DIPLOMARBEIT

Bayesian Estimation Using Deep Learning-Based Feature Extraction and Pseudo-Labels

ausgeführt zur Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Franz Hlawatsch
Dipl.-Ing. Thomas John Bucco

am
Institute of Telecommunications

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Michael Girsch

Wien, September 2023

Abstract

The fields of deep learning and Bayesian estimation offer signal processing methods that exhibit unique and contrasting capabilities. The main advantage of Bayesian estimation methods is that they provide a detailed description of the signal processing results in the form of the posterior distribution, which is considerably more informative than the point estimates provided by conventional deep learning methods. On the other hand, deep learning enjoys huge popularity across various disciplines for its ability to solve complex tasks such as object detection and localization.

In this thesis, we present several contributions related to the combination of Bayesian estimation and deep learning-based feature extraction. First, we propose a methodology for solving an intractable Bayesian estimation problem for which an accurate statistical model is unavailable. Second, we introduce three approaches for feeding back transformed estimates produced by a Bayesian estimator as corrected features, called "pseudo-labels," which allow us to retrain a deep neural network in a semi-supervised way. Two of these approaches incorporate statistical information from the Bayesian estimator in the training of the deep neural network.

As a practical application of the proposed framework and pseudo-labeling techniques, we consider a target tracking problem. We develop a tracking system consisting of a deep neural network and a sequential Bayesian estimator, in particular a Kalman filter. Our numerical results demonstrate that the proposed pseudo-labeling techniques can lead to an improved deep neural network performance compared with a deep neural network trained by conventional semi-supervised learning.

Kurzfassung

Deep learning und Bayessche Schätzung sind wichtige Methodiken der Signalverarbeitung, die unterschiedliche Stärken aufweisen. Die von Bayesschen Schätzern berechneten A-posteriori-Verteilungen sind wesentlich informativer als die von Methoden des deep learning durchgeführten Punktschätzungen. Dessen ungeachtet hat sich deep learning als ein mächtiges Werkzeug für viele komplexe Problemstellungen erwiesen, beispielsweise zur Objektdetektion und -lokalisierung.

In dieser Arbeit präsentieren wir eine Methodik, die Bayessche Schätzung und deep learning-basierte Methoden zur Merkmalsextraktion verbindet. Zunächst schlagen wir ein Verfahren zur Lösung eines Bayesschen Schätzproblems vor, für welches eine statistische Beschreibung nicht verfügbar ist. Weiters beschreiben wir drei Methoden für die Rückkopplung transformierter Schätzergebnisse eines Bayesschen Schätzers als korrigierte Merkmale. Diese korrigierten Merkmale, welche auch als „Pseudo-Labels“ bezeichnet werden, erlauben ein verbessertes Training eines tiefen neuronalen Netzes auf der Grundlage von halbüberwachtem Lernen. Zwei der vorgestellten Methoden ermöglichen die Einbindung der vom Bayesschen Schätzer gelieferten statistischen Information in das Training des tiefen neuronalen Netzes.

Als praktische Anwendung der vorgestellten Methodik betrachten wir ein Zielverfolgungsproblem, für das wir ein aus einem tiefen neuronalen Netz und einem sequentiellen Bayesschen Schätzer (Kalman-Filter) bestehendes System entwickeln. Unsere Simulationsergebnisse zeigen, dass tiefe neuronale Netze mit der vorgeschlagenen „Pseudo-Label“-Methodik eine bessere Leistungsfähigkeit aufweisen als tiefe neuronale Netze, die konventionelles Training auf der Grundlage von halbüberwachtem Lernen verwenden.

Acknowledgments

I would like to express my gratitude to my supervisors Ao. Univ. Prof. Franz Hlawatsch and Dipl.-Ing. Thomas John Bucco for their continuous support, expertise, and guidance. Furthermore, I would like to thank all colleagues at the Institute of Telecommunications for many helpful discussions.

I am deeply thankful to my family for their love and encouragement during my studies. Last but not least, I also want to thank the friends I made during my time at TU Wien.

Contents

List of Figures

List of Abbreviations

DAG	Directed Acyclic Graph
DNN	Deep Neural Network
GD	Gradient Descent
HMM	Hidden Markov Model
i.i.d.	independent and identically distributed
KF	Kalman Filter
MMSE	Minimum Mean Squared Error
NN	Neural Network
pdf	probability density function
ReLU	rectified linear unit
SGD	Stochastic Gradient Descent

1. Introduction

1.1. Motivation

When using stochastic parameter estimation methods, we often face the problem that it is difficult to describe the statistical relationship between the raw measurements and the parameters of interest, rendering such estimation methods inaccurate or useless. For instance, in image processing, formulating a statistical relationship between images and the parameters of interest can be challenging or impossible. Therefore, we usually preprocess the raw measurements to obtain "simplified measurements", for which the statistical relationship with the parameters of interest is known. In this work, we propose to use a deep neural network (DNN) as a feature extraction stage that converts raw measurements into simplified measurements. Subsequently, a Bayesian estimator is used to estimate the parameters of interest.

Another issue addressed in this work is a lack of labeled data for the training of DNNs. Semi-supervised learning is a powerful approach that enables the use of unlabeled data to augment a small set of labeled data in the training of DNNs. One particular semi-supervised learning approach is pseudo-labeling, which creates labels for the unlabeled data using a DNN that is initially trained using only the labeled data. In the next step, the DNN can be retrained using the unlabeled training data paired with the pseudo-labels in addition to the labeled training data. However, using conventional DNN architectures, we cannot determine the quality of the pseudo-labels automatically. It may happen that the DNN trained with a small amount of labeled training data performs poorly and thus produces pseudo-labels of low accuracy. Therefore, it would be desirable to develop methods that, first, can make pseudo-labels more accurate and, second, weigh them in the subsequent training according to their forecasted accuracy. Bayesian estimation methods provide a quantitative characterization of uncertainty. Therefore, we propose to use a Bayesian estimator that processes the DNN outputs and produces the pseudo-labels as well as a characterization of uncertainty, which are used to retrain the DNN.

As a use case for this proposed framework combining a DNN and a Bayesian estimator, we consider a target tracking problem. Concretely, a DNN is used to produce preliminary estimates of the position of a moving target from a sequence of images (thus, the feature extraction task here amounts to object localization). A sequential Bayesian estimator then calculates estimates of the target position and velocity from preliminary position estimates provided by the DNN. The estimates calculated by the sequential estimator as well as the statistical information are processed to retrain the DNN.

1.2. State of the Art

One approach to obtaining a quantitative characterization of uncertainty of the DNN's output is constituted by Bayesian neural networks [4]. Rather than producing a point estimate, Bayesian neural networks produce a posterior distribution at their output. However, it remains difficult to design efficient training algorithms for this type of DNNs.

An extensive overview of the semi-supervised learning framework can be found in [5, 6]. The authors of [7] use Bayesian methods and semi-supervised learning for handwritten character recognition. In [8], hybrid methods using both DNNs and conventional inference methods are described. The framework proposed in this thesis belongs to the "DNN-aided inference" category.

Many papers addressing the problem of object localization/detection and classification using DNNs focus on the DNN architecture [9–11]. The combination of DNNs and Bayesian estimation is a promising approach to target tracking. In [12], the combination of a Kalman filter (KF) and a DNN is used for a 3D multi-object tracking problem. However, the DNN is trained using conventional supervised learning, and the statistical information provided by the KF is not used for training the DNN. The DNN that is used for object localization is described in [13]. The work in [14] combines a DNN and a particle-based sum-product algorithm for target tracking. For feature extraction supervised learning is used [15].

1.3. Contribution and Outline

The contributions of this thesis are as follows. We propose a combination of DNNs and Bayesian estimators for estimation problems where an accurate statistical model relating the measurements and the parameters of interest does not exist. We present a method for creating pseudo-labels, i.e., corrected feature vectors, using the results of a Bayesian estimator. Moreover, we propose a method for training a DNN in a semi-supervised way using statistical information obtained from the Bayesian estimator. These methods are then applied to the problem of tracking a single target.

The thesis is structured as follows:

- In Chapter 2, we describe the fundamentals of deep learning and certain concepts relevant to the subsequent chapters.
- Chapter 3 presents the proposed combination of Bayesian estimation and DNN-based feature extraction. In particular, three methods for creating pseudo-labels and retraining the DNN in a semi-supervised way are proposed. Two of these methods leverage statistical information provided by the Bayesian estimator for improved DNN training.
- In Chapter 4, we consider sequential estimation and review the general Bayesian filter as well as the KF. Then, we extend the methods from Chapter 3 to the sequential case.
- In Chapter 5, a combination of DNN feature extraction and sequential Bayesian estimation using a KF is applied to the problem of tracking a single target. The proposed method is based on a modified KF.
- Chapter 6 presents and discusses simulation results for the target tracking problem considered in Chapter 5. In particular, the DNN performance is evaluated for different pseudo-labeling methods, and it is shown that the estimates obtained with the combined DNN-KF method are more robust than the estimates produced by the DNN alone.
- Chapter 7 summarizes the main results and suggests future research directions.

2. Deep Learning

Deep learning has emerged as a powerful methodology in the area of machine learning. Deep learning methods are capable of handling different tasks related to applications in engineering and natural sciences, e.g., image classification [16], solving partial differential equations [17], and protein structure prediction [18]. In this chapter, some important concepts in the field of deep learning will be summarized. We introduce a neural network (NN) as a parameterized function, distinguish between NNs and DNNs, and describe the relation between DNNs and function learning. Furthermore, we describe how to optimize the performance of DNNs, i.e., training. References for this chapter are [19–22].

2.1. Deep Neural Networks

In the most general form, an NN can be thought of as a parameterized function that approximates another function based on a limited number of observations (data set). The name originates from biology, where in the brain signals are transmitted and processed via neurons. NNs allow for a representation that consists of connected nodes (neurons) that can process information. An NN is specified by its architecture, i.e., the number of nodes, the activation function (a nonlinear transformation), and the training routine used to optimize the parameters for a given data set such that a good approximation of the target function is obtained.

Let us consider a function

$$\boldsymbol{y} = f^*(\boldsymbol{x}), \tag{2.1}$$

with input space $\mathcal{X}$ and output space $\mathcal{Y}$ and their respective elements, i.e., $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$. In this text, we restrict ourselves to real-valued inputs and outputs, i.e., $\mathcal{X} \subseteq \mathbb{R}^m$ and $\mathcal{Y} \subseteq \mathbb{R}^n$. The NN is a parameterized mapping of the following form,

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\phi}) \text{ with parameters } \boldsymbol{\phi} \in \mathcal{P}. \tag{2.2}$$

Given an NN with fixed architecture, we solve an optimization problem to find the optimal

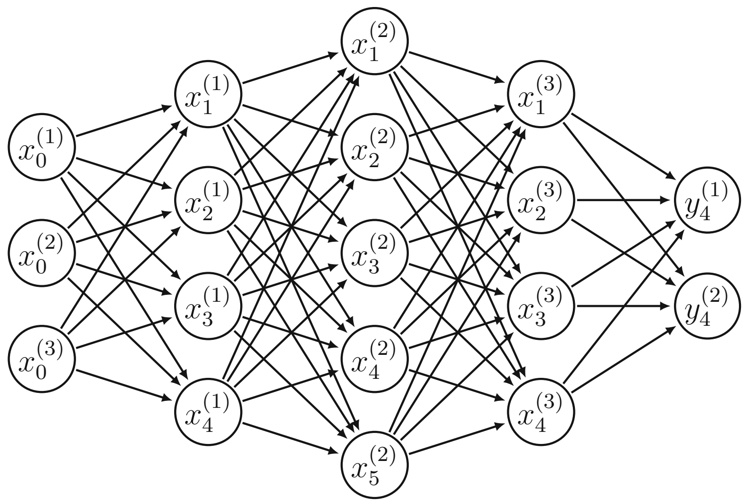


Figure 2.1.: DNN with $L = 5$ layers, input dimension $m = 3$ and output dimension $n = 2$. We use a modification of the LaTeX code provided in [1].

parameters ϕ^* for which $f(\boldsymbol{x};\boldsymbol{\phi}^*)$ best approximates the function f^* which is assumed to be unknown. The parameter $\boldsymbol{\phi}$ is a sequence of matrix-vector tuples [22, Chapter 1], i.e.,

$$\boldsymbol{\phi} = \big(\boldsymbol{\phi}_l\big)_{l=1}^{L} = \big(\boldsymbol{W}_l, \boldsymbol{b}_l\big)_{l=1}^{L} \in \mathcal{P}\,, \text{ where } \mathcal{P} = \bigtimes_{l=1}^{L} \big(\mathbb{R}^{N_l \times N_{l-1}} \times \mathbb{R}^{N_l}\big), \tag{2.3}$$

with $N_0 = m, N_L = n,\ N_1, \dots, N_{L-1} \in \mathbb{N},\ \boldsymbol{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}$, and $\boldsymbol{b}_l \in \mathbb{R}^{N_l}$ for $l = 1, \dots, L$. The matrices $\boldsymbol{W}_l$ are called weight matrices and the vectors $\boldsymbol{b}_l$ are called bias vectors. The l-th tuple $(\boldsymbol{W}_l, \boldsymbol{b}_l)$ describes one layer of the NN. With an input vector $\boldsymbol{x} = \boldsymbol{x}_0$ the i-th entry of the l-th layer output $\boldsymbol{x}_l$ is calculated as [19, Chapter 2]

$$\boldsymbol{x}_l^{(i)} = \rho\Big(\big(\boldsymbol{W}_l \boldsymbol{x}_{l-1}\big)^{(i)} + \boldsymbol{b}_l^{(i)}\Big), \quad i = 1, \dots, N_L; \quad l = 1, \dots, L-1, \tag{2.4}$$

where $\rho : \mathbb{R} \to \mathbb{R}$ is a generally nonlinear activation function, whereas the output vector $\boldsymbol{y}$ is calculated as

$$\boldsymbol{y} = \boldsymbol{W}_L \boldsymbol{x}_{L-1} + \boldsymbol{b}_L. \tag{2.5}$$

An NN with more than one hidden layer, i.e., a layer between input and output layer, is called a DNN. An NN can be visualized by a directed acyclic graph (DAG), as shown in Figure 2.1.

2.2. Supervised and Semi-Supervised Learning

In this section, we introduce supervised and semi-supervised learning for DNNs and present some basic notions that will be used later on.

Supervised Learning

Generally speaking, in a supervised learning [23] task we want to find an approximation f of an unknown function $f^* : \mathcal{X} \to \mathcal{Y}$, which is a mapping from the domain set $\mathcal{X}$ to the target or label set $\mathcal{Y}$. The approximation is based on a limited set of observations, also referred to as the training data set $\mathcal{S}_{\mathrm{L}}$, i.e., which consists of data pairs $\big(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})}\big) \in \mathcal{X} \times \mathcal{Y}$, where we write $\boldsymbol{x}_m^{(\text{labeled})}$ and $\boldsymbol{y}_m^{(\text{labeled})}$ to indicate that the data pair is created by the unknown function f^*, i.e.,

$$\boldsymbol{y}_m^{(\text{labeled})} = f^*\left(\boldsymbol{x}_m^{(\text{labeled})}\right). \tag{2.6}$$

The labeled data set is

$$\mathcal{S}_{\mathrm{L}} = \left\{\left(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})}\right)\right\}_{m=1}^{M}. \tag{2.7}$$

The performance of the approximation or predictor f is measured using a loss function $\mathcal{L}(f(\boldsymbol{x}), \boldsymbol{y})$. A popular example for $\mathcal{L}(f(\boldsymbol{x}), \boldsymbol{y})$ is the squared Euclidean norm. The loss for the labeled data pair $\big(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})}\big)$ is given by

$$\mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})}), \boldsymbol{y}_m^{(\text{labeled})}\big) = \|f(\boldsymbol{x}_m^{(\text{labeled})}) - \boldsymbol{y}_m^{(\text{labeled})}\|^2. \tag{2.8}$$

In deep learning we try to approximate the function f^* by a DNN $f(\,\cdot\,; \boldsymbol{\phi})$. This is achieved by minimizing the empirical risk. For a general training data set $\mathcal{S} = \big\{\big(\boldsymbol{x}_m, \boldsymbol{y}_m\big)\big\}_{m=1}^{M}$ and a predictor $f(\,\cdot\,; \boldsymbol{\phi})$, the empirical risk is defined as [22, Chapter 1]

$$\mathcal{R}(\boldsymbol{\phi}) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}\big(f(\boldsymbol{x}_m; \boldsymbol{\phi}), \boldsymbol{y}_m\big). \tag{2.9}$$

In supervised learning, we consider the empirical risk over the set $\mathcal{S}_{\mathrm{L}}$, i.e.,

$$\mathcal{R}(\boldsymbol{\phi}) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})}; \boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\big). \tag{2.10}$$

The optimal predictor $f(\,\cdot\,; \boldsymbol{\phi})$ is found by minimizing the empirical risk with respect to the parameter $\boldsymbol{\phi}$, i.e.,

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi} \in \mathcal{P}}{\operatorname{argmin}}\, \mathcal{R}(\boldsymbol{\phi}). \tag{2.11}$$

The optimal DNN exhibits good generalization ability if

$$f(\boldsymbol{x};\boldsymbol{\phi}^*) \approx \boldsymbol{y}, \quad \forall\ (\boldsymbol{x},\boldsymbol{y}) \notin \mathcal{S}_{\mathrm{L}} \quad \text{s.t.} \quad \boldsymbol{y} = f^*(\boldsymbol{x}). \tag{2.12}$$

Semi-supervised Learning

Constructing sufficiently large labeled data sets can be costly, especially when the labeling process requires trained personnel. Semi-supervised learning [6] tackles this problem by using both labeled and unlabeled data. To use semi-supervised learning for DNNs, we need the concept of *pseudo-labels*. Pseudo-labels were first introduced in [24] as an entropy regularization for classification tasks, where the pseudo-label for a given unlabeled input is simply the predicted class label, assuming a certain threshold for the prediction probability is exceeded. In this work, we consider a regression task with a labeled data set $\mathcal{S}_{\mathrm{L}} = \big\{\big(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})}\big)\big\}_{m=1}^{M}$ and an unlabeled set $\mathcal{S}_{\mathrm{UL}} = \{\boldsymbol{x}_n\}_{n=1}^{N}$, where $N \gg M$. We train the DNN using the labeled data set $\mathcal{S}_{\mathrm{L}}$ first, in order to obtain a provisional optimal parameter $\boldsymbol{\phi}^*$. Next, the outputs of the DNN for the data vectors $\boldsymbol{x}_n$ from the unlabeled data set $\mathcal{S}_{\mathrm{UL}}$ are computed, i.e.,

$$\boldsymbol{y}_n = f(\boldsymbol{x}_n;\boldsymbol{\phi}^*), \quad n = 1,\dots,N. \tag{2.13}$$

These output vectors $\boldsymbol{y}_n$ are also referred to as pseudo-labels and enable us to create the data set $\mathcal{S}_{\mathrm{PL}} = \Big\{\Big(\boldsymbol{x}_n, \boldsymbol{y}_n\Big)\Big\}_{n=1}^{N}$. Now we can augment the labeled data set by $\mathcal{S}_{\mathrm{PL}}$ to obtain a new training data set

$$\mathcal{S}_{\text{new}} = \mathcal{S}_{\mathrm{L}} \cup \mathcal{S}_{\mathrm{PL}}. \tag{2.14}$$

The DNN is then retrained using the new data set $\mathcal{S}_{\text{new}}$. The augmented empirical risk is given by

$$\mathcal{R}_{\text{new}}(\boldsymbol{\phi}) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})};\boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\big) + \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \boldsymbol{y}_n\big). \tag{2.15}$$

In order to further improve the performance, the process of augmenting the labeled data set and retraining the DNN using the augmented data set can be repeated.

2.3. Training

Training the DNN amounts to calculating the optimal parameter $\boldsymbol{\phi}$. Recalling Equation (2.3) and (2.11), we compute the entries of the weight matrices $\boldsymbol{W}_l$ and bias vectors $\boldsymbol{b}_l$ for all

$l = 1, \dots, L$ that minimize the empirical risk $\mathcal{R}(\boldsymbol{\phi})$. However, for a DNN the empirical risk is a nonlinear function of the parameter $\boldsymbol{\phi}$, so that the optimization problem in Equation (2.11) cannot be solved analytically. Thus, we resort to numerical methods. A further difficulty is that the empirical risk can have multiple global and local minima.

In practice, one numerical technique called gradient descent (GD) [25] turned out to be very powerful. Any (local or global) minimum of the empirical risk satisfies the equation

$$\nabla \mathcal{R}(\boldsymbol{\phi}) = \mathbf{0}, \tag{2.16}$$

where ∇ is the gradient with respect to $\boldsymbol{\phi}$, i.e., consisting of the partial derivatives of the entries of all $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$, and $\mathbf{0}$ is the zero vector. Such a minimum can be found in an iterative manner by taking steps in the direction of the largest decrease of the empirical risk, i.e., $-\nabla \mathcal{R}(\boldsymbol{\phi})$. At the i-th iteration, the update equation for the new parameters $\boldsymbol{\phi}^{(i+1)}$ is given by

$$\boldsymbol{\phi}^{(i+1)} = \boldsymbol{\phi}^{(i)} - \eta \nabla \mathcal{R}(\boldsymbol{\phi}^{(i)}), \tag{2.17}$$

where $\eta > 0$ denotes the learning rate. The value of the learning rate η determines the step size in the update equation. The gradient $\nabla \mathcal{R}(\boldsymbol{\phi})$ can be calculated by the backpropagation algorithm, see [26]. The algorithm starts with an initial value $\boldsymbol{\phi}^{(0)}$ and stops after finding a minimum, i.e., when $|\nabla \mathcal{R}(\boldsymbol{\phi}^{(i)})| < \varepsilon$ for some small $\varepsilon > 0$ or when an upper bound on the number of iterations $T_{\max}$ is reached. An important modification of gradient descent is called stochastic gradient descent (SGD) [20]. For large data sets, i.e., a high value of M, calculating (2.11) is computationally challenging. Therefore, the idea of SGD is to estimate $\mathcal{R}(\boldsymbol{\phi})$ by considering a small subset $\mathcal{S}'_{\mathrm{L}} \subset \mathcal{S}_{\mathrm{L}}$ of size $M' \ll M$ that is randomly chosen, i.e.,

$$\mathcal{R}(\boldsymbol{\phi}) \approx \frac{1}{M'} \sum_{m \in \mathcal{S}'_{\mathrm{L}}} \mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})}; \boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\big). \tag{2.18}$$

In the semi-supervised learning framework, we additionally have to use a subset $\mathcal{S}'_{\mathrm{PL}} \subset \mathcal{S}_{\mathrm{PL}}$, with $\boldsymbol{y}_n$ calculated by (2.13) and $|\mathcal{S}'_{\mathrm{PL}}| = N' \ll N$. The data sets of reduced size $\mathcal{S}'_{\mathrm{L}}$ and $\mathcal{S}'_{\mathrm{PL}}$ allow us to estimate $\mathcal{R}_{\text{new}}(\boldsymbol{\phi})$ by

$$\mathcal{R}_{\text{new}}(\boldsymbol{\phi}) \approx \frac{1}{M'} \sum_{m \in \mathcal{S}'_{\mathrm{L}}} \mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})}; \boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\big) + \frac{1}{N'} \sum_{n \in \mathcal{S}'_{\mathrm{PL}}} \mathcal{L}\big(f(\boldsymbol{x}_n; \boldsymbol{\phi}), \boldsymbol{y}_n\big). \tag{2.19}$$

3. Deep Neural Network Aided Bayesian Estimation

3.1. Motivation

We consider a Bayesian estimation problem where both the parameter vector $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p$ and the data vector $\boldsymbol{x} \in \mathcal{X}$ are modeled as random quantities. The goal is to find an estimate $\hat{\boldsymbol{\theta}}(\boldsymbol{x})$ of the random vector $\boldsymbol{\theta}$ given an observation $\boldsymbol{x}$. In conventional Bayesian estimation, we assume the following distributions to be known:

- the prior probability density function (pdf) $p(\boldsymbol{\theta})$,
- the likelihood function $p(\boldsymbol{x}|\boldsymbol{\theta})$, which captures the statistical relation between the random data $\boldsymbol{x}$ and the random parameter $\boldsymbol{\theta}$.

