz	153601
S:z nut	Acc	10 kHz	153601
S:x top	Acc	10 kHz	153601
S:y top	Acc	10 kHz	153601
S:z top	Acc	10 kHz	153601
S:Nominal rotational speed	TNC opt	1 kHz	16384
S:Actual rotational speed	TNC opt	1 kHz	16384
S:Actual position of the position encoder(dy/dt)	TNC opt	1 kHz	16384
S:Actual position of the motor encoder(dy/dt)	TNC opt	1 kHz	16384

Table A.1.: Description of recorded signals

List of Figures

List of Tables

Bibliography

[1] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao. "Deep learning and its applications to machine health monitoring". In: *Mechanical Systems and Signal Processing* 115 (2019), pp. 213–237. DOI: https://doi.org/10.1016/j.ymssp.2018.05.050.

[2] B. Denkena, B. Bergmann, and A. Schmidt. "Preload monitoring of single nut ball screws based on sensor fusion". In: *CIRP Journal of Manufacturing Science and Technology* 33 (2021), pp. 63–70. DOI: https://doi.org/10.1016/j.cirpj.2021.02.006.

[3] P. Li, X. Jia, J. Feng, H. Davari, G. Qiao, Y. Hwang, and J. Lee. "Prognosability study of ball screw degradation using systematic methodology". In: *Mechanical Systems and Signal Processing* 109 (2018), pp. 45–57. DOI: https://doi.org/10.1016/j.ymssp.2018.02.046.

[4] V. Pandhare, X. Li, M. Miller, X. Jia, and J. Lee. "Intelligent Diagnostics for Ball Screw Fault Through Indirect Sensing Using Deep Domain Adaptation". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–11. DOI: 10.1109/TIM.2020.3043512.

[5] M. Azamfar, X. Li, and J. Lee. "Intelligent ball screw fault diagnosis using a deep domain adaptation methodology". In: *Mechanism and Machine Theory* 151 (2020), p. 103932. DOI: https://doi.org/10.1016/j.mechmachtheory.2020.103932.

[6] M. Benker, R. Kleinwort, and M. F. Zäh. "Estimating remaining useful life of machine tool ball screws via probabilistic classification". In: *IEEE International Conference on Prognostics and Health Management (ICPHM)*. San Francisco, USA, 2019, pp. 1–7. DOI: 10.1109/ICPHM.2019.8819445.

[7] W. Lee, J. W. Lee, M. Hong, S.-H. Nam, Y. Jeon, and M. Lee. "Failure Diagnosis System for a Ball-Screw by Using Vibration Signals". In: *Shock and Vibration* 2015 (2015), pp. 1–9. DOI: 10.1155/2015/435870.

[8] Y. Deng, D. Shichang, J. Shiyao, Z. Chen, and X. Zhiyuan. "Prognostic study of ball screws by ensemble data-driven particle filters". In: *Journal of Manufacturing Systems* 56 (2020), pp. 359–372. DOI: https://doi.org/10.1016/j.jmsy.2020.06.009.

[9] S. Sagiroglu and D. Sinanc. "Big data: A review". In: *International Conference on Collaboration Technologies and Systems (CTS)*. San Diego, USA, 2013, pp. 42–47. DOI: 10.1109/CTS.2013.6567202.

[10] O. Calin. *Deep Learning Architectures: A Mathematical Approach*. 1st. Springer Publishing Company, Incorporated, 2020. ISBN: 3030367207.

[11] L. Shiloh-Perl and R. Giryes. "Introduction to deep learning". In: *ArXiv* (2020). DOI: 10.48550/ARXIV.2003.03253.

[12] K. Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. DOI: https://doi.org/10.1016/0893-6080(91)90009-T.

[13] B. Ding, H. Qian, and J. Zhou. "Activation functions and their characteristics in deep neural networks". In: *Chinese Control And Decision Conference (CCDC)*. Shenyang, China, 2018, pp. 1836–1841. DOI: 10.1109/CCDC.2018.8407425.

[14] A. K. Dubey and V. Jain. "Comparative Study of Convolution Neural Network's Relu and Leaky-Relu Activation Functions". In: *Applications of Computing, Automation and Wireless Systems in Electrical Engineering* 553 (2019), pp. 873–880. DOI: 10.1007/978-981-13-6772-4_76.

[15] S. Ruder. "An overview of gradient descent optimization algorithms". In: *ArXiv* (2016). DOI: 10.48550/ARXIV.1609.04747.

[16] A. Zenghui, L. Shunming, W. Jinrui, X. Yua, and X. Kun. "Generalization of deep neural network for bearing fault diagnosis under different working conditions using multiple kernel method". In: *Neurocomputing* 352 (2019), pp. 42–53. DOI: https://doi.org/10.1016/j.neucom.2019.04.010.

[17] A. Lydia and S. Francis. "Adagrad - An Optimizer for Stochastic Gradient Descent". In: *International Journal of Information and Computing Science* 6.5 (2019), pp. 566–568. DOI: http://ijics.com/gallery/92-may-1260.pdf.

[18] K. O'Shea and R. Nash. "An Introduction to Convolutional Neural Networks". In: *ArXiv* (2015). DOI: https://doi.org/10.48550/arXiv.1511.08458.

[19] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. "Deformable Convolutional Networks". In: *IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy, 2017, pp. 764–773. DOI: 10.1109/ICCV.2017.89.

[20] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. "Visual Domain Adaptation: A survey of recent advances". In: *IEEE Signal Processing Magazine* 32.3 (2015), pp. 53–69. DOI: 10.1109/MSP.2014.2347059.