In many applications, it is difficult to work with the likelihood function because the dimension of the measurement space $\mathcal{X}$ is too large for directly computing the estimate $\hat{\boldsymbol{\theta}}(\boldsymbol{x})$ or it is impossible to characterize the statistical dependence between the raw data $\boldsymbol{x}$ and the parameter $\boldsymbol{\theta}$. Therefore, we consider a lower-dimensional ”feature space” $\mathcal{Y}$ (i.e., $\dim(\mathcal{Y}) < \dim(\mathcal{X})$) where the statistical dependence between $\boldsymbol{\theta}$ and $\boldsymbol{y} \in \mathcal{Y}$ is known, described by a simplified likelihood function $p(\boldsymbol{y}|\boldsymbol{\theta})$. A DNN can then be used to extract features $\boldsymbol{y}$ from the observation $\boldsymbol{x}$. The Bayesian estimate $\hat{\boldsymbol{\theta}}(\boldsymbol{y})$ can then be calculated by using the prior pdf $p(\boldsymbol{\theta})$ and the simplified likelihood function $p(\boldsymbol{y}|\boldsymbol{\theta})$.

Of course, one could ask why we would not estimate $\boldsymbol{\theta}$ directly from $\boldsymbol{x}$ using only a DNN, without a Bayesian estimation stage. The main advantage of Bayesian estimation is that the statistical model constituted by the likelihood function $p(\boldsymbol{y}|\boldsymbol{\theta})$ and the prior pdf $p(\boldsymbol{\theta})$ provides information about the parameter $\boldsymbol{\theta}$ that is not leveraged by a DNN.

Secondly, when estimating $\boldsymbol{\theta}$ directly from the observed data $\boldsymbol{x}$ using a DNN, typically a more complex DNN architecture and more labeled training data are needed. We consider scenarios where extracting simple features $\boldsymbol{y}$ with a DNN can be achieved using conventional architectures and a reasonable amount of training data, whereas training a DNN

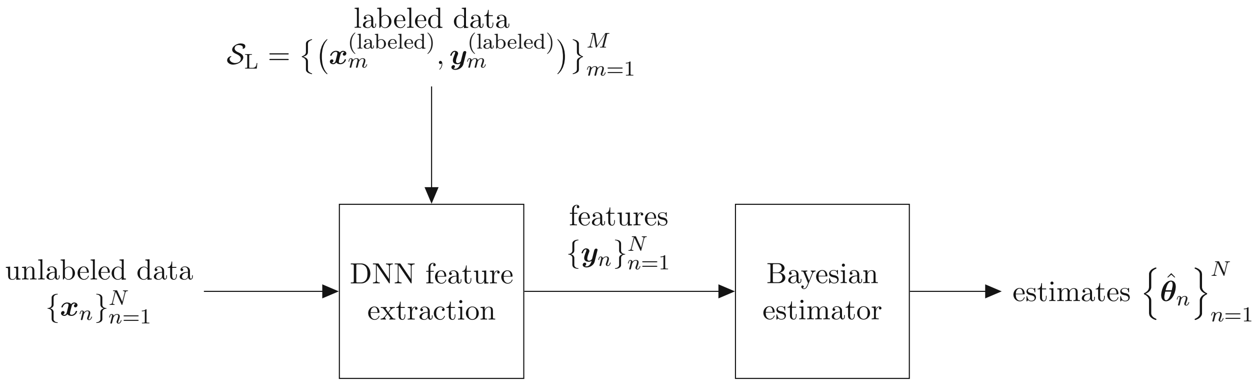

labeled data
$\mathcal{S}_{\mathrm{L}} = \{(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})})\}_{m=1}^{M}$
unlabeled data
$\{\boldsymbol{x}_n\}_{n=1}^{N}$
DNN feature extraction
features
$\{\boldsymbol{y}_n\}_{n=1}^{N}$
Bayesian estimator
estimates $\{\hat{\boldsymbol{\theta}}_n\}_{n=1}^{N}$

Figure 3.1.: Block diagram of the serial setup consisting of deep learning based feature extraction and a Bayesian estimator.

to directly infer $\boldsymbol{\theta}$ from $\boldsymbol{x}$ is not feasible. To summarize, in a first step, we extract features $\boldsymbol{y}$ from raw data $\boldsymbol{x}$ by using a DNN, and in a second step, we leverage knowledge of the statistical behavior of these features by performing Bayesian estimation of $\boldsymbol{\theta}$ from $\boldsymbol{y}$. References for this chapter are [27–29].

3.2. Serial Setup

Our goal is to calculate parameter estimates $\hat{\boldsymbol{\theta}}_n$ for a set of observations data $\boldsymbol{x}_n$, $n = 1, \dots, N$. Since this is too difficult, we use a DNN $f(\,\cdot\,;\boldsymbol{\phi})$ to extract a feature vector $\boldsymbol{y}_n$ from each $\boldsymbol{x}_n$. In a first step, we employ supervised learning to train the DNN with the labeled data set $\mathcal{S}_{\mathrm{L}} = \{(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})})\}_{m=1}^{M}$. Following Equation (2.10) and (2.11), we find $\boldsymbol{\phi}^*$ by minimizing the empirical risk over the set $\mathcal{S}_{\mathrm{L}}$, i.e.,

$$\boldsymbol{\phi}^* = \operatorname*{argmin}_{\boldsymbol{\phi}\in\mathcal{P}} \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})};\boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\big). \tag{3.1}$$

Subsequently, the trained DNN (with $\boldsymbol{\phi} = \boldsymbol{\phi}^*$) extracts features $\boldsymbol{y}_n$ from the unlabeled observations $\boldsymbol{x}_n$, i.e.,

$$\boldsymbol{y}_n = f(\boldsymbol{x}_n, \boldsymbol{\phi}^*), \quad n = 1, \dots, N. \tag{3.2}$$

After the features $\boldsymbol{y}_n$ are obtained by the DNN, the remaining task is to estimate the parameter $\boldsymbol{\theta}_n$ given the features $\boldsymbol{y}_n$. Knowing the simplified likelihood function $p(\boldsymbol{y}_n|\boldsymbol{\theta}_n)$ and the prior distribution $p(\boldsymbol{\theta}_n)$, the posterior pdf can be calculated by using Bayes' rule,

$$\begin{tikzcd} \boldsymbol{x}_n \arrow[r] & \boldsymbol{y}_n \arrow[r] & \boldsymbol{\theta}_n \end{tikzcd}$$

Figure 3.2.: Bayesian network describing the statistical relationship between observation $\boldsymbol{x}_n$, feature vector $\boldsymbol{y}_n$, and parameter $\boldsymbol{\theta}_n$ for the ideal scenario.

i.e.,

$$p(\boldsymbol{\theta}_n|\boldsymbol{y}_n) = \frac{p(\boldsymbol{y}_n|\boldsymbol{\theta}_n)p(\boldsymbol{\theta}_n)}{p(\boldsymbol{y}_n)}, \tag{3.3}$$

where we call $p(\boldsymbol{y}_n)$ the evidence, which can be calculated by marginalization

$$p(\boldsymbol{y}_n) = \int_\Theta p(\boldsymbol{y}_n|\boldsymbol{\theta}_n)p(\boldsymbol{\theta}_n)d\boldsymbol{\theta}_n = \int_\Theta p(\boldsymbol{y}_n, \boldsymbol{\theta}_n)d\boldsymbol{\theta}_n. \tag{3.4}$$

The posterior $p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)$ allows to calculate any Bayesian estimator. For example, the minimum mean squared error (MMSE) estimate $\hat{\boldsymbol{\theta}}_n$ for the DNN output $\boldsymbol{y}_n = f(\boldsymbol{x}_n; \boldsymbol{\phi}^*)$ is given by

$$\hat{\boldsymbol{\theta}}_n = \mathcal{E}\{\boldsymbol{\theta}_n|\boldsymbol{y}_n\} = \int_\Theta \boldsymbol{\theta}_n p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)d\boldsymbol{\theta}_n, \tag{3.5}$$

where the posterior pdf $p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)$ can be calculated according to Equation (3.3) and (3.4). An illustration of this setup is given in Figure 3.1.

In the ideal case, the feature $\boldsymbol{y}_n$ suffices to estimate $\boldsymbol{\theta}_n$ without the loss of any information. This can be formalized by stating that $\boldsymbol{\theta}_n$ is conditionally independent of $\boldsymbol{x}_n$ given $\boldsymbol{y}_n$, i.e., $\boldsymbol{\theta}_n \perp\!\!\!\perp \boldsymbol{x}_n|\boldsymbol{y}_n$. However, in the general case, we expect that information will be lost by the DNN feature extraction. A Bayesian network for the ideal case is given in Figure 3.2.

3.3. Feedback of Pseudo-Labels

In Section 2.2, we saw that unlabeled data can be incorporated into the training of a DNN. This was achieved by introducing pseudo-labels, which are the outputs of the DNN after the initial training phase, as stated by Equation (3.2). The initial optimal parameter $\boldsymbol{\phi}^*$ was obtained based on labeled training data according to Equation (3.1). Finally, the DNN is retrained with the augmented data set

$$\mathcal{S}_{\text{new}} = \left\{\left(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})}\right)\right\}_{m=1}^{M} \cup \left\{\left(\boldsymbol{x}_n, \boldsymbol{y}_n\right)\right\}_{n=1}^{N}. \tag{3.6}$$

The new optimal parameter $\boldsymbol{\phi}^*_{\text{new}}$ is obtained as

$$\boldsymbol{\phi}^*_{\text{new}} = \underset{\boldsymbol{\phi}\in\mathcal{P}}{\operatorname{argmin}} \left\{ \frac{1}{M}\sum_{m=1}^{M} \mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})};\boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\big) + \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \boldsymbol{y}_n\big) \right\}. \tag{3.7}$$

Now two important questions arise. On the one hand, it may happen that the DNN does not generalize well to unseen data due to a not sufficiently large labeled data set $\mathcal{S}_\text{L}$ or the optimization algorithm gets stuck in a local optimum with a high value of the empirical risk $\mathcal{R}(\boldsymbol{\phi}^*)$. Retraining the DNN with pseudo-labels, which exhibit low accuracy, is expected to lead to poor performance. Therefore, we would like to know if it is possible to generate more accurate feature vectors $\hat{\boldsymbol{y}}_n$ by transforming the parameter estimates $\hat{\boldsymbol{\theta}}_n$.

On the other hand, we want to investigate if we can use the statistical information of the Bayesian estimator in the training of the DNN. Potentially, this may be achieved by using an alternative loss function, where we somehow want to weigh data pairs $(\boldsymbol{x}_n, \hat{\boldsymbol{y}}_n)$ based on the statistical information of the corresponding parameter estimate $\hat{\boldsymbol{\theta}}_n$.

In the following sections, we explore three different ways of retraining the DNN. The first option, namely *hard pseudo-labeling*, only tries to generate pseudo-labels $\hat{\boldsymbol{y}}_n$ which provide a more accurate feature than the original DNN outputs $\boldsymbol{y}_n$. Second, we have *semi-soft pseudo-labeling*, where we want to utilize the statistical information of the posterior covariance matrix. This can be achieved by using the inverse of this matrix as a weighting factor in a modified loss function. In *soft pseudo-labeling*, we want to use the posterior pdf in the loss function by forming an expectation over $p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)$.

3.3.1. Hard Pseudo-Labels

The first possibility of retraining the DNN is using the "hard" point estimates $\hat{\boldsymbol{\theta}}_n$, hence the name hard pseudo-labels. The Bayesian estimates $\hat{\boldsymbol{\theta}}_n$ can be used to generate corrected feature vectors $\hat{\boldsymbol{y}}_n$ by introducing a mapping from the the parameter space Θ to the feature space $\mathcal{Y}$, i.e.,

$$\hat{\boldsymbol{y}}_n = \boldsymbol{y}(\hat{\boldsymbol{\theta}}_n). \tag{3.8}$$

We will refer to this mapping as the *feedback function*. Note that we use the same symbol for the feedback function $\boldsymbol{y}(\cdot)$ and the feature vector $\boldsymbol{y}$. The construction of the feedback function highly depends on the application, hence a general statement cannot be given. An example for $\boldsymbol{y}(\,\cdot\,)$, which we will consider in Chapter 5, is the transformation of the four-dimensional parameter estimate $\hat{\boldsymbol{\theta}}_n$ consisting of position and velocity components to a two-dimensional feature vector $\hat{\boldsymbol{y}}_n$ consisting only of position estimates. The set of hard

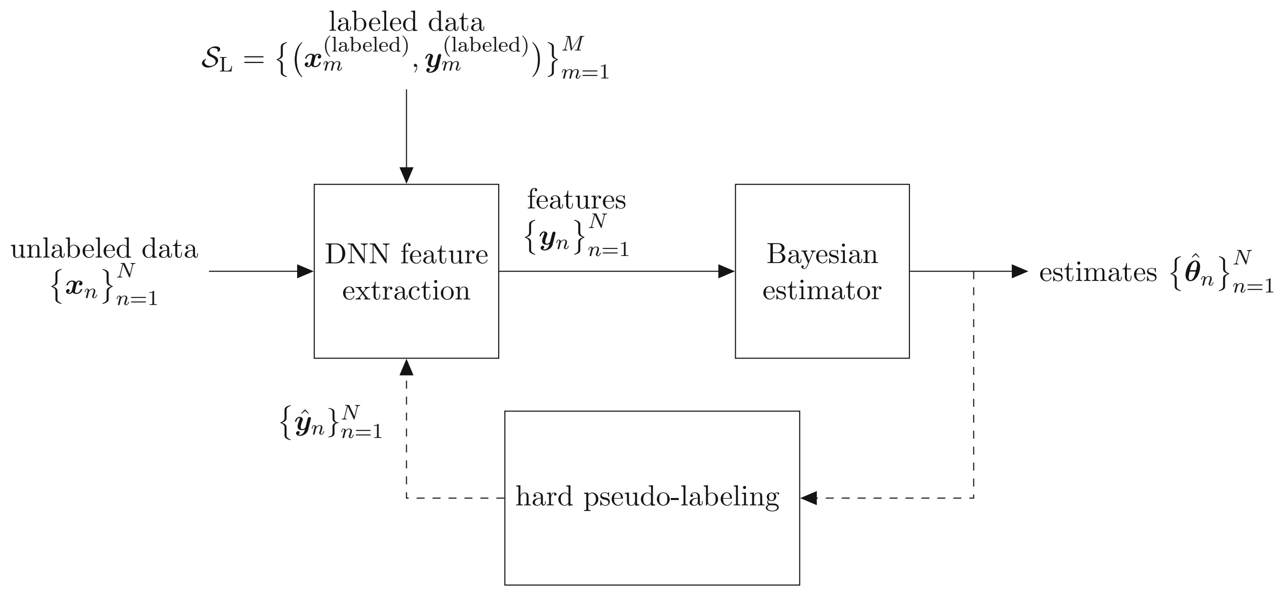

labeled data
$\mathcal{S}_{\text{L}} = \big\{\big(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})}\big)\big\}_{m=1}^{M}$
unlabeled data
$\{\boldsymbol{x}_n\}_{n=1}^{N}$
DNN feature extraction
features
$\{\boldsymbol{y}_n\}_{n=1}^{N}$
Bayesian estimator
estimates $\{\hat{\boldsymbol{\theta}}_n\}_{n=1}^{N}$
$\{\hat{\boldsymbol{y}}_n\}_{n=1}^{N}$
hard pseudo-labeling

Figure 3.3.: Block diagram of the semi-supervised estimation-learning method using feedback of hard pseudo-labels $\boldsymbol{y}_n$.

pseudo-labels is given by

$$\mathcal{S}_{\text{PL}} = \big\{\big(\boldsymbol{x}_n, \hat{\boldsymbol{y}}_n\big)\big\}_{n=1}^{N}. \tag{3.9}$$

The augmented data set is given as the union of the labeled data set $\mathcal{S}_{\text{L}}$ and the set of hard pseudo-labels $\mathcal{S}_{\text{PL}}$, i.e.,

$$\mathcal{S}_{\text{new}} = \big\{\big(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})}\big)\big\}_{m=1}^{M} \cup \big\{\big(\boldsymbol{x}_n, \hat{\boldsymbol{y}}_n\big)\big\}_{n=1}^{N}. \tag{3.10}$$

We can retrain the DNN, similar to Equation (3.7), to obtain the parameter vector

$$\boldsymbol{\phi}_{\text{new}}^{*} \triangleq \underset{\boldsymbol{\phi}\in\mathcal{P}}{\operatorname{argmin}} \bigg\{ \frac{\lambda}{M} \sum_{m=1}^{M} \mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})}; \boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\big) + \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}\big(f(\boldsymbol{x}_n; \boldsymbol{\phi}), \hat{\boldsymbol{y}}_n\big) \bigg\}, \tag{3.11}$$

where we have an additional tuning factor λ to balance the contribution between exact labels $\boldsymbol{y}_m^{(\text{labeled})}$ and hard pseudo-labels $\hat{\boldsymbol{y}}_n$. This can be expected to improve on the DNN accuracy provided that the pseudo-labels $\hat{\boldsymbol{y}}_n$ are more accurate than the non-corrected features $\boldsymbol{y}_n$. The method with feedback of hard pseudo-labels is depicted in Figure 3.3.

3.3.2. Semi-Soft Pseudo-Labels

In semi-soft pseudo-labeling, the basic idea is based on extending hard pseudo-labeling by incorporating the statistical information provided by the Bayesian estimator into the training of the DNN. This is achieved by modifying the loss function. For for the data pair $(\boldsymbol{x}_n, \hat{\boldsymbol{y}}_n)$ the modified loss function is defined as

$$\mathcal{L}_{\text{s}}\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \hat{\boldsymbol{y}}_n\big) \triangleq (f(\boldsymbol{x}_n;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_n)^T \big(\boldsymbol{C}_n^{(\mathcal{Y})}\big)^{-1}(f(\boldsymbol{x}_n;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_n), \tag{3.12}$$

with a weighting matrix $\boldsymbol{C}_n^{(\mathcal{Y})}$. This matrix is derived from the posterior covariance matrix $\boldsymbol{C}_n$ via a mapping

$$\boldsymbol{C}_n^{(\mathcal{Y})} = \boldsymbol{Y}(\boldsymbol{C}_n). \tag{3.13}$$

Similarly to the feedback function $\boldsymbol{y}(\,\cdot\,)$, also the choice of the function $\boldsymbol{Y}(\,\cdot\,)$ is based on the specific application considered. An example will be presented in Chapter 5, where we reduce the dimension of a 4×4 matrix by extracting the left upper block matrix of size 2×2.

For semi-soft pseudo-labeling, the set of pseudo-labels is given by

$$\mathcal{S}_{\text{PL}} = \big\{\big(\boldsymbol{x}_n, \hat{\boldsymbol{y}}_n, \boldsymbol{C}_n^{(\mathcal{Y})}\big)\big\}_{n=1}^{N}. \tag{3.14}$$

The empirical risk for semi-soft pseudo-labeling is

$$\begin{aligned}\mathcal{R}_{\text{new}}(\boldsymbol{\phi}) &= \frac{\eta}{M}\sum_{m=1}^{M}\mathcal{L}\big(f(\boldsymbol{x}_m^{(\text{labeled})};\boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\big) + \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_{\text{s}}(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \boldsymbol{y}(\hat{\boldsymbol{\theta}}_n)) \\ &= \frac{\eta}{M}\sum_{m=1}^{M}\mathcal{L}\left(f(\boldsymbol{x}_m^{(\text{labeled})};\boldsymbol{\phi}), \boldsymbol{y}_m^{(\text{labeled})}\right) \\ &\quad + \frac{1}{N}\sum_{n=1}^{N}(f(\boldsymbol{x}_n;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_n)^T \big(\boldsymbol{C}_n^{(\mathcal{Y})}\big)^{-1}(f(\boldsymbol{x}_n;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_n),\end{aligned} \tag{3.15}$$

with a tuning factor η, which needs to be chosen appropriately based on the entries of $\big(\boldsymbol{C}_n^{(\mathcal{Y})}\big)^{-1}$. We see that we have two contributions namely, the loss due to the labeled training data using a loss function $\mathcal{L}$ and the loss due to pseudo-labels using the modified loss function $\mathcal{L}_{\text{s}}$. For the latter, the statistical information of the Bayesian estimation stage is utilized by a weighting matrix $\big(\boldsymbol{C}_n^{(\mathcal{Y})}\big)^{-1}$. The block diagram in Figure 3.4 illustrates semi-soft pseudo labeling.