[21] Q. Li, B. Tang, L. Deng, Y. Wu, and Y. Wang. "Deep balanced domain adaptation neural networks for fault diagnosis of planetary gearboxes with limited labeled data". In: *Measurement* 156 (2020), p. 107570. DOI: https://doi.org/10.1016/j.measurement.2020.107570.

[22] S. Li, C. Liu, Q. Lin, B. Xie, Z. Ding, G. Huang, and J. Tang. "Domain conditioned adaptation network". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. New York, USA, 2020, pp. 11386–11393. DOI: 10.1609/aaai.v34i07.6801.

[23] Y. Li, K. Swersky, and R. Zemel. "Generative Moment Matching Networks". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Vol. 37. Lille, France, 2015, pp. 1718–1727. DOI: 10.48550/ARXIV.1502.02761.

[24] Z. Feng, M. Liang, and F. Chu. "Recent advances in time–frequency analysis methods for machinery fault diagnosis: A review with application examples". In: *Mechanical Systems and Signal Processing* 38.1 (2013), pp. 165–205. DOI: https://doi.org/10.1016/j.ymssp.2013.01.017.

[25] F. Hlawatsch and G. Boudreaux-Bartels. "Linear and quadratic time-frequency signal representations". In: *Signal Processing Magahao ine, IEEE* 9 (1992), pp. 21–67. DOI: 10.1109/79.127284.

[26] M. Sifuzzaman, M. Islam, and M. Ali. "Application of Wavelet Transform and its Advantages Compared to Fourier Transform". In: *Journal of Physical Sciences* 13.1 (2009), pp. 121–134. DOI: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.457.8548&rep=rep1&type=pdf.

[27] M. J. Whelan, D. Verstraete, A. Ferrada, E. L. Droguett, V. Meruane, and M. Modarres. "Deep Learning Enabled Fault Diagnosis Using Time-Frequency Image Analysis of Rolling Element Bearings". In: *Shock and Vibration* 2017 (2017), p. 5067651. DOI: 10.1155/2017/5067651.

[28] T. A. Harris and J. I. McCool. "On the Accuracy of Rolling Bearing Fatigue Life Prediction". In: *Journal of Tribology* 118.2 (1996), pp. 297–308. DOI: 10.1115/1.2831299.

[29] T. L. Nguyen, S.-K. Ro, and J.-K. Park. "Study of ball screw system preload monitoring during operation based on the motor current and screw-nut vibration". In: *Mechanical Systems and Signal Processing* 131 (2019), pp. 18–32. DOI: https://doi.org/10.1016/j.ymssp.2019.05.036.

[30] M. Ringnér. "What is principal component analysis?" In: *Nature Biotechnology* 26.3 (2008), pp. 303–304. DOI: 10.1038/nbt0308-303.

[31] R. Wald, T. Khoshgoftaar, and A. Napolitano. "Comparison of Stability for Different Families of Filter-Based and Wrapper-Based Feature Selection". In: *12th International Conference on Machine Learning and Applications*. Vol. 2. Washington DC, United States, 2013, pp. 457–464. DOI: 10.1109/ICMLA.2013.162.

[32] X. Li, W. Zhang, and Q. Ding. "A robust intelligent fault diagnosis method for rolling element bearings based on deep distance metric learning". In: *Neurocomputing* 310 (2018), pp. 77–95. DOI: https://doi.org/10.1016/j.neucom.2018.05.021.

[33] L. Guo, Y. Lei, S. Xing, T. Yan, and N. Li. "Deep Convolutional Transfer Learning Network: A New Method for Intelligent Fault Diagnosis of Machines With Unlabeled Data". In: *IEEE Transactions on Industrial Electronics* 66.9 (2019), pp. 7316–7325. DOI: 10.1109/TIE.2018.2877090.

[34] J. Singh, M. Azamfar, A. Ainapure, and J. Lee. "Deep learning-based cross-domain adaptation for gearbox fault diagnosis under variable speed conditions". In: *Measurement Science and Technology* 31.5 (2020), p. 055601. DOI: 10.1088/1361-6501/ab64aa.

[35] S. Kang, W. Chen, Y. Wang, X. Na, Q. Wang, and V. I. Mikulovich. "Method of state identification of rolling bearings based on deep domain adaptation under varying loads". In: *IET Science, Measurement & Technology* 14.3 (2020), pp. 303–313. DOI: https://doi.org/10.1049/iet-smt.2019.0043.

[36] R. Aljundi and T. Tuytelaars. "Lightweight Unsupervised Domain Adaptation by Convolutional Filter Reconstruction". In: *Computer Vision – ECCV Workshops*. Vol. 9915. Amsterdam, Netherlands, 2016, pp. 508–515. DOI: https://doi.org/10.1007/978-3-319-49409-8_43.

[37] O. F. Gabriel Michau Thomas Palmé. "Deep Feature Learning Network for Fault Detection and Isolation". In: *Proceedings of the Annual Conference of the PHM Society*. Vol. 9. 1. St. Petersburg, USA, 2017, p. 2446. DOI: https://doi.org/10.36001/phmconf.2017..

[38] L. Zhang, H. Gao, J. Wen, S. Li, and Q. Liu. "A deep learning-based recognition method for degradation monitoring of ball screw with multi-sensor data fusion". In: *Microelectronics Reliability* (2017). DOI: 10.1016/j.microrel.2017.03.038.

[39] M. Zhang, D. Wang, W. Lu, J. Yang, Z. Li, and B. Liang. "A Deep Transfer Model With Wasserstein Distance Guided Multi-Adversarial Networks for Bearing Fault Diagnosis Under Different Working Conditions". In: *IEEE Access* 7 (2019), pp. 65303–65318. DOI: 10.1109/ACCESS.2019.2916935.