In the following, we motivate the definition of $\mathcal{L}_{\text{s}}$ in Equation (3.12) as a modified

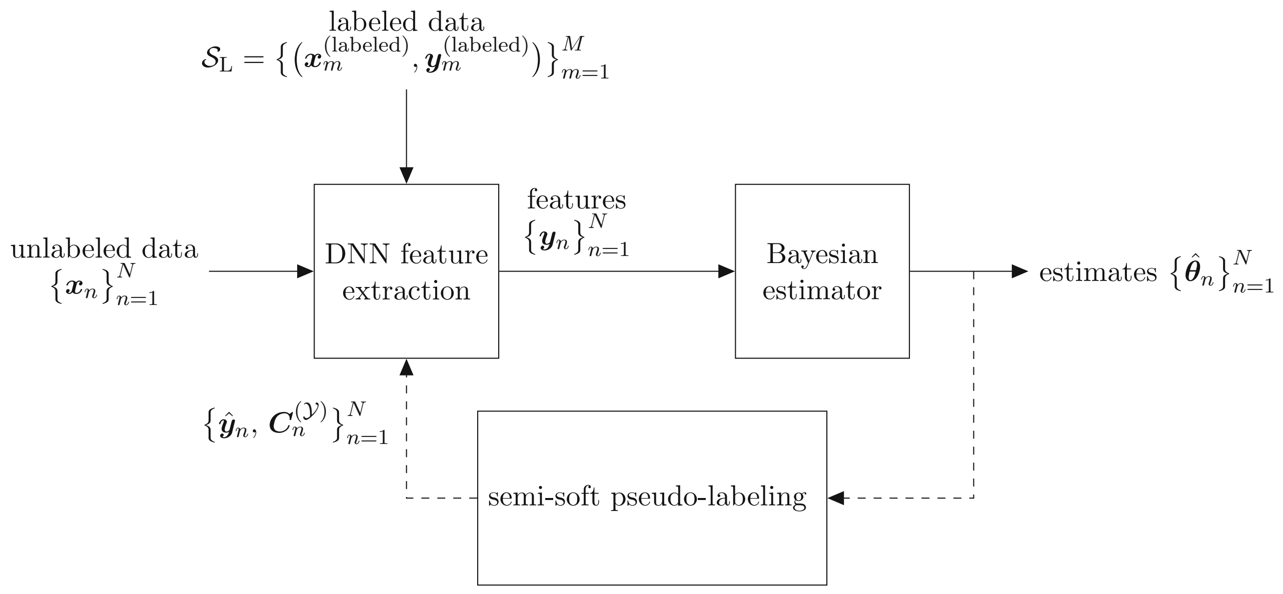

labeled data
$\mathcal{S}_\mathrm{L} = \{(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})})\}_{m=1}^{M}$
unlabeled data
$\{\boldsymbol{x}_n\}_{n=1}^{N}$
DNN feature extraction
features
$\{\boldsymbol{y}_n\}_{n=1}^{N}$
Bayesian estimator
estimates $\{\hat{\boldsymbol{\theta}}_n\}_{n=1}^{N}$
$\{\hat{\boldsymbol{y}}_n, \boldsymbol{C}_n^{(\mathcal{Y})}\}_{n=1}^{N}$
semi-soft pseudo-labeling

Figure 3.4.: Block diagram of the semi-supervised estimation-learning method using feedback of hard point estimates $\hat{\boldsymbol{y}}_n$ and transformed posterior covariance matrices $\boldsymbol{C}_n^{(\mathcal{Y})}$.

loss function. We recognize that $\mathcal{L}_\mathrm{s}$ has a similar structure as the squared Mahalanobis distance, see [30, 31], which for a realization $\boldsymbol{x} \in \mathbb{R}^L$ of a random vector with mean vector $\boldsymbol{\mu}$ and positive definite covariance matrix $\boldsymbol{C}$ is given by

$$d_\mathrm{M}^2(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{C}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}). \tag{3.16}$$

Next, we compare each component of Equation (3.12) and (3.16) and draw parallels. For d_M, we calculate the distance of a point $\boldsymbol{x}$ relative to the mean $\boldsymbol{\mu}$. In the modified loss $\mathcal{L}_\mathrm{s}$ we measure the distance relative to the pseudo-label $\hat{\boldsymbol{y}}_n$. Choosing the MMSE estimator for the parameter $\boldsymbol{\theta}_n$, we obtain the posterior mean. Since all estimates of the parameter $\boldsymbol{\theta}_n$ exist in Θ, we use the feedback function $\boldsymbol{y}(\,\cdot\,)$ to transform the point estimate into the feature space $\mathcal{Y}$. We recognize that d_M is centered around the mean $\boldsymbol{\mu}$, whereas $\mathcal{L}_\mathrm{s}$ is centered around a transformed posterior mean. Basically, every estimator could be used for $\boldsymbol{\theta}_n$, but the analogy drawn here only holds for the MMSE estimate. Similar to the transformation of the estimate $\hat{\boldsymbol{\theta}}_n$ using the feedback function, also the posterior covariance matrix $\boldsymbol{C}_n$ needs to be transformed, i.e., $\boldsymbol{Y}(\boldsymbol{C}_n)$. We again see the similarity between the Mahalanobis distance d_M and the modified loss $\mathcal{L}_\mathrm{s}$, where the inverse of the covariance $\boldsymbol{C}$ is used for calculating d_M and the inverse of the transformed posterior covariance matrix $\boldsymbol{C}_n^{(\mathcal{Y})}$ for the modified loss $\mathcal{L}_\mathrm{s}$.

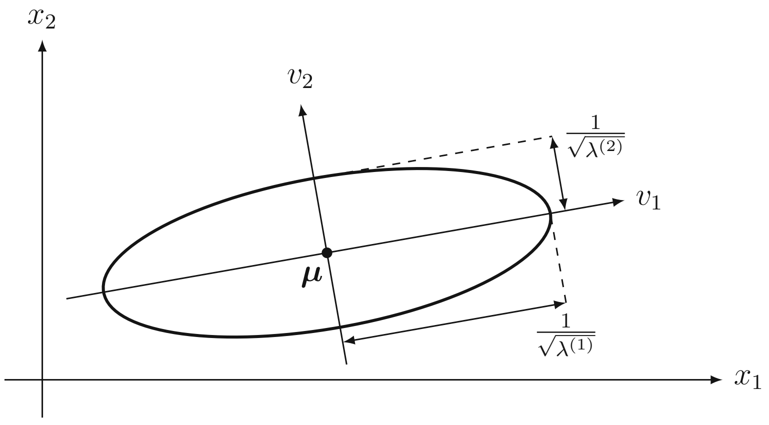

x_2
v_2
$\frac{1}{\sqrt{\lambda^{(2)}}}$
v_1
$\boldsymbol{\mu}$
$\frac{1}{\sqrt{\lambda^{(1)}}}$
x_1

Figure 3.5.: Illustration of the ellipsoid corresponding to the Mahalanobis distance for $L = 2$.

In the following, we give a graphical interpretation of the Mahalanonbis distance, [29, Chapter 2]. Let $\lambda^{(l)} > 0$ and $\boldsymbol{u}_l \in \mathbb{R}^L$ denote the the l-th eigenvalue and eigenvector of the matrix $\boldsymbol{C}$, respectively. Inserting the eigenvalue decomposition for the inverse of the covariance matrix $\boldsymbol{C}$ gives us

$$\boldsymbol{C}^{-1} = \boldsymbol{U}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}^T, \tag{3.17}$$

where $\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_L]$ and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda^{(1)}, \dots, \lambda^{(L)})$ are the matrices consisting of all eigenvectors and eigenvalues on the main diagonal, respectively. With the new coordinate system $\boldsymbol{v} = [v^{(1)}, \dots, v^{(L)}]^T$, given by

$$\boldsymbol{v} = \boldsymbol{U}^T(\boldsymbol{x} - \boldsymbol{\mu}), \tag{3.18}$$

the squared Mahalanobis distance $d^2_{\mathrm{M}}(\boldsymbol{x})$ can be rewritten as

$$d^2_{\mathrm{M}}(\boldsymbol{x}) = \boldsymbol{v}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{v} = \sum_{l=1}^{L} \frac{(v^{(l)})^2}{\lambda^{(l)}}. \tag{3.19}$$

For a fixed value d^2 of the squared Mahalanobis distance, (i.e., $d^2 = d^2_{\mathrm{M}}(\boldsymbol{x})$), Equation (3.19) defines an ellipsoid in the $\boldsymbol{v}$ coordinate system, with the length of the principal axes given by

$$l^{(i)} = \frac{1}{\sqrt{\lambda^{(i)}}}. \tag{3.20}$$

Figure 3.5 shows an example of such an ellipsoid for $L = 2$, displayed in the original ($\boldsymbol{x}$) coordinate system.

Gaussian case

For a Gaussian posterior $p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)$ and a linear feedback function $\boldsymbol{y}(\,\cdot\,)$ the features will also be Gaussian distributed according to

$$\mathcal{N}(\boldsymbol{y}_n;\hat{\boldsymbol{y}}_n,\boldsymbol{C}_n^{(\mathcal{Y})}) = \frac{1}{(2\pi)^{\frac{\dim(\mathcal{Y})}{2}}\det(\boldsymbol{C}_n^{(\mathcal{Y})})^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(\boldsymbol{y}_n-\hat{\boldsymbol{y}}_n)^T\left(\boldsymbol{C}_n^{(\mathcal{Y})}\right)^{-1}(\boldsymbol{y}_n-\hat{\boldsymbol{y}}_n)\right), \quad (3.21)$$

where $\hat{\boldsymbol{y}}_n$ is now the transformed MMSE estimate of $\boldsymbol{\theta}_n$. The Gaussian distribution (3.21) can be rewritten by using Equation (3.19)

$$\mathcal{N}(\boldsymbol{v}_n;\boldsymbol{0},\boldsymbol{\Lambda}_n) = \prod_{l=1}^{\dim(\mathcal{Y})}\frac{1}{\left(2\pi\lambda_n^{(l)}\right)^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\frac{(v_n^{(l)})^2}{\lambda_n^{(l)}}\right), \quad (3.22)$$

with $\lambda_n^{(i)}$ being the eigenvalues of $\boldsymbol{C}_n^{(\mathcal{Y})}$. The vector $\boldsymbol{v}_n$ is given by

$$\boldsymbol{v}_n = \boldsymbol{U}_n^T(\boldsymbol{y}_n-\hat{\boldsymbol{y}}_n) \quad (3.23)$$

with the eigenvalue decomposition of the transformed posterior covariance

$$\boldsymbol{C}_n^{(\mathcal{Y})} = \boldsymbol{U}_n\boldsymbol{\Lambda}_n\boldsymbol{U}_n^T. \quad (3.24)$$

Note that $\lambda_n^{(i)}$ represents the variance of the Gaussian distribution along the direction of $v_n^{(i)}$. Because the $\lambda_n^{(i)}$'s describe the variances along the new coordinate system, we can use them in the loss function as weights. Intuitively speaking, a small variance means that most parameters $\boldsymbol{\theta}_n$ lie in a small interval which corresponds to a high precision of the estimate. In the loss function, we want to emphasize these estimates, therefore we use the reciprocal of the eigenvalue as a weighting factor. Conversely, a large variance means a low precision of the estimate. Such estimates shall contribute only little to the loss which again leads to the motivation of using the reciprocal of the eigenvalue as a weighting factor.

3.3.3. Soft Pseudo-Labels

Instead of using hard point estimates $\hat{\boldsymbol{y}}_n$ as ground truth, see Section 3.3.1 and 3.3.2, soft pseudo-labeling follows a different approach. As for semi-soft pseudo-labeling, the statistical information of the Bayesian estimator is incorporated into the training of the DNN via a modified loss function. For soft pseudo-labeling $\mathcal{L}_\text{s}$, is given as the posterior

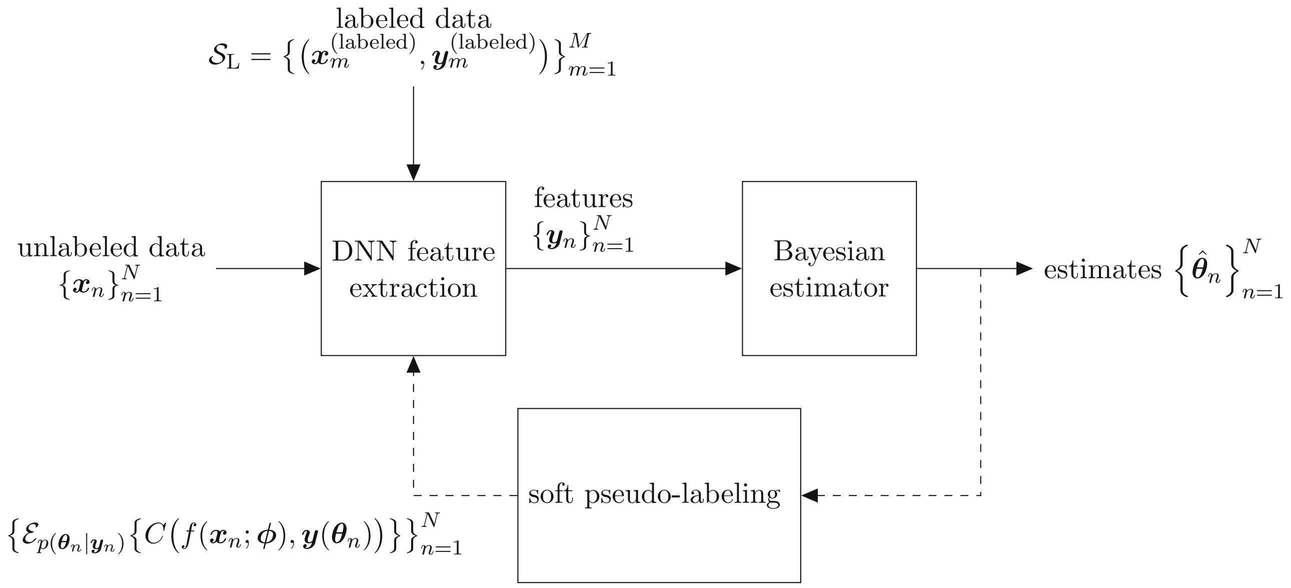

labeled data
$\mathcal{S}_{\mathrm{L}} = \{(\boldsymbol{x}_m^{(\text{labeled})}, \boldsymbol{y}_m^{(\text{labeled})})\}_{m=1}^{M}$
unlabeled data
$\{\boldsymbol{x}_n\}_{n=1}^{N}$
DNN feature extraction
features
$\{\boldsymbol{y}_n\}_{n=1}^{N}$
Bayesian estimator
estimates $\{\hat{\boldsymbol{\theta}}_n\}_{n=1}^{N}$
soft pseudo-labeling
$\{\mathcal{E}_{p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)}\{C(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \boldsymbol{y}(\boldsymbol{\theta}_n))\}\}_{n=1}^{N}$

Figure 3.6.: Block diagram of the semi-supervised learning-estimation method using feedback of the posterior distribution $p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)$.

expectation of some suitably chosen cost function [32], i.e.,

$$\mathcal{L}_{\mathrm{s}}(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \boldsymbol{y}(\boldsymbol{\theta}_n)) \triangleq \mathcal{E}_{p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)}\big\{C\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \boldsymbol{y}(\boldsymbol{\theta}_n)\big)\big\}, \tag{3.25}$$

with some non-negative cost function [22, Chapter 1]

$$C : \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times \mathcal{Y} \to \mathbb{R}_0^+,$$

where $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ denotes the set of functions mapping $\mathcal{X} \to \mathcal{Y}$. This is a reasonable loss function because via the cost function $C(\,\cdot\,)$ we can measure the distance between the DNN output $\boldsymbol{y}_n = f(\boldsymbol{x}_n;\boldsymbol{\phi})$ and the transformed estimate $\boldsymbol{y}(\boldsymbol{\theta}_n)$. In general, the cost function should be chosen such that a large difference between $f(\boldsymbol{x}_n;\boldsymbol{\phi})$ and $\boldsymbol{y}(\boldsymbol{\theta}_n)$ will lead to large value of $C\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \boldsymbol{y}(\boldsymbol{\theta}_n)\big)$. Via the expectation operator, we incorporate the posterior distribution into the loss function. This means for highly probable values of $\boldsymbol{\theta}_n$ we have a larger contribution to $\mathcal{L}_{\mathrm{s}}$ due to high values of $p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)$.

Calculating the expectation in Equation (3.25) can be challenging. First, we need the posterior distribution, which can be obtained by Bayes' rule. This involves solving, a possibly high dimensional integral, to get the evidence $p(\boldsymbol{y}_n)$. Furthermore, we need to compute the posterior expectation of $C\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}), \boldsymbol{y}(\boldsymbol{\theta}_n)\big)$ which may not be solvable in closed form for certain choices of $C(\,\cdot\,)$ and feedback functions $\boldsymbol{y}(\boldsymbol{\theta}_n)$. If an analytical solution for 3.25 cannot be obtained, we may use Monte Carlo methods [29, Chapter 11] to approximate the integral, i.e.,

$$\begin{aligned}\mathcal{E}_{p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)}\{C\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_n)\big)\} &= \int_{\Theta} p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)C\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_n)\big)d\boldsymbol{\theta}_n \\ &\approx \frac{1}{L_{\mathrm{MC}}}\sum_{l=1}^{L_{\mathrm{MC}}} C\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_n^{(l)})\big), \end{aligned} \tag{3.26}$$

where we draw L_{MC} independent and identically (i.i.d.) distributed samples from the posterior pdf, i.e., $\boldsymbol{\theta}_n^{(l)} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)$. In Section 5.3.3 we will present an application where we can find closed-form solutions of Equation (3.25).

For soft pseudo-labeling, the set of pseudo-labels is given as

$$\mathcal{S}_{\mathrm{PL}} = \big\{\big(\boldsymbol{x}_n, \boldsymbol{y}(\boldsymbol{\theta}_n)\big\}_{n=1}^{N}. \tag{3.27}$$

The overall empirical risk is given as

$$\begin{aligned}\mathcal{R}_{\mathrm{new}}(\boldsymbol{\phi}) &= \frac{\mu}{M}\sum_{m=1}^{M}\mathcal{L}\big(f(\boldsymbol{x}_m^{(\mathrm{labeled})};\boldsymbol{\phi}),\boldsymbol{y}_m^{(\mathrm{labeled})}\big) + \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_{\mathrm{s}}(f(\boldsymbol{x}_n;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_n)) \\ &= \frac{\mu}{M}\sum_{m=1}^{M}\mathcal{L}\left(f(\boldsymbol{x}_m^{(\mathrm{labeled})};\boldsymbol{\phi}),\boldsymbol{y}_m^{(\mathrm{labeled})}\right) + \frac{1}{N}\sum_{n=1}^{N}\mathcal{E}_{p(\boldsymbol{\theta}_n|\boldsymbol{y}_n)}\{C\big(f(\boldsymbol{x}_n;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_n)\big)\}.\end{aligned}$$

with tuning factor μ, similar to 3.15, to balance the weight contribution between labeled data and pseudo-labels. Figure 3.6 shows a block diagram for soft pseudo-labeling.

4. Deep Neural Network Aided Sequential Bayesian Estimation

In this chapter, we want to extend the semi-supervised learning-estimation method to sequential Bayesian estimation. Rather than considering a set of observations $\{\boldsymbol{x}_n\}$, features $\{\boldsymbol{y}_n\}$ and parameter estimates $\{\hat{\boldsymbol{\theta}}_n\}$ we consider sequences of observations, features, and parameter estimates $\boldsymbol{x}_k$, $\boldsymbol{y}_k$, $\hat{\boldsymbol{\theta}}_k$, $k = 1, \ldots,$ where we have a time index k. We start by presenting the general sequential Bayesian estimator and then specialize on the linear-Gaussian case, namely the KF. References for this chapter are [2, 27, 28].

4.1. State-Space Model

Let us consider a general state-space model [33] for a sequence of parameters $\boldsymbol{\theta}_k$ consisting of a state-transition model

$$\boldsymbol{\theta}_k = \boldsymbol{a}_k(\boldsymbol{\theta}_{k-1}, \boldsymbol{u}_k), \quad k = 1, 2, \ldots, \tag{4.1}$$

and a measurement model

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{\theta}_k, \boldsymbol{w}_k), \quad k = 1, 2, \ldots, \tag{4.2}$$

with generally time-dependent, nonlinear and known functions $\boldsymbol{a}_k(\,\cdot\,,\,\cdot\,)$ and $\boldsymbol{h}_k(\,\cdot\,,\,\cdot\,)$. In Equation (4.1) we have white driving noise $\boldsymbol{u}_k$ with known pdf $p(\boldsymbol{u}_k)$, whereas in Equation (4.2) we have white measurement noise $\boldsymbol{w}_k$ again with known pdf $p(\boldsymbol{w}_k)$. The state-transition model is initialized with a realization of $\boldsymbol{\theta}_0$ with known pdf $p(\boldsymbol{\theta}_0)$. Furthermore, driving noise $\boldsymbol{u}_k$, measurement noise $\boldsymbol{w}_k$ and initial parameter $\boldsymbol{\theta}_0$ are independent.

Equation (4.1) and (4.2) describe a hidden Markov model (HMM) with unobserved parameter $\boldsymbol{\theta}_k$, forming a Markov chain, and measurements $\boldsymbol{y}_k$. A Bayesian network of the HMM is given in Figure 4.1.

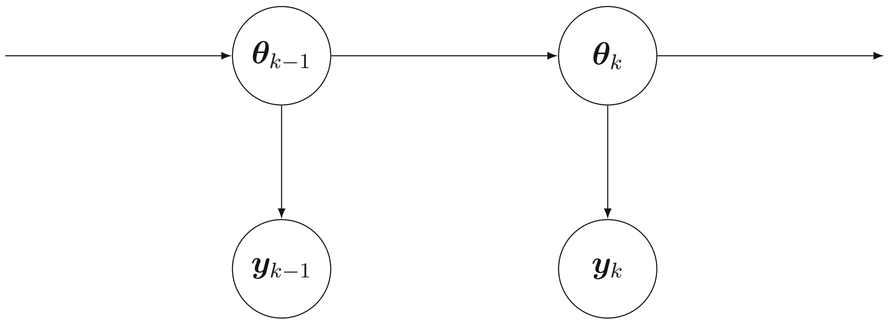


Figure 4.1.: Bayesian network of the HMM [2, Chapter 1] for the state-transition model (4.1) and measurement model (4.2).

Due to the HMM we have the following independence properties:

- For the state-transition pdf $p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1})$ the following holds

$$p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}) = p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, \boldsymbol{y}_{1:k-1}), \tag{4.3}$$

  with $\boldsymbol{y}_{1:k-1} = [\boldsymbol{y}_1^T, \dots, \boldsymbol{y}_{k-1}^T]^T$.

- For the likelihood function $p(\boldsymbol{y}_k|\boldsymbol{\theta}_k)$ we have

$$p(\boldsymbol{y}_k|\boldsymbol{\theta}_k) = p(\boldsymbol{y}_k|\boldsymbol{\theta}_k, \boldsymbol{y}_{1:k-1}). \tag{4.4}$$

4.2. Sequential Bayesian Estimation

Our goal is to derive an expression for the posterior pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})$, with $\boldsymbol{y}_{1:k} \triangleq [\boldsymbol{y}_1^T, \dots, \boldsymbol{y}_k^T]^T$, which can be used to calculate different Bayesian estimators, e.g. MMSE estimate

$$\hat{\boldsymbol{\theta}}_{k|k} = \int_{\Theta} \boldsymbol{\theta}_k p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}) d\boldsymbol{\theta}_k. \tag{4.5}$$

The calculation of the posterior distribution $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})$ is achieved in two steps, namely a prediction step and an update step [2, 34]:

1. PREDICTION STEP: First, we calculate the predicted posterior $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})$. Due to Equation (4.3) we have the property

$$p(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}) = p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1})$$

$$= p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1})p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}), \tag{4.6}$$

where $p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1})$ is the posterior pdf at time step $k-1$. Marginalization of $\boldsymbol{\theta}_{k-1}$ gives the predicted posterior

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}) = \int_{\Theta} p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1})p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1})d\boldsymbol{\theta}_{k-1}. \tag{4.7}$$

2. UPDATE STEP: Next, the posterior pdf at time step k can be calculated by using Bayes' rule, i.e.,

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}) = p(\boldsymbol{\theta}_k|\boldsymbol{y}_k, \boldsymbol{y}_{1:k-1}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{\theta}_k, \boldsymbol{y}_{1:k-1})p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})}. \tag{4.8}$$

Using Equation (4.4), the expression for the posterior in (4.8) simplifies to

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})}. \tag{4.9}$$

The denominator of Equation (4.9) can be expanded as

$$p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) = \int_{\Theta} p(\boldsymbol{y}_k|\boldsymbol{\theta}_k, \boldsymbol{y}_{1:k-1})p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})d\boldsymbol{\theta}_k = \int_{\Theta} p(\boldsymbol{y}_k|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})d\boldsymbol{\theta}_k, \tag{4.10}$$

where Equation (4.4) was used again.

4.3. Kalman Filter

The KF [27, Chapter 13] is a method for sequential MMSE estimation based on a linear-Gaussian state-space model for the parameter vector at time k $\boldsymbol{\theta}_k$, involving measurements (or in our case features) $\boldsymbol{y}_k \in \mathbb{R}^n$. The state-transition model is

$$\boldsymbol{\theta}_k = \boldsymbol{A}_k\boldsymbol{\theta}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_k \quad \text{for} \quad k = 1, 2, 3, \dots, \tag{4.11}$$

with the state-transition matrix $\boldsymbol{A}_k \in \mathbb{R}^{p \times p}$, the driving noise vector $\boldsymbol{u}_k \in \mathbb{R}^r$, and a matrix $\boldsymbol{B}_k \in \mathbb{R}^{p \times r}$. The measurement model is

$$\boldsymbol{y}_k = \boldsymbol{H}_k\boldsymbol{\theta}_k + \boldsymbol{w}_k \quad \text{for} \quad k = 1, 2, 3, \dots, \tag{4.12}$$

with a matrix $\boldsymbol{H}_k \in \mathbb{R}^{n \times p}$ and the measurement noise $\boldsymbol{w} \in \mathbb{R}^n$.

We make the following assumptions:

- The driving noise $\boldsymbol{u}_k$ is a zero-mean and white random process, i.e.,

$$\mathrm{E}\{\boldsymbol{u}_k\} = \boldsymbol{0}, \quad \mathrm{cov}\{\boldsymbol{u}_k, \boldsymbol{u}_j\} = \mathrm{E}\{(\boldsymbol{u}_k - \mathrm{E}\{\boldsymbol{u}_k\})(\boldsymbol{u}_j - \mathrm{E}\{\boldsymbol{u}_j\})^T\} = \mathrm{E}\{\boldsymbol{u}_k \boldsymbol{u}_j^T\} = \boldsymbol{C}_{u,k}\delta_{kj}, \tag{4.13}$$

  with $\delta_{kj} = 1$ for $k = j$ and $\delta_{kj} = 0$ for $k \neq j$.

- The measurement noise $\boldsymbol{w}_k$ is a zero-mean and white generally non-stationary process, i.e.,

$$\mathrm{E}\{\boldsymbol{w}_k\} = \boldsymbol{0}, \quad \mathrm{cov}\{\boldsymbol{w}_k, \boldsymbol{w}_j\} = \mathrm{E}\{\boldsymbol{w}_k \boldsymbol{w}_j^T\} = \boldsymbol{C}_{w,k}\delta_{kj}. \tag{4.14}$$

- The driving noise $\boldsymbol{u}_k$, the measurement noise $\boldsymbol{w}_k$ and the initial parameter $\boldsymbol{\theta}_0$ are jointly Gaussian, with each of them distributed according to:

$$p(\boldsymbol{u}_k) = \mathcal{N}(\boldsymbol{u}_k; \boldsymbol{0}, \boldsymbol{C}_{u,k}) \tag{4.15}$$
$$p(\boldsymbol{w}_k) = \mathcal{N}(\boldsymbol{w}_k; \boldsymbol{0}, \boldsymbol{C}_{w,k}) \tag{4.16}$$
$$p(\boldsymbol{\theta}_0) = \mathcal{N}(\boldsymbol{\theta}_0; \boldsymbol{\mu}_\theta, \boldsymbol{C}_\theta) \tag{4.17}$$

- The driving noise $\boldsymbol{u}_k$ and the measurement noise $\boldsymbol{w}_k$ are uncorrelated, i.e.,

$$\mathrm{cov}\{\boldsymbol{u}_k, \boldsymbol{w}_k\} = \mathrm{E}\{\boldsymbol{w}_k \boldsymbol{u}_k^T\} = \boldsymbol{0}\ \forall k. \tag{4.18}$$

- The initial state $\boldsymbol{\theta}_0$ is uncorrelated withe the driving noise $\boldsymbol{u}_k$ and the measurement noise $\boldsymbol{w}_k$, i.e.,

$$\mathrm{cov}\{\boldsymbol{u}_k, \boldsymbol{\theta}_0\} = \mathcal{E}\{\boldsymbol{u}_k \boldsymbol{\theta}_0^T\} = \boldsymbol{0} \quad \text{and} \quad \mathrm{cov}\{\boldsymbol{w}_k, \boldsymbol{\theta}_0\} = \mathcal{E}\{\boldsymbol{w}_k \boldsymbol{\theta}_0^T\} = \boldsymbol{0}. \tag{4.19}$$

- For the initial parameter values $\boldsymbol{\theta}_0$ we assume that the mean $\boldsymbol{\mu}_\theta$ and covariance matrix $\boldsymbol{C}_\theta$ are known.

For the linear-Gaussian state-transition (4.11) and measurement model (4.12) the integrals of the prediction (4.7) and update step (4.10) can be solved in closed form. We have the following result (derivation in Appendix A):

1. **KF Prediction step:**

   a) Predicted posterior mean: $\hat{\boldsymbol{\theta}}_{k|k-1} = \boldsymbol{A}_k \hat{\boldsymbol{\theta}}_{k-1|k-1}$

b) Predicted posterior covariance: $\boldsymbol{C}_{k|k-1} = \boldsymbol{A}_k \boldsymbol{C}_{k-1|k-1} \boldsymbol{A}_k^T + \boldsymbol{B}_k \boldsymbol{C}_{u,k} \boldsymbol{B}_k^T$

2. **KF Update step:**

a) Kalman gain matrix: $\boldsymbol{K}_k = \boldsymbol{C}_{k|k-1} \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{C}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{C}_{w,k})^{-1}$

b) Posterior mean: $\hat{\boldsymbol{\theta}}_{k|k} = \hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k \hat{\boldsymbol{\theta}}_{k|k-1})$

c) Posterior covariance: $\boldsymbol{C}_{k|k} = (\mathbf{I} - \boldsymbol{K}_k \boldsymbol{H}_k)\boldsymbol{C}_{k|k-1}$

The KF recursion given above, performed for time steps $k = 1, 2, \ldots,$ is initialized with

$$\hat{\boldsymbol{\theta}}_{0|0} = \boldsymbol{\mu}_\theta \quad \text{and} \quad \boldsymbol{C}_{0|0} = \boldsymbol{C}_\theta. \tag{4.20}$$

Since $\hat{\boldsymbol{\theta}}_{0|0}$ and $\boldsymbol{C}_{0|0}$ is assumed to be known, we can precalculate $\boldsymbol{C}_{k|k-1}$, $\boldsymbol{C}_{k|k}$ and $\boldsymbol{K}_k$ previous to observing features/measurements $\boldsymbol{y}_k$.

4.3.1. Error Covariance

In this section, we want to calculate the predicted error covariance matrix $\boldsymbol{C}^{(e)}_{k|k-1}$ and the posterior error covariance matrix $\boldsymbol{C}^{(e)}_{k|k}$. As a first step, we introduce the posterior estimation error as

$$\boldsymbol{e}_{k|k} = \boldsymbol{\theta}_k - \hat{\boldsymbol{\theta}}_{k|k}, \tag{4.21}$$

which allows us to write the posterior covariance as

$$\boldsymbol{C}^{(e)}_{k|k} = \mathcal{E}\big\{\big(\boldsymbol{e}_{k|k} - \mathcal{E}\{\boldsymbol{e}_{k|k}\}\big)\big(\boldsymbol{e}_{k|k} - \mathcal{E}\{\boldsymbol{e}_{k|k}\}\big)^T\big\}. \tag{4.22}$$

Because the MMSE estimator is unbiased, i.e., $\mathcal{E}\big\{\boldsymbol{e}_{k|k}\big\} = \mathbf{0}$ we have the following identity

$$\boldsymbol{C}^{(e)}_{k|k} = \mathcal{E}\{\boldsymbol{e}_{k|k}\boldsymbol{e}^T_{k|k}\} = \mathcal{E}\big\{\big(\boldsymbol{\theta}_k - \hat{\boldsymbol{\theta}}_{k|k}\big)\big(\boldsymbol{\theta}_k - \hat{\boldsymbol{\theta}}_{k|k}\big)^T\big\} = \boldsymbol{C}_{k|k}. \tag{4.23}$$

Similar to Equation (4.21) the predicted estimation error is given by

$$\boldsymbol{e}_{k|k-1} = \boldsymbol{\theta}_k - \hat{\boldsymbol{\theta}}_{k|k-1}, \tag{4.24}$$

with the predicted error covariance

$$\boldsymbol{C}^{(e)}_{k|k-1} = \mathcal{E}\big\{\big(\boldsymbol{e}_{k|k-1} - \mathcal{E}\{\boldsymbol{e}_{k|k-1}\}\big)\big(\boldsymbol{e}_{k|k-1} - \mathcal{E}\{\boldsymbol{e}_{k|k-1}\}\big)^T\big\}. \tag{4.25}$$

Due to the KF equations, the predicted posterior estimate is given by

$$\hat{\boldsymbol{\theta}}_{k|k-1} = \boldsymbol{A}_k \hat{\boldsymbol{\theta}}_{k-1|k-1}. \tag{4.26}$$

Since the MMSE commutes over affine transformation, as in (4.26), $\hat{\boldsymbol{\theta}}_{k|k-1}$ is also a MMSE estimator. Following the same argument as for $\boldsymbol{C}_{k|k}^{(e)}$, we get

$$\boldsymbol{C}_{k|k-1}^{(e)} = \boldsymbol{C}_{k|k-1}. \tag{4.27}$$

4.4. Feedback of Pseudo-Labels

The proposed structure for feeding back pseudo-labels, as introduced in Section 3.3, can easily be extended to sequential Bayesian estimation, e.g., using the KF. Here, we want to calculate estimates $\hat{\boldsymbol{\theta}}_{k|k}$ for a sequence of parameters that satisfy the state-transition model (4.1) based on a sequence of features/measurements that satisfy the measurement model (4.2). In order to connect a general sequential Bayesian estimator and a DNN, each feature vector $\boldsymbol{y}_k$ is associated with an observation $\boldsymbol{x}_k$. This allows us to calculate parameter estimates $\hat{\boldsymbol{\theta}}_{k|k}$ from observations $\boldsymbol{x}_k$. In the following sections, we specialize hard, semi-soft, and soft pseudo-labeling to the case of sequential Bayesian estimation.

4.4.1. Hard Pseudo-Labeling

Using the feedback function $\boldsymbol{y}(\,\cdot\,)$ we can calculate the corrected feature vector at time k based on k measurements in the following way

$$\hat{\boldsymbol{y}}_{k|k} = \boldsymbol{y}(\hat{\boldsymbol{\theta}}_{k|k}). \tag{4.28}$$

For a sequence of length K, we can construct the data set consisting of hard pseudo-labels as

$$\mathcal{S}_{\mathrm{PL}} = \left\{(\boldsymbol{x}_k, \hat{\boldsymbol{y}}_{k|k})\right\}_{k=1}^{K}. \tag{4.29}$$

Similar to Equation (3.11) we can write the empirical risk as

$$\mathcal{R}_{\mathrm{new}}(\boldsymbol{\phi}) = \frac{\lambda}{M} \sum_{m=1}^{M} \mathcal{L}\big(f(\boldsymbol{x}_m^{(\mathrm{labeled})}; \boldsymbol{\phi}), \boldsymbol{y}_m^{(\mathrm{labeled})}\big) + \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}\big(f(\boldsymbol{x}_k; \boldsymbol{\phi}), \hat{\boldsymbol{y}}_{k|k}\big). \tag{4.30}$$

4.4.2. Semi-Soft Pseudo Labeling

For semi-soft pseudo-labeling, we first transform the updated posterior covariance according to

$$\boldsymbol{C}_{k|k}^{(\mathcal{Y})} = \boldsymbol{Y}(\boldsymbol{C}_{k|k}). \tag{4.31}$$

In analogy to Equation (3.12) the modified loss function is then

$$\mathcal{L}_{\mathrm{s}}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\hat{\boldsymbol{y}}_{k|k}\big) = (f(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_{k|k})^T \big(\boldsymbol{C}_{k|k}^{(\mathcal{Y})}\big)^{-1}(f(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_{k|k}). \tag{4.32}$$

For the set of pseudo-labels $\mathcal{S}_{\mathrm{PL}} = \big\{\big(\boldsymbol{x}_k, \hat{\boldsymbol{y}}_{k|k}, \boldsymbol{C}_{k|k}^{(\mathcal{Y})}\big)\big\}_{k=1}^{K}$, we can rewrite the empirical risk for the sequential case

$$\begin{aligned}\mathcal{R}_{\mathrm{new}}(\boldsymbol{\phi}) = &\frac{\eta}{M}\sum_{m=1}^{M}\mathcal{L}\left(f(\boldsymbol{x}_m^{(\mathrm{labeled})};\boldsymbol{\phi}),\boldsymbol{y}_m^{(\mathrm{labeled})}\right)\\ &+\frac{1}{K}\sum_{k=1}^{K}(f(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_{k|k})^T\big(\boldsymbol{C}_{k|k}^{(\mathcal{Y})}\big)^{-1}(f(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_{k|k}).\end{aligned} \tag{4.33}$$

4.4.3. Soft Pseudo-Labeling

For soft pseudo-labeling, the modified loss is based on the posterior density $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})$. We can write $\mathcal{L}_{\mathrm{s}}$ as

$$\mathcal{L}_{\mathrm{s}}(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)) \triangleq \mathcal{E}_{p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})}\big\{C\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big)\big\}. \tag{4.34}$$

For the set of soft pseudo-labels $\mathcal{S}_{\mathrm{PL}} = \big\{\big(\boldsymbol{x}_k,\boldsymbol{y}(\boldsymbol{\theta}_k)\big\}_{k=1}^{K}$, the empirical risk is given by

$$\mathcal{R}_{\mathrm{new}}(\boldsymbol{\phi}) = \frac{\mu}{M}\sum_{m=1}^{M}\mathcal{L}\big(f(\boldsymbol{x}_m^{(\mathrm{labeled})};\boldsymbol{\phi}),\boldsymbol{y}_m^{(\mathrm{labeled})}\big) + \frac{1}{K}\sum_{k=1}^{K}\mathcal{E}_{p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})}\big\{C\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big)\big\}. \tag{4.35}$$

5. Application to Target Tracking

We now will apply the theory presented in Chapters 3 and 4 to target tracking. More specifically, we consider the task of estimating over time the position and velocity of an object (or target) moving in the two-dimensional real space $\mathbb{R}^2$ using DNN-aided sequential Bayesian estimation. At each time step $k \in \{1, \dots, K\}$, an image is captured and processed by a DNN, yielding a feature in the form of a two-dimensional "predicted position" vector $\boldsymbol{y}_k$. This task is also referred to as object localization. The predicted position $\boldsymbol{y}_k$ is then used as the measurement vector for a KF which tries to improve the position prediction $\boldsymbol{y}_k$ obtained by the DNN, besides estimating the two-dimensional velocity of the object. This improvement is possible because the KF, in contrast to the DNN, exploits the state-transition model describing the change of position and velocity over time as well as the measurement model describing the process of obtaining noisy position vectors. Furthermore, we propose a method for generating pseudo-labels based on the improved position estimates produced by the KF, as well as a method of retraining the DNN using the pseudo-labels and the statistical information provided by the KF.

5.1. Motion and Observation Model

We consider an object moving across the two-dimensional real plane. This movement is represented at each time step k by the state

$$\boldsymbol{\theta}_k = \left[p_k^{(1)},\, p_k^{(2)},\, v_k^{(1)},\, v_k^{(2)}\right]^T, \tag{5.1}$$

where $p_k^{(1)}$ and $p_k^{(1)}$ denote the object's position coordinates at time k and $v_k^{(1)}$ and $v_k^{(2)}$ describe the velocities in the respective directions. In the simple case of a constant-velocity movement along a straight line, the time evolution of the two-dimensional position vector $\boldsymbol{p}_k = \left[p_k^{(1)},\, p_k^{(2)}\right]^T$ is described as

$$\boldsymbol{p}_k = \boldsymbol{p}_{k-1} + \Delta t\, \boldsymbol{v}_{k-1}, \tag{5.2}$$

where Δt is the time period between two successive time instances $k-1$ and k and $\boldsymbol{v}_{k-1} = \left[v_{k-1}^{(1)},\ v_{k-1}^{(2)}\right]^T$ is the two-dimensional velocity vector. We can extend this model according to

$$\boldsymbol{p}_k = \boldsymbol{p}_{k-1} + \Delta t\, \boldsymbol{v}_{k-1} + \boldsymbol{u}_{p,k}, \tag{5.3}$$

where $\boldsymbol{u}_{p,k} = \left[u_{p,k}^{(1)},\ u_{p,k}^{(2)}\right]^T$ is a random perturbation affecting the position. In addition, we introduce a random perturbation $\boldsymbol{u}_{v,k} = \left[u_{v,k}^{(1)},\ u_{v,k}^{(2)}\right]^T$ of the velocity vector $\boldsymbol{v}_k$, thus obtaining the final a linear state-evolution model

$$\boldsymbol{\theta}_k = \boldsymbol{A}\boldsymbol{\theta}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k, \tag{5.4}$$

with

$$\boldsymbol{A} \triangleq \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B} \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{u}_k = \begin{bmatrix} u_{p,k}^{(1)} \\ u_{p,k}^{(2)} \\ u_{v,k}^{(1)} \\ u_{v,k}^{(2)} \end{bmatrix}. \tag{5.5}$$

This motion model is also referred to as nearly constant-velocity model [35], with the vector $\boldsymbol{u}_k$ being the driving noise. The driving noise $\boldsymbol{u}_k$ follows a zero-mean Gaussian distribution, i.e.,

$$p(\boldsymbol{u}_k) = \mathcal{N}(\boldsymbol{u}_k; \boldsymbol{0}, \boldsymbol{C}_u), \text{ with } \boldsymbol{C}_u = \operatorname{diag}\left(\left(\sigma_u^{(1)}\right)^2, \ldots, \left(\sigma_u^{(4)}\right)^2\right). \tag{5.6}$$

A graphical visualization can be seen in Figure 5.1.

Our measurements are the feature vectors $\boldsymbol{y}_k$ produced by the DNN. In the absence of a more pertinent model of the $\boldsymbol{y}_k$, we model them as noisy versions of the position vector $\boldsymbol{p}_k = \left[p_k^{(1)},\ p_k^{(2)}\right]^T$, i.e.,

$$\boldsymbol{y}_k = \boldsymbol{p}_k + \boldsymbol{w}_k, \tag{5.7}$$

with measurement noise vector $\boldsymbol{w}_k = [w_k^{(1)},\ w_k^{(2)}]^T$. This can be rewritten in term of the state vector $\boldsymbol{\theta}_k$ in (5.1) as

$$\boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{\theta}_k + \boldsymbol{w}_k, \tag{5.8}$$

with

$$\boldsymbol{H} \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{5.9}$$

We see that the measurement vector $\boldsymbol{y}_k$ is a linear function of the state $\boldsymbol{\theta}_k$. Like the driving

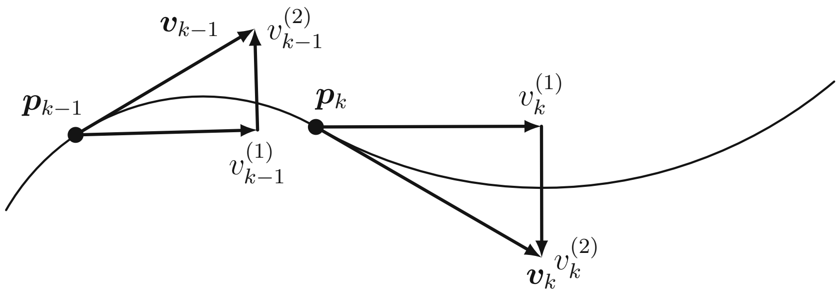


Figure 5.1.: Visualzation of the nearly constant velocity model, describing the movement of a point target along a trajectory in $\mathbb{R}^2$.

noise $\boldsymbol{u}_k$, also the measurement noise $\boldsymbol{w}_k$ follows a Gaussian distribution

$$p(\boldsymbol{w}_k) = \mathcal{N}(\boldsymbol{w}_k; \boldsymbol{0}, \boldsymbol{C}_{w,k}), \text{ with } \boldsymbol{C}_{w,k} = \operatorname{diag}\left(\left(\sigma_{w,k}^{(1)}\right)^2, \left(\sigma_{w,k}^{(2)}\right)^2\right). \tag{5.10}$$

5.2. Deep Neural Network for Position Prediction

To complete the position prediction task, we use a modified architecture of a previously proposed DNN, see [11]. We use a simplified version of configuration B given in [11] in order to achieve faster training times without sacrificing too much in prediction accuracy. A graphical visualization of this DNN is given in Figure 5.2 where the input is an image containing a blue circular target. The DNN predicts the coordinates of the center of this circular target, i.e., $\boldsymbol{p}_k = \left[p_k^{(1)}, p_k^{(2)}\right]^T$. The network consists of a convolutional block of ten weight layers and a fully connected block of two layers. The convolutional block has five sub-blocks, where each sub-block consists of two convolutional layers with the number of filters in each sub-block specified in Figure 5.2 followed by a max pooling layer. After the convolutional block, we have a flattening operation followed by another fully connected layer at the output to give the two-dimensional position prediction. We set the dropout parameter to 0.5 for regularization. The nonlinear activation function for both the convolutional block as well as the fully connected block is the rectified linear unit (ReLU) function, that is $\rho(x) = \max(0, x)$. In summary, we have a DNN with 12 weight layers, specified through training. The total number of weights is 8.7×10^6. The modifications to configuration B given in [11] are as follows. First, we decreased the number of filters with each convolutional sub-block. That is we start with eight filters in the first two

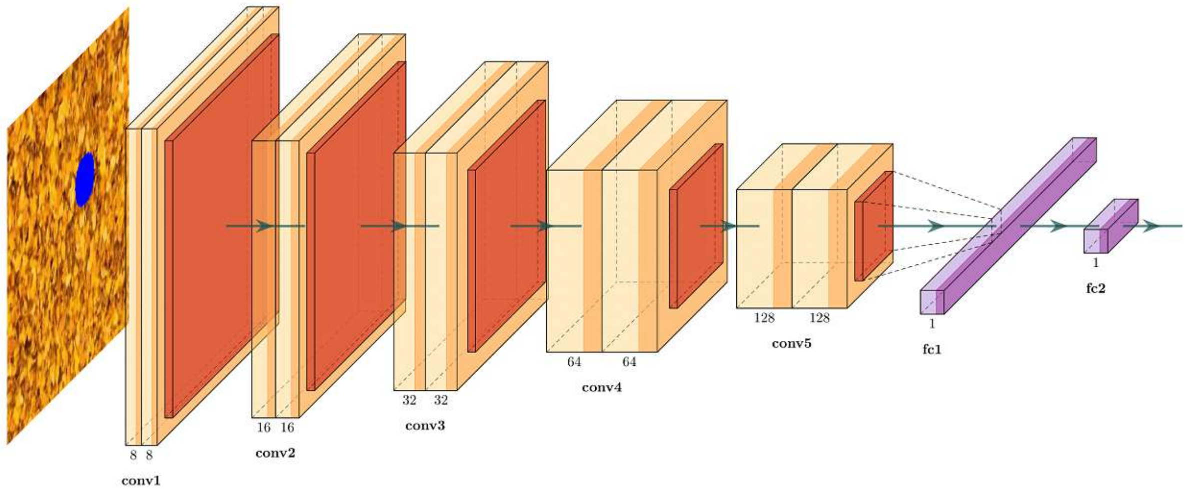


Figure 5.2.: DNN architecture used for position prediction. We see the first part consisting of five convolutional sub-blocks **conv1** up to **conv5**, indicated by the yellow cuboids. The number of filters is specified beneath. After each convolutional sub-block, we can see the max pooling layer, indicated by the red cuboid. The fully connected block is given by the two purple layers **fc1** and **fc2**. A modification of the LaTeX code provided in [3] is used.

convolutional layers, where in [11] the initial amount is 64. Second, we reduce the number of fully connected layers by one. Furthermore, because configuration B was used for a classification task we omit the soft-max layer at the end.

5.3. Feedback of Position Estimates

In this section, we specialize hard, semi-soft, and soft pseudo-labeling to the application of target tracking. In view of our measurement model (5.8), $\boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{\theta}_k + \boldsymbol{w}_k$, we here define the feedback function $\boldsymbol{y}(\,\cdot\,)$, (see Equation (3.8)), as

$$\boldsymbol{y}(\boldsymbol{\theta}_k) = \boldsymbol{H}\boldsymbol{\theta}_k = \begin{bmatrix} p_k^{(1)} \\ p_k^{(2)} \end{bmatrix}. \tag{5.11}$$

We can use the KF to sequentially calculate the MMSE estimate of the state $\boldsymbol{\theta}_k$, i.e.,

$$\hat{\boldsymbol{\theta}}_{k|k} = \left[\hat{p}_{k|k}^{(1)},\ \hat{p}_{k|k}^{(2)},\ \hat{v}_{k|k}^{(1)},\ \hat{v}_{k|k}^{(2)}\right]^T \tag{5.12}$$

and calculate corrected features

$$\hat{\boldsymbol{y}}_{k|k} = \boldsymbol{H}\hat{\boldsymbol{\theta}}_{k|k}. \tag{5.13}$$

5.3.1. Hard Pseudo-Labels

Let us consider the first option of feeding back corrected feature vectors, namely hard pseudo-labeling, as introduced in Section 4.4.1. The hard pseudo-label is given in Equation (5.12). The statistical information provided by the KF, i.e., the posterior distribution as expressed by $\mathcal{N}(\boldsymbol{\theta}_k; \hat{\boldsymbol{\theta}}_{k|k}, \boldsymbol{C}_{k|k})$ is not used. Choosing the squared Euclidean norm for the loss function $\mathcal{L}$ leads to

$$\begin{aligned}\mathcal{L}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}), \hat{\boldsymbol{y}}_{k|k}\big) &= \|\hat{\boldsymbol{y}}_{k|k} - f(\boldsymbol{x}_k;\boldsymbol{\phi})\|^2 \\ &= \left\| \begin{bmatrix} \hat{p}^{(1)}_{k|k} - f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi}) \\ \hat{p}^{(2)}_{k|k} - f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi}) \end{bmatrix} \right\|^2 \\ &= \big(\hat{p}^{(1)}_{k|k} - f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2 + \big(\hat{p}^{(2)}_{k|k} - f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2, \end{aligned} \tag{5.14}$$

where $f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})$ and $f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})$ denote the DNN prediction's first and second component respectively.

5.3.2. Semi-Soft Pseudo-Labels

For semi-soft pseudo-labeling as introduced in Section 4.4.2, we use the point estimates $\hat{\boldsymbol{y}}_{k|k}$ and the inverse of a transformed posterior covariance matrix as a weighting matrix in the modified loss $\mathcal{L}_\text{s}$, see Equation (4.32). The matrix $\boldsymbol{C}^{(\mathcal{Y})}_{k|k}$ involved in $\mathcal{L}_\text{s}$ is obtained from the posterior covariance matrix $\boldsymbol{C}_{k|k}$ by

$$\boldsymbol{C}^{(\mathcal{Y})}_{k|k} = \boldsymbol{Y}(\boldsymbol{C}_{k|k}) = \boldsymbol{H}\boldsymbol{C}_{k|k}\boldsymbol{H}^T \tag{5.15}$$

which results in extracting the upper left matrix of size 2×2. The modified loss function is then

$$\begin{aligned}\mathcal{L}_\text{s}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}), \hat{\boldsymbol{y}}_{k|k}\big) &= \big(f(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_{k|k}\big)^T \big(\boldsymbol{C}^{(\mathcal{Y})}_{k|k}\big)^{-1} \big(f(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{\boldsymbol{y}}_{k|k}\big) \\ &= \big[f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{p}^{(1)}_{k|k},\ f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{p}^{(2)}_{k|k}\big]^T \big(\boldsymbol{H}\boldsymbol{C}_{k|k}\boldsymbol{H}^T\big)^{-1} \begin{bmatrix} f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{p}^{(1)}_{k|k} \\ f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi}) - \hat{p}^{(2)}_{k|k} \end{bmatrix}. \end{aligned} \tag{5.16}$$

5.3.3. Soft Pseudo-Labels

For soft pseudo-labeling, as introduced in Section 4.4.3, we choose the squared Euclidean norm as the cost function $C(\,\cdot\,)$ in Equation (4.34). Thus, we have to calculate the expected cost of

$$\|\boldsymbol{y}(\boldsymbol{\theta}_k) - f(\boldsymbol{x}_k;\boldsymbol{\phi})\|^2 = (p_k^{(1)} - f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi}))^2 + (p_k^{(2)} - f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi}))^2, \tag{5.17}$$

over the posterior distribution $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})$. This results in the loss function

$$\mathcal{L}_\mathrm{s}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big) = \mathcal{E}_{p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})}\big\{\|\boldsymbol{y}(\boldsymbol{\theta}_k) - f(\boldsymbol{x}_k;\boldsymbol{\phi}))\|^2\big\}. \tag{5.18}$$

We obtain

$$\begin{aligned}
\mathcal{L}_\mathrm{s}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big) &= \mathcal{E}_{p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})}\Big\{\big(\theta_k^{(1)} - f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2 + \big(\theta_k^{(2)} - f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2\Big\} \\
&= \int_\Theta \big(\theta_k^{(1)} - f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2 p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k + \int_\Theta \big(\theta_k^{(2)} - f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2 p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k \\
&= \int_\Theta \big(\theta_k^{(1)}\big)^2 p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k - 2f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})\int_\Theta \theta_k^{(1)} p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k \\
&\quad + \big(f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2\int_\Theta p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k + \int \big(\theta_k^{(2)}\big)^2 p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k \\
&\quad - 2f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})\int_\Theta \theta_k^{(2)} p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k + \big(f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2\int_\Theta p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k \\
&= \int_\Theta \big(\theta_k^{(1)}\big)^2 p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k - 2f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})\hat{\boldsymbol{\theta}}_{k|k}^{(1)} + \big(f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2 \\
&\quad + \int_\Theta \big(\theta_k^{(2)}\big)^2 p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k - 2f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})\hat{\boldsymbol{\theta}}_{k|k}^{(2)} + \big(f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})\big)^2,
\end{aligned} \tag{5.19}$$

with

$$\hat{\boldsymbol{\theta}}_{k|k}^{(1)} = \int_\Theta \theta_k^{(1)} p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k \quad \text{and} \quad \hat{\boldsymbol{\theta}}_{k|k}^{(2)} = \int_\Theta \theta_k^{(2)} p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})d\boldsymbol{\theta}_k. \tag{5.20}$$

The loss function $\mathcal{L}_\mathrm{s}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big)$ forms part of the empirical risk $\mathcal{R}_\mathrm{new}(\boldsymbol{\phi})$, according to (4.35). During retraining of the DNN, we minimize $\mathcal{R}_\mathrm{new}(\boldsymbol{\phi})$, which involves calculating the gradient with respect to the parameter vector $\boldsymbol{\phi}$. Therefore, in Equation (5.19), we can omit terms that do not depend on $\boldsymbol{\phi}$, yielding the following contribution to the overall loss

$$\mathcal{L}_{\mathrm{s}}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big) = f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})(f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi}) - 2\hat{\boldsymbol{\theta}}^{(1)}_{k|k}) + f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})(f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi}) - 2\hat{\boldsymbol{\theta}}^{(2)}_{k|k}) + \mathrm{c}, \tag{5.21}$$

where c is a constant (not depending on $\boldsymbol{\phi}$). We note that $\hat{\boldsymbol{\theta}}^{(1)}_{k|k}$ and $\hat{\boldsymbol{\theta}}^{(2)}_{k|k}$ are provided by the KF. Using (5.21) the gradient of $\mathcal{L}_{\mathrm{s}}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big)$ with respect to $\boldsymbol{\phi}$ is obtained as

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\mathrm{s}}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big) = \nabla_{\boldsymbol{\phi}}\big(f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})(f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi}) - 2\hat{\boldsymbol{\theta}}^{(1)}_{k|k})\big) + \nabla_{\boldsymbol{\phi}}\big(f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})(f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi}) - 2\hat{\boldsymbol{\theta}}^{(2)}_{k|k})\big). \tag{5.22}$$

Thus, calculating $\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\mathrm{s}}\big(f(\boldsymbol{x}_k;\boldsymbol{\phi}),\boldsymbol{y}(\boldsymbol{\theta}_k)\big)$ reduces to calculating $\nabla_{\boldsymbol{\phi}}f^{(1)}(\boldsymbol{x}_k;\boldsymbol{\phi})$ and $\nabla_{\boldsymbol{\phi}}f^{(2)}(\boldsymbol{x}_k;\boldsymbol{\phi})$.

5.4. Kalman Filter for Position and Velocity Estimation

In this section, we present two heuristics for a modified KF. These modifications allow us to partly compensate the simplified measurement model given by Equation (5.8), estimate the unknown matrix $\boldsymbol{C}_{w,k}$, and detect outliers among the sequence of position predictions obtained by the DNN. We use the state-transition model as given by Equation (5.4). The driving noise covariance matrix $\boldsymbol{C}_u$ is assumed to be known whereas the measurement covariance matrix $\boldsymbol{C}_{w,k}$ is unknown.

Mean and Covariance Estimation

We recall the measurement noise in (5.10) was assumed to be a zero mean Gaussian distribution. If the DNN introduces a systematic offset, this assumption is not satisfied anymore. Therefore we have to estimate the average deviation of the DNN prediction $\boldsymbol{y}_k$ compared to the true position $\boldsymbol{p}_k$. For a sequence of images $\boldsymbol{x}_k$ and DNN outputs $\boldsymbol{y}_k$ for $k = 1, \dots, K$ with known parameter values $\boldsymbol{\theta}_k$ we can solve Equation (5.8) for $\boldsymbol{w}_k$

$$\boldsymbol{w}_k = \boldsymbol{y}_k - \boldsymbol{H}\boldsymbol{\theta}_k, \tag{5.23}$$

and calculate the mean as

$$\hat{\boldsymbol{\mu}}_w = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{w}_k. \tag{5.24}$$

The sample covariance matrix can be calculated as

$$\hat{\boldsymbol{C}}_w = \frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{w}_k - \hat{\boldsymbol{\mu}}_w)(\boldsymbol{w}_k - \hat{\boldsymbol{\mu}}_w)^T. \tag{5.25}$$

For sequences, with unknown state vectors $\boldsymbol{\theta}_k$ we can use $\hat{\boldsymbol{\mu}}_w$ to create corrected measurements

$$\boldsymbol{y}_{k,\mathrm{corr}} = f(\boldsymbol{x}_k; \boldsymbol{\phi}) - \hat{\boldsymbol{\mu}}_w = \boldsymbol{y}_k - \hat{\boldsymbol{\mu}}_w. \tag{5.26}$$

Outlier Detection

We can use the predicted posterior mean $\hat{\boldsymbol{\theta}}_{k|k-1}$ to evaluate the accuracy of the DNN output $\boldsymbol{y}_k$. This can be achieved by computing the predicted error

$$\hat{\boldsymbol{e}}_{k|k-1} = \left|\boldsymbol{y}_{k,\mathrm{corr}} - \boldsymbol{H}\hat{\boldsymbol{\theta}}_{k|k-1}\right|, \tag{5.27}$$

where $|\cdot|$ is applied component-wise and therefore yielding

$$\hat{\boldsymbol{e}}_{k|k-1} = \begin{bmatrix} \hat{e}^{(1)}_{k|k-1} \\ \hat{e}^{(2)}_{k|k-1} \end{bmatrix} = \begin{bmatrix} \left|\boldsymbol{y}^{(1)}_{k,\mathrm{corr}} - \hat{p}^{(1)}_{k|k-1}\right| \\ \left|\boldsymbol{y}^{(2)}_{k,\mathrm{corr}} - \hat{p}^{(2)}_{k|k-1}\right| \end{bmatrix}. \tag{5.28}$$

If the predicted error component $\hat{e}^{(i)}_{k|k-1}$ exceeds a certain threshold this means, that with a high probability, the DNN output $\boldsymbol{y}_{k,\mathrm{corr}}$ has low accuracy corresponding to a large noise variance along that component. Therefore we introduce a function, that allows to adjust the measurement noise variances over time, i.e.,

$$\left[\sigma^{(i)}_{w,k}\right]_{\mathrm{new}} = \begin{cases} C, & \hat{e}^{(i)}_{k|k-1} \geq \sigma_{\mathrm{threshold}} \\ \sigma^{(i)}_{w,k}, & \text{else.} \end{cases} \tag{5.29}$$

The time dependent estimate of the measurement covariance matrix $\hat{\boldsymbol{C}}_{w,k}$ is given by

$$\hat{\boldsymbol{C}}_{w,k} \triangleq \begin{bmatrix} \left[\sigma^{(1)}_{w,k}\right]^2_{\mathrm{new}} & 0 \\ 0 & \left[\sigma^{(2)}_{w,k}\right]^2_{\mathrm{new}} \end{bmatrix}. \tag{5.30}$$

The modified KF equations are given by:

1. **KF Prediction step:**
   a) Predicted posterior mean: $\hat{\boldsymbol{\theta}}_{k|k-1} = \boldsymbol{A}\hat{\boldsymbol{\theta}}_{k-1|k-1}$
   b) Predicted posterior covariance: $\boldsymbol{C}_{k|k-1} = \boldsymbol{A}\boldsymbol{C}_{k-1|k-1}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{C}_u\boldsymbol{B}^T$
2. **KF Update step:**
   a) Kalman gain matrix: $\boldsymbol{K}_k = \boldsymbol{C}_{k|k-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{C}_{k|k-1}\boldsymbol{H}^T + \hat{\boldsymbol{C}}_{w,k})^{-1}$
   b) Posterior mean: $\hat{\boldsymbol{\theta}}_{k|k} = \hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_{k,\text{corr}} - \boldsymbol{H}\hat{\boldsymbol{\theta}}_{k|k-1})$
   c) Posterior covariance: $\boldsymbol{C}_{k|k} = (\mathbf{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{C}_{k|k-1}$

6. Simulations

In this chapter, we evaluate the performance of DNN aided-sequential Bayesian estimation for a single target tracking scenario including a comparison between hard, semi-soft, and soft pseudo-labeling. The target or object of interest is a circle moving across an image. Such image data is the input to a DNN, which performs the feature extraction task in our case object localization, where the DNN output is a position prediction of the circle's center. We compare the performance between five different training setups. First, we have a DNN and a KF (see Figure 3.1). Here the DNN is trained using only labeled training data. Second, a DNN where pseudo-labels are directly created by the DNN. Additionally, we have a DNN and a KF where the DNN is trained using the pseudo-labeling schemes shown in Figures 3.3, 3.4, and 3.6. In Section 6.1, we show that incorporating statistical information into the training process can lead to DNNs with improved performance. In Section 6.2, we present some tracking results over time using different driving noise values. In Section 6.3, we address the question of tracking robustness, i.e., a comparison between DNN and KF position estimates under different image variations.

6.1. DNN Localization Performance Evaluation

In this section, we evaluate the accuracy of the DNN by calculating the average component error

$$e_{\mathrm{DNN}}^{(i)} = \frac{1}{L}\sum_{l=1}^{L}\left|f^{(i)}(\boldsymbol{x}_l;\boldsymbol{\phi}) - p_l^{(i)}\right|, \quad i \in \{1,2\}, \tag{6.1}$$

and the average Euclidean distance

$$e_{\mathrm{euc}} = \frac{1}{L}\sum_{l=1}^{L}\sqrt{\left|f^{(1)}(\boldsymbol{x}_l;\boldsymbol{\phi}) - p_l^{(1)}\right|^2 + \left|f^{(2)}(\boldsymbol{x}_l;\boldsymbol{\phi}) - p_l^{(2)}\right|^2}, \tag{6.2}$$

between the center of the object and the prediction of the DNN for a sample size of L. In Equation (6.1) and (6.2) $p_l^{(i)}$ denotes the circle's center position in the i-th component of the l-th sample and $\boldsymbol{x}_l$ being the l-th image. The input image is of size 128×128 pixels.

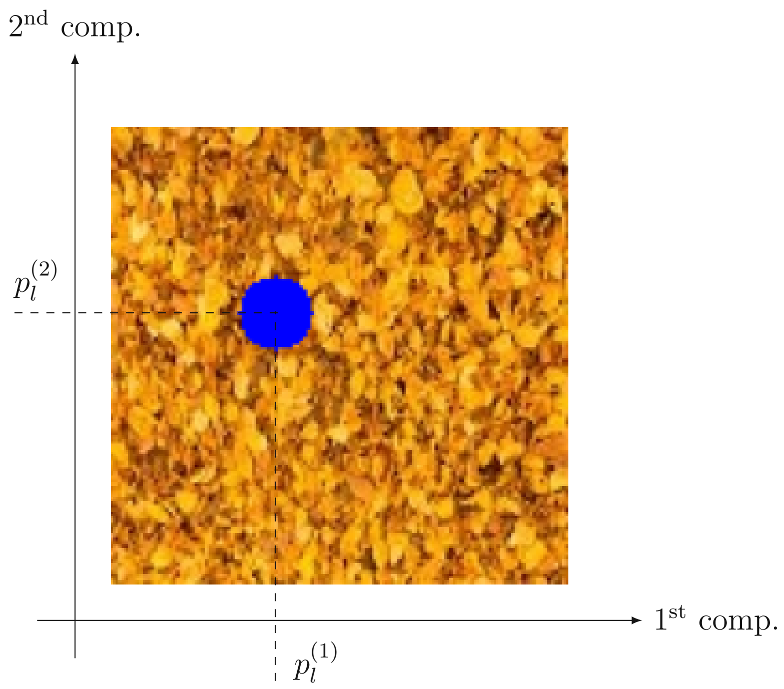


Figure 6.1.: Synthetic sample image. The object of interest is the blue circle with coordinates of the center being $\boldsymbol{p}_l = \left[p_l^{(1)},\ p_l^{(2)}\right]^T$.

The background image without the target was taken from [36]. An example image that is used at the input of the DNN is shown in Figure 6.1.

For the minimization of the empirical risk $\mathcal{R}(\boldsymbol{\phi})$, we use a modification of SGD [37]. According to [38], smaller batch sizes lead to better generalizing DNNs. Testing different values shows that a batch size of eight leads to a good DNN performance for our target tracking scenario. The number of epochs is 20. Each DNN is retrained three times and the DNN with the best-performing weights is used for the comparison. The DNN architecture used in this chapter is described in Section 5.2.

Evaluation Procedure

As a baseline for comparison with our novel pseudo-labeling techniques, we consider a conventional pseudo-labeling approach where pseudo-labels are created directly by using the output of the DNN. For this setup, we will use the name DNN labeling, which is depicted in Figure 6.2.

To evaluate the effect of the different pseudo-labeling schemes shown in Figure 3.3, 3.4, 3.6, and 6.2 on the performance of the DNN-based position predictor we use the following procedure.

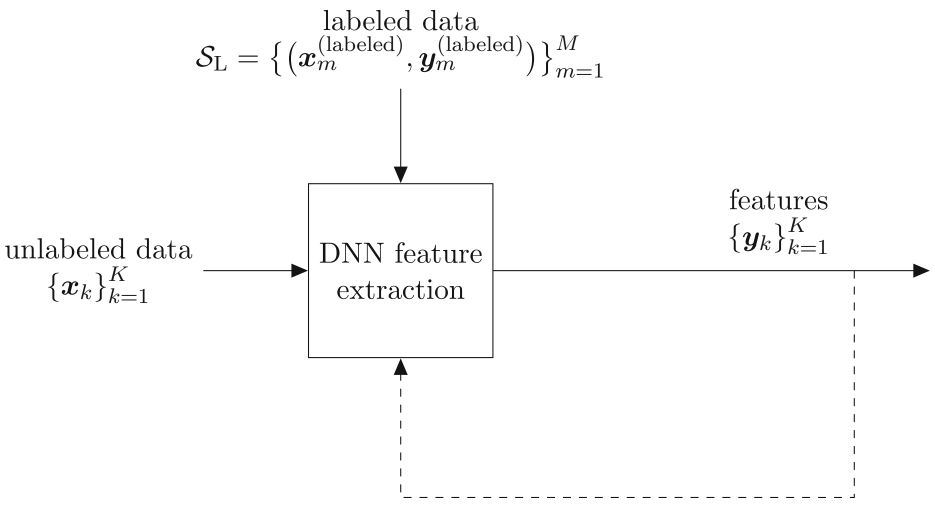

labeled data
$\mathcal{S}_\mathrm{L} = \left\{\left(\boldsymbol{x}_m^{(\mathrm{labeled})}, \boldsymbol{y}_m^{(\mathrm{labeled})}\right)\right\}_{m=1}^{M}$
unlabeled data
$\{\boldsymbol{x}_k\}_{k=1}^{K}$
DNN feature extraction
features
$\{\boldsymbol{y}_k\}_{k=1}^{K}$

Figure 6.2.: Block diagram of DNN labeling.

1. We train the DNN-based feature extractor using labeled training data. For our simulations, we use a labeled training data set $\mathcal{S}_\mathrm{L}$ consisting of $M = 1000$ images. In the labeled training data, the object is uniformly distributed over the image.

2. We create an unlabeled training sequence according to the state-transition model given in Equation (5.4) with driving noise $\left(\sigma_u^{(i)}\right)^2 = 10^{-4}$. This sequence consists of $K = 2000$ images and is used to create the set of pseudo-labels $\mathcal{S}_\mathrm{PL}$.

3. We create pseudo-labels according to Figure 3.3, 3.4, 3.6, and 6.2. We note that the DNN position predictions are mean corrected as introduced in Section 5.4.

4. We retrain the DNN using labeled training data from step 1 and pseudo-labels from step 3.

5. We evaluate the performance of the five different DNNs: (1) trained with only labeled data, (2) trained with pseudo-labels obtained from the DNN, (3) trained with hard pseudo-labels, (4) trained with semi-soft pseudo-labels, and (5) trained with soft pseudo-labels. This evaluation is performed using $L = 10000$ images as test data, where the object is uniformly distributed over the image. We calculate the average component error $e_\mathrm{DNN}^{(i)}$ and the average Euclidean distance e_euc between the center of the target and the output of the DNN, see Equation (6.1) and (6.2).

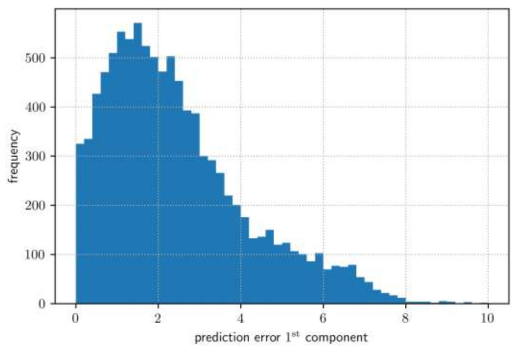
	reduced training	DNN labeling	hard pseudo-labeling	semi-soft pseudo-labeling	soft pseudo-labeling
$e_{\text{DNN}}^{(1)}$	2.47	2.39	2.48	**1.53**	1.56
$e_{\text{DNN}}^{(2)}$	2.83	2.15	1.62	**1.57**	1.96
e_{euc}	4.32	3.35	3.19	**2.34**	2.84

Table 6.1.: Table comparing the error of the same DNN architecture, but using different training data. The test sample size is $L = 10000$.

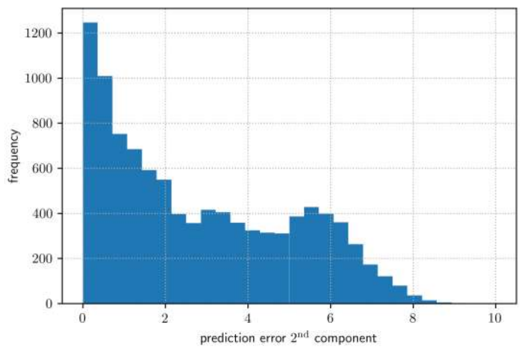
Performance comparison

In Table 6.1 the average component error $e_{\text{DNN}}^{(i)}$ and the average Euclidean distance e_{euc} are shown for all five possibilities of training the DNN described in this text. The column with the name "reduced training" corresponds to training the DNN with only labeled data and no pseudo-labels. For the average error in the 1st component $e_{\text{DNN}}^{(1)}$, we see that retraining with pseudo-labels created by the DNN increases the prediction accuracy of the first component. We also observe that hard pseudo-labeling does not lead to better prediction performance but incorporating the statistical information gives higher accuracy, as semi-soft and soft pseudo-labeling performs best. For the average error in the 2nd component $e_{\text{DNN}}^{(2)}$, retraining the DNN with hard pseudo-labels leads to higher prediction accuracy compared to the DNN trained with reduced training and pseudo-labels created by the DNN. Again, as for $e_{\text{DNN}}^{(1)}$, incorporating statistical information into the training results in reducing the average component error, but we see that hard pseudo-labeling has a lower average component error than soft pseudo-labeling. When the average Euclidean distance is compared, we see that semi-soft pseudo-labeling gives the best result, followed by soft and hard pseudo-labeling. Moreover, hard pseudo-labeling performs better than DNN labeling and reduced training. In summary, we see that higher prediction accuracy is achieved by feature correction performed by a sequential Bayesian estimator and that incorporating statistical information in the training gives an additional performance gain.

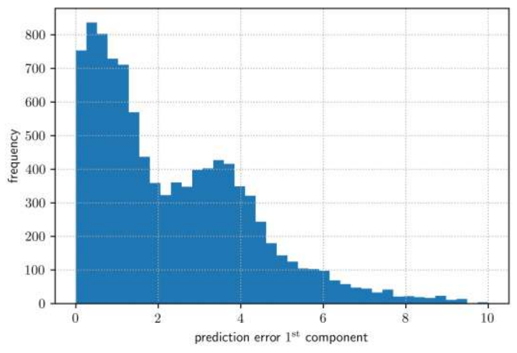
In Figure 6.3 and 6.4, error histograms evaluated for 10000 images of test data are given. For Figure 6.3, we do not use a Bayesian estimator for feature correction, i.e., we have the DNN with reduced training and DNN labeling. In Figure 6.4, we see the histograms for hard, semi-soft, and soft pseudo-labeling. When we compare Figures 6.3a and 6.3c, we recognize that retraining with DNN labels gives us higher frequency in low error events.



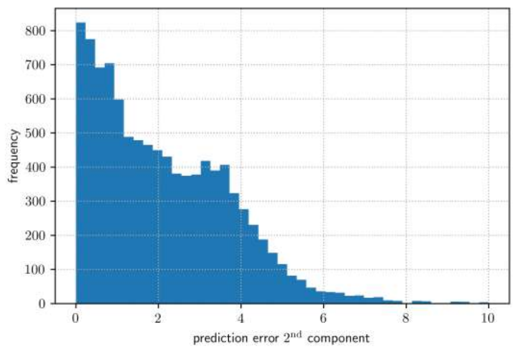
(a) 1st component error histogram (reduced training)



(b) 2nd component error histogram (reduced training)



(c) 1st component error histogram (DNN labeling)

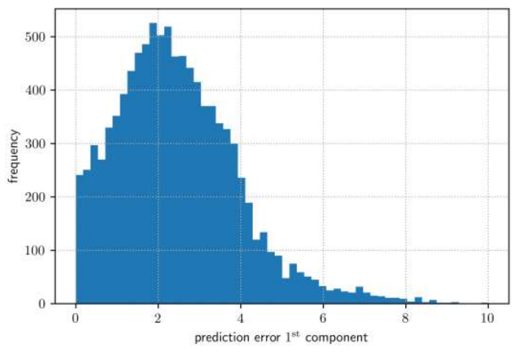


(d) 2nd component error histogram (DNN labeling)

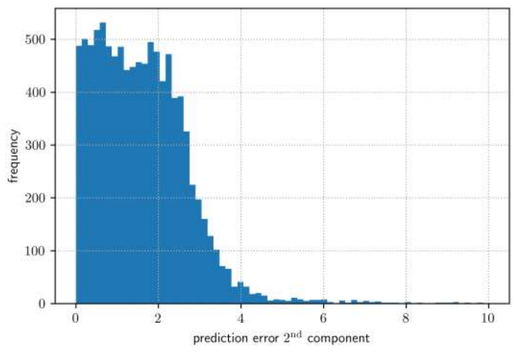
Figure 6.3.: Error histograms for DNN with reduced training (upper row) and using DNN labels (lower row)

However, the tails behave similarly. When comparing Figures 6.3b and 6.3d, we observe that Figure 6.3d has a shorter tail than the histogram given in Figure 6.3b, meaning fewer occurrences of high error events but also fewer occurrences of low error events.

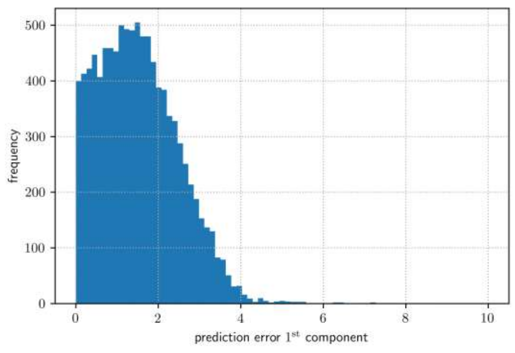
Next, we compare the histograms of the DNN outputs for hard, semi-soft, and soft pseudo-labeling. Starting with the 1st component average error, Figure 6.3a, 6.4c and 6.4e, we see that for soft pseudo-labeling we have the most occurrences of close-to-zero error events, whereas semi-soft pseudo-labeling gives the shortest tail. For the error along the 2nd component, Figure 6.4b, 6.4d and 6.4f, the amount of low-error events is similar for all three cases but we see that semi-soft pseudo labeling gives the strongest decay in regard to higher-error events.



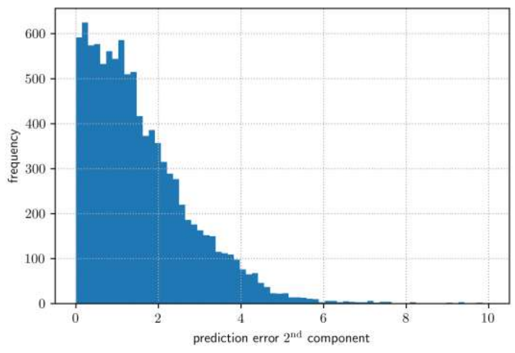
(a) 1st component error histogram (hard pseudo-labeling)



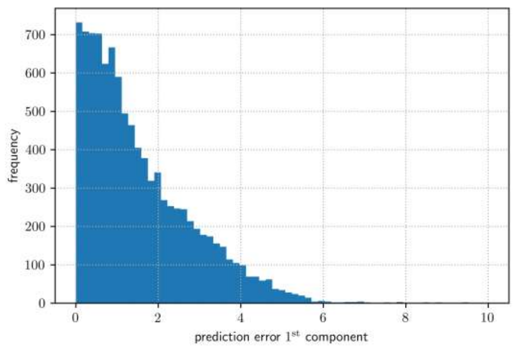
(b) 2nd component error histogram (hard pseudo-labeling)



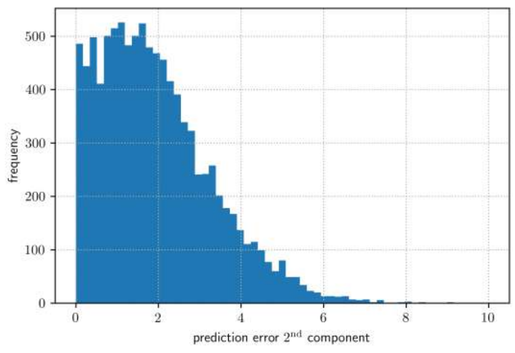
(c) 1st component error histogram (semi-soft pseudo labeling)



(d) 2nd component error histogram (semi-soft pseudo labeling)

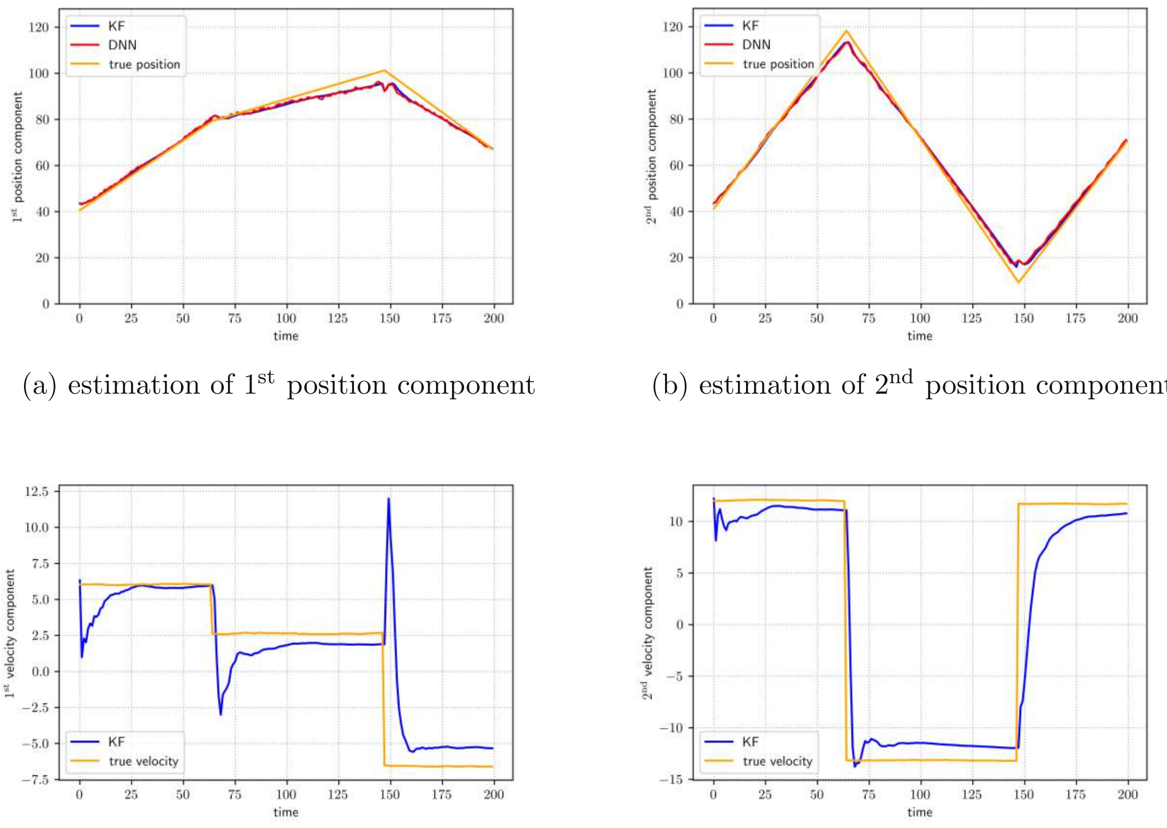


(e) 1st component error histogram (soft pseudo-labeling)



(f) 2nd component error histogram (soft pseudo-labeling)

Figure 6.4.: Error histograms for DNN with hard pseudo-labeling (upper row), semi-soft pseudo-labeling (middle row) and soft pseudo-labeling (lower row)



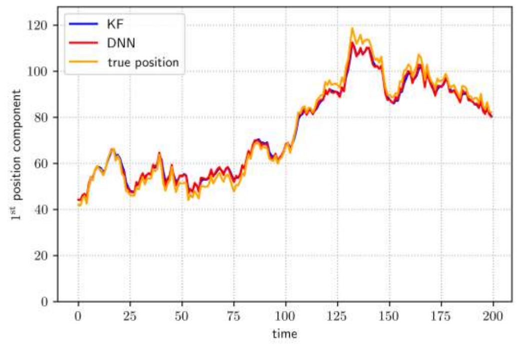
(a) estimation of 1st position component (b) estimation of 2nd position component

(c) estimation of 1st velocity component (d) estimation of 2nd velocity component

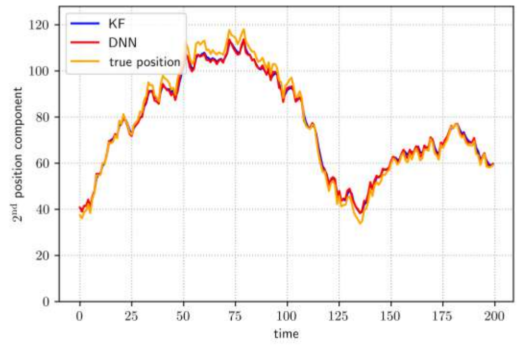
Figure 6.5.: Tracking a sequence of 200 images using the DNN with reduced training. For the driving noise variances we use small values: $\left(\sigma_u^{(i)}\right)^2 = 10^{-4}$, $i \in \{1, \dots, 4\}$. For the position components, we compare the DNN outputs, KF estimates, and the true position. For the velocity we only consider the true values and the KF estimates.

6.2. Tracking Performance

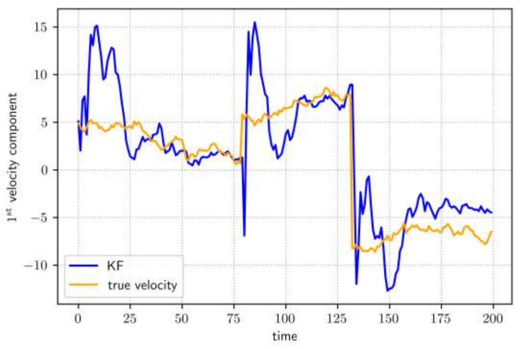
In this section, we present and discuss simulation results for a single target scenario. The target maneuvers according to a modified variant of the state-transition model given in Equation (5.4). Whenever the object reaches the image edge, a new random direction will be assigned. We will consider different values for the driving noise covariance matrix entries $\left(\sigma_u^{(i)}\right)^2$ and compare position predictions from the DNN, position estimates from the KF, and the true position of the object in the test data. The DNN position predictions are mean corrected according to Section 5.4. Furthermore, we discuss velocity estimation of the KF compared to the true velocity. In order to see the feature correction of the KF, we will first track sequences of images using the DNN with reduced training. To check if



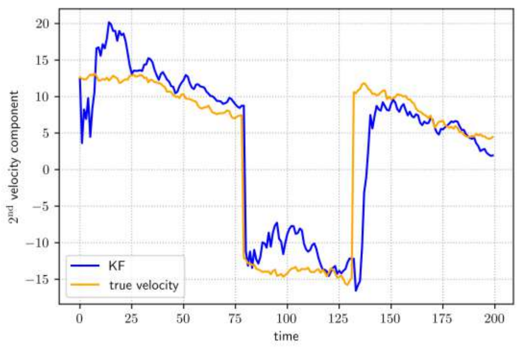
(a) estimation of 1st position component



(b) estimation of 2nd position component



(c) estimation of 1st velocity component

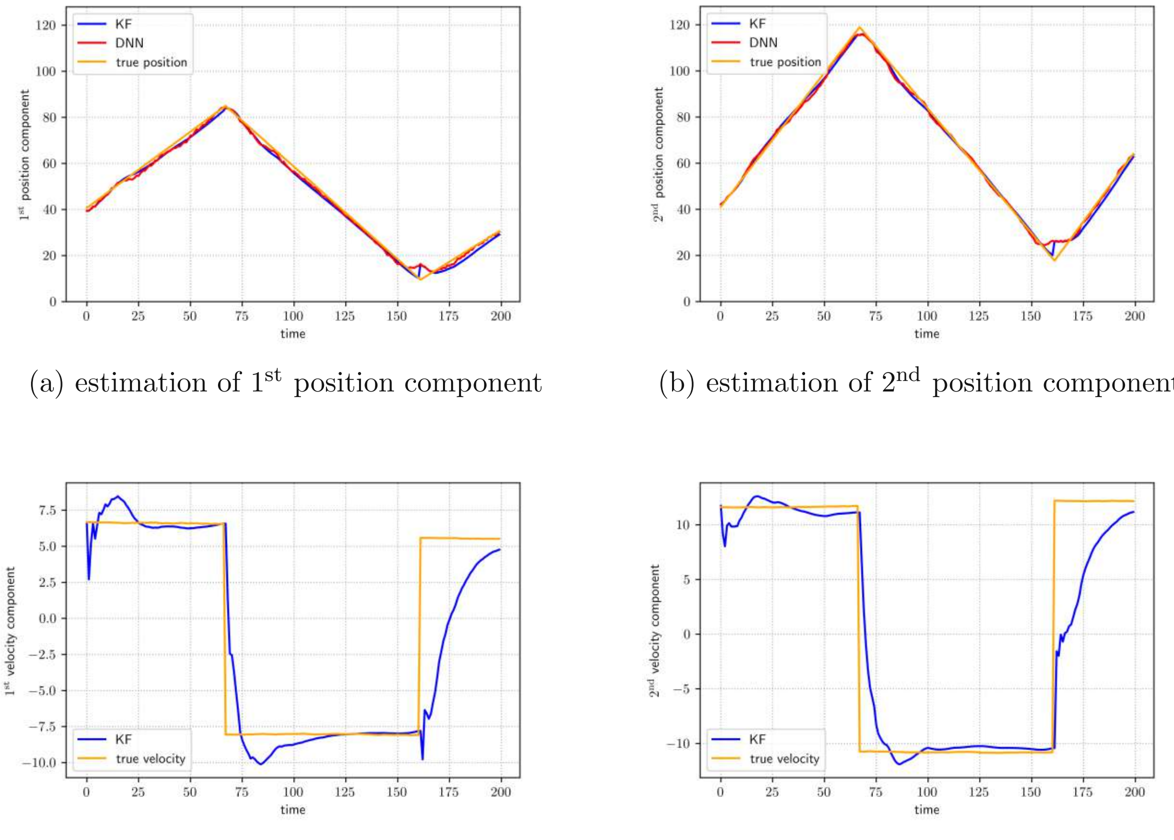


(d) estimation of 2nd velocity component

Figure 6.6.: Tracking a sequence of 200 images using the DNN with reduced training. For the driving noise variances we use large values: $\left(\sigma_u^{(1,2)}\right)^2 = 10$ and $\left(\sigma_u^{(3,4)}\right)^2 = 0.1$.

the KF can improve the DNN position prediction for retrained DNNs, we also consider sequences where the DNN trained with soft pseudo-labels is used.

In Figure 6.5, we see position and velocity estimates over time for a sequence of $K = 200$ images. Here, we use the DNN with reduced training. The variance of the driving noise for each component is $\left(\sigma_u^{(i)}\right)^2 = 10^{-4}$. For the majority of time steps, we see a good consensus among the true position, the corrected DNN predictions, and KF estimates. However, the DNN accuracy is lower when the object moves towards the boundaries of the image. For the velocity curves, we only compare the KF estimates and the true velocity, since the DNN architecture described in Section 5.2 is only able to provide position predictions. We see an almost piecewise constant function since the driving noise value is low. Positive velocities correspond to moving in the positive direction of the respective coordinate, while a negative velocity means moving in the opposite direction of that coordinate.



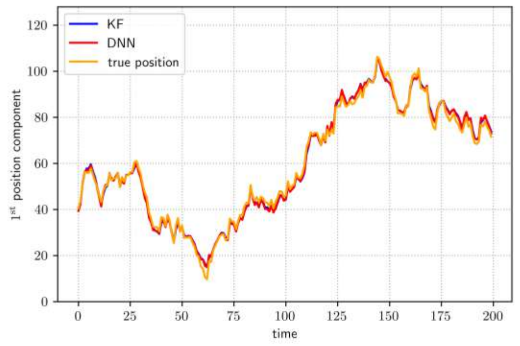
(a) estimation of 1st position component (b) estimation of 2nd position component

(c) estimation of 1st velocity component (d) estimation of 2nd velocity component

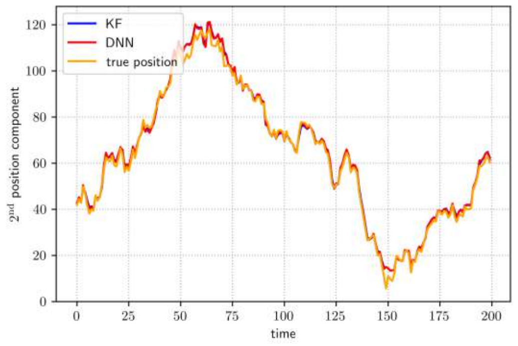
Figure 6.7.: Tracking a sequence of 200 images using the DNN trained with soft pseudo-labels. For the driving noise variances we choose small values: $\left(\sigma_u^{(i)}\right)^2 = 10^{-4},\ i \in \{1, \ldots, 4\}$.

The posterior covariance of the KF $\boldsymbol{C}_{k|k}$ needs to be reinitialized, when the object hits the boundary, otherwise the KF would not be able to accurately estimate the position and velocity. Whenever the velocity vector is changed, we see a transient behavior of the KF estimates in the velocity components and a sharp bend in the position components.

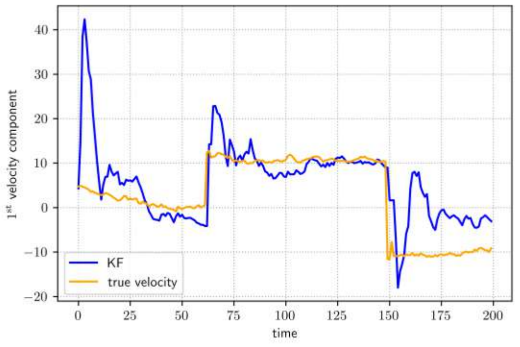
In Figure 6.6, we again consider a sequence of 200 images created according to the modified state-transition model in Equation (5.4). As before, we use the DNN with reduced training. However, this time we choose higher values for the driving noise, specifically $\left(\sigma_u^{(1,2)}\right)^2 = 10$ and $\left(\sigma_u^{(3,4)}\right)^2 = 0.1$. The motivation for using different values between position and velocity components of the driving noise comes from the direction changes when the object hits the image boundary. For even higher driving noise values we were not able to control the object's trajectory such that it stays within the image boundaries.



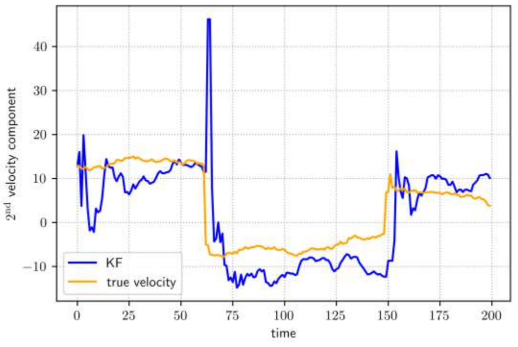
(a) estimation of 1st position component



(b) estimation of 2nd position component



(c) estimation of 1st velocity component



(d) estimation of 2nd velocity component

Figure 6.8.: Tracking a sequence of 200 images using a DNN trained with soft pseudo-labels. For the driving noise variances we choose large values: $\left(\sigma_u^{(1,2)}\right)^2 = 10$ and $\left(\sigma_u^{(3,4)}\right)^2 = 0.1$.

Due to the high variance in the position predictions, a very clear trajectory as given in Figure 6.5 is not visible anymore. For the velocity predictions, we also see that the true velocity is not piecewise constant anymore and that the error of the KF is higher compared to Figure 6.5.

Figure 6.7 shows tracking results using a DNN that was trained using soft pseudo-labeling. The driving noise variance is $\left(\sigma_u^{(i)}\right)^2 = 10^{-4}$. We see a strong consensus among the true position, the corrected DNN predictions, and KF estimates. Also, the KF velocity estimates show high accuracy after the transients. Finally, Figure 6.8 provides the position and velocity comparison for a DNN trained with soft pseudo-labeling with driving noise values chosen as $\left(\sigma_u^{(1,2)}\right)^2 = 10$ and $\left(\sigma_u^{(3,4)}\right)^2 = 0.1$.

	reduced training/ small $\sigma_u^{(i)}$	reduced training/ large $\sigma_u^{(i)}$	soft pseudo-labeling/ small $\sigma_u^{(i)}$	soft pseudo-labeling/ large $\sigma_u^{(i)}$
$e_{\text{DNN,corr}}^{(1)}/e_{\text{KF}}^{(1)}$	2.22/2.14	2.28/2.43	1.40/1.77	1.45/1.53
$e_{\text{DNN,corr}}^{(2)}/e_{\text{KF}}^{(2)}$	2.28/2.14	2.23/2.29	1.29/1.41	1.45/1.55

Table 6.2.: Table comparing the DNN and KF error for sequences of 200 images.

For the sequences depicted in Figures 6.5, 6.6, 6.7, and 6.8 we calculate the average component error between the corrected DNN output and the true position, i.e.,

$$e_{\text{DNN,corr}}^{(i)} = \frac{1}{K}\sum_{k=1}^{K} \left|\boldsymbol{y}_{k,\text{corr}}^{(i)} - p_k^{(i)}\right|, \quad i \in \{1,2\}, \tag{6.3}$$

where $\boldsymbol{y}_{k,\text{corr}}^{(i)}$ is the i-th component of the corrected feature vector according to Equation 5.26. Additionally, we calculate the average component error between the KF position estimate and the true position

$$e_{\text{KF}}^{(i)} = \frac{1}{K}\sum_{k=1}^{K} \left|\hat{\boldsymbol{\theta}}_{k|k}^{(i)} - p_k^{(i)}\right|, \quad i \in \{1,2\}, \tag{6.4}$$

with the posterior mean $\hat{\boldsymbol{\theta}}_{k|k}$ which can be obtained by the KF update step 2b) of the modified KF, see Section 5.4. The error results are depicted in Table 6.2. We see that for the DNN with reduced training and low values of the driving noise variance $\left(\sigma_u^{(i)}\right)^2 = 10^{-4}$, the KF is able to correct the DNN position predictions, as $e_{\text{KF}}^{(i)}$ is smaller than $e_{\text{DNN,corr}}^{(i)}$. However, for the DNN with reduced training and high values of $\left(\sigma_u^{(i)}\right)^2$ the KF feature correction fails. Regardless of the driving noise values $\left(\sigma_u^{(i)}\right)^2$, for the DNN using soft pseudo-labeling, the KF is not able to provide more accurate position estimates $\hat{\boldsymbol{\theta}}_{k|k}^{(1,2)}$ than the corrected DNN predictions $\boldsymbol{y}_{k,\text{corr}}^{(1,2)}$.

6.3. Robustness Analysis

A further question we would like to address is if the KF can introduce robustness to the object localization. In particular, we will consider three aspects. First, changing the shape of the object in a way such that it was not present during training. Second, varying the

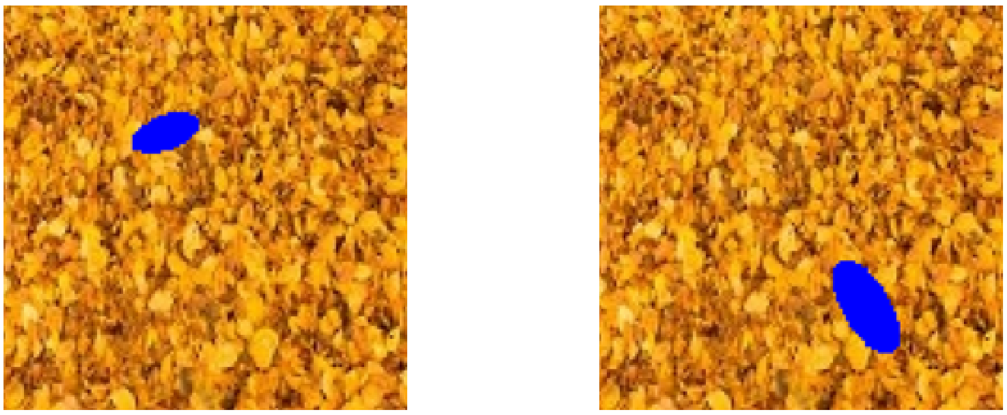


Figure 6.9.: Two sample images containing ellipses with different sizes and orientations.

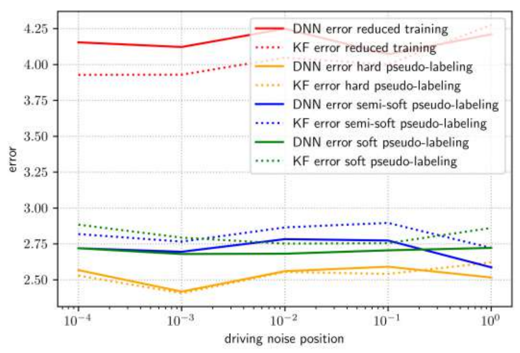
image noise variance. Finally, we will test the outlier detection introduced in Section 5.4

6.3.1. Driving Noise and Object Shape Variation

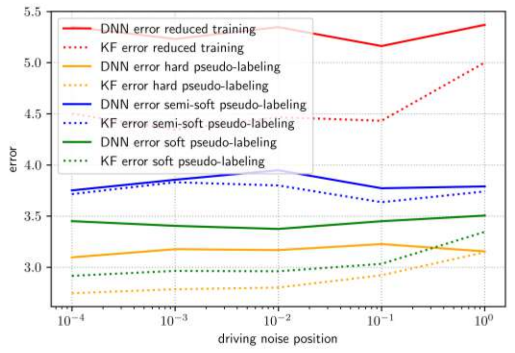
In this section, we will study the effects on tracking accuracy by changing the driving noise matrix and varying the object shape over time. For this, we calculate the sum of the average DNN component errors $e^{(i)}_{\text{DNN,corr}}$ and average KF component errors $e^{(i)}_{\text{KF}}$ for a sequence of 10000 images. Again, we use the motion model provided in Equation (5.4).

In Figure 6.10a, we show the effect of changing the position components of the driving noise on the tracking performance. In order to only study the influence of the position noise, the velocity components of the driving noise are set to a small value of $\left(\sigma_u^{(3,4)}\right)^2 = 10^{-4}$. The object shape for this figure is the same as given in the training process. First of all, we see that the DNN with the reduced amount of training data, i.e., solid red curve, gives the lowest accuracy. The feature correction by the KF gives better tracking results, which is represented by the dashed red line lying below the solid red one. Retraining the DNN using pseudo-labels and statistical information gives a large performance gain. For DNNs trained with hard, semi-soft, and soft pseudo-labels the DNN error is much smaller compared to the model with reduced training. However, the KF in this case is not able to improve the performance further by doing feature correction, as the dashed lines lie above the solid ones. Moreover, for tracking objects according to the motion model in Equation (5.4), we note that the mean correction, as given in Equation (5.26), seems to work best for hard pseudo-labeling as it gives the best overall performance.

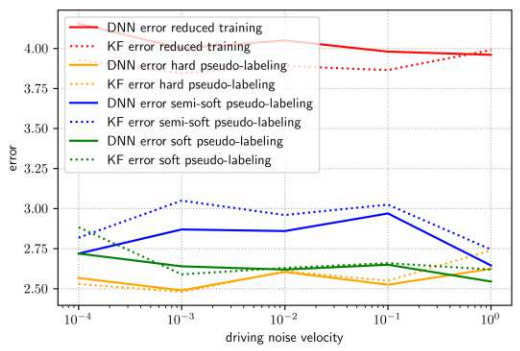
Figure 6.10b shows tracking results for again varying the position components of the driving noise, but additionally, the shape of the object varies over time. In the labeled training data, the object shape is a blue circle with a fixed radius. Now, we track a



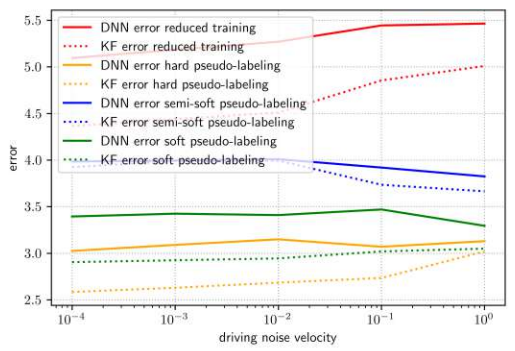
(a) Varying position components of driving noise without object shape changes



(b) Varying position components of driving noise with object shape changes



(c) Varying velocity components of driving noise without object shape changes

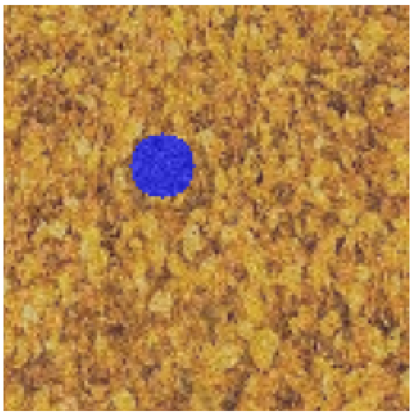


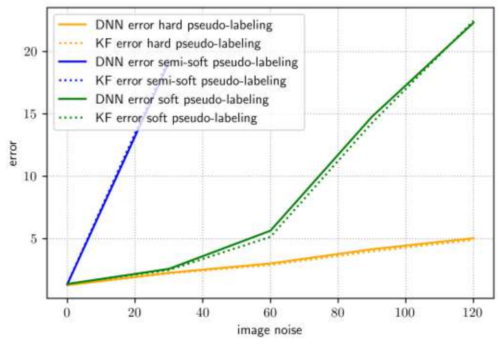
(d) Varying velocity components of driving noise with object shape changes

Figure 6.10.: Sum of average position errors for DNN and KF, while varying driving noise components of position and velocity. In Subfigures 6.10b and 6.10d the object shape varies over time.

sequence of a rotating ellipse with changing lengths of the principal axes over time as well. Two example images are depicted in Figure 6.9. First of all, we note that the overall error gets higher as for all curves we have larger error values. For the DNN with reduced training, the gap between the KF and the DNN gets wider, meaning the KF can give a larger improvement. In the previously described case (see Figure 6.10a) the KF could not improve the tracking performance of the retrained models. This time we see that for hard, semi-soft, and soft pseudo-labeling, the dashed lines lie below the solid lines, meaning KF feature correction gives additional performance gain. We note that for tracking objects with shapes different from those presented to the DNN during training, the KF can improve tracking performance and therefore makes the tracking system more robust.

When changing the velocity components of the driving noise, while position components


(a) Example image with noise variance value of 20.



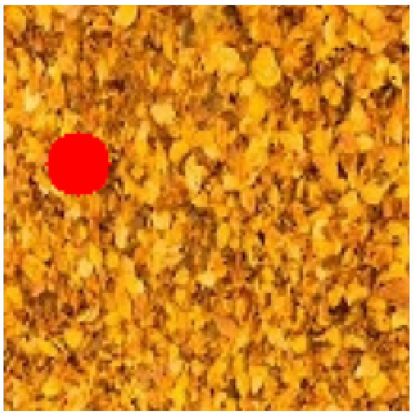
(b) Error plot for varying image noise variance.

Figure 6.11.: Varying image noise variance for hard, semi-soft, and soft-pseudo labeling. We compare corrected DNN position predictions and KF position estimates.

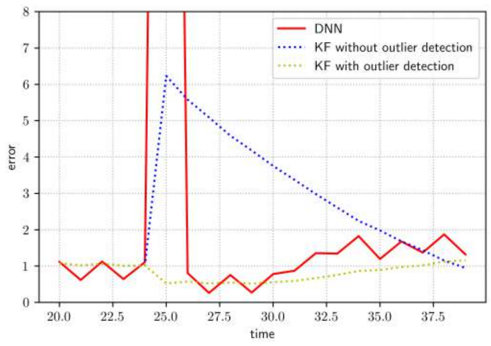
remain at $\left(\sigma_u^{(i)}\right)^2 = 10^{-4}$, we can make similar observations. In Figure 6.10c, we see again that retraining the DNN gives a large increase in performance. But still, hard pseudo-labeling seems to perform best for tracking. Figure 6.10d confirms that for tracking sequences with object shape variations over time, which are not present in the training data, the KF feature correction gives better results than corrected DNN-based position prediction on its own.

6.3.2. Image Noise

In the next step of doing a robustness analysis of DNN-aided sequential Bayesian estimation for the single target tracking problem, we will add Gaussian noise to each test image and track again a sequence of 10000 images. As before, the motion model in Equation (5.4), modified so that the target does not leave the boundaries of the image, is used. The sum of the average DNN component errors $e_{\text{DNN,corr}}^{(i)}$ and the sum of the average KF component errors $e_{\text{KF}}^{(i)}$ for different image noise variances is depicted in Figure 6.11b. We see that the DNN trained with hard pseudo-labels gives the most robust results. For soft pseudo-labeling, the DNN does not give reliable position predictions for image noise variances exceeding values of 60. Semi-soft pseudo-labeling gives the least accurate predictions. For hard and soft pseudo-labeling the KF gives a very small increase in accuracy compared to the corrected DNN predictions. However, this improvement is negligible.



(a) 25th image with red circle as the target.



(b) Error plot over time.

Figure 6.12.: Solid red curves are the DNN prediction with an outlier at time step 25. The blue dashed line depicts the KF error without the outlier detection, which shows a strong transient behavior. The yellow dashed line is the KF error with outlier detection. The transient behavior is strongly suppressed.

6.3.3. Outlier Detection

In order to test the outlier detection proposed in Section 5.4, the color of the object is changed to a color that was not present in the training data. This is done so at the 25th image in the sequence, see Figure 6.12a. The error plot is given in Figure 6.12b. At time step 25, we have an error value of 50.2. Until this time step, the curves of the KF with and without the outlier detection overlap perfectly. The KF without outlier detection treats every DNN prediction in the same way, which leads to the transient behavior seen by the blue line in Figure 6.12b. We see that for a time interval of over 10 time steps, the KF error for the KF without outlier detection is higher than the DNN error. The KF with outlier detection changes the value of the measurement covariance matrix $\hat{\boldsymbol{C}}_{w,k}$ according to Equation (5.29). For the KF with outlier detection, we only have slightly larger errors than the DNN directly after the outlier and we do not observe the transient behavior. Thus, the KF with outlier detection improves the robustness of the tracking system.

7. Conclusion

Summary of Main Results

In this work, we developed an estimation framework that combines DNN feature extraction with Bayesian estimation. Our basic approach is to consider the output of the DNN as measurements for a non-sequential or sequential Bayesian estimator. We proposed three different methods for retraining the DNN using pseudo-labels generated by the Bayesian estimator. In the hard pseudo-labeling method, the Bayesian estimator calculates corrected features. In the semi-soft pseudo-labeling method, statistical information provided by the Bayesian estimator is incorporated into DNN training via the posterior covariance matrix. In the soft pseudo-labeling method, DNN training involves a posterior expectation.

As an application, we considered a simplified target tracking scenario where the task was to estimate the position and velocity of a circle moving across an image. In our simulations, the DNN was a simplified version of an architecture proposed in [11], and the sequential Bayesian estimator was a KF. We found that the KF can improve the DNN position predictions. Therefore, the KF produces more accurate pseudo-labels than the DNN, resulting in an improved feature extraction performance of the DNN when retrained using the pseudo-labels.

Next, by considering a more challenging tracking scenario we demonstrated that the proposed DNN-KF combination increases the robustness of the target tracking relative to DNN position predictions. This was shown by tracking sequences where the shape of the object varies over time in a way that is not represented in the training data. Finally, we showed that the use of the KF allows the detection of outliers.

Future Work

The work presented in this thesis can be continued and extended in several directions, including the following:

- The linear measurement model used in this work may be too simplistic for more realistic tracking scenarios or different DNN architectures. Therefore, more advanced estimators that are not limited to linear-Gaussian models, such as the particle filter, are of interest.

- In the current form, the DNN is retrained using labeled data and pseudo-labels. For more sophisticated DNN architectures, which usually require much more training data than the architectures considered in this work, it would be desirable to retrain the DNN using only pseudo-labels and initialize the weights of the DNN with those obtained from the initial training process using labeled data. However, retraining the DNN by using only pseudo-labels that are created from sequences where the object shape differs from that in the labeled training will lead to the problem of catastrophic forgetting, i.e., the DNN will perform poorly on test data that is similar to the labeled data. One method to tackle this problem is presented in [39], where a regularizing term is added to the loss function.

- An extension of our method to the tracking of multiple targets would be of practical interest. The task of the DNN would then be to predict the number of targets in addition to their locations, and the subsequent Bayesian estimation stage would track the states of all targets.

- For a more conclusive assessment of the proposed DNN-aided Bayesian estimation framework a performance evaluation using real data rather than synthetic data would be desirable.

- Developing applications of the proposed framework to other estimation problems besides target tracking would be interesting.

- Finally, a theoretical analysis of the serial setup can be expected to yield valuable insights. In particular, it would be interesting to investigate the influence of the DNN-based feature extraction stage on the error of the Bayesian estimator. This can possibly be achieved by considering approximation bounds such as those presented in [40–42].

A. Derivation of Kalman Filter

In the following, we derive the KF equations stated in Section 4.3. We will use [27,28,34]. We recall from 4.2 that sequential Bayesian filtering consists of the following two steps:

1. PREDICTION STEP: We want to calculate the predicted posterior distribution $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})$ according to

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}) = \int_{\Theta} p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1})p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1})d\boldsymbol{\theta}_{k-1}. \tag{A.1}$$

2. UPDATE STEP: We update the predicted posterior distribution to obtain the posterior

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})}, \tag{A.2}$$

where the denominator can be expanded as

$$p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) = \int_{\Theta} p(\boldsymbol{y}_k|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})d\boldsymbol{\theta}_k. \tag{A.3}$$

PREDICTION STEP

To calculate $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1})$ in Equation A.1, we need expressions of $p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1})$ and $p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1})$. Due to the state-transition (4.11), the Gaussianity of the driving noise according to Equation (4.15), and the independence of $\boldsymbol{u}_k$ and $\boldsymbol{\theta}_{k-1}$, the state-transition pdf $p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1})$ is given by

$$p(\boldsymbol{\theta}_k|\boldsymbol{\theta}_{k-1}) = \mathcal{N}(\boldsymbol{\theta}_k; \boldsymbol{A}_k\boldsymbol{\theta}_{k-1}, \boldsymbol{B}_k\boldsymbol{C}_{u,k}\boldsymbol{B}_k^T). \tag{A.4}$$

Later we will prove that the posterior pdf $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})$ is Gaussian under a Gaussian initialization. Therefore, we can assume that the posterior at time $k-1$, i.e. $p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1})$, is also Gaussian distributed according to

$$p(\boldsymbol{\theta}_{k-1}|\boldsymbol{y}_{1:k-1}) = \mathcal{N}(\boldsymbol{\theta}_{k-1}; \hat{\boldsymbol{\theta}}_{k-1|k-1}, \boldsymbol{C}_{k-1|k-1}), \tag{A.5}$$

with the posterior mean $\hat{\boldsymbol{\theta}}_{k-1|k-1}$ and posterior covariance matrix $\boldsymbol{C}_{k-1|k-1}$. We will need the following identity several times, [28, Appendix D],

$$\mathcal{N}(\boldsymbol{r};\boldsymbol{H}\boldsymbol{x},\boldsymbol{R})\mathcal{N}(\boldsymbol{x};\boldsymbol{p},\boldsymbol{P}) = \mathcal{N}(\boldsymbol{r};\boldsymbol{H}\boldsymbol{p},\boldsymbol{C})\mathcal{N}(\boldsymbol{x};\boldsymbol{e},\boldsymbol{E}) \tag{A.6}$$

with

$$\boldsymbol{E} = (\boldsymbol{P}^{-1} + \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H})^{-1}, \tag{A.7}$$
$$\boldsymbol{e} = \boldsymbol{E}(\boldsymbol{P}^{-1}\boldsymbol{p} + \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{r}), \tag{A.8}$$
$$\boldsymbol{C} = \boldsymbol{R} + \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^T. \tag{A.9}$$

Inserting Equation (A.4) and (A.5) into (A.1), we obtain

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}) = \int_\Theta \mathcal{N}(\boldsymbol{\theta}_k;\boldsymbol{A}_k\boldsymbol{\theta}_{k-1},\boldsymbol{B}_k\boldsymbol{C}_{u,k}\boldsymbol{B}_k^T)\mathcal{N}(\boldsymbol{\theta}_{k-1};\hat{\boldsymbol{\theta}}_{k-1|k-1},\boldsymbol{C}_{k-1|k-1})d\boldsymbol{\theta}_{k-1}. \tag{A.10}$$

Next, using the identity in (A.6), with $\boldsymbol{r} = \boldsymbol{\theta}_k$, $\boldsymbol{H} = \boldsymbol{A}_k$, $\boldsymbol{x} = \boldsymbol{\theta}_{k-1}$, $\boldsymbol{R} = \boldsymbol{B}_k\boldsymbol{C}_{u,k}\boldsymbol{B}_k^T$, $\boldsymbol{p} = \hat{\boldsymbol{\theta}}_{k-1|k-1}$, and $\boldsymbol{P} = \boldsymbol{C}_{k-1|k-1}$, the predicted posterior is given by

$$\begin{aligned} p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}) &= \int_\Theta \mathcal{N}(\boldsymbol{\theta}_k;\boldsymbol{A}_k\hat{\boldsymbol{\theta}}_{k-1|k-1},\boldsymbol{B}_k\boldsymbol{C}_{u,k}\boldsymbol{B}_k^T + \boldsymbol{A}_k\boldsymbol{C}_{k-1|k-1}\boldsymbol{A}_k^T)\mathcal{N}(\boldsymbol{\theta}_{k-1};\boldsymbol{e}_{\theta_{k-1}},\boldsymbol{E}_{\theta_{k-1}})d\boldsymbol{\theta}_{k-1} \\ &= \mathcal{N}(\boldsymbol{\theta}_k;\boldsymbol{A}_k\hat{\boldsymbol{\theta}}_{k-1|k-1},\boldsymbol{B}_k\boldsymbol{C}_{u,k}\boldsymbol{B}_k^T + \boldsymbol{A}_k\boldsymbol{C}_{k-1|k-1}\boldsymbol{A}_k^T)\int_\Theta \mathcal{N}(\boldsymbol{\theta}_{k-1};\boldsymbol{e}_{\theta_{k-1}},\boldsymbol{E}_{\theta_{k-1}})d\boldsymbol{\theta}_{k-1} \\ &= \mathcal{N}(\boldsymbol{\theta}_k;\boldsymbol{A}_k\hat{\boldsymbol{\theta}}_{k-1|k-1},\boldsymbol{B}_k\boldsymbol{C}_{u,k}\boldsymbol{B}_k^T + \boldsymbol{A}_k\boldsymbol{C}_{k-1|k-1}\boldsymbol{A}_k^T), \end{aligned} \tag{A.11}$$

with a mean vector $\boldsymbol{e}_{\theta_{k-1}}$ and a covariance matrix $\boldsymbol{E}_{\theta_{k-1}}$ which do not depend on $\boldsymbol{\theta}_{k-1}$, i.e.,

$$\boldsymbol{e}_{\theta_{k-1}} = \boldsymbol{E}_{\theta_{k-1}}\big(\boldsymbol{C}_{k-1|k-1}^{-1}\hat{\boldsymbol{\theta}}_{k-1|k-1} + \boldsymbol{A}_k^T(\boldsymbol{B}_k\boldsymbol{C}_{u,k}\boldsymbol{B}_k^T)^{-1}\boldsymbol{\theta}_k\big), \tag{A.12}$$
$$\boldsymbol{E}_{\theta_{k-1}} = \big(\boldsymbol{C}_{k-1|k-1}^{-1} + \boldsymbol{A}_k^T(\boldsymbol{B}_k\boldsymbol{C}_{u,k}\boldsymbol{B}_k^T)^{-1}\boldsymbol{A}_k\big)^{-1}. \tag{A.13}$$

We can rewrite Equation (A.11) as

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}) = \mathcal{N}(\boldsymbol{\theta}_k;\hat{\boldsymbol{\theta}}_{k|k-1},\boldsymbol{C}_{k|k-1}), \tag{A.14}$$

with the predicted mean

$$\hat{\boldsymbol{\theta}}_{k|k-1} \triangleq \boldsymbol{A}_k\hat{\boldsymbol{\theta}}_{k-1|k-1}, \tag{A.15}$$

and the predicted covariance matrix

$$\boldsymbol{C}_{k|k-1} \triangleq \boldsymbol{B}_k \boldsymbol{C}_{u,k} \boldsymbol{B}_k^T + \boldsymbol{A}_k \boldsymbol{C}_{k-1|k-1} \boldsymbol{A}_k^T. \tag{A.16}$$

UPDATE STEP

Next, we calculate $p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k})$ according to Equation (A.2). The missing pdfs in (A.2) are the likelihood function $p(\boldsymbol{y}_k|\boldsymbol{\theta}_k)$ and the denominator $p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})$. Because of the measurement model in (4.12), the Gaussianity of the measurement noise according to (4.16), and the independence of $\boldsymbol{\theta}_k$ and $\boldsymbol{w}_k$, the likelihood function is given by

$$p(\boldsymbol{y}_k|\boldsymbol{\theta}_k) = \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}_k\boldsymbol{\theta}_k, \boldsymbol{C}_{w,k}). \tag{A.17}$$

Inserting (A.14) and (A.17) we can write the numerator of (A.2) as

$$p(\boldsymbol{y}_k|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}) = \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}_k\boldsymbol{\theta}_k, \boldsymbol{C}_{w,k})\mathcal{N}(\boldsymbol{\theta}_k; \hat{\boldsymbol{\theta}}_{k|k-1}, \boldsymbol{C}_{k|k-1}). \tag{A.18}$$

Applying the identity in Equation (A.6) with $\boldsymbol{r} = \boldsymbol{y}_k$, $\boldsymbol{H} = \boldsymbol{H}_k$, $\boldsymbol{x} = \boldsymbol{\theta}_k$, $\boldsymbol{R} = \boldsymbol{C}_{w,k}$, $\boldsymbol{p} = \hat{\boldsymbol{\theta}}_{k|k-1}$, and $\boldsymbol{P} = \boldsymbol{C}_{k|k-1}$ gives

$$p(\boldsymbol{y}_k|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k-1}) = \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}_k\hat{\boldsymbol{\theta}}_{k|k-1}, \boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)\mathcal{N}(\boldsymbol{\theta}_k; \hat{\boldsymbol{\theta}}_{k|k}, \boldsymbol{C}_{k|k}), \tag{A.19}$$

with

$$\hat{\boldsymbol{\theta}}_{k|k} = \boldsymbol{C}_{k|k}\big(\boldsymbol{C}_{k|k-1}^{-1}\hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1}\boldsymbol{y}_k\big), \tag{A.20}$$

$$\boldsymbol{C}_{k|k} = \big(\boldsymbol{C}_{k|k-1}^{-1} + \boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1}\boldsymbol{H}_k\big)^{-1}. \tag{A.21}$$

Next, we consider the denominator of (A.2) as given by (A.3). Inserting expression (A.19) into (A.3) gives

$$\begin{aligned} p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) &= \int_\Theta \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}_k\hat{\boldsymbol{\theta}}_{k|k-1}, \boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)\mathcal{N}(\boldsymbol{\theta}_k; \hat{\boldsymbol{\theta}}_{k|k}, \boldsymbol{C}_{k|k})d\boldsymbol{\theta}_k \\ &= \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}_k\hat{\boldsymbol{\theta}}_{k|k-1}, \boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)\int_\Theta \mathcal{N}(\boldsymbol{\theta}_k; \hat{\boldsymbol{\theta}}_{k|k}, \boldsymbol{C}_{k|k})d\boldsymbol{\theta}_k \\ &= \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}_k\hat{\boldsymbol{\theta}}_{k|k-1}, \boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T). \end{aligned} \tag{A.22}$$

Inserting the result for the numerator in (A.19) and the result for the denominator in (A.22) into (A.2) gives the Gaussian posterior pdf

$$p(\boldsymbol{\theta}_k|\boldsymbol{y}_{1:k}) = \frac{\mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}_k\hat{\boldsymbol{\theta}}_{k|k-1}, \boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)\mathcal{N}(\boldsymbol{\theta}_k; \hat{\boldsymbol{\theta}}_{k|k}, \boldsymbol{C}_{k|k})}{\mathcal{N}(\boldsymbol{y}_k; \boldsymbol{H}_k\hat{\boldsymbol{\theta}}_{k|k-1}, \boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)} = \mathcal{N}(\boldsymbol{\theta}_k; \hat{\boldsymbol{\theta}}_{k|k}, \boldsymbol{C}_{k|k}). \tag{A.23}$$

The expressions (A.20) for the posterior mean $\hat{\boldsymbol{\theta}}_{k|k}$ and (A.21) for the posterior covariance matrix $\boldsymbol{C}_{k|k}$ can be put in a different form. Indeed, an alternative expression for the posterior covariance matrix $\boldsymbol{C}_{k|k}$ can be obtained by applying the matrix inversion lemma [43],

$$(\boldsymbol{A}+\boldsymbol{U}\boldsymbol{B}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{B}^{-1}+\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1} \tag{A.24}$$

with $\boldsymbol{A} = \boldsymbol{C}_{k|k-1}^{-1}$, $\boldsymbol{U} = \boldsymbol{H}_k^T$, $\boldsymbol{B} = \boldsymbol{C}_{w,k}^{-1}$, and $\boldsymbol{V} = \boldsymbol{H}_k$ to Equation (A.21), which gives

$$\begin{aligned}
\boldsymbol{C}_{k|k} &= \boldsymbol{C}_{k|k-1} - \boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)^{-1}\boldsymbol{H}_k\boldsymbol{C}_{k|k-1} \\
&= \big(\mathbf{I} - \boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)^{-1}\boldsymbol{H}_k\big)\boldsymbol{C}_{k|k-1} \\
&= (\mathbf{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_{k|k-1},
\end{aligned} \tag{A.25}$$

with the Kalman gain matrix

$$\boldsymbol{K}_k \triangleq \boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)^{-1}. \tag{A.26}$$

Next, inserting expression (A.25) for $\boldsymbol{C}_{k|k}$ into Equation (A.20), the posterior mean $\hat{\boldsymbol{\theta}}_{k|k}$ can be written as

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{k|k} &= (\mathbf{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_{k|k-1}(\boldsymbol{C}_{k|k-1}^{-1}\hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1}\boldsymbol{y}_k) \\
&= \hat{\boldsymbol{\theta}}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{H}_k\hat{\boldsymbol{\theta}}_{k|k-1} + (\mathbf{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1}\boldsymbol{y}_k.
\end{aligned} \tag{A.27}$$

The last term of Equation (A.27) can be rewritten as

$$\begin{aligned}
&(\mathbf{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1}\boldsymbol{y}_k \\
&= (\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1} - \boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1})\boldsymbol{y}_k \\
&= \big(\underbrace{\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)^{-1}}_{\boldsymbol{K}_k}(\boldsymbol{C}_{w,k} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T)\boldsymbol{C}_{w,k}^{-1} \\
&\quad - \boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1}\big)\boldsymbol{y}_k \\
&= \big(\boldsymbol{K}_k(\mathbf{I} + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1}) - \boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{C}_{w,k}^{-1}\big)\boldsymbol{y}_k
\end{aligned}$$

$$= \boldsymbol{K}_k \boldsymbol{y}_k. \tag{A.28}$$

Inserting the last expression back into (A.27) finally gives

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{k|k} &= \hat{\boldsymbol{\theta}}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{H}_k \hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{K}_k \boldsymbol{y}_k \\ &= \hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k \hat{\boldsymbol{\theta}}_{k|k-1}). \end{aligned} \tag{A.29}$$

In the prediction step, we made the assumption that the posterior pdf at time step $k-1$ follows a Gaussian distribution. This will be satisfied if the initial state $\boldsymbol{\theta}_0$ is Gaussian as we assumed in Equation (4.17). More specifically, we initialize the KF recursion with posterior mean and posterior covariance given by

$$\hat{\boldsymbol{\theta}}_{0|0} = \boldsymbol{\mu}_\theta \quad \text{and} \quad \boldsymbol{C}_{0|0} = \boldsymbol{C}_\theta, \tag{A.30}$$

where $\boldsymbol{\mu}_\theta$ and $\boldsymbol{C}_\theta$ are the mean and covariance matrix of the initial state $\boldsymbol{\theta}_0$ respectively.

We conclude by summarizing the KF equations.

1. **KF Prediction step:**
   a) Predicted posterior mean: $\hat{\boldsymbol{\theta}}_{k|k-1} = \boldsymbol{A}_k \hat{\boldsymbol{\theta}}_{k-1|k-1}$
   b) Predicted posterior covariance: $\boldsymbol{C}_{k|k-1} = \boldsymbol{A}_k \boldsymbol{C}_{k-1|k-1} \boldsymbol{A}_k^T + \boldsymbol{B}_k \boldsymbol{C}_{u,k} \boldsymbol{B}_k^T$

2. **KF Update step:**
   a) Kalman gain matrix: $\boldsymbol{K}_k = \boldsymbol{C}_{k|k-1} \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{C}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{C}_{w,k})^{-1}$
   b) Posterior mean: $\hat{\boldsymbol{\theta}}_{k|k} = \hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k \hat{\boldsymbol{\theta}}_{k|k-1})$
   c) Posterior covariance: $\boldsymbol{C}_{k|k} = (\mathbf{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{C}_{k|k-1}$

3. **Initialization:**

   $\hat{\boldsymbol{\theta}}_{0|0} = \boldsymbol{\mu}_\theta, \quad \boldsymbol{C}_{0|0} = \boldsymbol{C}_\theta$

Bibliography

[1] I. Neutelings, *Plot Fully Connected Neural Net.* `https://tikz.net/neural_networks/`.

[2] S. Särkkä and L. Svensson, *Bayesian Filtering and Smoothing.* Cambridge University Press, 2023.

[3] H. Iqbal, *Plot Neural Net.* `https://github.com/HarisIqbal88/PlotNeuralNet/`.

[4] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on bayesian neural networks—a tutorial for deep learning users," *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29–48, 2022.

[5] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, pp. 373–440, 2020.

[6] X. Yang, Z. Song, I. King, and Z. Xu, "A survey on deep semi-supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, pp. 8934–8954, sep 2023.

[7] R. Kunwar, U. Pal, and M. Blumenstein, "Semi-supervised online bayesian network learner for handwritten characters recognition," in *2014 22nd International Conference on Pattern Recognition*, pp. 3104–3109, 2014.

[8] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning: Key approaches and design guidelines," in *2021 IEEE Data Science and Learning Workshop (DSLW)*, pp. 1–6, 2021.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 00, pp. 580–587, June 2014.

[11] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations (ICLR 2015)*, pp. 1–14, Computational and Biological Learning Society, 2015.

[12] X. Weng, J. Wang, D. Held, and K. Kitani, “3d multi-object tracking: A baseline and new evaluation metrics,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2020.

[13] S. Shi, X. Wang, and H. Li, “Pointrcnn: 3d object proposal generation and detection from point cloud,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–779, 2019.

[14] F. Meyer and J. L. Williams, “Scalable detection and tracking of geometric extended objects,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 6283–6298, 2021.

[15] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, “Class-balanced grouping and sampling for point cloud 3d object detection,” 2019.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[17] C. Michoski, M. Milosavljević, T. Oliver, and D. R. Hatch, “Solving differential equations using deep neural networks,” *Neurocomputing*, vol. 399, pp. 193–212, 2020.

[18] J. Jumper, R. Evans, A. Pritzel, and et al., “Highly accurate protein structure prediction with alphafold,” *Nature*, vol. 596, pp. 583–589, 2021.

[19] P. Petersen, *Neural Network Theory*. University of Vienna, 2021.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[21] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From theory to algorithms*. Cambridge University Press, 2014.

[22] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen, *Mathematical Aspects of Deep Learning*. Cambridge University Press, 2022.

[23] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning, second edition*. Adaptive Computation and Machine Learning series, MIT Press, 2018.

[24] D.-H. Lee, “Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks,” *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 07 2013.

[25] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[26] D. Rumelhart, G. Hinton, and R. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[27] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume 1: Estimation Theory.* Pearson Education.

[28] R. P. Mahler, *Statistical Multisource-Multitarget Information Fusion*, vol. 685. Artech House, 2007.

[29] C. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[30] P. C. Mahalanobis, “On the generalised distance in statistics,” *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936. Retrieved 2016-09-27.

[31] R. De Maesschalck, D. Jouan-Rimbaud, and D. Massart, “The mahalanobis distance,” *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, 2000.

[32] J. Dorazil, *private conversation, February 2023.*

[33] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking,” *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[34] H. Masnadi-Shirazi, A. Masnadi-Shirazi, and M.-A. Dastgheib, “A step by step mathematical derivation and tutorial on kalman filters,” in *arXiv*, 2019.

[35] X. Rong Li and V. Jilkov, “Survey of maneuvering target tracking. part i. dynamic models,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[36] R. Ganz, *Object Localization using Keras.* `https://medium.com/analytics-vidhya/object-localization-using-keras-d78d6810d0be`.

[37] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[38] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” in *International Conference on Learning Representations*, 2017.

[39] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the National Academy of Sciences*, vol. 114, pp. 3521–3526, mar 2017.

[40] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[41] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989.

[42] D. Yarotsky, “Error bounds for approximations with deep relu networks,” *Neural Networks*, vol. 94, pp. 103–114, 2017.

[43] H. V. Henderson and S. R. Searle, “On deriving the inverse of a sum of matrices,” *SIAM Review*, vol. 23, pp. 53–60, Jan. 1981.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

....	....
Ort, Datum	Unterschrift